RCL Reference Manual

Ver. 4.02

RCL: R Command Language

NITTO DENKO

Barcode printers supporting DURA PRINTER Command
(described as “RCL (R Command Language) ” in this document)
are currently the following models from the DURA PRINTER series.

- ● DURA PRINTER R
- ● DURA PRINTER SR
- ● DURA PRINTER SRS
- ● DURA PRINTER LSP5300
- ● DURA PRINTER LP5320

PRINTED IN JAPAN　　７Ｃ５０８６１４

CONTENTS

1. Introduction

1.1. About RCL

RCL is an interpreter language for carrying out all of the functions of a bar code printer (referred to below simply as "printer"). Using RCL, bar code labels can be created to meet any requirements.
The combination of RCL with a printer can be used to print text (including kanji) and graphics, logos, bar codes (UPC, EAN, Code39, Interleaved 2 of 5, Code93, Code128, CodaBar), and two-dimensional symbols (QR Code, Code49, etc.).

1.2. About This Manual

This manual is intended for programmers using RCL for the first time, and describes in detail RCL command syntax. It will also serve as a reference book for users with previous RCL experience for checking on command syntax, and for finding more advanced functionality.

Note
This manual describes all of the commands in RCL, and therefore, depending on the model of printer used, these may not all be supported. The command support table in Chapter 25 shows which printers support which commands. Note that some commands may also not be supported depending on the ROM version installed.

1.2.1. Commands Always Required

Some commands are always required in order to print labels with RCL; others are optional for use as required. There are three commands which are always required: the header, SPB (Start Processing Buffer), and TRM (TeRMinate command); these may not be omitted. The sample label printing programs given in this manual are assumed always to have the header, SPB, and TRM commands appended.

1.2.2. Parameter Format

In RCL, all parameters are separated by semicolons. This applies whether the parameters are integers, decimal numbers, or ASCII character strings.
If the semicolon is omitted, the separation of parameters (data fields) is not recognized. The sample programs in this manual show parameters separated by semicolons as shown in the following example:

```
HBR;100;VBR;50;DBBX;10;25;600;300;
```

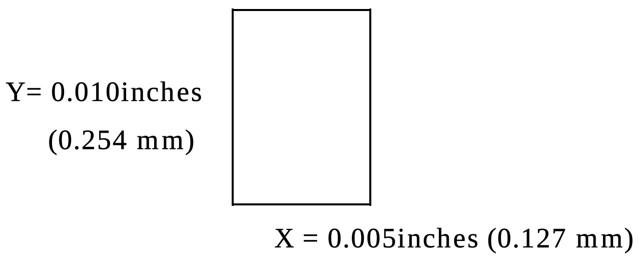
1.2.3. Command Notation

The commands in this manual are English mnemonics for the command names in capital letters (e.g. SPB, TRM, and so on). However, the language is not case-sensitive, and lowercase letters may also be used.
When command names are shown in full, capital letters are use for the mnemonic letters, thus: "TeRMinate". In this case the mnemonic is "TRM". In RCL, except for cases such as "HOME", the full command name is not accepted, and the mnemonic must be used. For example: not "TERMINATE", but "TRM".

1.3. Dimensional Units for the Printing Position (Pixels)

In RCL, the units for determining the printing position are referred to as "pixels". The minimum horizontal and vertical pixel dimensions are as follows:

Minimum horizontal pixel dimension: (X) = 0.005inches (0.127 mm)
Minimum vertical pixel dimension: (Y) = 0.010inches (0.254 mm)



For example, horizontally "10 pixels" refers to a 0.05”(1.27 mm) movement of the printing position. Vertically, "10 pixels" refers to a 0.1”(2.54 mm) movement of the printing position.

Note
These pixel dimensions are independent of the resolution of the printer. For example, whether the printer resolution is 200 dpi, 300 dpi or 400 dpi, the specifications for printing position are always in the units X = 0.005”(0.127 mm), Y = 0.010”(0.254 mm).

1.4. Command Format

RCL is designed as a high-level interpreted language. It includes some fifty commands for defining and printing bar code labels. Almost all of these commands require parameters, and some commands require particular data. There is some variation from command to command, but data values are specified in pixels, in decimal or hexadecimal notation. Inserting comments in the program will make it easier to read.

1.4.1. Command Names

The RCL command names are three- or four-letter mnemonics. In many cases the mnemonic is simply the initial letters of the full English form of the command; for example, SPB is an abbreviation of "Start Processing Buffer," and DVL is "Draw Vertical Line." In other cases two or more letters are taken from a single word. For example, MRK is from "MaRK" and TRM from "TeRMinate." There are then a few cases, such as HOME, LOGO, BOTH, and FLIP, where the mnemonic is simply the command name.

1.4.2. Parameters

Almost all commands use parameters to select a particular function, or specify dimensions. When a command has more than one parameter, they must be specified in the correct order, or an error will result. Depending on the command, parameters may be pixels unit, or decimal or hexadecimal values. The following table gives examples of parameter formats.

Table 1

Mnemonic	Meaning	Parameter	Units	Example
HBR	Horizontal Base Reference	X-position	X pixels	HBR;100;
VBR	Vertical Base Reference	Y-position	Y pixels	VBR;50;
BCLC	Bar Code Label Count	Number of copies	Decimal value	BCLC;5;
IDF	Increment/Decrement Field	Amount	Decimal value	IDF;-3;
DHR	Define Human Readable (vector font)	Font number	Hexadecimal or decimal	DHR;$8000; DHR;32768;
SPB	Start Processing Buffer	None	–	SPB;

1.4.3. Specifying Commands and Parameters

As shown by the examples in Table 1, command names and parameters are always delimited by semicolons. When a command requires parameters, the correct number of parameters must be present. For example, the Draw BOX command (DBOX) requires four parameters: the X start coordinate, Y start coordinate, width, and height. The order of command specification is also important: the header and processor must come at the beginning of the command sequence, and positioning commands must come before the objects they position. An object must also be defined before it can be positioned on a label. The sequence of commands is terminated by the TRM command.

1.4.4. Specifying Data Values

Data values may includes any alphanumeric characters (or kanji) from the printer's font. These data values, like command names are delimited by semicolons, and must be enclosed in double quotation marks ("). The following are examples of correctly specified data values.

```
DHR;1;"Label One";     BCST;"*25D9B01*";BSTP;
"1st Entry";           "1234567890";
```

1.4.5. Comments

Comments may be included in a program, enclosed in hash marks (#) and separated by semicolons. The following restrictions apply to comments:

・Comments may not come before the header.

・Comments may not be interspersed with the parameters to a command.

・Comments cannot be included in text strings.

・Comments cannot be included within the header or function-specifying commands.

The following are correct and incorrect examples.

```
Legals                              Illegal
~^"FILE";1;0;100;0;#Header#;        ~^"FILE";#Header#;1;0;50;0;
VBR;10;HBR;20;#Starting Point#;     VB#Starting Point#;R;10;HBR;20;
DHR;1;"Data Field One";#1#;         DHR;1;"Data Field #1# One";
```

The string enclosed by the hash marks is ignored, and not printed on the label. However, if the program is listed, using the line printer mode, the comments are also printed. (See Section 1.6.2.)

Note
Be careful not to omit the closing double quote or hash mark at the end of a data value or comment. Unless this terminator character is present, printing will not be carried out correctly.

1.5. Program Format

RCL is a structured programming language, and in some cases this means that before using a particular command it is necessary to use another command. Using a flowchart, the command structure and the relations among commands can be expressed graphically. The rest of the section discusses mandatory commands and the printing of serial numbers.

1.5.1. Mandatory Commands

As described above, in RCL, three commands are always required to print a label: the header, and the SPB and TRM commands. These correspond to the header, processor, and command sequence terminator. If any one of these three is missing, no printing will occur.

Some commands for controlling objects (items, such as bar codes or text strings, to be printed) must be used as a set. For example, to create a bar code at least three commands are required: a bar code selector (BSYM／BDEF), bar code start (BCST), and bar code stop (BSTP).
To print text in a vector font requires a minimum of four commands: define human readable (DHR), define character height (DCH), define character width (DCW), and inter-character space (ICS).
At least one pair of positioning commands is required for each label printed: horizontal base reference (HBR) and vertical base reference (VBR). Other commands can be added or removed as required for a particular application.

1.5.2. Printing Serial Numbers

When printing a number of labels, it is possible to include a serial number within either a bar code or a text field. For this purpose, a number of commands must be issued, including mark (MRK), return (RET), and bar code label count (BCLC). For a bar code, the bar code increment/decrement (BCID) and bar save address length (BSAL) commands must be defined. For text, the increment/decrement field (IDF) and save address length (SAL) commands must be defined. In addition to these commands, the draw white box (DWBX) must be used to clear a repeated printing area. For more details on serial numbers, see Chapter 11.

1.6. Using RCL

RCL is designed to be used with a computer system with a serial interface of Centronics parallel interface. This section discusses file management and the use of the line printer function.

1.6.1. File Creation, Saving, and Use

Program files are created on the host using a text editor. These files can be sent directly to the printer or saved on the host for subsequent download to the printer. All program files must be saved using the file management utilities on the host computer so that they can be used later.

1.6.2. Using the Line Printer Function

Either before or after printing labels with the printer, it is very convenient to have a printed copy of the command sequence. This can be done with a line printer or a serial printer. Sending the commend sequence to a correctly connected printer provides a listing of the command sequence. This will be useful when a hard copy is needed, or if no full-screen editor is available.

2.Header

An RCL program must start with a header. This defines the name, length, start position, number of labels to be printed, and other important information about the labels. If any of this information is omitted, other label definitions are not possible.

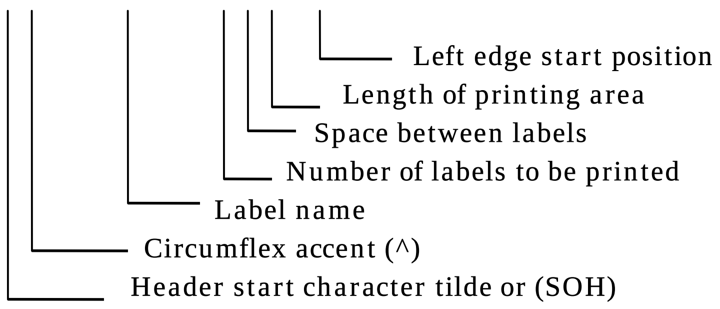
2.1.Syntax

The header consists of the following seven components. They are all mandatory.

1. Header start character (SOH)
2. Circumflex accent (^)
3. Label name (enclosed in " ")
4. Number of labels to be printed
5. Space between labels (not used)
6. Length of printing area (in Y pixel units)
7. Left edge start position (X pixel units)

Except for the header start character, these fields are the same as any other in RCL, and are delimited by semicolons, as shown in the following example.

```
˜^"Example1";3;0;400;50;
```



All fields must be defined, even if a field has a zero value.。

2.2.Header Start Character

For the header start character, either a tilde (˜=7E$_H$) or an SOH control code (01$_H$) may be used. It is not possible to use both together. The start character must be immediately followed by a circumflex accent (^=5E$_H$).

The header start character must be the first character of a program.

Note The 'H' in the notation "7E$_H$" H indicates that the value is in hexadecimal.

If the SOH or tilde (˜) character is preceded by an asterisk, the printer switches to extended memory communications mode.

If the SOH and tilde (˜) is preceded by a function setting command or an extended memory command such as DIR or LOAD, then the printed executes this command. For details see Section 12.1 and Chapter 14.

2.3.Label Name

The second field of the header is the label name. The label name may be any required length, and is always enclosed in double quotation marks (". . .").

Note
The label name must be enclosed in quotation marks. If the quotation marks are omitted, the command sequence will not be processed, and the printer will ignore this command sequence.

2.3.1.Label Name Format

The label name is used to identify the labels.
The label name is a sequence of ASCII characters, enclosed in double quotation marks.

The following are examples of correct label names.

"Label1" "AIAG20" "!@#$%^&* () _+" "test 2"

In the extended memory communications mode, the label name is used as the file name. When a specification is made to save the RCL command sequence to extended memory, the command sequence sent to the printer is written to extended memory as a text file.
Thereafter, in extended memory communications mode when a file is specified in extended memory with the * command, the data is found by comparing this label name with the file name specified in the * command.

The label name is treated as follows.

1) If the label name consists of nine or more characters, the first eight characters are used. Thus "NameOfFileA" is treated as "NameOfFi".
2) If a label name includes a space character (20_H) the file name comparison goes only as far as the space. Thus "File A" and "File" are treated as the same.

Note
When writing command sequences to extended memory, it is simplest to restrict label names to not more than eight characters.

The following example includes two sets of quotation marks, and is therefore an error.
""TEST3""

2.4.Number of Labels to be Printed

This is the total number of labels to be printed in a batch. The minimum is 1, and the maximum 16,777,215.

2.5.Space Between Labels

This function is not currently supported. Specify a zero value.

2.6.Length of Printing Area

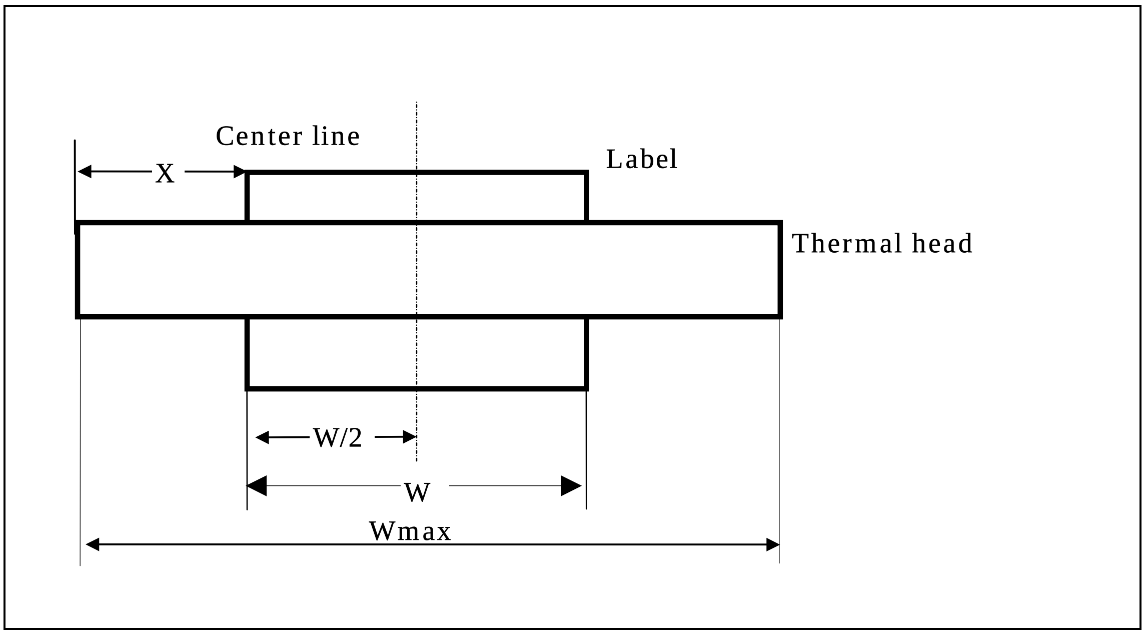
Specify the overall length in vertical pixel units (0.254 mm) of the print buffer in which the object is to be printed. This is not the physical length of the label, but the length of the printed area on the label. The origin is the top left corner. The length of the printing area varies from printer to printer: consult the specifications for the printer.

2.7.Left Edge Start Position

This parameter sets the left edge position where printing starts (i.e. the coordinate origin).
For some printer models labels are inserted aligned to the left edge, and for others the positioning is centered. For left-aligned printers, always set this value to zero.
For center-aligned printers, if the width of the labels used is less than the maximum printing width of the printer, use the following expression to obtain the setting value. If the width of the labels is more than the maximum printing width, set the value to zero.

X= ((Wmax - W) /2) / 0.005

X:	left edge start position
W:	label width (inch)
Wmax:	maximum printing width of printer (inch)



When the left edge start position is set, the horizontal base reference command (HBR) is based on this position. Then the horizontal position relative command (HPR) can be used for positions relative to the horizontal base reference.

Note
The label alignment and maximum printing width (Wmax) for each model of printer is shown is shown in the specifications in Chapter 18.

3. Program Control

It is necessary to indicate the beginning and end of a program sequence.
The start position is defined by the first character in the first line after the command header. The end position is the last command in the last line of the command sequence. Section 3.1 describes the following commands, one of which controls the program operation.

Start Processing Buffer (SPB)
ReStart Processing Buffer (RSPB)
TeRMinate Processing Buffer (TRM)
BReaK (BRK)

3.1. Buffer Processing

The processing of a program is controlled by the four commands: Start Processing Buffer (SPB), ReStart Processing Buffer (RSPB), TeRMinate processing buffer (TRM), and BReaK (BRK). Of these, SPB and TRM are required in all programs. RSPB can be used as required, in place of SPB when it is desired not to clear a bitmap image from a previous command sequence from the print buffer before beginning processing. BRK is used to break a long command sequence into a number of blocks.

3.1.1. Start Processing Buffer (SPB)

The SPB command clears the print buffer and starts processing of the command sequence. The commands and data following SPB are treated as the data for the labels already defined in the header. The SPB command immediately follows the header data, as shown in the following example:

```
~^"Example1";1;0;400;50;
SPB;
```

3.1.2. ReStart Processing Buffer (RSPB)

The RSPB command is similar to SPB, but does not clear the print buffer. RSPB can be used in place of SPB in the following two ways.
Firstly, since the buffer is not cleared, it can be used to resume processing without deleting the label image printed immediately previously. This increases processing efficiency.
Secondly, this command can be combined with the break command (see Section 3.1.4). SPB and RSPB are used for printing bar code and other label data as serial numbers.

3.1.3. TeRMinate Processing Buffer (TRM)

The TRM command indicates the end of a command sequence, and instructs the printer to process the labels and begin printing. When there is data in the printer buffer, the TRM command is used to print this data.
TRM must be the last command of a command sequence, and as shown in the following example must be followed by a semicolon and backslash.

```
~^"Example1";1;0;400;50;
SPB;
TRM;¥
```

The minimum requirement is a header, SPB command, and TRM command. If any one of these three is missing, the RCL processing will not start. Naturally, in the example shown above, since no bar codes, rectangles or other label objects are defined, nothing will be printed. However, issuing this command sequence will cause the printer to execute the command sequence, and move by the distance specified as the length of the printing area for the labels currently loaded.

3.1.4. BReaK (BRK)

RCL converts the mnemonics in a command sequence to hexadecimal codes, reducing the volume to about 1/3, and then first stores them in a control buffer memory. In the case of a long program, to prevent this memory capacity from being exceeded, the break command (BRK) must be used in place of the TRM command, to split the command sequence into a number of blocks. See Chapter 18 for the memory capacity, which is listed under the printer specifications.
The break command (BRK) instructs the printer to process the command sequence terminated by the break; by beginning the next block with an RSPB command, an overlay is possible. The printer then waits on standby through the following blocks, without printing anything, until a block is ended by a TRM command. The BRK command can be used as many times as is necessary. The BRK command must come in the position normally occupied by the TRM command (until the final TRM). Then except for the first block, the SPB command must be replaced by an RSPB command. Like the TRM command, the BRK command must be immediately followed by a backslash.

The following example prints a logo (see Chapter 13), and uses two command sequences, with BRK and RSPB commands.

```
~^"REGISTER";1;0;50;0;
SPB;
VBR;5;HBR;150;
LOGO;4;19;
0;$0F;$F8;0;
0;$70;$07;0;
1;$C0;1;$C0;
```

```
7;0;0;$70;
$0E;$0F;$F0;$38;
$1C;$0F;$F8;$1C;
$38;$0C;$18;$0E;
$30;$0C;$18;6;
$70;$0C;$38;7;
$70;$0F;$F0;7;
$70;$0F;$F0;7;
$30;$0C;$38;6;
$38;$0C;$18;$0E;
$1C;$0C;$18;$1C;
$0E;$0C;$18;$38;
$7;0;0;$70;
1;$C0;1;$C0;
0;$70;$07;0;
0;$0F;$F8;0;
BRK;¥

~^"COPYRIGHT";1;0;50;0;
RSPB;
VBR;5;HBR;200;
LOGO;4;20;
1;$FF;$FF;$C0;
3;$FF;$FF;$E0;
7;0;0;$70;
$0E;0;0;$38;
$1C;$0F;$F8;$1C;
$1C;$1F;$FC;$1C;
$1C;$38;$0E;$1C;
$1C;$70;$07;$1C;
$1C;$70;0;$1C;
$1C;$70;0;$1C;
$1C;$70;0;$1C;
$1C;$70;0;$1C;
$1C;$70;7;$1C;
$1C;$38;$0E;$1C;
$1C;$1F;$FC;$1C;
$1C;$0F;$F8;$1C;
```

```
$0E;0;0;$38;
7;0;0;$70;
3;$FF;$FF;$E0;
1;$FF;$FF;$C0;
TRM;¥
```

4.Print Position Movement Commands

There are absolute positioning and relative positioning commands for controlling the position of objects (printed bar codes and text) on a label. These positioning commands are necessary to ensure that the various types of object described in Chapters 5 to 8 of this manual are printed in the correct positions. This section describes in detail how to move the print position. Read this chapter before attempting to print anything on a label.

4.1. Origin

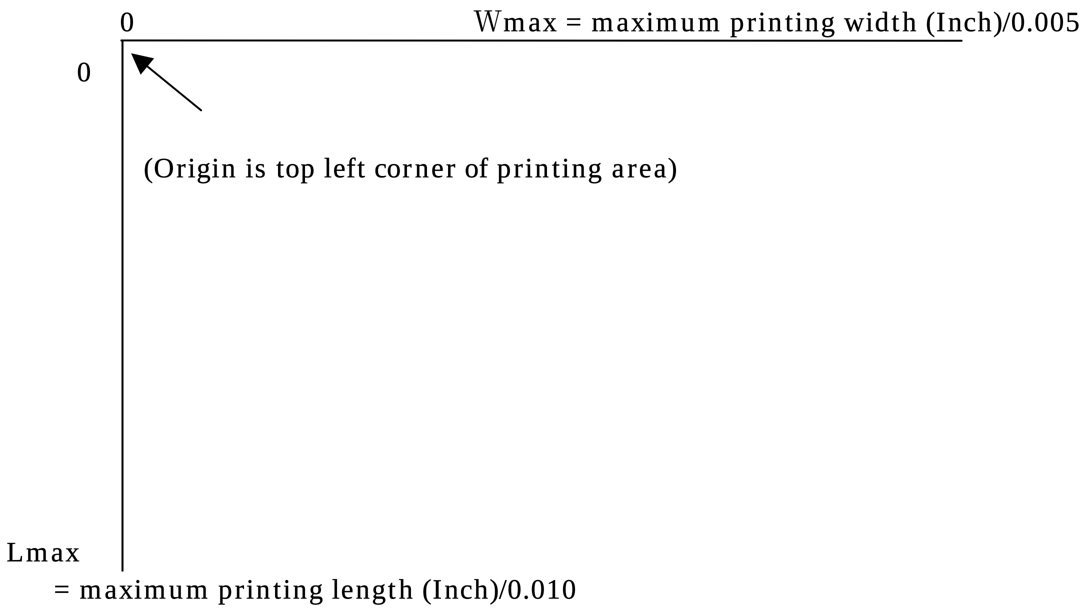
The print position of an object is determined in an x-y coordinate system. This is not a conventional Cartesian coordinate system, in that negative coordinates cannot be used. The origin, (0, 0) is the top left corner of the printing area. All absolute positioning commands are based on this origin, whereas relative positioning commands are based on relative coordinates.

4.1.1. X and Y Values

The maximum x-value (horizontal coordinate) is determined by the maximum printing width, divided by the horizontal pixel size 0.005inches(0.127 mm). An object can be printed at any position in the range from 0 to the maximum x-coordinate.

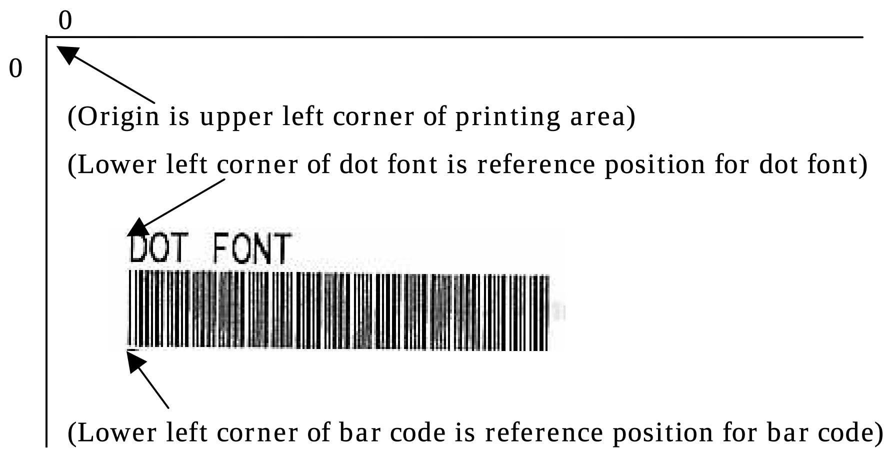
The y-coordinate is computed in the vertical pixel size 0.010inches(0.254 mm). The maximum y-coordinate value (printing range) is the value of the maximum printing length (INCH) divided by the vertical pixel size 0.010inches(0.254mm).

The maximum values in the RCL coordinate system are shown below. (Figure not actual size)


0
Wmax = maximum printing width (Inch)/0.005
0
(Origin is top left corner of printing area)
Lmax
= maximum printing length (Inch)/0.010

4.2. Printing Start Reference Position

Depending on the type of an object, the printing start reference position varies. For dot font text, lines, boxes, logos, and two-dimensional symbols of the matrix type, the reference position is the upper left corner. For vector font text and bar codes, it is the lower left corner.


0
0
(Origin is upper left corner of printing area)
(Lower left corner of dot font is reference position for dot font)
DOT FONT
(Lower left corner of bar code is reference position for bar code)

4.3. Absolute Positioning Commands

An object is positioned in absolute x-y coordinates from the horizontal base reference (HBR command) and vertical base reference (VBR command). These commands set the base reference on which subsequent positions are computed.

4.3.1. Horizontal Base Reference (HBR)

The HBR command specifies the absolute position on the x-axis.
The value is specified in units of horizontal pixels 0.005inches(0.127mm). The range of values is from 0 to the maximum printing width (inche)/0.005.

If the HBR command is not specified, the default value of the horizontal base reference is 20.

The syntax of this command is as follows.

HBR;*XX*;
XX = number of horizontal pixels 0.005inches(0.127mm)

The following example sets the horizontal base reference to 0.5inches(12.7mm).

HBR;*100*;

4.3.2. Vertical Base Reference (VBR)

The VBR command specifies the absolute position on the y-axis.

The value is specified in units of vertical pixels 0.010inches(0.254mm). The range of values is from 0 to the maximum printing length (inch)/0.010.

If the VBR command is not specified, the default value of the vertical base reference is 50.

The syntax of this command is as follows.

```
VBR;YY;
```

YY = number of vertical pixels 0.010inches(0.254mm)

The following example sets the horizontal base reference to 0.15inches(3.81 mm)

```
VBR;15;
```

4.4. Relative Positioning Commands

These two commands, horizontal position relative (HPR) and vertical position relative (VPR), are used to position an object relative to the last printed position (the cursor position). This is useful when it is the relative positions of objects that are important, or when objects must be positioned relative to a base point on the label.

4.4.1. Horizontal Position Relative (HPR)

The HPR command moves the x-coordinate of the print position with respect to the cursor position where the last printing occurred. The minimum HPR value is 1 horizontal pixel 0.005inches(0.127mm), and the maximum value is given by the following formula:

HPR max. = Wmax - HBR

For example, if Wmax is 1008, and the horizontal base reference value is set to 100, the maximum value for the cursor position is 1008－100, or 908 pixels.

If the HPR command is issued a number of times, the movement values specified are added cumulatively. In the following example, the cursor is moved twice along the x-axis.

```
HBR;20;HPR;30;HPR;100;
```

In this case, the initial position on the x-axis is 20, and the cursor is moved first by 30 pixels to the 50 position, then by a further 100 pixels to the 150 position. Since the HPR command has a cumulative effect, it is important to keep an accurate tally of all of the values specified by the HPR command in the program.
As long as the absolute coordinate after movement does not go negative, the parameter to the HPR command can be negative. For example, the following commands are valid:

```
HBR;100;HPR;-50;
```

The syntax of this command is as follows.

```
HPR;XX;
```

XX = number of horizontal pixels 0.005inches(0.127mm)

4.4.2. Vertical Position Relative (VPR)

The VPR command moves the y-coordinate of the print position with respect to the cursor position where the last printing occurred. The minimum VPR value is 1 vertical pixel 0.010inches(0.254mm), and the maximum value is given by the following formula:

VPR max. = (printing length) - VBR

For example, if the label printing length is set to 100 pixels in the header, and the vertical base reference value is set to 10, the maximum VPR value on the label is 100－10, or 90 pixels.

Like the HPR command, the VPR command is cumulative. The following is an example:

```
VBR;50;VPR;25;VPR;60;
```

In this example, from the absolute position on the y-axis of 50, the cursor is first moved down by 25 pixels to the 75 position, then down by a further 60 pixels to the 135 position.
As long as the absolute coordinate after movement does not go negative, the parameter to the VPR command can be negative.

The syntax of this command is as follows.

```
VPR;YY;
```

YY = number of vertical pixels 0.010inches(0.254mm)

4.4.3. End Of Line (EOL)

The EOL command returns the horizontal position to the current setting of the horizontal base reference. This is thus equivalent to a carriage return function, and does not perform a line feed.

The syntax of this command is as follows.

```
EOL;
```

This command can be used almost as a newline command. Normally at the end of a line of text it returns the cursor position to the beginning of the current line. This command does not affect the vertical positioning. It is therefore necessary when printing text to issue a vertical positioning command to move down to the position of the next line of text.

The following example shows the use of the EOL command.

```
~^"IL4-6";1;0;100;0;
SPB;VBR;505;HBR;100;
"This is the first line.";EOL;VPR;30;
"This is the next line.";
TRM;¥
```

```
          This is the first line.
          This is the next line.
```

If the EOL command is omitted, the following is the result.

```
          This is the first line.
                                : This is the next line.
```

4.4.4. HOME Position (HOME)

The HOME command affects both vertical and horizontal positions, moving the cursor to the current base reference position set by the HBR and VBR commands.

The syntax of this command is as follows.

HOME;

In the following example, the HOME command is used to ensure that the text and underline begin from the same position.

```
~^"IL4-7";1;0;200;0;
SPB;VBR;100;HBR;200;
"12345";HOME;
HLT;1;DHL;0;0;112;
TRM;¥
```

```
                  １２３４５
                  ‾‾‾‾‾‾‾‾‾‾
```

5.Horizontal and Vertical Lines

There are four commands which can be used to draw vertical and horizontal lines on labels, in a variety of lengths and thickness: horizontal line thickness (HLT), draw horizontal line (DHL), vertical line thickness (VLT), and draw vertical line (DVL).
Section 5.1 describes horizontal lines, and Section 5.2 describes vertical lines.

5.1. Horizontal Lines

A horizontal line can be drawn anywhere on the label, with a length from 0.005inches (0.127 mm) to the maximum printing width, and a thickness from 0.010inches(0.254 mm) to 1inches(25.4 mm.) Set the horizontal line thickness, then draw the line on the label. The horizontal line thickness command also determines the thickness of horizontal lines using in drawing boxes. For the maximum printing width of the printer, see the specifications in Chapter 18.

5.1.1. Horizontal Line Thickness (HLT)

This command sets the thickness of horizontal lines, and also of the horizontal lines used when drawing boxes. The line thickness must be set before drawing the line or box. The syntax of this command is as follows.

```
HLT;YY;
```

YY = number of vertical pixels 0.010inches(0.254 mm)

The setting value is in the range from 1 pixel 0.010inches(0.254 mm) to 100 pixels 1inches(25.4 mm). If the HLT command is not issued, the default value is 3 pixels 0.030inches(0.762 mm). Once the HLT command is issued, the setting remains valid until changed by another HLT command.

5.1.2.Draw Horizontal Line (DHL)

The DHL command draws a horizontal line on the label. This command has three parameters.

```
DHL;X;Y;XX;
```

X = x-coordinate of start position
Y = y-coordinate of start position
XX = length of line in horizontal pixels 0.005inches(0.127 mm)

This draws a horizontal line from the position offset from the current cursor position by the X and Y values. To draw the line from the current cursor position, set both X and Y to zero.

Note

If the HLT and VLT settings specify a thick line, it is possible for the thermal transfer head to overheat locally, and melt the base film of the thermal transfer ribbon, causing ribbon breaks and degraded printing. In this case, reduce the line thickness.

Note that the start position indicates the upper left corner of the line. As the line gets thicker, it widens downwards from the upper left start position.
Another important point about drawing horizontal and vertical lines, is that the cursor position remains unchanged. That is, as soon as the line is drawn, the cursor returns to the previous position.

5.2. Vertical Lines

To draw a vertical line, use the vertical line thickness (VLT) and draw vertical line (DVL) commands. A vertical can have a thickness from 1 pixel 0.005inches(0.127 mm) to 200 pixels 1inches(25.4 mm), and a length from 2 pixels 0.010inches(0.254 mm) to the maximum printing length.
First define the line thickness with the VLT command, then use the DVL command to draw a line with that thickness.
For the maximum printing length of the printer, see the specifications in Chapter 18.

5.2.1. Vertical Line Thickness (VLT)

The VLT command sets the thickness of vertical lines, and also of the vertical lines used when drawing boxes. The syntax of this command is as follows.

VLT;*XX*;
XX = number of horizontal pixels 0.005inches(0.127 mm)

The minimum thickness of a line is 1 pixel 0.005inches(0.127 mm), and the maximum thickness 200 1inches(25.4 mm). If the VLT command is not issued, the default value is 2 pixels 0.010inches(0.254 mm). Once the VLT command is issued, the setting remains valid until changed by another VLT command.

Note

To draw horizontal and vertical lines of the same thickness, make the pixel count for vertical lines twice that for horizontal lines. This is because the printer pixels are rectangles of height twice their width. For example, if the HLT setting is 2, the line thickness is 2×0.010”(0.254mm) = 0.020”(0.508 mm), and the VLT should be 4. This makes the same thickness: 4 × 0.005”(0.127mm) = 0.020” (0.508 mm).

5.2.2.Draw Vertical Line (DVL)

To draw a vertical line anywhere on the label, use the DVL command. This command has three parameters, and all are required.

DVL;*X*;*Y*;*YY*;

X = x-coordinate of start position
Y = y-coordinate of start position
YY = length of line in vertical pixels 0.010inches(0.254 mm)

This draws a horizontal line from the position offset from the current cursor position by the X and Y values. To draw the line from the current cursor position, set both X and Y to zero.

5.3. Diagonal Lines

To draw a diagonal line, use the draw diagonal line (DDL) command.
The DHL and DVL commands use the HLT and VLT command settings to determine the line thickness, but DDL draws a line whose thickness is set by the printer software. The width of a line drawn by the DDL command depends on the line direction, but varies between approximately 0.010inches(0.254mm) and 0.014inches(0.356 mm).

5.3.1. Draw Diagonal Line (DDL)

This command draws a diagonal line of any length from any position on the label. The syntax of the DDL command is as follows.

DDL;*X1*;*Y1*;*X2*;*Y2*;

X1 = x-coordinate of start position
Y1 = y-coordinate of start position
X2 = x-coordinate of end position
Y2 = y-coordinate of end position

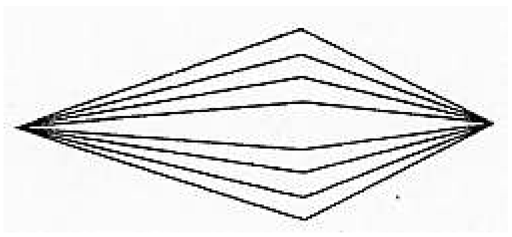
This draws a line from the start point (X1, Y1) to the end point (X2, Y2) in coordinates relative to the current cursor position. The cursor position does not change after drawing the line.

The following example shows the use of the DDL command.

```
~^"DDL";1;0;400;0;SPB;UTOF;4100;


HBR;  0;VBR;100;
DDL;400; 25;100; 75;
DDL;100; 75;400;125;
DDL;400;125;600; 75;
DDL;600; 75;400; 25;
DDL;400; 38;100; 75;
```

```
DDL;100; 75;400;113;
DDL;400;113;600; 75;
DDL;600; 75;400; 38;
DDL;400; 50;100; 75;
DDL;100; 75;400;100;
DDL;400;100;600; 75;
DDL;600; 75;400; 50;
DDL;400; 63;100; 75;
DDL;100; 75;400; 88;
DDL;400; 88;400; 88;
DDL;400; 88;600; 75;
DDL;600; 75;400; 63;
TRM;¥
```



Note
It is not possible to use the DDL command if X1=X2 (horizontal line) or Y1=Y2 (vertical line). In these cases use the DHL or DVL command.

6.Boxes

RCL makes it easy to draw boxes of various sizes on the label. The types of box which can be drawn are: outline frames, solid black rectangles ("black boxes") , "reverse video" boxes (complement boxes), and frames which erase a rectangle ("white boxes") . The commands for these are: Draw BOX (DBOX), Draw Black BoX (DBBX), Draw Complement BoX (DCBX), and Draw White BoX (DWBX).
This chapter begins by describing the settings such as thickness and position common to these box types, and then describes the command for each box type in detail.

