



Programmer's Reference Manual

Intermec
Fingerprint® v8.71

Intermec Technologies Corporation

Worldwide Headquarters
6001 36th Ave.W.
Everett, WA 98203
U.S.A.

www.intermec.com

The information contained herein is provided solely for the purpose of allowing customers to operate and service Intermec-manufactured equipment and is not to be released, reproduced, or used for any other purpose without written permission of Intermec Technologies Corporation.

Information and specifications contained in this document are subject to change without prior notice and do not represent a commitment on the part of Intermec Technologies Corporation.

© 2005-2006 by Intermec Technologies Corporation. All rights reserved.

The word Intermec, the Intermec logo, Norand, ArciTech, Beverage Routebook, CrossBar, dcBrowser, Duratherm, EasyADC, EasyCoder, EasySet, Fingerprint, INCA (under license), i-gistics, Intellitag, Intellitag Gen2, JANUS, LabelShop, MobileLAN, Picolink, Ready-to-Work, RoutePower, Sabre, ScanPlus, ShopScan, Smart Mobile Computing, SmartSystems, TE 2000, Trakker Antares, and Vista Powered are either trademarks or registered trademarks of Intermec Technologies Corporation.

There are U.S. and foreign patents as well as U.S. and foreign patents pending.

Microsoft, Windows, and the Windows logo are registered trademarks of Microsoft Corporation in the United States and/or other countries.

The name Centronics is wholly owned by GENICOM Corporation.

TrueDoc is a registered trademark of Bitstream, Inc.

TrueType is a trademark of Apple Computer Inc.

Unicode is a trademark of Unicode Inc.

Document Change Record

This page records changes to this document.

Version	Date	Description of Change
001	9/2006	<p>This document has been assigned a new part number. The previous version was 1-960582-07.</p> <p>Added Aztec symbology information and ECI support for Datamatrix and Aztec symbologies.</p> <p>Added SYSVAR (53) value.</p>

Contents

Before You Begin	xix
Safety Information	xix
Global Services and Support	xix
Warranty Information	xix
Web Support	xix
Telephone Support	xx
Who Should Read This Manual	xx
Related Documents	xx
1 Introduction	1
About Intermec Fingerprint	2
Fingerprint Documentation	2
Differences Between Fingerprint 8.71.0 and 10.0.0	3
What's New in Intermec Fingerprint v8.71	3
About the Printer File System and Directories	4
Files in Intermec Fingerprint v8.71	5
2 Fingerprint Commands	7
ABS	8
ACTLEN	8
ALIGN (AN)	9
ASC	11
BARADJUST	12
BARCODENAME\$	13
BARFONT (BF)	14
BARFONT ON/OFF (BF ON/OFF)	15
BARHEIGHT (BH)	16
BARMAG (BM)	17
BARRATIO (BR)	17
BARSET	18

Contents

BARTYPE (BT).....	19
BEEP.....	20
BREAK.....	20
BREAK ON/OFF.....	22
BUSY	23
CHDIR.....	23
CHECKSUM	24
CHR\$.....	24
CLEANFEED.....	25
CLEAR	26
CLIP	26
CLL.....	27
CLOSE	28
COM ERROR ON/OFF	29
COMBUF\$.....	30
COMSET	31
COMSET OFF.....	33
COMSET ON	34
COMSTAT.....	35
CONT	36
COPY.....	36
COUNT&	37
CURDIR\$.....	39
CSUM.....	39
CUT	40
CUT ON/OFF.....	41
DATE\$	41

DATEADD\$	42
DATEDIFF	44
DBBREAK	45
DBBREAK OFF	45
DBEND	46
DBSTDIO	46
DBSTEP	47
DELETE	48
DELETEPFSVAR	48
DEVICES	48
DIM	51
DIR	52
DIRNAME\$	55
END	56
EOF	56
ERL	57
ERR	58
ERR\$	59
ERROR	59
EXECUTE	61
FIELD	62
FIELDNO	62
FILE& LOAD	63
FILENAME\$	64
FILES	65
FLOATCALC\$	66

Contents

FONT (FT).....	67
Adding More Fonts.....	68
FONTD	69
FONTNAME\$.....	70
FONTS.....	71
FOR...TO...NEXT	71
FORMAT	73
FORMAT DATE\$	74
FORMAT INPUT	75
FORMAT TIME\$.....	76
FORMAT\$	77
FORMFEED (FF).....	80
FRE.....	81
FUNCTEST	82
FUNCTEST\$	83
GET	84
GETASSOC\$	85
GETASSOCNAME\$	86
GETPFSVAR	86
GOSUB.....	87
GOTO	88
HEAD.....	89
IF...THEN...(ELSE)	91
IMAGE BUFFER MIRROR.....	92
IMAGE BUFFER SAVE.....	93
IMAGE LOAD	94
IMAGENAME\$	95

IMAGES	95
IMMEDIATE	96
INKEY\$	98
INPUT (IP).....	99
INPUT ON/OFF	100
INPUT#	101
INPUT\$	102
INSTR	103
INVIMAGE (II)	104
KEY BEEP	104
KEY ON/OFF.....	105
KEYBMAP\$	106
KILL	108
LAYOUT.....	109
LAYOUT END.....	112
LAYOUT INPUT.....	113
LAYOUT RUN.....	114
LBLCOND.....	115
LED ON/OFF	117
LEFT\$.....	118
LEN	118
LET.....	119
LINE INPUT	120
LINE INPUT#.....	120
LIST	121
LISTPFSVAR	122
LOAD	122

Contents

LOC	123
LOF.....	124
LSET.....	125
LTS& ON/OFF	126
MAG.....	126
MAKEASSOC.....	127
MAP	128
MERGE	129
MIBVAR&	130
MID\$	131
MKDIR.....	132
NAME DATE\$.....	132
NAME WEEKDAY\$.....	133
NASC	134
NASCD.....	136
NEW.....	136
NORIMAGE (NI)	137
ON BREAK GOSUB	137
ON COMSET GOSUB	138
ON ERROR GOTO.....	139
ON GOSUB.....	140
ON GOTO.....	141
ON HTTP GOTO	142
ON KEY GOSUB.....	142
ON/OFF LINE.....	144
ON MIBVAR& GOSUB	144
OPEN	145

OPTIMIZE BATCH ON/OFF.....	147
PORIGIN	148
PORTOUT ON/OFF.....	149
PRBAR (PB)	149
PRBOX (PX).....	150
PRBUF	155
PRIMAGE (PM).....	156
PRINT (?)	157
PRINT KEY ON/OFF	158
PRINT#	158
PRINTFEED (PF)	159
PRINTONE	161
PRINTONE#	162
PRLINE (PL)	162
PRPOS (PP)	163
PRSTAT	165
PRTXT (PT).....	167
PUT	169
RANDOM	170
RANDOMIZE	170
READY.....	171
REBOOT	172
REDIRECT OUT.....	172
REM (').	173
REMOVE IMAGE	173
RENDER ON/OFF	174
RENUM	175

Contents

REPRINT ON/OFF	175
RESUME	176
RETURN	177
REWINDCONTROL	178
REWINDVOID	179
RIGHT\$	179
RSET	180
RUN	181
SAVE	182
SET FAULTY DOT	183
SETASSOC	183
SETPFSVAR	184
SETSTDIO	185
SETUP	186
SETUP GET	189
SETUP KEY	189
SETUP WRITE	190
SGN	191
SORT	192
SOUND	193
SPACE\$	194
SPLIT	195
STOP	196
STORE IMAGE	196
STORE INPUT	198
STORE OFF	199
STR\$	199

STRING\$	200
SYSHEALTH	200
SYSHEALTH\$	201
SYSVAR	202
TAGFIELD	209
TAGFORMAT	210
TAGPROTECT	212
TAGREAD	212
TAGWRITE	213
TESTFEED	214
TICKS	215
TIME\$	215
TIMEADD\$	216
TIMEDIFF	217
TRANSFER KERMIT	218
TRANSFER NET	219
TRANSFER STATUS	220
TRANSFER ZMODEM	221
TRANSFER\$	222
TRANSFERSET	222
TRON/TROFF	223
VAL	224
VERBON/VERBOFF	224
VERSION\$	225
WEEKDAY	226
WEEKDAY\$	227
WEEKNUMBER	228

Contents

WHILE...WEND	229
XORMODE ON/OFF	230
External Command; Account Secret	230
External Command; ZMODEM	232
3 About Image Transfer Protocols	233
About Image Protocols	234
Intelhex	234
UBI00	234
UBI01	234
UBI02	234
UBI03	235
Image Format	235
Encoding a Bitmap As a Bit Representation	236
Encoding a Bitmap With an RLL Algorithm	237
About the UBI10 Protocol.....	239
Protocol Description.....	239
Frame Definitions.....	239
PRBUF Protocol	241
2-Byte Header	241
RLL Buffer Format.....	241
RLL Format	241
4 Character Sets and Fonts	243
Introduction	244
Roman 8 Character Set NASC 1	245
French Character Set NASC 33	246
Spanish Character Set NASC 34.....	247
Italian Character Set NASC 39.....	248
English (UK) Character Set NASC 44	249
Swedish Character Set NASC 46	250
Norwegian Character Set NASC 47	251
German Character Set NASC 49	252
Japanese Latin Character Set NASC 81	253
Portuguese Character Set NASC 351	254
PCMAP Character Set NASC -1	255
ANSI Character Set NASC -2.....	256
MS-DOS Latin 1 Character Set NASC 850	257
MS-DOS Greek 1 Character Set NASC 851	258
MS-DOS Latin 2 Character Set NASC 852	259
MS-DOS Cyrillic Character Set NASC 855	260
MS-DOS Turkish Character Set NASC 857.....	261
Windows Latin 2 Character Set NASC 1250	262
Windows Cyrillic Character Set NASC 1251	263

Windows Latin 1 Character Set NASC 1252	264
Windows Greek Character Set NASC 1253	265
Windows Latin 5 Character Set NASC 1254	266
Windows Baltic Rim Character Set NASC 1257	267
OCR-A BT Character Set.....	268
OCR-B 10 Pitch BT Character Set.....	269
DingDings SWA Character Set.....	270
 About the UTF-8 Character Set.....	271
Example	273
 Scalable Fonts	273
Bitmap Fonts.....	275
Font Aliases	275
 5 Bar Codes	277
Supported Symbologies	278
General Rules for Bar Code Printing	280
One- and Two-Dimensional Bar Codes	281
Aztec	282
Code 39	283
Code 128	284
Data Matrix	286
EAN-8	288
EAN-13	288
EAN-128	289
Interleaved 2 of 5	290
MaxiCode.....	291
About Structured Append Mode	292
Primary and Secondary Message Data Elements	292
MicroPDF417	293
Setting the Number of Rows and Columns	294
PDF417	295
QR Code	298
RSS-14	299
RSS-14 Truncated.....	300
RSS-14 Stacked	301
RSS-14 Stacked Omnidirectional	302
RSS-14 Limited	303
RSS-14 Expanded.....	304
RSS-14 Expanded Stacked	305
UPC-A.....	307
UPC-E	307
 About AddOn Codes.....	308

About Composite Bar Codes	309
About the Linear Components	309
About the Composite Components	309
About the Human Readable Section	310
Supported Component Combinations	310
UPC-A Composite Symbol	310
EAN-13 Composite Symbol	310
EAN-8 Composite Symbol	311
UPC-E Composite Symbol	311
UCC/EAN-128 Composite Symbol	311
RSS Composite Symbol	312
About CC-ABC Composite Symbols	312
CC-A	313
CC-B	313
CC-C	313
EAN8 Composite With CC-A or CC-B	314
EAN13 Composite With CC-A or CC-B	315
UPC-E Composite With CC-A or CC-B	316
UPC-A Composite With CC-A or CC-B	317
EAN.UCC 128 Composite With CC-C	318
EAN.UCC 128 Composite With CC-A or CC-B	320
RSS-14 (Composite)	322
RSS-14 Truncated (Composite)	323
RSS-14 Stacked (Composite)	324
RSS-14 Stacked Omnidirectional (Composite)	326
RSS-14 Limited (Composite)	327
RSS-14 Expanded (Composite)	329
RSS-14 Expanded Stacked (Composite)	330
Setup Bar Codes	332
6 RFID Tag Formats	333
About EPC Tag Formats	334
About the Uniform Resource Identifier (URI)	334
Tag Formats	335
SGTIN-64	335
SGTIN-96	335
SSCC-64	336
SSCC-96	336
SGLN-64	336
SGLN-96	337
GRAI-64	337
GRAI-96	337
GIAI-64	338
GIAI-96	338
GID-96	338
USDOD-64	338
USDOD-96	339
Other EPC Tag Input Methods	339

EPC Tag Writing Example.....	340
About Tag Memory Allocation Standards	341
UCode EPC 1.19	342
ISO 18000-6B (UCode HSL) Tag Memory Allocation.....	343
Class 1 (EPC Class 1 Version 1) Tag Memory Allocation	344
EPCglobal Class 1 Gen 2 Tag Memory Allocation	345
About Non-Standard Tag Formats	346
7 Error Codes	347
Error Code Messages	348
A Printer Keyboard Layouts	355
PF2i, PF4i, PF4ci	356
PM4i.....	358
PX4i, PX6i Standard Keyboard	359
PX4i, PX6i Alphanumeric Keyboard	361
About ID Numbers and ASCII Values	361
About the PD41	363

Contents

Before You Begin

This section provides you with safety information, technical support information, and sources for additional product information.

Safety Information

Your safety is extremely important. Read and follow all warnings and cautions in this document before handling and operating Intermec equipment. You can be seriously injured, and equipment and data can be damaged if you do not follow the safety warnings and cautions.

This section explains how to identify and understand dangers, warnings, cautions, and notes that are in this document. You may also see icons that tell you when to follow ESD procedures and when to take special precautions for handling optical parts.



A caution alerts you to an operating procedure, practice, condition, or statement that must be strictly observed to prevent equipment damage or destruction, or corruption or loss of data.



Note: Notes either provide extra information about a topic or contain special instructions for handling a particular condition or set of circumstances.

Global Services and Support

Warranty Information

To understand the warranty for your Intermec product, visit the Intermec web site at www.intermec.com and click **Service & Support > Warranty**.

Disclaimer of warranties: The sample code included in this document is presented for reference only. The code does not necessarily represent complete, tested programs. The code is provided “as is with all faults.” All warranties are expressly disclaimed, including the implied warranties of merchantability and fitness for a particular purpose.

Web Support

Visit the Intermec web site at www.intermec.com to download our current manuals (in PDF). To order printed versions of the Intermec manuals, contact your local Intermec representative or distributor.

Visit the Intermec technical knowledge base (Knowledge Central) at intermec.custhelp.com to review technical information or to request technical support for your Intermec product.

Telephone Support

These services are available from Intermec.

Services	Description	In the USA and Canada call 1-800-755-5505 and choose this option
Order Intermec products	<ul style="list-style-type: none">• Place an order.• Ask about an existing order.	1 and then choose 2
Order Intermec media	Order printer labels and ribbons.	1 and then choose 1
Order spare parts	Order spare parts.	1 or 2 and then choose 4
Technical Support	Talk to technical support about your Intermec product.	2 and then choose 2
Service	<ul style="list-style-type: none">• Get a return authorization number for authorized service center repair.• Request an on-site repair technician.	2 and then choose 1
Service contracts	<ul style="list-style-type: none">• Ask about an existing contract.• Renew a contract.• Inquire about repair billing or other service invoicing questions.	1 or 2 and then choose 3

Outside the U.S.A. and Canada, contact your local Intermec representative. To search for your local representative, from the Intermec web site, click **Contact**.

Who Should Read This Manual

This document explains how to use Intermec Fingerprint to program Intermec printers, and includes a command reference chapter with descriptions, syntax, and related information for each command.

Before you work with Fingerprint, you should be familiar with your network, general networking terms, such as IP address, and your bar code label printers.

Related Documents

The Intermec web site at www.intermec.com contains our documents (as PDF files) that you can download for free.

To download documents

- 1 Visit the Intermec web site at www.intermec.com.
- 2 Click **Service & Support > Manuals**.
- 3 In the **Select a Product** field, choose the product whose documentation you want to download.

To order printed versions of the Intermec manuals, contact your local Intermec representative or distributor.



1 Introduction

About Intermec Fingerprint

Intermec Fingerprint is a BASIC-inspired, printer-resident programming language for use with the Intermec EasyCoder bar code label printers. Fingerprint is an easy-to-use intelligent programming tool for label formatting and printer customization, allowing you to design your own label formats and write your own printer application software. You may easily create a printer program that exactly fulfills your own unique requirements. Improvements or changes due to new demands can be implemented quickly and at minimal cost.

Fingerprint also contains an easy-to-use slave protocol, Intermec Direct Protocol, which allows layouts and variable data to be downloaded from a host and combined into labels, tickets and tags with a minimum of programming. Intermec Direct Protocol also includes a versatile error handler and a flexible counter function.

Fingerprint version 8.71.0 is supported by these EasyCoder printers:

- PF2 and PF4i
- PM4i
- PX4i and PX6i

Fingerprint version 10.0.0 is supported only by the EasyCoder PD41 and includes the same functionality as v8.70.0. Other Intermec printers may only support earlier versions of Fingerprint, or may require different firmware. For more information, see your printer documentation.

Fingerprint Documentation

This Programmer's Reference Manual contains detailed information on all programming instructions in the Intermec Fingerprint programming language. It also contains other program-related information common to all Fingerprint-compatible Intermec printers.

If you are new to Fingerprint, you can get started with the *Fingerprint v8.XX Tutorial*.

Although the Intermec Direct Protocol is a subset of Intermec Fingerprint, Direct Protocol is described in more detail in the *Intermec Direct Protocol v8.XX Programmer's Reference Manual*.

All information needed by the operator, like how to set up and run the printer, how to load the media or ribbon supply and how to maintain the printer, can be found in the User's Guide for that printer.

PDF versions of other Fingerprint manuals can be found at www.intermec.com. To learn how to download other Intermec manuals, see “[Related Documents](#)” on page [xx](#).

Differences Between Fingerprint 8.71.0 and 10.0.0

Fingerprint 10.0.0 has been designed for use with the EasyCoder PD41. Although most commands work exactly the same as for Fingerprint 8.XX series, the differences in the printer platforms cause the firmware to have some important differences. Some general differences are listed below:

- Because the EasyCoder PD41 has no display, commands affecting the display have no effect.
- The EasyCoder PD41 has only a Print button. Applications requiring user input on the printer keyboard will therefore not work. Changing SETUP parameters through the console is not possible.
- The EasyCoder PD41 does not support as many optional interface cards. This affects the devices available.
- The EasyCoder PD41 does not send a power fail signal. Thus variables cannot be saved with the SETPFSVAR command unless the printer is manually turned off. This naturally affects the functionality of GETPFSVAR, DELETEPFSVAR and LISTPFSVAR.

What's New in Intermec Fingerprint v8.71

These features have been added for Fingerprint 8.71:

- New SYSVAR value (53): Sets or reads the highest allowed outer diameter (in mm) of the ribbon supply. For more information, see “[SYSVAR](#)” on page 202.
- Support for Aztec bar code symbology.
- Extended Channel Interpretation (ECI) support for Data Matrix and Aztec bar codes.
- The SETUP KEY command enables or disables setting up the printer via its built-in keypad. For more information, see “[SETUP KEY](#)” on page 189.
- The TAGFIELD command includes EPCglobal Class 1 Gen 2 tag segment names. For more information, see “[TAGFIELD](#)” on page 209.
- The TESTFEED command can identify RFID media. For more information, see “[TESTFEED](#)” on page 214.

About the Printer File System and Directories

Two parts of the printer memory support the use of directories:

- the read-only memory (rom).
- the read/write permanent storage memory (c).

Directories cannot be used in any other parts of the memory or in CompactFlash memory cards (card1).

The slash letter (/) is used as a divisor between directories and files. For example, the path “/c/DIR1/DIR2/FILE” refers to a file or directory named FILE in the directory DIR2, which in its turn is located in the directory DIR1 in the root of the device /c (the printer’s permanent memory). The maximum length of a path is 255 characters.

The “old” device names (c:, rom:, tmp:, and so on) are now aliases (“shortcuts”) to the new directories (/c/, /rom/, /tmp/, and so on). The file STDIO on c: (/c) can thus be accessed using either c:STDIO or /c/STDIO. Writing c: is equivalent to writing /c/.

The commands and output formats are designed to be as backwards-compatible as possible while giving the user access to the new features—directories. Examples of this are:

- FILES give a size of 0 for directories to minimize impact on applications that parse the output.
- FILENAME\$ only report files to minimize impact on applications that use FILENAME\$ to get file listings.

To relieve the user from always having to use the entire path when referring to a directory above the current one, each directory (including the root directories) contains a “parent directory”. This parent directory is called “..”. It refers to the directory’s parent directory. It is listed by FILES,A.

Each directory also has a reference to itself (“.”), that is, “/c./DIR1/..FILE” refers to “/c/FILE” (or, using the legacy format, to “c:FILE”).

Example:

CHDIR “/c/DIR1/DIR2”	Changes the directory
COPY “..//DIR3/FILE”, “FILE”	Copies /c/DIR1/DIR3/FILE to /c/DIR1/DIR2/FILE
CHDIR “..”	Go up to “/c/DIR1”
CHDIR “..//”	Go up to /c. Note that a trailing slash (/) may be used.



Note: A file or directory name may contain all printable characters except “:” (colon) and “/” (slash). Only /c (c:) supports creating and removing directories.

Files in Intermec Fingerprint v8.71

The Intermec Fingerprint v8.71 firmware contains the following files. Files in Fingerprint 10.0.0 (for EasyCoder PD41 only) may differ slightly. Some files in “/c/” only appear after default settings have been changed.

In device "/rom/"

.FONTALIAS	Creates reference fonts
.coms	System file
.profile	System file
.setup.saved	Default setup values
.uartx	System file
.ubipfr1.bin	Standard fonts
.wi_firm	Firmware for EasyLAN
CHESS2X2.1	Standard image for test labels
CHESS4X4.1	Standard image for test labels
DIAMONDS.1	Standard image for test labels
ERRHAND.PRG	Error Handler
FILELIST.PRG	List the lines of a file
GLOBE.1	Standard image for test labels
LBSHTXT.PRG	Intermec Shell auxiliary file
LINE_AXP.PRG	Intermec Shell Line Analyzer
LSHOPXP1.SUB	Intermec Shell auxiliary file
MKAUTO.PRG	Create a startup (autoexec) file
PUP.BAT	Intermec Shell Startup file
SHELLXP.PRG	Intermec Shell startup program
WINXP.PRG	Intermec Shell auxiliary file
default.html	EasyLAN home page
home.htmf	EasyLAN home page
htmlhead.htmf	EasyLAN home page
images/	EasyLAN home page
itclogo1.gif	EasyLAN home page
monitor	System file
nav.html	EasyLAN home page
passwd	Default password file
restrictions	Default restriction file
secure/	EasyLAN home page
configj.js	EasyLAN home page
configtree.html	EasyLAN home page
ddns.html	EasyLAN home page

empty.htm	EasyLAN home page
ftie4style.css	EasyLAN home page
ftiens4.js	EasyLAN home page
ftv2blank.gif	EasyLAN home page
ftv2doc.gif	EasyLAN home page
ftv2folderclosed.gif	EasyLAN home page
ftv2folderopen.gif	EasyLAN home page
ftv2lastnode.gif	EasyLAN home page
ftv2link.gif	EasyLAN home page
ftv2mlastnode.gif	EasyLAN home page
ftv2mnode.gif	EasyLAN home page
ftv2node.gif	EasyLAN home page
tv2plastnode.gif	EasyLAN home page
ftv2root.gif	EasyLAN home page
ftv2vertline.gif	EasyLAN home page
general.html	EasyLAN home page
mail.html	EasyLAN home page
main.html	EasyLAN home page
snmp.html	EasyLAN home page
tcpip.html	EasyLAN home page
view.html	EasyLAN home page
wlan.html	EasyLAN home page
support.htmf	EasyLAN home page

In device “/c”

.setup.saved	Current setup values (prt section)
boot/	
passwd	Password storage
rmmcz	Kernel file
psa	Kernel file
STDIO	Intermec Shell auxiliary file
APPLICATION	Intermec Shell auxiliary file
ADMIN/	
restrictions	Restrictions storage

To read the contents of these files, run the FILELIST.PRG program or COPY the file in question to the serial port “uart1:”.



2 Fingerprint Commands

This chapter lists all Fingerprint commands alphabetically, including each command's purpose, syntax, ranges and settings, and related information.

In the command syntax, certain punctuation marks are used to indicate various types of data. Do not include punctuation marks unless noted for that command.

Convention	Description
[]	Entries within brackets are optional.
<i>keyword1</i> <i>keyword2</i>	Indicates "either-or." Choose only one of the keywords.
< >	Grouping of data.
.....	Repetition of the preceding data type.
" "	Quotation mark (ASCII 34 dec.).
ø	Carriage return or linefeed.
<scon>	String constant.
<ncon>	Numeric constant.
<sexp>	String expression.
<nexp>	Numeric expression.
<svar>	String variable.
<nvar>	Numeric variable.
<stmt>	Statement.
<line label>	Line label.

Uppercase letters indicate keywords and must be entered exactly as described. Lowercase letters are also allowed unless stated otherwise.

ABS

- Purpose** Returns the absolute value of a numeric expression.
- Syntax** ABS (<nexp>)
where <nexp> is a numeric expression from which the absolute value will be returned.
- Remarks** The absolute value of a number is always positive or zero. Note that the expression must be enclosed within parentheses.
- Example 1** PRINT ABS (20-25)
results in:
5
- Example 2** PRINT ABS (25-20)
results in:
5
- Example 3** PRINT ABS (5-5)
results in:
0
- Example 4** PRINT ABS (20*-5)
results in:
100

ACTLEN

- Purpose** Returns the length of the most recently executed PRINTFEED, FORMFEED, or TESTFEED statement.
- Syntax** ACTLEN
- Remarks** The length of the most recently executed paper feed operation, resulting from a PRINTFEED, FORMFEED, or TESTFEED statement, is returned as a number of dots. Due to technical reasons concerning the stepper motor control and label gap detection, a small deviation from the expected result may occur.
- Example** In this example, a 12 dots/mm printer is loaded with 90-mm (1080 dots) labels separated by a 3-mm (36 dots) gap. Start- and stopadjust setup values are both set to 0:
- ```
10 FORMFEED
20 PRINT ACTLEN
RUN
```
- results in:  
1121

The deviation from the expected result (1116) is normal and should have no practical consequences (less than 1 mm).

## ALIGN (AN)

**Purpose** Statement specifying which part (anchor point) of a text field, bar code field, image field, line, or box will be positioned at the insertion point.

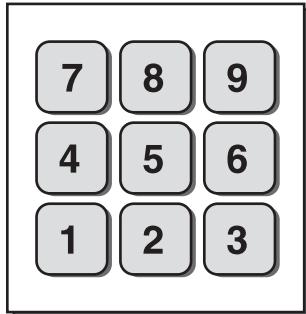
**Syntax** ALIGN | AN<nexp>

where <nexp> is the anchor point of the object. Range is 1 to 9. Default value is 1.

Reset to default by PRINTFEED execution.

**Remarks** Each text, bar code, or image field has nine possible anchor points, whereas lines and boxes have three. One of these points must be selected, or the default value (1) is used. The selected anchor point determines the position of the object relative to the insertion point, which is in turn determined by the nearest preceding PRPOS statement. Furthermore, the field will be rotated around the anchor point according to the nearest preceding DIR statement.

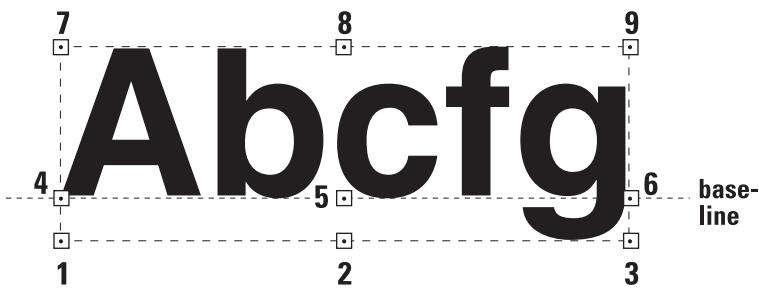
The nine anchor points of a text, bar code, or image field are oriented like the numeric keys on a computer keyboard:



Lines and boxes have three anchor points only: left, center, and right.

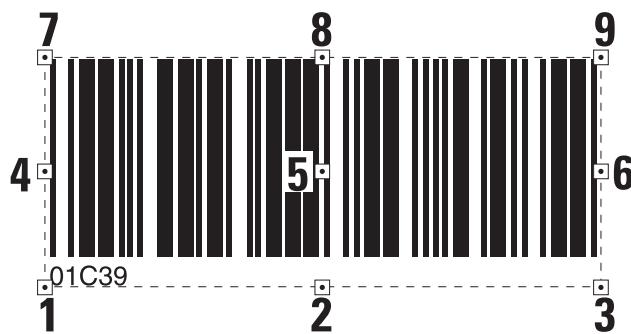
The anchor points for field types are illustrated next.

### Text Field Anchor Points



A text field makes up an imaginary box limited in width by the length of the text, and in height by the matrix size of the selected font. In text fields, the anchor points 4, 5, and 6 are situated on the baseline.

### **Bar Code Field Anchor Points**



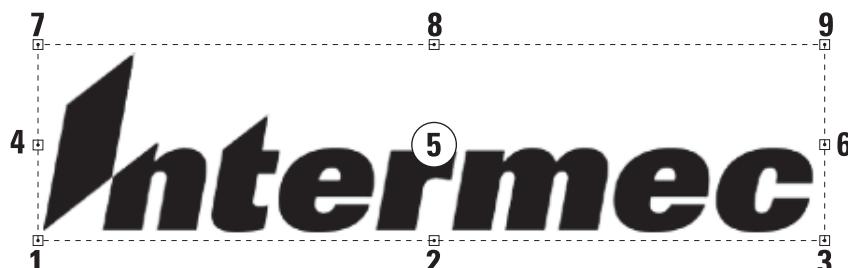
A bar code field makes up an imaginary box sufficiently large to accommodate the bar code interpretation, regardless if it will be printed or not (provided that the selected type of bar code may include an interpretation at all).

However, for EAN and UPC codes, the box is restricted in width by the size of the bar pattern, not by the interpretation. This implies that the first digit of the bar code interpretation will be outside the imaginary box:



For composite bar codes, the human readable bar code interpretation for the 2-dimensional element is outside the imaginary box.

### **Image Field Anchor Points**

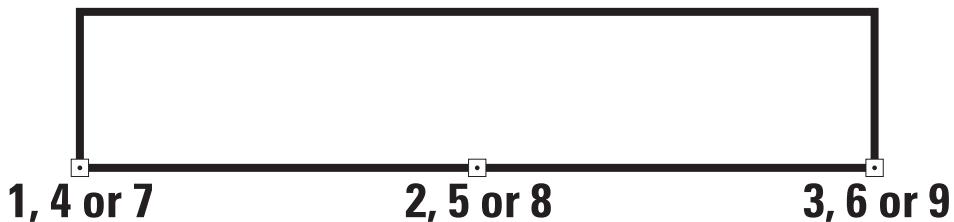


The size of an image field is decided when the field is created. Note that an image field consists of the entire area of the original image, including a white or transparent background.

### **Line Anchor Points**



### Box Anchor Points



The anchor points are situated at the lower side of the line or box relative to the selected print direction. Lines and boxes have only three anchor points, each of which can be specified by means of three different numbers.

A special case is multi-line text fields in a box. The fields can be aligned in nine positions in relation to the box, while the box itself only has three anchor points, as described above. For more information on alignment of multi-line text fields, see “[PRBOX \(PX\)](#)” on page 150.

**Example** This example prints one line of text aligned left on the baseline:

```
10 PRPOS 30,250
20 DIR 1
30 ALIGN 4
40 FONT "Swiss 721 BT"
50 PRTXT "Hello!"
60 PRINTFEED
RUN
```

The text “Hello!” is positioned with the baseline aligned left to the insertion point specified by the coordinates X=30; Y=250 in line 10.

## ASC

**Purpose** Returns the decimal ASCII value of the first character in a string expression.

**Syntax** ASC(<sexp>)

where <sexp> is a string expression from which the ASCII decimal value of the first character is returned.

**Remarks** ASC is the inverse function of CHR\$. The decimal ASCII value is given according to the selected character set (see NASC statement).

**Example 1**

```
10 A$="GOOD MORNING"
20 PRINT ASC(A$)
RUN
```

results in:

71

**Example 2**

```
10 B$="123456"
20 C% = ASC(B$)
30 PRINT C%
RUN
```

results in:

49

## BARADJUST

**Purpose** Statement for enabling/disabling automatic adjustment of bar code position in order to avoid faulty printhead dots.

**Syntax** BARADJUST<exp1>, <exp2>

where:

<exp1> is the maximum left offset in dots.

<exp2> is the maximum right offset in dots.

Default is 0,0 (BARADJUST disabled).

**Remarks** Occasionally a printer may have to run for some time with a faulty printhead before a replacement printhead can be installed.

Single faulty dots will produce very thin “white” lines along the media. This may be tolerable for text, graphics, and vertical (ladder) bar codes, but for horizontal bar codes (picket fence), this condition may render the bar code unreadable.

However, if the bar code is moved slightly to the left or right, the trace of a faulty dot may come between the bars of the bar code and the symptom is remedied temporarily. The statement attempts to adjust the bar code even if several faulty dots have been specified, but returns error 1052 if it was unable to adjust the bar code successfully.

The BARADJUST statement allows the Fingerprint firmware to automatically readjust the bar code position within certain limits when a faulty dot is detected (see “[HEAD](#) on page 89) and marked as faulty (see “[SET FAULTY DOT](#) on page 183). The maximum deviation from the original position, as specified by the PRPOS statement, can be set up separately for the directions left and right. Setting both parameters to 0 (zero) disables BARADJUST.

The BARADJUST statement does not work with:

- Vertically printed bar codes (ladder style)
- Stacked bar codes (such as Code 16K)
- Bar codes with horizontal lines (such as DUN-14/16)
- EAN/UPC-codes (interpretation not repositioned)

**Examples** This example enables BARADJUST within 10 dots to the left and 5 dots to the right of the original position for a specific bar code, then disables BARADJUST:

```
10 BARADJUST 10,5
20 PRPOS 30,100
30 BARSET "CODE39",2,1,3,120
40 BARFONT ON
50 PRBAR "ABC"
60 BARADJUST 0,0
70 PRINTFEED
```

## BARCODENAME\$

- Purpose** Returns the names of the bar code generators stored in the printer's temporary memory ("tmp:").
- Syntax** BARCODENAME\$(<nexp>)
- where <nexp> returns either false or true. False (0) indicates first bar code. True ( $\neq 0$ ) indicates next bar code.
- Remarks** BARCODENAME\$(0) produces the first bar code name in alphabetical order. BARCODENAME\$( $\neq 0$ ) produces the next name. This statement can be repeated as long as there are bar code names remaining.
- Example** Use a program like this to list the names of all bar codes in "tmp:". Note that bar codes with dynamic downloading will not appear before they have been called by a BARSET or BARTYPE statement.

```

10 A$ = BARCODENAME$ (0)
20 IF A$ = "" THEN END
30 PRINT A$
40 A$ = BARCODENAME$ (-1)
50 GOTO 20
RUN

```

results in:

```

ADDON2
ADDON5
C2OF5
C2OF5IND
C2OF5INDC
C2OF5MAT
CODABAR
CODE11
CODE128
CODE128A
CODE128B
CODE128C
CODE16K
CODE39
CODE39A
CODE39C
CODE49
CODE93
DATAMATRIX
DUN
EAN128
EAN128A
EAN128B
...

```

## BARFONT (BF)

|                |                                                                                                                                                                                                                                                                                              |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b> | Statement specifying fonts for the printing of bar code interpretation.                                                                                                                                                                                                                      |
| <b>Syntax</b>  | BARFONT   BF [#<ncon>,] <sexp1> [, <nexp1> [, <nexp2> [, <nexp3> [, <nexp4> [, <nexp5> [, <nexp6> [, <nexp7> [, <nexp8> [, <nexp9>]]]]]]] [ON]                                                                                                                                               |
|                | where:                                                                                                                                                                                                                                                                                       |
| #<ncon>        | (optional) is the start parameter in the syntax and specifies which parameter in the syntax is the first parameter in the statement (possibly bypassing some of the initial parameters). Default is #1.                                                                                      |
| <sexp1>        | is the name of the font selected for bar code interpretations. Corresponds to the FONT statement, but only affects bar code interpretation. Double-byte fonts cannot be used. Default is Swiss 721 BT.                                                                                       |
| <nexp1>        | is the height in points of the font. Default is 12. Corresponds to the FONT statement, but only affects bar code interpretation.                                                                                                                                                             |
| <nexp2>        | is the clockwise slant in degrees. Range is 0° to 90°. Values greater than 65° to 70° will be unreadable. Slanting increases clockwise. Default is 0. Corresponds to the FONT statement, but only affects bar code interpretation.                                                           |
| <nexp3>        | is the distance in dots between the bottom of the bar code pattern and the top of the character cell. Default is 6.                                                                                                                                                                          |
| <nexp4>        | is the magnification in regard to height. Range is 1 to 4. Default is 1. Corresponds to the MAG statement.                                                                                                                                                                                   |
| <nexp5>        | is the magnification in regard to width. Range is 1 to 4. Default is 1. Corresponds to the MAG statement.                                                                                                                                                                                    |
| <nexp6>        | is the width enlargement in percent relative to the height. Range is 1 to 1000 (for example, 200 doubles the width). Default is 100. Not valid for bitmap fonts. When using this parameter, all parameters in the syntax must be included in the statement (name, height, slant, and width). |
| <nexp7>        | is the insertion point for the bar code interpretation. A value between 1 and 9 can be set, corresponding to positions in the figure. The values 4, 5, and 6 are interpreted as 7, 8, and 9 respectively. Default is 0 (disabled). This function overrides the ALIGN command.                |
| <nexp8>        | is the horizontal offset in dots from the insertion point for the bar code interpretation. Default is 0. The offset is set with respect to the bar code direction, not necessarily the paper feed direction.                                                                                 |

*<nexp9>* is the vertical offset in dots from the insertion point for the bar code interpretation. Default is 0. The offset is set with respect to the bar code direction, not necessarily the paper feed direction.

**Remarks** Reset to default by executing a PRINTFEED.

If a MAG statement is executed after a BARFONT statement, the size of the barfont is affected by the MAG statement.

The printing of bar code interpretation can be enabled by a trailing ON, which corresponds to a BARFONT ON statement. Exceptions to this condition are as follows:

- In all EAN and UPC bar codes, the interpretation is an integrated part of the code. Such an interpretation is not affected by a BARFONT statement, but will be printed in accordance to specification, provided that interpretation printing has been enabled by a BARFONT ON statement.
- Certain bar codes, like Code 16K, cannot contain any interpretation at all. In this case, the selected barfont will be ignored.

**Example** Programming a Code 39 bar code, selecting the same barfont for all directions, and enabling the printing of the bar code interpretation, can be done this way:

```

10 PRPOS 30,400
20 DIR 1
30 ALIGN 7
40 BARSET "CODE39",2,1,3,120
50 BARFONT "Swiss 721 BT",10,8,5,1,1,100 ON
60 PRBAR "ABC"
70 PRINTFEED
80 END

```

## BARFONT ON/OFF (BF ON/OFF)

**Purpose** Enables or disables the printing of bar code interpretation.

**Syntax** BARFONT | BF ON | OFF [, <nexp>]

where *<nexp>* disables or enables guard bar printing for EAN/UPC bar codes. Default is enabled.

Reset to default by executing a PRINTFEED.

**Remarks** Typically, you start a print program by selecting a suitable bar code interpretation font with the BARFONT command. Then you use BARFONT ON and BARFONT OFF to control whether to print the interpretation or not, depending on the application.

BARFONT ON can be replaced by a BARFONT statement appended by a trailing ON. When printing EAN/UPC bar codes, the trailing parameter 0 can be specified after the BARFONT OFF command to disable the printing of the guard bars, which by default will print.

BARFONT OFF, 0 disables printing of the guard bars. BARFONT OFF,1 is the default (guard bars enabled) and is equivalent to BARFONT OFF.

**Example 1** This example sets up a Code 39 bar code, selects a barfont for each direction and enables the printing of the bar code interpretation. Compare with the example for BARFONT statement:

```
10 PRPOS 30,400
20 DIR 1
30 ALIGN 7
40 BARSET "CODE39",2,1,3,120
50 BARFONT "Swiss 721 BT",10,8,5,1,1
60 BARFONT ON
70 PRBAR "ABC"
80 PRINTFEED
90 END
```

**Example 2** This example sets up an EAN8 bar code and disables the guard bars:

```
10 BARTYPE "EAN8"
20 BARFONT OFF,0
30 PRBAR "1234567"
40 PRINTFEED
```

## **BARHEIGHT (BH)**

**Purpose** Statement specifying the height of a bar code.

**Syntax** BARHEIGHT | BH<nexp>

where <nexp> is the height of the bars in the bar code expressed in number of dots. Default is 100. Reset to default by executing a PRINTFEED.

**Remarks** In bar codes consisting of several elements on top of each other (for example, Code 16K) the barheight specifies the height of one element. The height is not affected by BARMAG statements. BARHEIGHT can be replaced by a parameter in the BARSET statement.

**Example** This example sets up a Code 39 bar code, selects a barfont for all directions and enables printing of the bar code interpretation:

```
10 PRPOS 30,400
20 DIR 1
30 ALIGN 7
40 BARTYPE "CODE39"
50 BARRATIO 2,1
60 BARHEIGHT 120
70 BARMAG 3
80 BARFONT "Swiss 721 BT"ON
90 PRBAR "ABC"
100 PRINTFEED
```

A more compact method is illustrated by the example for BARSET.

## BARMAG (BM)

**Purpose** Statement specifying a magnification for the width of the bars in a bar code.

**Syntax** BARMAG | BM<*nexp*>

where <*nexp*> is the magnification for the width of the bars in the bar code. Range depends on type of bar code. Default is 2.

Reset to default by executing a PRINTFEED.

**Remarks** The magnification only affects the bar code width, not the height of the bars (see BARHEIGHT). For example, by default BARRATIO is 3:1 and the BARMAG is 2, which means that the wide bars will be 6 dots wide and the narrow bars will be 2 dots wide ( $2 \times 3:1 = 6:2$ ).

The magnification also affects the interpretation in EAN/UPC bar codes, since the interpretation is an integrated part of the EAN/UPC code. BARMAG can be replaced by a parameter in the BARSET statement.

**Example** This example sets up a Code 39 bar code, selects a barfont for all directions and enables printing of the bar code interpretation:

```
10 PRPOS 30,400
20 DIR 1
30 ALIGN 7
40 BARTYPE "CODE39"
50 BARRATIO 2,1
60 BARHEIGHT 120
70 BARMAG 3
80 BARFONT "Swiss 721 BT" ON
90 PRBAR "ABC"
100 PRINTFEED
```

A more compact method is illustrated by the BARSET example.

## BARRATIO (BR)

**Purpose** Specifies the ratio between the wide and the narrow bars in a bar code.

**Syntax** BARRATIO | BR<*nexp1*>,<*nexp2*>

where:

<*nexp1*> is the thickness of the wide bars relative to the narrow bars.

<*nexp2*> is the thickness of the narrow bars relative to the wide bars.

Default ratio is 3:1.

Reset to default by executing a PRINTFEED.

**Remarks** This statement specifies the ratio between the wide and the narrow bars in a bar code. To decide the width of the bars in number of dots, the ratio must be multiplied by the BARMAG value.

For example:

The default BARRATIO is 3:1 and the default BARMAG is 2.

$(3:1) \times 2 = 6:2$

That is, the wide bars are 6 dots wide and the narrow bars are 2 dots wide.

Note that certain bar codes, such as EAN/UPC codes, have a fixed ratio. In those cases, BARRATIO is ignored. BARRATIO can be replaced by two parameters in the BARSET statement. For more information, see Chapter 5, “Bar Codes.”

- Example** This example sets up a Code 39 bar code, selects a barfont for all directions and enables printing of the bar code interpretation:

```
10 PRPOS 30,400
20 DIR 1
30 ALIGN 7
40 BARTYPE "CODE39"
50 BARRATIO 2,1
60 BARHEIGHT 120
70 BARMAG 3
80 BARFONT "Swiss 721 BT"ON
90 PRBAR "ABC"
100 PRINTFEED
```

A more compact method is illustrated by the BARSET example.

## **BARSET**

**Purpose** Specifies a bar code and sets additional parameters for complex bar codes.

**Syntax** BARSET [#<ncon>, ] [<sexp>[,<nexp1>[,<nexp2>[,<nexp3>[,<nexp4>[,<nexp5>[,<nexp6>[,<nexp7>[,<nexp8>[,<nexp9>[,<nexp10>[,<nexp11>]]]]]]]]]]]

where:

#<ncon> is the start parameter and specifies which parameter in the syntax is the first optional parameter (#1-#12). This way you may bypass some of the initial parameters (for example, ratio and enlargement). Default is #1.

<sexp> (#1) is the bar code type. No default. Corresponds to BARTYPE.

<nexp1> (#2) is the ratio of the large bars. Default is 3. Corresponds to BARRATIO.

<nexp2> (#3) is the ratio of the small bars. Default is 1. Corresponds to BARRATIO.

<nexp3> (#4) is the enlargement. Default is 2. Corresponds to BARMAG.

<nexp4> (#5) is the height of the code in dots. Default is 100. Corresponds to BARHEIGHT. For QR Code, this parameter specifies the model (1 or 2).

<nexp5> (#6) bar code specific parameter. Default is 2.

<nexp6> (#7) bar code specific parameter. Default is 3.

<exp7> (#8) bar code specific parameter. Default is 1.  
 <exp8> (#9) bar code specific parameter. Default is 0.  
 <exp9> (#10) bar code specific parameter. Default is 0.  
 <exp10> (#11) bar code specific parameter. Default is 0.  
 <exp11> (#12) bar code specific parameter. Default is 0.

Reset to default by executing a PRINTFEED. For more information on bar code specific parameters, see Chapter 5, “Bar Codes.”

**Remarks** This statement can replace the statements BARHEIGHT, BARRATIO, BARTYPE, and BARMAG. Although primarily intended for 2-dimensional and composite bar codes, BARSET can be used for any type of bar code if irrelevant parameters are left out (for example, <exp5> to <exp11>).



**Note:** The parameter descriptions under the syntax do not apply to all bar codes that can be specified using BARSET. Some bar codes have other meanings or use certain parameters only as placeholders.

Bar code specific parameters have different meanings for each bar code, or are used only as placeholders. Refer to the descriptions of the various 2D and composite bar codes in Chapter 5 for complete syntax descriptions and programming examples.

**Example** This example shows how to use a BARSET statement to specify a Code 39 bar code:

```

10 PRPOS 30,400
20 DIR 1
30 ALIGN 7
40 BARSET "CODE39",2,1,3,120
50 BARFONT "Swiss 721 BT",10,8,5,1,1 ON
60 PRBAR "ABC"
70 PRINTFEED

```

Compare this to the BARTYPE example.

## BARTYPE (BT)

**Purpose** Statement specifying the type of bar code.

**Syntax** BARTYPE | BT<sexp>

where <sexp> specifies the type of bar code and must be a valid bar code name. There is no default value.

**Remarks** The selected bar code type must exist in the printer’s memory and must be entered as a string expression. For a list of the bar codes that are included in the Intermec Fingerprint firmware and their respective designations, see Chapter 5, “Bar Codes.” BARTYPE can be replaced by a parameter in the BARSET statement.

**Example** This example sets up a Code 39 bar code, selects a barfont for all directions, and enables printing of the bar code interpretation:

```
10 PRPOS 30,400
20 DIR 1
30 ALIGN 7
40 BARTYPE "CODE39"
50 BARRATIO 2,1
60 BARHEIGHT 120
70 BARMAG 3
80 BARFONT "Swiss 721 BT" ON
90 PRBAR "ABC"100 PRINTFEED
```

A more compact method is illustrated by the BARSET example.

## BEEP

**Purpose** Statement ordering the printer to emit a beep.

**Syntax** BEEP

**Remarks** This statement makes the printer emit a beep of ~800 Hz for 1/4 of a second. If a different frequency and/or duration is desired, use a SOUND statement instead.



**Note:** The EasyCoder PD41 does not have a beeper and makes no sound, but printer activity waits for the length of the BEEP duration before continuing.

**Example** In this example, the printer beeps when an error occurs:

```
10 ON ERROR GOTO 1000
.....
.....
.....
1000 BEEP
1010 RESUME NEXT
```

## BREAK

**Purpose** Statement specifying a break interrupt character separately for the keyboard and each serial communication channel.

**Syntax** BREAK<exp1>,<exp2>

where:

<exp1> is one of the following devices:

```
0 = "console:"
1 = "uart1:"
2 = "uart2:"
3 = "uart3:"
7 = "uart4:"
8 = "uart5"
```

*<nexp2>* is the decimal ASCII value for the break interrupt character.

Default: Comm. channels: ASCII 03 decimal

Console: ASCII 158 decimal (<Shift> + <Pause>)

**Remarks** The execution of a program can be interrupted using a method specified by the BREAK statement. You can also interrupt or resume printing by pressing the **Pause** or the **Print** key on the printer front panel.

To issue a break interrupt, by default, hold down **Shift** and press **Pause**. Together these keys produce the ASCII character 158 decimal (128 + 30). It is not possible to perform a break from the console on the PD41 printer.

It is possible to remap the keyboard, which may affect the keys used for break interrupt. Please refer to the variable KEYBMAP\$.

Another method is to transmit the character ASCII 03 decimal (default) to the printer on one of the serial communication channels. The execution will be interrupted regardless of any INPUT waiting (that is, INPUT [#], LINE INPUT [#], and INPUT\$).

The BREAK statement allows you to specify other ways of interrupting the execution, such as by pressing another combination of keys on the printer keyboard or transmitting another ASCII character from the host.

A specified break interrupt character is saved in the temporary memory until the printer is restarted or REBOOTed, which may be confusing when switching between programs. To change a break interrupt character, specify a new one for the same device using a BREAK statement, or to remove it from memory use a BREAK OFF statement.

The use of break interrupt is enabled or disabled separately for each device by BREAK ON or BREAK OFF statements. By default, break interrupt on the “console:” is enabled, and break interrupt on any of the communication channels is disabled.

Intermec strongly recommends that you include some facility for issuing a break interrupt from the host computer in startup (autoexec) files. If not, you may find yourself with an erroneous program running in a loop without being able to break it!

**Example 1** In this example, the ASCII character 127 decimal is selected and enabled as BREAK character on the communication channel “uart1:”:

```
10 BREAK 1,127
20 BREAK 1 ON
.....
.....
.....
```

**Example 2** In this example, BREAK characters are specified for both the keyboard (“console:”) and the serial communication channel “uart1:”. The loop can be interrupted either by pressing the key usually marked “F1” on the printer’s keyboard, or by typing an uppercase X on the keyboard of the host:

```
10 BREAK 0,1:BREAK 1,88
20 BREAK 0 ON:BREAK 1 ON
30 GOTO 30
RUN
```

Reset BREAK to default by turning the printer off and on.

## **BREAK ON/OFF**

**Purpose** Statement enabling or disabling break interrupt separately for the keyboard and each serial communication channel.

**Syntax** BREAK<exp>ON | OFF

where <exp> is one of the following devices:

0 = “console:”  
1 = “uart1:”  
2 = “uart2:”  
3 = “uart3:”  
7 = “uart4:”  
8 = “uart5”

By default, the COM ports are disabled and the console is enabled.

**Remarks** The use of the break interrupt specified by a BREAK statement can be enabled or disabled separately for each serial communication channel, or for the printer’s built-in keyboard by BREAK ON or BREAK OFF statements.

By default, break interrupt is enabled from the printer’s keyboard and disabled from all communication channels. BREAK OFF deletes any existing break interrupt character stored in the printer’s temporary memory for the specified device.



**Note:** It is not possible to perform a break from the console on the PD41 printer.

**Example** In this example, the ASCII character 127 decimal is selected and enabled as BREAK character on the communication channel “uart1:”. At the same time, BREAK from the printer’s keyboard is disabled.

```
10 BREAK 1,127
20 BREAK 1 ON:BREAK 0 OFF
.....
.....
.....
```

## BUSY

**Purpose** Statement ordering a busy signal, for example XOFF, CTS/RTS, or PE, to be transmitted from the printer on the specified communication channel.

**Syntax** BUSY [<nexp>]

where <nexp> is optional and specifies the channel as:

- 1 = "uart1:"
- 2 = "uart2:"
- 3 = "uart3:"
- 4 = "centronics:"
- 7 = "uart4:"
- 8 = "uart5"

**Remarks** Communication protocol usually contains some “busy” signal, which tells the host computer that the printer is unable to receive data.

The BUSY statement allows you to order a busy signal to be transmitted on the specified communication channel. If no channel is specified, the signal will be transmitted on the standard OUT communication channel. For more information, see [“SETSTDIO” on page 185](#).

To allow the printer to receive more data, use a READY statement.

For the optional “centronics:” communication channel, BUSY/READY control the PE (paper end) signal on pin 12 according to an error-trapping routine (BUSY = PE high).

**Example** This example prevents the printer from receiving more data on “uart2:” during the process of printing a label (running this example requires an optional interface board to be fitted):

```
10 FONT "Swiss 721 BT"
20 PRTXT "HELLO!"
30 BUSY2
40 PRINTFEED
50 READY2
RUN
```

## CHDIR

**Purpose** Statement specifying the current directory.

**Syntax** CHDIR<scon>

where <scon> specifies the current directory (see DEVICES). Default is “/c”.

**Remarks** By default, the printer’s permanent memory (“/c”) is the current directory. The current directory is used if the Intermec Fingerprint instruction does not contain any reference to a directory (such as FILES).

This implies that to access the temporary memory (“tmp:”), the storage part of the RTC/Dallas key circuit (“storage:”), or an optional memory card (“/rom” or “card1:”), you must include such a reference in your instructions (such as FILES “/rom”).

The CHDIR statement allows you to appoint another directory than “/c” as the current directory, implying that you must specify the permanent memory (“/c”) whenever you want to access it.

- Example** In this example, the current directory is changed to “card1:”, all files in “card1:” are listed, and finally the current directory is changed back to “/c”. This example is included to illustrate the principles of changing the current directory. It is more efficient to use FILES “card1:” to read its contents.

```
10 CHDIR"card1:"
20 FILES
30 CHDIR"/c"
RUN
```

## **CHECKSUM**

- Purpose** Statement calculating the checksum of a range of program lines in connection with the transfer of programs.
- Syntax** CHECKSUM(<nexp1>, <nexp2>)  
where:  
  <nexp1> is the number of the first line in a range of program lines.  
  <nexp2> is the number of the last line in a range of program lines.
- Remarks** The checksum is calculated from parts of the internal code using an advanced algorithm. Intermec recommends that you let the printer calculate the checksum before the transfer of a program. After the transfer is completed, let the receiving printer do the same calculation and compare the checksums.
- Example** In this example, the checksum is calculated for all program lines between line 10 and line 2000 in the program “DEMO.PRG”.

```
NEW
LOAD "DEMO . PRG"
PRINT CHECKSUM(10, 2000)
```

results in:

60095

## **CHR\$**

- Purpose** Returns the readable character from a decimal ASCII code.
- Syntax** CHR\$(<nexp>)  
where <nexp> is the decimal ASCII code to be converted to a readable character.

**Remarks** This function is useful for entering characters that cannot be produced from the keyboard of the host, such as non-printable characters ASCII 0-31 dec. Only integers between 0 and 255 are allowed. Values less than 0 or larger than 255 will result in an error 41, “Parameter out of range.”

**Example** In this example, the decimal ASCII code for “A” is 65 and for “B” is 66.

```
10 A$ = CHR$(65)
20 B$ = CHR$(40+26)
30 PRINT A$
40 PRINT B$
RUN
```

results in:

```
A
B
```

## CLEANFEED

**Purpose** Statement running the printer’s feed mechanism.

**Syntax** CLEANFEED<*nexp*>

where <*nexp*> is the feed length expressed as a positive or negative number of dots.

**Remarks** The CLEANFEED statement activates the stepper motor that drives the printer’s platen roller (the rubber roller beneath the printhead). In case of thermal transfer printers, it also often drives the ribbon mechanism. The motor runs regardless of possible error conditions, such as if the printhead is lifted, or if there is no ribbon or media supply left. Thus, the CLEANFEED statement is suitable for cleaning and for the loading of transfer ribbon.

A positive CLEANFEED value makes the stepper motor rotate the rollers forward, such as when feeding out a label.

A negative CLEANFEED value makes the stepper motor rotate the rollers backwards, such as when pulling back a label.

Executing a CLEANFEED, as opposed to TESTFEED, does not affect the adjustment of the label stop sensor or black mark sensor, regardless of what type of media or other object passes the sensor.