6.1. Box Command Syntax

The syntax for the four box-drawing commands is the same. All four parameters must be specified each time a line is drawn. The parameters are as follows.

```
DBOX;X;Y;XX;YY;
```

X = x-coordinate of start position of box
Y = y-coordinate of start position of box
XX = width of box in horizontal pixels
YY = height of box in vertical pixels

The box can be drawn at any position on the label. The minimum width of a box is 0.020inches(0.508 mm), and the minimum height is 0.020inches(0.508 mm). The maximum size of a box is determined by the label size.

6.2. Specifying the Box Position

Each time a box is drawn, the position is determined by the current cursor position and the offset coordinates specified in the parameters. These offset coordinates (positive or negative) are added to the current cursor position coordinates to get the start position of the box.
For example, suppose the current cursor position is (10, 20), and the following DBOX command is issued.

```
HBR;10;VBR;20;
DBOX;0;0;600;300;
```

The box drawing starts from the horizontal position 10 (horizontal pixels) and vertical position 20 (vertical pixels). The box width is 3inches(600×0.005),76.2 mm (600×0.127), and the height is 3inches(300×0.010),76.2 mm (300×0.254 mm). In other words the box extends from x-coordinate 10 to 610 and from y-coordinate 20 to 320.

After drawing a box, the cursor position returns to the previous position. In the above example the cursor returns to the position (10, 20). Drawing a box or logo does not move the cursor. This is an extremely important point not to forget when positioning objects on the label.

6.3. Outline Boxes

An outline box consists only of the four bounding black lines, and does not affect the interior. Use the DBOX command to draw an outline box.

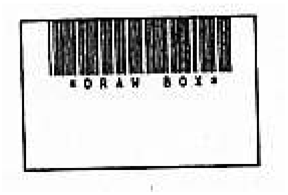
6.3.1. Draw BOX (DBOX)

The DBOX command draws an outline box on the label. The thicknesses of the boundary lines are determined by the HLT and VLT command settings (see Sections 5.1.1 and 5.2.1). If the thicknesses are not set, the default for horizontal lines is 0.030inches(0.762 mm), and for vertical lines is 0.010inches(0.254 mm). To make the vertical and horizontal lines the same thickness, use the HLT and VLT commands to set the line thicknesses before issuing this command.

6.3.2. Using the Box-Drawing Command

For bar code labels, the DBOX command is very important. Firstly, it can be used to draw the outside boundary of the label.
The following example shows this.

```
~^"DBOX";1;0;27;266;SPB;UTOF;0354;
HBR;0;VBR;0;
HBR; 30;VBR; 17;BDEF; 1;BNEW; 2;BWEW; 5;BICG; 2;BCSH; 17;BCST;
"*DRAW BOX*";
BSTP;
HBR; 30;VBR; 19;DDF; 3; 10;DFM; 1; 1;DFO; 1; 1;DFS; 10;
"*DRAW BOX*";
HLT; 1;VLT; 2;
HBR; 0;VBR; 0;DBOX; 0; 0;236; 27;
TRM;¥
```



6.4. BlackBox

A solid rectangle is a box, after the interior has been filled with solid black. This obliterates any other objects, so a solid rectangle must not be used to overlay other objects.

Note
A large solid rectangle may cause the thermal transfer head to overheat locally, and melt the base film of the thermal transfer ribbon, causing ribbon breaks and degraded printing. In this case, reduce the rectangle size, or use some other way of reducing the load on the printer.

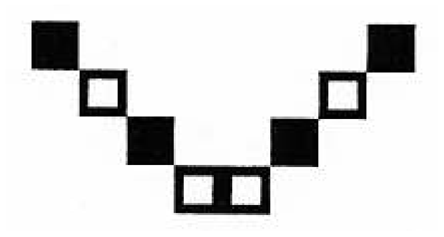
6.4.1. Draw Black Box (DBBX)

The syntax of the DBBX command is the same as that of DBOX. The only difference between the commands is that DBBX fills in the box with solid black.

6.4.2. Example Using the Draw Black Box Command

The following example is a program to draw a number of boxes, half with outlines only (DBOX), and the remainder as solid black rectangles.

```
~^"DBBX";1;0;200;20;SPB;UTOF;2100;
HBR; 0;VBR; 0;
HLT; 5 ;  VLT; 10;
DBBX;100; 0; 50; 25;
DBOX;150; 25; 50; 25;
DBBX;200; 50; 50; 25;
DBOX;250; 75; 50; 25;
DBBX;450; 0; 50; 25;
DBOX;400; 25; 50; 25;
DBBX;350; 50; 50; 25;
DBOX;300; 75; 50; 25;
TRM;¥
```



6.5. Complement Boxes

A complement box inverts the black-white sense of every pixel within the rectangle, changing black to white and white to black.

6.5.1. Draw Complement Box (DCBX)

The DCBX command specifies the rectangle within which the pixels are inverted. RCL allows any textS, lines, and boxes to be inverted, but not bar codes.

To produce text in white-on-black, first specify the text size, then the text to be inverted, then finally specify the area for the inversion, i.e. the complement box.

6.6. White Boxes

A "white box" erases the space within the rectangle, to clear part of a label on which various objects have been drawn. The DWBX command clears the rectangle specified by the parameters. This is used mainly for serial numbers, where before incrementing or decrementing the number, the previous data is cleared. (See Chapter 11 for details of serial number printing.)

6.6.1. Draw White Box (DWBX)

The DWBX command is useful when a command sequence is used to print a number of labels. Generally, on a predefined label, the DWBX command is used to erase a bar code, and replace it with a bar code for new data. When a DWBX is used for serial numbering, the specified area is completely cleared, but other objects in the label are not cleared.

Note

Avoid large areas of solid black (DBBX command) or complement areas (DCBX command). These may cause the printer ribbon to break or wrinkle.

7.Bar Codes

RCL supports seven of the widely used bar codes: Code 39, Interleaved 2 of 5 (ITF), UPC, EAN, CodaBar, Code93, and Code128.

Bar codes can be printed in four orientations, at 0, 90, 180, and 270 degrees. The orientations at 0 degrees and 180 degrees have the bar code elements parallel to the label feed direction, and are referred to as "picket fence" bar codes. The orientations at 90 degrees and 270 degrees have the bar code elements at right angles to the label feed direction, and are referred to as "ladder" bar codes.

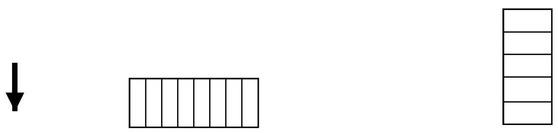
Bar codes are at the heart of RCL, and there are a number of commands to assist with printing bar codes. Any bar code requires at least three commands.
These are bar code selection (Bar code SYMbol, BSYM, and Bar code DEFinition, BDEF), bar code start (BCST), and bar code stop (BSTP), and these are described in Sections 7.1 and 7.2.

For each bar code type, there are also special commands.

For picket fence Code39 and ITF, there are three commands: bar narrow element width (BNEW), bar wide element width (BWEW), and bar code symbol height (BCSH). For Code39 there is also a bar code inter-character gap (BICG) command.
When printing these two types of bar code as ladder bar codes, in place of the BNEW, BWEW, and BICG commands, use the bar code characters per inch command (BCPI).

For UPC and EAN, the UPC magnification (UMAG) is required, and for Code93, Code128, and CodaBar, the bar code characters per inch (BCPI) and bar code symbol height (BCSH) commands are required. These are described in Sections 7.3, 7.4, and 7.5, respectively.
Finally, to print a serial number in a bar code, further commands are required. This is described in Chapter 11.

Note



(Label feed direction) (picket fence) (ladder)

The use of picket fence bar codes is strongly recommended.

In some cases, high-density ladder bar codes may not be able to be read.

7.1. Types of Bar Code

RCL supports seven types of bar code: Code39, ITF, UPC, EAN, Code93, Code128, and CodaBar. These are the types most commonly used.

There are two variants of Code39, with and without check characters. The UPC and EAN types are also subdivided.

> Note
> It is the programmer's responsibility to investigate the latest specifications for the bar code type being used, and to follow guidelines specified.

7.1.1. Bar Code Symbol Selection (BSYM)

The BSYM (Bar code SYMbol) command is always required, to select the type of bar code used, and the direction of printing. (To maintain compatibility with RCL, in place of BSYM it is also possible to use the BDEF command. BDEF is described in Section 7.1.3.)
The bar code type must be selected before creating a bar code. The syntax of this command is as follows.

BSYM;*A*;*B*;
A = integer specifying bar code type, as shown in table below
B = integer 1–4, specifying bar code orientation

The following table shows the BSYM command parameter values for each bar code type.

Bar code type	A	Bar code type	A
Code39	1	EAN13	16
CodaBar	3	EAN8	17
Code39 (MOD43)	5	EAN13 2 Char.	18
ITF	8	EAN13 5 Char.	19
UPC-A	10	EAN8 2 Char.	20
UPC-A 2 Char.	11	EAN8 5 Char.	21
UPC-A 5 Char.	12	Code128	25
UPC-E	13	Casecode Code128	27
UPC-E 2 Char.	14	Code93	30
UPC-E 5 Char.	15	ITF(MOD10Wait3)	52
		CodaBar(MOD16)	54

Bar code orientation	B
0 degrees	1

90 degrees	2
180 degrees	3
270 degrees	4

7.1.2. Bar code DEFinition(BDEF)

The BDEF command is similar to the BSYM command, but only supports two bar code orientations: 0 degrees and 270 degrees.

The bar code type must be selected before creating a bar code. The syntax of this command is as follows.

```
BDEF;A;
```

A = integer 1–55 selected from following table

The following table shows the BDEF command parameter value for each bar code type.

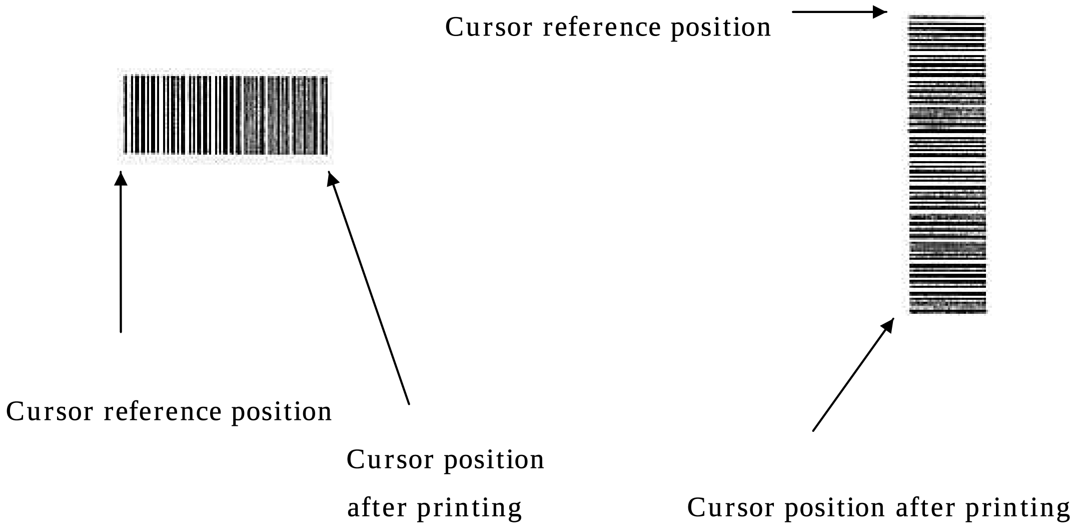
Bar code type	A	Bar code type	A
Code39	1	EAN 8 5 Char.	21
Code39 ladder	2	Code128	25
CodaBar	3	Code128 ladder	26
CodaBar ladder	4	Casecode Code128	27
Code39 (MOD43)	5	Ditto, ladder	28
Ditto, ladder	6	Code93	30
ITF	8	Code93 ladder	31
ITF ladder	9	UPC A ladder	40
UPC-A	10	UPC A 2 Char. ladder	41
UPC-A 2 Char.	11	UPC A 5 Char. ladder	42
UPC-A 5 Char.	12	UPC E ladder	43
UPC-E	13	UPC E 2 Char. ladder	44
UPC-E 2 Char.	14	UPC E 5 Char. ladder	45
UPC-E 5 Char.	15	EAN 13 ladder	46
EAN-13	16	EAN 8 ladder	47
EAN-8	17	EAN 13 2 Char. ladder	48
EAN 13 2 Char.	18	EAN 13 5 Char. ladder	49
EAN 13 5 Char.	19	EAN 8 2 Char. ladder	50
EAN 8 2 Char.	20	EAN 8 5 Char. ladder	51
ITF (MOD10 Wait 3)	52	ITF (MOD10 Wait 3) ladder	53
CodaBar (MOD16)	54	CodaBar (MOD16) ladder	55

If neither a BDEF command nor a BSYM command is issued, the default value of 1 (Code39) is assigned. If a value not listed in the tables above is specified, again the setting defaults to 1. Once a BDEF command or BSYM command has been issued to select a bar code type, this type remains selected until a new selection is made.

7.2. Bar Code Printing

Use the HBR, VBR, HPR, VPR, HOME, and EOL commands to position a bar code (see Chapter 4 for details of these commands.) The bar code is printed with the lower left corner at the cursor position.

In other words, for a picket fence bar code in the 0 degrees orientation, the bar code is printed to extend up and to the right from the cursor position, and for a ladder bar code in the 270 degrees orientation to extend right and down from the cursor position. When the bar code printing is completed, the base position for the next object is the lower right corner of a picket fence bar code and the lower left corner of a ladder bar code.
The origin is the upper left corner of the printing area.


Cursor reference position
Cursor reference position
Cursor position after printing
Cursor position after printing

For printer models with a verify function to be able to scan the bar code correctly, when setting the bar code position, the following rules must be observed.

1. There must be a blank space of at least 3 mm before and after the bar code.
2. The bar code must comply with the latest version of the specifications. In particular, note the specification of the bar code height, the element dimensions, and the bar code density.

7.2.1. Bar Code Data Entry

Bar code data must always be included in double quotation marks ("..."). Depending on the bar code type, only certain ASCII characters may be allowed, and the number of characters may also be fixed. Code39 and CodaBar require the programmer to define a start character and a stop character. CodaBar also allows a check character

to be added if required. For ITF, UPC, and EAN, the start character and stop character are determined automatically.
Code128 adds a modulo 103 check character to the end of the bar code data. Casecode Code128 data is subject to restriction. This is described elsewhere.
There are no restrictions on the number of bar codes which can be printed within the printing area.

For UPC and EAN, the bar code data can be converted to readable form and printed together. For Code93, ITF, Code39, Code128, and CodaBar, the readable form must be printed separately with a text command. (See Chapter 10.) In this base the bar code data and readable text should be the same.

7.2.2. Bar Code StarT (BCST)

The BCST command instructs the printer to print the following data as a bar code. If the BCST command is omitted, the data will be treated as ordinary text, and printed in text form. The syntax of this command is as follows.

```
BCST;
```

7.2.3. Bar Code SToP (BSTP)

The BSTP command is similar to the BCST command, but marks the end of the data. This command must immediately follow the last character of the bar code data. If the BSTP command is omitted, the printer cannot tell where the end of the bar code is. The syntax of this command is as follows.

```
BSTP;
```

The following example, selects a Code39 bar code, and enters the data.

```
   BDEF;1;BCST;"*A1Z0B2Y9*";BSTP;
```

7.3. Code39 and ITF

Code39 and ITF bar codes are both defined similarly. First use BSYM or BDEF to define the bar code type, then BCST and BSTP to delineate the beginning and end of the bar code data, and then to print the bar code, bar code narrow element width (BNEW), bar code wide element width (BWEW), and bar code symbol height (BCSH) settings must be made.

For Code39 the bar code inter-character gap can be set with the BICG command, but this command is not normally required.
As described in the previous section, the BNEW, BWEW, and BICG commands are used to set the density of a picket fence bar code. For a ladder bar code, in place of these three commands, a bar code characters per inch (BCPI) command is used.
There are two types of Code39 bar code, depending on whether or not there is a check character. With the specification BSYM;1;x;, BDEF;1;, and BDEF;2; there is no check character. In other words, the bar code data enclosed in quotes is reproduced unaltered as the Code39 bar code. With specifications BSYM;5;x;, BDEF;5;, and BDEF;6; the printer calculates a modulo 43 check character from the input bar code data, and adds it immediately before the stop character. For this reason, the number of digits in the encoded data is more than that input. An HIBC label using Code39 has this type of modulo 43 check character. The modulo 43 check character is computed as follows.

1. Each character is assigned the value in the following table.

Char-a cter	Value	Char-a cter	Value	Char-a cter	Value	Char-a cter	Value	Char-a cter	Value
0	0	9	9	I	18	R	27	—	36
1	1	A	10	J	19	S	28	.	37
2	2	B	11	K	20	T	29	space	38
3	3	C	12	L	21	U	30	$	39
4	4	D	13	M	22	V	31	/	40
5	5	E	14	N	23	W	32	+	41
6	6	F	15	O	24	X	33	%	42
7	7	G	16	P	25	Y	34		
8	8	H	17	Q	26	Z	35		

2. The sum of all of the character values is taken modulo 43.

3. The character equivalent to the result is taken as the check character.

Example:
If the bar code data is "DATA":
D = 13, A = 10, T = 29, A = 10
13＋10＋29＋10=62

62 mod 43=19
19 = "J"
Therefore when the check character is added the resulting string is を"DATAJ".

This section describes Code39 and ITF bar codes.

7.3.1. Bar Narrow Element Width (BNEW)

The BNEW command sets the width of a narrow element of a picket fence Code39 or ITF bar code. For Code39, if there is no BICG command, it also sets the inter-character space.

The syntax of this command is as follows.

```
BNEW;X;
```

X = element width in head density units

The maximum and minimum values for BNEW are determined by the type of bar code being used. For more details, refer to the relevant standards. If the BNEW command is not issued, the default value is 3.

7.3.2. Bar Wide Element Width (BWEW)

The BWEW command sets the width of a wide element of a picket fence Code39 or ITF bar code.

For either bar code, the ratio of widths of the wide and narrow elements is generally between 2:1 and 3:1. The syntax of this command is as follows.

```
BWEW;X;
```

X = element width in head density units

If the BWEW command is not issued, the default value is 9.

7.3.3. Bar Code Symbol Height (BCSH)

The BCSH command sets the height of the bar code, excluding the readable form.
For bar codes such as Code39, ITF, Code93, Code128, and CodaBar, the height must be set with this command. For UPC and EAN bar codes, the UMAG command automatically sets the height, and this command is not required. However, for UPC or EAN bar codes, to restrict the height of the bar code printed, use the BCSH command. In this way it is possible to print a bar code at less than the standard height.

The syntax of this command is as follows.

For a picket fence bar code:

```
BCSH;YY;
```

YY = height of bar code in vertical pixels 0.010inches(0.254 mm)

For a ladder bar code:

```
BCSH;XX;
```
XX = height of bar code in horizontal pixels 0.005inches(0.127 mm)

The maximum and minimum values for BCSH are determined by the type of bar code being used. For more details, refer to the relevant standards. If the BCSH command is not issued, the default value is 50 ((0.5")12.7 mm) for a picket fence bar code, and 50 ((0.25")6.35 mm) for a ladder bar code.

Note
When the orientation of the bar code is changed with the BSYM or BDEF command, the BCSH command must be issued again. If it is not, then when changing from a picket fence bar code to a ladder bar code, the ladder bar code will be printed at half of the height of the picket fence bar code. In the reverse direction, a picket fence bar code will be printed at twice the height of the corresponding ladder bar code.

7.3.4. Bar Code Inter-Character Gap (BICG)

The BICG command sets the character spacing for Code39.
The syntax of the BICG command is as follows.

```
BICG;X;
```
X = character spacing in head density units

The maximum and minimum values for BICG are determined by the type of bar code being used. For more details, refer to the relevant standards. If the BCSH command is not issued, the bar code inter-character gap is the same as the width of a narrow element.

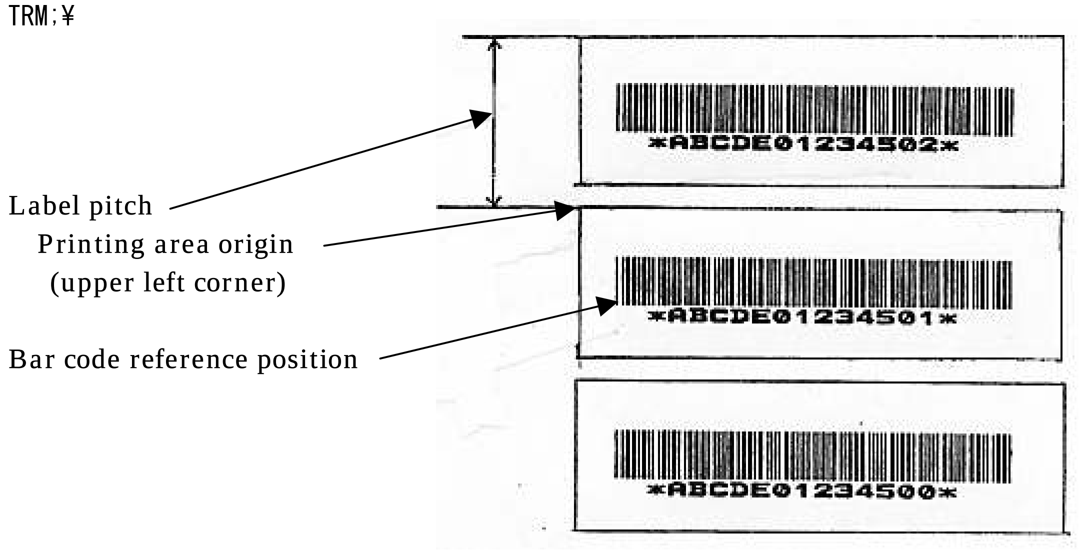
7.3.5. Examples Using Code39 and ITF

The following example creates a bar code using the commands described so far.
This example prints a Code39 bar code on a 20 mm×70 mm label.
The printing area is the length of the label (20 mm), thus 20/0.254 = 79 pixels.
The label pitch is 23 mm, and the UTOF value is 354. (23/0.254×10 = 905)

Note
For Code39 it is important not to omit the terminating asterisk.

```
~^"SAMPLE";5;0;79;187;
SPB;UTOF;905;
BSYM;1;1;                           #CODE39  : 0 deg. #;
BNEW;3;BWEW;7;BICG;3;
DDF;4;10;DFM;2;2;DFS;3;DFO;1;1;     #I      FONT    : 0 deg. #;
MRK;                                #LOOP#;
```

```
HBR;0;VBR;0;DWBX;0;0;394;79;NUM;
BCLC;1;BCID;1;
IDF;1;
HBR;33;VBR;25;BCSH;25;                      #BarCode Data#;
BCST;"*ABCDE01234500";BSAL;2;"*";BSTP;
HBR;66;VBR;27;                              #Font Data#;
"*ABCDE01234500";SAL;2;"*";
RET;
TRM;¥
```



7.3.6. Code39 and ITF Ladder Bar Codes

The density of Code39 and ITF ladder bar codes is set using the BCPI command in place of the BNEW, BWEW, and BICG commands. For details of the BCPI command, see Section 7.5.1.
For a Code39 ladder bar code, the inter-character gap is equal to the narrow element width.

7.4. UPC and EAN

UPC, and the corresponding European version EAN, can be printed in the same way. Using BSYM or BDEF to select the bar code type, and BCST and BSTP to denote the beginning and end of the bar code, the settings are the same as for other bar codes. The extra command required for these bar codes is the UPC magnification setting (UMAG command).

7.4.1. Nominal Size and Magnification

The printer can print UPC/EAN picket fence bar codes in eight sizes, and UPC/EAN bar codes in three sizes. These sizes are specified as ratios to the nominal size. For a picket fence bar code the minimum value of the magnification is 0.76, and the maximum value is 2.08; for a ladder bar code the minimum value is 0.77 and the maximum value is 1.54.

7.4.2. UPC Magnification (UMAG)

The UMAG command specifies the magnification for printing a UPC/EAN bar code. The larger the magnification the larger the resulting bar code. The syntax of this command is as follows.

```
UMAG;A;
```

A = integer value from separate table

> Note
> Even for the same UMAG value, the printed sizes of a picket fence bar code and ladder bar code are different.

The UMAG command automatically determines the bar and space dimensions, the bar code height, size, and layout of readable characters.

7.4.3. Examples Using UPC and EAN

After specifying UMAG, next enter the bar code data between BCST and BSTP commands, surrounded by quotes. The start character, stop character and check digit are computed automatically, and need not be specified. There are different subtypes of UPC and EAN, according to the number of digits, and rigid rules for each subtype. The following table shows the number of digits which must be enclosed between the quotation marks for each of the UPC and EAN subtypes.
Note that for UPC-E, the number of digits either before or after compression may be entered.

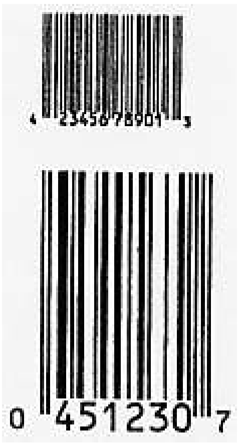
BSYM A value	BDEF values	Type	Digits	BSYM A value	BDEF values	Type	Digits
10	10,40	UPC-A	11	16	16,46	EAN 13	12
11	11,41	UPC-A 2	13	17	17,47	EAN 8	7
12	12,42	UPC-A 5	16	18	18,48	EAN 13 2	14
13	13,43	UPC-E	11 or 7	19	19,49	EAN 13 5	17
14	14,44	UPC-E 2	13 or 9	20	20,50	EAN 8 2	9
15	15,45	UPC-E 5	16 or 12	21	21,51	EAN 8 5	12

If the bar code height is too high for the specified magnification, by inserting a BCSH command between the BSYM or BDEF command and the BCST command, the bar code height can be reduced. By using BCSH to specify the height in vertical pixels (0.254 mm), for a particular magnification the height can be restricted as required. For a UPC/EAN bar code whose height is restricted, the magnification must be specified with the UMAG command.

The following examples show RCL being used to print different UPC/EAN bar codes.

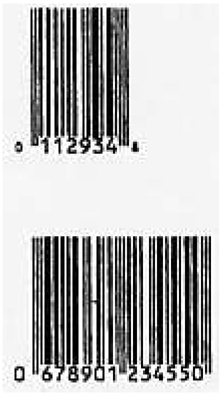
Example 1

```
~^"UMAG1";1;0;400;0;SPB;UTOF;4100;
HBR; 50;VBR;130;BDEF;10;UMAG;1;BCST;
"42345678901";BSTP;
HBR; 50;VBR;350;BDEF;13;UMAG;7;BCST;
"04500000123";BSTP;
TRM;¥
```



Example 2

```
~^"UMAG2";1;0;400;0;SPB;UTOF;4100;
HBR; 50;VBR; 80;BDEF;13;UMAG;2;BCST;
"01129000003";BSTP;
HBR; 50;VBR;350;BDEF;16;UMAG;2;BCST;
"067890123455";BSTP;
TRM;¥
```



7.5. CodaBar

A CodaBar bar code is defined with the three mandatory commands, BSYM or BDEF, BCST, and BSTP, and also using the BCSH command bar code characters per inch (BCPI) command.

7.5.1. Bar code Characters Per Inch (BCPI)

The BCPI command specifies the number of characters per inch in a CodaBar bar code. This command automatically determines the widths of narrow and wide elements. The syntax of this command is as follows.

```
BCPI;A;
```

A = integer from separate table specifying density (cpi)
The density setting values are listed in the tables in Chapter 19.

7.5.2. Examples Using CodaBar

A CodaBar bar code including any number of digits can be printed. The programmer must encode the start character and stop character in the data. To print a human-readable form, this must be specified with a separate text printing command. (See Chapter 8.)

There are four variants of the CodaBar format: without a check character, with a modulo 11 check character, with a modulo 10 check character, and with both modulo 10 and modulo 11 check characters, and any of these four variants can be selected for printing. To insert a modulo 11 check character, put an '@' character at the required point in the data string; to insert a modulo 10 check character, put a '#' character at the required point in the data string.

When using check characters, special rules must be followed. These are as follows.

Modulo 11 check character (@)

1. Only digits 0 to 9 can be used in the bar code data.
2. The data for a bar code is a maximum of six digits. To these six characters are added the start character, stop character, and check character, for a total of nine characters.
3. If more than six digits are included in the bar code data, the first six digits are taken, and the check character calculated. The remaining digits are printed in the bar code, but not included in the modulo 11 calculation. If there are characters other than numeric digits (0–9), these characters are ignored for the check character calculation, and are not printed in the bar code.
4. The software computes the check character, and prints it in the position of the '@' character.
5. The modulo 11 check character is computed as follows.

Step 1: Take six digits following the start character.
Step 2: Multiply the digits obtained in step 1 by the respective weightings 7, 6, 5, 4, 3, and 2, then add to form the weighted sum S.
Step 3: Find the remainder (R) of the weighted sum S modulo 11.
Step 4: Subtract the remainder R from 11: this gives the check character.

Example:
Input data: "A@237352B"

First six digits: "237352"

Weighting		Value	
7	x	2	= 14
6	x	3	= 18
5	x	7	= 35
4	x	3	= 12
3	x	5	= 15
2	x	2	= 4
		S	= 98

R = 98 modulo 11 = 10
C = 11–10 = 1 ← modulo 11 check character

Modulo 10 check character (#)

1. Only digits 0 to 9 can be used in the bar code data.
2. If there are characters other than numeric digits (0–9), these characters are ignored for the check character calculation, and are not printed in the bar code.
3. The software computes the check character, and prints it in the position of the '#' character.
4. The modulo 10 check character is computed as follows.

Step 1: Excluding the start character, stop character, and check character, number the digits from the end (1, 2, 3,...).
Step 2: Double the values of odd-numbered digits.
Step 3: Add the values of even-numbered digits.
Step 4: Take the total of the values in steps 2 and 3.
Step 5: Subtract the total arrived at in step 4 from the closest larger multiple of 10. This is the check character C.

Example:
Input data: "A123541#B"

Digit position	6 5 4 3 2 1
Value	1 2 3 5 4 1

2 + 5 + 1 = 8 x 2 = 16
1 + 3 + 4 = 8

Total 24

C = 30 – 24 = 6 ← modulo 10 check character
The bar code data is therefore "A1235416B"

The following example shows various CodaBar bar codes being printed.

```
 ~^″CODABAR″;1;0;400;0;SPB;UTOF;4100;
HBR;150;VBR; 50;BDEF;3;BCPI; 1;BCSH; 40;
BCST;″C34567890123D″;BSTP;
HBR;210;VBR; 70;DHR;$8002;DCH;15;DCW;30;ICS; 2;
″C34567890123D″;
HBR;150;VBR;200;BCPI; 0;BCSH; 75;
BCST;″B123456@C″;BSTP;
HPR;380;HBR;190;VBR;220;
″B123456C″;
TRM;¥
```

C34567890123D

B123456C

7.6. Code93

A Code93 bar code, like CodaBar, requires the three bar code commands BSYM or BDEF, BCST, and BSTP, and also the BCSH and BCPI commands.

7.6.1. Code93 Bar Code Characters Per Inch (BCPI)

In the same way as for CodaBar, the BCPI command specifies the number of characters per inch in a Code93 bar code. This command automatically sets the widths of elements in the Code93 bar code. The syntax of this command is as follows.

```
BCPI;A;
```

A = density value (cpi) from separate table

The density setting values are listed in the Appendix.

7.6.2. Examples Using Code93

Code93 uses a combination of 43 characters (0–9, A–Z, six special characters, and space), start and stop characters, and four control characters in a data character set which allows all 128 ASCII characters to be represented.

Code93 includes two check digits, and thus keeps reading errors to the absolute minimum. Also, any number of characters may be used in a bar code. The start and stop characters are automatically added, but a to print a human-readable form, this must be specified with a separate text printing command. (See Chapter 8.)

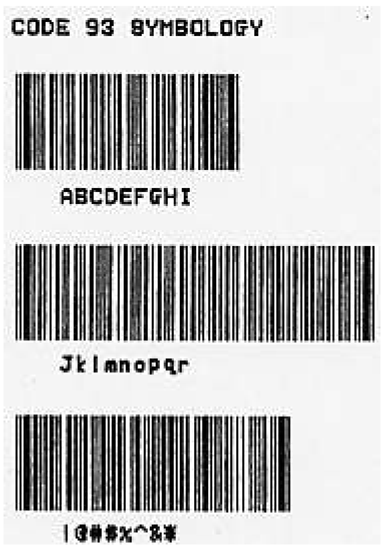
Note

If BCPI is set to zero, it may not be possible to print a readable product bar code.

Except for NULL (00_H), ENQ (05_H), DC2 (12_H), DC4 (14_H), and CAN (18_H), all control characters can be included as readable characters in a bar code.

The following example shows various Code93 bar codes which can be printed.

```
~^"CODE93";1;0;400;0;SPB;UTOF;4100;
HBR;300;VBR; 20;DHR; 1;DCH;12;DCW;24;ICS; 4;
"CODE 93 SYMBOLOGY";
HBR; 20;VBR; 90;BDEF;30;BCSH;50;BCPI; 2;BCST;
"ABCDEFGHI";BSTP;
HBR; 70;VBR;110;
"ABCDEFGHI";
HBR; 20;VBR;180;BCST;
"Jklmnopqr";BSTP;
HBR; 70;VBR;200;
"Jklmnopqr";
HBR; 20;VBR;270;BCST;
"!@#$%&*";BSTP;
HBR; 70;VBR;290;
"!@#$%^&*";
TRM;¥
```



7.7. Code128

A Code128 bar code, like Code93 and CodaBar, requires the three standard bar code commands BSYM or BDEF, BCST, and BSTP, and also the BCSH and BCPI commands.

7.7.1. Code128 Bar Code Characters Per Inch (BCPI)

In the same way as for Code93, the BCPI command specifies the number of characters per inch in a Code128 bar code. This command automatically sets the widths of elements in the Code128 bar code. The syntax of this command is as follows.

```
BCPI;A;
```

A = integer from separate table specifying density (cpi)
The density setting values are listed in the tables in Chapter 19.

Note
If BCPI is set to zero, it may not be possible to print a readable product bar code.

7.7.2. Examples Using Code128

As its name suggests, the Code128 format allows all 128 characters of the ASCII character set to be printed in a bar code. Code93 also allows the whole ASCII character set to be encoded, but for lowercase letters, special characters, and control characters, Code128 requires fewer symbols. For coding numeric values only, the high-density encoding (Code subset C) can be used.

Code128 has three different subsets: Code A, Code B, and Code C. The subset being used can selected with the corresponding START code, and it is also possible to switch to a different subset in the middle of a bar code, using a special CODE character, or change one character only to a different subset using a SHIFT character.

The following is a summary of the three subsets; for more details, refer to a description of AIM, USD-6 Code128.

- Code subset A provides the uppercase letters, numbers, and special characters of a standard ASCII keyboard, together with control codes 00_H to $1F_H$.
- Code subset B provides the alphanumeric characters including lowercase letters of a standard ASCII keyboard, together with and special Code128 characters.
- Code subset C provides high-density characters, encoding two-digit numbers from 00 to 99, and Code128 special characters.

Code128 allows any number of characters to be used in a bar code.
A START character (code subset indicator) must be specified. To print a human-readable form, this must be specified with a separate text printing command. (See Chapter 8.)

7.7.2.1. Selecting and Using Subsets

Select the indicator character from the following table for the code subset to be used, and specify this as the first character of the bar code data.

Code subset	Indicator
A	A
B	B
C	C

Note

If an erroneous indicator character is specified, this is taken as 'A' (default value).

Within each code subset, special character data is specified by the following method. To switch the code subset at an intermediate point in the data, refer to this table. (For example, to switch from code subset A to C, insert the sequence '@D' between the subset A and subset C data.)

Code A	Code B	Code C	Chars.	Code A	Code B	Code C	Chars.
@	@		@@	CodeC	CodeC		@D
FNC3	FNC3		@A	CodeB	FNC4	CodeB	@E
FNC2	FNC2		@B	FNC4	CodeA	CodeA	@F
SHIFT	SHIFT		@C	FNC1	FNC1	FNC1	@G

To facilitate the use of a host from which it is difficult to transmit control characters or lowercase letters (e.g. IBM, EBCDIC), the following extensions are made to the code subsets and the corresponding data.

1) In both of code subsets A and B, all characters from the ASCII character set can be transferred.
   In the following example, instead of inserting hexadecimal values (CR=$0D_H$, LF=$0A_H$) in the command sequence, the carriage return and line feed to be printed in the bar code are replaced by special bar code characters (code subset A [CR] = code subset B 'm'; code subset A [LF] = code subset B 'j').

   Example::
   BCST;"ATOHOKUmjINC";BSTP;

2) In the following method, codes from code subset C can be inserted in のcode subsets A and B.
   @Y = start data from code subset C
   @Z = end data from code subset C

Note

The '@Y' and '@Z' sequences themselves are not printed in the bar code. They are simply escape sequences to indicate to the software that in code subset C data, characters from code subset A or B are being sent.

Example:
This example uses '@Y'/'@Z' sequences in code subset A.

BCST;"ATEST@Y7774@Z@E@Y737867@Z";BSTP;

Chars.	Meaning
A	Select code subset A.
TEST	Bar code data in code subset A
@Y	Escape sequence indicating that following data is represented in code subset C
77	Carriage return represented in code subset C (within code subset A data)
74	Line feed represented in code subset C (within code subset A data)
@Z	End data in code subset C.
@E	Select code subset B.
@Y	Begin data in code subset C.
73	'i' represented in code subset C (within code subset B data)
78	'n' represented in code subset C (within code subset B data)
67	'c' represented in code subset C (within code subset B data)
@Z	End data in code subset C.

Note
All control codes can be printed as bar codes except NULL (00_H), ENQ (05_H), BS (08_H), FF ($0C_H$), DC2 (12_H), DC4 (14_H), CAN (18_H), and DEL ($7F_H$).

The following pages show various example bar codes with Code128.

Example: switching from code subset A to code subset C
BCST;"ATEST@D123";BSTP;

Example: switching from code subset C to code subset B
BCST;"C1234@ETEST";

The following example shows various Code128 bar codes which can be printed.

```
~^"CODE128A";1;0;400;0;SPB;UTOF;4100;
HBR; 10;VBR; 20;DHR;32768;DCH;12;DCW;22;ICS; 4;
"CODE 128 SYMBOLOGY - SUBSET A WITH FUNCTIONS";
HBR; 50;VBR; 90;BDEF;25;BCSH;50;BCPI; 2;
BCST;"AABCDE@AFGHIJKLM";BSTP;
HBR;100;VBR;110;
"ABCDEFGHIJKLM";
HBR; 50;VBR;170;
BCST;"ANOPQR@GSTUVWXYZ";BSTP;
HBR;100;VBR;190;
"NOPQRSTUVWXYZ";
HBR; 50;VBR;250;
BCST;"AABCDEFGHIJK@BLM";BSTP;
HBR;100;VBR;270;
"ABCDEFGHIJKLM";
HBR; 50;VBR;330;
BCST;"ANOPQR@GSTUVWXYZ";BSTP;
HBR;100;VBR;350;
"NOPQRSTUVWXYZ";
TRM;¥
```

CODE 128 SYMBOLOGY – SUBSET A WITH FUNCTIONS

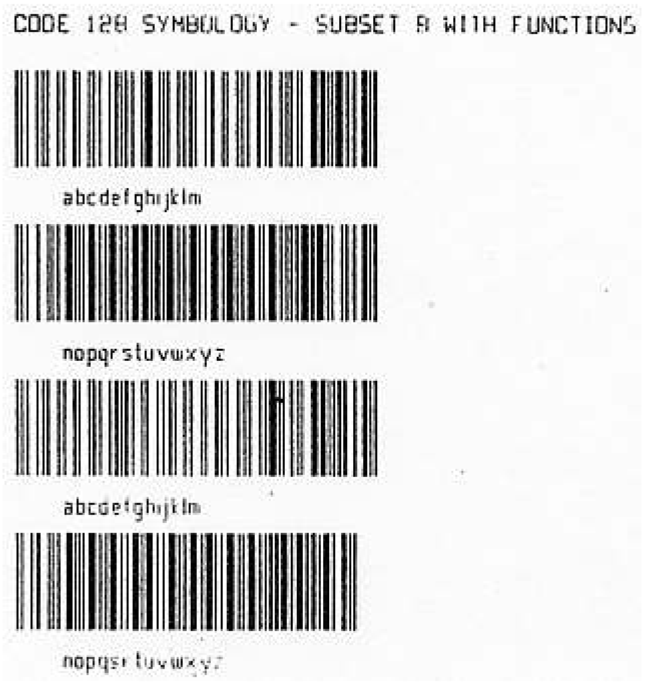
ABCDEFGHIJKLM

NOPQRSTUVWXYZ

ABCDEFGHIJKLM

NOPQRSTUVWXYZ

```
~^"CODE128B";1;0;400;0;SPB;UTOF;4100;
HBR; 50;VBR; 20;DHR;32768;DCH;12;DCW;22;ICS; 4;
"CODE 128 SYMBOLOGY - SUBSET B WITH FUNCTIONS";
HBR; 50;VBR; 90;BDEF;25;BCSH;50;BCPI; 2;
BCST;"Babcde@Afghijklm";BSTP;
HBR;100;VBR;110;
"abcdefghijklm";
HBR; 50;VBR;170;
BCST;"Bnopqr@Gstuvwxyz";BSTP;
HBR;100;VBR;190;
"nopqrstuvwxyz";
HBR; 50;VBR;250;
BCST;"Babcdefghijk@Blm";BSTP;
HBR;100;VBR;270;
"abcdefghijklm";
HBR; 50;VBR;330;
BCST;"Anopqr@gstuvwxyz";BSTP;
HBR;100;VBR;350;
"nopqsrtuvwxyz";
TRM;¥
```


CODE 128 SYMBOLOGY - SUBSET B WITH FUNCTIONS

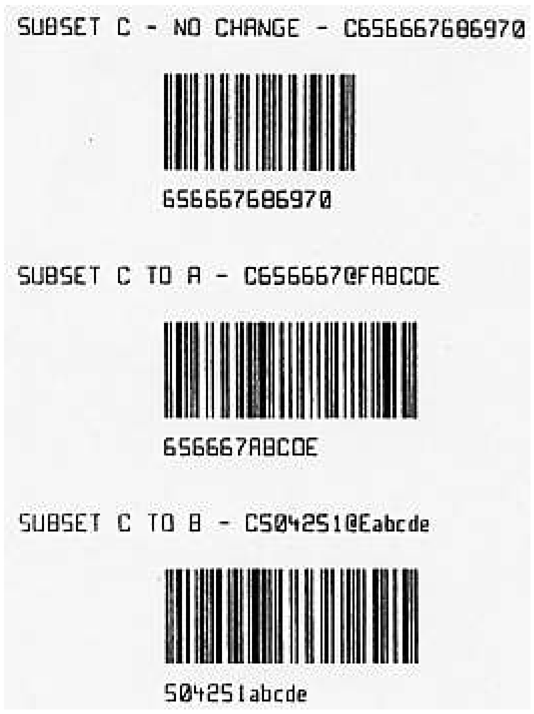
abcdefghijklm

nopqrstuvwxyz

abcdefghijklm

nopqsrtuvwxyz

```
~^"CODE128C";1;0;400;0;SPB;UTOF;4100;
HBR;200;VBR; 20;DHR;32768;DCH;12;DCW;22;ICS; 4;
"SUBSET C - NO CHANGE - C656667686970";
HBR; 50;VBR; 90;BDEF;25;BCSH;50;BCPI; 2;
BCST;"C656667686970";BSTP;
HBR;150;VBR;110;
"656667686970";
HBR;200;VBR;150;
"SUBSET C TO A - C656667@FABCDE";
HBR; 50;VBR;220;
BCST;"C656667@FABCDE";BSTP;
HBR;150;VBR;240;
"656667ABCDE";
HBR;200;VBR;280;
"SUBSET C TO B - C504251@Eabcde";
HBR; 50;VBR;350;
BCST;"C504251@Eabcde";BSTP;
HBR;150;VBR;370;
"504251abcde";
TRM;¥
```


```
SUBSET C - NO CHANGE - C656667686970

656667686970

SUBSET C TO A - C656667@FABCDE

656667ABCDE

SUBSET C TO B - C504251@Eabcde

504251abcde
```

7.8. Casecode Code128 (EAN–128)

Casecode Code128 is a distribution application using Code128. This section describes the specification and check character.
For more details of Casecode Code128, refer to "Application Specification for the UCC–128 Serial Shipping Container Code (With Symbol and Shipping Label Guideline)" from Uniform Council Inc.

7.8.1. Casecode Code128 Specification

As described below, the Casecode Code128 format comprises a start character C, function code 1, qualifier, data, two check characters, and a stop character.

The UCC-128 symbols included in the Serial Shipping Container Code use code subset C and high-density printing to reduce the size of the bar code. Then by following the start character C immediately by a function code 1, this bar code is indicated as a UCC-128 application.

The qualifier is a two-digit number, indicating which UCC-128 application is being used. The code for a serial shipping container (standard carton ID) is 00.

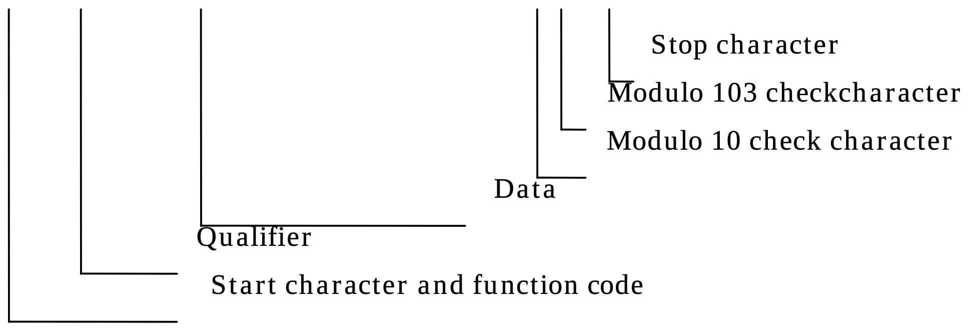
After the qualifier, a 17-digit numeric value is required. This number is further divided into three sections. The first digit indicates the packing type, the next seven digits are the UCC Manufacturer ID, and the remaining nine digits are the shipping container serial number.

UCC-128 requires two check characters: modulo 10 and modulo 103.
The modulo 10 check character is calculated from 19 digits: the data plus qualifier. Then the 20-digit value formed by this 19-digit value plus the modulo 10 check character is printed as a 10-digit high-density code. The modulo 103 check character is calculated according to the Code128 bar code specification.

After the modulo 103 check character a stop character is required.

Example: UCC-128 data

START-C/FNC1 00 00012345555555555 8 C STOP


Stop character
Modulo 103 checkcharacter
Modulo 10 check character
Data
Qualifier
Start character and function code

Modulo 10 check character

The UCC-128 modulo 10 check character is different from the CodaBar modulo 10 check character, in that 3 is used as the weighting factor.
The UCC-128 modulo 10 check character is computed as follows.

Step 1: Number the digits of the qualifier and 19-digit data from the end.

Step 2: Add the values of odd-numbered digits multiplied by 3.

Step 3: Add the values of even-numbered digits.

Step 4: Take the total of the values in steps 2 and 3.

Step 5: Subtract the total arrived at in step 4 from the closest larger multiple of 10.
This is the check character C.

Example:

Input data: "0000012345555555555"

Digit position	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Value	0	0	0	0	0	1	2	3	4	5	5	5	5	5	5	5	5	5	5

```
        0 + 0 + 0 + 2 + 4 + 5 + 5 + 5 + 5 + 5  = 31 x 3 = 93
            0 + 0 + 1 + 3 + 5 + 5 + 5 + 5 + 5           = 29
                                               ─────────────
                                                 Total  = 122
C = 130 – 122 = 8   ←   modulo 10 check character
```

7.8.2. Examples Using Casecode Code128

The method of printing a Casecode Code128 bar code is the same as for a Code128 bar code, except for the different number of digits. In other words, the five commands, BSYM or BDEF, BCST, BSTP, BCPI, and BCSH are required. For the relation between the density of the printed bar code and the BCPI parameter, see Section 7.7.1.

When Casecode Code128 is selected, the start character, function code 1, modulo 10 and modulo 103 check characters, and stop character are automatically added to the input data. Enter the 19-digit data for the UCC-128 symbol code excluding these values. If a value of less than 19 digits is entered, it is filled out to 19 digits with trailing zeros. If the value is longer than 19 digits, only the first 19 digits are used, and the rest are ignored. If there are any characters other than numeric digits in the data, this results in a command error.

To print a human-readable form, this must be specified separately from the bar code. In RCL there is a function for printing Casecode Code128 readable characters, with a modulo 10 check character added to the character string.
For more details, see the font selection tables in Chapter 21.

```
~^"CASECODE";1;0;400;0;SPB;UTOF;4100;
HBR;150;VBR;200;BDEF;27;BCSH;50;BCPI; 3;
BCST;"0000052177177444130";BSTP;
HBR;150;VBR;230;DHR;$12;DCH;15;DCW;30;ICS;10;
"0000052177177444130";
TRM;¥
```

00000521771774441303

```
~^"CASECODE";1;0;400;0;SPB;UTOF;4100;
HBR;300;VBR; 20;BDEF;28;BCSH;100;BCPI; 3;
BCST;"0000012345555555555";BSTP;
HBR;270;VBR; 20;DHR;$612;DCH;15;DCW;30;ICS;5;
"0000012345555555555";
TRM;¥
```

00000123455555555558

8.Verify Function

This chapter describes the commands relating to verifiers for printers equipped with them.

A printer equipped with a verifier checks bar codes after printing them, and if a defect is found, cancels the bar code by printing a "void" mark, and reprints it.
By carrying out bar code checking, the APPC (Automatic Print Process Control) function for printing energy optimization is achieved.

Note that these commands are only valid for a printer with a verifier, and are ignored by other printers.

For more details of the verify function, refer to the user documentation.

8.1. Verifier Control Commands

8.1.1. Verifier Mode (VFYM)

The VFYM command sets the number of times the bar code must be read correctly, and also the verify gap.

The syntax of this command is as follows.

VFYM;*XX*;*YY*;

XX＝verify mode

0: One correct reading out of two verify operations
1: Two correct readings out of two verify operations
2: Three correct readings out of three verify operations

YY＝verify gap

Specifies the start position for the second and third verify attempts.
The units depend on the LF pitch; refer to the printer specification for details.

・The first scan position is in the range 0.0315 to 0.0551”
(0.8 to 1.4 mm).

Note
Setting the verify mode to 1 or 2 will increase the proportion of void bar codes.
Normally use verify mode 0.
If the VFYM command is not issued, the default is verify mode 0, verify gap 0.020”(0.5 mm).

8.1.2. APpc OFf (APOF)

The APOF command stops the verifier, so that individual bar codes are not checked. Once the APOF command is issued, printed bar codes are not verified until the APON command is issued. In other words, it is possible to select particular bar codes within a label to be verified.

By checking the printed bar codes, printers with a verifier can optimize the printing energy. If the APOF command is used for all bar codes, the printing energy is controlled only by the temperature of the printing head.

8.1.3. APpc ON (APON)

The APON command activates the verifier. Once the APON command is issued, printed bar codes are verified until the APOF command is issued. When the printer is powered on, the default state is APPC on.

The syntax of the APOF and APON commands is as follows.

```
HBR;10;VBR;100;BCST;"*123456*";BSTP;#VERIFICATION
ENABLED#;HBR;10;VBR;200; APOF;BCST;"*123456*";BSTP;
APON;#DISABLE VERIFICATION#;
```

8.1.4. Disable Verification (DVFY)

The DVFY command disables verification for all bar codes on the labels. The difference from the APOF function, is that APOF actually stops the operation of the verifier mechanism, whereas with the DVFY function the bar codes are scanned and checked, but no "void" marks are printed, and there is no reprinting. The scan result is still, however, used to control the printing energy.

> Note
> It is also possible to disable the verify function with the printer function settings. Normally, instead of using the APON, APOF, and DVFY commands, use the printer function settings to achieve the same result. (The printer function settings take precedence over the commands.)

8.1.5. set Assured Quality Level (AQL)

The AQL command determines the tolerances for the elements of a bar code in the verification process. The tolerance value is the value set in the printer software plus the value specified by this command in the range 0.0005”(0.0125 mm) to 0.0035”(0.0875 mm).

The syntax of this command is as follows.

AQL;*X*;
    X = integer from 0 to 7

The setting is the minimum unit of 0.0005”(0.0125 mm) multiplied by X.

The value set by the printer software is the same as the value laid down by the standard for the particular type of bar code, and in almost all cases a setting AQL;0; will be adequate. However, if because of the ribbon or label quality the ink transfer is suboptimal, or if when printing high density bar codes too many "void" marks are printed, then increasing the tolerance will reduce the error rate. Even so, the labels printed will generally be adequate for reading by general-purpose bar code readers.

The last AQL setting in the command sequence takes effect. In other words, if there is more than one AQL command in a command sequence, only the last one is acted on.

If a value for the AQL command parameter of 8 or more is specified, correct operation of the printer cannot be guaranteed. Avoid using a value of 8 or more.

8.1.6. NO RePrint (NORP)

The NORP command disables the reprinting of labels after recovery from an error causing a label to be aborted.
When the printer receives this command, if an error occurs during printing ("Supply empty" or similar), after recovery from the error printing resumes from the label after the one which was interrupted.
This does not affect verify errors, which cause printing of a "void" mark and reprinting as normal.
The setting of this command is valid during printing of the specified format only.
The syntax of this command is as follows (insert the NORP command after the SPB and UTOF commands).

```
~^"SAMPLE1";1;0;24;187;
SPB;
UTOF;0354;
NORP;
MRK;HBR;0;VBR;0;DWBX;0;0;398;24;NUM;BCLC;1;
～
RET;
TRM;¥
```

Note

- If the NORP command is used, there may be labels for which no verify check was carried out, missing label numbers, or shortages in the number of labels printed.
- When the NORP command is not used, the completion of printing is taken as the end of verifying data, and therefore if an error occurs ("Supply empty" or similar) before the end of verification, after recovery from the error the same data will be reprinted.

8.2. Half Dots

A bar code can be printed using half dot control. Half dot control means that by energizing the thermal head with more energy than normal the ink around the thermal element is also melted, forming a thicker bar, on in the reverse direction, by applying less energy than normal a thinner bar can be printed. This does not function with heat-sensitive paper, as there is then no ink ribbon.

For UPC/EAN, the software automatically determines whether half dot control printing is necessary, and makes the appropriate setting. Programmer specification of half dot control is only available for picket fence Code39 and picket fence ITF bar codes.

Normally it is preferable to print bar codes without using this function. Half dot control may be useful in particular cases, for example when printing at a particular density.

8.2.1. Half Dot On (HALF)

When printing normal label stock using a thermal transfer ribbon, use this command to make the printed bars wider. The syntax of this command is as follows.

```
HALF;
```

The HALF command must appear before the beginning of the bar code to be printed with half dot control. Half dot control remains in effect until the HOFF (half dot off) command described below is issued. The HALF command has three effects:

(1)The ink width is increased by about 1/4 dot on each side of a bar, thus adding 1/2 dot in total to the bar width. For example, a bar 2 dots wide becomes 2.5 dots wide.

(2)A further white half dot is added to all spaces in the bar code. For example, a space 2 dots wide is increased to 2.5 dots.

(3)A new bar code error tolerance is assigned from the table, so that the verifier operates correctly on bar codes printed with the half dot function.

8.2.2. Half Dot On Bar (HLFB)

Use this command when printing with a thermal transfer ribbon on label stock on which the ink does not spread easily, to print with the bars narrower. The syntax of this command is as follows.

```
HLFB;
```

The HLFB command must appear before the beginning of the bar code to be printed with half dot control. Half dot control remains in effect until the HOFF (half dot off) command described below is issued. The HLFB command has the same three effects as the HALF command.

8.2.3. Half Dot Off (HOFF)

The HOFF command turns the half dot function off. The syntax of this command is as follows.

```
HOFF;
```

The HOFF command must be specified after the last label for which the half dot function is to be applied. If the HALF command is not executed, the half dot function does not take effect. (Default is HOFF.)

It is not possible to combine HALF and HOFF bar codes. Even if the HALF function is only required for one bar code, half dot processing must be carried out for all bar codes. It is particularly important that for UPC/EAN bar codes at a particular magnification half dot processing is carried out automatically, and is not carried out in other cases.

Note

Do not use the half dot function except on printers with a built-in verifier.

8.3. Bar Codes Subject to Verification

A maximum of 15 bar codes in each label can be verified by the printer. However, this excludes bar codes for which verification is disabled with the APOF command.

For scanning with the verifier, there are restrictions on the height of the bar code (BCSH). Refer to the specification for the particular model.

The printer can scan bar codes, but cannot determine whether the readable text and bar code actually agree.

A printer with a verifier cannot verify ladder bar codes.

9.Two-Dimensional Symbols

Of the common two-dimensional symbols, RCL supports QR Code, and Code49. Code49 can be printed in four orientations, at 0, 90, 180, and 270 degrees. The orientations at 0 degrees and 180 degrees have the bar code elements parallel to the label feed direction, and are referred to as "picket fence" bar codes. The orientations at 90 degrees and 270 degrees have the bar code elements at right angles to the label feed direction, and are referred to as "ladder" bar codes. RCL has a number of commands to assist with printing two-dimensional symbols. Any two-dimensional symbol requires at least three commands.
These are two-dimensional bar code symbol selection (BSYM), bar code start (BCST), and bar code stop (BSTP).

Further, each type of two-dimensional symbol requires its own special commands.
QR Code uses three commands, for setting cell width, error recovery level, and master pattern.
Code49 requires bar code density and bar code height settings.

9.1. Selecting Two-Dimensional Symbols

To create a two-dimensional symbol, first the appropriate type of two-dimensional symbol must be selected. The two-dimensional symbol selection command is used to select the type of two-dimensional symbol used.

9.1.1. Types of Two-Dimensional Symbols

RCL supports two types of two-dimensional symbol: QR code and Code49.

Note
It is the programmer's responsibility to investigate the latest specifications for the two-dimensional symbol type being used, and to follow guidelines specified.

9.1.2. Two-Dimensional Symbol Selection (BSYM, BDEF)

The BSYM command is used to select the type of two-dimensional symbol to be printed. The type of two-dimensional symbol must be selected before creating a two-dimensional symbol.

The syntax of this command is as follows.

BSYM;*A*;*B*;

A: integer specifying two-dimensional symbol type, as shown in table below
B: integer 1–4, specifying two-dimensional symbol orientation
(For QR code, B has no significance)

The following table shows the BSYM command parameter values for each two-dimensional symbol type.

Two-dimensional symbol type	A
QR Code model 1	100
QR Code model 2	101
Micro QR Code	102
Code49	60

Orientation	B
0 degrees	1
90 degrees	2
180 degrees	3
270 degrees	4

The BDEF and BSYM commands are similar. However, the BDEF command only supports orientations of 0 degrees and 270 degrees. (For QR code, 0 degrees only) The type of two-dimensional symbol must be selected before creating a two-dimensional symbol.

The syntax of this command is as follows.

```
BDEF;A;
```

A = integer 60, 61 or 100, 101, 102, from table below

The following table shows the BDEF command parameter value for each two-dimensional symbol type.

Two-dimensional symbol type	A
Code49	60
Code49 ladder	61
QR Code model 1	100
QR Code model 2	101
Micro QR	102

Once a BDEF command or BSYM command has been issued to select a two-dimensional symbol type, this type remains selected until a new selection is made.

9.2. Two-Dimensional Symbol Printing

For two-dimensional symbols, as for bar codes, use the HBR, VBR, HPR, VPR, HOME, and EOL commands to position the printing. Code49 symbols are printed with the lower left corner at the cursor position. In other words, a picket fence symbol in the 0 degrees orientation is printed to extend up and to the right from the cursor position, and a ladder symbol in the 270 degrees orientation is printed to extend right and down from the cursor position. When the printing is completed, the base position for the next object is the lower right corner of a picket fence symbol and the lower left corner of a ladder symbol.

For QR code, the origin is the upper left corner of the printing area. In other words the symbol is printed to extend down and to the right from the cursor position.

9.2.1. Symbol Data Entry

The symbol data must always be included in double quotation marks ("...").
Code49 allows the entire ASCII character set (128 characters) to be used as data. Additionally, special FUNC1, FUNC2, and FUNC3 codes can be used.
The input of these is described elsewhere.
Again, with Code49, each row of the symbol has a modulo 49 check character, and in the last row between four and six check characters are added.
QR Code allows alphanumerics, binary, and kanji data to be entered. The input of these is described elsewhere.

For both Code49 and QR Code, readable characters must be printed separately with the text command.
In this case, ensure that the symbol and readable characters contain the same data

9.2.2. Symbol Data Start (BCST)

The BCST (Bar Code STart) command instructs the printer to print the following data as a two-dimensional symbol. If the BCST command is omitted, the data will be treated as ordinary text, and printed in text form.

The syntax of this command is as follows.

```
BCST;
```

9.2.3. Symbol Data Stop (BSTP)

The BSTP (Bar Code StoP) command forms a pair with the BCST command, and marks the end of the two-dimensional symbol data. This command must immediately follow the last character of the bar code data. If the BSTP command is omitted, the printer cannot tell where the end of the two-dimensional symbol is.

The syntax of this command is as follows.

```
BSTP;
```

The following example selects a Code49 symbol, and enters the data.

```
BSYM;60;1;
BCST;"CODE49 BARCODE";BSTP;
```

9.3. Code49

Code49 uses DTDS or BSYM/BDEF to define the type of symbol, and BCST and BSTP to start and end the symbol data, then to print the symbol it requires the symbol density (BCPI) and symbol height (BCSH) to be set.

9.3.1. Code49 Bar Code Characters Per Inch (BCPI)

The BCPI command specifies the number of characters per inch in a Code49 symbol. This command automatically sets the widths of elements in the Code49 symbol.

The syntax of this command is as follows.

BCPI;*A*;

A = integer from following table specifying density (cpi)
The density setting values are listed in the tables in Chapter 19.

9.3.2. Code49 Symbol Height (BCSH)

The BCSH command sets the height of one row of the Code49 symbol.
The height of the symbol separator bar is derived from the modular dimension specified by the BCPI command. In other words, the overall height H of the symbol is given by the following expression:

$H = ((h+1)*R+1)*X$
H = symbol height
h is the height is the bars in module multiples (integer).
R is the number of rows.
Note: separator bar is taken as X modules.

The syntax of this command is as follows.

・For a picket fence symbol

BCSH;*YY*;
YY = symbol height in 0.010inches(0.254 mm) units

・For a ladder symbol

BCSH;*XX*;
XX = symbol height in 0.005inches(0.127 mm) units

For Code49, the minimum recommended symbol height is 8X, where X is the module dimension.
The default value is 50.

9.3.3. Examples Using Code49

Code49 allows a bar code to be printed including all 128 characters of the ASCII character set, and function codes FUNC1, FUNC2, and FUNC3.
In Code49, when five or more numeric digits appear in a sequence, they are compressed and encoded in a numeric data mode.
Code49 provides from two to eight bar code rows, divided by separator bars. Each row has a row check character added.

Again, in the last row between four and six check characters for the whole symbol are added.

Note
Except for NULL (00_H), ENQ (05_H), DC2 (12_H), DC4 (14_H), and CAN (18_H), all control characters can be printed in a bar code.

Entering function codes
The following table shows the character sequences used to enter the special function codes FUNC1, FUNC2, and FUNC3.

Data code	Input character sequence
@	@@
FUNC1	@A
FUNC2	@B
FUNC3	@C

If an '@' character appears in a combination not shown in the table, both the '@' character and the following character are treated as data characters.
In other words, both '@@D' and '@D' result in the two-character data sequence '@D'.

The following example shows bar codes which can be printed with Code49.

```
~^"CODE49";1;0;120;0;
SPB;HBR;0;VBR;90;
BSYM;60;1;BCPI;5;BCSH;16;
BCST;"RCL Ver.4.00";BSTP;
HBR;0;VBR;95;DDF;3;1;DFM;2;2;
"RCL Ver.4.00";
TRM;¥
```

RCL PLUS VERSION 4.00

9.4. QR Code

For QR Code symbols, in addition to the symbol selection commands DTDS and BSYM/BDEF, and symbol start/stop commands BCST and BSTP, the QR Code

symbol cell size setting command QRCS, QR error recovery level command QREL or TDEL, and QR mask pattern setting command QRMP are required.
Also note that the QR Code image expansion is different from other bar codes, in that the origin is not at the lower left corner but the upper left corner.

9.4.1. QR Code Cell Size (QRCS)

The QRCS command sets the size of a square cell in a QR Code.
The syntax of this command is as follows.

```
QRCS;X;
```

X = integer specifying number of dots in a cell
The default value is 2.

9.4.2. QR Code Error Recovery Level (QREL)

Error recovery level settings are provided for QR Code by using QREL and TDEL command.

The QREL command is available for only QR Code Model 1.
The TDEL command is available for QR Code Model 1 and 2.

The QREL command carries out error recovery level settings.
Four levels for error recovery ratios are provided.
The higher the error recovery level, the more data is required for recovery.
The syntax of this command is as follows.

```
QREL;L;
```

L = error level from the following table

L	Error recovery level
1	High density level (recovery level L) approx. 7% recovery
2	Standard level (recovery level M) approx. 15% recovery
3	High reliability level (recovery level Q) approx. 25% recovery
4	Very high reliability level (recovery level H) approx. 30% recovery

If the setting value is not a value in this table, the default of 3 applies.

9.4.3. Two-Dimensional Error Recovery Level (TDEL)

The TDEL command carries out error recovery level settings.
The following four error recovery levels are provided.
The higher the error recovery level, the more data is required for recovery.
The syntax of this command is as follows.

```
TDEL;L;
```

L = error level from the following table

L	Error recovery level
1	High density level (recovery level L) approx. 7% recovery

2	Standard level (recovery level M) approx. 15% recovery
3	High reliability level (recovery level Q) approx. 25% recovery
4	Very high reliability level (recovery level H) approx. 30% recovery

If the setting value is not a value in this table, the default of 4 applies.

9.4.4. QR Code Mask Pattern (QRMP)

The QRMP command selects a mask pattern for the symbol. In a QR Code, for accurate reading, the black and white cells must be laid out in a proper balance; masking is applied to achieve this and also as far as possible to avoid the characteristic position-detection pattern "1011101" from occurring within the symbol. There are eight masking patterns, numbered from 0 to 7.

The syntax of this command is as follows.

```
QRMP;X;
```

X = specified as follows (integer 0 to 9)

X: is mask pattern
0–7: select mask 0 to 7
8: no mask
9: automatic selection

If the setting value is not one of the above values, the default of 9 applies.

When automatic selection is used, the printer compares the number of continuous same-color patterns, same-color blocks, 1011101 patterns, and overall black cell ration for each of the eight mask patterns, and applies a system of penalty points to determine the pattern giving the best score.
If a mask number from 0 to 7 is specified, the corresponding mask is used.
If 8 is specified, no masking is carried out.

Note
Unless there is a particular reason for doing otherwise, use the automatic selection setting.

9.4.5. Entry of QR Code Printing Data

The following character modes are available for entering data in a QR Code.

・Numeric mode (N):	decimal digits only (0–9)
・Alphanumeric mode (A):	decimal digits, capital letters, and 9 other characters (total 45)
・Binary mode (B):	JIS 8 unit code
・Kanji mode (K):	data including kanji in shift-JIS code

It is also possible to combine these modes.

When printing data is entered, an identifier must be added at the beginning to indicate the input mode.
The printer uses this mode identifier to determine how to encode the data.
To combine modes, use a comma as a separator between different modes.

```
Example: BCST;"N1234567,AABCD01,B0002ab";BSTP;
```

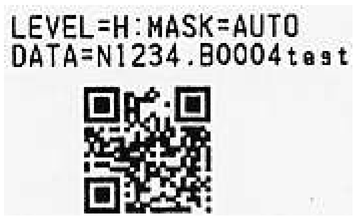
The 'B' identifier is followed by a four-digit number indicating the number of binary characters.
In this case "1234567" is data in numeric mode, "ABCD01" is data in alphanumeric mode, and "ab" is data in binary mode.
QR Code allows numeric, alphanumeric, binary, and kanji data to be combined.

9.4.6. Examples Using QR Code

QR Code allows alphanumeric, binary, and kanji data to be printed.
QR Code has 22 versions, from 1 to 22, according to the number of cells.
The printer selects the version printed to accommodate the number of data values and the error correction level.

The following shows examples of QR Code printing.

```
~^"QR CODE";1;0;150;200;
SPB;UTOF;1500;HBR;0;VBR;0;
DDF;8;10;DFM;2;2;
HBR;5;VBR;10;"LEVEL=H: MASK=AUTO";
HBR;5;VBR;25;"DATA=N1234,B0004test";
BDEF;100;QREL;3;QRCS;8;QRMP;9;
HBR;80;VBR;50;
BCST;"N1234,B0004test";BSTP;
TRM;
```



10.Text and Readable Characters

RCL has a function to add a check character to text data in the same way as for bar code data.

This section describes commands controlling readable characters for text and bar codes, with separate descriptions for vector fonts, dot fonts, and outline fonts.

10.1. Vector Fonts

10.1.1. Text Command Syntax

To specify text using a vector font, there are four commands: DHR (Define Human Readable) to specify a vector font, DCH (Define Character Height), DCW (Define Character Width), and ICS (Inter-Character Space).
The syntax of these four commands is as follows.

DHR;*AA*;DCH;*YY*;DCW;*XX*;ICS;*XX*;

AA = decimal (hexadecimal specification also possible: see Section 10.1.4, "Vector Font Specification")
YY = number of vertical pixels (0.010inches(0.254 mm))
XX = number of horizontal pixels (0.005inches(0.127 mm))

If none of these commands appears in the command sequence, the default values are used.
The default values of each command are as follows:

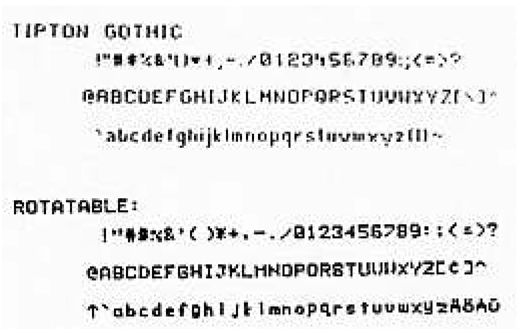
```
DHR;32768;DCH;20;DCW;40;ICS;4;
```

10.1.2. Vector Fonts

There are three vector fonts: Tipton Gothic proportional, Tipton Gothic monospaced, and rotatable. Tipton Gothic also provides bold, italic , condensed (80% of normal) and expanded (120% of normal) styles. The rotatable font can be printed in eight orientations, and supports seven European language character sets.

10.1.3. Character Set

Tipton Gothic has a standard 96-character ASCII character set. Characters from ASCII space (32) to delete (127) can be printed.
The rotatable font uses the standard ASCII character set, except that different multinational characters can be printed in place of six punctuation characters (_, {, }, ~, |, ¥). These two character sets are shown below.


TIPTON GOTHIC
ROTATABLE:

10.1.4. Vector Font Specification (Define Human Readable: DHR)

The DHR command specifies a vector font, and also the style and text orientation.

The font specification may be in decimal or hexadecimal notation. The syntax of this command is as follows.

```
DHR;AAAA;   or   DHR;$HHHH;
 AAAA = decimal   HHHH = hexadecimal
  $ = hexadecimal sign
```

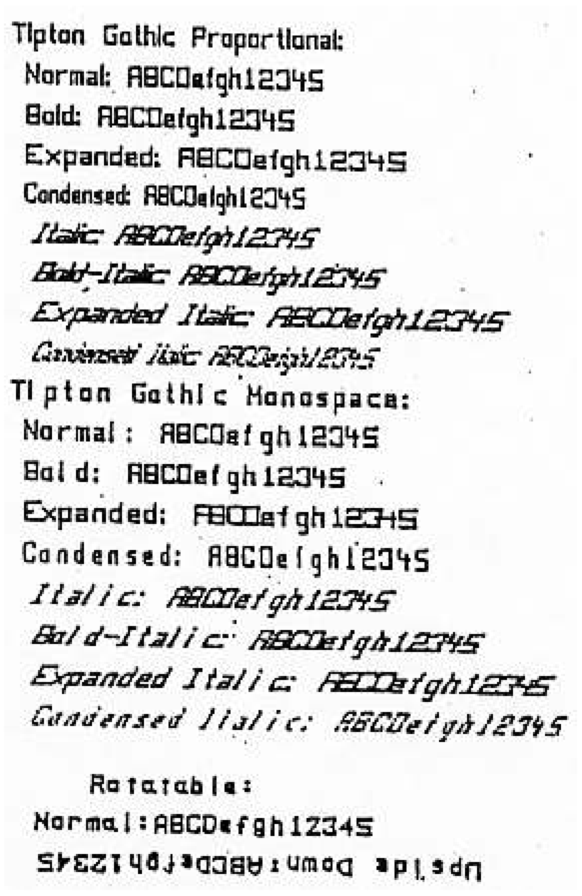
The following table shows the hexadecimal and decimal values for the vector font rotatable font style and orientation.

Font name	Style	Hexadecimal value	Decimal value
Tipton Gothic proportional		$8000	32768
Tipton Gothic proportional	Bold	$C000	49152
Tipton Gothic proportional	Expanded	$8800	34816
Tipton Gothic proportional	Condensed	$9000	36864
Tipton Gothic proportional	Italic	$2000	8192
Tipton Gothic proportional	Bold italic	$4000	16384
Tipton Gothic proportional	Expanded italic	$0800	2048
Tipton Gothic proportional	Condensed italic	$1000	4096
Tipton Gothic monospaced		$8002	32770
Tipton Gothic monospaced	Bold	$C002	49154
Tipton Gothic monospaced	Expanded	$8802	34818
Tipton Gothic monospaced	Condensed	$9002	36866
Tipton Gothic monospaced	Italic	$0002	2
Tipton Gothic monospaced	Bold italic	$4002	16386
Tipton Gothic monospaced	Expanded italic	$0802	2050
Tipton Gothic monospaced	Condensed italic	$1002	4098

Font name	Style (orientation)	Hexadecimal value	Decimal value
Rotatable		$1	1
Rotatable 45x	Left to right	$101	257
Rotatable 90x	Bottom to top	$201	513
Rotatable 135x	Right to left	$301	769
Rotatable 180x	Vertically inverted	$401	1025
Rotatable 225x	Right to left	$501	1281
Rotatable 270x	Top to bottom	$601	1537
Rotatable 315x	Left to right	$701	1793
Rotatable with mod 10/11			
Rotatable		$10	16
Rotatable 90x	Bottom to top	$210	528
Rotatable 180x	Vertically inverted	$410	1040
Rotatable 270x	Top to bottom	$610	1552
Rotatable with mod 43			
Rotatable		$11	17
Rotatable 90x	Bottom to top	$211	529
Rotatable 180x	Vertically inverted	$411	1041
Rotatable 270x	Top to bottom	$611	1553
Rotatable with mod 10 (Readable Casecode Code 128)			
Rotatable		$12	18
Rotatable 90x	Bottom to top	$212	530
Rotatable 180x	Vertically inverted	$412	1042
Rotatable 270x	Top to bottom	$612	1554
Multinational character sets			
English character set		$3	3
French character set		$4	4
Swedish/Finnish char. set		$5	5
Danish character set		$6	6
Italian character set		$7	7
German character set		$8	8
Spanish character set		$9	9

Note
Indications "mod 10", "mod 11", "mod 43", refer to versions with check characters modulo 10, 11, and 43 respectively.