Note that CLEANFEED, as opposed to FORMFEED, always must be specified with regard to feed length.

**Example** This example pulls a cleaning card back and forth under the printhead three times. To set this up, three 1200-dot long positive CLEANFEEDs and three 1200-dot long negative CLEANFEEDs are performed:

```
10 FOR A%=1 TO 3
20 CLEANFEED 1200
30 CLEANFEED -1200
40 NEXT
RUN
```

## **CLEAR**

**Purpose** Statement clearing strings, variables, and arrays in order to free memory space.

**Syntax** CLEAR

**Remarks** The CLEAR statement empties all strings, sets all variables to zero, and resets all arrays to their default values, making more free memory space available.

**Example** In this example, the CLEAR statement empties the strings to free up memory:

```
10 A$ = "ABCDEFGHIJKLMNPQRSTUVWXYZ"
20 B$ = "abcdefghijklmnopqrstuvwxyz"
30 FOR I%=0 TO 3 :FOR J%=0 TO 3 :FOR K%=0 TO 20
40 C$(I%,J%)=C$(I%,J%)+A$
50 NEXT K%:NEXT J%:NEXT I%
60 PRINT "String A before: ";A$
70 PRINT "String B before: ";B$
80 PRINT "Free memory before: ";FRE(1)
90 CLEAR
100 PRINT "String A after: ";A$
110 PRINT "String B after: ";B$
120 PRINT "Free memory after: ";FRE(1)
RUN
```

results in:

```
String A before: ABCDEFGHIJKLMNOPQRSTUVWXYZ
String B before: abcdefghijklmnopqrstuvwxyz
Free memory before: 1867368
String A after:
String B after:
Free memory after: 1876200
Ok
```

## **CLIP**

**Purpose** Statement for enabling/disabling the printing of partial fields.

**Syntax** CLIP [BARCODE [HEIGHT | INFORMATION | X | Y] ] [ON | OFF]

where:

BARCODE              toggles between partial bar code fields enable/disable.

HEIGHT              clips the height of the bar so the bar code will fit inside the print window. A one-dimensional bar code may still be readable.

INFORMATION        clips the bar code lengthwise, so some bars will be missing, making the bar code unreadable.

X                    clips the part of the bar code that is outside the X-dimension of the print window.

**Y** clips the part of the bar code that is outside the Y-dimension of the print window.

**ON** enables use of partial text, image, line, and box fields.

**OFF** disables use of partial text, image, line, and box fields.

**Remarks** “Partial fields” means that the firmware will accept and print text, bar code, image, lines, and box fields even if they extend outside the print window as specified by the printer setup with regard to X-Start, Width, and Length.

Negative PRPOS values are allowed. However, all parts of the fields outside the print window are excluded from the printout and the field will be clipped at the borders of the print window.

There are two main cases:

- CLIP BARCODE [HEIGHT|INFORMATION|X|Y] is used for bar code fields only. Note that some bar codes, like Maxicode, consist of images and should in this context be regarded as image fields.
- CLIP ON|OFF is only used for text, image, line, and box fields. When the use of partial fields is disabled, error 1003 (“Field out of label”) results if any field extends outside the print window. Note the difference between the physical size of the label and the size of the print window specified by the printer setup, which determines where the fields will be clipped.

**Example** In this example, only the last part of the text will be printed:

```
10 CLIP ON
20 PRPOS -100,30
30 PRTXT "INTERMEC PRINTER"
40 PRINTFEED
RUN
```

## CLL

**Purpose** Statement for partial or complete clearing of the print image buffer.

**Syntax** CLL [<nexp>]

where <nexp> (optional) specifies the field from which the print image buffer should be cleared. If no value for <nexp> is specified, the entire print image buffer is cleared.

**Remarks** The print image buffer stores the printable image after processing while awaiting the printing to be executed. The buffer can be cleared, partially or completely, by the use of a CLL statement:

- CLL<nexp> partially clears the buffer from a specified field to the end of the program. The field is specified by a FIELDNO function. Partial clearing is useful in connection with print repetition. To avoid superfluous reprocessing, one or several fields can be erased from the

buffer and be replaced by other information, while the remaining parts of the label are kept in the buffer.

Note that there must be no changes in the layout between the PRINTFEED and the CLL statements, or the layout will be lost. Also note that partial clearing always starts from the end, which means that the fields which are executed last are cleared first.

- CLL (without any field number) clears the buffer completely. When certain error conditions have occurred, it is useful to be able to clear the print image buffer without having to print a faulty label. Should the error be attended to, without the image buffer being cleared, there is a risk that the correct image will be printed on top of the erroneous one on the same label. It is therefore advisable to include a CLL statement in your error-handling subroutines, when you are working with more complicated programs, in which all implications may be difficult to grasp.

**Example 1** This example demonstrates partial clearing. Two labels are printed, each with two lines of text. After the first label is printed, the last line is cleared from the print image buffer and a new line is added in its place on the second label:

```
10 PRPOS 100,300
20 FONT "Swiss 721 BT"
30 PRTXT "HAPPY"
40 A%=FIELDNO
50 PRPOS 100,250
60 PRTXT "NEW YEAR! "
70 PRINTFEED
80 CLL A%
90 PRPOS 100,250
100 PRTXT "BIRTHDAY! "
110 PRINTFEED
RUN
```

**Example 2** This example demonstrates complete clearing. The print image buffer is completely cleared if error 1030 ("Character missing in chosen font") occurs.

```
10 ON ERROR GOTO 1000
.....
.....
.....
1000 IF ERR=1030 GOSUB 1100
1010 RESUME NEXT
.....
.....
1100 CLL
1110 PRINT "CHARACTER MISSING"
1120 RETURN
```

## CLOSE

**Purpose** Statement closing one or several files and/or devices for input/output.

**Syntax** CLOSE [ [#] <nexp> [, [#] <nexp>... ] ]

where:

# (optional) indicates that whatever follows is a number.

<nexp> is the number assigned to a file or device when it was OPENed.

**Remarks** This statement revokes OPEN. Only files or devices which already have been OPENed, can be CLOSEd.

A CLOSE statement for a file or device OPENed for sequential output indicates that the data in the buffer will be written to the file/device in question automatically before the channel is closed.

When a file OPENed for random access is CLOSEd, all its FIELD definitions will be lost.

END, NEW, and RUN also close all open files and devices.

**Examples** This statement closes all open files and devices:

200 CLOSE

A number of files or devices (No. 1-4) can be closed simultaneously using any of the following types of statement:

200 CLOSE 1,2,3,4

or

200 CLOSE #1,#2,#3,#4

or

200 CLOSE 1,2,#3,4

## COM ERROR ON/OFF

**Purpose** Statement enabling/disabling error handling on the specified communication channel.

**Syntax** COM.ERROR<nexp>ON | OFF

where <nexp> is one of the following communication channels:

- 1 = "uart1:"
- 2 = "uart2:"
- 3 = "uart3:"
- 4 = "centronics:"
- 7 = "uart4:"
- 8 = "uart5"

Default is COM ERROR OFF on all channels.

**Remarks** This function is closely related to COMSET, ON COMSET GOSUB, COMSET ON, COMSET OFF, COMSTAT, and COMBUF\$. Each character received is checked for the following errors:

- Received break
- Framing error
- Parity error
- Overrun error

If any of these errors occur and COM ERROR is ON for the channel in question, reception will be interrupted. This condition can be read by means of a COMSTAT function, but you cannot read exactly what type of error has occurred. COM ERROR OFF disables this type of error handling for the specified channel.

COM ERROR ON cannot be used with USB (COM channel #6).

**Example** In this example, a message appears on the printer display if reception is interrupted by any of the four specified COMSET conditions:

```
10 COM ERROR 1 ON
20 A$="Max. number of char. received"
30 B$="End char. received"
40 C$="Attn. string received"
50 D$="Communication error"
60 COMSET 1, "A", CHR$(90), "#", "BREAK", 20
70 ON COMSET 1 GOSUB 1000
80 COMSET 1 ON
90 IF QDATA$="" THEN GOTO 90
100 END
1000 QDATA$=COMBUF$(1)
1010 IF COMSTAT(1) AND 2 THEN PRINT A$
1020 IF COMSTAT(1) AND 4 THEN PRINT B$
1030 IF COMSTAT(1) AND 8 THEN PRINT C$
1040 IF COMSTAT(1) AND 32 THEN PRINT D$
1050 PRINT QDATA$:RETURN
```

## **COMBUF\$**

**Purpose** Reads the data in the buffer of the communication channel specified by a COMSET statement.

**Syntax** COMBUF\$ (<nexp>)

where <nexp> is one of the following communication channels:

```
1 = "uart1:"
2 = "uart2:"
3 = "uart3:"
4 = "centronics:"
5 = "net1:"
6 = "usb1:"
7 = "uart4:"
8 = "uart5"
```

**Remarks** This function is closely related to COMSET, ON COMSET GOSUB, COMSET ON, COMSET OFF, COM ERROR ON/OFF, and COMSTAT.

Using COMBUF\$, the buffer can be read and its content used in your program. When the communication has been interrupted by “end character”, “attention string”, or “max. no. of char.” (see COMSET), you may use an ON COMSET GOSUB subroutine and assign the data from the buffer to a variable as illustrated in the example below.

Note that COMBUF\$ filters out any incoming ASCII 00 dec. characters (NUL) by default. Filtering can be enabled/disabled using SYSVAR(44).

**Example** In this example, the data from the buffer is assigned to the string variable A\$ and printed on the screen:

```

1 REM Exit program with #STOP&
10 COMSET1,"#", "&","ZYX", "=", 50
20 ON COMSET 1 GOSUB 2000
30 COMSET 1 ON
40 IF A$ <> "STOP" THEN GOTO 40
50 COMSET 1 OFF
.....
.....
1000 END
2000 A$= COMBUF$(1)
2010 PRINT A$
2020 COMSET 1 ON
2030 RETURN

```

## COMSET

**Purpose** Statement setting the parameters for background reception of data to the buffer of a specified communication channel (see COMBUF\$).

**Syntax** COMSET<*nexp1*>, <*sexp1*>, <*sexp2*>, <*sexp3*>, <*sexp4*>, <*nexp2*>

where:

<*nexp1*> is one of the following communication channels:

```

1 = "uart1:"
2 = "uart2:"
3 = "uart3:"
4 = "centronics:"
5 = "net1:"
6 = "usb1:"
7 = "uart4:"
8 = "uart5"

```

<*sexp1*> specifies the start of the message string. Maximum 12 characters.

<*sexp2*> specifies the end of the message string. Maximum 12 characters.

<*sexp3*> specifies characters to be ignored. Maximum 42 characters.

<*sexp4*> specifies the attention string. Maximum 12 characters.

<nexp2> specifies the max. number of characters to be received. Enter a value > or = 1. If <nexp2> = 0, the first character is lost.

**Remarks** Data can be received by a buffer on each of the communication channels without interfering with the running of the current program. At an appropriate moment, the program can fetch the data in the buffer and use them according to your instructions. Such background reception has priority over any ON KEY GOSUB statement.

Related instructions are COMSTAT, ON COMSET GOSUB, COMSET ON, COMSET OFF, COM ERROR ON/OFF, and COMBUF\$. The communication channels are explained in connection with the DEVICES statement.

The start and end strings are character sequences which tell the printer when to start or stop receiving data. Each string can be up to 12 characters and may be “ ”.

It is possible to make the printer ignore certain characters. Such characters are specified in a string of up to 42 characters, where the order of the individual characters does not matter, and may be “ ”. The attention string (maximum 12 characters) interrupts reception and may be “ ”.

The length of the COMSET strings are checked before they are copied into the internal structure. If any of these strings are too long, error 26 (“Parameter too large”) occurs.

When the printer receives the specified maximum number of characters without previously encountering any end string or attention string, the transmission is interrupted. The maximum number of characters also decides how much of the memory will be allocated to the buffer.

The reception of data to the buffer can be interrupted by four conditions:

- If an end string is encountered.
- If an attention string is encountered.
- If the maximum number of characters is received.
- If error-handling is enabled for the communication channel in question (see COM ERROR ON/OFF) and a communication error occurs.

This condition can be checked by a COMSTAT function. Any interruption will have a similar effect as a COMSET OFF statement (interrupting reception), but the buffer will not be emptied and can still be read by a COMBUF\$ function. After reception is interrupted, an ON COMSET GOSUB statement can be issued to control what happens next. COMSET does not support auto-hunting (see SETSTDIO).

**Example** This example shows how to open “uart1:” for background communication.

Any record starting with the character # and ending with the character & will be received. The characters X, Y and Z are ignored. The character = stops reception. A maximum of 50 characters are allowed.

```

1 REM Exit program with #STOP&
10 COMSET1,"#", "&","ZYX", "=", 50
20 ON COMSET 1 GOSUB 2000
30 COMSET 1 ON
40 IF A$ <> "STOP" THEN GOTO 40
50 COMSET 1 OFF
.....
.....
1000 END
2000 A$= COMBUF$(1)
2010 PRINT A$
2020 COMSET 1 ON
2030 RETURN

```

## COMSET OFF

**Purpose** Statement turning off background data reception and emptying the buffer of the specified communication channel.

**Syntax** COMSET<*nexp*>OFF

where <*nexp*> is one of the following communication channels:

```

1 = "uart1:"
2 = "uart2:"
3 = "uart3:"
4 = "centronics:"
5 = "net1:"
6 = "usb1:"
7 = "uart4:"
8 = "uart5"

```

**Remarks** This statement is closely related to COMSET, ON COMSET GOSUB, COMSTAT, COMSET ON, COM ERROR ON/OFF, and COMBUF\$. The COMSET OFF statement closes the reception and empties the buffer of the specified communication channel.

**Example** In this example, the COMSET OFF statement closes "uart1:" for background reception and empties the buffer:

```

1 REM Exit program with #STOP&
10 COMSET1,"#", "&","ZYX", "=", 50
20 ON COMSET 1 GOSUB 2000
30 COMSET 1 ON
40 IF A$ <> "STOP" THEN GOTO 40
50 COMSET 1 OFF
.....
.....
1000 END
2000 A$= COMBUF$(1)
2010 PRINT A$
2020 COMSET 1 ON
2030 RETURN

```

## **COMSET ON**

**Purpose** Statement emptying the buffer and turning on background data reception on the specified communication channel.

**Syntax** COMSET<nbsp>ON

where <nbsp> is one of the following communication channels:

1 = "uart1:"  
2 = "uart2:"  
3 = "uart3:"  
4 = "centronics:"  
5 = "net1:"  
6 = "usb1:"  
7 = "uart4:"  
8 = "uart5"

**Remarks** This statement is closely related to COMSET, ON COMSET GOSUB, COMSTAT, COMSET OFF, COM ERROR ON/OFF, and COMBUF\$.

Use COMSET ON to open any of the communication channels for background data reception with an empty buffer, provided the communication parameter for the channel has already been set up by a COMSET statement.

When reception is interrupted by the reception of an end character, an attention string or the maximum number of characters, issue a new COMSET ON to empty the buffer and reopen reception.

**Example** In this example, the COMSET ON statement on line 30 opens "uart1:" for background reception. After the buffer has been read, it is emptied and the reception is reopened by a new COMSET ON statement in the subroutine on line 2020:

```
1 REM Exit program with #STOP&
10 COMSET1,"#", "&", "ZYX", "=", 50
20 ON COMSET 1 GOSUB 2000
30 COMSET 1 ON
40 IF A$ <> "STOP" THEN GOTO 40
50 COMSET 1 OFF
.....
.....
1000 END
2000 A$= COMBUF$(1)
2010 PRINT A$
2020 COMSET 1 ON
2030 RETURN
```

## COMSTAT

**Purpose** Reads the status of a communication channel buffer.

**Syntax** COMSTAT(<nexp>)

where <nexp> is one of the following communication channels:

- 1 = "uart1:"
- 2 = "uart2:"
- 3 = "uart3:"
- 4 = "centronics:"
- 5 = "net1:"
- 6 = "usb1:"
- 7 = "uart4:"
- 8 = "uart5"

**Remarks** This function is closely related to COMSET, ON COMSET GOSUB, COMSET ON, COMSET OFF, COM ERROR ON/OFF, and COMBUF\$. It allows you to find out if the buffer is able to receive background data, or if not, what condition has caused the interruption.

The buffer status is indicated by a numeric expression, which is the sum of the values given by the following conditions:

| Value  | Condition                                                      |
|--------|----------------------------------------------------------------|
| 0 or 1 | Copy of hardware handshake bit (not on "net1:" or "usb1:").    |
| 2      | Interruption: Maximum characters received.                     |
| 4      | Interruption: End character received.                          |
| 8      | Interruption: Attention string received.                       |
| 32     | Interruption: Communication error (not on "net1:" or "usb1:"). |

**Example** In this example, a message appears on the screen when the reception is interrupted by any of four specified COMSET conditions:

```

10 COM ERROR 1 ON
20 A$="Max. number of char. received"
30 B$="End char. received"
40 C$="Attn. string received"
50 D$="Communication error"
60 COMSET 1, "A",CHR$(90), "#", "BREAK", 20
70 ON COMSET 1 GOSUB 1000
80 COMSET 1 ON
90 IF QDATA$="" THEN GOTO 90
100 END
1000 QDATA$=COMBUF$(1)
1010 IF COMSTAT(1) AND 2 THEN PRINT A$
1020 IF COMSTAT(1) AND 4 THEN PRINT B$
1030 IF COMSTAT(1) AND 8 THEN PRINT C$
1040 IF COMSTAT(1) AND 32 THEN PRINT D$
1050 PRINT QDATA$
1060 RETURN

```

## **CONT**

**Purpose** Statement for resuming execution of a program that has been interrupted by means of a STOP, BREAK, or DBBREAK statement.

**Syntax** CONT

**Remarks** When you use a CONT statement to resume program execution after a STOP, BREAK, or DBBREAK, execution continues at the point where the break occurred with the STDIO settings restored.

CONT is usually used in conjunction with DBBREAK or STOP for debugging. When execution is stopped, you can examine or change the values of variables using direct mode statements, and then use CONT to resume execution. CONT is invalid if the program has been edited during the break.

It is also possible to resume execution at a specified program line using a GOTO statement in the immediate mode.

**Example**

```
10 A%=100
20 B%=50
30 IF A%=B% THEN GOTO QQQ ELSE STOP
40 GOTO 30
50 QQQ:PRINT "Equal"
Ok
RUN
Break in line 30
Ok
PRINT A%
100
Ok
PRINT B%
50
Ok

B%=100
Ok
CONT
Equal
Ok
```

## **COPY**

**Purpose** Statement for copying files.

**Syntax** COPY<sexp1>[,<sexp2>]

where:

<sexp1> is the name and (optional) directory of the original file.

<sexp2> (optional) is a new name and/or directory for the copy.

**Remarks** This statement allows you to copy a file to another name and/or directory as an alternative to LOADING the file in question and then SAVEing it.

If no directory is specified for the original and/or copy, the current directory is used by default. By default, the current directory is “/c”, which is the printer’s permanent memory. If the file is to be copied from or to another directory than the current one, the file name must contain a directory reference.

A file cannot be copied to the same name in the same directory. In addition to copying files to the printer’s permanent or temporary memory or a DOS-formatted memory card, a file can also be copied to an output device such as the printer’s display or a serial communication channel.

Copying a program to the standard OUT channel has the same effect as LOADING and LISTing it.

Note that bitmap fonts and images are not files and therefore cannot be copied.

In the next examples, “/c” is the current directory.

**Example 1** Copying a file from “card1:” to the current directory without changing the file name:

```
COPY "card1:LABEL1.PRG"
```

**Example 2** Copying a file from “card1:” to the current directory and changing the file name:

```
COPY "card1:FILELIST.PRG", "COPYTEST.PRG"
```

**Example 3** Copying a file from “/c” to a directory other than the current one without changing the file name:

```
COPY "/c/FILELIST.PRG", "card1:FILELIST.PRG"
```

**Example 4** Copying a file in the current directory to a new name within the same directory:

```
COPY "LABEL1.PRG", "LABEL2.PRG"
```

**Example 5** Copying a file in the current directory to serial channel “uart1:”:

```
COPY "LABEL1.PRG", "uart1:"
```

## COUNT&

**Purpose** Statement for creating a counter (Intermec Direct Protocol only).

**Syntax** COUNT& <sexp1>, <nexp1>, <sexp2>

where:

<sexp1> is the type of counter parameter to be set:

START: Start value

WIDTH: Minimum number of digits

COPY: Number of copies before update

INC: Increment/decrement at update

STOP: Stop value

RESTART: Restart counting at this value

<nexp1> is the counter reference number (integers only).

<sexp2> is the parameter value.

**Remarks** This instruction can only be used in the Intermec Direct Protocol.

The counters can be used in text and bar code fields. The counters are global, which means that they are not connected to any special label or layout, but will be updated at every execution of PRINTFEED statements where the counter in question is used.

Counters are designated using positive integers. When used for printing, they are referred to by “CNT<ncon>\$” variables, where <ncon> is the number of the counter as specified by COUNT&, for example CNT5\$. A counter variable without a matching counter is regarded as a common string variable.

The value of the start, stop, and restart parameters decide the type of counter (alpha or numeric). If different types of counters are specified in these parameters, the last entered parameter decides the type. Alpha counters count A to Z. Numeric counters use numbers with no practical limit.

Counters are not saved in the printer memory, but are recreated after each power up. Therefore, you may need to save the COUNT& statements as a file in the host.

## **START**

Decides the first value to be printed:

- If a single letter is entered (A-Z), the counter becomes an alpha counter. Default is A for alpha counters.
- If one or several digits are entered the counter is numeric. Numeric values can be positive or negative. Negative values are indicated by a leading minus sign. Default is 1 for numeric counters.

## **WIDTH**

This parameter can only be used in numeric counters and sets the minimum number of digits to be printed. If the counter value contains a lesser number of digits, leading zero (0) characters are added until the specified number of digits is obtained. If the number of digits in the counter value is equal to or larger than specified in the width parameter, the value is printed in its entirety. Default is 1 (no leading zeros).

## **COPY**

Determines how many copies (of labels, graphics, and so on) will be printed before the counter is updated according to the INC parameter. Default is 1.

## **INC**

Sets the value by which the counter should be incremented or decremented when it is updated. In case of decrementation, the value should contain a leading minus sign. Default is 1.

**STOP**

Decides the value after which the counter should start all over again at the value specified by the RESTART parameter. If a single letter is entered (A to Z), the counter becomes an alpha counter, and if one or several digits are entered the counter is numeric. When a counter is decremented, a stop value less than the start value must be given, since the default stop value will never be reached otherwise. Default is 2,147,483,647 (numeric) or Z (alpha).

**RESTART**

Sets the value at which the counter starts over again after exceeding the STOP parameter value. If a single letter is entered (A to Z), the counter becomes an alpha counter, and if one or several digits are entered the counter is numeric. Default is 1 (numeric) or A (alpha).

- Examples** This example creates a counter that starts at 100 and is updated by a value of 50 after every second label until the value 1000 is reached. Then the counter starts again at the value 200. All values are expressed as 4-digit numbers with leading zeros.

```
COUNT& "START",1,"100" .
COUNT& "WIDTH",1,"4" .
COUNT& "COPY",1,"2" .
COUNT& "INC",1,"50" .
COUNT& "STOP",1,"1000" .
COUNT& "RESTART",1,"200" .
```

**CURDIR\$**

- Purpose** Returns the current directory as the printer stores it.

**Syntax** CURDIR\$

- Example** CHDIR "/c/DIR1/DIR2"
PRINT CURDIR\$

results in:

/c/DIR1/DIR2

**CSUM**

- Purpose** Statement calculating the checksum of an array of strings.

**Syntax** CSUM<ncon>, <svar>, <nvar>

where:

<ncon> is the type of checksum calculation:

1: Longitudinal Redundancy Check (LRC)  
XOR in each character in each string array[0][0] xor array[0][1]  
... array[n][n]

2: Diagonal Redundancy Check (DRC)  
right rotation, then XOR on each character in each string  
rot(array[0][0] xor array[0][1])

3: Longitudinal Redundancy Check (LRC)  
Strip string of DLE (0x10) before doing the LRC

*<svar>* If *<ncon>* = 1 or 2: The array of strings of which the checksum is to be calculated.

If *<ncon>* = 3: Checksum string.

*<nvar>* is the variable in which the result will be presented.

**Remarks** These types of checksum calculation can only be used for string arrays, not for numeric arrays. In case of CSUM 3,*<svar>*,*<nvar>*, the resulting variable will be the in data for next CSUM calculation, unless the variable is reset.

**Example** This example calculates the DRC checksum of an array of strings:

```
10 ARRAY$ (0) = "ALPHA"
20 ARRAY$ (1) = "BETA"
30 ARRAY$ (2) = "GAMMA"
40 ARRAY$ (3) = "DELTA"
50 CSUM 2,ARRAY$,B%
60 PRINT B% :REM DRC CHECKSUM
RUN
```

results in:

252

## CUT

**Purpose** Statement activating an optional cutter.

**Syntax** CUT

**Remarks** This statement only works with printers fitted with a cutter, which cuts non-adhesive paper strip or through the liner between self-adhesive labels.

When a PRINTFEED statement is executed, the printer feeds out a certain amount of media according to the startadjust and stopadjust, as explained in the printer user's guide. Then the cutter can be activated by a CUT statement.

**Example** This program orders the printer to print text and then cut off the media:

```
10 PRPOS 250,250
20 DIR 1
30 ALIGN 4
40 FONT "Swiss 721 BT"
50 PRTXT "Hello everybody!"
60 PRINTFEED
70 CUT
RUN
```

## CUT ON/OFF

- Purpose** Statement enabling or disabling automatic cutting after PRINTFEED execution. Optionally, CUT ON/OFF can adjust the media feed before and after the cutting.
- Syntax** CUT [<nexp>] ON | CUT OFF  
where <nexp> is optional and sets the length of media to be fed out before cutting and pulled back after cutting. Default is CUT OFF.
- Remarks** This statement makes it possible to enable or disable automatic execution of a CUT operation directly after the execution of each PRINTFEED statement.  
If any extra media feed in connection with the cutting operation is required, use startadjust and stopadjust setup, or specify the desired length of media to be fed out before the cutting is performed and pulled back afterwards in the CUT ON statement.  
The amount of media feed (<nexp>) will not automatically be reset to 0 (zero) by a CUT OFF statement, but must be manually be specified as 0 (CUT 0 ON:CUT OFF). However, rebooting the printer resets it to 0.
- Example** This example enables automatic cutting and orders the printer to print text and feed out an extra amount of strip before cutting the media. The media is then pulled back the same distance:

```
10 CUT 280 ON
20 PRPOS 250,250
30 DIR 1
40 ALIGN 4
50 FONT "Swiss 721 BT"
60 PRTXT "Hello everybody!"
70 PRINTFEED
RUN
```

## DATE\$

- Purpose** Variable for setting or returning the current date.
- Syntax 1** Setting the date:  
DATE\$=<sexp>  
where <sexp> sets the current date by a 6-digit number specifying the year, month, and day in the format YYMMDD.
- Syntax 2** Returning the date:  
<svar>=DATE\$ [ (<sexp>) ]  
where:  
<svar> returns the current date according to the printer's calendar.  
<sexp> (optional) is a flag "F", indicating that the date will be returned according to the format specified by FORMAT DATE\$.

**Remarks** This variable works best if a real-time clock circuit (RTC) is fitted on the printer's CPU board. The RTC has a battery backup circuit and keeps record of the time if the power is turned off or lost.

If no RTC is installed, the internal clock is used. After startup, an error occurs when trying to read the date or time before the internal clock has been manually set by means of either a DATE\$ or a TIME\$ variable. If only the date is set, the internal clock starts at 00:00:00 and if only the time is set, the internal clock starts at Jan 01, 1980.

After setting the internal clock, you can use the DATE\$ and TIME\$ variables the same way as when an RTC is included, until a power off or REBOOT causes the date and time values to be lost.

Date is always entered and, by default, returned in the format YYMMDD, where:

- YY = Last two digits of the year (for example, 2007 = 07).
- MM = Two digits representing the month. Range is 01 to 12.
- DD = Two digits representing the day. Range is 01 to 31.

Example: December 1, 2007 is entered as "071201".

The built-in calendar corrects illegal values for the years 1980-2048, for example the illegal date 031232 will be corrected to 040101. The format for how the printer returns dates can be changed by means of a FORMAT DATE\$ statement and returned by DATE\$("F").