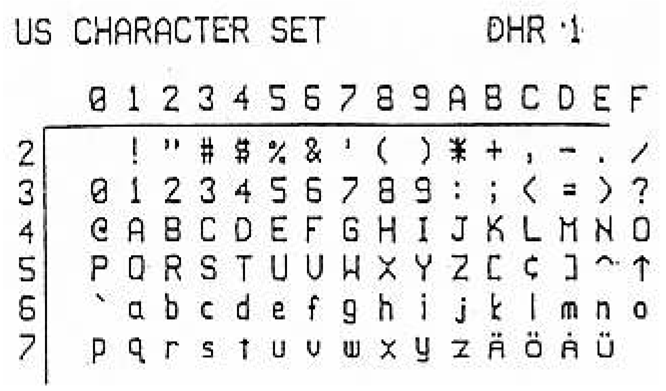
This shows various vector fonts and styles which can be printed by RCL.



Tipton Gothic Proportional:
Normal: ABCDefgh12345
Bold: ABCDefgh12345
Expanded: ABCDefgh12345
Condensed: ABCDefgh12345
Italic: ABCDefgh12345
Bold-Italic: ABCDefgh12345
Expanded Italic: ABCDefgh12345
Condensed Italic: ABCDefgh12345
Tipton Gothic Monospace:
Normal: ABCDefgh12345
Bold: ABCDefgh12345
Expanded: ABCDefgh12345
Condensed: ABCDefgh12345
Italic: ABCDefgh12345
Bold-Italic: ABCDefgh12345
Expanded Italic: ABCDefgh12345
Condensed Italic: ABCDefgh12345

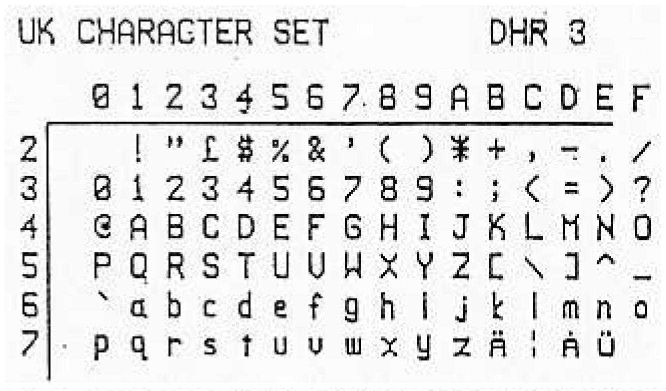
Rotatable:
Normal: ABCDefgh12345
Upside Down: ABCDefgh12345

The following table lists the hexadecimal values in multinational character sets. (Example: 21_H = '!')



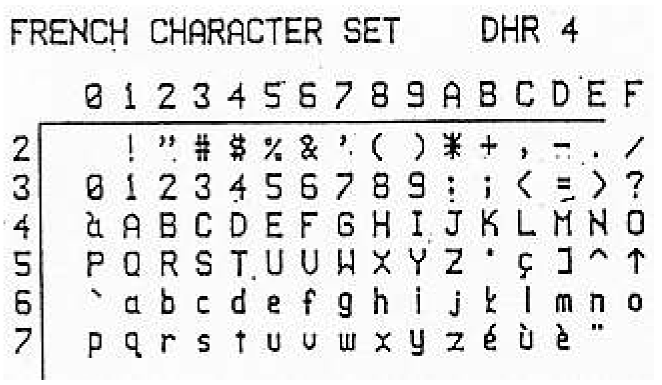
US CHARACTER SET DHR 1

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	¢	]	^	↑
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	Ä	Ö	Å	Ü	



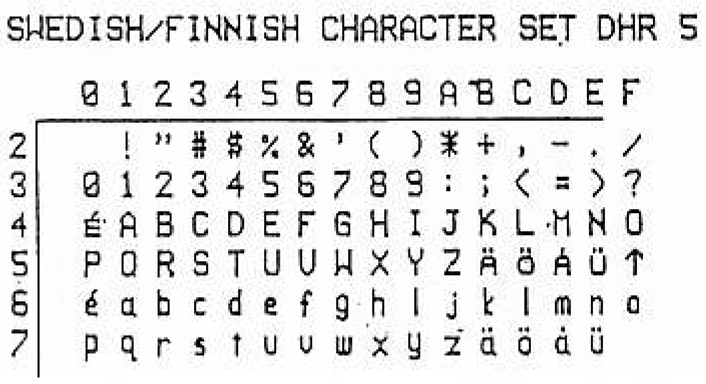
UK CHARACTER SET DHR 3

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	£	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	Ä	¦	Å	Ü	



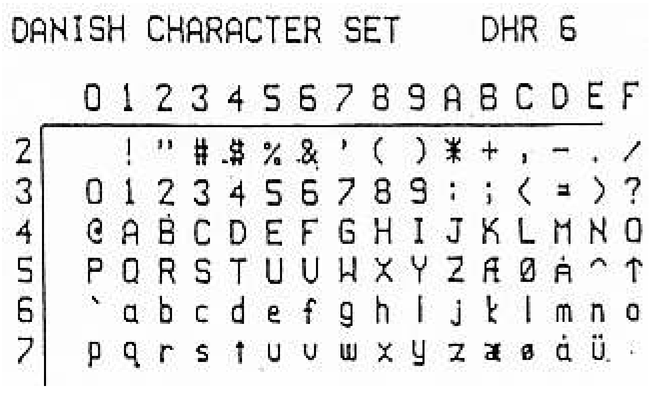
FRENCH CHARACTER SET DHR 4

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	à	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	°	ç	]	^	↑
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	é	ù	è	¨	



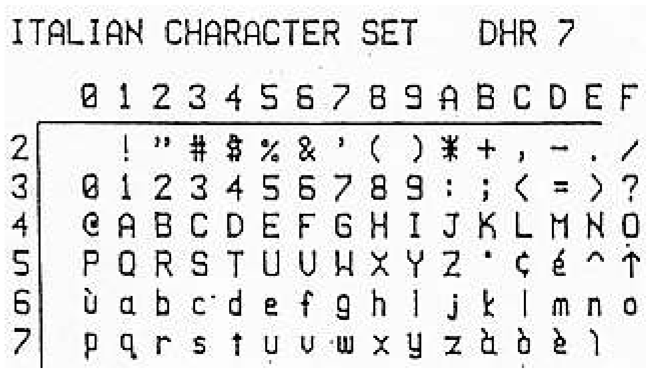
SWEDISH/FINNISH CHARACTER SET DHR 5

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	É	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	Ä	Ö	Å	Ü	↑
6	é	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	ä	ö	å	ü	



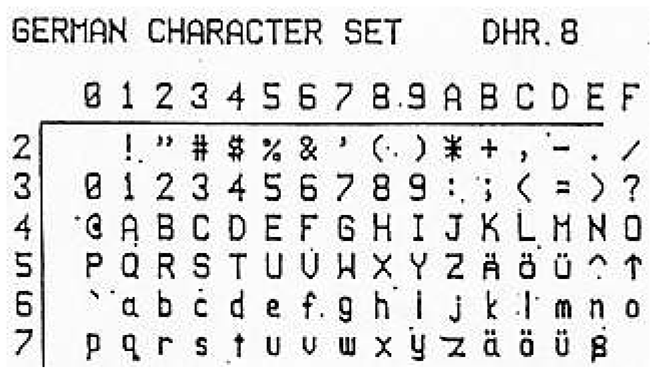
DANISH CHARACTER SET DHR 6

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	Æ	Ø	Å	^	↑
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	æ	ø	å	ü	



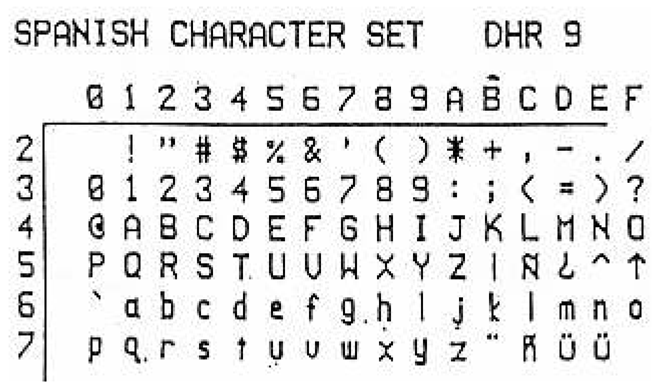
ITALIAN CHARACTER SET DHR 7

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	°	ç	é	^	↑
6	ù	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	à	ò	è	ì	



GERMAN CHARACTER SET DHR. 8

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	Ä	Ö	Ü	^	↑
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	ä	ö	ü	ß	



SPANISH CHARACTER SET DHR 9

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	¡	Ñ	¿	^	↑
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	¨	ñ	ü	Ü	

10.1.5. Character Size Selection

Two commands are used to select the character size: DCH (Define Character Height) and DCW (Define Character Width). There are variations from font to font, but the height setting is in the range 0.05inches(1.27 mm) to maximum printing length, and the width setting in the range 0.05inches(1.27 mm) to maximum printing width. The ICS command (Inter-Character Space) sets the space between characters from 0.010inches(0.254 mm) to 2.5inches(63.5 mm).
The syntax of these commands is as follows.

DCH;*YY*;DCW;*XX*;ICS;*XX*;
YY = number of vertical pixels (0.010inches(0.254 mm))
XX = number of horizontal pixels (0.005inches(0.127 mm))

10.1.5.1. Define Character Height (DCH)

The DCH command specifies the height of readable text using a vector font, in pixel units. Used together with the DHR and DCW commands, this command allows text to be printed in a variety of fonts and sizes. When Tipton Gothic is selected with the DHR command, the minimum character height is 10 pixels (0.1inches(2.54 mm)), and the maximum height is 330 pixels (3.3inches(83.8 mm)). If the rotatable font is selected, the minimum height is 5 pixels (0.05inches(1.27 mm)), and the maximum height is the maximum printing length.

Note
When using Tipton Gothic font condensed or italic, make the character size at least DCH;30;.

The size specified by the DCH command is the overall size, including descenders and ascenders. Capital letters and lowercase letters with ascenders ('d', 'l', etc.) occupy approximately the upper 70% of the height specified by DCH; lowercase letters with descenders ('p', 'y', etc.) occupy the lower 70%. Lowercase letters without ascenders or descenders ('a', 'c', etc.) occupy approximately 50% of the specified height.

10.1.5.2. Define Character Width (DCW)

The DCW command specifies the width of readable text characters, in units of one-half of a horizontal pixel (0.005inches(0.127 mm)). The width is thus given by the following expression:

Width (X pixels) = DCW/2

The minimum DCW value for the Tipton Gothic font is 20, and the maximum value is 1000. For the rotatable font, the minimum value 10, and the maximum value 1536.

Note

When using Tipton Gothic font condensed or italic, make the character size at least DCW;60;.

10.1.5.3. Inter-Character Space (ICS)

The ICS command specifies the space between characters in horizontal pixels (0.005inches(0.127 mm)). The minimum value is 2 pixels (0.010inches(0.254 mm)), and the maximum value 500 pixels (2.5inches(63.5 mm)). The default value is 4 (0.02inches(0.508 mm)). The value specified by the ICS command is added to the width of the character specified by the DCW command to determine the position of the next character. For the rotatable and Tipton Gothic monospaced fonts, the inter-character space is fixed, but the Tipton Gothic proportional font has variable spacing.

Note

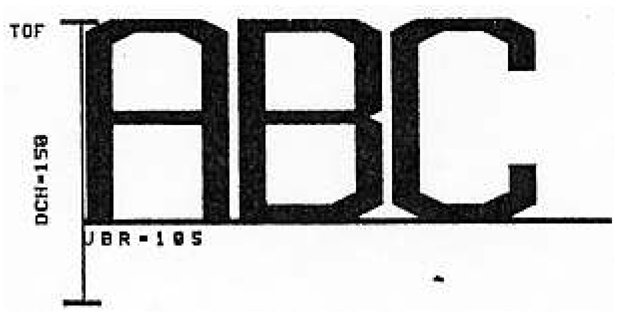
When using the Tipton Gothic monospaced font in condensed or expanded form, adjust the spacing with the ICS command, or the spacing will be inappropriate. For condensed printing halve the ICS value, for example from 4 to 2, and for expanded printing increase the ICS value from by three times, for example from 4 to 12.

Specify the ICS command value as an even number of pixels. If an odd number of pixels is specified, it is automatically rounded up to the next even number. In other words, specifying 3 is the same as specifying 4, and 9 the same as 10.

10.1.5.4. Text Positioning

Both absolute and relative positioning commands are used in positioning text and readable characters on a label.
For a vector font, the start position is the lower left corner of the first character. The text is printed in sequence from this point.
The following example shows the print start position as the lower left corner of a capital 'A' in a vector font.

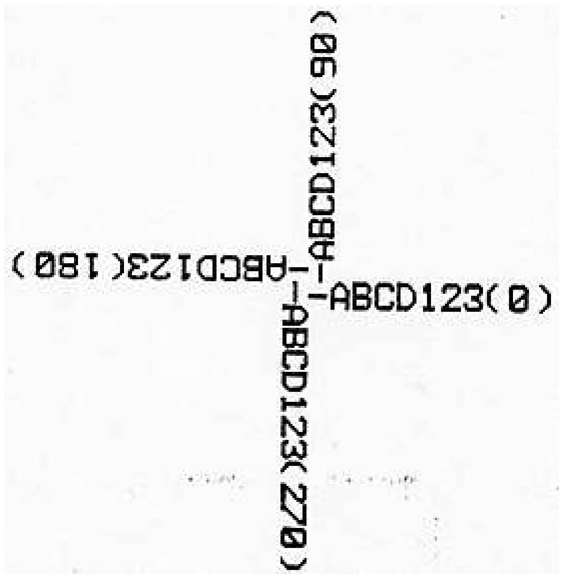
```
~^"POINT";1;0;160;0;SPB;UTOF;1600;
HBR; 50;VBR; 10;DHR; 1;DCH; 10;DCW; 20;ICS; 4;
"TOF";
HBR; 0;VBR; 0;DHL;100; 0; 40;
HBR; 0;VBR; 0;DVL;120; 0;150;
HBR; 0;VBR; 0;DHL;100;147; 40;
HBR; 75;VBR;115;DHR; $201;DCH; 10;DCW; 20;ICS; 4;
"DCH=150";
HBR;125;VBR;105;DHR;$8000;DCH;150;DCW;300;ICS;10;
"ABC";
HBR; 0;VBR; 0;DHL;120;105;550;EOL;
HBR;125;VBR;120;DHR; $1;DCH; 10;DCW; 20;ICS;10;
"VBR=105";
TRM;¥
```



10.1.5.5. Print Start Position for Rotated Text

When text is rotated with a rotatable font, the start position changes with the rotation. The print start position of the rotated text is as follows.

```
~^"KAITEN";1;0;350;0;SPB;UTOF;3600;
HBR;350;VBR;165;
DHR; $1;DCH;20;DCW;40;ICS; 4;        # orient 0 #;
"-ABCD123 (0) ";HOME;
DHR;$201;DCH;20;DCW;40;ICS; 4;       # orient 90 #;
"-ABCD123 (90) ";HOME;
DHR;$401;DCH;20;DCW;40;ICS; 4;       # orient 180 #;
"-ABCD123 (180) ";HOME;
DHR;$601;DCH;20;DCW;40;ICS; 4;       # orient 270 #;
"-ABCD123 (270) ";HOME;
TRM;¥
```



10.1.6. Entering Text Data

Specify the text enclosed in double quotation marks ("..."). Any number of characters can be specified in a text string. It is important, however, to make sure that the text will fit on the label. The following is an example of text input syntax.

```
DHR;$8000;DCH;20;DCW;40;ICS;4;
"Text String";
```

10.1.7. Rotatable Font with Check Characters

With the rotatable font the printer software can be used to calculate a check character check character and add it to text data. The check character can be modulo 10, modulo 11, or modulo 43.

10.1.7.1. Rotatable Font with Modulo 10 or Modulo 11 Check Characters

When the DHR parameter is specified as $10, $210, $410, or $610, either or both of the modulo 10 and modulo 11 check characters are calculated and added to the printed text. The positions are specified by @ and # characters.

@ indicates the modulo 11 check character, and # the modulo 10 check character.

Example:
DHR;$10;---;"@#5678";---

The result of this printing is as follows.

□ □ 5 6 7 8

modulo 10 check digit for data "5678"
modulo 11 check digit for data "5678"

This function is useful for CodaBar readable text.
The method of performing the modulo 10 and modulo 11 calculations is as described in Section 7.5.2.

10.1.7.2. Rotatable Font with Modulo 43 Check Character

When the DHR parameter is specified as $11, $211, $411, or $611, a modulo 43 check character is calculated and added to the end of the printed text.

Example:
DHR;$11;---;"ABC123"---

The result of this printing is as follows.

A B C 1 2 3 □

modulo 43 check character for data "ABC123"

This function is useful for Code39 modulo 43 readable text.
The method of performing the modulo 43 calculation is as described in Section 7.3.

10.1.7.3. Rotatable Font with Modulo 10 Check Character

(For Casecode Code 128 Readable Characters)

When the DHR parameter is specified as $12, $212, $412, or $612, a modulo 10 check character is calculated and added to the end of the printed text. This function is used for the Casecode Code 128 readable characters. The number of characters in the data is not checked, and the length of the data is the responsibility of the programmer.

The modulo 10 check character used for the Casecode Code 128 readable characters is calculated as follows.

Step 1:	Number the digits of the data from the least significant (1, 2, 3,...).
Step 2:	Add the values of odd-numbered digits, and multiply by 3.
Step 3:	Add the values of even-numbered digits.
Step 4:	Take the total of the values in steps 2 and 3.
Step 5:	Subtract the total arrived at in step 4 from the closest larger multiple of 10. This is the modulo 10 check character C.

Example:
Input data: "123541"

```
Digit position      6 5 4 3 2 1
Value               1 2 3 5 4 1
-------------------------------
                      2 + 5 + 1 = 8 x 3   = 24
                    1 + 3 + 4             =  8
                                          ----
                            Total           32
```

C = 40–32 = 8 ← modulo 10

The readable characters are therefore printed as "1235418".

10.2. Dot Fonts

10.2.1. Text Command Syntax

To specify text using a dot font, there are four commands: DDF (Define Dot Font), DFM (Dot Font Magnification), DFS (Dot Font Spacing), and DFO (Dot Font Orientation).
The syntax of these four commands is as follows.

```
DDF;A;B;DFM;XM;YM;DFO;O;K;DFS;X;
```

A = dot font identifier
B = country code
XM = horizontal magnification
YM = vertical magnification
O = text orientation
K = writing direction (vertical/horizontal)
X = inter-character space

If none of these commands appears in the command sequence, the default values are used.
The default values of each command are as follows:

```
DDF;3;1;DFM;1;1;DFO;1;1;DFS;2;
```

If a dot font is not specified, printing uses a vector font.

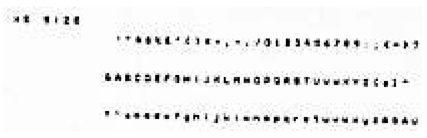
10.2.2. Fonts

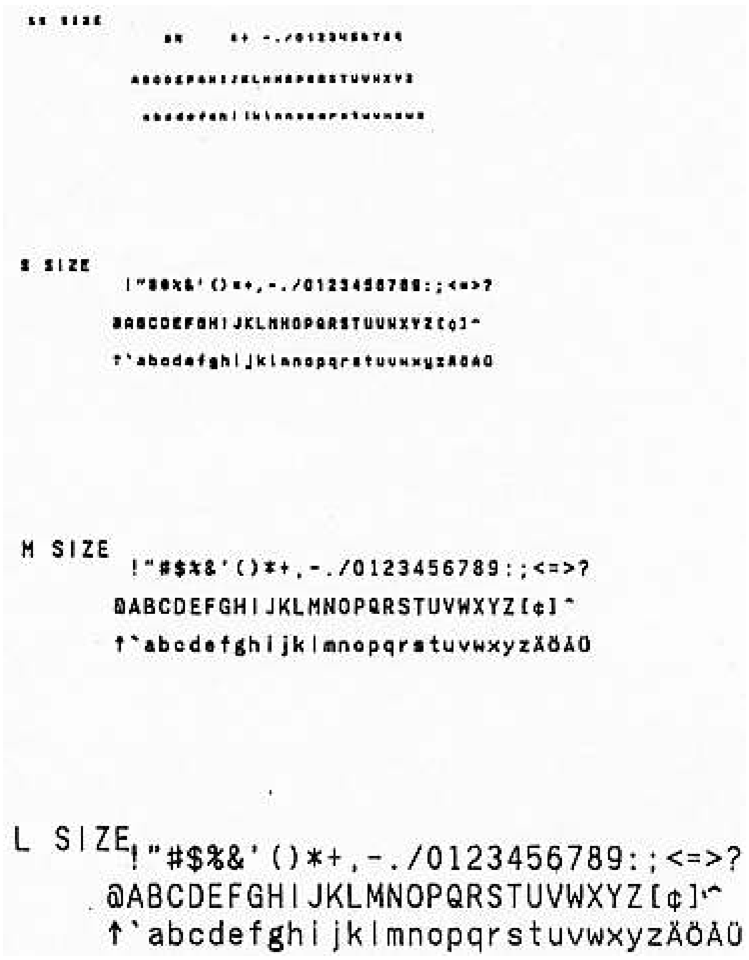
Dot fonts can be expanded, and the inter-character spacing adjusted.

The sizes of the dot fonts are listed in tables for each model, in Chapter 20.

10.2.3. Character Set

The alphanumeric dot fonts (XS, SS, S, M, L, and OCR-B sizes) have a standard 96-character ASCII character set. Characters from ASCII space (32) to delete (127) can be printed.
An example of printing with an alphanumeric dot font is shown below.





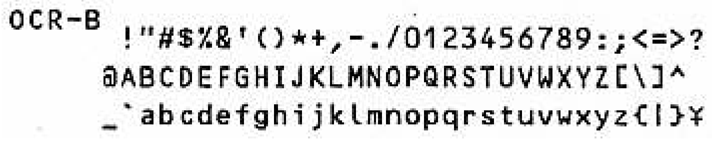
XS SIZE #% *+ -./0123456789
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

S SIZE !"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[¢]^
↑`abcdefghijklmnopqrstuvwxyzÄÖÅÜ

M SIZE !"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[¢]^
↑`abcdefghijklmnopqrstuvwxyzÄÖÅÜ

L SIZE !"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[¢]^
↑`abcdefghijklmnopqrstuvwxyzÄÖÅÜ

The OCR-B font allows all characters from the JIS-X9001 optical character recognition (alphanumeric) character forms in size I, subset 3 to be printed, excluding the currency symbol and separator. However, these characters are not guaranteed to be read by an optical character reader. The following is a printing Example:.



OCR-B !"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^
_`abcdefghijklmnopqrstuvwxyz{|}¥

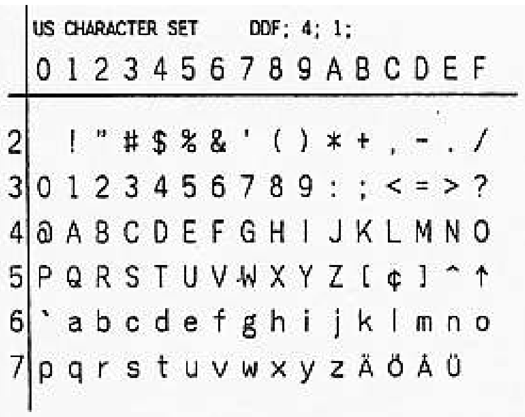
10.2.4. Define Dot Font (DDF)

The DDF command selects the dot font and specifies the country code. The fonts are listed in the selection tables in Chapter 21.

When an alphanumeric font is specified, the country code can be used to specify variant character sets for different currency symbols or accented letters. Selecting the country code for Japan provides the graphic characters specified for Roman letters by the JIS-X0201 information exchange standard code.

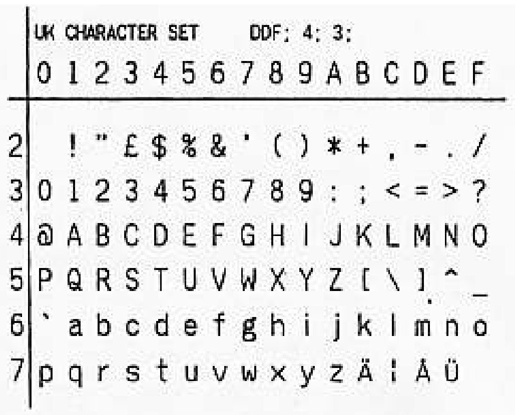
When OCR-B font is specified, specify the country code as 10 (Japan). For an alphanumeric font, when the country code is 10 (Japan), the printed codes and fonts are as follows.

The following table lists the values for each country character set in hexadecimal.
(E.g. 21_H = !)



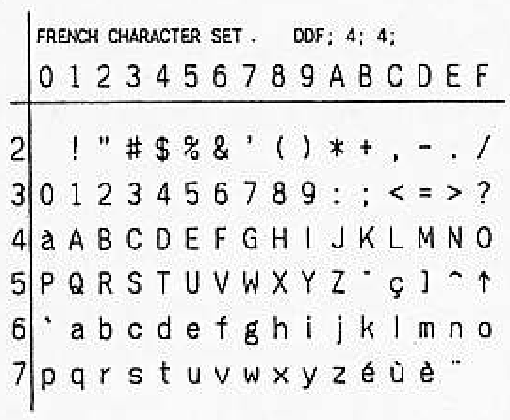
US CHARACTER SET DDF; 4; 1;

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	¢	]	^	↑
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	Ä	Ö	Å	Ü	



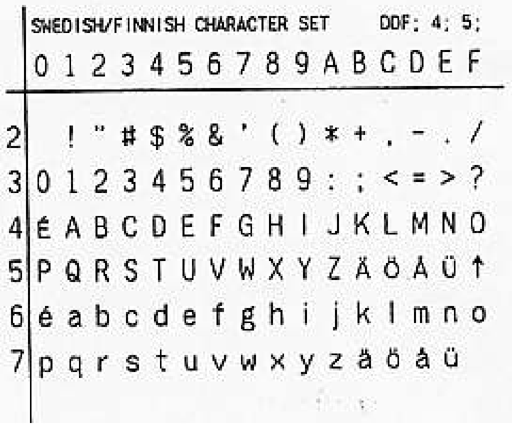
UK CHARACTER SET DDF; 4; 3;

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	£	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	Ä	¦	Å	Ü	



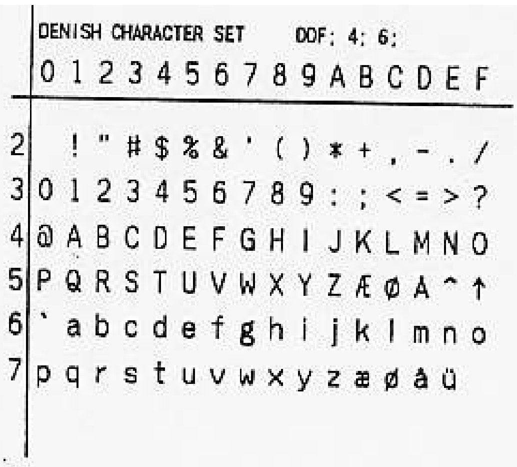
FRENCH CHARACTER SET . DDF; 4; 4;

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	à	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	°	ç	]	^	↑
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	é	ù	è	¨	



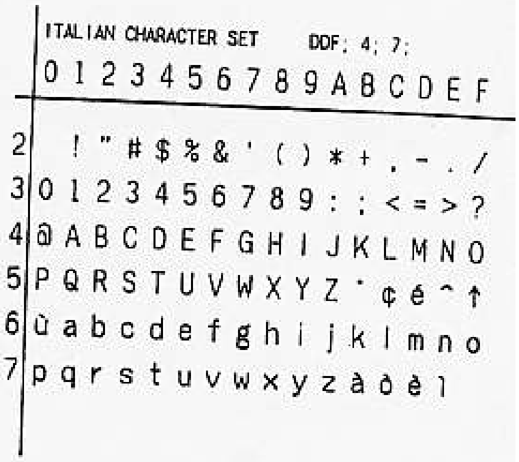
SWEDISH/FINNISH CHARACTER SET DDF; 4; 5;

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	É	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	Ä	Ö	Å	Ü	↑
6	é	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	ä	ö	å	ü	



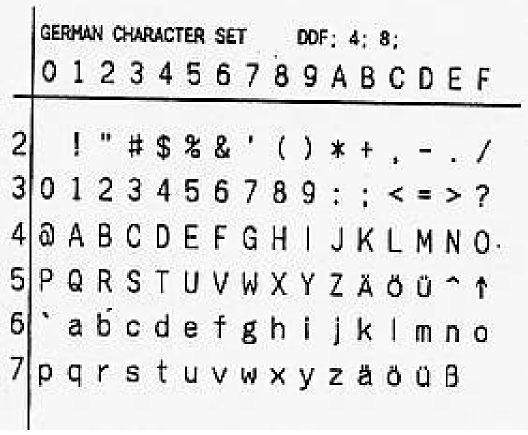
DENISH CHARACTER SET DDF; 4; 6;

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	Æ	Ø	Å	^	↑
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	æ	ø	å	ü	



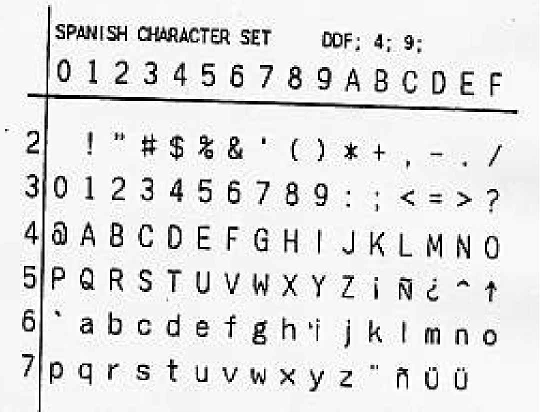
ITALIAN CHARACTER SET DDF; 4; 7;

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	˚	¢	é	^	↑
6	ù	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	à	ò	è	ì	



GERMAN CHARACTER SET DDF; 4; 8;

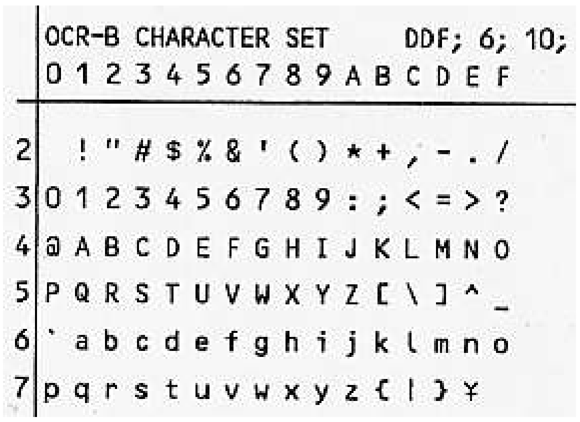
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	Ä	Ö	Ü	^	↑
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	ä	ö	ü	ß	



SPANISH CHARACTER SET DDF; 4; 9;

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	¡	Ñ	¿	^	↑
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	¨	ñ	Ü	ü	

The following table lists the OCR-B codes in hexadecimal.



OCR-B CHARACTER SET DDF; 6; 10;

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	¥	

10.2.5. Character Size Selection

The DFM (Dot Font Magnification) command is used to increase the size of characters in a dot font. A dot font can be magnified by a factor between 1 and 16, both horizontally and vertically.

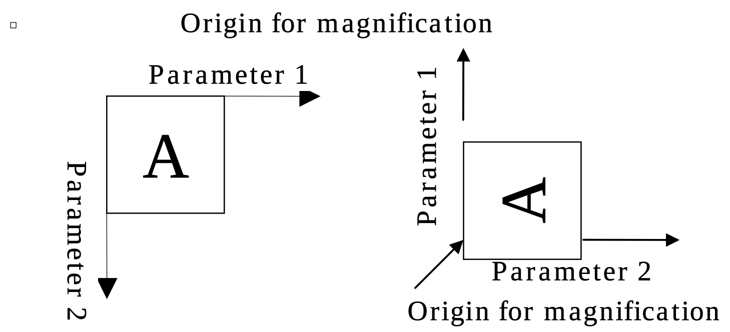
10.2.5.1. Dot Font Magnification (DFM)

The DFM command specifies the magnification factors vertically and horizontally for the font.
The syntax of this command is as follows.

```
DFM;XM;YM;
```

XM = horizontal magnification: integer from 1 to 16
YM = vertical magnification: integer from 1 to 16

The relation between the printing start position and the directions of magnification is shown below.



10.2.5.2. Dot Font Spacing (DFS)

The DFS command specifies the dot font inter-character spacing, in 0.005inches (0.127 mm) units. The syntax of this command is as follows.

```
DFS;X;
```

X = spacing (0.005inches (0.127 mm) units)

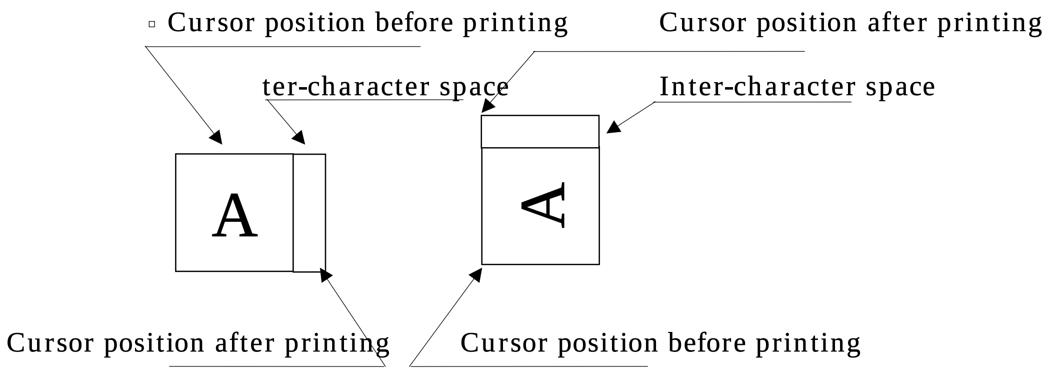
The actual spacing is X× 0.005inches(0.127mm)

If the inter-character spacing is not specified with the DFS command, the default value is 2 (0.010inches).

10.2.5.3. Text Positioning

Both absolute and relative positioning commands are used in positioning text and readable characters on a label. (See Chapter 4.)

The printing start reference position for dot font text is the upper left corner of the first character. Successive characters then follow in sequence.
The printing start position for a dot font is shown below.


Cursor position before printing
Cursor position after printing
ter-character space
Inter-character space
A
A
Cursor position after printing
Cursor position before printing

10.2.5.4. Dot Font Orientation (DFO)

The DFO command specifies the orientation of the text string, and also whether the writing direction is horizontal or vertical.
The syntax of this command is as follows.

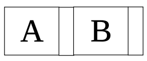
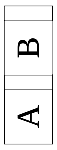
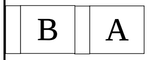
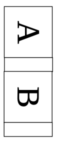
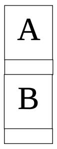
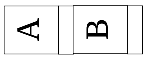
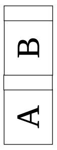
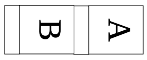
```
DFO;O;K;
```

O = text orientation
K = writing direction (vertical/horizontal)

Text orientation	O
0°	1
90°	2
180°	3
270°	4

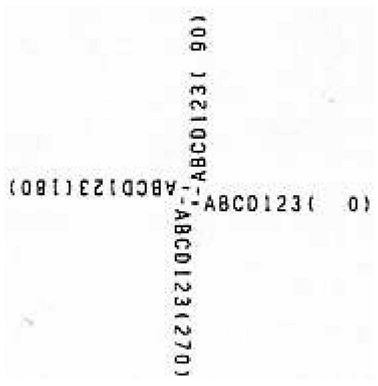
Writing	K
Horizontal	1
Vertical	2

The following examples show the various settings for the DFO command.

O / K	1	2	3	4
1				
2				

The following shows examples of program using the dot font and printing samples.

```
~^″DOTFONT″;1;0;400;0;SPB;UTOF;4100;
HBR;300;VBR;200;
DDF; 4; 1;DFM; 1; 1;DFS; 5;
DFO; 1; 1;                  # rot 0 #;
″-ABCD123 ( 0) ″;HOME;
DFO; 2; 1;                  # rot 90 #;
″-ABCD123 ( 90) ″;HOME;
DFO; 3; 1;                  # rot 180 #;
″-ABCD123 (180) ″;HOME;
DFO; 4; 1;                  # rot 270 #;
″-ABCD123 (270) ″;HOME;
TRM;¥
```



10.3. Outline Fonts

By using an outline font, text can be printed without jaggies.

10.3.1. Text Command Syntax

To specify text using an outline font, there are four commands: DDF (Define Dot Font), DCS (Dot Character Size), DFO (Dot Font Orientation), and DFS (Dot Font Spacing).

The syntax of these four commands is as follows.

DDF;*A*;*B*;DCS;*XD*;*YD*;DFO;*O*;*K*;DFS;*X*;

A = font identifier
B = country code
XD = horizontal dots
YD = vertical dots
O = text orientation
K = writing direction (vertical/horizontal)
X = inter-character space L (0.005inches(0.127 mm))

10.3.2. Fonts

The size of characters in an outline font can be specified in the range 48 to 256 dots both vertically and horizontally. An outline font can be rotated, or the character spacing specified.

10.3.3. Character Set

The alphanumeric outline font has a standard 96-character ASCII character set. Characters from ASCII space (32) to delete (127) can be printed.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	–	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	¥	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

10.3.4. Font Selection

The DDF (Define Dot Font) command selects the outline font.
The syntax of this command is as follows.

```
DDF;A;B;
    A = font identifier
    B = country code
```

font identifier	A
outline font (ASCII codes)	101

If the font identifier is not an outline font, a dot font is selected.

When specifying an outline font, specify the country code as 10 (Japan).

10.3.5. Character Size Selection

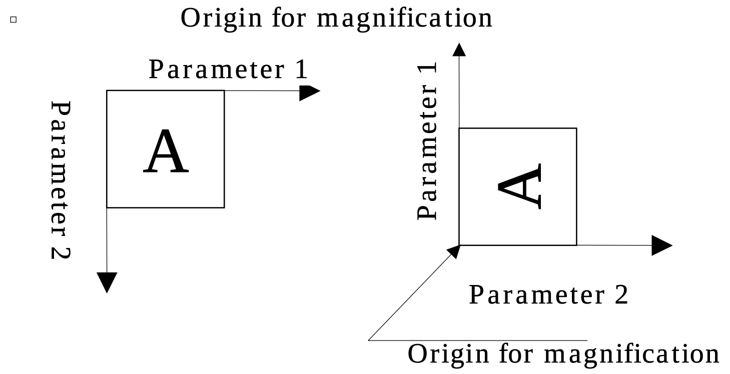
10.3.5.1. Define Character Size (DCS)

The DCS command specifies the size of text in an outline font. Both vertical and horizontal sizes can be set in the range from 48 to 256 dots.

The DCS command specifies the vertical and horizontal character size in pixels.
The syntax of this command is as follows.

```
DCS;XD;YD;
            XD = character width in dots (pixels) - integer from 48 to 256
            YD = character width in dots (pixels) - integer from 48 to 256
```

The relation between the printing start position and the text orientation is shown below.



Text at 0°   Text at 90°

10.3.5.2. Outline Font Spacing (Dot Font Spacing: DFS)

The DFS command specifies the outline font inter-character spacing, in 0.005inches(0.127 mm) units. The syntax of this command is as follows.。

DFS;*X*;

X = spacing 0.005inches(0.127 mm) units

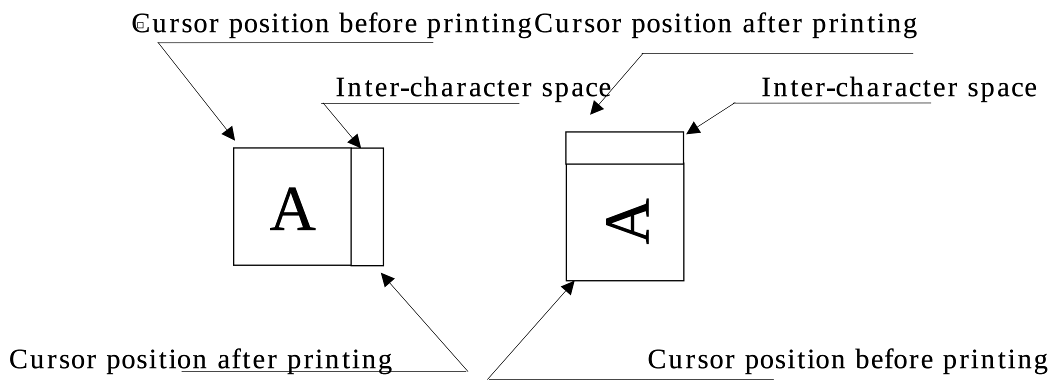
The actual spacing is X×0.005inches(0.127 mm).

If the inter-character spacing is not specified with the DFS command, the default value is 2 (0.010inches(0.254 mm)).

10.3.5.3. Text Positioning

The printing start reference position for outline font text is the upper left corner of the first character. Successive characters then follow in sequence.

The printing start position for a outline font is shown below.


Cursor position before printing
Cursor position after printing
Inter-character space
Inter-character space
A
A
Cursor position after printing
Cursor position before printing

10.3.5.4. Dot Font Orientation (DFO)

The DFO command specifies the orientation of the text string, and also whether the writing direction is horizontal or vertical.

The syntax of this command is as follows.

DFO;*O*;*K*;

O = text orientation

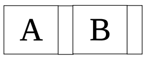
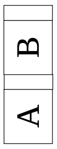
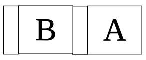
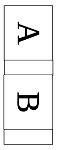
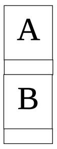
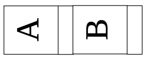
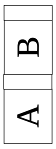
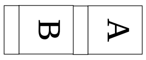
K = writing direction (vertical/horizontal)

Text orientation	*O*
0°	1
90°	2
180°	3
270°	4

Writing	*K*
Horizontal	1
Vertical	2

The following examples show the various settings for the DFO command.

O	1	2	3	4

K				
1				
1				

10.3.6. Entering Text Data

Specify the text enclosed in double quotation marks ("..."). Any number of characters can be specified in a text string. It is important, however, to make sure that the text will fit on the label. The following is an example of text input syntax.

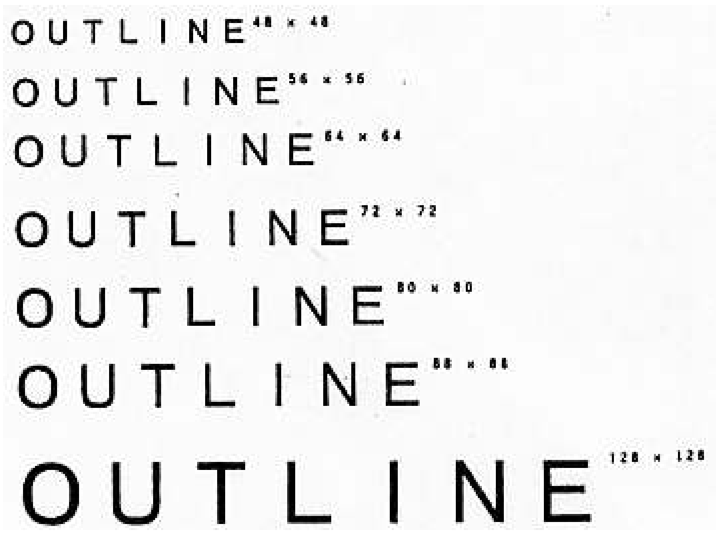
```
DDF;101;10;DCS;48;48;DFO;1;1;DFS;2;
"Text String"
```

The following example shows an outline font.

Note

The character sizes vary, because of white space surrounding the characters.

```
~^″OUTLINE2″;1;0;400;0;
SPB;VBR;10;HBR;50;
DDF;101;10;DFS;5;DFO;1;1;DCS;48;48;
″OUT LINE FONT″;
DDF;3;1;DFM;1;1;DFS;5;DFO;1;1;
″48 x 48″;
VBR;40;HBR;90;DDF;101;10;DCS;56;56;
″OUT LINE FONT″;
DDF;3;1;DFM;1;1;DFS;5;DFO;1;1;
″56 x 56″;
VBR;70;HBR;140;DDF;101;10;DCS;64;64;
″OUT LINE FONT″;
DDF;3;1;DFM;1;1;DFS;5;DFO;1;1;
″64 x 64″;
VBR;110;HBR;200;DDF;101;10;DCS;72;72;
″OUT LINE FONT″;
DDF;3;1;DFM;1;1;DFS;5;DFO;1;1;
″72 x 72″;
VBR;150;HBR;250;DDF;101;10;DCS;80;80;
″OUT LINE FONT″;
DDF;3;1;DFM;1;1;DFS;5;DFO;1;1;
″80 x 80″;
VBR;190;HBR;300;DDF;101;10;DCS;88;88;
″OUT LINE FONT″;
DDF;3;1;DFM;1;1;DFS;5;DFO;1;1;
″88 x 88″;
VBR;240;HBR;350;DDF;101;10;DCS;128;128;
″OUT LINE FONT″;
DDF;3;1;DFM;1;1;DFS;5;DFO;1;1;
″128 x 128″;
TRM;¥
```



11.Serialization

When using bar codes, it is sometimes necessary to add serial numbers to a label. RCL provides two ways of adding serial numbers. The entire label can be reconstructed as each label is printed, or the looping method can be used. A loop is used for the part of printing labels in which the serial number is added, and repeating until the necessary number of labels has been printed.

RCL allows three types of serial number to be specified: numeric (NUM), alphabetic (ALPH), and alphanumeric (BOTH).

In addition to the commands for controlling the loop, it is necessary to specify the type of the serial number data. Separate commands must be issued for a bar code serial number and a text or readable character serial number. For example, to produce a Code39, ITF, or CodaBar bar code and readable character serial numbers requires two sets of serial number command.

11.1. Looping

To print a number of labels, using a loop to increment or decrement a bar code or text item is highly efficient. This is because the printer does not have to create the entire image for each label, but only regenerate the loop portion. This section describes the three commands used for programming a loop: mark (MRK), bar code label count (BCLC), and return (RET).

To define the beginning and end of a loop the two commands MRK (mark) and RET (return) are required. As their names suggest, the MRK command marks the point in the program where the loop starts, and the RET command is placed at the end of the program, immediately before the TRM command, to return processing to the MRK position.

The total number of labels printed is determined by the header command portion which defines the number of labels. To increment or decrement a field, it is necessary to specify the number of labels to be printed with those numbers, and use the BCLC (Bar Code Label Count) command as a repeat count.
A bar code or text being printed as a serial number must be erased before printing the following label, using the draw white box command (DWBX).

11.1.1. MaRK (MRK)

The MRK command defines the beginning of a group of commands used as a loop. The syntax of the MRK command is as follows.

```
~^"Example";3;0;400;50;
SPB;DHR;1;
MRK;HBR;100;VBR;50;"12345";
```

In this case the text field "12345" is incremented or decremented while three labels are printed.

On one label, a maximum of 32 fields for serial numbers can be set. However, even when using more than one field for serial numbers, the MRK command is placed before the first of these fields only. Once one MRK command has been inserted, it is not necessary to insert a putative next MRK command. Since program control always returns to the (last) MRK command before the RET command, if a second MRK command is inserted, the first MRK command is ignored.

11.1.2. Bar Code Label Count (BCLC)

The BCLC command sets the repeat count, i.e. the number of copies of the same label required when printing serial numbers. The syntax of this command is as follows.

```
BCLC;A;
```

A = integer from 0 to 65,535

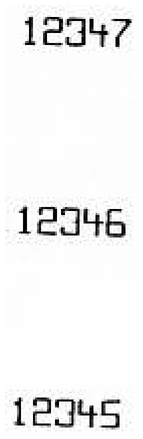
If there is no BCLC command, the default the default repeat count is 1, which means that the serial number value is incremented or decremented for each label. Specifying a repeat count of 2 means that two labels will be printed with each serial number. Note that the BCLC command is used to control label counts for both bar codes and readable characters.

11.1.3. RETurn (RET)

The RET command returns program control to the preceding MRK command, causing repeated printing of the intervening contents. The data before the MRK command is not repeated, and the same label is printed during printing. Repetition occurs until the number of labels specified in the header has been printed.

The following examples shows the use of the RET command. This short program prints three labels, incrementing the number by one.

```
~^"IL9-1";3;0;100;0;
SPB;MRK;VBR;30;HBR;0;BCLC;1;
DWBX;0;-30;200;50;
"12345";SAL;1;EOL;
RET;TRM;¥
```



Like the MRK command, there must be one RET command for each label. The RET command may be placed anywhere after the MRK command, but immediately before the TRM command is optimal.

Note

If the BRK command is used to break a single label image into a number of command blocks, the sequence from MRK to RET must be included within the last command block terminated by the TRM command.

```
~^ ... # header of first block #;
SPB;
...
BRK;¥; # end of first block #;
...
~^ ... # header of last block #;
RSPB;
...
MRK;   # start of loop #;
...
RET;   # end of loop #;
TRM;¥; # end of last block #;
```

11.1.4. Clearing a Serialized Field with DWBX

When printing a bar code as a serial number, before printing the next bitmap, it is necessary to use the draw white box (DWBX) command to clear the bar code or readable characters from the bitmap. Otherwise the new field will be superimposed on the previous field.

Place the DWBX command after the MRK command, and before incrementing or decrementing the serial number field. Since the DWBX command clears the area specified by its arguments, the new bar code data can be printed in this position. When determining the position to be cleared, remember that the upper left corner of the area to be cleared is the start position, whereas the lower left corner of a bar code is the start position.

The following example shows how to use the DWBX command to erase a Code39 bar code and print a serial number.

```
~^"DWBX";3;0;39;187;SPB;UTOF;0512;
MRK;HBR; 0;VBR; 0;DWBX; 0; 0;394;39;
HBR; 27;VBR; 28;BDEF; 1;BNEW; 2;BWEW; 5;BICG; 2;BCSH; 28;BCST;
"*RCL 00";BSAL;2;"*";BSTP;
HBR; 27;VBR; 30;DDF; 6;10;DFM; 1; 1;DFO; 1; 1;DFS; 8;
"*RCL 00";SAL;2;"*";
RET;
```

```
TRM;¥
```

RCL PLUS 02

RCL PLUS 01

RCL PLUS 00

11.1.5. Clearing a Serialized Field with the Printer Auto-Erase Function

11.1.5.1. Erase Mode ON (EMON)

For models before RCL Version 4.00, when printing a serial number it was necessary before printing the next bitmap to erase the image within a frame by using the DWBX command. From RCL Version 4.00, it is possible to have the printer automatically erase the portion of the serial number, and omit the DWBX command. The command to enable this function is the erase mode on command. When this mode is enabled it is not necessary to erase the area within the serial number portion.

When the printer has the erase mode switched on with this command, a serial number field specified by SAL, BSAL, VLP, and BVLP commands is erased by the printer itself before drawing the next bitmap.

> Note
> With the erase mode on, if a command is issued to erase the area within the box, the same area of image will be erased twice. If using the automatic erase mode, remove the white box drawing command.

The syntax of this command is as follows.

```
EMON;
```

The following example shows the example previously shown using the DWBX command, but using the erase mode (EMON command).

```
~^"DWBX";3;0;39;187;SPB;EMON;UTOF;0512;
MRK;HBR; 0;VBR; 0;
HBR; 27;VBR; 28;BDEF; 1;BNEW; 2;BWEW; 5;BICG; 2;BCSH; 28;BCST;
"*RCL 00";BSAL;2;"*";BSTP;
HBR; 27;VBR; 30;DDF; 6;10;DFM; 1; 1;DFO; 1; 1;DFS; 8;
"*RCL 00";SAL;2;"*";
RET;TRM;¥
```

11.2. Incrementing

RCL allows three kinds of data, numeric, alphabetic, and alphanumeric to be used for a serial number. This section describes the three commands used to control incrementing.

11.2.1. Numeric (NUM)

The NUM command is used to increment or decrement numeric data (digits 0—9) only. NUM is the default increment setting, so after issuing an ALPH or BOTH command, unless there is a need to return to a numeric serial number, a NUM command is not necessary. (See below.)

With NUM, characters other than numeric digits 0123456789 cannot be used. If characters other than these are used the serial number is not printed. The syntax of this command is as follows.

```
NUM;
```

The increment command must come before the field to be made the serial number. When printing more than one serial number field, unless another increment command is specified, the NUM command is valid for all fields.
The following example shows characters being incremented by 2 (IDF = 2), with both correct and incorrect examples.

```
  Legal                         Illegal
BCLC;1;IDF;2;NUM;             BCLC;1;IDF;2;NUM;
"999898";SAL;3;               "999A98";SAL;3;
```

```
Result (printing three labels)
999898                        999A98
999900                        999A00  (Note that 'A' is not part of serial number.)
999902                        999A02
```

11.2.2. Alphabetic (ALPH)

The ALPH command is used for printing capital letters A to Z as a serial number. Lowercase letters cannot be used in a serial number. The syntax of this command is as follows.

```
ALPH;
```

An alphabetic increment proceeds from A to B, then C, and on to Z, then returns to A, while the next digit position is advanced. As for the NUM command, the ALPH command must come before the field to be made the serial number, and remains valid until a different increment command is issued.
The following example shows the use of the ALPH command, with characters incremented by two, and with correct and incorrect examples.

```
Correct example              Incorrect example
BCLC;1;IDF;2;ALPH;           BCLC;1;IDF;2;ALPH;
"ZZZZZX";SAL;3;              "ZZZ3ZX";SAL;3;

Result (printing three labels)
ZZZZZX                       ZZZ3ZX
ZZZZZZ                       ZZZ3ZZ   (Note that '3' is not part of serial number.)
ZZZAAB                       ZZZ3AB
```

11.2.3. Alphanumeric (BOTH)

The BOTH command allows both numeric and alphabetic data as the serial number. Numeric data uses the integers 0 to 9, and the correct alphabetic characters are the capital letters A to Z. Lowercase letters cannot be used in a serial number. The syntax of this command is as follows.

```
BOTH;
```

With BOTH, values change in the following character sequence:

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ012...

The BOTH command must come before the field to be made the serial number, and remains valid until a different increment command is issued. The following example shows the use of the ALPH command, with characters incremented by 12 (IDF = 12), and with correct and incorrect examples.

```
   Legal                         Illegal
BCLC;1;IDF;12;BOTH;          BCLC;1;IDF;12;BOTH;
"ZZZ0Z96";SAL;4;             "ZZZ0Z9b";SAL;4;

Result (printing four labels)

ZZZ0Z96                      ZZZ0Z9b   (Lowercase 'b' not part of serial number.)
ZZZ0ZA8                      ZZZ0Z9b
ZZZ0ZBA                      ZZZ0Z9b
ZZZ0ZCC                      ZZZ0Z9b
```

11.3. Bar Code Serial Numbers

To make a bar code a serial number, in addition to the commands already described, the bar code increment/decrement command (BCID) and Bar Save Address Length (BSAL) command are also used. The BCID command specifies the amount of the increment or decrement, and the BSAL command specifies the number of digits within the bar code to be changed by the serial number setting.

11.3.1. Bar Code Increment/Decrement (BCID)

The BCID command specifies by how much the bar code serial number is incremented or decremented. A positive number indicates an increment, and a negative number a decrement.
The syntax of this command is as follows.

```
BCID;A;
```
A = integer from -32,768 to +32,767

If the BCID command is not issued, the default value is 1, and therefore the bar code value is incremented by 1 for each label. For example it changes from 0 to 1, then to 2, and so on. The next example shows the BCID command as an increment function and also a decrement function.

```
BCID;10; # increment bar code by 10 for each label #;
BCID;-10; # decrement bar code by 10 for each label #;
```

11.3.2. Bar Save Address Length (BSAL)

The BSAL command is placed immediately after the value to be the bar code serial number. The BSAL command specifies how many digits within the bar code data are to be the serial number. This command has no default value, so must always be defined to create a serial number. The syntax of this command is as follows.

```
BSAL;A;
```
A = integer from 0 to 255

Following the BSAL command, is specified the number of digit positions in the serial number field counting from the last digit in the field. The BSAL command must be placed immediately after the value to be the bar code serial number.

Next, various applications of BSAL are shown.

```
BDEF;1;BCSH;50;BCST;"*123456789";BSAL;4;"*";BSTP;
```

In this example, four digits counting from the end of a Code39 bar code (6789) are used as a serial number. Note that BSAL appears before the stop character (*). If BSAL is placed after the stop character, since the stop character is a special character, the serial number will stop.

```
BDEF;10;UMAG;0;BCST;"01234567890";BSAL;5;BSTP;
```

The second example uses the last five digits (67890) of a UPC-A bar code as a serial number. The first six digits (012345) do not change.

```
BDEF;3;BCSH;50;BCPI;1;BCST;"A123456";BSAL;6;"@C";BSTP;
```

The third example the six data characters of a CodaBar bar code (123456) as a serial number. The start character (A), check character (@), and stop character (C) do not form part of the serial number.

```
BDEF;1;BCSH;75;NUM;BCST;"*0123";BSAL;3;ALPH;"ABC";BSAL;3;"*";BSTP;
```

The last example uses two methods for a Code39 bar code serial number. The integer 123 is a numeric serial number, and the characters "ABC" form an alphabetical order serial number.

11.3.3. Bar Code Variable Field Length and Position (BVLP)

The BVLP command is used to specify the length and position of the field to be incremented or decremented. This command is used together with a bar code field. This command is specified immediately after the last number of the field to be incremented (decremented).

BVLP;*A*;*B*;

A = serial number length: integer 0 to 255
B = serial number position: integer from 1 to 255

Following the BVLP command the value B specifies the position of the serial number counting from the last character of the field (the last character of the field being counted as 1), and the value A following the command indicates the number of characters or numeric digits in the serial number counting from the position specified by B.
The serial number length cannot exceed (total digit positions in field − (serial number digit position−1)). The serial number digit position also cannot exceed the total number of digit positions in the field.

11.4. Serializing Text

Text and bar code readable characters can form a serial number as long as the text is an integer or capital letters. For a text serial number, in addition to the commands (MRK, TRM, BCLC, and DWBX) described above, the increment/decrement field (IDF) and save address length (SAL) commands are used. This section describes the IDF and SAL commands in detail, and shows examples of text serial numbers.

11.4.1. Increment/Decrement Field (IDF)

The IDF command applies to text in the same way that the BCID command applies to a bar code, and specifies the amount by which the text is incremented. The syntax of the IDF command is as follows.

IDF;*A*;

A = integer from –32,768 to +32,767

A positive number indicates an increment, and a negative number a decrement. The following examples show the text being incremented and decremented.

```
IDF;100; # increment bar code by 100 for each label #;
IDF;-50; # decrement bar code by 50 for each label #;
```

11.4.2. Save Address Length (SAL)

The SAL command is used to specify the character positions within the field which will be incremented or decremented. In the same way that BSAL is used with a bar code, SAL is used with a text field Both commands are specified immediately after the last number of the field to be incremented (decremented). The syntax of the SAL command is as follows.

SAL;*A*;
 A = integer from 0 to 255

Following the SAL command, is specified the number of digit positions in the serial number field counting from the last digit in the field. The total number of digit positions in the serial number cannot exceed the total number of digits in the field. For example, for a serial number within a field specified as 234789, the maximum number of digits which can be specified is 6.

The following example illustrates the use of the SAL command.

```
~^"Example1";3;0;400;50;
SPB;BCLC;1;IDF;-1;
MRK;"12345";SAL;1;
```

In this example, the last digit (5) of the numeric field 12345 is decremented.

11.4.3. Variable Field Length and Position (VLP)

The VLP command is used to specify the length and position of the field to be incremented or decremented. This command is used together with a text field.
This command is specified immediately after the last number of the field to be incremented (decremented).
The syntax of the VLP command is as follows.

VLP;*A*;*B*;
 A = serial number length: integer 0 to 255
 B = serial number position: integer from 1 to 255

Following the VLP command the value B specifies the position of the serial number counting from the last character of the field, and the value A following the command indicates the number of characters or numeric digits in the serial number counting from the position specified by B.

The serial number length cannot exceed (total digit positions in field − (serial number digit position−1)). The serial number digit position also cannot exceed the total number of digit positions in the field.

The following example shows the use of the VLP command.

```
~^"Example1";3;0;400;50;
SPB;IDF;1;
MRK;"1234567890";VLP;3;5;
```

In this example, three digits 456 of the numeric field 1234567890 from the fifth from last, the digit 6 are incremented.

11.4.4. Serial Number Skip Function (Except: EXCP)

The EXCP command allows specified characters to be skipped when printing a serial number.

EXCP;*CCC*;

CCC: characters to be skipped, in ascending order

The characters which can be specified by C are 0 to 9 and A to Z.
In combination with a numeric serial number (NUM), alphabetic serial number (ALPH), or alphanumeric serial number (BOTH), particular characters can be omitted from a serial number.
If there is a character other than 0 to 9 and A to Z, or the characters are not in ascending order, a command error results.

- It is not possible to make a plurality of different settings of characters to be omitted within a single format.

If more than one EXCP command is entered, only the last is valid.

- Combination other than numeric serial number (NUM), alphabetic serial number (ALPH), and alphanumeric serial number (BOTH) are not possible.

  Moreover, within a single format, these types of serial number cannot be mixed.

Example:

(A) When a numeric serial number (NUM) is selected

The NUM command specifies numeric data (0–9) for a serial number.
This is incremented or decremented skipping the numeric data specified by the EXCP command.

Any characters other than numeric digits specified to be skipped by the EXCP command are ignored.

Example not using EXCP	Example using EXCP		
BCLC;1;IDF;1;NUM;	BCLC;1;IDF;1;<u>EXCP;6;</u>NUM;		
"000005";SAL;3;	"000005";SAL;3;		
Result (printing three labels)			
000005	000005		6 is a skipped character
000006	000007	←	so this is incremented
000007	000008		as on the left

(B) When an alphabetic serial number (ALPH) is selected

The ALPH command specifies capital letters A to Z for a serial number.
This is incremented or decremented skipping the alphabetic data specified by the EXCP command.
Any characters other than capital letters specified to be skipped by the EXCP command are ignored.

Example not using EXCP	Example using EXCP		
BCLC;1;IDF;1;ALPH;	BCLC;1;IDF;1;EXCP;B;ALPH;		
"AAAAAA";SAL;2;	"AAAAAA";SAL;2;		
Result (printing three labels)			
AAAAAA	AAAAAA		B is a skipped character
AAAAAB	AAAAAC	←	so this is incremented
AAAAAC	AAAAAD		as on the left

(C) When an alphanumeric serial number (BOTH) is selected serial number

The BOTH command specifies both digits 0 to 9 and capital letters A to Z for a serial number.
This is incremented or decremented skipping the alphanumeric data specified by the EXCP command.

With BOTH, data characters change in the sequence below.

If the EXCP command specifies I and O (and IDF is 1), I and O are skipped as follows:

0123456789ABCDEFGHJKLMNPQRSTUVWXYZ0123

Not using EXCP (IDF = 3)	Using EXCP (IDF = 3)
BCLC;1;IDF;3;BOTH;	BCLC;1;IDF;3;<u>EXCP;I</u>O;BOTH;
"77777C";SAL;1;	"77777C";SAL;1;

Result (printing three labels)

```
77777C          77777C
77777F          77777F
77777I          77777J
77777L          77777M
77777O          77777Q
```

8 9 A B $\underline{\text{C}}$ D E $\underline{\text{F}}$ G H $\underline{\text{J}}$ K L $\underline{\text{M}}$ N P $\underline{\text{Q}}$ R S $\underline{\text{T}}$ U …＊

① ② ③ ④ ⑤

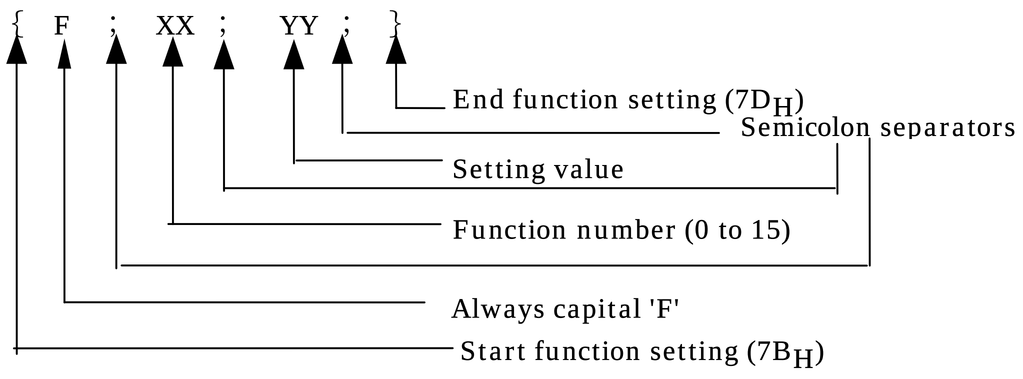
Skipped characters are excluded when calculating the increment.
In other words, as shown in the figure above, the incrementing or decrementing character sequence is based on the characters with the skipped characters omitted. The same is true of NUM and ALPH.

12.Printer Control

12.1. Function Setting Command

By sending a command from the computer, the function setting value can be changed. (This command is called the function setting command.)

Function setting command


{ F ; XX ; YY ; }
End function setting (7D$_H$)
Semicolon separators
Setting value
Function number (0 to 15)
Always capital 'F'
Start function setting (7B$_H$)

The terminating '$_H$' on 7D$_H$ and 7B$_H$ indicates a one-byte control code (hexadecimal).

Note

- For the function numbers and setting values, refer to the printer function setting tables.
- Position the function setting command before the RCL command header.
- It is not possible to include comments in a function setting command.

12.2. Form Length (Label Pitch)

The print buffer length must be specified in the header (see Section 2.6). The physical length of the labels must be set with the label length setting (Use Top Of Form: UTOF) command. This command is described here.

12.2.1. Method of Use

When the length of the printing area specified in the command header is the same as the form length, the UTOF command may be omitted. If these lengths are different, in order to correctly specify form length and the position of the image on the label to the printer, the UTOF command is required.

12.2.2. Label Length Setting (Use Top Of Form:　UTOF)

The UTOF command specifies the sum of the physical label length and the spacing between labels, that is to say, the distance from the top of one label to the top of the next (label pitch). The syntax of this command is as follows.

```
UTOF;A;
```

A = number from 1 to 10000:　label pitch (in mm) divided by 0.0254.

If the form length is 50 mm, use the following specification.

```
UTOF;1968;
```

The UTOF command must be placed immediately after the SPB command. In the following example, a Code39 bar code is printed on a label with a length of 47.1 mm. Of this length, the actual area used for printing has a printing length specified in the header of 180 pixels (45.72 mm).

```
~^"IL10-3";1;0;180;0;
SPB;UTOF;1855;VBR;50;HBR;50;
BDEF;1;BCSH;50;BNEW;3;BWEW;8;
BCST;"*100100*";BSTP;EOL;
VPR;20;HPR;110;DHR;$8002;DCH;15;DCW;30;ICS;2;
"*100100*";
TRM;¥
```

100100

12.3. Inverted Printing

12.3.1. Method of Use

Since the printer prints from the bottom of the image to the top, the operator can see the label in the correct orientation. However, there may be cases in which it is required to print upside down, from the top. The FLIP command inverts the image through 180 degrees within the printer.

12.3.2. Flip (FLIP)

The FLIP command sends an instruction to the printer to invert the image through 180 degrees. The syntax of this command is as follows.

```
FLIP;XX;
```

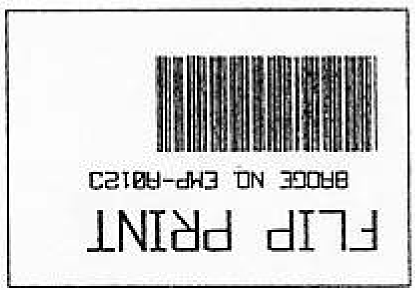
XX = label width, in horizontal pixels (0.127 mm)

Regardless of the value of this argument, the whole image is always inverted.

12.3.3. Position

The FLIP inverts the labels and prints as they are fed out. The top of the label is printed first and the bottom last. Also, the right side of the label is on the left and the left side of the label is on the right. The relative position of objects (bar codes, text, and other printing on the label) on the label is not changed.

The following shows an example of printing using the FLIP command.



```
~^"IL10-4";1;0;150;0;
SPB;FLIP;420;HLT;1;VLT;2;VBR;0;HBR;0;
DBOX;0;0;420;145;
VBR;40;HBR;35;
DHR;$8000;DCH;30;DCW;60;ICS;4;
"FLIP PRINT";EOL;
VPR;20;HPR;30;DCH;12;DCW;28;ICS;2;"BADGE NO. EMP-A0123";EOL;
VPR;60;BDEF;1;BCSH;50;BNEW;2;BWEW;6;
BCST;"*EMP-A0123*";BSTP;EOL;
TRM;¥
```

12.4. Remote Operation

It is possible to send commands to the printer from the host computer to reinitialize the printer, control pausing and unpausing, and if the printer is connected to the host through a serial interface, to obtain status data returned by the printer.

The remote operation commands, unlike the commands described so far, are executed as soon as received by the printer. Ordinary commands, on the other hand, are stored in the input buffer, and executed in sequence. Another difference is that normally commands are alphanumeric character strings, whereas the remote operation commands are one-byte control characters. This section describes the remote operation commands and how to use them.

Note

These commands are not accepted if the input buffer is full.

12.4.1. Printer Initialization (CAN: 18$_H$)

Printer initialization (CAN: 18$_H$) returns all command parameters to their default settings, and resets any error state such as a command error. When the printer receives a CAN command, all printer control parameters are cleared, and the state is the same as after powering on.

12.4.2. Pause (DC2: 12$_H$)

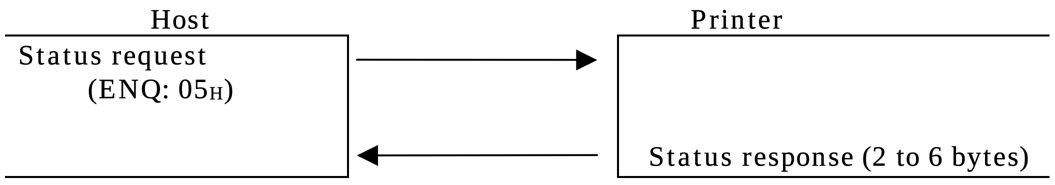
If the printer is ready to print, the DC2 command puts it in the paused state. This is the same as pressing the Pause button on the front panel of the printer. To return the printer to the ready state, press the pause button again, or send the unpause command.

12.4.3. Unpause (DC4: 14$_H$)

When the printer is paused and it receives a DC4 command, it returns to the ready state. This is the same as pressing the Pause button on the front panel of the printer. Note that if the printer is paused as a result of a printer error, the DC4 command is ignored.

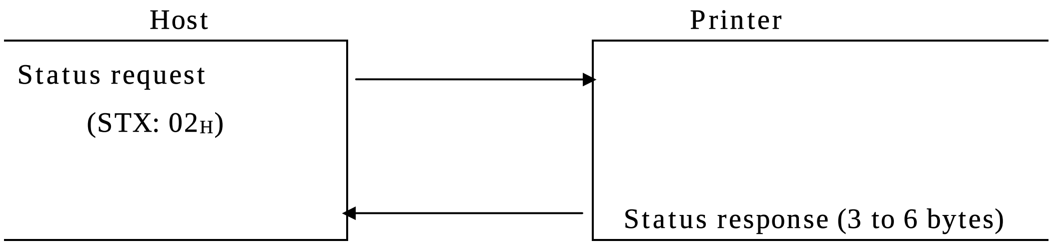
12.4.4. Status Request (ENQ: 05$_H$)

If the printer is connected to the host computer through a serial interface, then when an ENQ command is received the printer transmits its current status information to the host. The serial interface supports duplex communications (from the host to the printer and from the printer to the host), but the parallel (Centronics) interface only supports communications from the host to the printer. For this reason, the ENQ command is ignored on a parallel interface. For details of the status response, see the status tables in Chapter 24.



12.4.5. Status Request (STX: 02$_H$)

If the printer is connected to the host computer through a serial interface, then when an STX command is received the printer transmits its current status information to the host. For details of the status response, see the status tables in Chapter 24.



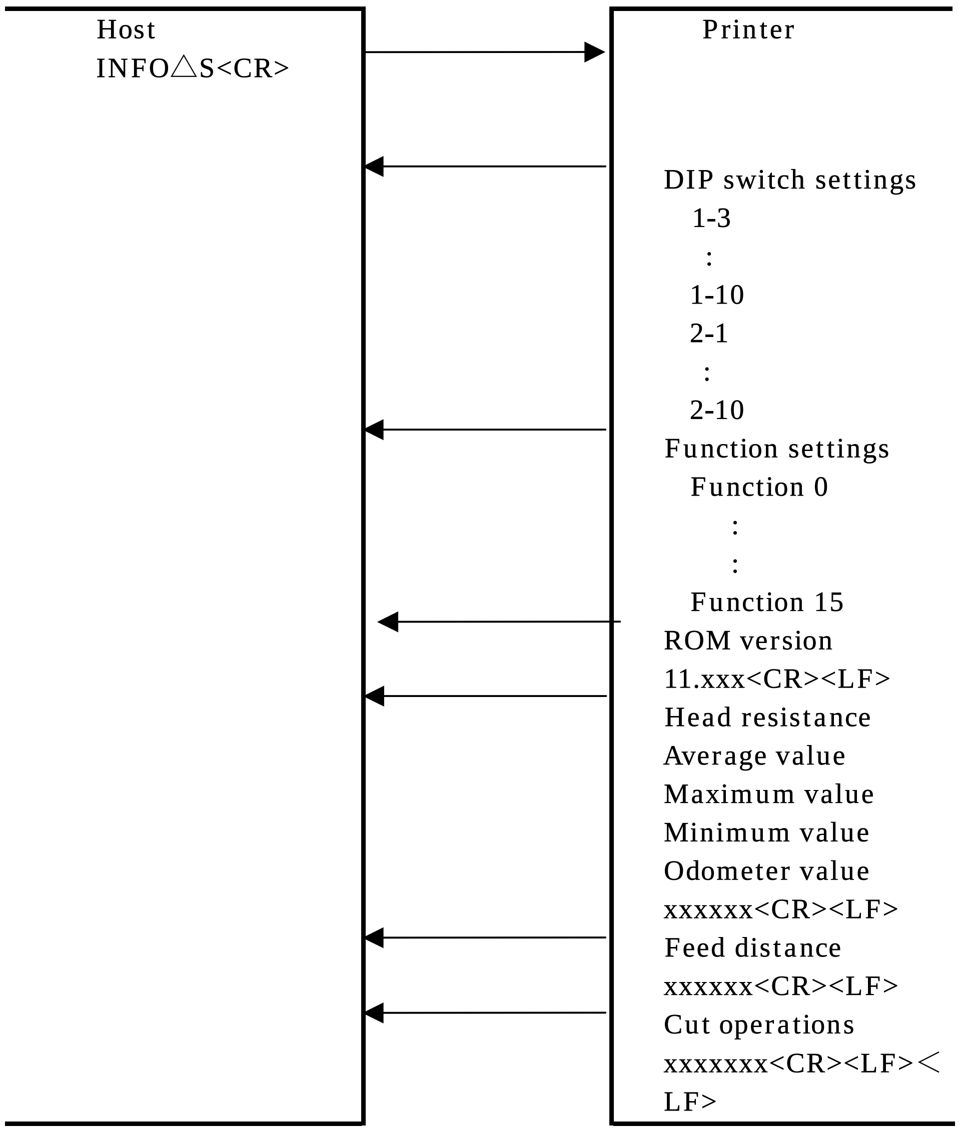
12.4.6. Printer Information Request (INFO)

This command is used to obtain the printer function settings, ROM version, and so forth.