**Example** This example sets and returns the date in two different formats:

```
10 DATE$ = "071201" (sets date)
20 FORMAT DATE$ "DD/MM/YYYY" (sets date format)
30 PRINT DATE$ (returns unformatted date)
40 PRINT DATE$("F") (returns formatted date)
RUN
```

results in:

```
071201
01/12/2007
```

## **DATEADD\$**

**Purpose** Returns a new date after a number of days have been added to, or subtracted from, the current date or a specified date.

**Syntax** DATEADD\$ ( [<sexp1>, ] <nexp> [ , <sexp2> ] )

where:

<sexp1> is any date given according to the DATE\$ format.

<nexp> is the number of days to be added to (or subtracted from) the current date or (optional) the date specified by <sexp1>.

<sexp2> (optional) is one or two flags, "F" or "M":

“F” specifies that the date will be returned according to the format specified by FORMAT DATE\$.

“M” specifies the value given in <nexp> indicates months instead of days.

**Remarks** The original date (<sexp1>) should be entered according to the syntax for the DATE\$ variable, that is in the order YYMMDD, where:

- YY = Last two digits of the year (for example, 2007 = 07).
- MM = Two digits representing the month. Range is 01 to 12.
- DD = Two digits representing the day. Range is 01 to 31.

Example: December 1, 2007 is entered as “071201”.

The built-in calendar corrects illegal values for the years 1980-2048, for example the illegal date 031232 will be corrected to 040101. When for example, adding one month to the date 2008-01-31, the result will be rounded off to 2008-02-29 (2004 is a leap year).

Specify the number of days to be added or subtracted as a positive or negative numeric expression.

If no “F” flag is included in the DATEADD\$ function, the result is returned according to the DATE\$ format, see above.

If the DATEADD\$ function includes an “F” flag, the result is returned in the format specified by FORMAT DATE\$.

“F” and “M” flags can be combined in the same instruction without any separating character, that is, “FM” or “MF”. No other characters are accepted.

### Examples

```

10 DATE$ = "071201"
20 A%=15
30 B%=-10
40 FORMAT DATE$ "DD/MM/YY"
50 PRINT DATEADD$ ("071201",A%)
60 PRINT DATEADD$ ("071201",A%, "F")
70 PRINT DATEADD$ (B%, "F")
RUN

```

resulting in:

```

071216
16/12/07
21/11/07
DATE$="040131"
FORMAT DATE$ "YYYY/MM/DD"
? DATEADD$(1,"F"),DATEADD$(1,"M"),DATEADD$(1,"FM")
2008/02/01 080229 2008/02/29
Ok
? DATEADD$ ("080131",1,"F"),_←
←DATEADD$ ("080131",2,"M"),←
_←DATEADD$ (080131",3,"FM")
2008/02/01 080331 2008/04/30
Ok

```

## DATEDIFF

**Purpose** Returns the difference between two dates as a number of days.

**Syntax** DATEDIFF(<sexp1>, <sexp2>)

where:

<sexp1> is one of two dates (date 1).

<sexp2> is the other of two dates (date 2).

**Remarks** To get the result as a positive numeric value, the two dates should be entered with the earlier of the dates (date 1) first and the later of the dates (date 2) last. See the first example below.

If the later date (date 2) is entered first, the resulting value will be negative, as seen in the second example.

Both dates should be entered according to the syntax for the DATE\$ variable, that is in the order YYMMDD, where:

- YY = Last two digits of the year (for example, 2007 = 07).
- MM = Two digits representing the month. Range is 01 to 12.
- DD = Two digits representing the day. Range is 01 to 31.

Example: December 1, 2007 is entered as "071201".

The built-in calendar corrects illegal values for the years 1980-2048, for example the illegal date 071232 will be corrected to 080101.

**Example 1** This example calculates the difference in days between October 1, 2007 and November 30, 2007:

```
10 A% = DATEDIFF("071001", "071130")
20 PRINT A%
RUN
```

results in:

60

**Example 2** This example demonstrates what happens if the later date is entered first:

```
10 A% = DATEDIFF("071130", "071001")
20 PRINT A%
RUN
```

results in:

-60

## DBBREAK

- Purpose** Statement for adding or deleting a breakpoint for the Fingerprint Debugger.
- Syntax** DBBREAK<*nexp*> | <*sexp*> [ON | OFF]  
where:  
 <*nexp*> is the line number where the debugger breaks, and serves as the name of the breakpoint.  
 <*sexp*> is the line label where the debugger breaks, and serves as the name of the breakpoint.  
 ON adds the specified breakpoint (default).  
 OFF deletes the specified breakpoint.
- Remarks** The execution of a program will break at each program line specified as a breakpoint, and the message “break in line *nnn*” is transmitted on the Debug STDOUT port. If a CONT statement is issued, the execution continues at the next line, whereas if RUN is issued, the execution starts again from the first program line.  
 The line number or line label does not have to exist when a breakpoint is added, but if a nonexistent breakpoint is deleted an error 39 or 70 occurs.  
 There is no error given if a breakpoint is added more than once. When a breakpoint is deleted, all breakpoints with the same name are deleted at the same time. There will only be one break for each line even if there are more than one breakpoint on that line.  
 When a NEW statement is issued, all breakpoints are deleted. If a breakpoint is set on a line with a call to a FOR or WHILE loop, there will only be one break on that line (the first time it is executed).  
 Related instructions are DBBREAK OFF, DBEND, DBSTDIO, DBSTEP, and STOP.

**Example**

```
10 PRINT "A"
20 PRINT "B"
30 PRINT "C"
DBBREAK 20 ON
RUN
```

results in:

```
A
Break in line 20
```

## DBBREAK OFF

- Purpose** Statement for deleting all breakpoints for the Fingerprint Debugger.
- Syntax** DBBREAK OFF

**Remarks** This statement is similar to DBBREAK<*nexp*>|<*sexp*>OFF but deletes all breakpoints instead of just one breakpoint at the time.  
Related instructions are DBBREAK, DBEND, DBSTDIO, and DBSTEP.

## **DBEND**

**Purpose** Statement for terminating the Fingerprint Debugger.

**Syntax** DBEND

**Remarks** This statement is used for terminating the Fingerprint Debugger prematurely and restore the STDIO settings as they were before the Debugger was started.

Related instructions are DBBREAK, DBBREAK OFF, DBSTDIO, and DBSTEP.

## **DBSTDIO**

**Purpose** Statement for selecting the standard IN/OUT channel for the Fingerprint Debugger.

**Syntax** DBSTDIO <*nexp1*>, <*nexp2*> [, <*sexp1*>, <*sexp2*>]  
DBSTDIO [<*nexp1*>, <*nexp2*>, ] <*sexp1*>, <*sexp2*>

where:

<*nexp1*> is the desired Debug STDIN channel:

0 = “console:”  
1 = “uart1:” (default)  
2 = “uart2:”  
3 = “uart3:”  
4 = “centronics:”  
5 = “net1:”  
6 = “usb1:”  
7 = “uart4:”  
8 = “uart5:”

<*nexp2*> is the desired Debug STDOUT channel:

0 = “console:”  
1 = “uart1:” (default)  
2 = “uart2:”  
3 = “uart3:”  
5 = “net1:”  
6 = “usb1:”  
7 = “uart4:”  
8 = “uart5:”

<*sexp1*> Preamble (default: empty string)

<*sexp2*> Postamble (default: empty string)

**Remarks** The maximum size of the preamble and postamble strings is 12 characters. Related instructions are CONT, DBBREAK, DBBREAK OFF, DBEND, DBSTEP, and STOP.

**Example** This statement selects “uart2:” as Debug STDIO channel. Preamble is specified as “in” and postamble as “out”:

```
DBSTDIO 2,2,"in","out"
```

## DBSTEP

**Purpose** Statement for specifying the interval between breaks for the Fingerprint Debugger and execute the program accordingly.

**Syntax** DBSTEP<ncon>

where <ncon> is the number of lines to be executed before break. Default is 1 line.

**Remarks** If <ncon> is omitted, one line is executed, but if <ncon> = 0, nothing happens.

DBSTEP cannot be used in execution mode, which results in error 78. When DBSTEP is used on the last line in a program, the line is executed but there is no break.

If DBSTEP is used in a program with a FOR or WHILE loop, there is only one break on the line that calls for the FOR or WHILE loop (the first time it is executed).

Related instructions are CONT, DBBREAK, DBBREAK OFF, DBEND, and DBSTDIO.

**Example**

```
10 PRINT "11"
20 PRINT "22"
30 PRINT "33"
40 PRINT "44"
50 PRINT "55"
60 PRINT "66"
70 PRINT "77"
80 PRINT "88"
90 PRINT "99"
DBSTEP 4
11
22
33
44
Break in line 50
Ok
DBSTEP
55
Break in line 60
Ok
DBSTEP 2
66
77
Break in line 80
```

CONT  
88  
99  
Ok

## **DELETE**

**Purpose** Statement deleting one or several consecutive program lines from the printer's working memory.

**Syntax** DELETE<ncon1> [-<ncon2>]

where:

<ncon1> is the line, or the first line in a range of lines, to be deleted.

<ncon2> (optional) is the last line in a range of program lines to be deleted.

**Remarks** This statement can only be used for editing the current program in the Immediate Mode and cannot be included as a part of the program execution.

**Examples** DELETE 50 deletes line 50 from the program.

DELETE 50-100 deletes line 50 thru 100 from the program.

DELETE 50- deletes all lines from line 50 to the end of the program.

DELETE -50 deletes all lines from the start of the program to line 50.

## **DELETEPFSVAR**

**Purpose** Statement for deleting variables saved at power failure.

**Syntax** DELETEPFSVAR<sexp>

where <sexp> is the name of the variable to be deleted.

**Remarks** Related instructions are SETPFSVAR, GETPFSVAR, and LISTPFSVAR.

**Examples** DELETEPFSVAR "QCPS%"  
DELETEPFSVAR "QS\$"

## **DEVICES**

**Purpose** Statement for returning the names of all devices on the standard OUT channel.

**Syntax** DEVICES

**Remarks** All devices available to the user in Intermec Fingerprint firmware will be listed whether they are installed or not. The only exception is that "centronics:" will be listed for the EasyCoder PD41 only if installed.

There are also a number of devices for internal use only. The list below indicates if and how the device can be OPENed (see OPEN statement). If you try to OPEN a disconnected or uninstalled device, the message “Error in file name” is printed to the standard OUT channel (see SETSTDIO). Note that all names of devices are lowercase and most are appended by a colon (:).

| Device           | Explanation                       | Can be opened for   |
|------------------|-----------------------------------|---------------------|
| c: (= “/c/”)     | Printer’s permanent memory        | Input/Output/Random |
| card1:           | CompactFlash memory card          | Input/Output/Random |
| centronics:      | Parallel communication port       | Input               |
| console:         | Printer’s display and/or keyboard | Input/Output        |
| dll:             | Special applications only         | —                   |
| finisher:        | The finisher interface            | Input/Output        |
| lock:            | Electronic keys                   | Input               |
| net1:            | EasyLAN                           | Input/Output        |
| par:             | Special applications only         | —                   |
| rom: (= “/rom/”) | Kernel and Read-only memory card  | Input               |
| rs485:           | RS-485 communication              | Input/Output        |
| storage:         | Electronic keys                   | Input/Output/Random |
| tmp:             | Printer’s temporary memory        | Input/Output/Random |
| uart1:           | Serial communication port         | Input/Output        |
| uart2:           | Serial communication port         | Input/Output        |
| uart3:           | Serial communication port         | Input/Output        |
| uart4:           | Serial communication port         | Input/Output        |
| uart5:           | Serial communication port         | Input/Output        |
| usb1:            | Serial communication port         | Input/Output        |
| wand:            | Data from Code 128 bar code       | Input               |

- c: is the printer’s permanent read/write memory (Flash SIMMs). It supports file systems with directories, and retains its content when the power is switched off. For compatibility with programs created in previous versions of Intermec Fingerprint, the term “ram:” will also be accepted.
- card1: is a read/write DOS-formatted CompactFlash memory card installed in the printer.
- centronics: is the Centronics parallel port. Three different types can be selected by means of SYSVAR(25).
- console: is the printer’s display and keyboard. The keyboard can be used for input only and the display for output only.
- dll: is used for special applications only.
- finisher: is the device controlling the finisher interface, where a device such as a cutter can be connected.

- lock: is an electronic key item that has been specified as locked by means of special software. An electronic key may contain several key items with different properties (counter, lock, or storage). The device name calls all key items with the corresponding properties. Each key item has a 4-character name, usually appended by a delimiter (“?”) and a 4-character password.
- net1: is the communication channel for an EasyLAN interface board.
- par: is used for special applications only.
- rom: is both the read-only kernel sectors in the Boot-Bank flash SIMM, and any resource files on a CompactFlash memory card installed in the printer. It supports file systems with directories.
- rs485: is used in connection with RS-485 point-to-point or multidrop communication to specify the RS-485 protocol and the protocol address of the unit (for example, “rs485:23”).
- storage: is all electronic key items in the printer that has been specified as storage by means of special software. Note that this memory is comparatively slow.
- tmp: is the printer’s temporary read/write memory (SDRAM SIMMs). It will lose its content when the power is turned off or at a power failure. Valuable data that cannot be recreated should be copied to “/c/”. One advantage of using “tmp:” instead of “/c” is that data can be written to SDRAM faster than to the flash memory. To speed up operation, the Intermec Fingerprint firmware (except program modules with dynamic downloading) is copied from “/rom/” to “tmp:” at startup and used from “tmp:”.
- uart1: is the standard RS-232 port.
- uart2: is an additional serial port on an optional interface board.
- uart3: is an additional serial port on an optional interface board.
- uart4: is an additional serial port on an optional interface board.
- uart5: is an additional serial port on an optional interface board.
- usb1: is the standard USB (Universal Serial Bus) port.
- wand: is any input from an Code 128 bar code not containing any FNC3 character via a bar code wand or reader connected to the wand interface.

**Example** DEVICES

results in:

```
c:
card1:
centronics: (only if an optional parallel interface board is fitted)
console:
dll:
finisher:
lock:
net1: (only if an EasyLAN interface board is fitted)
par:
rom:
rs485: (only if an optional serial interface board is fitted)
storage:
tmp:
uart1:
uart2: (only if an optional serial interface board is fitted)
uart3: (only if an optional serial interface board is fitted)
uart4: (only if two optional serial interface boards are fitted)
uart5: (only if two optional serial interface boards are fitted)
usb1:
wand:
```

## DIM

**Purpose** Statement specifying the dimensions of an array.

**Syntax** DIM<<ivar>>|<svar>>(<nexp1>[,<nexp2>...]) . . .  
[,<<ivar>>|<svar>>(<nexp1>[,<nexp2>...])]

where:

<ivar>|<svar> is the name of the array.

<nexp1> is the maximum subscript value for the first dimension.

<nexp2-10> (optional) are the maximum subscript values for the following dimensions (2 through 10).

**Remarks** An array is created by entering a variable followed by a number of subscripts (maximum of 10) separated by commas. All subscripts are enclosed by parentheses and each subscript represents a dimension. The number of subscripts in an array variable, the first time (regardless of line number) it is referred to, determines its number of dimensions. By default, the number of elements in each dimension is restricted to four (numbered 0 to 3).

Note that 0 = 1st element, 1 = 2nd element, and so on. If more than 4 elements in any dimension are desired, a DIM statement must be issued. For example, ARRAY\$(1,2,3) creates a three-dimensional array, where the dimensions each contain 4 elements (0-3) respectively. This corresponds to the statement DIM ARRAY\$(3,3,3).

It is not possible to change the number of dimensions of an array that already has been created during runtime, and error 57 (“Subscript out of range”) occurs in this case.

Considering the limited amount of printer memory and other practical reasons, be careful not to make the arrays larger than necessary. A DIM statement can be used to limit the amount of memory set aside for the array.

- Example 1** This example creates an array containing three dimensions with 13 elements each:

```
100 DIM NAME$(12,12,12)
```

- Example 2** In this example, two one-dimensional arrays are created on the same program line:

```
10 DIM PRODUCT$(15), PRICE%(12)
20 PRODUCT$(2) = "PRINTER"
30 PRICE%(2) = 1995
40 PRINT PRODUCT$(2); " $" ; PRICE%(2)
RUN
```

results in:

```
PRINTER $1995
```

## **DIR**

**Purpose** Statement specifying the print direction.

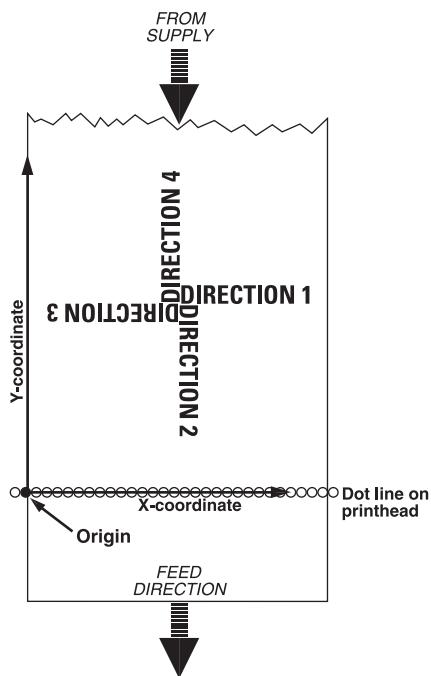
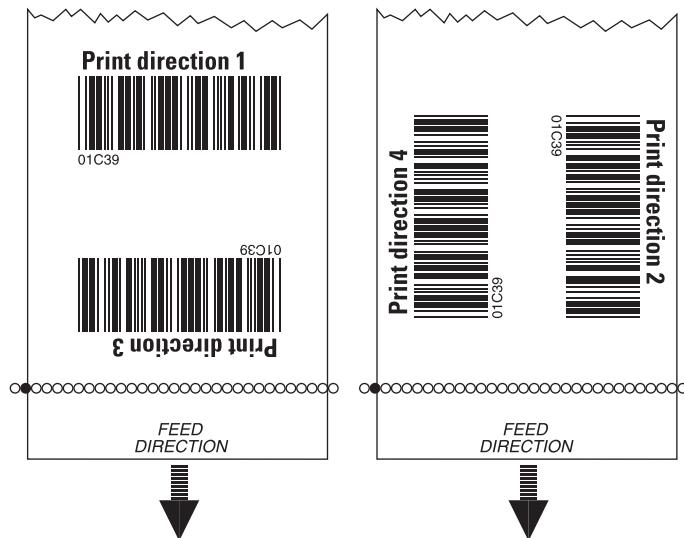
**Syntax** DIR<nexp>

where <nexp> is the print direction (1, 2, 3, or 4). Default is 1.

Reset to default by executing a PRINTFEED.

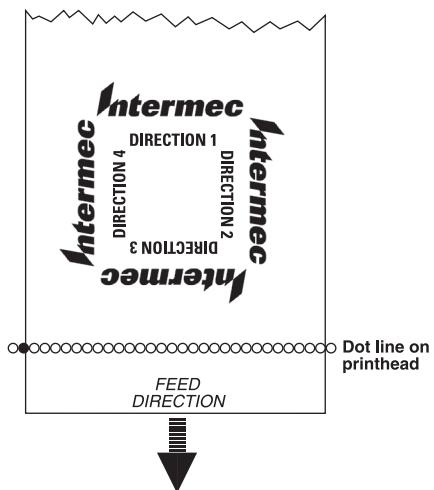
**Remarks** A change of print direction affects all printing statements (PRTXT, PRBAR, PRIMAGE, PRBOX, or PRLINE) that are executed later in the program until a new DIR statement or a PRINTFEED statement is executed.

The print direction is specified relative to the media feed direction as shown in the next illustrations:

**Print Direction for Text Fields****Print Direction for Bar Code Fields**

Horizontal "picket fence" printing vs. vertical "ladder" printing.

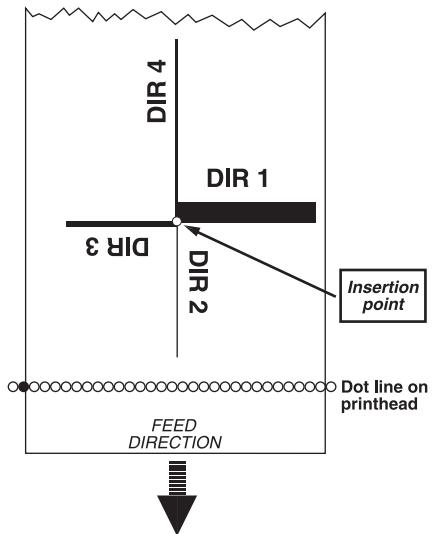
**Print Direction for Images**



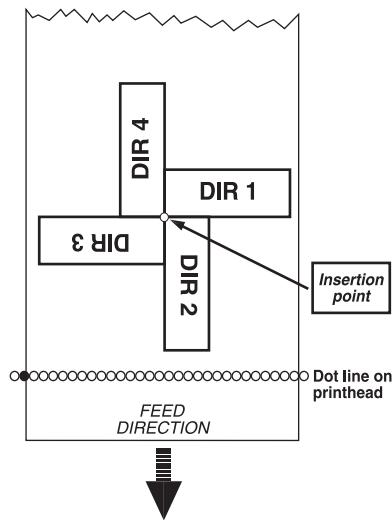
The relation of the image and the print direction depends how the image was drawn. An image can only be “rotated” 180°. Thus, it may be useful to have two copies of the image available with different extensions for either horizontal or vertical printing:

- For DIR 1 & 3, use extension .1.
- For DIR 2 & 4, use extension .2.

**Print Direction for Lines**



### Print Direction for Boxes



**Example 1** This example prints a label with one line of text and draws a line beneath the text:

```

10 PRPOS 30,300
20 DIR 1
30 ALIGN 4
40 FONT "Swiss 721 BT",18
50 PRTXT "TEXT PRINTING"
60 PRPOS 30,280
70 PRLINE 555,10
80 PRINTFEED
RUN

```

**Example 2** This example is similar to Example 1 but prints the same information vertically, requiring new positioning to avoid Error 1003, “Field out of label.”

```

10 PRPOS 300,30 (new position)
20 DIR 4 (new direction)
30 ALIGN 4
40 FONT "Swiss 721 BT",18
50 PRTXT "TEXT PRINTING"
60 PRPOS 320,30 (new position)
70 PRLINE 555,10
80 PRINTFEED
RUN

```

## DIRNAME\$

**Purpose** Returns the names of the directories stored in the specified part of the printer’s memory.

**Syntax** DIRNAME\$ [ (<sexp>) ]

where <sexp> is the name of the memory device from which the first directory name will be listed.

**Remarks** In *<sexp>*, parts of directory names and wildcards (\*) are allowed. If *<sexp>* is omitted, the next directory name in the same memory device is listed.

DIRNAME\$ can be repeated. When there are no directories left to list, the output string is empty. Also see “**FILENAME\$**” on page 64.

**Example**

```
FILES,A
Files on /c
. / 0 .. / 0
.setup.saved 239 DIR1/
STDIO 3
4124672 bytes free 242 bytes used
PRINT DIRNAME$("/c/")
.Ok
PRINT DIRNAME$
.Ok
PRINT DIRNAME$
DIR1
Ok
```

## END

**Purpose** Statement ending the execution of the current program or subroutine and closing all OPENed files and devices.

**Syntax** END

**Remarks** END can be placed anywhere in a program, but is usually placed at the end. It is also useful for separating the “main” program from possible subroutines with higher line numbers. It is possible to issue several END statements in the same program.

**Example** A part of a program which produces fixed line-spacing might look like this:

```
10 FONT"Swiss 721 BT"
20 X%=300:Y%=350
30 INPUT A$
40 PRPOS X%,Y%
50 PRTXT A$
60 Y%=Y%-50
70 IF Y%>=50 GOTO 30
80 PRINTFEED
90 END
```

The Y-coordinate is decremented by 50 dots for each new line until it reaches the value 50. The END statement terminates the program.

## EOF

**Purpose** Function for checking for an end-of-file condition.

**Syntax** EOF (*<nexp>*)

where *<nexp>* is the number assigned to the file when it was OPENed.

**Remarks** The EOF function can be used with files OPENed for sequential input (in connection with the statements INPUT#, LINE INPUT#, and INPUT\$) to avoid the error condition “Input past end” which has no error message.

When the EOF function encounters the end of a file, it returns the value -1 (true). If not, it returns the value 0 (false).

**Example**

```
10 DIM A%(10)
20 OPEN "DATA" FOR OUTPUT AS #1
30 FOR I%=1 TO 10
40 PRINT #1, I%*1123
50 NEXT I%
60 CLOSE #1
70 OPEN "DATA" FOR INPUT AS #2
80 I%=0
90 WHILE NOT EOF(2)
100 INPUT #2, A%(I%):PRINT A%(I%)
110 I%=I%+1:WEND
120 IF EOF(2) THEN PRINT "End of File"
RUN
```

results in:

```
1123
2246
3369
4492
5615
6738
7861
8984
10107
11230
End of File
```

## ERL

**Purpose** Returns the number of the line on which an error condition has occurred.

**Syntax** ERL

**Remarks** Also useful in connection with an ON ERROR GOTO statement.

**Example 1** You can check at which line the last error since power up occurred like this:

```
PRINT ERL
```

resulting in:

```
40
```

**Example 2** In this example, the line number of the line where an error has occurred determines the action to be taken. In this case the font size is too large for the label width:

```
10 ON ERROR GOTO 1000
20 FONT "Swiss 721 BT",100
30 PRTXT "HELLO EVERYBODY"
40 PRINTFEED
50 END
```

```
1000 IF ERL=40 THEN PRINT "PRINT ERROR"
1010 RESUME NEXT
RUN
```

resulting in:

```
PRINT ERROR
```

- Example 3** You can use the ERL function in programs without line numbers also, since these programs automatically generate hidden line numbers that are revealed when the program is LISTed. This is the same program as above but without line numbers:

```
NEW
IMMEDIATE OFF
ON ERROR GOTO QAAA
FONT "Swiss 721 BT",100
PRTXT "HELLO EVERYBODY"
PRINTFEED
END
QAAA: IF ERL=40 THEN PRINT "PRINT ERROR"
RESUME NEXT
IMMEDIATE ON
RUN
```

results in:

```
PRINT ERROR
```

## ERR

**Purpose** Returns the code number of an error.

**Syntax** ERR

**Remarks** The firmware can detect a number of error conditions which are represented by code numbers as described in Chapter 7, “Error Codes.” The ERR function enables the program to read the coded error number. You can design your program to take proper action depending on which type of error has occurred.

**Examples** In this example, the error code determines the action to be taken:

```
10 ON ERROR GOTO 1000
.
.
100 PRTXT "HELLO"
110 PRINTFEED
120 END
.
.
1000 IF ERR=1005 THEN PRINT "OUT OF PAPER"
1010 RESUME NEXT
```

You can also check the number of the last error since power up:

```
PRINT ERR
```

resulting in:

```
1022
```

## ERR\$

**Purpose** Returning the explanation of an error code in plain text.

**Syntax** `ERR$(<nexp>)`

where `<nexp>` is the error code number

**Remarks** The explanation of the error is returned in English. For more information, see Chapter 7, “Error Codes.”

**Example** `PRINT ERR$(1003)`

results in:

Field out of label

## ERROR

**Purpose** Statement for defining error messages and enabling error handling for specified error conditions (Intermec Direct Protocol only).

**Syntax** `ERROR <nexp> [, <sexp>]`

where:

`<nexp>` is the number of the error condition.

`<sexp>` is the desired error message.

**Remarks** The `ERROR` statement can only be used in the Intermec Direct Protocol for the purpose of enabling error-handling and creating customized error messages, as described below.

The built-in error-handler of the Intermec Direct Protocol will always handle these standard errors:

| Number | Error condition              | Message displayed on printer |
|--------|------------------------------|------------------------------|
| 15     | Font not found               | Font not found               |
| 18     | Disk full                    | Disk full                    |
| 37     | Cutter device not found      | Cutter device not found      |
| 43     | Memory overflow              | Memory overflow              |
| 1003   | Field out of label           | Field out of label           |
| 1005   | Out of paper                 | Out of paper                 |
| 1006   | No field to print            | No field[s]                  |
| 1022   | Head lifted                  | Head lifted                  |
| 1027   | Out of transfer ribbon       | Out of ribbon                |
| 1031   | Next label not found         | Label not found              |
| 1058   | Transfer ribbon is installed | Ribbon installed             |
| 1606   | Testfeed not done            | Testfeed not done            |

Other errors will not be handled unless they have been specified by an ERROR statement. For more information on error numbers, see Chapter 7, “Error Codes.”

The ERROR statement also allows you to edit a suitable message in any language. This message appears in the printer display if the error occurs. The error message is truncated to 33 characters. Characters 1 through 16 appear on the upper line and characters 18 through 33 appear on the lower line. Character 17 is always ignored.