When the INFO command is sent from the printer, as shown in the following figure, the settings of the DIP switches, the function settings, the ROM version, the head resistance value, odometer value, feed distance value, and cutting operation count are returned.

The INFO command format and response is as follows.

INFO△S＜CR>　　　　Note: a triangle indicates a space.



The information items are separated by CR ($0D_H$) and LF ($0A_H$), and after the last item, the cut operation count, there is one CR and two LFs.

The values of each item are in the following format.

DIP switch settings

When ON: ON△△<CR><LF>
When OFF: OFF△<CR><LF>

Function settings

For 0 to 9:   03<CR><LF>
For 10 or more: 13<CR><LF>
Negative: -10<CR><LF>

ROM version

11. xxx<CR><LF>
xxx is the ROM version.

Head resistance (ohms)

xxxx<CR><LF>

Odometer value, feed distance (m)

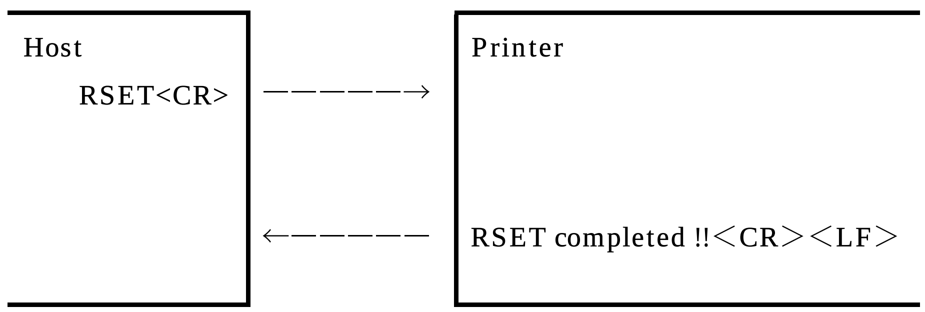
xxxxxx<CR><LF>

Cut operation count

xxxxxxx<CR><LF>

12.4.7. Reset Command (RSET)

This command resets the feed distance stored in backup memory in the printer to zero. When the RSET command is sent from the host, the printer clears the feed distance, and when the clear operation is completed returns a message.
The RSET command and response are shown in the following figure.



RSET<CR> ——————→

←—————— RSET completed !!<CR><LF>

13.Logos

Special objects and images which cannot be drawn using the line and box commands can be created with the LOGO command. This chapter describes the LOGO command, and gives examples of logos (bitmaps) created with RCL.

13.1. Logo (LOGO)

```
LOGO;B;R;DATA;. . . ;DATA;
```

B = number of bytes in each row of data
R = number of rows of data
DATA = byte values of data, in decimal or hexadecimal

The number of bytes of data supplied to the LOGO command must be equal to B ×R.
For example, consider the following command:

```
LOGO;15;10;
```

Here, 15 bytes by 10 rows amounts to 150 bytes. Therefore, the data part of the LOGO command must consist of 150 bytes.

The data is normally specified by a decimal number for each byte, but a dollar sign in front of a byte value indicates that it is written in hexadecimal notation. Within the data for one logo command, decimal data and hexadecimal data can be mixed.

Note
The bitmap data does not correspond to logical pixels, but physical pixels. The size of the physical pixels depends on the head density.

Note
Do not use a number of bytes in one row of data which exceeds the printing width. If the size of a format program when using the LOGO command is large, use the BRK command. (See Section 3.1.4.)

13.2. Logo Bitmap Expansion

For models before RCL Version 4.00, because of the limitations of the printer internal control buffer size, when sending a large volume of data such as a logo to the printer it was necessary to use BRK and RSPB commands to break the data into a number of format data blocks, but from RCL Version 4.00 the logo data is not first stored in the control buffer before drawing, but is written directly to the image buffer. Thus data which previously was broken into a number of format files with BRK and RSPB commands need no longer be so, and can be entered as a single format. There is no setting command for this function. (The change is only to the internal processing in the printer.)

A logo, like other objects, can be positioned on the label using either absolute positioning or relative positioning. It is important to remember that the printer normally starts drawing a byte at the left edge. If before printing a logo the cursor is in the middle of a byte of data, the printer will start drawing the logo from the left edge of the byte. As for lines and boxes, a logo does not change the cursor position.

13.3. Logo Command Specifying Data

13.3.1. Logo Data Format Specification (LOGD)

This command allows the logo command to select the representation of image data.
By choosing a new format, the logo data can be entered with approximately one-half of the previous input.

The syntax of this command is as follows.

LOGD;*F;C;R;Data…Data*;

F = selects logo data format:
0: decimal/hexadecimal type (conventional data format)
1: hexadecimal character format (compressed data format)
(The compressed data format represents each byte of binary data by a two-byte character representation.)
E.g. binary 1110 0011 (E3hex) is represented by "E3".

C = number of bytes per row
R = number of lines in logo
Data = byte values of data, in decimal or hexadecimal

The number of bytes of data supplied to the logo command must be equal to C× R.
In format 0:
Normally a byte of data is shown in decimal, and after the byte value is a semicolon separator. It is also possible to put a dollar sign in front of the value expressed in hexadecimal. It is also possible to mix the two representations within a logo data set.
In format 1:
The data representation is entirely hexadecimal, and no semicolon separator or preceding dollar sign is necessary to indicate hexadecimal. There should, however, be a semicolon at the end.

Note
It is not possible to store logo data in format 1 (hexadecimal character type) directly using the store logo in extended memory function. For saving in extended memory, the data must be created with the LOGO command.

13.4. Logo Examples

The following are examples of the use of the logo function for two special characters: "registered trade mark" and "copyright."

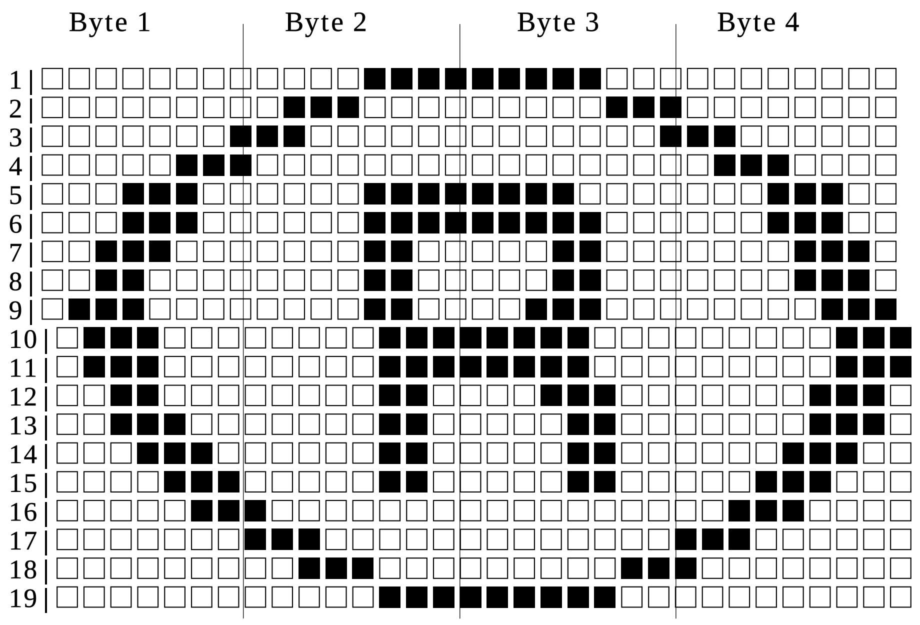
```
~^"REGISTER";1;0;150;0;
SPB;
VBR;5;HBR;120;
LOGO;4;19;
0;$0F;$F8;0;
0;$70;$07;0;
1;$C0;1;$C0;
7;0;0;$70;
$0E;$0F;$F0;$38;
$1C;$0F;$F8;$1C;
$38;$0C;$18;$0E;
$30;$0C;$18;6;
$70;$0C;$38;7;
$70;$0F;$F0;7;
$70;$0F;$F0;7;
$30;$0C;$38;6;
$38;$0C;$18;$0E;
$1C;$0C;$18;$1C;
$0E;$0C;$18;$38;
$7;0;0;$70;
1;$C0;1;$C0;
0;$70;$07;0;
0;$0F;$F8;0;
BRK;¥

~^"COPYRIGHT";1;0;50;0;
RSPB;
VBR;5;HBR;200;
LOGO;4;20;
```

```
1;$FF;$FF;$C0;
3;$FF;$FF;$E0;
7;0;0;$70;
$0E;0;0;$38;
$1C;$0F;$F8;$1C;
$1C;$1F;$FC;$1C;
$1C;$38;$0E;$1C;
$1C;$70;$07;$1C;
$1C;$70;0;$1C;
$1C;$70;0;$1C;
$1C;$70;0;$1C;
$1C;$70;0;$1C;
$1C;$70;7;$1C;
$1C;$38;$0E;$1C;
$1C;$1F;$FC;$1C;
$1C;$0F;$F8;$1C;
$0E;0;0;$38;
7;0;0;$70;
3;$FF;$FF;$E0;
1;$FF;$FF;$C0;
TRM;¥
```

Note

- This example shows portions printed as a single dot, but in practice at least two dots should be used.
- If the black portion is too large, the printing quality made deteriorate.



1	□□□■	A	■□■□
2	□□■□	B	■□■■
3	□□■■	C	■■□□
4	□■□□	D	■■□■
5	□■□■	E	■■■□
6	□■■□	F	■■■■
7	□■■■		
8	■□□□		
9	■□□■		

```
~^"LOGO";5;0;39;170;SPB;UTOF;0512;

HBR;160;VBR; 20;
LOGO;12;87;
$00;$00;$00;$00;$00;$1f;$c0;$00;$00;$00;$00;$00;
$00;$00;$00;$00;$03;$ff;$c0;$00;$00;$00;$00;$00;
$00;$00;$00;$00;$1f;$ff;$c0;$00;$00;$00;$00;$00;
$00;$00;$00;$00;$7f;$ff;$c0;$00;$00;$00;$00;$00;
$00;$00;$00;$03;$ff;$ff;$c0;$00;$18;$00;$00;$00;
$00;$00;$00;$0f;$ff;$e0;$00;$00;$1e;$00;$00;$00;
$00;$00;$00;$1f;$fe;$00;$00;$00;$1f;$00;$00;$00;
$00;$00;$00;$7f;$f0;$00;$00;$00;$1f;$c0;$00;$00;
$00;$00;$00;$ff;$80;$00;$00;$00;$1f;$e0;$00;$00;
$00;$00;$01;$fe;$00;$00;$00;$00;$1f;$f0;$00;$00;
$00;$00;$07;$fc;$00;$00;$00;$00;$1f;$fc;$00;$00;
$00;$00;$0f;$f0;$00;$00;$00;$00;$1f;$fe;$00;$00;
$00;$00;$1f;$e0;$00;$00;$00;$00;$1f;$ff;$00;$00;
$00;$00;$3f;$80;$00;$00;$00;$00;$1f;$3f;$80;$00;
$00;$00;$7f;$00;$00;$00;$00;$00;$1f;$1f;$c0;$00;
$00;$00;$fe;$00;$00;$00;$00;$00;$1f;$0f;$e0;$00;
$00;$00;$fc;$00;$00;$00;$00;$00;$1f;$07;$e0;$00;
$00;$01;$f8;$00;$00;$00;$00;$00;$1f;$03;$f0;$00;
$00;$03;$f0;$00;$00;$00;$00;$00;$1f;$01;$f8;$00;
$00;$07;$e0;$00;$00;$00;$00;$00;$1f;$00;$fc;$00;
$00;$07;$e0;$00;$00;$00;$00;$00;$1f;$00;$fc;$00;
$00;$0f;$c0;$00;$00;$00;$00;$00;$1f;$00;$7e;$00;
$00;$1f;$80;$00;$00;$01;$e0;$00;$1f;$00;$3f;$00;
$00;$1f;$80;$00;$00;$01;$e0;$00;$1f;$00;$3f;$00;
$00;$3f;$00;$00;$00;$01;$e0;$00;$1f;$00;$1f;$80;
$00;$3e;$00;$00;$00;$01;$e0;$00;$1f;$00;$0f;$80;
$00;$3e;$00;$00;$00;$01;$f0;$00;$3f;$00;$0f;$80;
$00;$7c;$00;$00;$00;$01;$f0;$00;$3e;$00;$07;$c0;
$00;$7c;$00;$00;$00;$01;$f8;$00;$7e;$00;$07;$c0;
$00;$fc;$00;$00;$00;$01;$fc;$00;$fc;$00;$07;$e0;
$00;$f8;$00;$00;$00;$01;$ff;$03;$f8;$00;$03;$e0;
$00;$f8;$00;$00;$00;$01;$ff;$ff;$f0;$00;$03;$e0;
$01;$f8;$00;$00;$00;$01;$ff;$ff;$e0;$00;$03;$f0;
$01;$f0;$00;$00;$00;$01;$ef;$ff;$c0;$00;$01;$f0;
```

```
$01;$f0;$00;$00;$00;$01;$e3;$ff;$00;$00;$01;$f0;
$01;$f0;$00;$00;$00;$01;$e0;$00;$00;$00;$01;$f0;
$01;$f0;$00;$00;$00;$01;$e0;$00;$00;$00;$01;$f0;
$03;$e0;$00;$00;$00;$01;$e0;$00;$00;$00;$00;$f8;
$03;$e0;$00;$00;$00;$01;$e0;$00;$00;$00;$00;$f8;
$03;$e0;$00;$00;$00;$01;$e0;$00;$00;$00;$00;$f8;
$03;$e0;$00;$00;$00;$01;$e0;$00;$00;$00;$00;$f8;
$03;$e0;$00;$00;$0f;$f1;$e0;$00;$00;$00;$00;$f8;
$03;$e0;$00;$00;$3f;$fd;$e0;$00;$00;$00;$00;$f8;
$03;$e0;$00;$00;$7f;$ff;$e0;$00;$00;$00;$00;$f8;
$03;$e0;$00;$00;$ff;$ff;$e0;$00;$00;$00;$00;$f8;
$03;$e0;$00;$01;$fc;$3f;$e0;$00;$00;$00;$00;$f8;
$03;$e0;$00;$01;$f0;$0f;$e0;$00;$00;$00;$00;$f8;
$03;$e0;$00;$03;$e0;$07;$e0;$00;$00;$00;$00;$f8;
$03;$e0;$00;$03;$c0;$03;$e0;$00;$00;$00;$00;$f8;
$03;$e0;$00;$07;$c0;$03;$e0;$00;$00;$00;$00;$f8;
$01;$f0;$00;$07;$80;$01;$e0;$00;$00;$00;$01;$f0;
$01;$f0;$00;$07;$80;$01;$e0;$00;$00;$00;$01;$f0;
$01;$f0;$00;$07;$c0;$01;$e0;$00;$00;$00;$01;$f0;
$01;$f0;$00;$07;$c0;$01;$e0;$00;$00;$00;$01;$f0;
$01;$f8;$00;$03;$e0;$01;$e0;$00;$00;$00;$03;$f0;
$00;$f8;$00;$03;$e0;$01;$e0;$00;$00;$00;$03;$e0;
$00;$f8;$00;$01;$f0;$01;$e0;$00;$00;$00;$03;$e0;
$00;$fc;$00;$01;$f8;$00;$00;$00;$00;$00;$07;$e0;
$00;$7c;$00;$00;$fc;$00;$00;$00;$00;$00;$07;$c0;
$00;$7c;$00;$00;$7e;$00;$00;$00;$00;$00;$07;$c0;
$00;$3e;$00;$00;$3f;$00;$00;$00;$00;$00;$0f;$80;
$00;$3e;$00;$00;$1f;$80;$00;$00;$00;$00;$0f;$80;
$00;$3f;$00;$00;$0f;$80;$00;$00;$00;$00;$1f;$80;
$00;$1f;$80;$00;$07;$c0;$00;$00;$00;$00;$3f;$00;
$00;$1f;$80;$00;$07;$c0;$00;$00;$00;$00;$3f;$00;
$00;$0f;$c0;$00;$03;$e0;$00;$00;$00;$00;$7e;$00;
$00;$07;$e0;$00;$03;$e0;$00;$00;$00;$00;$fc;$00;
$00;$07;$e0;$00;$01;$e0;$00;$00;$00;$00;$fc;$00;
$00;$03;$f0;$00;$01;$e0;$00;$00;$00;$01;$f8;$00;
$00;$01;$f8;$00;$01;$e0;$00;$00;$00;$03;$f0;$00;
$00;$00;$fc;$00;$03;$e0;$00;$00;$00;$07;$e0;$00;
$00;$00;$fe;$00;$03;$c0;$00;$00;$00;$0f;$e0;$00;
```

```
$00;$00;$7f;$00;$07;$c0;$00;$00;$00;$1f;$c0;$00;
$00;$00;$3f;$80;$0f;$c0;$00;$00;$00;$3f;$80;$00;
$00;$00;$1f;$c0;$3f;$80;$00;$00;$00;$ff;$00;$00;
$00;$00;$0f;$f0;$ff;$80;$00;$00;$01;$fe;$00;$00;
$00;$00;$07;$ff;$ff;$00;$00;$00;$07;$fc;$00;$00;
$00;$00;$03;$ff;$fe;$00;$00;$00;$0f;$f0;$00;$00;
$00;$00;$01;$ff;$fc;$00;$00;$00;$3f;$e0;$00;$00;
$00;$00;$00;$7f;$f0;$00;$00;$01;$ff;$c0;$00;$00;
$00;$00;$00;$1f;$80;$00;$00;$0f;$ff;$00;$00;$00;
$00;$00;$00;$00;$00;$00;$00;$ff;$fe;$00;$00;$00;
$00;$00;$00;$00;$00;$00;$ff;$ff;$f8;$00;$00;$00;
$00;$00;$00;$00;$00;$00;$ff;$ff;$c0;$00;$00;$00;
$00;$00;$00;$00;$00;$00;$ff;$ff;$00;$00;$00;$00;
$00;$00;$00;$00;$00;$00;$ff;$f8;$00;$00;$00;$00;
$00;$00;$00;$00;$00;$00;$ff;$00;$00;$00;$00;$00;
TRM;¥
```

The following example shows the logo command for the above data in compressed format.

```
~^″LOGO″;5;0;39;170;SPB;UTOF;0512;HBR;160;VBR; 20;
LOGD;1;12;87;
00000000001fc00000000000000000000003ffc00000000000
000000001fffc000000000000000000007fffc00000000000
00000003ffffc00018000000000000000fffe000001e000000
0000001ffe0000001f0000000000007ff00000001fc00000
000000ff800000001fe00000000001fe000000001ff00000
000007fc000000001ffc000000000ff0000000001ffe0000
00001fe0000000001fff000000003f80000000001f3f8000
00007f00000000001f1fc0000000fe00000000001f0fe000
0000fc00000000001f07e0000001f800000000001f03f000
0003f000000000001f01f8000007e000000000001f00fc00
0007e000000000001f00fc00000fc000000000001f007e00
001f80000001e0001f003f00001f80000001e0001f003f00
003f00000001e0001f001f80003e00000001e0001f000f80
003e00000001f0003f000f80007c00000001f0003e0007c0
007c00000001f8007e0007c000fc00000001fc00fc0007e0
00f800000001ff03f80003e000f800000001fffff00003e0
01f800000001ffffe00003f001f000000001efffc00001f0
```

```
01f000000001e3ff000001f001f000000001e000000001f0
01f000000001e000000001f003e000000001e000000000f8
03e000000001e000000000f803e000000001e000000000f8
03e000000001e000000000f803e000000ff1e000000000f8
03e000003ffde000000000f803e000007fffe000000000f8
03e00000ffffe000000000f803e00001fc3fe000000000f8
03e00001f00fe000000000f803e00003e007e000000000f8
03e00003c003e000000000f803e00007c003e000000000f8
01f000078001e000000001f001f000078001e000000001f0
01f00007c001e000000001f001f00007c001e000000001f0
01f80003e001e000000003f000f80003e001e000000003e0
00f80001f001e000000003e000fc0001f8000000000007e0
007c0000fc00000000000007c0007c00007e00000000007c0
003e00003f00000000000f80003e00001f80000000000f80
003f00000f80000000001f80001f800007c0000000003f00
001f800007c0000000003f00000fc00003e0000000007e00
0007e00003e000000000fc000007e00001e000000000fc00
0003f00001e000000001f8000001f80001e000000003f000
0000fc0003e000000007e0000000fe0003c00000000fe000
00007f0007c00000001fc00000003f800fc00000003f8000
00001fc03f80000000ff000000000ff0ff80000001fe0000
000007ffff00000007fc0000000003fffe0000000ff00000
000001fffc0000003fe000000000007ff0000001ffc00000
0000001f8000000fff00000000000000000000fffe000000
000000000000fffff8000000000000000000ffffc0000000
000000000000ffff00000000000000000000fff800000000
000000000000ff0000000000;
TRM;¥
```

14.Extended Memory Operations

In addition to the command mode in which, as described in previous chapters, various label formatting commands are sent to the printer using RCL, there is also an extended memory mode, in which label data or logo data can be previously written to extended memory. In the extended memory mode, it is possible to print simply by referring to the data from the data field within the format. This chapter describes the commands required for operation in the extended memory mode.

14.1. Writing a Format to Extended Memory

A label format to be written to extended memory is created with RCL. The printer can write the label format directly to extended memory. The procedure for writing the label format data to extended memory in the printer is as follows.

1. The label format data to be saved in extended memory is created using RCL. In labels with a text or bar code field (variable field) in which the data is determined at printing time, in place of the normal "---" an FLD command is used.
2. The download start command (FS: 1CH) is sent. When the printer receives the FS command, until it receives a download end (EM: 19H) command, the whole command sequence is written to extended memory.
3. Send the command sequence created in Step 1 from the host. A number of label formats can be saved in a single extended memory card. When sending a number of formats to be written to a card, download the formats one at a time, thus: (FS: 1CH) (format) (EM: 19H). If the extended memory becomes full during writing, the printer error LED lights, indicating an extended memory overflow.

Note
The download start and end commands are one-byte control codes: send 1C$_H$ (hexadecimal) to start and 19$_H$ (hexadecimal) to end.

In other words, the command sequence for downloading sent from the host is as follows.

(1C$_H$)

```
   ~ ^"file1.RCL";...;TRM;¥
```

(19$_H$)

(1C$_H$)

```
    ~^"file2.RCL";...;TRM;¥
```

(19$_H$)

(1C$_H$)

```
   ~ ^"file3.RCL";...;TRM;¥
```

(19$_H$)

Note

- Be sure to include just one format between the (FS: $1C_H$) and the (EM: 19_H).
- The file name extension (RCL) may be omitted. The extension (RLC) will be added to the file name in the printer when it is written to extended memory.
- Limit file names to eight characters.

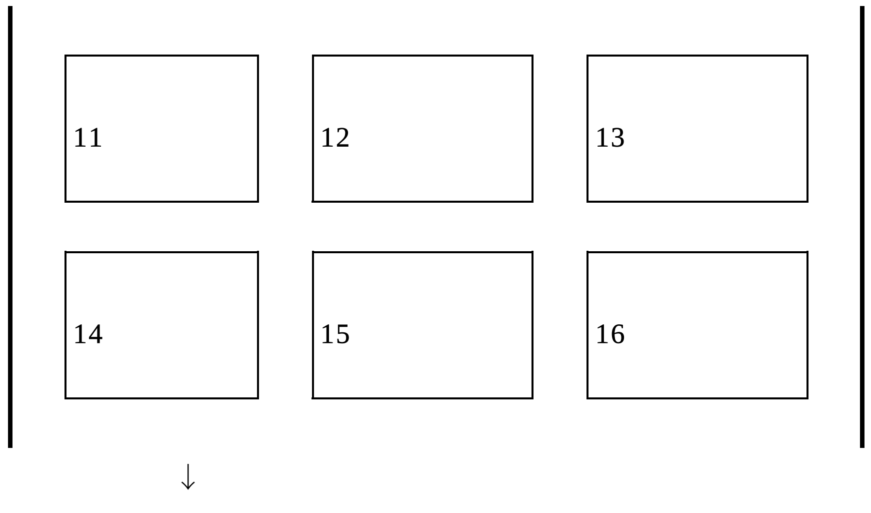
14.1.1. Field (FLD)

The FLD command is used specify a variable data field. The syntax of the FLD command is as follows.

FLD;(+/-offset)<user prompt;NN>=field name

offset:	integer from 0 to 32,767
user prompt :	character string to be sent in response to the LOAD command
NN:	two-digit number specifying maximum number of characters in field
field name:	field name referenced by the FLD command

The offset parameter relates to printing when a number of labels are arranged side by side.
Refer to the numbers in the following diagram.



Direction of label feed

If there are three labels in a row, two rows deep, for a total of six labels including serial numbers, then when printing, the labels arranged side by side must be formatted as a set. In this case, the three labels arranged side by side are treated as a set, and there must be a series of commands such that for each of these labels the increment is +3. If the serial number field being printed is a variable data field, an initial value must be assigned to the variable data field for each of the first three labels to be printed. However, in an example such as the one shown, since the initial values in the second and third fields are determined by the value in the first field, it is necessary to transfer the first (reference) field from the host, and the others are determined by the offset from this reference value.

In the figure above, if label number 11 is taken as the reference, the initial values for labels 12 and 13 can be specified by the offset in the FLD command. In other words, this is as follows.

```
FLD;(+1),FLD;(+2)
```

The syntax of the FLD command for the reference label is FLD;(+0).

14.2. Printing in Extended Memory Mode

In extended memory mode label formats are previously stored in extended memory, and then a format file in extended memory is specified by the host, an initial value for variable fields in the format are entered, and the number of labels to be printed is specified. These commands are extended memory communications commands. This section describes the three extended memory communications commands and the DIR and LOAD commands for checking on the contents written to extended memory.

14.2.1. File Name Specification Command (*)

This extended memory communications command specifies the file in extended memory to be printed. The command format is simple, consisting of an asterisk followed by the file name. There must be a carriage return ($0D_H$) at the end of the file name.

```
*filename <CR>
 (*filename.RCL <CR>)
```

After powering on, and after the end of a printing operation, when the system is waiting for a command, the printer accepts either an RCL header command or an asterisk. When it receives a header command it goes into command mode, and when it receives an asterisk into extended memory mode. Note that the extended memory communications commands and RCL command mode commands cannot be mixed.

If the file indicated after the asterisk is not present in extended memory, an extended memory error results.

14.2.2. Variable Data Field Initial Value Setting Command (/)

This extended memory communications command specifies the initial value for a variable data field in the format. The syntax of this command is as follows.

```
/fieldname=initial_value<CR>
```

The parameter fieldname is the field name entered in the FLD command when the format data is written to extended memory.

If there is more than one variable data field in a single file, one '/' command is required for each. There must be a carriage return ($0D_H$) at the end of each line.

```
/fieldname1=initial value1<CR>
/fieldname2=initial value2<CR>
/fieldname3=initial value3<CR>
```

14.2.3. Batch Cut Number (%)

The third extended memory communications command specifies the batch cut number. This is also a one-byte character. There must be a carriage return ($0D_H$) at the end of the line.

%N 〈CR〉
    N = integer from 1 to 999999

When the printer receives the '%' command, it sets the batch cut number.

14.2.4. Print Count Specification Command (!)

This third extended memory communications command specifies the print count. This is also a one-byte character. There must be a carriage return ($0D_H$) at the end of the line.

```
! N <CR>
```

N = integer from 1 to 999999

When it receives the '!' command, the printer starts printing. If no initial value has been specified for a variable data field, the field is treated as though there were no data (Null).

The following is an example label format using the FLD command.

```
#
 SAMPLE PROGRAM IC CARD
  LABEL HEIGHT      6 mm    LENGTH 50 mm
  LABEL PITCH 9 mm


#;
```

```
#Start of download #;

# One-byte control code (1C_H) [cannot be seen on printout]#;
  ~^"ICCARD";1;0;24;187;SPB;UTOF;0354;
BSYM; 1; 1;BNEW; 3;BWEW; 7;BICG; 3;
DDF; 6;10;DFM; 1; 1;DFS; 3;DFO; 1; 1;
MRK;
 HBR; 0;VBR; 0;DWBX; 0; 0;394; 24;NUM;
 BCLC; 1;BCID; 1;
 IDF; 1;
  HBR; 33;VBR; 14;BCSH; 18;
  BCST;
   "*";FLD;(0)<;11>=DATA;BSAL;2;"*";
  BSTP;
 HBR; 66;VBR; 15;
     FLD;(0)<;11>=DATA;SAL;2;
RET;
TRM;¥
#End of download #;
# One-byte control code (19_H) [cannot be seen on printout]#;
```

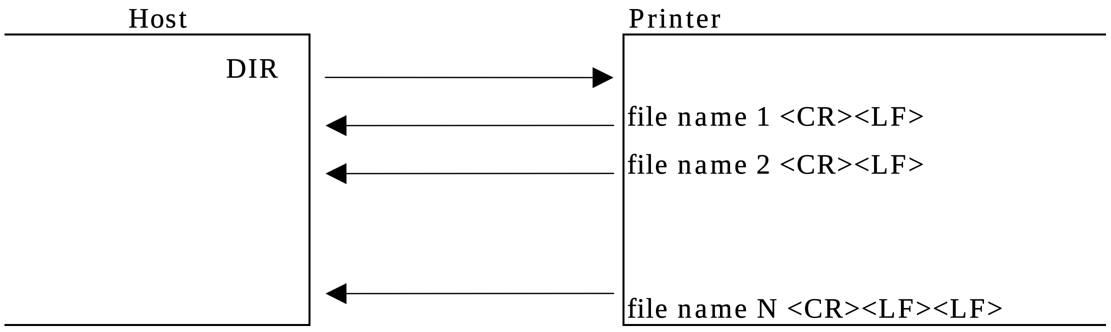
The following shows an example of extended memory communications commands with data applied to this format.

```
*ICCARD <CR>
/DATA=ABCDEFGHI00<CR>
!10<CR>
```

Note
The function setting command cannot be saved in extended memory.

14.2.5. File Name Request Command (DIR)

The DIR command is used to obtain a list of the names of files saved in extended memory. When the DIR command is sent from the host, the printer returns the names of files saved in extended memory. This command requires duplex communications, and is therefore only supported by the serial interface. The DIR command and the corresponding response are shown in the following figure.

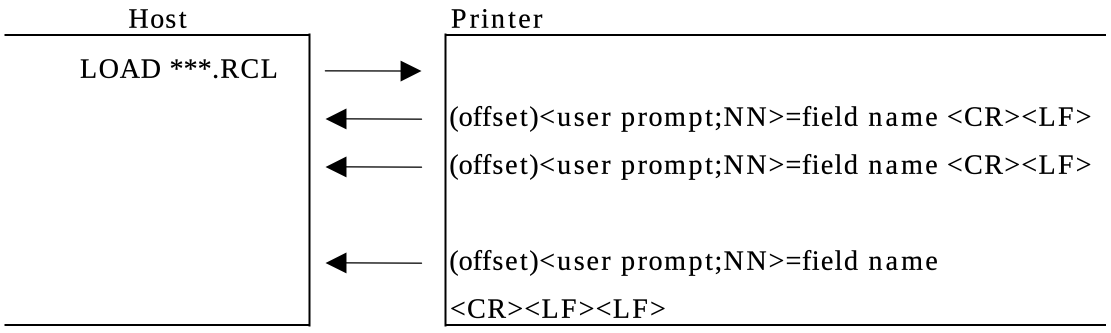


The file names are separated by CR ($0D_H$) and LF ($0A_H$). The last file name, is followed by one CR and two LF codes.

14.2.6. Field Name Request Command (LOAD)

In the same way that the DIR command returns file names, the LOAD command returns the contents of variable data fields specified by the FLD command. This applies only to the specified file. The LOAD command is only supported by the serial interface. The format of the LOAD command and its response is as follows.

LOAD(*filename*)

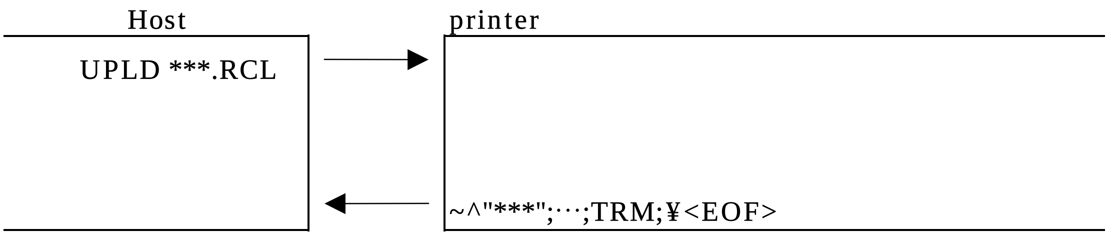

Host
Printer
LOAD ***.RCL
(offset)<user prompt;NN>=field name <CR><LF>
(offset)<user prompt;NN>=field name <CR><LF>
(offset)<user prompt;NN>=field name
<CR><LF><LF>

The user prompt, NN, and field name, are the values specified in the FLD command.

The field names are separated by CR ($0D_H$) and LF ($0A_H$). The last field name, is followed by one CR and two LF codes.

14.2.7. Format Upload Command (UPLD)

The UPLD command is used to upload a format stored in extended memory. The UPLD command is only supported by the serial interface. The format of the UPLD command and its response is as follows.



The last character of the upload data is EOF ($1A_H$), allowing the file to be saved.

14.2.8. Delete File Command (DELF)

The DELF command is used to delete a file from extended memory.
Specify the file name including the extension.

Example

```
DELF TEST.RCL
```

14.3. Writing a Logo to Extended Memory

It is possible to write logo data to extended memory.
Once logo data has been written to extended memory, printing can start faster than if the logo data has to be sent from the host.
The logo data written to extended memory is handled as a file (extension .RLG).

14.3.1. Writing a Logo File

・Command (MAX): Horizontally: total head dots (dots) × Vertically: number of lines in maximum printing length (dots)

(1)Command format: (1CH) ”file ”;x;y;n;data ; (19H)
Specify the file name including the extension.
x: number of bytes in 1 row (x = 1 to (total dots)/8)
y: number of rows of data (y = 1 to maximum printing length lines)
n: number of logos (n = 1 or more)

(2) Number of logos
A number of logos may be included in one file if they are all the same size.
It is, however, necessary for the user to keep track of which logo is which. (Normally, set n = 1.)

(3) Data portion

- The format of the logo data is decimal representation or hexadecimal representation with an attached '$'. This is the same as conventional logo data.

- As the printer writes the data between the (1CH) and the (19H) to extended memory, it checks for insufficient data, but does not check whether the data is correct.

(4) Example of logo data

```
                                                                      Bytes
   7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0   1    2    3    4
 1 □□□□□□□■■■■■■■■■■■■■■■■■■■□□□□□□ 01H  FFH  FFH  COH
 2 □□□□□□■■■■■■■■■■■■■■■■■■■■■□□□□□ 03H  FFH  FFH  EOH
 3 □□□□□■■■□□□□□□□□□□□□□□□□□■■■□□□□ 07H  00H  00H  70H
 4 □□□□■■■□□□□□□□□□□□□□□□□□□□□■■■□□□ 0EH  00H  00H  38H
 5 □□□■■■□□□□□□■■■■■■■■■□□□□□□■■■□□ 1CH  0FH  F8H  1CH
 6 □□□■■■□□□□□■■■■■■■■■■■□□□□□■■■□□ 1CH  1FH  FCH  1CH
 7 □□□■■■□□□□■■■□□□□□□□■■■□□□□■■■□□ 1CH  38H  0EH  1CH
 8 □□□■■■□□□■■■□□□□□□□□□■■■□□□■■■□□ 1CH  70H  07H  1CH
 9 □□□■■■□□□■■■□□□□□□□□□□□□□□□■■■□□ 1CH  70H  00H  1CH
10 □□□■■■□□□■■■□□□□□□□□□□□□□□□■■■□□ 1CH  70H  00H  1CH
11 □□□■■■□□□■■■□□□□□□□□□□□□□□□■■■□□ 1CH  70H  00H  1CH
12 □□□■■■□□□■■■□□□□□□□□□■■■□□□■■■□□ 1CH  70H  00H  1CH
13 □□□■■■□□□□■■■□□□□□□□■■■□□□□■■■□□ 1CH  38H  0EH  1CH
14 □□□■■■□□□□□■■■■■■■■■■■□□□□□■■■□□ 1CH  1FH  FCH  1CH
15 □□□■■■□□□□□□■■■■■■■■■□□□□□□■■■□□ 1CH  0FH  F8H  1CH
16 □□□□■■■□□□□□□□□□□□□□□□□□□□□■■■□□□ 0EH  00H  00H  31H
17 □□□□□■■■□□□□□□□□□□□□□□□□□■■■□□□□ 07H  00H  00H  70H
18 □□□□□□■■■■■■■■■■■■■■■■■■■■■□□□□□ 03H  FFH  FFH  EOH
19 □□□□□□□■■■■■■■■■■■■■■■■■■■□□□□□□ 01H  FFH  FFH  COH
```

```
(1CH)
"LOGFILE1.RLG";4;19;1;
1;$FF;$FF;$C0;
3;$FF;$FF;$E0;
7;0;0;$70;
$0E;0;0;$38;
$1C;$0F;$F8;$1C;
$1C;$1F;$FC;$1C;
$1C;$38;$0E;$1C;
$1C;$70;$07;41C;
$1C;$70;0;$1C;
$1C;$70;0;$1C;
$1C;$70;0;$1C;
$1C;$70;0;$1C;
$1C;$38;$0E;$1C;
$1C;$1F;$FC;$1C;
$1C;$0F;$F8;$1C;
$0E;0;0;$31;
7;0;0;$70;
3;$FF;$FF;$E0;
1;$FF;$FF;$C0;
(19H)
```

For example: writing above logo data in decimal notation to extended memory

(5) Errors

- If the printer is not equipped with extended memory, then an extended memory error results.