ANSI control characters can be used in the error message string. For more information, see “Printer Function Control; Display” in the *Intermec Fingerprint Tutorial*.

An empty string removes any previously entered message for the error in question. Likewise, an existing message can be replaced by a new one. When a standard error or an error defined by an ERROR statement is detected, the printer sets its standard IN port to BUSY, sets the Status LED to red, and displays the error messages. When you press the printer’s Print key, the error message is cleared, the LED is set to green, and the standard IN port is set to READY. In some cases, the error must also be cleared (for example, by loading more labels).



**Note:** For printers without the Intermec Ready-to-Work Indicator, the green Ready LED is switched off and the red Error LED is switched on when an error is detected. When the error is cleared, the Error LED is switched off and the Ready LED is turned on.

Error messages are not saved in the printer memory, but new ERROR statements have to be downloaded after each power up. Intermec recommends that you save a set of ERROR statements as a file in the host computer.

The ERROR statements affect both the error messages in the printer display and the error messages returned to the host via the standard OUT channel (see SETSTDIO statement).

By default, no error messages are returned to the host in the Intermec Direct Protocol, since the statement INPUT ON sets the verbosity level to off (SYSVAR (18)= 0). However, the verbosity level can be changed by VERBON/VERBOFF statements or the SYSVAR (18) system variable.

Different types of error messages to return on the standard OUT channel can be selected by the SYSVAR (19) system variable. If SYSVAR (19) is set to 2 or 3, the error message specified by ERROR is transmitted. If no such error message is available, a standard error message in English will be transmitted. For more information, see Chapter 7, “Error Codes.”

**Example** In these examples, a few errors are specified. Note the blank spaces for character position 17 in each message (space characters are indicated by doubleheaded arrows):

```
ERROR 1010, "HARDWARE.....ERROR"↔.
ERROR 1029, "PRINTHEAD.VOLT-..AGE.TOO.HIGH"↔.
```

## EXECUTE

**Purpose** Statement for executing a Fingerprint program line or a file with Fingerprint program lines from within another Fingerprint program.

**Syntax** EXECUTE<*sexp*>

where <*sexp*> is one line of Fingerprint instructions or the name of a file containing at least one line of a Fingerprint program.

**Remarks** This statement allows you to create a library of layouts, subroutines, texts, or other functions which can be executed as a part of a program without having to merge the programs.

- The program called by EXECUTE must not contain any line numbers or line labels.
- If the EXECUTE statement is followed by a string of Fingerprint instructions, they should be separated by colons.

When an error occurs in an EXECUTE file, the line number in the error message is that of the EXECUTE file, not of the program where the EXECUTE statement is issued.

- EXECUTE is only allowed in the execute mode, not in the immediate mode (yields error 69).
- Recursive call of EXECUTE is not allowed (yields error 78).

**Example** This example shows how a preprogrammed file containing a bar code is executed as a part of a Fingerprint program, where the input data and printfeed are added:

```
IMMEDIATE OFF
DIR 1
ALIGN 7
BARSET "CODE39",2,1,3,120
BARFONT "Swiss 721 BT",10,8,5,1,1
BARFONT ON
IMMEDIATE ON
SAVE "tmp:BARCODE.PRG",L
NEW
10 PRPOS 30,400
20 EXECUTE "tmp:BARCODE.PRG"
30 PRBAR "ABC"
40 PRINTFEED
RUN
```

## **FIELD**

**Purpose** Statement for creating a single-record buffer for a random file and dividing the buffer into fields to which string variables are assigned.

**Syntax** FIELD [#] <nexp1>, <nexp2>AS<svar1> [, <nexp3>AS<svar2>...]

where:

# (optional) indicates that whatever follows is a number.

<nexp1> is the number assigned to the file when it was OPENed.

<nexp2-n> is the number of bytes to be reserved for the string variable that follows (Null not allowed).

<svar1-n> is the designation of the string variable, for which space has been reserved.

**Remarks** The buffer is divided into fields, each of which is given an individual length in bytes. A string variable is assigned to each field. This statement does not put any data in the buffer, but only creates and formats the buffer, allowing you to place the data using LSET and RSET statements.

Before using this statement, consider the maximum number of characters (including space characters) needed for each variable. Make sure the total does not exceed the record size given when the file was OPENed (default is 128 bytes). When a file is CLOSEd, all its FIELD definitions are lost.

**Example** This example opens and formats a file buffer for a single record. The buffer is divided into three fields, with the size of 25, 30, and 20 bytes respectively.

```
10 OPEN "ADDRESSES" AS #8 LEN=75
20 FIELD#8,25 AS F1$, 30 AS F2$, 20 AS F3$
```

Imagine a spreadsheet matrix where the file is the complete spreadsheet, the records are the lines and the fields are the columns. The buffer can only contain one such line at the time.

## **FIELDNO**

**Purpose** Gets the current field number for partial clearing of the print buffer by a CLL statement.

**Syntax** FIELDNO

**Remarks** By assigning the FIELDNO function to one or several numeric variables, you can divide the print buffer into portions, which can be cleared using a CLL statement.

**Example**

```
10 PRPOS 100,300
20 FONT "Swiss 721 BT"
30 PRTXT "HAPPY"
40 A%=FIELDNO
50 PRPOS 100,250
60 PRTXT "NEW YEAR"
70 B%=FIELDNO
```

```

80 PRPOS 100, 200
90 PRTXT "EVERYBODY! "
100 PRINTFEED
110 CLL B%
120 PRPOS 100,200
130 PRTXT "TO YOU! "
140 PRINTFEED
150 CLL A%
160 PRPOS 100,250
170 PRTXT "BIRTHDAY"
180 PRPOS 100,200
190 PRTXT "DEAR TOM! "
200 PRINTFEED
RUN

```

yields three labels:

| <b>Label 1</b> | <b>Label 2</b> | <b>Label 3</b> |
|----------------|----------------|----------------|
| HAPPY          | HAPPY          | HAPPY          |
| NEW YEAR       | NEW YEAR       | BIRTHDAY       |
| EVERYBODY!     | TO YOU!        | DEAR TOM!      |

## FILE& LOAD

**Purpose** Statement for receiving and storing binary files in the printer's memory.

**Syntax** FILE& LOAD [*<nexp1>*, ] *<sexp>*, *<nexp2>* [, *<nexp3>*]

where:

*<nexp1>* (optional) is the number of bytes to skip before starting to read the file data.

*<sexp>* is the name for the file when stored in the printer's memory. Maximum length is 30 characters including any extension.

*<nexp2>* is the size of the file in bytes.

*<nexp3>* optionally specifies a communication channel OPENed for INPUT by the number assigned to the device. Default is Std IN channel.

**Remarks** This statement prepares the printer to receive a binary file on the standard IN channel (see SETSTDIO statement) or on another communication channel OPENed for INPUT. Image files and font files can also be downloaded using the IMAGE LOAD statement.

Unlike the IMAGE LOAD statement, FILE& LOAD does not immediately install the fonts, but the font files remain in the printer's memory until next power-up. The optional first parameter makes it possible to use this statement in MSDOS (CR/LF problem).

The name of the file, when stored in the printer memory, may consist of up to 30 characters, including possible extension. The size of the original file should be given in bytes according to its size in the host.

Before the FILE& LOAD statement can be used on a serial channel, the setup must be changed to 8 characters, RTS/CTS handshake. When a FILE& LOAD statement is executed, the execution stops and waits for the number of bytes specified in the statement to be received. During the transfer of file data to the printer, there is a 25 second timeout between characters. If a new character is not received within the timeout, an error 80 (“Download timeout”) occurs. When the specified number of characters is received, the execution resumes.

**Example**    10 OPEN "uart2:" FOR INPUT AS 5  
20 FILE& LOAD "FILE1.PRG", 65692, 5  
30 CLOSE 5

## **FILENAME\$**

**Purpose**    Function returning the names of the files stored in the specified part of the printer’s memory.

**Syntax**    FILENAME\$ [ (<sexp>) ]

where <sexp> is the name of the memory device from which the first file name (in alphabetical order) will be listed. Parts of file names and wildcards (\*) are allowed. Maximum size is 30 characters.

If <sexp> is omitted, the next file name in the same device is listed. This function can be repeated. When there are no files left to list, the output string is empty.

**Remarks**    The specified memory device must be mounted. The file name is case-sensitive and must correspond to the name of the file stored in the memory device. Wildcards (\* = ASCII 42 dec.) can be used. The list may include all types of files. Even system file names preceded by a period character (for example, .FONTALIAS) may be listed.

No directories are listed and the order of listing is not specified. To list directories, see “[DIRNAME\\$](#)” on page 55.

**Example**    This example shows how all files in the printer permanent memory (/c) are listed:

```
FILES, A
Files on /c
./ 0 .. / 0
.setup.saved 239 DIR1/ 0
STDIO 3
4124672 bytes free 242 bytes used
PRINT FILENAME$("/c/")
.setup.saved
Ok
PRINT FILENAME$
STDIO
PRINT FILENAME$
Ok
```

## FILES

**Purpose** Statement for listing the files stored in one of the printer's directories to the standard OUT channel.

**Syntax** FILES [<sexp>] [, R] [, A]

where:

<sexp> (optional) specifies the directory. For more information, see “[DEVICES](#)” on page 48.

R lists directories recursively.

A lists all files including system files (files with a name starting with a period (.) character).

**Remarks** If no directory is specified, files in the printer's current directory are listed. By default, the current directory is the printer's permanent memory (“/c”). To change the current directory, see “[CHDIR](#)” on page 23.

By including a reference to a memory device (“/c”, “tmp:”, “/rom”, “card1:”, “lock:”, or “storage:”), the files of the specified directory are returned without changing the current directory.

If the “A” flag is omitted, all files, except system files, will be listed. The flags A and R can be entered in any order, but R is always processed first. The number of bytes for each file and the total number of free and used bytes in the specified directory will also be included in the list.

**Examples** The presentation may look like this on the host screen:

```
FILES "/c",R
Files on /c
STDIO 2 FILE2 4
DIR1/ 0
Files on /c/DIR1/
FILE1 4 DIR2/ 0
STDIO 2
No files on /c/DIR1/DIR2
4121600 bytes free 12 bytes used
FILES,R,A
Files on /c
./ 0 ../ 0
DIR1/ 0 FILE2 4
STDIO 2 .setup.saved 239
Files on /c/DIR1/
./ 0 ../ 0
DIR2/ 0 STDIO 2
FILE1 4
Files on /c/DIR1/DIR2/
./ 0 ../ 0
4121600 bytes free 251 bytes used
FILES "/c/DIR1"
Files on /c/DIR1
FILE1 4 DIR2/ 0
STDIO 2
4121600 bytes free 6 bytes used
```

## FLOATCALC\$

**Purpose** Function for calculation with float numbers.

**Syntax** FLOATCALC\$ (<sexp1>, <sexp2>, <sexp3> [, <nexp1>] )

where:

<sexp1> is the first operand.

<sexp2> is the operator (+, -, \*, or /).

<sexp3> is the second operand.

<nexp1> (optional) is the precision in decimals (default 10).

**Remarks** Operands are float numbers, that is, a string of digits with a decimal point to separate decimals from integers. Operands can also contain leading plus (+), minus (-), and space characters. Space characters are ignored.

The usual mathematical rules apply to plus and minus signs. All other characters, or plus, minus, and space characters in positions other than leading, generate errors.

Note the mathematical rules:

- - yields +

- + yields -

+ - yields -

+ + yields +

The following arithmetic operators are allowed:

+ (addition) ASCII 043 dec

- (subtraction) ASCII 045 dec

\* (multiplication) ASCII 042 dec

/ (division) ASCII 047 dec

Other operators or characters generate an error.

The optional precision parameter specifies the number of decimals in the result of the calculation. The result is truncated accordingly. For example, if the number of decimals is specified as 5, the result 5.76123999 is presented as 5.76123.

The result of a FLOATCALC\$ function can be formatted using a FORMAT\$ function.

**Example 1** Addition:

```
A$ = "234.9"
B$ = "1001"
PRINT FLOATCALC$ (A$, "+" ,B$,5)
```

results in:

1235.90000

**Example 2** Subtraction:

```
A$ = "234.9"
C% = 2
PRINT FLOATCALC$ (A$, "-", 100.013, C%)
```

results in:

134.88

## FONT(FT)

**Purpose** Statement for selecting a scalable TrueType or TrueDoc single-byte font, or a single-byte bitmap font, for printing the subsequent PRTXT statements.

**Syntax** FONT | FT<sexp1> [, <nexp1> [, <nexp2> [, <nexp3>] ] ]

where:

<sexp1> is the name of the font. Default is Swiss 721 BT.

<nexp1> (optional) is the height in points of the font. Default is 12. Use MAG to enlarge with bitmap fonts.

<nexp2> is the clockwise slant in degrees. Range is 0 to 90. Default is 0. Does not work with bitmap fonts.

<nexp3> is the width enlargement in percent relative to the height. Range is 1 to 1000. Default is 100. Does not work with bitmap fonts.

Reset to default by executing a PRINTFEED.

**Remarks** Intermec Fingerprint supports scalable fonts in TrueType and TrueDoc format that comply with the Unicode standard. A large number of scalable fonts are available on special request, so it is possible that your printer is fitted with a unique selection of fonts. Use a FONTS statement to list the names of all fonts installed in your own printer to the standard OUT channel.

To maintain compatibility with programs created in earlier versions of Intermec Fingerprint, you can also specify bitmap font names, such as “SW030RSN” or “MS060BMN.2”. For a standard bitmap font name, the firmware selects the corresponding scalable font in the printer’s memory and sets font parameters so its direction, appearance, and size come as close to the specified bitmap font as possible. The standard complement of outline fonts must already be in the printer memory.

Nonstandard bitmap fonts can also be used. They will retain the bitmap format but will not produce any outline fonts. Any extension to the bitmap font name is ignored. For more information, see Chapter 4, “Character Sets and Fonts.”

The height of the font, including ascenders and descenders, is given in points (1 point = 1/72 inch, approximately 0.35 mm), which means that text is printed in the same size regardless of the printhead density of the printer. Sizes less than 4 points will be unreadable.

In case of bitmap fonts, use MAG to enlarge the font instead of specifying a font height. Any font may be magnified up to 4 times separately in regard of height and width using a MAG statement. Bitmap fonts will get somewhat jagged edges when magnified, whereas outline fonts will remain smooth. For more information, see “[MAG](#)” on page 126.

Slanting means that you can create the same effect as in ITALIC characters. Higher values increase the amount of clockwise slant, and values greater than 65-70° will be unreadable. Slanting cannot be used with bitmap fonts.

**10% SLANT**

**10% Slant Example**

**20% SLANT**

**20% Slant Example**

A scalable font can enlarged in terms of width relative to height. The value is given as a percentage (range is 1 to 1000). This means that if the value is 100, there is no change in the appearance of the characters. A value of 50 reduces the font by half, and a value of 200 doubles the size. When using this parameter, all parameters in the syntax (name, height, slant, and width) must be included in the statement.

## **Adding More Fonts**

The standard complement of fonts listed in Chapter 4 can be supplemented with more fonts using three methods:

- Downloading fonts from a Font Install Card

The card must be inserted before the printer is started. At startup the fonts are automatically downloaded, installed, and permanently stored in the printer memory. The fonts can be used without the card.

- Using fonts from a Font Card

The card must be inserted before the printer is started. At startup the fonts are automatically installed, but not copied to the printer memory. Thus, the card must always be installed to use these fonts.

- Downloading font files

Font files can be downloaded and installed by means of either of the two statements IMAGE LOAD and TRANSFER KERMIT. There is no need to restart the printer before using these fonts.

You can create font aliases to produce shorter or more easily recognized names. For more information, see “[Font Aliases](#)” on page 275.

**Example 1** This example prints one line of 12p text with the default direction and alignment:

```
10 FONT "Swiss 721 BT"
20 PRTXT "HELLO"
30 PRINTFEED
RUN
```

**Example 2** This example prints the same text but at 24p, with 20° slant, and with 75% width:

```
10 FONT "Swiss 721 BT", 24, 20, 75
20 PRTXT "HELLO"
30 PRINTFEED
RUN
```

## FONTD

**Purpose** Statement for selecting a scalable TrueType or TrueDoc double-byte font for the printing of the subsequent PRTXT statements.

**Syntax** FONTD<sexp1>[,<nexp1>[,<nexp2>[,<nexp3>]]]

where:

<sexp1> is the name of the font. Default is none.

<nexp1> (optional) is the height of the font in points. Default is 12.

<nexp2> is the clockwise slant in degrees. Range is 0 (default) to 90.

<nexp3> is the width enlargement in percent relative to the height. Range is 1 to 1000. Default is 100.

Reset to default by executing a PRINTFEED or CLL.

**Remarks** This statement is identical to the FONT statement, but is used for fonts specified by a double byte (16 bits) instead of a single byte (7 or 8 bits). To use a double-byte font, a double-byte character set must be selected using a NASCD statement. Usually, if the first byte has an ASCII value between 161 dec. (A1 hex) and 254 dec (FE hex), the character is treated as a double-byte character and the firmware waits for next byte to make the 16 bit address complete. The character is printed using the font specified by FONTD and according to the character set specified by NASCD and the Unicode standard.

On the other hand, if the first byte has an ASCII value below 161 dec. (A1 hex), the character is treated as a single byte character, and the next byte received is regarded as the start of a new character. This implies that the character set specified by NASC and the font specified by FONT is used. However, the selected Unicode double-byte character set may specify some other ASCII value as the breaking point between single and double byte character sets.

Note that 8 bit communication must be selected.

Only writing from left to right in the selected print direction is supported.

**Example** In this example, the text contains both single- and double-byte fonts. The double-byte font and its character set are stored in a Font Install Card:

```
10 NASC 46
20 FONT "Swiss 721 BT", 24, 10
30 FONTD "DLC Ming Medium"
40 NASCD "rom:BIG5.NCD"
50 PRTXT CHR$(65);CHR$(161);CHR$(162)
60 PRINTFEED
RUN
```

This program yields a printed text line that starts with the Latin character A (ASCII 65 dec.) followed by the character in the DLC Ming Medium font that corresponds to the address 161+162 dec. in the character set “BIG5.NCD”.

## **FONTNAME\$**

**Purpose** Returns the names of the fonts stored in the printer's memory.

**Syntax** FONTNAME\$ (<nexp>)

where <nexp> is the result of the expression. False (0) indicates first font. True (\_,0) indicates next font.

**Remarks** FONTNAME\$ (0) produces the first name in the memory.

FONTNAME\$ (\_,0) produces next name. Can be repeated as long as there are any fontnames left.

**Example** This example lists all font names:

```
10 A$ = FONTNAME$ (0)
20 IF A$ = "" THEN END
30 PRINT A$
40 A$ = FONTNAME$ (-1)
50 GOTO 20
RUN
```

results in:

```
-UPC11.1
-UPC11.2
-UPC21.1
-UPC21.2
-UPC31.1
-UPC31.2
-UPC51.1
-UPC51.2
Century Schoolbook BT
DingDings SWA
Dutch 801 Bold BT
Dutch 801 Roman BT
Futura Light BT
Letter Gothic 12 Pitch BT
MS030RMN
MS030RMN.1
MS030RMN.2
MS050RMN
```

```

MS050RMN.1
MS050RMN.2
MS060BMN
MS060BMN.1
MS060BMN.2
Monospace 821 BT
Monospace 821 Bold BT
OB035RM1
etc.

```

## FONTS

**Purpose** Statement returning the names of all fonts stored in the printer's memory to the standard OUT channel.

**Syntax** FONTS

**Example** Send this statement to return a list of all stored fonts:

```
FONTS
```

results in:

```

Century Schoolbook BT DingDings SWA
Dutch 801 Bold BT Dutch 801 Roman BT
Futura Light BT Letter Gothic 12 Pitch BT
MS030RMN MS030RMN.1
MS030RMN.2 MS050RMN
MS050RMN.1 MS050RMN.2
MS060BMN MS060BMN.1
MS060BMN.2 Monospace 821 BT
Monospace 821 Bold BT OB035RM1
OB035RM1.1 OB035RM1.2
OCR-A BT OCR-B 10 Pitch BT
Prestige 12 Pitch Bold BT SW020BSN
SW020BSN.1 SW020BSN.2
SW030RSN SW030RSN.1
SW030RSN.2 SW050RSN
SW050RSN.1 SW050RSN.2
SW060BSN SW060BSN.1
SW060BSN.2 SW080BSN
SW080BSN.1 SW080BSN.2
SW120BSN SW120BSN.1
SW120BSN.2 Swiss 721 BT
Swiss 721 Bold BT Swiss 721 Bold Condensed BT
Zurich Extra Condensed BT
3569264 bytes free 1717240 bytes used
Ok

```

## FOR...TO...NEXT

**Purpose** Statement for creating a loop in the program execution, where a counter is incremented or decremented until a specified value is reached.

**Syntax** FOR<nvar>=<nexp1>TO<nexp2> [STEP<nexp3>] NEXT [<nvar>]

where:

- <nvar> is the variable to be used as a counter.
- <nexp1> is the initial value of the counter.
- <nexp2> is the final value of the counter.
- <nexp3> is the value of the increment or decrement.

**Remarks** This statement is always used in connection with a NEXT statement.

The counter <nvar> is given an initial value by the numeric expression <nexp1>. If no increment value (STEP <nexp3>) is given, the value 1 is assumed. A negative increment value produces a decremental loop.

Each time the statement NEXT is encountered, the loop is executed again until the final value (specified by <nexp2>), is reached. Then the execution proceeds from the first line after the NEXT statement.

If the optional variable is omitted in the NEXT statement, the program execution loops back to the most recently encountered FOR statement.

If the NEXT statement includes a variable, the execution loops back to the FOR statement specified by the same variable.

FOR...NEXT loops can be nested, which means that a loop can contain another loop, and so on. However, each loop must have a unique counter designation and the inside loop must be concluded by a NEXT statement before the outside loop can be executed.

**Example 1** In this example, the counter A% is incremented from 10 to 50 in steps of 20:

```
10 FOR A%=10 TO 50 STEP 20
20 PRINT A%
30 NEXT
RUN
```

results in:

```
10
30
50
```

**Example 2** In this example, the counter B% is decremented from 50 to 10 in steps of 20:

```
10 FOR A%=50 TO 10 STEP -20
20 PRINT A%
30 NEXT
RUN
```

results in:

```
50
30
10
```

# FORMAT

**Purpose** Statement for formatting either the printer's permanent memory or a CompactFlash memory card.

**Syntax** FORMAT<sexp> [, <nexp1> [, <nexp2>] [, A]]

where:

<sexp> specifies the device to be formatted either as "/c" or "card1:"

<nexp1> specifies the number of entries in the root directory (only applicable when <sexp> = "card1:" and "A" flag is set). Default is 208 entries.

<nexp2> specifies the number of bytes per sector (only applicable when <sexp> = "card1:" and "A" flag is set). Default is 512 bytes per sector.

**Remarks** FORMAT "/c" :

- Formats the printer permanent memory partially or completely. System files are distinguished by a leading period character (for example, .setup.saved). This makes it possible to format the permanent memory without removing the system files.

If no "A" flag is included in the statement, all files excluding those starting with a period character (.) will be removed ("soft" formatting).

If an "A" flag is included in the statement, all files including those starting with a period character (.) will be removed ("hard" formatting).

Be careful. There is no way to undo a FORMAT operation.

FORMAT "card1:"

- Formats a CompactFlash card inserted in the printer's optional memory card adapter to MS-DOS format. Optionally, you can specify the number of entries in the root directory (that is number of files on the card) and the number of bytes per sector, provided an "A" flag is included in the statement ("hard" formatting).

When a FORMAT statement is executed, any existing data or previous formatting in the card is erased. After formatting, such a memory card can be OPENed for INPUT/OUTPUT/APPEND or RANDOM access and can also be used in a PC for storing MS-DOS files. The DOS-formatted memory card is referred to as device "card1:".

**Example 1** In this example, issuing the statement FILES before and after a FORMAT "/c" statement shows how the memory is affected. Note that system files starting with a period character are not removed, since the FORMAT statement does not contain any "A" flag:

FILES "/c", A

resulting in:

```
Files on /c
./ 0 ./ 0
APPLICATION 1 boot/ 0
ADMIN/ 0 .setup.saved 222
STDIO 4
2222080 bytes free 227 bytes used

Ok
FORMAT "/c"
Ok
```

**Example 2**    FILES "/c",A  
resulting in:

```
Files on /c
./ 0 ./ 0
boot/ 1 ADMIN/ 0
.setup.saved 222
222412 bytes free 222 bytes used
```

In the following statement, a CompactFlash memory card is formatted to MS-DOS format in the immediate mode. The number of entries is increased from 208 (default) to 500 and the size of the sectors is decreased from 512 bps (default) to 256 in order to make the card better suited for more but smaller files. The “A” flag specifies “hard” formatting.

```
FORMAT "card1:",500,256,A
```

## **FORMAT DATE\$**

**Purpose** Statement for specifying the format of the string returned by DATE\$("F") and DATEADD\$(....., "F") instructions.

**Syntax** FORMAT DATE\$<sexp>

where <sexp> is a string representing the order between year, month and date plus possible separating characters. Default format is YYMMDD, where:

- YY = Last two digits of the year (for example, 2007 = 07).
- MM = Two digits representing the month. Range is 01 to 12.
- DD = Two digits representing the day. Range is 01 to 31.

Example: December 1, 2007 is entered as “071201”.

Reset to default by: Empty string (“ ”)

**Remarks** DATE\$ and DATEADD\$ only return formatted dates if these functions include the flag “F”.

In the FORMAT DATE\$ statement, each Y, M or D character generates one digit from the number of the year, month or day respectively, starting from the end. If the number of Ys exceeds 4, or the number of Ms or Ds exceeds 2, the exceeding characters generate leading space characters.

Examples (the year is 2007):

Y generates 7  
 YY generates 07  
 YYY generates 007  
 YYYY generates 2007  
 YYYYYY generates .2007 (. represents a space)

Characters other than Y, M, or D are treated as separators and are returned as entered.

The date format is saved in the temporary memory and must be transmitted to the printer after each power-up.

**Example 1** This example changes the date format to British standard:

```
FORMAT DATE$ "DD/MM/YY"
```

**Example 2** This changes the date format back to default (YYMMDD):

```
FORMAT DATE$ " "
```

**Example 3** This changes the date format to Swedish standard:

```
FORMAT DATE$ "YY-MM-DD"
```

## FORMAT INPUT

**Purpose** Statement for specifying separators for the LAYOUT RUN statement used in the Intermec Direct Protocol.

**Syntax** FORMAT INPUT<sexp1>[,<sexp2>[,<sexp3>[,<sexp4>]]]

where:

<sexp1> is the start -of-text separator, default STX (ASCII 02 dec.).

<sexp2> is the end-of-text separator, default EOT (ASCII 04 dec.).

<sexp3> is the field separator, default CR (ASCII 13 dec.).

<sexp4> is a string of characters to be filtered out.

**Remarks** The LAYOUT RUN statement is used in the Intermec Direct Protocol to transmit variable data to a predefined layout. By default, the string of input data to the various layout fields starts with a STX character and ends with an EOT character. The various fields are separated by CR (carriage return) characters.

To provide full compatibility with various protocols and computer systems, these separators can be changed at will by means of the FORMAT INPUT statement. Each separator can have a maximum length of 10 characters.

As an option, it is possible to specify a string (up to 10 characters) to be filtered out. By default, the string is empty and is reset to default if a new FORMAT INPUT with less than four arguments is issued.

There is a timeout if ETX is not found within 60 seconds after STX has been received.

Always execute the FORMAT INPUT statement in the Immediate Mode.

If you are using the Intermec Direct Protocol, exit it using an INPUT OFF statement before changing the separators using a FORMAT INPUT statement. Then you can enter the Intermec Direct Protocol again using an INPUT ON statement.

An error occurs if you issue a FORMAT INPUT statement where one, two or three separators are identical to those already in effect without leaving the Intermec Direct Protocol.

If a certain separating character cannot be produced by the keyboard of the host, use a CHR\$ function to specify the character by its ASCII value. The separators are stored in the temporary memory and must be transmitted to the printer after each power-up.

**Example** This example changes the start-of-text separator to #, the end-of-text separator to LF (linefeed), and the field separator to @ after temporarily switching to Immediate Mode.

```
INPUT OFF .
FORMAT INPUT "#",CHR(10), "@" .
INPUT ON .
```

## **FORMAT TIME\$**

**Purpose** Statement for specifying the format of the string returned by TIME\$(“F”) and TIMEADD\$(“F”) instructions.

**Syntax** FORMAT TIME\$<sexp>

where <sexp> is a string in this format:

H represents hours in a 24 hour cycle (one digit per H).

h represents hours in a 12 hour cycle (one digit per h).

M represents minutes (one digit per M).

S represents seconds (one digit per S).

P represents AM/PM in connection with a 12 hour cycle.

p represents am/pm in connection with a 12 hour cycle.

All other characters produce separator characters.

Default string is HHMMSS. Reset to default by sending an empty string.

**Remarks** Each H, h, M, and S character generates one digit. If the number of each character exceeds 2, leading space characters are inserted. Each uppercase or lowercase P character generates one character of AM/PM or am/pm respectively, when a 12-hour cycle is selected.

Hour, minute and second fields are right-justified, whereas am/pm and AM/PM fields are left-justified.

Example (the hour is 8 o'clock in the morning):

h generates 8  
hh generates 08  
hhh generates .08  
pp generates am

To get 12-hour cycle, all hour format characters must be lowercase "h".

Separating characters are returned as entered in the string. Any character other than H, h, M, S, P, or p is regarded as a separator character.

The time format is saved in the temporary memory and has to be transmitted to the printer after each power-up.

**Example 1** Changing the time format according to Swedish standard:

```
FORMAT TIME$ "HH.MM.SS"
```

**Example 2** Changing the date format to British standard:

```
FORMAT TIME$ "hh:MM pp"
```

## FORMAT\$

**Purpose** Function for formatting a number represented by a string.

**Syntax** FORMAT\$ (<sexp1>, "<nexp1>b" | "d")

where:

<sexp1> is a string of integers or ASCII characters.

<nexp1> is the number of bytes to output in integer to ASCII conversion.

Or,

```
FORMAT$ (<sexp1>, <sexp2>)
```

where:

<sexp1> is the string of numerals, optionally with decimals, which is to be formatted.

<sexp2> specifies the format of the string.

**Remarks** FORMAT\$ can be used for conversion of strings from ASCII to integer numbers and vice versa when used as in the first syntax above.

<nexp1>b = Specifies conversion from integer to ASCII format.

This converts *<sexp1>* from a number in integer format to the corresponding characters in ASCII format. *<nexp1>* specifies the number of bytes to output. For example FORMAT\$("1380337988", "4b") yields RFID.

d = Specifies conversion from ASCII format to integer.

The string *<sexp1>* is translated from an ASCII string to the same number in integer format. For example, FORMAT\$("A", "d") yields 65.

FORMAT\$ is also used to convert a number to a specific display format. The original string (*<sexp1>*) is a string of digits, optionally with a decimal point to separate decimals from integers. It can also contain leading plus (+), minus (-), and space characters. Space characters are ignored, whereas the usual mathematical rules apply to plus and minus signs.

All other characters (or plus, minus, and space characters in other positions than leading) generate errors. The format is specified by a string (*<sexp2>*). The string can contain any characters, but some have special meanings. Note the explanation of the following characters.

0 = Digit place holder, display a digit or zero.

If the input number has fewer digits than there are zeros (on either side of the decimal separator) in the format string, leading or trailing zeros are displayed.

If the number has more digits to the left side of the decimal separator than there are zeros to the left side of the separator in the format string the digits will be displayed. If the number has more digits to the right of the separator than there are zeros to the right of the decimal separator in the format string, the decimal will be truncated to as many decimal places as there are zeros.

# = Digit placeholder, display a digit or nothing.

If there is a digit in the expression being formatted in the position where the # appears in the format string, display the digit or otherwise display nothing in that position. If the number has more digits to the left side of the decimal separator than there are # to the left side of the separator in the format string the digits will be displayed.

. = Decimal separator, to separate the integer and the decimal digits.

, = Decimal separator, to separate the integer and the decimal digits.

\ = Display the next character in the format string.

The backslash itself is not displayed. To display a \, use two backslashes.

Only the space character is displayed in the formatted string without a backslash.

space = Space

A space is displayed as a literal character wherever it is in the expression format:

- An empty format string is equivalent to “0.#####”.
- 0 and # cannot be mixed in every way. Before the decimal separator, use # first and then 0. After the decimal separator, use 0 first and then #.

For example: #####00.000### is OK and #00##0.###0#00 is not.

- A point or a comma separates integers and decimals. The decimal separator used in the format will be the returned separator type. Independent of the separator type in the number, the format type will control the return type. Default type is a point.
- A format can consist of separators as space between thousands or from a unit (such as \$, as in this example: “\$ ### ### 000.00”).
- The attached number string is truncated to the decimal quantity of the format.

Characters are not displayed on the left side of the decimal separator if there is a # on the left side of the characters, and the string to be formatted does not have a digit in the same position as the #. On the right side of the decimal separator, characters are not displayed if there is a # on the right side of the characters, and the string to be formatted does not have a digit in the same position as the #. For example:

Format string: “\\$#\t\|e\|x\|t0.0\|t\|e\|x\|t#\|\$”

String to be formatted: 1.1 55 0.33 55.33

Returned strings: \$1.1\$ \$5text5.0\$ \$0.3text3\$ \$5text5.3text3\$

Input number: “5” “-5” “0.5” “55555” “0.66666666666666”

Input format: Returned number:

“ ” => 5 -5 0.5 55555 0.6666666666

“0” => 5 -5 0 55555 0

“0.00” => 5.00 -5.00 0.50 55555.00 0.66

“\\$0,0” => \$5,0 \$-5,0 \$0.5 \$55555,0 \$0,6

“0.0##” => 5.0 -5.0 0.5 55555.0 0.666

“###\,000.0” => 005.0 -005.0 000.5 55,555.0 000.6

“# 0 0.0” => 0 5.0 -0 5.0 0 0.5 555 5 5.0 0 0.6

**Example 1** This example demonstrates addition:

```
B$="234.9"
C$="1001"
D$="# ##0.##"
A$=FLOATCALC$(B$, "+" , C$, 15)
PRINT A$
```

results in:

```
"1235.90000000000000"
PRINT FORMAT$(A$, D$)
```

results in:

"1 235.9"

**Example 2** This example demonstrates subtraction:

```
A$=FLOATCALC$ ("234.90", "-", "100.013", 2)
PRINT A$
```

results in:

"134.88"

```
PRINT FORMAT$ (A$, "\$ 0,000#")
```

results in:

"\\$ 134,880"

Note: If a higher precision is used in FLOATCALC\$, A\$ yields "\$134,887".

**Example 3** This example demonstrates multiplication:

```
B$="3"
A$=FLOATCALC$ ("100", "*", B$, 1)
PRINT A$
```

results in:

"300.0"

```
C$="0 0 0,00###"
PRINT FORMAT$ (A$, C$)
```

results in:

"3 0 0,00"

**Example 4** This example demonstrates division:

```
B$="1.0"
A$=FLOATCALC$ (B$, "/", "3.0")
PRINT A$
```

results in:

"0.3333333333"

```
PRINT FORMAT$ (A$, "\$ 000.00###")
```

results in:

"\\$ 000.3333"

## FORMFEED (FF)

**Purpose** Statement for feeding out or pulling back a certain length of media.

**Syntax** FORMFEED | FF [<nexp>]

where <nexp> (optional) is the feed length expressed as a positive or negative number of dots.

**Remarks** If no value is entered after the FORMFEED statement, the printer feeds out one single label, ticket, tag, or portion of continuous stock according to the printer setup. For more information, see the user's guide for your printer.

If a value is entered after the FORMFEED statement, the media is fed out or pulled back by the corresponding number of dots:

- A positive number of dots makes the printer feed out the specified length of media.
- A negative number of dots makes the printer pull back the specified length of media. To avoid causing a media jam, do not enter a value larger than the length of the label.

It is important whether a FORMFEED statement is issued before or after a PRINTFEED statement:

- FORMFEED statement issued before PRINTFEED affects the position of the origin on the first copy to be printed.
- FORMFEED statement issued after PRINTFEED does not affect the position of the origin on the first copy, but the next copy will be affected. Do not use FORMFEED as a replacement for start- and stopadjustments in the Setup Mode or in connection with batch printing.

**Example 1** This example prints a line of text and feeds out an extra 60 dots of media after printing:

```
10 FONT "Swiss 721 BT"
20 PRPOS 30,200
30 PRTXT "HELLO"
40 PRINTFEED
50 FORMFEED 60
RUN
```

**Example 2** This example pulls back the media 20 dots before printing:

```
10 FORMFEED -20
20 FONT "Swiss 721 BT"
30 PRPOS 30,200
40 PRTXT "HELLO"
50 PRINTFEED
RUN
```

In this case, the positioning of the text line will be performed after the media has been pulled back.

## FRE

**Purpose** Function returning the number of free bytes in a specified part of the printer memory.

**Syntax** FRE (<<*sexp*> | <*nexp*>>)

where:

*<sexp>* is the designation of the part of the printer's memory from which the number of free bytes should be returned, for example “/c”, “tmp:”, “card1:”.

*<nexp>* is a dummy argument that returns the number of free bytes in the printer's temporary memory (“tmp:”).

**Remarks** The firmware looks for a colon (:) character in the argument for the FRE function. If the argument is a valid name of a memory device, the number of free bytes in that device is returned.

If the argument specifies device “card1:”, but no card is inserted, error 1039 (“Not mounted”) occurs.

If the name of a device that is not a part of the printer memory (for example “uart1:” or “console:”), is entered as an argument, FRE returns 0.

See DEVICES for more information on memory and non-memory devices.

If the argument contains a colon, but is not a valid name of any device (for example “QWERTY:”), error 1013 (“Device not found”) occurs.

Any argument that does not include a colon character (for example, “7” or “QWERTY”) returns the amount of free bytes in the printer temporary memory (“tmp:”).

**Example 1** PRINT FRE ("tmp:")

results in:

2382384

**Example 2** PRINT FRE ("uart1:")

results in:

0

**Example 3** PRINT FRE(1)

results in:

2382384

## **FUNCTEST**

**Purpose** Statement for performing various hardware tests.

**Syntax** FUNCTEST*<sexp>*,*<svar>*

where:

*<sexp>* is the type of test to be performed:  
“CARD”  
“HEAD” (only “HEAD” yields a meaningful response)  
“KERNEL”  
“ROMn”

*<svar>* is the variable in which the result will be placed.

**Remarks** The test has a number of possible responses as described in the next table.

| Response             | Description                                                                                                                                                            |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <sexp> = “CARD”      | Not supported.                                                                                                                                                         |
| NOT IMPLEMENTED      |                                                                                                                                                                        |
| <sexp> = “HEAD”      | The test was successful. n is the number of dots on the printhead.                                                                                                     |
| HEAD OK, SIZE:n DOTS |                                                                                                                                                                        |
| HEAD LIFTED          | Printhead is lifted and must be lowered before test can be performed.                                                                                                  |
| FAULTY PRINthead     | One or more dots on the printhead are not working.<br>Note that the voltage for the printhead is not checked.<br>Use the HEAD function for additional printhead tests. |
| <sexp> = “KERNEL”    | Not supported.                                                                                                                                                         |
| NOT IMPLEMENTED      |                                                                                                                                                                        |
| <sexp> = “ROMn”      | Not supported.                                                                                                                                                         |
| NOT APPLICABLE       |                                                                                                                                                                        |

Any other input to <sexp> yields an empty string.

**Example** This example shows how a test program using the FUNCTEST statement may be composed:

```
10 FUNCTEST "HEAD", A$
20 PRINT "HEADTEST:", A$
RUN
```

resulting in:

```
HEADTEST: HEAD OK, SIZE:832 DOTS
Ok
```

## FUNCTEST\$

**Purpose** Function returning the result of various hardware tests.

**Syntax** FUNCTEST\$ (<sexp>)

where <sexp> is the type of test to be performed:

“CARD”

“HEAD” (only “HEAD” yields a meaningful response)

“KERNEL”

“ROMn”

**Remarks** Responses are described in the next table.

| Response             | Description                                                                                                                                                            |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <sexp> = "CARD"      | Not supported.                                                                                                                                                         |
| NOT IMPLEMENTED      |                                                                                                                                                                        |
| <sexp> = "HEAD"      | The test was successful. n is the number of dots on the printhead.                                                                                                     |
| HEAD OK, SIZE:n DOTS |                                                                                                                                                                        |
| HEAD LIFTED          | Printhead is lifted and must be lowered before test can be performed.                                                                                                  |
| FAULTY PRINthead     | One or more dots on the printhead are not working.<br>Note that the voltage for the printhead is not checked.<br>Use the HEAD function for additional printhead tests. |
| <sexp> = "KERNEL"    | Not supported.                                                                                                                                                         |
| NOT IMPLEMENTED      |                                                                                                                                                                        |
| <sexp> = "ROMn"      | Not supported.                                                                                                                                                         |
| NOT APPLICABLE       |                                                                                                                                                                        |

Any other input to <sexp> yields an empty string.

**Example** This example shows how a test program using the FUNCTEST\$ function may be composed (compare with the example for FUNCTEST statement):

```
PRINT "HEADTEST:", FUNCTEST$ ("HEAD")
```

results in:

```
HEADTEST: HEAD OK, SIZE:1280 DOTS
Ok
```

## GET

**Purpose** Statement for reading a record from a random file to a random buffer.

**Syntax** GET [#]<nexp1>,<nexp2>

where:

# (optional) indicates that whatever follows is a number.

<nexp1> is the number assigned to the file when it was OPENed.

<nexp2> is the number of the record. Must be \_, 0.

**Remarks** The GET statement is used to read a certain record in a certain random file to a buffer, where the record will be assigned to variables according to the FIELD statement given for the buffer. After the GET statement has been executed, you can use references to the variables defined by the FIELD statement to read the characters in the random buffer.

Numeric expressions converted to string expressions (by STR\$ functions before being put into the buffer) can be converted back to numeric expressions using VAL functions.

**Example**

```
10 OPEN "PHONELIST" AS #8 LEN=26
20 FIELD#8,8 AS F1$, 8 AS F2$, 10 AS F3$
30 SNAME$="SMITH"
```

```

40 CNAME$="JOHN"
50 PHONE$="12345630"
60 LSET F1$=SNAME$
70 LSET F2$=CNAME$
80 RSET F3$=PHONE$
90 PUT #8,1
100 CLOSE#8
RUN
SAVE "PROGRAM 1.PRG "
NEW
10 OPEN "PHONELIST" AS #8 LEN=26
20 FIELD#8,8 AS F1$, 8 AS F2$, 10 AS F3$
30 GET #8,1
40 PRINT F1$,F2$,F3$
RUN

```

results in:

```
SMITH --- JOHN ----- 12345630
```

## GETASSOC\$

**Purpose** Function for getting a value from a string association.

**Syntax** GETASSOC\$ (<sexp1>, <sexp2>)

where:

<sexp1> is the name of the association (case-sensitive).

<sexp2> is the name of a tuple in the association.

**Remarks** An association is an array of tuples, where each tuple consists of a name and a value.

**Example** This example defines a string, including three string names associated with three start values, and changes one of them (time):

```

10 QUERYSTRING$="time=UNKNOWN&label=321&desc=DEF"
20 MAKEASSOC"QARRAY",QUERYSTRING$,"HTTP"
30 QTIME$=GETASSOC$("QARRAY","time")
40 QLABELS%=VAL(GETASSOC$("QARRAY","label"))
50 QDESC$=GETASSOC$("QARRAY","desc")
60 PRINT"time=";QTIME$,"LABEL=";QLABELS%,"DESCRIPTION=";QDESC$
70 SETASSOC"QARRAY","time",time$
80 PRINT"time=";GETASSOC$("QARRAY","time")
RUN

```

resulting in:

```
time=UNKNOWN LABEL=321 DESCRIPTION=DEF
time=153355
```

## GETASSOCNAME\$

**Purpose** Function for traversing the tuples of a string association.

**Syntax** GETASSOCNAME\$ (<sexp>, <nexp>)

where:

<sexp> is the association to be traversed (case-sensitive).

<nexp> specifies the tuple in the association:

<nvar> = 0 specifies first tuple.

<nvar> \_, 0 specifies next tuple.

**Remarks** An association is an array of tuples, where each tuple consists of a name and a value. To get the first position in the string association, <nvar> should be zero. Consecutive calls to GETASSOCNAME\$ with <nvar> non zero will traverse all variables in an undefined order. When a blank string (" ") is returned, the last variable has been traversed.

**Example** This example shows how "QARRAY" is traversed (run example from GETASSOC first):

```
10 LVAL$=GETASSOCNAME$ ("QARRAY", 0)
20 WHILE LVAL$<>""
30 RVAL$=GETASSOC$ ("QARRAY", LVAL$)
40 PRINT LVAL$; "="; RVAL$
50 LVAL$=GETASSOCNAME$ ("QARRAY", 1)
60 WEND
RUN
```

results in:

```
label=321
desc=DEF
time=153355
```

## GETPFSVAR

**Purpose** Function for recovering saved variables.

**Syntax** GETPFSVAR (<sexp> [, D] )

where:

<sexp> is the name of the variable (uppercase characters only).

D (optional) specifies that the variable is to be deleted after recovery.

**Remarks** This function is used to recover variables registered to be saved at power failure by means of a SETPFSVAR statement, and returns either -1 on success or 0 at failure.

If a D flag is included, the variable is deleted after it has been recovered. This can be used to make sure that the variable is up to date and that no old obsolete value is recovered. The variable name is limited to a length of twenty characters.

Related instructions are SETPFSVAR, DELETEPFSVAR, and LISTPFSVAR.

**Example**

```

10 IF NOT GETPFSVAR ("QS$") THEN QS$ ="<this is the default
value, set a new one>"
20 IF NOT GETPFSVAR ("QCPS%") THEN PRINT "No copies
available":END
30 QSTATUS%=GETPFSVAR ("AWE$",D):IF QSTATUS%
THEN PRINT "Recovered successfully!"

40 SETPFSVAR "QCPS%"
50 'Build label
60
99 PRINTFEED; QCPS%=QCPS%
100

```

## GOSUB

**Purpose** Statement for branching to a subroutine.

**Syntax** GOSUB<ncon>|<line label>

where:

<ncon> is the number of the first line in the desired subroutine.

<line label> is the label of the first line in the desired subroutine.

**Remarks** After branching, the subroutine will be executed line by line until a RETURN statement is encountered.

The same subroutine can be branched to many times from different lines in the main program. GOSUB always remembers where the branching took place, which makes it possible to return to the correct line in the main program after the subroutine has been executed.

Subroutines may be nested, which means that a subroutine may contain a GOSUB statement for branching to a secondary subroutine and so on. Subroutines are normally placed on program lines with higher numbers than the main program. The main program should be appended by an END statement to avoid unintentional execution of subroutines.

**Example 1** This example makes use of line numbers:

```

10 PRINT "This is the main program"
20 GOSUB 1000
30 PRINT "You're back in the main program"
40 END
1000 PRINT "This is subroutine 1"
1010 GOSUB 2000
1020 PRINT "You're back from subroutine 2 to 1"
1030 RETURN
2000 PRINT "This is subroutine 2"
2010 GOSUB 3000
2020 PRINT "You're back from subroutine 3 to 2"
2030 RETURN
3000 PRINT "This is subroutine 3"
3010 PRINT "You're leaving subroutine 3"

```

```
3020 RETURN
RUN
```

resulting in:

```
This is the main program
This is subroutine 1
This is subroutine 2
This is subroutine 3
You're leaving subroutine 3
You're back from subroutine 3 to 2
You're back from subroutine 2 to 1
You're back in the main program
Ok
```

- Example 2** In this example, line numbers have been omitted and line labels are used to make the program branch to subroutines:

```
IMMEDIATE OFF
PRINT "This is the main program"
GOSUB SUB1
PRINT "You're back in the main program"
END
SUB1: PRINT "This is subroutine 1"
GOSUB SUB2
PRINT "You're back from subroutine 2 to 1"
RETURN
SUB2: PRINT "This is subroutine 2"
GOSUB SUB3
PRINT "You're back from subroutine 3 to 2"
RETURN
SUB3: PRINT "This is subroutine 3"
PRINT "You're leaving subroutine 3"
RETURN
```

## GOTO

**Purpose** Statement for branching unconditionally to a specified line.

**Syntax** GOTO< ncon > | < line label >

where:

< ncon > is the number of the line to be branched to.

< line label > is the label of the line to be branched to.

**Remarks** If the specified line contains an executable statement, both that statement and all that follows are executed. If the specified line does not exist, an error condition occurs.

The GOTO statement can also be used in the immediate mode to resume execution of a program which has been terminated using a STOP statement at a specified program line.

**Example** In this example the first bar of the tune “Colonel Bogey” is played only if the title is entered correctly. The message “Try again” is displayed until you type the right name.

```

10 A$="COLONEL BOGEY"
20 B$="TRY AGAIN"
30 INPUT "TITLE"; C$
40 IF C$=A$ GOTO 100 ELSE PRINT B$
50 GOTO 30
60 END
100 SOUND 392,15
110 SOUND 330,20
120 SOUND 330,15
130 SOUND 349,15
140 SOUND 392,15
150 SOUND 659,25
160 SOUND 659,20
170 SOUND 523,25
180 GOTO 60
RUN

```

resulting in:

TITLE?

Note the way GOTO is used in line 50 to create a loop, which makes the printer await the condition specified in line 40 before the execution is resumed. Instead of line numbers, line labels can be used following the same principles as illustrated in the second example for the GOSUB statement.

## HEAD

**Purpose** Returns the result of a thermal printhead check.

**Syntax 1** HEAD(<nexp1>)

where <nexp1> is defined as follows:

<nexp1> > or = 0      Specifies the number of a dot for which the resistance in ohms will be returned.

<nexp1> = -1      Printhead check:  
                        Returns -1 (true) if OK.  
                        Returns 0 (false) if error.

<nexp1> = -7      Returns mean printhead resistance in ohms.

**Syntax 2** <nexp2> = HEAD(<sexp>)

where:

<nexp2>    returns the number (quantity) of faulty dots.

<sexp>    returns the dot number and resistance for each faulty dot.

**Remarks** This function allows you to examine the printhead with regard to dot resistance.

There is no guarantee that all defect “dots” will be detected by the HEAD function, since only the resistance is checked. For example, dirty or cracked dots can only be detected visually.



**Note:** The EasyCoder PD41 does not have resistance measuring. Running the HEAD command on this platform does not return actual values.

The detection of a possibly faulty dot or printhead by means of the dot sensing facility does not automatically imply that the printhead is defective, or that replacement is covered by the warranty. Intermec reserves the right to examine the printhead for purposes of determining warranty coverage.

**<nexp1> or = 0**

A positive value specifies a single dot on the printhead and returns its resistance value as a number of ohms. A dot resistance value that deviates considerably from the mean resistance value of the printhead indicates that the dot may be faulty. The dot numbering starts at 0 (zero), that is, in a 832 dots printhead, the dots are numbered 0-831. On the EasyCoder PD41 the nominal resistance of the dot will be returned.

**<nexp1> = -1**

A check of the complete printhead is performed. PD41 always returns -1.

HEAD(-1)=-1     The printhead is within the allowed limits (no dot is more than  $\pm 15\%$  from the mean resistance value). This does not guarantee the printout quality.

HEAD(-1)=0     A possible error has been detected.

**<nexp1> = -7**

The mean resistance value in ohms of all dots of the printhead is returned.

The PD41 will return the nominal resistance value of the dots.

The second version of the HEAD function measures the dot resistance for every dot in the printhead. Faulty dots are reported to the system, so you do not need to use a SET FAULTY DOT statement to report bad dots one at a time. The PD41 returns 0, and <sexp> is empty.

**Examples**     Read the resistance value of dot No. 5:

```
PRINT HEAD (5)
```

Perform a printhead check:

```
PRINT HEAD (-1)
```

Read the printhead's mean resistance value:

```
PRINT HEAD (-7)
```

Check printhead for faulty dots and their respective resistance values:

```
A% = HEAD (B$)
```

results in:

Ok

```
PRINT A%
```

```

5
Ok
PRINT B$

25, 2944

42, 2944

106, 2944

107, 2944

140, 2944
Ok

```

## IF...THEN...(ELSE)

- Purpose** Statement for conditional execution controlled by the result of a numeric expression.

```

Syntax IF<nexp> [,] THEN<stmt1> [ELSE<stmt2>]
IF<nexp> [,] THEN .
<stmt1> .
[...<stmt1+n>] .
[ELSE .
<stmt2> .
[...<stmt2+n>]] .
ENDIF .

```

where:

- <nexp> is a numeric expression either true or false.
- <stmt1> is the statement or list of statements telling the program what to do should the IF-condition be true.
- <stmt2> is an optional statement or list of statements specifying what will happen should the IF-condition be false.

- Remarks** THEN and ELSE statements may be nested. Multiple THEN and ELSE statements can alternatively be entered on separate lines. If so, the instruction should be appended by ENDIF. See second example below.

- Example 1** These two examples illustrates the different syntaxes:

```

10 A%=100:B%=20
20 C$="A LARGER THAN B"
30 D$="A NOT LARGER THAN B"
40 IF A%>B% THEN PRINT C$ ELSE PRINT D$
RUN

```

resulting in:

```
A LARGER THAN B
```

- Example 2**
- ```

10 A%=VAL(TIME$)
20 IF A%>120000 THEN
30 PRINT "TIME IS ";TIME$; ". ";
40 PRINT "GO TO LUNCH!"
50 ELSE
60 PRINT "CARRY ON - ";

```

```
70 PRINT "THERE'S MORE WORK TO DO!"  
80 ENDIF  
RUN
```

resulting in:

```
TIME IS 121500. GO TO LUNCH!
```

Example 3 IF ... THEN are often used in connection with GOTO. In this example, line numbering is used. Also see the example for the GOTO statement.

```
10 A%=100  
20 B%=50  
30 IF A%=B% THEN GOTO 50 ELSE PRINT "NOT EQUAL"  
40 END  
50 PRINT "EQUAL":END  
RUN
```

resulting in:

```
NOT EQUAL
```

Example 4 This example corresponds to the preceding example, but line labels are used instead of line numbers.

```
IMMEDIATE OFF  
A%=100  
B%=50  
IF A%=B% THEN GOTO QQQ ELSE PRINT "NOT EQUAL"  
END  
QQQ: PRINT "EQUAL":END  
IMMEDIATE ON  
RUN
```

resulting in:

```
NOT EQUAL
```

IMAGE BUFFER MIRROR

Purpose Statement for mirror the print image around the Y-axis.

Syntax IMAGE BUFFER MIRROR

Remarks This statement mirrors the current defined image buffer around the Y-axis, that is, the feed direction. Fields defined after the IMAGE BUFFER MIRROR statement is executed are rendered normally. The image buffer width is always 8-bit aligned, even when the X-start parameter in the setup is not. Thus, Intermec recommends that you test that the mirrored image is printed sidewise where intended. In some cases, a small correction using the PRPOS statement or the X-start parameter could be necessary.

Example

```
NEW  
10 PRPOS 50,300  
20 FONT "Swiss 721 BT",40  
30 PRTXT "MIRROR"  
40 IMAGE BUFFER MIRROR  
50 PRPOS 50,100  
60 PRTXT "NORMAL"  
70 PRINTFEED
```



IMAGE BUFFER SAVE

Purpose Statement for saving the content of the image buffer as a file.

Syntax IMAGE BUFFER SAVE<*sexp*>

where <*sexp*> is the desired name of the file, with an optional reference to the device where the file should be saved.

Remarks This statement saves the current content of the print buffer as an image file in RLL format. After saving, the file is automatically installed as an image which can be printed using a PRIMAGE statement in DIR 1 and DIR 3. This way, you can create label templates to which variable data can easily be added.

The size of the print buffer image depends on the size of the print image at the moment the buffer is saved. The width is decided by the Media, Media Size, Width setup value with the first pixel according to the Media, Media Size, Xstart setup value. The height is decided by the actual height in the y-dimension of the print image. Note that space characters or invisible “white” parts of an image are included in the height of the print image, even if they are not visible on the printed label.

Example IMAGE BUFFER SAVE "TEMPLATE7"

IMAGE LOAD

Purpose Statement for receiving, converting and installing image and font files.

Syntax IMAGE LOAD [<nexp1>,] <sexp1>, <nexp2> [, <sexp2> [, <nexp3>]]

where:

<nexp1> (optional) is the number of bytes to skip before starting to read the data.

<sexp1> is the desired name of the image or font to be created.

<nexp2> is the size of the original file in number of bytes.

<sexp2> is an optional flag:

“S” specifies that the image or font will be saved in the printer’s permanent memory (“/c”). This option may result in poor performance.

An empty string (“ ”) specifies that the image or font will be stored in the printer’s temporary memory (“tmp:”).

<nexp3> (optional) specifies a communication channel OPENed for INPUT by the number assigned to the device. Default is Std IN channel.