14.3.2. Logo Printing

(1) Command format: `LOGF;"file name ";`

Specify the file containing the logo to be used (file name specified when the logo was saved).
The file extension may be omitted.

`XLOG;n;`

Logo number: position of the logo in the file specified in LOGF (n = 1 or more).

(2) Command example:

```
~^"LOGOTEST";1;0;100;0;
SPB;UTOF;1100;HBR;100;VBR;50;
LOGF;"LOGFILE1";  ←──────────Use logo in file specified:
                                  "LOGFILE1.RLG"


XLOG;1;  ←────────────────────Use first logo in file
TRM;¥
```

The logo is rendered at the position specified by HBR and VBR.

- When using a logo, use "LOGF" and "XLOG" as a pair.
- If there is no file specification, an extended memory error occurs.
- When a logo in another logo file has been used, the logo file must be specified again with "LOGF".
- There are no restrictions on the image size of a logo file which can be rendered. Any logo data can be created as long as it does not exceed the extended memory size and rendered label size. Saving a logo file to extended memory and recalling it with a LOGF command offers the following features, compared with writing the logo directly in the RCL command sequence with a LOGO command.
  1. The logo image data does not need to be transferred, and the printing start is faster.
  2. The logo file data does not pass through the CMDBUF buffer in the way that the LOGO command does, and is written directly to the image buffer, thus avoiding the command length restrictions relating to the CMDBUF size.

```
~^"FileA";1;0;100;0;
```

```
SPB;
LOGF;"LogoA";     ← No limit on logo size in LogoA.RLG
XLOG;1;
TRM;¥
```

(3) Example of using a number of logo files in a single label

Between the RCL command header and the buffer processing end (TRM;¥) or break (BRK;¥) the logo data from a number of logo files can be used. However, there is a limit on the maximum number of files that can be open at a time. For example, the example given is of the case in which the number of files which can be opened simultaneously (the maximum number of open extended memory files) is eight.

1. When seven or fewer logos are to be used on a label, it is possible to recall the seven logo files as shown below. Note that one file is already open, since the RCL file is already open.

```
~^"FileA";1;0;100;0;
SPB;
HBR;10;VBR;10;LOGF;"Logo1";XLOG;1;
HBR;20;VBR;20;LOGF;"Logo2";XLOG;1;
HBR;30;VBR;30;LOGF;"Logo3";XLOG;1;
HBR;40;VBR;40;LOGF;"Logo4";XLOG;1;
HBR;50;VBR;50;LOGF;"Logo5";XLOG;1;
HBR;60;VBR;60;LOGF;"Logo6";XLOG;1;
HBR;70;VBR;70;LOGF;"Logo7";XLOG;1;
TRM;¥        The TRM;¥ command closes the seven open logo files
```

2. To use eight or more logos on a label, the format file must be split up, to create the label image by overlaying the format files. The next example uses two format files stored in extended memory to print using fourteen logos.

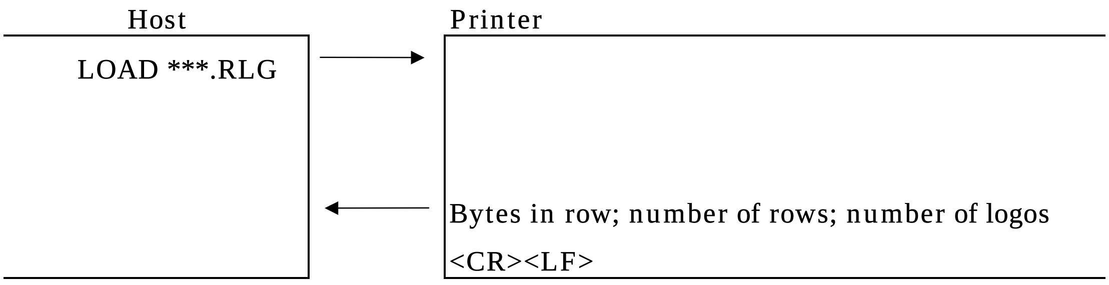
```
(1CH)
~^"FileC";1;0;100;0;
SPB;
HBR;10;VBR;10;LOGF;"Logo1";XLOG;1;
HBR;20;VBR;20;LOGF;"Logo2";XLOG;1;
HBR;30;VBR;30;LOGF;"Logo3";XLOG;1;
HBR;40;VBR;40;LOGF;"Logo4";XLOG;1;
HBR;50;VBR;50;LOGF;"Logo5";XLOG;1;
HBR;60;VBR;60;LOGF;"Logo6";XLOG;1;
HBR;70;VBR;70;LOGF;"Logo7";XLOG;1;
```

```
BRK;¥
(19H)
(1CH)
~^”FileD”;1;0;100;0;
RSPB;
HBR;10;VBR;20;LOGF;”Logo8”;XLOG;1;
HBR;20;VBR;30;LOGF;”Logo9”;XLOG;1;
HBR;30;VBR;40;LOGF;”Logo10”;XLOG;1;
HBR;40;VBR;50;LOGF;”Logo11”;XLOG;1;
HBR;50;VBR;60;LOGF;”Logo12”;XLOG;1;
HBR;60;VBR;70;LOGF;”Logo13”;XLOG;1;
HBR;70;VBR;10;LOGF;”Logo14”;XLOG;1;
TRM;¥
(19H)
```

```
*FileC
!1            When FileC is used for image rendering, seven logo files are open.
*FileD
!1            When FileD is used for image rendering, seven logo files are open.
              On the printed labels, the fourteen logo file images are used.
```

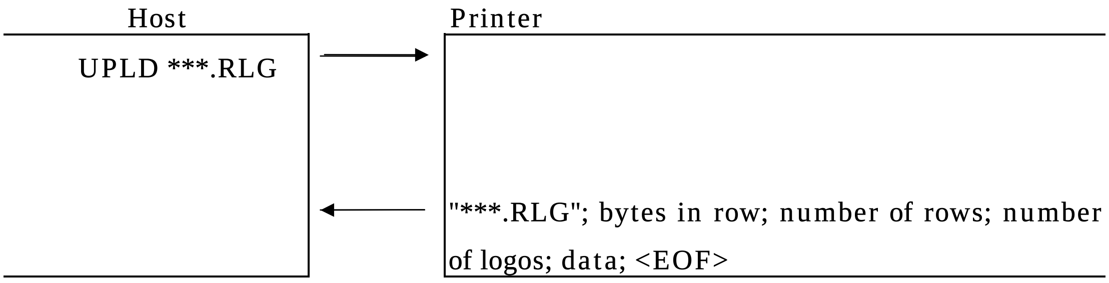
14.3.3. Logo Size Request Command (LOAD)

The logo LOAD command returns the number of bytes in a row, the number of data items, and the number of logos.



14.3.4. Logo Data Upload (UPLD)

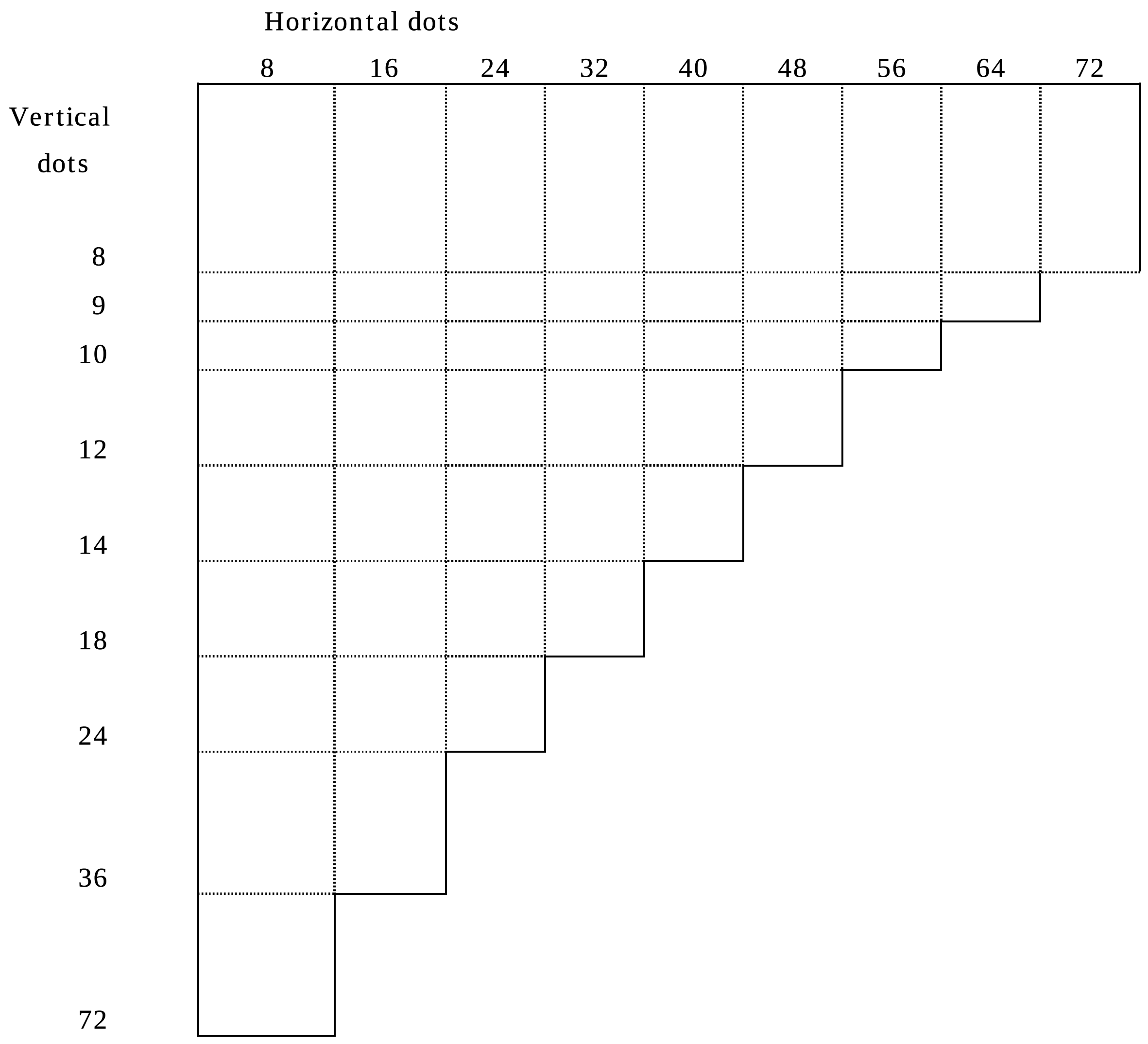
The logo UPLD command is used to upload logo data saved in extended memory.


Host
Printer
UPLD ***.RLG
"***.RLG"; bytes in row; number of rows; number of logos; data; <EOF>

14.4. Writing External Font (Gaiji) Files

User-designed dot font data can be stored as an external font file, and printed using the same procedure as for the internal dot fonts. Characters can also be rotated or zoomed.
The font sizes which can be stored are the combinations of horizontal by vertical sizes within the outline in the figure below.

External font sizes (horizontal and vertical dot sizes of a character1)



The command for saving an external font file is as follows.

<FS>“file name ”;
0;X;Y;C1;N1;C2;N2;.....;$FF;data ;<EM>

Specify the file name including the extension.
0: function setting flag. Always set to 0.

X: horizontal dot dimension of character
Y: vertical dot dimension of character
C1;N1;C2;N2;.....;: Specify area in code tables to store character. See the next item for details.
$FF: terminator of area specification portion
data : external font data. See the next item for details.

The horizontal dot dimension, vertical dot dimension, area specification and data values may be decimal or hexadecimal preceded by a dollar sign.

14.4.1. External Fonts and Code Tables

An external font is a dot font designed by the user, containing a set of characters in the same typeface, which can be saved in extended memory. The full ASCII character set allows 95 characters to be included, from code 20hex to 7Ehex. A set of numeric digits only, or capital letters only, or a combination thereof can also be created. It is also possible to include Japanese kana characters. The external fonts support single-byte codes, which can be from 20hex to 7Ehex, and from 80hex to 0FFhex.

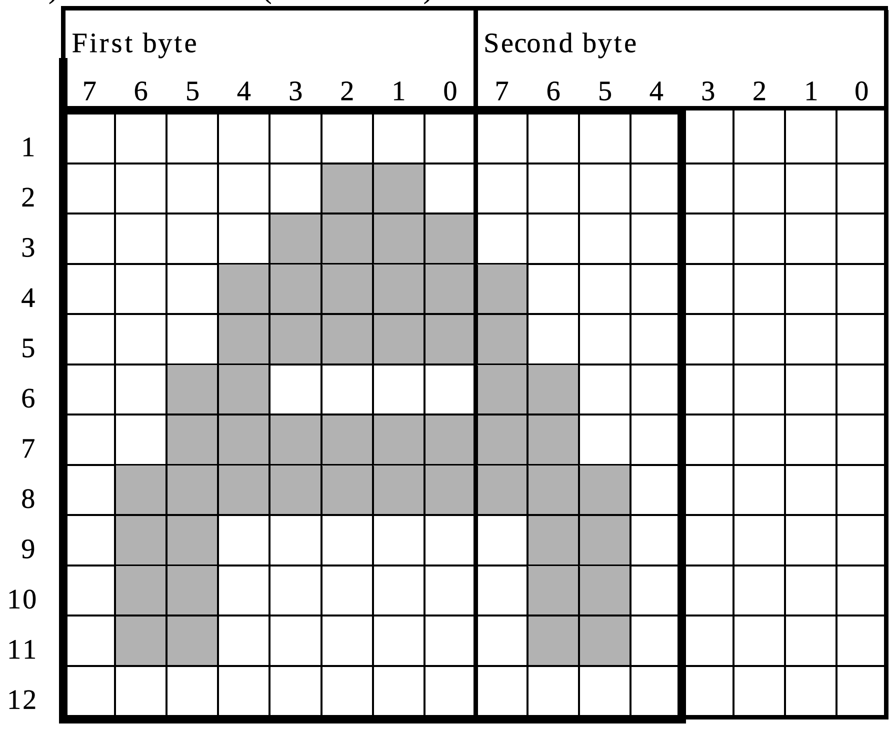
The definition of the code to which a character is assigned is carried out by the command parameters "Specify area in code tables to store character". Cx,Nx; indicates an area of Nx characters from character Cx. By a combination of the Cx;Nx; pairs any required set of character codes can be assigned, even if not contiguous. For example, to assign characters to the numeric digits (10 codes from 30hex) and capital letters (26 codes from 41hex), for a total of 36 characters, the following command is used for saving.

```
<FS>“file name ”;0;X;Y;$30;10;$41;26;$FF;data ;<EM>
```

14.4.2. Data Portion

The data for one character must always be specified in byte units. The external font data must also be supplied with the characters in increasing character code order. The following example shows the data for a 12 x 12 dot capital 'A' to be saved in an external font.

'A' (41h) character data (12×12 dots)



	First byte	Second byte
1	→ 00 H	00 H
2	→ 06 H	00 H
3	→ 0F H	00 H
4	→ 1F H	80 H
5	→ 19 H	80 H
6	→ 30 H	C0 H
7	→ 3F H	C0 H
8	→ 7F H	E0 H
9	→ 60 H	60 H
10	→ 60 H	60 H
11	→ 60 H	60 H
12	→ 00 H	00 H

The actual font size is 12×12 dots, but to put the row ends on byte boundaries, the data size for a character is 16×12 dots. The character data must be left justified with each row on a bye boundary.

The hexadecimal representation of the data for the "A" above is shown on the left in the figure below. For a full ASCII external font file, this is the data for 41h on the right of the figure. When the data for character codes other than 41h is created in the same way, and embedded in the command data, the entire external font registration command is as the whole of the right side of the figure.

```
$00;$00;
$06;$00;
$0F;$00;
$1F;$80;
$19;$80;
```

```
<FS>"GAIJI1.RFN";0;12;12;$20;95;$FF;
20h data (32 bytes)
21h data (32 bytes)
```

```
$30;$C0;                                :
$3F;$C0;                        41h data (32 bytes)
$7F;$E0;
$60;$60;
$60;$60;                    :
$60;$60;                        7Eh data (32 bytes)
$00;$00;
                              <EM>
```

Font size

The following restrictions apply to the font size when expanding or rotating an external font.

If the number of bytes in one row of a character of the external font is m, and the height in dots is n, then the following must hold:

For expansion:	$m \times n \leqq 1152$	(bytes)
For rotation:	$m \times n \leqq 184$	(bytes)

External font printing (FNTF)

Printing with an external font is achieved with a combination of the external font selection command (FNTF) and dot font command (DDF). The external font selection command specifies which of a number of external font files which have been saved is to be used. This command is, however, necessary even if there is only one external font file saved in extended memory. On the other hand, if the FNTF command is issued a number of times with different values for the variable N "DDF setting value" then a number of different external fonts can be used in a label.

FNTF;"file name ";N;
N: DDF setting value, any integer in range 200 to 299.
file name: name of the external font file to be used. Use the file name specified when the font was registered. The file name extension may be omitted.

DDF;A;B;
A: specify dot font. Use a value of N specified with FNTF.
B: Currently has no significance. Specify 10.

The following example uses the external font selection command to assign the external font file "GAIJI1.RFN" to DDF;200;.

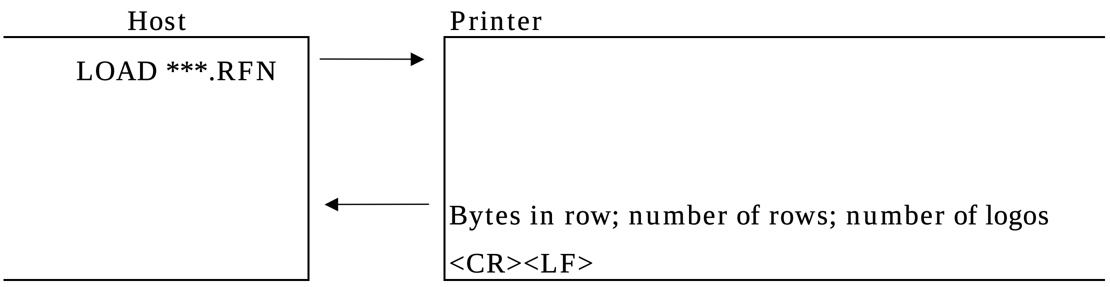
```
~^"GAIJITST";1;0;100;0;
SPB;HBR;100;VBR;50;
```

```
FNTF;"GAIJI1";200;
DDF;200;10;
DFM;1;2;DFO;2;2;DFS;3;
"GAIJI FONT TEST PRINT";
TRM;¥
```

Positioning with an external font is carried out using the HBR and VBR commands.
To use an external font, make the FNTF specification first, and then the dot font selection (DDF).
If there is no FNTF specification, printing will be carried out with the dot font default selection of alphanumeric Ｍ (DDF;3;).

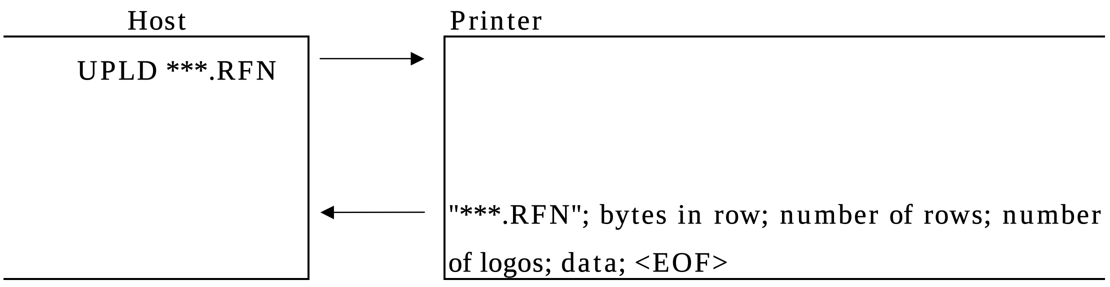
14.4.3. External Font Size Request Command (LOAD)

The external font LOAD command returns the number of bytes in a row, the number of data items, and the number of logos.



14.4.4. Logo Data Upload (UPLD)

The logo UPLD command is used to upload logo data saved in extended memory.



14.5. Notes on the Use of Extended Memory

In extended memory the internal data format of a format file is the same text format as in the RCL command sequence, but to increase processing speed during access, logo files are stored in a binary format. The size of the file in binary format is approximately 1/4 the file size transferred from the host.

The number of files which can be saved in extended memory depends on the printer model (the extended memory used). For details, see the printer specifications in the Appendix.

When recalling a format file stored in extended memory, the end of the file is detected by the backslash marking the end of the RCL command sequence. It is therefore only possible to include one RCL header and TRM;¥ (or BRK;¥) in a single format file.

A file name of a file to be saved in extended memory must use only the characters allowed in a DOS file name. Some of the characters allowed in an RCL command header file name are not allowed.

15.Cutter Function

This chapter describes functions relating to the cutter when the printer is equipped with a cutter.

15.1. Cutter Control Commands

The cutter control commands comprise the cutter on (CTON) and cutter off (CTOF) commands, and the batch size batch size (Number of Batch Cut: NUBC) command.

These commands are only valid for a printer equipped with a cutter, and are simply ignored if the printer has no cutter, or the cutter is disabled by the printer settings.

15.1.1. Cutter On (CTON)

The CTON command activates the cutter to cut printed labels.
When this command is issued, the labels are cut each time a batch of the size specified by the NUBC command has been printed. When the system is powered on, the default state is for the cutter to be on.

The syntax of this command is as follows.

```
CTON;
```

15.1.2. Cutter Off (CTOF)

The CTOF command stops cutting of labels by the cutter.
When this command is issued, the cutter does not operate. This remains the case until CTON command or CAN command is sent, or until the printer is powered off.

The syntax of this command is as follows.

```
CTOF;
```

Include either of the CTON and CTOF commands in a single batch*. If the command is omitted, the setting from the previous batch will continue in effect. If more than one specification is made in a batch, the latest specification takes effect.
* "Batch": here refers to an RCL command sequences enclosed between the header and a TRM;¥ (or BRK;¥) command.

15.1.3. Batch Size (NUBC)

The NUBC command (NUmber of Batch Cut) specifies the number of labels (cut units) printed before cutting. If this command is omitted, the parameter setting from the previous batch continues in effect. However, even before the number specified by this command is reached, when the number of labels specified in the header is reached, the cutter operates.

If the NUBC command is not specified, the default number of cut labels is 1.

The syntax of this command is as follows.

```
NUBC;N;
```

N = batch size (1 to 65536)

The following example shows cutting after printing the 4th, 8th, and 10th of a batch of 10.

```
~^”CT1”;10;0;100;0;
SPB;CTON;NUBC;4;
HBR;50;VBR;70;
BDEF;1;BCSH;50;BNEW;2;BWEW;5;BCST;
”*1234567890*”;
BSTP;
TRM;¥
```

16.Troubleshooting Guide

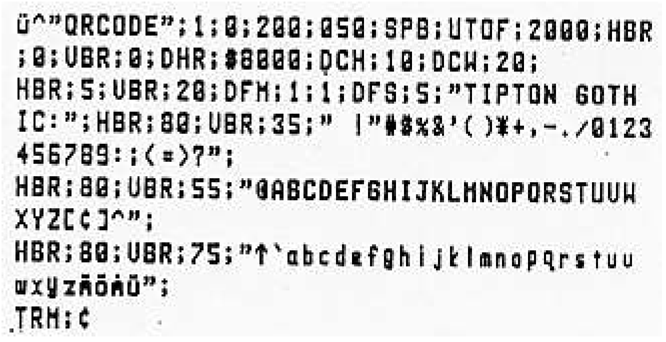
As with any other programming language, RCL produces errors in response to incorrect operations. The commonest errors are command syntax errors, typing errors, communications interface problems, and problems caused by incorrect layout of objects on the labels. This chapter describes the debugging method and hexadecimal dumps. and lists the commonest problems and their solutions.

16.1. Text Printing and Hexadecimal Dumps

By changing a printer setting, the printer can be made to print out the transmitted data in text form, or as a hexadecimal dump. (Use continuous forms of the maximum printer printing width, and set the printer form selection to the continuous mode.)

(1) Text printing

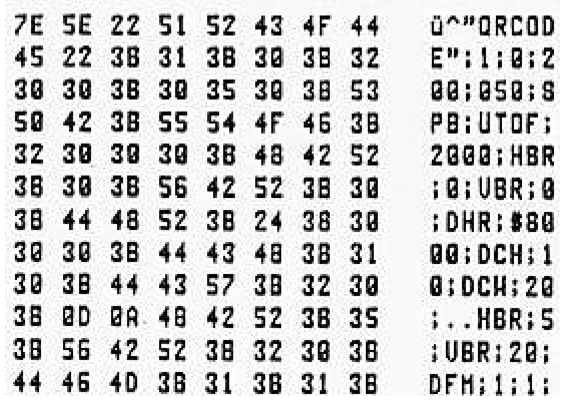
- For the method of switching the printer to text printing, refer to the operating instructions for the printer.
- Transfer the data from the host. The text printing occurs as follows.



```
Ü^"QRCODE";1;0;200;050;SPB;UTOF;2000;HBR
;0;UBR;0;DHR;#8000;DCH;10;DCW;20;
HBR;5;UBR;20;DFM;1;1;DFS;5;"TIPTON GOTH
IC:";HBR;80;UBR;35;" |"#$%&'( )¥+,-./0123
456789:;(=)?";
HBR;80;UBR;55;"@ABCDEFGHIJKLMNOPQRSTUVW
XYZ[¢]^";
HBR;80;UBR;75;"↑`abcdefghijklmnopqrstuv
wxyzÄÖÅÜ";
TRM;¢
```

(2) Hexadecimal dump

- For the method of switching the printer to text printing, refer to the operating instructions for the printer.
- Transfer the data from the host. The text printing occurs as follows.



```
7E 5E 22 51 52 43 4F 44   Ü^"QRCOD
45 22 3B 31 3B 30 3B 32   E";1;0;2
30 30 3B 30 35 30 3B 53   00;050;S
50 42 3B 55 54 4F 46 3B   PB;UTOF;
32 30 30 30 3B 48 42 52   2000;HBR
3B 30 3B 56 42 52 3B 30   ;0;VBR;0
3B 44 48 52 3B 24 38 30   ;DHR;$80
30 30 3B 44 43 48 3B 31   00;DCH;1
30 3B 44 43 57 3B 32 30   0;DCW;20
3B 0D 0A 48 42 52 3B 35   ;..HBR;5
3B 56 42 52 3B 32 30 3B   ;VBR;20;
44 46 4D 3B 31 3B 31 3B   DFM;1;1;
```

16.2. Commonly Occurring Problems

This section lists commonly occurring problems, with their causes and solutions.

16.2.1.1. Problem

Nothing prints when a command sequence is sent to the printer.

The error LED is not lit.

16.2.1.2. Cause

a. There is an unmatched quote mark (somewhere in the command sequence a closing quote (") has been omitted).

b. There is an incomplete comment (somewhere in the command sequence a closing hash (#) has been omitted from a comment).

c. The label printing length specified in the header is longer than the label length setting (UTOF parameter).

16.2.1.3. Solution

a. Find the unmatched quote mark and insert the missing closing quote.

b. Find the incomplete comment and insert the missing closing hash.

c. Check the label printing length specified in the header, and make it no longer than the label length setting (UTOF parameter).

16.2.1.4. Problem

When a command sequence is sent to the printer, instead of any labels, the command sequence or a part of it is printed in ASCII text format.

16.2.1.5. Cause

a. There is an unrecognized command or parameter.

b. There is a problem with the interface protocol between the host and the printer.

16.2.1.6. Solution

a. Look at the program preceding the point at which the ASCII text printing begins. When RCL encounters unrecognized data it switches to the text dump mode. In this case there may be a semicolon omitted, or a command argument omitted, or there may be too many arguments. Correct the error, and retransmit the program.

b. Check whether the handshake with the host computer is hardware or XON/XOFF. Check whether the host communications settings are correct. Check the cable connection between the host and the printer. If this does not resolve the problem, make a hexadecimal dump of the command sequence, and check the first line for protocol problems. To do this, compare the program with the hexadecimal dump, and if there are commands missing, for example, or parameters missing, there is a protocol problem.

16.2.1.7. Problem

When a command sequence is sent to the printer, instead of labels, a text dump of the commands is printed, beginning with the character sequence '^"FILE NAME";'.

16.2.1.8. Cause

a. The header start character is missing.

b. Possibly because of a parity error, the printer has not recognized the start character of the header.

c. The host has not transmitted an SOH character.

16.2.1.9. Solution

a. At the beginning of the command sequence, insert the header start character, SOH (01H) or '~'.

b. Check that the host and printer have the same parity settings.

    8 bits, 1 stop bit, no parity

    7 bits, 1 stop bit, even parity

    7 bits, 1 stop bit, odd parity

c. In place of SOH (01H), use a "twiddles" character '~' at the start of the header.

16.2.1.10. Problem

Even though the label stock is loaded correctly, a paper out or paper jam error occurs.

16.2.1.11. Cause

a. The sensor is not detecting the label stock.

b. The form length is not set correctly.

c. The transparency of the label stock is outside the specification limits of the printer.

16.2.1.12. Solution

a. Reload the label stock, and make sure that the carriage mechanism is correctly positioned.

b. Using the UTOF command, reset the form length. (More accurately)

c. Use a different type of label stock.

16.2.1.13. Problem

Almost all of the labels print correctly, but occasionally there are positioning errors.

16.2.1.14. Cause

The transparency of the paper is outside the specification limits of the printer.

16.2.1.15. Solution

Change the label stock.

16.2.1.16. Problem

When printing a series of labels, every other label is blank.

16.2.1.17. Cause

a. The print length specified in the header is the same length, or almost the same length as the label stock.

b. The form length is longer than the label stock being used.

16.2.1.18. Solution

a. Set the print length shorter than the label stock.

b. Using the UTOF command, reset the form length to be equal to the length of the label stock. (Very precisely)

16.2.1.19. Problem

Only part of the label is printed, and the lower portion is blank.

16.2.1.20. Cause

The print length in the header is too short, and the material to be printed on the lower part of the label is being lost.

16.2.1.21. Solution

a. Increase the print length in the header.
b. Reposition the objects which are not being printed, moving them upward on the label.

16.2.1.22. Problem

The first object to be printed at the top of the label (particularly text or a bar code) is not printed correctly.

16.2.1.23. Cause

If an object is positioned too close to the top of the label there may not be room to print it.

16.2.1.24. Solution

Increase the initial HBR or VBR setting, and move the first object downward.

16.2.1.25. Problem

Printed objects are printed over the left or right edges of the label stock, or objects are cut by the left or right edges of the label stock.

16.2.1.26. Cause

a. The left edge of the form is positioned too far away from the left edge of the label stock, so that there is not room to print objects close to the left edge of the label.
b. The HPR or HBR setting is not correct. Usually the value is too large.

16.2.1.27. Solution

a. Check that the left edge of the label stock using the left edge of the form is set correctly.
b. Check all of the HPR and HBR commands, and check that the values are correct. Reduce incorrect values, and retransmit.

16.2.1.28. Problem

When using the Tipton Gothic font, small text cannot be printed on a label.

16.2.1.29. Cause

The characters may be too small for this font. The Tipton Gothic font cannot print characters with a specification smaller than DCH;10;DCW;20;.

16.2.1.30. Solution

For characters with a specification smaller than DCH;10;DCW;20; use the rotatable font or dot font instead of the Tipton Gothic font.

16.2.1.31. Problem

Although Code39 or CodaBar bar codes can be scanned with the printer, they are not read correctly by another bar code reader.

16.2.1.32. Cause

a. The start and stop characters are incorrect.
b. The element width or density is incorrect.

16.2.1.33. Solution

a. Use an asterisk ('*') as the start and stop characters for a Code39 bar code bar code. This must be specified by the programmer.
b. Use A, B, C, and D as the start and stop characters for a CodaBar bar code. This must be specified by the programmer.
c. Compare the BWEW and BNEW command parameter values for Code39, or the BCPI command parameter for CodaBar with the latest specifications, and check that the correct values are specified in these commands.

16.2.1.34. Problem

When printing serial numbers in a bar code, the first label is printed correctly, but the remainder of the labels are invalid, and the bar code printed with a serial number cannot be scanned.

16.2.1.35. Cause

The DWBX (draw white box) command is not being used to erase the previous bar code, and the new bar code is therefore being overprinted on the old one.

16.2.1.36. Solution

a. Check that the DWBX command is being issued.
b. Check that the DWBX command is correctly positioned, and is erasing the whole of the bar code.
c. Enlarge the area erased by the DWBX command, to completely erase the bar code.

16.2.1.37. Problem

When printing a serial number in a bar code, the increment/decrement amount is incorrect, or before printing the serial number, an incorrect number of labels is printed.

16.2.1.38. Cause

a. A BCID and IDF or BCLC command appears after the serial number field in the command sequence, and therefore instead of the specified values the default values for these commands are being used.
b. The value specified in a BCID and IDF or BCLC command is incorrect or one of these commands is not present.

16.2.1.39. Solution

a. Place the BCID and IDF and BCLC command before the serial number field.
b. Recheck the command sequence, and confirm that the values specified in the BCID and IDF and BCLC commands are appropriate.

17.Command Summary

ALPH : ALPHabetic	Alphabetic serial number	11-6
APOF : APpc OFf	APPC off	8-2
APON : APpc ON	APPC on	8-2
AQL : Assure Quality Level	APPC level setting	8-2
BCID : Bar Code Increment/Decrement		11-8
BCLC : Bar Code Label Count		11-2
BCPI : Bar code Characters Per Inch		7-12, -15, -17, 9-4
BCSH : Bar Code Symbol Height		7-7, 9-4
BCST : Bar Code StarT		7-5, 9−3
BDEF : Bar code DEFinition		7-3, 9−1
BCIG : Bar Code Intercharacter Gap		7-8
BNEW : Bar Narrow Element Width		7-7
BOTH	Alphanumeric serial number	11-7
BRK : BReaK		3-2
BSAL : Bar Save Address Length		11-8
BSTP : Barcode SToP		7-5, 9-3
BSYM : Barcode SYMbol		7-2, 9-1
BVLP : Barcode Variable field Length and Position		11-9
BWEW : Bar Wide Element Width		7-7
CAN : CANcel	Initialize Printer	12-4
CTOF : CuTter Off		15-1
CTON : CuTter ON		15-1
DBBX : Draw Black BoX		6-3
DBOX : Draw BOX		6-2
DC2	Pause	12-4
DC4	End Pause	12-4
DCBX : Draw Complement BoX		6-4
DCH : Define Character Height		10-6
DCS : Define Character Size	For outline font	10-20
DCW : Define Character Width		10-6
DDF : Define Dot Font		10-14
DDL : Draw Diagonal Line		5-3
DELF : DELete File		14-7
DFM : Dot Font Magnification		10-16
DFO : Dot Font Orientation		10-17, -21

18.Printer Specifications

This chapter lists the maximum printing width and length for each model and the character size and barcode density for each model.