Remarks This statement prepares the printer to receive a .PCX image file, an image file in the internal Intermec Fingerprint bitmap format, or a font file on the standard IN channel (see SETSTDIO statement) or on another communication channel OPEN for INPUT. When the file is received, it is automatically converted to an image in the internal bitmap format of Intermec Fingerprint or to a scalable font.

The optional first parameter makes it possible to use this statement in MSDOS (CR/LF problem).

The name of an image may consist of a maximum of 30 characters including possible extensions. The image will have the same direction as the original image file and can only be rotated 180° using a DIR statement. Intermec recommends that you include the extension .1 or .2 to indicate the intended print direction. Font file names are restricted to only 30 characters. The size of the original file should be given in bytes according to its size in the host.

Before IMAGE LOAD can be used on a serial channel, the setup must be changed to 8 characters, CTS/RTS handshake. When an IMAGE LOAD statement is executed, the execution stops and waits for the number of bytes specified in the statement to be received. During the transfer of image file data to the printer, there is a 25-second timeout between characters.

If a new character has not been received within the timeout limit, Error 80 “Download timeout” occurs. When the specified number of characters has been received, the execution resumes. If the downloading was successful, the downloaded image or font will be installed automatically and can be used without any rebooting.

Example IMAGE LOAD "Logotype.1", 400, ""

IMAGENAME\$

Purpose Function returning the names of the images stored in the printer memory.

Syntax IMAGENAME\$ (<nexp>)

where <nexp> is the result of the expression which is either false or true:

- False (0) indicates first image.
- True (_,0) indicates next image.

Remarks This function can be used to produce a list of all images. You can also use the IMAGES statement.

Image files downloaded by means of a TRANSFER KERMIT statement will not be returned, since the software regards them as files rather than images.

IMAGENAME\$(0) produces the first name in the memory.

IMAGENAME\$(_,0) produces the next name. Can be repeated as long there are any image names left.

Example Use a program like this to list all image names:

```
10 A$=IMAGENAME$ (0)
20 IF A$="" THEN END
30 PRINT A$
40 A$=IMAGENAME$ (-1)
50 GOTO 20
RUN
```

resulting in:

```
CHESS2X2.1
CHESS4X4.1
DIAMONDS.1
GLOBE.1
Ok
```

IMAGES

Purpose Statement for returning the names of all images stored in the printer memory to the standard OUT channel.

Syntax IMAGES

Remarks This statement (or an IMAGENAME\$ function) can be used to list all image names. Image files downloaded by means of a TRANSFER KERMIT statement are not printed, since the firmware regards them as files rather than images.

Example Send this command to see a list of images stored in the printer memory:

IMAGES

resulting in:

```
CHESS2X2.1 CHESS4X4.1  
DIAMONDS.1 GLOBE.1  
3568692 bytes free 1717812 bytes used  
Ok
```

IMMEDIATE

Purpose Statement for enabling or disabling Immediate Mode in connection with program editing without line numbers, for reading the current mode, or for reading the current standard IN and OUT channels.

Syntax IMMEDIATE ON|OFF|MODE|STDIO

where:

ON enables Immediate Mode.

OFF disables Immediate Mode.

MODE Prints a line to the STDOUT port with information on the current status of the following modes (ON or OFF):

Execution

Immediate

Input

Layout Input

Debug STDIO (dbstdio)

STDIO prints two lines to the STDOUT port with information on current settings for the STDIN and STDOUT channels.

Remarks IMMEDIATE ON|OFF

Before starting to write a program without line numbers, the immediate mode must be disabled by means of an IMMEDIATE OFF statement. If not, each line will be executed immediately.

After an IMMEDIATE OFF statement, program lines can be entered without any leading line numbers. References to lines are done using “line labels,” which are called in GOTO or GOSUB and related statements.

A line label is a name followed by a colon (:). The label must not interfere with any keywords or start with a digit and the line must start with the line label. When a line label is used as a reference to another line (for example, within a GOTO statement) the colon should be omitted.

The program should be appended by a IMMEDIATE ON statement. At the execution of this statement, the program lines are numbered automatically in ten-step incremental order, starting with the first line (10-20-30-40-50...). These line numbers will not appear on the screen until the program is LISTed, LOADED, or MERGED. Line labels are not converted to line numbers.

Do not issue a RUN statement before the IMMEDIATE ON statement, or an error occurs.

IMMEDIATE MODE

Execution On/Off indicates if a Fingerprint program is running or not. Immediate On/Off indicates whether the Immediate Mode is enabled or disabled as specified by IMMEDIATE ON/OFF.

Input On/Off indicates whether the Direct Protocol is enabled or disabled as specified by INPUT ON/OFF.

Layout Input On/Off indicates whether or not a layout is being recorded in the Direct Protocol as specified by LAYOUT INPUT and LAYOUT END.

Dbstdio On/Off indicates whether the debug standard I/O is active or not.

The following conditions are not reported:

- Running a Fingerprint application.
- Execution of a TRANSFER KERMIT, FILE& LOAD, IMAGE LOAD, LOAD, and STORE INPUT instruction.
- Running external commands (ush), for example RUN“rz.....”
- Running the Setup Mode or execution of a SETUP statement.

IMMEDIATE STDIO

Two lines will be transmitted on the STDOUT port with information on the current STDIN and STDOUT channels regarding port, baud rate, character length, parity, and stop bits.

Example 1 A program can be written without using any line numbers, as illustrated by this short example. QQQ is used as a line label:

IMMEDIATE OFF

resulting in:

```
Ok
PRINT "LINE 1"
GOSUB QQQ
END
QQQ: PRINT "LINE 2"
RETURN
IMMEDIATE ON
Ok
RUN
```

resulting in:

```
LINE 1  
LINE 2  
Ok
```

Example 2 This example shows how the status of the various modes are checked:

```
IMMEDIATE MODE
```

resulting in:

```
execution=OFF, immediate=ON, input=OFF, layout input = Off
```

Example 3 This example shows how the status of the STDIN and STDOUT channels are checked:

```
IMMEDIATE STDIO
```

resulting in:

```
stdin=uart1:, 9600, 8, NONE, 1  
stdout=uart1:, 9600, 8, NONE, 1
```

INKEY\$

Purpose Function reading the first character in the receive buffer of the standard IN channel.

Syntax INKEY\$

Remarks For information on standard I/O channels, see SETSTDIO statement.

As opposed to the INPUT statement, INKEY\$ does not interrupt the program flow to wait for input data, unless a loop is created by means of a GOTO statement, see line 20 in the example below.

INKEY\$ is useful when the host computer is unable to end the input data with a “Carriage Return” (CR; ASCII 13 dec.), but must use some other character, for example “End of Text” (ETX; ASCII 3 dec.). Then a routine, which interprets the substitute character as a carriage return, can be created.

Example In this example, none of the characters received on the standard IN channel are printed on the host screen until a # character (ASCII 35 dec.) is encountered.

```
10 A$ = INKEY$  
20 IF A$ = "" GOTO 10  
30 IF A$ = CHR$(35) THEN PRINT B$  
40 IF A$ = CHR$(35) THEN END  
50 B$ = B$ + A$  
60 GOTO 10  
RUN
```

Type a number of characters on the keyboard of the host. They will not be printed on the host screen until you type a # character. Then all the characters appear simultaneously, except for the #-sign.

Note the loop between line 10 and 20, which makes the program wait for you to activate a key.

INPUT (IP)

- Purpose** Statement for receiving input data via the standard IN channel during the execution of a program.
- Syntax** INPUT | IP [<scon><;|,>] <<nvar>|<svar>> [, <<nvar>|<svar>> . . .]
where:
<scon><;|,> is an optional prompt string, followed by a semicolon or comma.
<<nvar>|<svar>> are variables to which the input data will be assigned.
- Remarks** For information on standard I/O channel, see “[SETSTDIO](#)” on page 185. During the execution of a program, an INPUT statement interrupts the execution. A question mark or a prompt is displayed on the host screen to indicate that the program is expecting additional data to be entered. The prompt can be used to tell the operator what type of data to enter. The prompt is appended by a question mark if a semicolon (;) is entered after the prompt string. If a comma (,) is used in that position, the printing of the question mark is suppressed. If a prompt is not used, the question mark is always displayed. Do not enter a comma or semicolon directly after the keyword, but only after the prompt, or to separate variables. The input data should be assigned to one or several variables. Each item of data should be separated from the next item by a comma. The number of data items entered must correspond to the number of variables in the list, or an error condition occurs. The variables may be any mix of string variables and numeric variables, but the type of input data must agree with the type of the variable to which the data is assigned. Input can also be done directly to the system variables TIME\$, DATE\$, and SYSVAR. The maximum number of characters that can be read using an INPUT statement is 32,767 characters. Note that INPUT filters out any incoming ASCII 00 dec. characters (NUL). INPUT does not support autohunting (see SETSTDIO).
- Examples** This example shows input to one numeric variable and one string variable:
- ```

10 INPUT "ADDRESS";A%,B$
20 PRINT A%;" ";B$
30 IF A% > 0 THEN GOTO 50
40 GOTO 10
50 END
RUN

```
- resulting in:
- ADDRESS?
- When the prompt “ADDRESS?” appears on the screen, you can type the input data on the terminal’s keyboard, for example:

999, HILL STREET

Note the separating comma. If the input text data contains a comma, which shall be printed, you must enclose the input data with quotation marks ("...."), for example:

999, "HILL STREET, HILLSBOROUGH"

Numeric input data must not include any decimal points.

**Example 2** This example shows how the date can be set directly from the keyboard of the host:

INPUT "Enter date: ",DATE\$

resulting in:

Enter date:

When the prompt “Enter date:” appears on the host screen, you can type the date as a six-digit combination of year, month and day (see DATE\$ variable). Time can also be set using the same method.

## **INPUT ON/OFF**

**Purpose** Statement enabling or disabling the Intermec Direct Protocol.

**Syntax** INPUT ON|OFF

Default: INPUT OFF

**Remarks** INPUT ON enables the Intermec Direct Protocol:

- Enables reception of input data to a stored layout
- Starts the error handler
- Sets the verbosity to off (SYSVAR (18) = 0)
- Shows “Direct Protocol 8.00” in the display

INPUT OFF disables the Intermec Direct Protocol:

- Disables reception of input data to a stored layout
- Stops the error handler
- Resets the verbosity to the level selected before last

INPUT ON was executed:

- Shows “Fingerprint 8.00” in the display

The following instructions will only work with the Intermec Direct Protocol, that is after an INPUT ON statement has been executed:

```
COUNT& ERROR FORMAT INPUT
INPUT OFF LAYOUT END LAYOUT INPUT
LAYOUT RUN
```

**Example** This example illustrates how the Intermec Direct Protocol is enabled, how new separators are specified, how a layout is stored in the printer's memory, how variable data are combined with the layout, and how a label is printed. Finally, the Intermec Direct Protocol is disabled:

```
INPUT ON ↴
FORMAT INPUT "#", "@", "&" ↴
LAYOUT INPUT "tmp:LABEL1" ↴
FT "Swiss 721 BT" ↴
PP 100,250 ↴
PT VAR1$ ↴
PP 100,200 ↴
PT VAR2$ ↴
LAYOUT END ↴
LAYOUT RUN "tmp:LABEL1" ↴
#Line number 1&Line number 2&@ ↴
PF ↴
INPUT OFF ↴
```

## INPUT#

**Purpose** Statement for reading a string of data from an OPEN device or sequential file.

**Syntax** INPUT#<nexp>, <<nvar>|<svar>> [, <<nvar>|<svar>> . . . ]

where:

<nexp> is the number assigned to the file or device when it was OPENed.

<<nvar>|<svar>> is the variable to which the input data will be assigned.

**Remarks** This statement resembles the INPUT statement, but allows the input to come from other devices than the standard IN channel or from various files. Like the INPUT statement, commas can be used to assign different portions of the input to different variables. INPUT# does not allow prompts to be used.

When reading from a sequential file, the records can be read one after the other by the repeated issuing of INPUT# statements with the same file reference.

Once a file record has been read, it cannot be read again until the file is CLOSEd and then OPENed again.

The maximum number of characters that can be read using an INPUT# statement is 32,767.

Note that INPUT# filters out any incoming ASCII 00 dec. characters (NUL).

**Example** This example assigns data from the first record in the sequential file “Addresses” to the three string variables A\$, B\$, and C\$ and from the second record in the same file to the string variables D\$ and E\$:

```
 . . .
 .
100 OPEN "ADDRESSES" FOR INPUT AS #5
110 INPUT#5, A$, B$, C$
120 INPUT#5, D$, E$
 .
 .
```

## **INPUT\$**

**Purpose** Function returning a string of data, limited in regard of number of characters, from the standard IN channel, or optionally from an OPENed file or device.

**Syntax** INPUT\$ (<nexp1> [, <nexp2>])

where:

<nexp1> is the number of characters to be read.

<nexp2> optionally specifies a file or device using the number assigned to it when it was OPENed.

**Remarks** If no file or device is specified, the input comes from the standard I/O channel. For more information, see “[SETSTDIO](#)” on page 185.

Otherwise, the input comes from the specified file or device. The execution is held until the specified number of characters has been received from the keyboard console, file, or communication channel. If a file does not contain the specified number of characters, the execution is resumed as soon as all available characters in the file have been received.

The maximum number of characters that can be returned using an INPUT\$ statement is 65,536 characters.

**Example 1** This example reads a sequence of 25 characters from the printer’s built-in keyboard and assigns them to a string variable named Z\$:

```
 . . .
 .
1000 OPEN "CONSOLE:" FOR INPUT AS #1
1010 Z$=INPUT$ (25,1)
 .
 .
```

**Example 2** In this example, 10 characters are read from the standard IN channel and assigned to a variable.

```
10 A$=INPUT$ (10)
```

## INSTR

**Purpose** Function searching a specified string for a certain character, or sequence of characters, and returning its position in relation to the start of the string.

**Syntax** INSTR ( [<nexp>, ] <sexp1>, <sexp2>)

where:

<nexp> (optional) is the position where the search will start.

<sexp1> is the string to be searched.

<sexp2> is the character(s) for which the string will be searched.

**Remarks** INSTR allows you to search a string for some particular character(s) and return the position of the character, or the first character in the sequence, as a number of characters positions measured from the start of the string.

As an option, it is possible to specify a certain position in the string from which the search will start. If no start position is specified, the search starts at the beginning of the string.

The result will be zero if:

- the start position value exceeds the length of the string.
- the string is empty.
- the searched combination of characters cannot be found.

**Example 1** In this example, the string “INTERMEC\_PRINTER\_AB” is searched for the character combination “AB”. No start position is specified.

```
10 A$="INTERMEC PRINTER AB"
20 B$="AB"
30 PRINT INSTR(A$,B$)
RUN
```

resulting in:

18

**Example 2** In this example, the string “INTERMEC\_PRINTER\_AB” is searched for the character “I” and the start position is specified as 4.

```
10 A$="INTERMEC PRINTER AB"
20 B$="I"
30 PRINT INSTR(4,A$,B$)
RUN
```

resulting in:

12

## **INVIMAGE (II)**

- Purpose** Statement for inverting the printing of text and images from “black-on-white” to “white-on-black.”
- Syntax** INVIMAGE | II  
Default is NORIMAGE.  
Reset to default by executing a PRINTFEED.
- Remarks** This statement can only be used in connection with the printing of text and images (PRTXT and PRIMAGE). In the matrix of the font or image, all “white” dots will be black and all black dots will be “white.” Obviously, “white” means that the media therefore will retain its original color, whereas “black” means the color of the printing.  
This implies that most fonts will be printed on a black background which ascends and descends slightly more than most of the characters. Not all fonts are suited for inverse printing. Thin lines, serifs, and ornaments may be difficult to distinguish. There may also be an imbalance between the ascending and descending black background.  
The same principles apply to images. The normally invisible background may be larger than expected or be less favorably balanced. Small “white” details tend to be blurred out by the black background. Therefore, before using an inverse image, make a printout sample. INVIMAGE is revoked by a NORIMAGE statement.

**Example**

```
10 PRPOS 30,300
20 DIR 1
30 ALIGN 4
40 INVIMAGE
50 FONT "Swiss 721 BT"
60 PRTXT "Inverse printing"
70 PRINTFEED
RUN
```

## **KEY BEEP**

- Purpose** Statement for resetting the frequency and duration of the sound produced by the beeper when any of the keys on the printer keyboard are pressed.
- Syntax** KEY.BEEP<nexp1>,<nexp2>  
where:  
<nexp1> is the frequency of the sound in Hz.  
<nexp2> is the duration of the sound in periods of 0.020 seconds each.  
(max. 15,0000 = 5 minutes). Default frequency is 1200 Hz.  
Default duration is 0.030 sec.
- Remarks** This statement sets the response for all keys of the printer. To turn off the audible key response, set the frequency to a value higher than 9999.

Note that the beeper is disabled during printing. The table below illustrates the relation between frequencies and the musical scale (same as in the SOUND statement).

| Frequency in Hz |            |            |            |            |
|-----------------|------------|------------|------------|------------|
| Pitch           | 1st octave | 2nd octave | 3rd octave | 4th octave |
| C               | 131        | 262        | 523        | 1047       |
| C#/Db           | 138        | 277        | 554        | 1109       |
| D               | 147        | 294        | 587        | 1175       |
| D#/Eb           | 155        | 311        | 622        | 1245       |
| E               | 165        | 330        | 659        | 1319       |
| F               | 175        | 349        | 699        | 1397       |
| F#/Gb           | 185        | 370        | 740        | 1480       |
| G               | 196        | 392        | 784        | 1568       |
| G#/Ab           | 208        | 415        | 831        | 1662       |
| A               | 220        | 440        | 880        | 1760       |
| A#/Bb           | 233        | 466        | 933        | 1865       |
| B               | 247        | 494        | 988        | 1976       |

Note that C in the 2nd octave (262 Hz) corresponds to “middle C.”

**Example** In this example, the beeper will produce an A in the one-line octave for 1 second each time a key is pressed down.

```
10 KEY BEEP 440,50
. . . .
. . . .
```

## KEY ON/OFF

**Purpose** Statement enabling or disabling a specified key on the printer’s front panel to be used in connection with an ON KEY...GOSUB statement.

**Syntax** KEY (<nexp>) OFF | ON

where:

<nexp> is the id. number of one of the keys on the printer’s front panel.

OFF|ON disables or enables the specified key.

**Remarks** Using an ON KEY... GOSUB statement, any key (except the <Shift> key) can be assigned to make the program branch to a subroutine. The keys are enabled/disabled individually and are specified by means of their respective id. numbers in unshifted and/or shifted position. To specify a shifted key, add 100 to the unshifted id. number of the key.

The EasyCoder only has one button, with id.number 17. The button generates event 117 when released.

Please note the difference between the id. numbers of the keys and the ASCII values they produce (see KEYBMAP\$).

**Example** In this example, the ◀/F1 key (id. No. 10) is first enabled, then used for branching to a subroutine and finally disabled.

```
10 KEY (10) ON
20 ON KEY (10) GOSUB 1000
30 KEY (10) OFF
```

## **KEYMAP\$**

**Purpose** Variable returning or setting the keyboard map table.

**Syntax** **Read the map table:**

```
<svar> = KEYMAP$(<nexp>)
```

where:

<svar> returns the keyboard mapping

<nexp> is the type of string to be returned:

0 Shift or Alt key not activated (64 characters)

1 Shift key activated (64 characters)

5 Alt key activated (64 characters) (Alphanumeric keyboard on EasyCoder PX4i and PX6i only)

**Set the map table:**

```
KEYMAP$(<nexp>) = <sexp>
```

where:

<nexp> is the type of string to be remapped:

0 Shift or Alt key not activated (64 characters)

1 Shift key activated (64 characters)

5 Alt key activated (64 characters) (Alphanumeric keyboard on EasyCoder PX4i and PX6i only)

<sexp> is the string specifying the ASCII value for each key position in the selected type of string.

**Remarks** In the KEYMAP\$ statement, each key on the printer front panel has two characteristics:

- The physical location (position number) of the key. This is not the same thing as the key's Id. number (see KEY ON/OFF or ON KEY GOSUB).
- The ASCII decimal value that is produced when the key is pressed. (Compare with BREAK.)

Refer to Appendix A for illustrations of position numbers, id. numbers, and ASCII values for the various printer models and keyboard types.

In principle, each physical key can produce two or three different ASCII values: one in unshifted position, another in shifted position, and for alphanumeric keyboards, a third when the <Alt> key is pressed.

One key is appointed <Shift> key. When the <Shift> key is pressed at the same time as another key, the unshifted ASCII value of the second key is increased by 128.

You can use the KEYBMAP\$ instruction in two ways:

- Reading the keyboard mapping.

You can read how the keyboard is mapped depending on if the <Shift> or <Alt> is activated or not. The printer returns a string of ASCII values in ascending key position number. Because many keys return non-printable ASCII values (ASCII 00-31 dec.), not all are returned to the host screen or printed on a label.

- Changing the keyboard mapping.

You can change the mapping of the keyboard so a key will produce another ASCII value than before. To do that, you must create a string which specifies the ASCII value for each of all 64 unshifted, shifted, or Alt-initiated key positions in ascending order. Regardless of what the keyboard looks like, there are always 64 theoretical key positions.



**Note:** Because the EasyCoder PD41 keyboard map is only 5 bytes long, Index zero indicates the Print button, and values 1 through 4 are reserved.

Characters that cannot be produced by the host keyboard can be substituted by CHR\$ functions, where the character is specified by its ASCII decimal value according to the selected character set (see NASC statement.) The same applies to special characters. Key positions which should be disabled or are not included in the physical keyboard can be mapped as NUL, using the function CHR\$(0). Note that the position of the <Shift> key cannot be remapped.

When a key is remapped, its id. number follows the key to its new position. Most keys on the alphanumeric keyboard do not have id. numbers.

There is no default mapping for the alphanumeric keyboard, so the printer assumes that a normal keyboard is fitted and reacts accordingly. To make full use of the alphanumeric keyboard for EasyCoder PX4i and PX6i, all keys have to be mapped. For more information, see Appendix A.



**Note:** In Setup Mode, the keys have fixed positions and are not affected by any KEYBMAP\$ statement. KEYBMAP\$ only affects the keys when used outside Setup Mode.

**Examples** The following example illustrates the mapping of the keyboard for EasyCoder PF4i (unshifted keys only).

```

10 B$=CHR$(1)+STRING$(4,0)+CHR$(2)+ STRING$(4,0)+CHR$(3)
20 B$=B$+STRING$(4,0)+CHR$(4)+STRING$(4,0)+ CHR$(5)+STRING$(9,0)
30 B$=B$+CHR$(13)+CHR$(28)+CHR$(29)+CHR$(30)+ STRING$(6,0)
40 B$=B$+.147"+CHR$(0)+"0258"+CHR$(0)+CHR$(8)+"369"+CHR$(0)+(CHR$(31) +
 STRING$(8,0)
50 KEYBMAP$(0)=B$
RUN

```

## KILL

**Purpose** Statement for deleting a file, directory, or complete directory sub-trees from the printer's memory or from a CompactFlash memory card inserted in the memory card adapter.

**Syntax** KILL<sexp> [, R [, A] ]

where:

<sexp> is the file or directory to be deleted.

R recursively removes all non-system files in the specified sub-tree and then removes all empty directories in the same sub-tree.

A (optional) specifies that system files also should be removed.

**Remarks** The name of the file to be deleted must match the name given when the file was saved. The name must include the extension. If no extension was entered manually by the operator when the file was SAVED, the extension ".PRG" is added automatically.

To KILL a file residing in another directory than the current one (see CHDIR statement), you must include a reference to the directory in question when you specify the file, for example "card1:<filename>.XYZ".

KILL cannot be used for files residing in "rom:", "storage:", or "lock:". A directory cannot be removed if it contains files or directories unless the R flag is included in the KILL statement. Otherwise error 1073 ("Directory not empty") occurs.

A trailing slash character (/) may be added to directory names, but is not necessary.

The A and R flags are only applicable when removing directories, or error 1034 ("Not a directory") occurs.

Note the symmetry with FILES. FILES<sexp>,R and FILES<sexp>,R,A list files and directories that will be removed using KILL<sexp>,R and KILL<sexp>,R,A respectively.

The current directory may be removed (for example KILL CURDIR\$,R), but the root directory of a device cannot be removed.

The current directory is not changed after such a command, but is invalid. A successful CHDIR statement is necessary to restore the current directory to one that exists (CHDIR".." may not work).

**Example**

```
10 ON ERROR GOTO 1000
20 CHDIR ("/c")
30 MKDIR "DIR1" Create the directory DIR1
40 FILES
50 COPY "STDIO", "DIR1" Copy STDIO into DIR1
60 FILES "DIR1" List files in DIR1
70 KILL "DIR1" Try to remove DIR1
80 KILL "DIR1",R Remove the directory recursively
90 FILES
100 END
```

```
1000 PRINT "error number "; ERR;"in line ";ERL
1010 RESUME NEXT
RUN
```

resulting in:

Files on /c

|       |   |             |   |
|-------|---|-------------|---|
| Dir1/ | 0 | APPLICATION | 0 |
| boot/ | 0 | ADMIN/      | 0 |
| STDIO | 4 |             |   |

22210562 bytes free 4 bytes used

STDIO 4

2220032 bytes free 4 bytes used

Error number 1073 in line 70

Files on /c

|             |   |       |   |
|-------------|---|-------|---|
| APPLICATION | 0 | boot/ | 0 |
| ADMIN/      | 0 | STDIO | 4 |

2222080 bytes free 4 bytes used

## LAYOUT

**Purpose** Statement for handling of layout files.

**Syntax** LAYOUT [F, ] <sexp1>, <sexp2>, <svar> | <sexp3>, <nvar> | <sexp4>

where:

F (optional) allows use of data and error files instead of arrays

<sexp1> is the layout file.

<sexp2> is the logotype name file.

<svar> | <sexp3> is the data array (<svar>) or data file (<sexp3>).

<nvar> | <sexp4> is the error array (<nvar>) or error file (<sexp4>).

**Remarks** The next table lists the format elements of a layout file in ascending order by bytes.

### **Layout File Format**

| <b>Byte #</b> | <b>Parameter</b>                                                                                                                                                      | <b>Layout Type</b>                   | <b>Input</b>                                                                                                     | <b>Notes</b>                    |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|------------------------------------------------------------------------------------------------------------------|---------------------------------|
| 0 - 1         | Element number                                                                                                                                                        |                                      | HH                                                                                                               |                                 |
| 2             | Layout type:<br>A Logotype by name<br>B Bar code<br>C Text<br>E Bar code extended field<br>H Barfont on/off<br>J Baradjust<br>L Logotype by number<br>S Line<br>X Box |                                      | C for all layout types                                                                                           | Note 1                          |
| 3             | Direction<br>Barfont on/off:<br>0: off, 1: on<br>Security                                                                                                             | A, B, C, L, S, X<br>H<br>E           | D<br>D<br>D                                                                                                      |                                 |
| 4             | Alignment<br>Aspect height ratio                                                                                                                                      | A, B, C, L, S, X<br>E                | D<br>D                                                                                                           |                                 |
| 5-8           | X-position<br>Aspect width ratio<br>Baradjust left                                                                                                                    | A, B, C, L, S, X<br>E<br>J           | DDDD<br>D<br>DDDD                                                                                                |                                 |
| 9-12          | Y-position<br>Rows in bar code<br>Baradjust right                                                                                                                     | A, B, C, L, S, X<br>E<br>J           | DDDD<br>D<br>DDDD                                                                                                |                                 |
| 13-22         | Font name<br>Logotype name<br>Bar code name<br>Barfont name<br>Line length<br>Box width<br>Columns in bar code<br>Truncate according to code specs                    | C<br>A<br>B<br>H<br>S<br>X<br>E<br>E | C (up to 10 chars)<br>C (up to 10 chars)<br>C (up to 10 chars)<br>C (up to 10 chars)<br>DDDD<br>DDDD<br>DD<br>DD | Note 2<br>Byte 13-14<br>Byte 15 |
| 23-42         | Fixed text or alphanumeric data<br>Fixed numeric data<br>Logotype number<br>Box height<br>Line thickness                                                              | B or C<br>B<br>L<br>X<br>S           | C (up to 20 chars)<br>D (up to 20 chars)<br>DD<br>DDDD<br>DDDD                                                   |                                 |
| 43-44         | Number of chars to print from bytes 23-42                                                                                                                             | B or C                               | DD                                                                                                               |                                 |
| 45-46         | Image type (I = inverse image)<br>Bar code ratio (wide:narrow bars)                                                                                                   | A, C, L<br>B                         | C<br>DD                                                                                                          |                                 |
| 47            | Vertical magnification<br>Bar code magnification                                                                                                                      | A, C, L<br>B                         | D or C<br>D or C                                                                                                 | Note 3<br>Note 3                |
| 48            | Horizontal magnification                                                                                                                                              | A, C, L                              | D                                                                                                                |                                 |
| 49-51         | Bar code height<br>Line thickness                                                                                                                                     | B<br>X                               | DDD<br>DDD                                                                                                       |                                 |

Input: H = hex digit, D = Numeric digit, C = Alpha character



**Note 1:** The bar code extended field record (E) corresponds to the six last parameters in the BARSET statement. Must have a lower element number than the corresponding bar code record (B), which specifies the other bar code parameters.