18.1. Specifications for Each Model

Model	R	SR	SRS	LSP5300 / LP5320
Head dot density	203DPI	400DPI	400DPI	300DPI
Linefeed pitch	0.005”	0.0025”	0.0025”	00328”
Total dots in head	1216	1536	1536	1536
Printing area length	0.24”—10.0”	0.2”—3.94”	0.2”—10.0”	0.79”—10.0”
Maximum printing width	6.0”	3.77”	3.77”	5.0”
Control buffer size	3K bytes	7K bytes	7K bytes	21K bytes
Verify function	Yes	Yes	No	No
OutLine Font	No (Option)	No (Option)	Yes	Yes
Label stock alignment	Centered	Centered	Centered	Centered
Extension memory type	Memory card	Memory card	Memory card	Flash memory card
Maximum file capacity	Depends on memory size	128	128	1024
Maximum files open	1	8	8	１６

19. Bar Code Densities for Individual Models

19.1. Units with 200DPI Heads (DURA PRINTER R)

19.1.1. Code39

Ladder bar codes

BCPI value	Narrow element width (inch)	Wide element width(inch)	Wide: narrow ratio	Density (CPI)
0	0.0098	0.0246	2.5:1	6.90
1	0.0098	0.0295	3.0:1	6.25
2	0.0148	0.0344	2.33:1	4.76
3	0.0148	0.0393	2.67:1	4.44
4	0.0197	0.0442	2.25:1	3.64

19.1.2. ITF

Ladder bar codes

BCPI value	Narrow element width (inch)	Wide element width (inch)	Wide: narrow ratio	Density (CPI)
0	0.0098	0.0246	2.5:1	12.50
1	0.0098	0.0295	3.0:1	11.11
2	0.0148	0.0344	2.33:1	8.70
3	0.0148	0.0393	2.67:1	8.00
4	0.0197	0.0442	2.25:1	6.67

19.1.3. UPC/EAN/JAN

Picket fence bar codes

UMAG	Module dimension (inch)	Magnifi-cation
0	0.0098	75.8
1	0.0123	94.7
2	0.0147	113.6
3	0.0172	132.6
4	0.0197	151.5
5	0.0221	170.5
6	0.0246	189.4
7	0.0270	208.3

Ladder bar codes

UMAG	Module dimension (inch)	Magnifi-cation
0	0.254	76.9
1	0.3175	96.2
2	0.381	115.5
3	0.4445	134.7
4	0.508	153.9

19.1.4. CodaBar

Picket fence bar codes

BCPI value	Narrow element width (inch)	Wide element width (inch)	Wide: narrow ratio	Density (CPI)
0	0.0098	0.0246	2.5:1	9.24
1	0.0098	0.0295	3.0:1	8.47
2	0.0148	0.0344	2.33:1	6.35
3	0.0148	0.0393	2.67:1	5.98
4	0.0197	0.0442	2.25:1	4.84
5	0.0197	0.0541	2.75:1	4.42

Ladder bar codes

BCPI value	Narrow element width (inch)	Wide element width (inch)	Wide: narrow ratio	Density (CPI)
0	0.0098	0.0246	2.5:1	9.09
1	0.0098	0.0295	3.0:1	8.33
2	0.0148	0.0344	2.33:1	6.25
3	0.0148	0.0393	2.67:1	5.88
4	0.0197	0.0442	2.25:1	4.76

19.1.5. Code93

Picket fence bar codes

BCPI value	Module dimension(inch)	Density (CPI)
0	0.00492	22.5
1	0.00984	11.29
2	0.01476	7.53
3	0.01968	5.64
4	0.0246	4.52

Ladder bar codes

BCPI value	Module dimension (inch)	Density (CPI)
0	0.00984	11.11
1	0.01476	7.41
2	0.01968	5.56
3	0.0246	4.44
4	0.02952	3.70

19.1.6. Code128

Picket fence bar codes

BCPI value	Module dimension (inch)	A and B density (CPI)	C density (CPI)
0	0.00492	18.47	36.95
1	0.00984	9.24	18.47
2	0.01476	6.16	12.32
3	0.01968	4.62	9.24
4	0.0246	3.69	7.39
5	0.02952	3.08	6.16
6	0.03444	2.64	2.64

Ladder bar codes

BCPI value	Module dimension (inch)	A and B density (CPI)	C density (CPI)
0	0.00984	9.09	18.18
1	0.01476	6.06	12.12
2	0.01968	4.55	9.09
3	0.0246	3.64	7.27
4	0.02952	3.03	6.06

19.1.7. Code49

Picket fence and ladder bar codes

BCPI value	Module dimension (inch)	Numeric only density (CPI)	Alphanumeric density (CPI)
0	0.00984	117.57	71.12
1	0.01476	78.38	47.47
2	0.01968	58.78	35.56
3	0.0246	47.03	28.45
4	0.02952	39.19	23.71

19.2. Units with 300DPI Heads (DURA PRINTER LSP5300/LP5320)

19.2.1. Code39

Ladder bar codes

BCPI value	Narrow element width (inch)	Wide element width(inch)	Wide: narrow ratio	Density (CPI)
0	0.00655	0.01639	2.5:1	10.5
1	0.00655	0.01967	3.0:1	9.5
2	0.00983	0.02295	2.33:1	7.2
3	0.00983	0.02623	2.67:1	6.7
4	0.01311	0.02951	2.25:1	5.5

19.2.2. ITF

Ladder bar codes

BCPI value	Narrow element width (inch)	Wide element width (inch)	Wide: narrow ratio	Density (CPI)
0	0.00655	0.01639	2.5:1	19.1
1	0.00655	0.01967	3.0:1	17.0
2	0.00983	0.02295	2.33:1	13.3
3	0.00983	0.02623	2.67:1	12.2
4	0.01311	0.02951	2.25:1	10.2

19.2.3. UPC/EAN/JAN

Picket fence bar codes

UMAG	Module dimension (inch)	Magnifi-cation
0	0.00655	50.5
1	0.00819	63.1
2	0.00983	75.7
3	0.01147	88.4
4	0.01311	101.0
5	0.01475	113.6
6	0.01639	126.2
7	0.01803	138.8

Ladder bar codes

UMAG	Module dimension (inch)	Magnifi-cation
0	0.00655	50.5
1	0.00819	63.1
2	0.00983	75.7
3	0.01147	88.4
4	0.01311	101.0

19.2.4. CodaBar

Picket fence bar codes

BCPI value	Narrow element width (inch)	Wide element width (inch)	Wide: narrow ratio	Density (CPI)
0	0.00655	0.01639	2.5:1	13.86
1	0.00655	0.01967	3.0:1	12.71
2	0.00983	0.02295	2.33:1	9.53
3	0.00983	0.02623	2.67:1	8.97
4	0.01311	0.02951	2.25:1	7.26
5	0.01311	0.03607	2.75:1	6.63

Ladder bar codes

BCPI value	Narrow element width (inch)	Wide element width (inch)	Wide: narrow ratio	Density (CPI)
0	0.00655	0.01639	2.5:1	13.86
1	0.00655	0.01967	3.0:1	12.71
2	0.00983	0.02295	2.33:1	9.53
3	0.00983	0.02623	2.67:1	8.97
4	0.01311	0.02951	2.25:1	7.26

19.2.5. Code93

Picket fence bar codes

BCPI value	Module dimension(inch)	Density (CPI)
0	0.00328	33.88
1	0.00655	16.94
2	0.00983	11.29
3	0.01311	8.47
4	0.01639	6.78

Ladder bar codes

BCPI value	Module dimension (inch)	Density (CPI)
0	0.00655	16.94
1	0.00983	11.29
2	0.01311	8.47
3	0.01639	6.78
4	0.01967	5.65

19.2.6. Code128

Picket fence bar codes

BCPI value	Module dimension (inch)	A and B density (CPI)	C density (CPI)
0	0.00328	27.72	55.44
1	0.00655	13.86	27.72
2	0.00983	9.24	18.48
3	0.01311	6.93	13.86
4	0.01639	5.54	11.09
5	0.01967	4.62	9.24
6	0.02295	3.96	7.92

Ladder bar codes

BCPI value	Module dimension (inch)	A and B density (CPI)	C density (CPI)
0	0.00655	13.86	27.72
1	0.00983	9.24	18.48
2	0.01311	6.93	13.86
3	0.01639	5.54	11.09
4	0.01967	4.62	9.24

19.2.7. Code49

Picket fence and ladder bar codes

BCPI value	Module dimension (inch)	Numeric only density (CPI)	Alphanumeric density (CPI)
0	0.00655	176.42	106.72
1	0.00983	117.61	71.15
2	0.01311	88.21	53.36
3	0.01639	70.57	42.69
4	0.01967	58.81	36.57

19.3. Units with 400DPI Heads (DURA PRINTER SR/SRS)

19.3.1. Code39

Ladder bar codes

BCPI value	Narrow element width (inch)	Wide element width(inch)	Wide: narrow ratio	Density (CPI)
0	0.01	0.025	2.5:1	6.90
1	0.01	0.03	3.0:1	6.25
2	0.015	0.035	2.33:1	4.76
3	0.015	0.04	2.67:1	4.44
4	0.02	0.045	2.25:1	3.64

19.3.2. ITF

Ladder bar codes

BCPI value	Narrow element width (inch)	Wide element width (inch)	Wide: narrow ratio	Density (CPI)
0	0.01	0.025	2.5:1	12.50
1	0.01	0.03	3.0:1	11.11
2	0.015	0.035	2.33:1	8.70
3	0.015	0.04	2.67:1	8.00
4	0.02	0.045	2.25:1	6.67

19.3.3. UPC/EAN/JAN

Picket fence bar codes

UMAG	Module dimension ()	Magnifi-cation
0	0.01	76.9
1	0.0125	96.2
2	0.015	115.5
3	0.0175	134.7
4	0.02	153.9
5	0.0225	173.2
6	0.025	192.4
7	0.0275	211.7

Ladder bar codes

UMAG	Module dimension (mm)	Magnifi-cation
0	0.01	76.9
1	0.0125	96.2
2	0.015	115.5
3	0.0175	134.7
4	0.02	153.9

19.3.4. CodaBar

Picket fence bar codes

BCPI value	Narrow element width (inch)	Wide element width (inch)	Wide: narrow ratio	Density (CPI)
0	0.01	0.025	2.5:1	9.09
1	0.01	0.03	3.0:1	8.33
2	0.015	0.035	2.33:1	6.25
3	0.015	0.04	2.67:1	5.88
4	0.02	0.045	2.25:1	4.76
5	0.02	0.055	2.75:1	4.35

Ladder bar codes

BCPI value	Narrow element width (inch)	Wide element width (inch)	Wide: narrow ratio	Density (CPI)
0	0.01	0.025	2.5:1	9.09
1	0.01	0.03	3.0:1	8.33
2	0.015	0.035	2.33:1	6.25
3	0.015	0.04	2.67:1	5.88
4	0.02	0.045	2.25:1	4.76

19.3.5. Code93

Picket fence bar codes

BCPI value	Module dimension(inch)	Density (CPI)
0	0.005	22.2
1	0.01	11.11
2	0.015	7.41
3	0.02	5.56
4	0.025	4.44
10	0.0075	14.81
11	0.0125	8.89
12	0.0175	6.35

Ladder bar codes

BCPI value	Module dimension (inch)	Density (CPI)
0	0.005	11.11
1	0.01	7.41
2	0.015	5.56
3	0.02	4.44
4	0.025	3.70
10	0.0075	14.81
11	0.0125	8.89
12	0.0175	6.35

19.3.6. Code128

Picket fence bar codes

BCPI value	Module dimension (inch)	A and B density (CPI)	C density (CPI)
0	0.005	18.18	36.36
1	0.01	9.09	18.18
2	0.015	6.06	12.12
3	0.02	4.55	9.09
4	0.025	3.64	7.27
5	0.03	3.03	6.06
6	0.035	2.60	5.19
10	0.0075	12.09	24.18
11	0.0125	7.26	14.52
12	0.0175	5.19	10.38

Ladder bar codes

BCPI value	Module dimension (inch)	A and B density (CPI)	C density (CPI)
0	0.01	9.09	18.18
1	0.015	6.06	12.12
2	0.02	4.55	9.09
3	0.025	3.64	7.27
4	0.03	3.03	6.06
10	0.0075	12.09	24.18
11	0.0125	7.26	14.52
12	0.0175	5.19	10.38

19.3.7. Code49

Picket fence and ladder bar codes

BCPI value	Module dimension (inch)	Numeric only density (CPI)	Alphanumeric density (CPI)
0	0.075	93.3	154.3
1	0.01	70.0	115.7
2	0.0125	56.0	92.6
3	0.015	46.7	77.1
4	0.0175	40.0	66.1

20. Dot Font Sizes for Individual Models

20.1. Units with 200DPI Heads (DURA PRINTER R)

DDF value	Font name	Width×height (dots)	Width×height (Inch)	Desc-enders (dots)	Character width×height (dots)	Character width×height (Inch)	Character set
1	Alpha-numeric XS	5×9	0.025 × 0.045	2	5×7	0.025 × 0.035	Alpha-numeric
2	Alpha-numeric S	7×14	0.035 × 0.070	3	7×11	0.035 × 0.055	Alpha-numeric
3	Alpha-numeric M	10×20	0.05 × 0.100	4	10×16	0.050 × 0.080	Alpha-numeric
4	Alpha-numeric L	15×30	0.075 × 0.150	6	15×24	0.075 × 0.120	Alpha-numeric
5	Alpha-numeric SS	6×8	0.030 × 0.040	—	6×8	0.030 × 0.040	Capitals and numeric
6	Pseudo OCR-B	14×27	0.070 × 0.135	5	14×22	0.070 × 0.110	Alpha-numeric
9	Alpha-numeric 2-byte kana	16×16	0.080 × 0.080	—	16×16	0.080 × 0.080	Alpha-numeric 2-byte kana
10	Alpha-numeric 1-byte kana	16×16	0.080 × 0.080	—	16×16	0.080 × 0.080	Alpha-numeric 1-byte kana
11	Alpha-numeric B	14×11	0.070 × 0.055	—	14×11	0.070 × 0.055	Alpha-numeric
12	Alpha-numeric N1	10×16	0.050 × 0.080	3	10×13	0.050 × 0.065	Alpha-numeric
13	Alpha-numeric N2	10×16	0.050 × 0.080	2	10×14	0.050 × 0.070	Alpha-numeric

20.2. Units with 300DPI Heads (DURA PRINTER LSP5300/LP5320)

DDF value	Font name	Width×height (dots)	Width×height (Inch)	Desc-enders (dots)	Character width×height (dots)	Character width×height (Inch)	Character set
1	Alpha-numeric XS	5×9	0.01640 × 0.02952	2	5×7	0.01640 × 0.02296	Alpha-numeric
2	Alpha-numeric S	7×14	0.02296 × 0.04591	3	7×11	0.02296 × 0.03607	Alpha-numeric
3	Alpha-numeric M	10×20	0.03280 × 0.06559	4	10×16	0.03280 × 0.05247	Alpha-numeric
4	Alpha-numeric L	15×30	0.04919 × 0.09839	6	15×24	0.04919 × 0.07871	Alpha-numeric
5	Alpha-numeric SS	6×8	0.01968 × 0.02624	—	6×8	0.01968 × 0.02624	Capitals and numeric
6	Pseudo OCR-B	14×27	0.04591 × 08855	5	14×22	0.04591 × 0.07215	Alpha-numeric
9	Alpha-numeric 2-byte kana	16×16	0.05247 × 0.05247	—	16×16	0.05247 × 0.05247	Alpha-numeric 2-byte kana
10	Alpha-numeric 1-byte kana	16×16	0.05247 × 0.05247	—	16×16	0.05247 × 0.05247	Alpha-numeric 1-byte kana
11	Alpha-numeric B	14×11	0.04591 × 0.03607	—	14×11	0.04591 × 0.03607	Alpha-numeric
12	Alpha-numeric N1	10×16	0.03280 × 0.05247	3	10×13	0.03280 × 0.04263	Alpha-numeric
13	Alpha-numeric N2	10×16	0.03280 × 0.05247	2	10×14	0.03280 × 0.04591	Alpha-numeric
14	Alpha-numeric NO1	12×20	0.03935 × 0.06559	2	11×18	0.03607 × 0.05903	Alpha-numeric
15	Alpha-numeric NO2	12×20	0.03935 × 0.06559	—	12×20	0.03935 × 0.06559	Alpha-numeric

16	Alpha-numeric NO3	15×24	0.04919 × 0.07871	2	15×22	0.04919 × 0.07215	Alpha-numeric
17	Alpha-numeric NO4	20×32	0.06559 × 0.10494	2	20×30	0.06559 × 0.09839	Alpha-numeric
18	Alpha-numeric NO5	24×36	0.07871 × 0.11806	2	23×34	0.07543 × 0.11150	Alpha-numeric
19	Alpha-numeric NA1	16×20	0.05247 × 0、06559	2	16×18	0.05247 × 0.05903	Alpha-numeric
20	Alpha-numeric NA2	20×28	0.06559 × 0.09183	4	20×24	0.06559 × 0.07871	Alpha-numeric

* For LP5320, the DDF values from 1 to 10 are only provided.
Also, the DDF 9 is changed to Pseudo OCR-A as shown below.

9	Pseudo OCR-A	15×22	1.25 × 1.83	—	15×22	1.25 × 1.83	Alpha-numeric

20.3. Units with 400DPI Heads (DURA PRINTER SR/SRS)

DDF value	Font name	Width×height (dots)	Width×height (Inch)	Desc-enders (dots)	Character width×height (dots)	Character width×height (Inch)	Character set
1	Alpha-numeric XS	5×9	0.0125 × 0.0225	2	5×7	0.0125 × 0.0175	Alpha-numeric
2	Alpha-numeric S	7×14	0.0175 × 0.0350	3	7×11	0.0175 × 0.0275	Alpha-numeric
3	Alpha-numeric M	10×20	0.0250 × 0.0500	4	10×16	0.0250 × 0.0400	Alpha-numeric
4	Alpha-numeric L	15×30	0.0375 × 0.0750	6	15×24	0.0375 × 0.0600	Alpha-numeric
5	Alpha-numeric SS	6×8	0.0150 × 0.0200	—	6×8	0.0150 × 0.0200	Capitals and numeric
6	Pseudo OCR-B	14×27	0.0350 × 0.0675	5	14×22	0.0350 × 0.0550	Alpha-numeric
9	Alpha-numeric 2-byte kana	16×16	0.0400 × 0.0400	—	16×16	0.0400 × 0.0400	Alpha-numeric 2-byte kana
10	Alpha-numeric 1-byte kana	16×16	0.0400 × 0.0400	—	16×16	0.0400 × 0.0400	Alpha-numeric 1-byte kana
11	Alpha-numeric B	14×11	0.0350 × 0.0275	—	14×11	0.0350 × 0.0275	Alpha-numeric
12	Alpha-numeric N1	10×16	0.0250 × 0.0400	3	10×13	0.0250 × 0.0325	Alpha-numeric
13	Alpha-numeric N2	10×16	0.0250 × 0.0400	2	10×14	0.0250 × 0.0350	Alpha-numeric
14	Alpha-numeric NO1	12×20	0.0300 × 0.0500	2	11×18	0.0275 × 0.0450	Alpha-numeric
15	Alpha-numeric NO2	12×20	0.0300 × 0.0500	—	12×20	0.0300 × 0.0500	Alpha-numeric

16	Alpha-numeric NO3	15×24	0.0375 × 0.0600	2	14×22	0.0350 × 0.0550	Alpha-numeric
17	Alpha-numeric NO4	20×32	0.0500 × 0.0800	2	19×30	0.0475 × 0.0750	Alpha-numeric
18	Alpha-numeric NO5	24×36	0.0600 × 0.0900	2	23×34	0.0575 × 0.0850	Alpha-numeric
19	Alpha-numeric NA1	16×20	0.0400 × 0.0500	2	15×18	0.0375 × 0.0450	Alpha-numeric
20	Alpha-numeric NA2	20×28	0.0500 × 0.0700	4	20×24	0.0500 × 0.0600	Alpha-numeric

21. Font Selection Tables

This chapter shows the font selection tables for each of the models.

21.1. DURA PRINTER R

Font selection	
Alphanumeric XS	1
Alphanumeric SS	5
Alphanumeric S	2
Alphanumeric M	3
Alphanumeric L	4
Pseudo OCR-B	6
Reserved	7
Reserved	8
Alphanumeric 2 -byte kana	9
Alphanumeric 1-byte kana	10

Country code	
USA	1
United Kingdom	3
France	4
Sweden/Finland	5
Denmark	6
Italy	7
Germany	8
Spain	9
Japan	10

21.2. DURA PRINTER LP5320

Font selection	
Alphanumeric XS	1
Alphanumeric SS	5
Alphanumeric S	2
Alphanumeric M	3
Alphanumeric L	4
Pseudo OCR-B	6
Kanji (16)	7
Kanji (24)	8
Pseudo OCR-A	9
Alphanumeric 1-byte kana	10

Country code	
USA	1
United Kingdom	3
France	4
Sweden/Finland	5
Denmark	6
Italy	7
Germany	8
Spain	9
Japan	10

21.3.DURA PRINTER SR/SRS and LSP5300

Font selection	
Alphanumeric XS	1
Alphanumeric SS	5
Alphanumeric S	2
Alphanumeric M	3
Alphanumeric L	4
Pseudo OCR-B	6
Kanji (16)	7
Kanji (24)	8
Alphanumeric 2-byte kana	9
Alphanumeric 1-byte kana	10
Alphanumeric B	11
Alphanumeric N1	12
Alphanumeric N2	13
Alphanumeric NO1	14
Alphanumeric NO2	15
Alphanumeric NO3	16
Alphanumeric NO4	17
Alphanumeric NO5	18
Alphanumeric NA1	19
Alphanumeric NA2	20

Country code	
USA	1
United Kingdom	3
France	4
Sweden/Finland	5
Denmark	6
Italy	7
Germany	8
Spain	9
Japan	10

With check digit	
Code39 MOD43	17
ITF MOD1OWait3	33
Codabar MOD10	49
Codabar MOD11	65
Codabar MOD16	81
Codabar MOD10&11	97
Case code128	113

22. Bar Code Selection Tables

The following tables show the BSYM and BDEF setting values supported by each model.

22.1. BSYM Settings

Bar code type	Setting	R	SR	SRS	LSP5300
Code39	1	Yes	Yes	Yes	Yes
CodaBar	3	Yes	Yes	Yes	Yes
Code39 (with modulo 43)	5	Yes	Yes	Yes	Yes
ITF	8	Yes	Yes	Yes	Yes
UPC-A	10	Yes	Yes	Yes	Yes
UPC-A 2 Char.	11	Yes	Yes	Yes	Yes
UPC-A 5 Char.	12	Yes	Yes	Yes	Yes
UPC-E	13	Yes	Yes	Yes	Yes
UPC-E 2 Char.	14	Yes	Yes	Yes	Yes
UPC-E 5 Char	15	Yes	Yes	Yes	Yes
EAN13	16	Yes	Yes	Yes	Yes
EAN8	17	Yes	Yes	Yes	Yes
EAN13 2 Char.	18	Yes	Yes	Yes	Yes
EAN13 5 Char.	19	Yes	Yes	Yes	Yes
EAN8 2 Char.	20	Yes	Yes	Yes	Yes
EAN8 5 Char.	21	Yes	Yes	Yes	Yes
Code128	25	Yes	Yes	Yes	Yes
Casecode Code128	27	Yes	Yes	Yes	Yes
Code93	30	Yes	Yes	Yes	Yes
ITF (with modulo 10 wait 3)	52	No	Yes	Yes	Yes
CodaBar (with modulo 16)	54	No	Yes	Yes	Yes
UPC-A (no HRI)	110	No	Yes	Yes	Yes
UPC-A 2 Char. (no HRI)	111	No	Yes	Yes	Yes
UPC-A 5 Char. (no HRI)	112	No	Yes	Yes	Yes
UPC-E (no HRI)	113	No	Yes	Yes	Yes
UPC-E 2 Char. (no HRI)	114	No	Yes	Yes	Yes
UPC-E 5 Char (no HRI)	115	No	Yes	Yes	Yes
EAN13 (no HRI)	116	No	Yes	Yes	Yes
EAN8 (no HRI)	117	No	Yes	Yes	Yes
EAN13 2 Char. (no HRI)	118	No	Yes	Yes	Yes

Bar code type	Setting	R	SR	SRS	LSP5300
EAN13 5 Char. (no HRI)	119	No	Yes	Yes	Yes
EAN8 2 Char. (no HRI)	120	No	Yes	Yes	Yes
EAN8 5 Char. (no HRI)	121	No	Yes	Yes	Yes
EAN13 (no check digit calculation)	216	No	Yes	Yes	Yes
EAN8 (no check digit calculation)	217	No	Yes	Yes	Yes

22.2. BDEF Settings

Bar code type	Setting	R	SR	SRS	LSP5300
Code39	1	Yes	Yes	Yes	Yes
Code39 ladder	2	Yes	Yes	Yes	Yes
CodaBar	3	Yes	Yes	Yes	Yes
CodaBar ladder	4	Yes	Yes	Yes	Yes
Code39 (with modulo 43)	5	Yes	Yes	Yes	Yes
Code39 (with modulo 43) ladder	6	Yes	Yes	Yes	Yes
ITF	8	Yes	Yes	Yes	Yes
ITF ladder	9	Yes	Yes	Yes	Yes
UPC-A	10	Yes	Yes	Yes	Yes
UPC-A 2 Char.	11	Yes	Yes	Yes	Yes
UPC-A 5 Char.	12	Yes	Yes	Yes	Yes
UPC-E	13	Yes	Yes	Yes	Yes
UPC-E 2 Char.	14	Yes	Yes	Yes	Yes
UPC-E 5 Char.	15	Yes	Yes	Yes	Yes
EAN-13	16	Yes	Yes	Yes	Yes
EAN-8	17	Yes	Yes	Yes	Yes
EAN 13 2 Char.	18	Yes	Yes	Yes	Yes
EAN 13 5 Char.	19	Yes	Yes	Yes	Yes
EAN 8 2 Char.	20	Yes	Yes	Yes	Yes
EAN 8 5 Char.	21	Yes	Yes	Yes	Yes
Code 128	25	Yes	Yes	Yes	Yes
Code 128 ladder	26	Yes	Yes	Yes	Yes
Casecode Code 128	27	Yes	Yes	Yes	Yes
Casecode Code 128 ladder	28	Yes	Yes	Yes	Yes
Code93	30	Yes	Yes	Yes	Yes
Code93 ladder	31	Yes	Yes	Yes	Yes

Bar code type	Setting	R	SR	SRS	LSP5300
UPC A ladder	40	Yes	Yes	Yes	Yes
UPC A 2 Char. ladder	41	Yes	Yes	Yes	Yes
UPC A 5 Char. ladder	42	Yes	Yes	Yes	Yes
UPC E ladder	43	Yes	Yes	Yes	Yes
UPC E 2 Char. ladder	44	Yes	Yes	Yes	Yes
UPC E 5 Char. ladder	45	Yes	Yes	Yes	Yes
EAN 13 ladder	46	Yes	Yes	Yes	Yes
EAN 8 ladder	47	Yes	Yes	Yes	Yes
EAN 13 2 Char. ladder	48	Yes	Yes	Yes	Yes
EAN 13 5 Char. ladder	49	Yes	Yes	Yes	Yes
EAN 8 2 Char. ladder	50	Yes	Yes	Yes	Yes
EAN 8 5 Char. ladder	51	Yes	Yes	Yes	Yes
ITF (with modulo 10 wait 3)	52	No	Yes	Yes	Yes
ITF (with modulo 10 wait 3) ladder	53	No	Yes	Yes	Yes
CodaBar (with modulo 16)	54	No	Yes	Yes	Yes
CodaBar (with modulo 16) ladder	55	No	Yes	Yes	Yes
UPC-A (no HRI)	110	No	Yes	Yes	Yes
UPC-A 2 Char. (no HRI)	111	No	Yes	Yes	Yes
UPC-A 5 Char. (no HRI)	112	No	Yes	Yes	Yes
UPC-E (no HRI)	113	No	Yes	Yes	Yes
UPC-E 2 Char. (no HRI)	114	No	Yes	Yes	Yes
UPC-E 5 Char. (no HRI)	115	No	Yes	Yes	Yes
EAN-13 (no HRI)	116	No	Yes	Yes	Yes
EAN-8 (no HRI)	117	No	Yes	Yes	Yes
EAN 13 2 Char. (no HRI)	118	No	Yes	Yes	Yes
EAN 13 5 Char. (no HRI)	119	No	Yes	Yes	Yes
EAN 8 2 Char. (no HRI)	120	No	Yes	Yes	Yes
EAN 8 5 Char. (no HRI)	121	No	Yes	Yes	Yes
UPC A ladder (no HRI)	140	No	Yes	Yes	Yes
UPC A 2 Char. ladder (no HRI)	141	No	Yes	Yes	Yes
UPC A 5 Char. ladder (no HRI)	142	No	Yes	Yes	Yes
UPC E ladder (no HRI)	143	No	Yes	Yes	Yes
UPC E 2 Char. ladder (no HRI)	144	No	Yes	Yes	Yes
UPC E 5 Char. ladder (no HRI)	145	No	Yes	Yes	Yes
EAN 13 ladder (no HRI)	146	No	Yes	Yes	Yes

EAN 8 ladder (no HRI)	147	No	Yes	Yes	Yes
Bar code type	Setting	R	SR	SRS	LSP5300
EAN 13 2 Char. ladder (no HRI)	148	No	Yes	Yes	Yes
EAN 13 5 Char. ladder (no HRI)	149	No	Yes	Yes	Yes
EAN 8 2 Char. ladder (no HRI)	150	No	Yes	Yes	Yes
EAN 8 5 Char. ladder (no HRI)	151	No	Yes	Yes	Yes
EAN13 (no check digit calculation)	216	No	Yes	Yes	Yes
EAN8 (no check digit calculation)	217	No	Yes	Yes	Yes
EAN13 ladder (no check digit calculation)	246	No	Yes	Yes	Yes
EAN8 ladder (no check digit calculation)	247	No	Yes	Yes	Yes

23. Two-Dimensional Symbol Selection Tables

Table showing support for DTDS settings

Symbol type	Setting	R	SR	SRS	LSP5300
QR code model 1	100	No	Yes	Yes	Yes
QR code model 2	101	No	Yes	Yes	Yes
Micro QR code	102	No	Yes	Yes	Yes
Code49	60	No	Yes	Yes	Yes

24.Status Response Formats

		LSP5300		R	
Status request	Bits	ENQ （05H）	STX（02H）	ENQ（05H）	STX（02H）
Header byte	7 • • • 0	Header always 02 hex	Header always 02 hex	Header always 02 hex	Header always 02 hex
Status byte 1	7	Always 0			
	6	Paused			
	5	Hardware error			
	4	Communications error			
	3	Cover open			
	2	Cutter error		Void error	
	1	Supply error			
	0	Command error			
Status byte 2	7	Footer always 03Hex	Undefined	Always 03 hex	Undefined Always 0
	6		Always 0		
	5		Peel-off label*		
	4		Label near empty		
	3		Ribbon End		
	2		Buffer >50% full		
	1		Printing		Printing
	0		Command data present		Command data present
Status byte 3	7 • • • 0		Footer always 03Hes		Footer Always 03 hex

* Label present in peel-off position

		SR		SRS	
Status request"	Bits	ENQ （05H）	STX（02H）	ENQ（05H）	STX（02H）
Header byte	7 • • • 0	Header always 02 hex	Header always 02 hex	Header always 02 hex	Header always 02 hex
Status byte 1	7	Always 0			
	6	Paused			
	5	Hardware error			
	4	Communications error			
	3	Cover open			
	2	Void error		Cutter error	
	1	Supply error			
	0	Command error			
Status byte 2	7		Undefined		Undefined
	6		Always 0		Always 0
	5		Peel-off label*		Peel-off label*
	4	Always	Undefined	Always	Undefined
	3	03 hex	Function setting	03 hex	Function setting
	2		Buffer >50% full		Buffer >50% full
	1		Printing		Printing
	0		Command data present		Command data present
Status byte 3	7 • • • 0		Footer Always03hex		Footer Always 03 hex

* Label present in peel-off position

25. Command Support Table

Categories in the table

Yes:	The Function is supported
No:	The Function is not suported, but the command is simply ignored.
Error:	The function is not supported and produces a command error.

cursor movement functions	Mnemonic	R	SR	SRS	LSP5300			
Vertical Position Relative	VPR	Yes	Yes	Yes	Yes			
Horizontal Position Relative	HPR	Yes	Yes	Yes	Yes			
Verticl Base Reference	VBR	Yes	Yes	Yes	Yes			
Horizontal Base Reference	HBR	Yes	Yes	Yes	Yes			
Home Position	HOME	Yes	Yes	Yes	Yes			
End Of Line	EOL	Yes	Yes	Yes	Yes			

BarCode setting functions		R	SR	SRS	LSP5300			
Barcode Stop	BSTP	Yes	Yes	Yes	Yes			
Barcode Start	BCST	Yes	Yes	Yes	Yes			
Barcode Nerrow Element Width	BNEW	Yes	Yes	Yes	Yes			
Bar wide Element width	BWEW	Yes	Yes	Yes	Yes			
Barcode Save Address Length	BSAL	Yes	Yes	Yes	Yes			
Barcode Definition	BDEF	Yes	Yes	Yes	Yes			
Barcode Symbol hight	BCSH	Yes	Yes	Yes	Yes			
Barcode Inter Charcter Gap	BICG	Yes	Yes	Yes	Yes			
Verifier(APPC) ON	APON	Yes	Yes	No	No			
Verifier(APPC) OFF	APOF	Yes	Yes	No	No			
Half Dot ON	HALF	Yes	Yes	Yes	Yes			
Half Dot ON Bar	HLFB	Error	Yes	Yes	Yes			
Half Dot Off	HOFF	Yes	Yes	Yes	Yes			
UPC Magnification	UMAG	Yes	Yes	Yes	Yes			
Barcode Charcter per Inch	BCPI	Yes	Yes	Yes	Yes			
Set Assured Quality Level	AQL	Yes	Yes	No	No			
Disable Verification	DVFY	Yes	Yes	No	No			
set Barcode Symbol	BSYM	Yes	Yes	Yes	Yes			
set # of Multi Void	NUMV	Yes	Yes	No	No			
Verification Mode	VFYM	Error	Yes	No	No			

Dot font setting functions	Mnemonic	R	SR	SRS	LSP5300			
Define Dotfont	DDF	Yes	Yes	Yes	Yes			
Dot font Magnification	DFM	Yes	Yes	Yes	Yes			
Dot font Space	DFS	Yes	Yes	Yes	Yes			
Dot font Orientation	DFO	Yes	Yes	Yes	Yes			
External font selection	FNTF	Error	Yes	Yes	Yes			

Vector font setteing functions(including outline font)	Mnemonic	R	SR	SRS	LSP5300			
Define Hurman readable	DHR	Yes	Yes	Yes	Yes			
Define Charcter Height	DCH	Yes	Yes	Yes	Yes			
Define Charcter Width	DCW	Yes	Yes	Yes	Yes			
Inter charcter Gap	ICS	Yes	Yes	Yes	Yes			
Define outline font Charcter Size	DCS	Yes	Yes	Yes	Yes			

Box drawing functions	Mnemonic	R	SR	SRS	LSP5300			
Draw outline Box	DBOX	Yes	Yes	Yes	Yes			
Draw Black Box	DBBX	Yes	Yes	Yes	Yes			
Draw White Box	DWBX	Yes	Yes	Yes	Yes			
Draw Complement Box	DCBX	Yes	Yes	Yes	Yes			

Line drawing functions	Mnemonic	R	SR	SRS	LSP5300			
Horizontal Line Thickness	HLT	Yes	Yes	Yes	Yes			
Vertical Line Thickness	VLT	Yes	Yes	Yes	Yes			
Draw Verticl Line	DVL	Yes	Yes	Yes	Yes			
Draw Horizontal Line	DHL	Yes	Yes	Yes	Yes			
Draw Diagonal Line	DDL	Yes	Yes	Yes	Yes			

Two Dimensional Symbol setting functions	Mnemonic	R	SR	SRS	LSP5300			
QR code Error Level	QREL	Error	Yes	Yes	Yes			
QR code Cell Size	QRCS	Error	Yes	Yes	Yes			
QR code Mask Pattern	QRMP	Error	Yes	Yes	Yes			
Define Two Dimentional symbol	DTDS	Error	Yes	Yes	Yes			
Two Dimensional Symbol setting functions	TDCW	Error	Yes	Yes	Yes			
Two Dimensional Symbol Height	TDCH	Error	Yes	Yes	Yes			
Two Dimensional Symbol Error Level	TDEL	Error	Yes	Yes	Yes			
Two Dimensional Symbol Matrix Size	TDRC	Error	Yes	Yes	Yes			
Two Dimensional Symbol Data mode	TDDM	Error	Yes	Yes	Yes			

serializing setting functions	Mnemonic	R	SR	SRS	LSP5300			
Barcode Inctement Decrement	BCID	Yes	Yes	Yes	Yes			
Barcode Label Count	BCLC	Yes	Yes	Yes	Yes			
Alphabetic serial number	ALPH	Yes	Yes	Yes	Yes			
Numetic serial number	NUM	Yes	Yes	Yes	Yes			
alphnumeric serial number	BOTH	Yes	Yes	Yes	Yes			
Text Increment Dectement	IDF	Yes	Yes	Yes	Yes			
Save Address Length	SAL	Yes	Yes	Yes	Yes			
Mark	MRK	Yes	Yes	Yes	Yes			
Return	RET	Yes	Yes	Yes	Yes			
Variable data Length/Position	VLP	Yes	Yes	Yes	Yes			
Bacode variable data Length/position	BVLP	Yes	Yes	Yes	Yes			
Except	EXCP	Yes	Yes	Yes	Yes			
Erase Mode ON	EMON	Error	Error	Yes	Yes			

Logo setting and drawing functions	Mnemonic	R	SR	SRS	LSP5300			
Draw Logo 1	LOGO	Yes	Yes	Yes	Yes			
Draw Logo 2	LOGD	Error	Error	Yes	Yes			
Logo File selection	LOGF	Error	Yes	Yes	Yes			
External Logo selection	XLOG	Error	Yes	Yes	Yes			

Operating setting functions	Mnemonic	R	SR	SRS	LSP5300			
Flip print	FLIP	Yes	Yes	Yes	Yes			
Break	BRK	Yes	Yes	Yes	Yes			
Start Processing Buffer	SPB	Yes	Yes	Yes	Yes			
Restart Processing Buffer	RSPB	Yes	Yes	Yes	Yes			
Terminater	TRM	Yes	Yes	Yes	Yes			
Field	FLD	Yes	Yes	Yes	Yes			
Cutter ON	CTON	Yes	No	Yes	Yes			
Cutter OFF	CTOF	Yes	No	Yes	Yes			
# of Bach Cut	NUBC	Yes	No	Yes	Yes			
No Reprint	NORP	Yes	Yes	Yes	Yes			
Use Top of Form	UTOF	Yes	Yes	Yes	Yes			

Printer control functions	Mnemonic	R	SR	SRS	LSP5300			
stetus request 1	<ENQ>	Yes	Yes	Yes	Yes			
status request 2	<STX>	Yes	Yes	Yes	Yes			
Printer Infomation request	INFO	Error	Error	Error	Yes			
Resett Trip metter	RSET	Error	Error	Error	Yes			
Pause ON	<DC2>	Yes	Yes	Yes	Yes			
Pa;use OFF	<DC4>	Yes	Yes	Yes	Yes			
Cancel	<CAN>	Yes	Yes	Yes	Yes			
Start Writting to Extended Memory	<FS>	Yes	Yes	Yes	Yes			
End Writting to Extended Memory	<EM>	Yes	Yes	Yes	Yes			