**Note 2:** The maximum font name length in the LAYOUT statement is 10 characters. Most font names in Intermec Fingerprint are longer. A workaround method is to use font name aliases with a maximum length of 10 characters. For more information, see Chapter 4, “Character Sets and Fonts.”



**Note 3:** If a magnification of 0-9 is sufficient, enter a numeric digit. If a higher magnification than 9 is required, enter the character with the ASCII decimal number that corresponds to the desired magnification minus 48. For example, if magnification 10 is desired, enter the character: (colon, ASCII 58 dec).

### Logotype name file format #1

Record 1–*n*, 10 bytes each. No embedded spaces in name.

C1...C10 Name for logotype 1

...

...

C1...C10 Name for logotype *n*

### Logotype name file format #2

Record 1–*n*, 13 bytes each. Records sorted in ascending logotype number order.

DD Logotype number (2 digits)

C Always “:” (colon). Separator. Distinguishes format 2.

C1...C10 Name of logotype (10 characters)



**Note:** Logotype name file formats #1 and #2 are alternative.

### Data array/file format

One array position/One file line. Sorted in ascending order.

HH Element number

C1...C*n* Data

If a data element cannot be used in the layout, an error occurs. The index of the unused element and error code -1 are placed in the error array/file.

### Error array/file format

Sorted in ascending order.

|                                   |                             |
|-----------------------------------|-----------------------------|
| Array position/File line 0:       | Record number for error 1   |
| Array position/File line 1:       | Error number for error 1    |
| ...                               |                             |
| ...                               |                             |
| Array position/File line $2n-2$ : | Record number for error $n$ |
| Array position/File line $2n-1$ : | Error number for error $n$  |

To improve performance, Intermec recommends that you create the layout and logotype name files in the printer's temporary memory ("tmp:"). Once they have been created in "tmp:", they can be copied to the printer's permanent memory to avoid losing them at power off.

Do not confuse this statement with the statements LAYOUT INPUT, LAYOUT END, and LAYOUT RUN.

**Example** Note that the 10 characters available to define a font in the LAYOUT statement in most cases cannot accommodate modern outline font names. Instead, use font aliases as described in Chapter 4. In this example, the font aliases are indicated by lowercase italic typing (lines 90–120, 150).

```
10 DIM QERR%(10)
20 LAYDATA$(0)="01DAY"
30 LAYDATA$(1)="04123456789012"
40 QERR%(0)=0
50 OPEN "tmp:LOGNAME.DAT" FOR OUTPUT AS 19
60 PRINT# 19,"DIAMONDS.1";
70 CLOSE 19
80 OPEN "tmp:AYOUT.DAT" FOR OUTPUT AS 6
90 PRINT# 6,"01C11100 10 font alias 00I 11 ";
100 PRINT# 6,"01C11100 40 font alias 00 22 ";
110 PRINT# 6,"01C11100 100 font aliasWEDNES 06I 11 ";
120 PRINT# 6,"01C11100 130 font aliasSATURNUS 05I 11 ";
130 PRINT# 6,"02L11300 70 1 33 ";
140 PRINT# 6,"03S11100 210 300 3 ";
150 PRINT# 6,"04H1 font alias ";
160 PRINT# 6,"04B14100 300 EAN13 0 312 100";
170 CLOSE 6
180 LAYOUT
"tmp:AYOUT.DAT", "tmp:LOGNAME.DAT", LAYDATA$, QERR%
190 IF QERR% (1) = 0 THEN GOTO 260
200 PRINT "-ERROR- LAYOUT 1"
210 I%=0
220 IF QERR%(I%)=0 THEN GOTO 260
230 PRINT " ERROR ";QERR%(I%+1); " in record ";QERR%(I%)
240 I%=I%+2
250 GOTO 220
260 PRINTFEED
```

## LAYOUT END

**Purpose** Statement for stopping the recording of a layout description and saving the layout (Intermec Direct Protocol only).

**Syntax** LAYOUT END

**Remarks** This statement can only be used in the Intermec Direct Protocol after a layout has been recorded by means of a LAYOUT INPUT statement. After a LAYOUT END statement has been executed, no more data is added to the layout.

By default, the layout is saved in the printer's permanent memory ("c"). Alternately, to speed up execution it can be saved in the temporary memory (see LAYOUT INPUT statement). The layout can be copied and killed as with any program file.

**Example** This example illustrates how to enable Intermec Direct Protocol, specify new separators, store a layout in the printer's temporary memory, combine variable data with the layout, print a label, and disable the Direct Protocol:

```
INPUT ON .
FORMAT INPUT "#", "@", "&" .
LAYOUT INPUT "tmp:LABEL1" .
FT "Swiss 721 BT".
PP 100,250 .
PT VAR1$.
PP 100,200 .
PT VAR2$.
LAYOUT END .
LAYOUT RUN "tmp:LABEL1" .
#Line number 1&Line number 2&@ .
PF .
INPUT OFF .
```

## LAYOUT INPUT

**Purpose** Statement for starting the recording of a layout description (Intermec Direct Protocol only).

**Syntax** LAYOUT INPUT <sexp>

where <sexp> is the desired name of the layout (maximum 30 characters) including the name of the device where the layout is to be stored.

**Remarks** This statement can only be used in the Intermec Direct Protocol and starts the recording of a layout. All formatting instructions (such as PRPOS, MAG, FONT, BARFONT, BARSET, PRTXT, PRBAR, PRIMAGE, PRBOX, PRLINE, and so on) transmitted to the printer on the standard IN channel after a LAYOUT INPUT statement and before a LAYOUT END statement are included in the layout.

Layouts cannot be created in "/c" (which by default is the current directory), but must be created in the printer's temporary memory ("tmp:"), or on a CompactFlash card ("card1:"). Once a layout has been created in the temporary memory ("tmp:"), it can be copied to either "/c" or "card1:" so it will not be lost at power-off or reboot.

Variable input data to text, bar code, and image fields can be provided separately using a LAYOUT RUN statement. Such variable data are indicated in the layout by string variables VARn\$ where “n” is the number of the field in the LAYOUT RUN string of data. For example, the statement PRTXT “Hello” in the layout results in a fixed text, whereas the statement PRTXT VAR1\$ results in a variable text provided by the first field in a LAYOUT RUN string.

The layout must not contain any PRINTFEED statements and will not be saved until a LAYOUT END statement is executed.

**Example** This example illustrates how to enable Intermec Direct Protocol, specify new separators, store a layout in the printer’s temporary memory, combine variable data with the layout, print a label, and disable the Direct Protocol:

```
INPUT ON .
FORMAT INPUT "#", "@", "&" .
LAYOUT INPUT "tmp:LABEL1" .
FT "Swiss 721 BT" .
PP 100,250 .
PT VAR1$.
PP 100,200 .
PT VAR2$.
LAYOUT END .
LAYOUT RUN "tmp:LABEL1" .
#Line number 1&Line number 2@ .
PF .
INPUT OFF .
```

## LAYOUT RUN

**Purpose** Statement for providing variable input data to a predefined layout (Intermec Direct Protocol only).

**Syntax** LAYOUT RUN <sexp>

where <sexp> is the name of the layout as specified in the LAYOUT INPUT statement.

**Remarks** This instruction can only be used in the Intermec Direct Protocol and is used to select a predefined layout in a specified part of the printer’s memory (see LAYOUT INPUT statement) and provide input to string variables in the layout. Such variables are indicated by VARn\$, where “n” indicates a field in the string of data that should follow the LAYOUT RUN statement.

The string of input data should be composed according to the following syntax, where <STX> is the start-of-text separator, <CR> is the field separator and <EOT> is the end-of-text separator (see FORMAT INPUT statement):

```
<STX><input to VAR1$><CR><input to VAR2$><CR>...<input to VARn$><CR><EOT>
```

Before reverting to normal Fingerprint printing after using variable data (LAYOUT INPUT, LAYOUT END, and LAYOUT RUN), the data must be cleared using LAYOUT RUN with an empty string (LAYOUT RUN “ ”).

**Example** This example illustrates how to enable Intermec Direct Protocol, specify new separators, store a layout in the printer temporary memory, combine variable data with the layout, print a label, and disable the Direct Protocol:

```
INPUT ON .
FORMAT INPUT "#", "@", "&" .
LAYOUT INPUT "tmp:LABEL1" .
FT "Swiss 721 BT" .
PP 100,250 .
PT VAR1$.
PP 100,200 .
PT VAR2$.
LAYOUT END .
LAYOUT RUN "tmp:LABEL1" .
#Line number 1&Line number 2&@ .
PF .
INPUT OFF .
```

## LBLCOND

**Purpose** Statement for overriding the media feed setup.

**Syntax** LBLCOND<exp1>, <exp2> | <exp3>

where:

<exp1> specifies the type of action:

- 0 Override the stop adjust.
- 1 Override the start adjust.
- 2 Turn off the Label Stop Sensor/Black Mark Sensor.
- 3 Select the mode specified by <exp3>

<exp2> specifies <exp1> = 0, 1, or 2 as a number of dots.

<exp3> specifies one of the following modes:

- 0 Legacy Mode
- 1 IPL Mode
- 2 Gap Truncate Mode

Default is LBLCOND 3,2.

**Remarks** This instruction allows you to override the printer's feed-adjust setup or to temporarily disable the label stop sensor or black mark sensor:

<exp1> = 0 Temporarily sets the stop adjust to the value specified by <exp2>.

<exp1> = 1 Temporarily sets the start adjust to the value specified by <exp2>.

- |                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $<\text{nexp1}> = 2$ | Makes the label stop sensor (LSS) or black mark sensor temporarily ignore any gaps or marks detected within the length of media feed specified by $<\text{nexp2}>$ . However, the label length must be greater than the distance between the LSS and the tear bar (if not, use LBLCOND 3,xx). This allows the use of labels of such shapes that would make the LSS react prematurely, or tickets with preprint at the back of the media that would interfere with the detection of the black mark. |
| $<\text{nexp1}> = 3$ | is useful as an alternative to LBLCOND 2,xx when the length of the label or ticket is shorter than the distance between the LSS and the tear bar. It makes it possible to select one of the modes specified by $<\text{nexp3}>$ .                                                                                                                                                                                                                                                                  |

**Legacy Mode ( $<\text{nexp3}> = 0$ )**

If the print image is longer than the physical length of the label or ticket, the print image will extend into the next label until the media feed stops according to the stop adjust setup (for example, when the gap becomes aligned with the tear bar). This means that the print image may be truncated, the next label may have to be discarded, and some of the print image may coincide with a gap or slot. This mode was called “Default Mode” in earlier versions of Intermec Fingerprint.

**IPL Mode ( $<\text{nexp3}> = 1$ )**

If the print image is longer than the physical length of the label or ticket, the print image will extend into the following label(s) until the entire print image has been printed. Then the media is fed out to the next gap or mark according to the stop adjust setup. This means that the print image will not be truncated but may extend into one or more consecutive labels, and some of the print image may coincide with gaps or slots.

**Gap Truncate Mode ( $<\text{nexp3}> = 2$ )**

If the print image is longer than the physical length of the label or ticket, only the part of the print image that fits on the label or ticket will be printed and the remainder will be ignored. This means that some of the print image may not be printed at all, but the following labels will not be affected.

Verifying a start adjust or stop adjust value in the Setup Mode by pressing key number 16 (normally labeled “Enter”), or by setting the value using a setup file or setup string, evokes any LBLCOND statement for the parameter in question.

The label stop sensor is returned to normal operation by the statement:

LBLCOND 2 , 0

All current LBLCOND statements are revoked at startup or the execution of a REBOOT statement. This means that the start and stop adjust are decided by the setup and the label stop sensor works normally.

**Example** In this example, the start adjust value in the setup mode is overridden and the label stop sensor is set to ignore any gaps in the web within 20 mm (160 dots at 8 dots/mm; 240 dots at 12 dots/mm) of media feed:

```
10 LBLCOND 1,5: LBLCOND 2,160
20 FONT "Swiss 721 BT"
30 PRTXT "Hello"
40 PRINTFEED
RUN
```

## LED ON/OFF

**Purpose** Statement for controlling the dual-color “Status” indicator.

**Syntax** LED<nexp>ON | OFF | BLINK [, DATAIN]

where <nexp> specifies the color of the Status indicator:

- 0 Controls the green LED (default ON).
- 1 Controls the red LED (default OFF).

**Remarks** Intermec Fingerprint printers are equipped with three LEDs on the front panel. The center LED (normally marked Status on the keyboard overlay) can indicate multiple conditions (for example, when an error occurs, when the printer is ready, or when data is received).

Under the Direct Protocol, the Status LED starts blinking green when data is received on the standard input channel, and switches to solid green when the channel has been silent for 0.8 seconds. Under the Direct Protocol, the Status LED may also be affected by the error handler. See “[ERROR](#) on page 59” for more information.

BLINK mode means that the “Status” LED is lit for 0.4 seconds at an interval of 0.4 seconds.

If the DATA IN flag is set with the BLINK mode, reception of data on the standard input channel controls whether the Status LED blinks or is switched off.

If you set LED 0 ON and LED 1 ON, the Status LED turns solid red.



**Note:** If the printer has a console that does not include the blue Intermec Ready-to-Work Indicator, please refer to the *Intermec Fingerprint v8.10 Programmer's Reference Manual* for a description of the LED ON/OFF statement.

**Example** In this example, the Status LED will be solid red if you attempt to run the program with a raised printhead. When you lower the printhead a label feeds out and the Status LED switches to solid green.

```
10 LED 0 ON
20 LED 1 OFF
30 ON ERROR GOTO 1000
40 PRPOS 30,300
50 FONT "Swiss 721 BT"
```

```
60 PRTXT "OK! "
70 PRINTFEED
80 LED 0 ON
90 LED 1 OFF
100 END
.....
.....
1000 LED 0 OFF
1010 LED 1 ON
1020 RESUME
```

## LEFT\$

**Purpose** Returns a specified number of characters from the start (the extreme left side) of a given string.

**Syntax** LEFT\$(<sexp>, <nexp>)

where:

<sexp> is the string from which the characters will be returned.

<nexp> is the number of characters to be returned.

**Remarks** This function is the complementary function for RIGHT\$, which returns the characters starting from the end (the extreme right side) of a string. If the number of characters to be returned is greater than the number of characters in the string, then the entire string is returned. If the number of characters is set to zero, a null string is returned.

**Example 1** 10 PRINT LEFT\$("THERMAL PRINTER", 7)  
RUN

results in:

THERMAL

**Example 2** 10 A\$="THERMAL PRINTER":B\$="LABEL"
20 PRINT LEFT\$(A\$, 8);LEFT\$(B\$, 10); "S"  
RUN

results in:

THERMAL LABELS

## LEN

**Purpose** Returns the number of character positions in a string.

**Syntax** LEN(<sexp>)

where <sexp> is the string from which the number of characters will be returned.

**Remarks** The number of characters to be returned includes unprintable characters, but the quotation marks enclosing the string expression are not included.

**Examples** In this example, lines 40 and 50 illustrate two ways of using the LEN function to add up the number of characters from several string expressions.

```
10 A$="INTERMEC" (8 char.)
20 B$="THERMAL" (7 char.)
30 C$="PRINTERS" (8 char.)
40 PRINT LEN(A$+B$+C$)
50 PRINT LEN(A$)+LEN(B$)+LEN(C$)
RUN
```

resulting in:

```
23
23
```

This example demonstrates that unprintable characters (such as space characters) are included in the value returned by the LEN function:

```
PRINT LEN("INTERMEC THERMAL PRINTERS")
```

results in:

```
25
```

## LET

**Purpose** Statement for assigning the value of an expression to a variable.

**Syntax** [LET] <<nvar>=<nexp>> | <<svar>=<sexp>>

where:

<nvar> is the numeric variable to which a value will be assigned.

<nexp> is the numeric expression from which the value will be assigned to the numeric variable.

Or,

<svar> is the string variable to which the content of the string expression will be assigned.

<sexp> is the string expression from which content will be assigned to the string variable.

**Remarks** LET is retained for compatibility with previous versions of Intermec Fingerprint. The equal sign (=) is sufficient to make the assignment. Both the expression and the variable must be either string or numeric.

**Example** 10 LET A%=100 (numeric variable)

20 B%=150 (numeric variable)

30 LET C\$="INTERMEC" (string variable)

40 D\$="THERMAL PRINTERS" (string variable)

50 PRINT A%+B%,C\$+" "+D\$

RUN

resulting in:

```
250 INTERMEC THERMAL PRINTERS
```

## **LINE INPUT**

**Purpose** Statement for assigning an entire line, including punctuation marks, from the standard IN channel to a single string variable.

**Syntax** LINE.INPUT[<scon>;]<svar>

where:

<scon>; is an optional prompt plus a semicolon.

<svar> is the string variable to which the input line is assigned.

**Remarks** For information on standard I/O channel, see SETSTDIO on “[SETSTDIO](#)” on page 185.

LINE INPUT differs from INPUT in that an entire line of up to 32,767 characters is read. Possible commas appear as punctuation marks in the string instead of dividing the line into portions.

During the execution of a program, a LINE INPUT statement interrupts the execution. You can create a prompt for the host screen that notifies the operator when a program is expecting additional data to be entered. The input is terminated and the program execution resumes when a carriage return character (ASCII 13 dec.) is encountered.

The carriage return character will not be included in the input line. Note that LINE INPUT filters out any incoming ASCII 00 dec. characters (NUL).

**Example** Print your own visiting card like this:

```
10 LINE INPUT "ENTER NAME: ";A$
20 LINE INPUT "ENTER STREET: ";B$
30 LINE INPUT "ENTER CITY: ";C$
40 LINE INPUT "ENTER STATE + ZIPCODE: ";D$
50 LINE INPUT "ENTER PHONE NO: ";E$
60 FONT "Swiss 721 BT", 8
70 ALIGN 5
80 PRPOS 160,300:PRTXT A$
90 PRPOS 160,250:PRTXT B$
100 PRPOS 160,200:PRTXT C$
110 PRPOS 160,150:PRTXT D$
120 PRPOS 160,100:PRTXT "Phone: "+E$
130 PRINTFEED
RUN
```

## **LINE INPUT#**

**Purpose** Statement for assigning an entire line, including punctuation marks, from a sequential file or a device to a single string variable.

**Syntax** LINE.INPUT#<nexp>,<svar>

where:

<nexp> is the number assigned to the file when it was OPENed.

*<svar>* is the string variable to which the input line is assigned.

**Remarks** This statement differs from the INPUT# statement in that an entire line of up to 32,767 characters will be read, and possible commas in the line are included in the string as punctuation marks.

When reading from a sequential file, the lines can be read one after the other by repeating the LINE INPUT# statement using the same file reference. Once a line has been read, it cannot be read again until the file is CLOSEd and then OPENed again.

The LINE INPUT# statement is useful when the lines in a file have been broken into fields. Note that LINE INPUT# filters out any incoming ASCII 00 dec. characters (NUL).

**Example** This example assigns data from the three first lines of the file "Addresses" to the string variables A\$, B\$, and C\$ respectively:

```
.
.
.
100 OPEN "ADDRESSES" FOR INPUT AS #5
110 LINE INPUT# 5, A$
120 LINE INPUT# 5, B$
130 LINE INPUT# 5, C$
.
.
.
```

## LIST

**Purpose** Statement for listing the current program completely or partially, or listing all variables, to the standard OUT channel.

**Syntax** LIST [ [*ncon1*] [-*ncont2*] ] | ,V | ,B]

where:

*<ncon1>* is a single line, or the first line number in a range of lines.

*<ncon2>* is optionally the last line number in a range of lines.

,V lists all variables.

,B lists all breakpoints.

**Remarks** This instruction is useful after LOADING a program, or if you have changed program lines, renumbered lines, or added new lines and want to bring some order in the presentation on the screen of the host. LIST also removes unnecessary characters and adds assumed keywords. The instruction is usually given in the immediate mode (on a line without any preceding line number).

The LIST statement can be used in several different ways:

- If no line number is entered after LIST, the entire current program is listed. If the program has been written without line numbers (see IMMEDIATE ON/OFF statements), the lines are automatically numbered in 10-step increments starting with 10 (10-20-30-40-50....).

- If a single line number is entered after LIST, only the specified line is listed.
- If a line number followed by a hyphen (-) is entered after LIST, all lines from the specified line to the end of the program are listed.
- If a hyphen (-) followed by line number is entered after LIST, all lines from the start of the program through the specified line are listed.
- If two line numbers are entered after LIST, they specify the first and last lines in a range of lines to be listed.
- If LIST,V is entered, all integer variables, integer array variables, string variables, and string array variables in the printer memory are listed.
- If LIST,B is entered, all breakpoints of the Fingerprint Debugger (see DBBREAK) are printed in line number order. Line labels that have not been updated (which occurs at program execution) may be misplaced.

|                 |           |                                             |
|-----------------|-----------|---------------------------------------------|
| <b>Examples</b> | LIST      | Lists all lines in the program.             |
|                 | LIST 100  | Lists line 100 only.                        |
|                 | LIST 100- | Lists all lines from line 100 to the end.   |
|                 | LIST -500 | Lists all lines from the start through 500. |
|                 | LIST,V    | Lists all variables.                        |
|                 | LIST,B    | Lists all breakpoints.                      |

## **LISTPFSVAR**

|                |                                                                  |
|----------------|------------------------------------------------------------------|
| <b>Purpose</b> | Statement for listing variables saved at power failure.          |
| <b>Syntax</b>  | LISTPFSVAR                                                       |
| <b>Remarks</b> | Related instructions are SETPFSVAR, GETPFSVAR, and DELETEPFSVAR. |
| <b>Example</b> | LISTPFSVAR<br>results in:<br>QS\$<br>QCPS%<br>A%                 |

## **LOAD**

|                |                                                                                                                                                  |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Purpose</b> | Statement for loading a copy of a program, residing in the current directory or in another specified directory, into the printer working memory. |
| <b>Syntax</b>  | LOAD<scon><br>where <scon> is the program to be loaded into the working memory.                                                                  |

**Remarks** If the program has the extension .PRG, the name of the program can be given with or without an extension. Otherwise, the extension must be included in the name. If the program resides in another directory than the current one (see CHDIR statement), the name must also contain a reference to the directory in question.

LOAD closes any open files and deletes all program lines and variables residing in the working memory before loading the specified program. If the previous program in the working memory has not been saved, it is lost and cannot be retrieved.

While the program is loaded, a syntax check is performed. If a syntax error is detected, loading is interrupted and an error message is transmitted on the standard OUT channel.

**Examples** This example loads the program “LABEL127.PRG” from the current directory:

LOAD "LABEL127"

or

LOAD "LABEL127.PRG"

When “Ok” appears on the screen, the loading is complete. Use a LIST statement to display the program on the screen of your terminal.

You may also load a program stored in another directory than the current one, such as from the read-only memory (“/rom”) or a CompactFlash memory card (“card1:”). Start the file name by specifying the directory as in these examples:

LOAD "/rom/MKAUTO.PRG"

or

LOAD "card1:PROGRAM1.PRG"

This creates a copy, which you can list or change and then save under a new name.

## LOC

**Purpose** Returns the current position in an OPEN file, or returns the status of the buffers in an OPEN communication channel.

**Syntax** LOC(<nexp>)

where <nexp> is the number assigned to the file or communication channel when it was OPENed.

**Remarks** In a random file, LOC returns the number of the last record read by a GET statement or written by a PUT statement.

In a sequential file, the number of 128-byte blocks that have been read or written since the file was OPENed is returned.

LOC also checks the receive or transmit buffer of the specified communication channel:

- If the channel is OPENed for INPUT, the remaining number of characters (bytes) to be read from the receive buffer is returned.
- If the channel is OPENed for OUTPUT, the remaining free space (bytes) in the transmit buffer is returned.

The number of bytes includes characters that will be MAPped as NUL.

**Examples** This example closes the file “addresses” when record 100 has been read from the file:

```
10 OPEN "ADDRESSES" FOR INPUT AS #1
.....
.....
.....
200 IF LOC(1)=100 THEN CLOSE #1
.....
.....
```

This example reads the number of bytes remaining to be received from the receive buffer of “uart2:”:

```
100 OPEN "uart2:" FOR INPUT AS #2
110 A%=LOC(2)
120 PRINT A%
```

## LOF

**Purpose** Returns the length in bytes of an OPEN sequential or random file, or returns the status of the buffers in an OPEN communication channel.

**Syntax** LOF(<nexp>)

where <nexp> is the number assigned to the file or communication channel when it was OPENed.

**Remarks** LOF also checks the receive or transmit buffer of the specified communication channel:

- If a channel is OPENed for INPUT, the remaining free space (bytes) in the receive buffer is returned.
- If a channel is OPENed for OUTPUT, the remaining number of characters to be transmitted from the transmit buffer is returned.

**Examples** This example illustrates how the length of the file “Pricelist” is returned:

```
10 OPEN "PRICELIST" AS #5
20 A%=LOF(5)
30 PRINT A%
. . .
. . .
```

The second example shows how the number of free bytes in the receive buffer of communication channel “uart2:” is calculated:

```

100 OPEN "uart2:" FOR INPUT AS #2
110 A% = LOF(2)
120 PRINT A%

```

## LSET

**Purpose** Statement for placing data left-justified into a field in a random file buffer.

**Syntax** LSET<svvar>=<sexp>

where:

<svvar> is the string variable assigned to the field by a FIELD statement.

<sexp> holds the input data.

**Remarks** After OPENing a file and formatting it using a FIELD statement, you can enter data into the random file buffer using the LSET and RSET statements (RSET right-justifies the data).

The input data can only be stored in the buffer as string expressions.

Therefore, a numeric expression must be converted to string format by the use of an STR\$ function before an LSET or RSET statement is executed.

If the length of the input data is less than the length of the field, the data is left justified and the remaining number of bytes are printed as space characters.

If the length of the input data exceeds the length of the field, the input data is truncated on the right side.

**Example**

```

10 OPEN "PHONELIST" AS #8 LEN=26
20 FIELD#8,8 AS F1$, 8 AS F2$, 10 AS F3$
30 SNAME$="SMITH"
40 CNAME$="JOHN"
50 PHONE$="12345630"
60 LSET F1$=SNAME$
70 LSET F2$=CNAME$
80 RSET F3$=PHONE$
90 PUT #8,1
100 CLOSE#8
RUN
SAVE "PROGRAM 1.PRG "
NEW
10 OPEN "PHONELIST" AS #8 LEN=26
20 FIELD#8,8 AS F1$, 8 AS F2$, 10 AS F3$
30 GET #8,1
40 PRINT F1$,F2$,F3$
RUN

```

resulting in:

```
SMITH — — JOHN — — — — 12345630
```

## LTS& ON/OFF

**Purpose** Statement for enabling or disabling the label taken sensor.

**Syntax** LTS& ON|OFF

Default is LTS& OFF.

**Remarks** The label taken sensor (LTS) is a photoelectric device that can be fitted at the printer's label outfeed slot and detects if a printed label or ticket has been removed. Usually, a self-adhesive label is not fed out completely, but will remain partly stuck to the liner so it does not fall off.

Using the LTS ON statement, you can order the printer to stop the execution at the next PRINTFEED statement until the LTS no longer detects a label. Then the PRINTFEED is executed. This is most useful when printing batches of labels or tickets. As soon as a label is taken, the next one is printed and awaits handling.

LTS& OFF revokes LTS& ON.

**Example**

```
10 LTS& ON
20 FOR A%=1 TO 5
30 B$=STR$(A%)
40 FONT "Swiss 721 BT"
50 PRPOS 200,200
60 PRTXT B$
70 PRINTFEED
80 NEXT
RUN
```

## MAG

**Purpose** Statement for magnifying a font, barfont, or image up to four times with regard to height and width.

**Syntax** MAG<nexp1>,<nexp2>

where:

<nexp1> is the magnification with regard to height. Range is 1 to 4.  
Default is 1.

<nexp2> is the magnification with regard to width. Range is 1 to 4.  
Default is 1.

Reset to default by executing a PRINTFEED.

**Remarks** Magnification makes the object grow in directions away from the selected anchor point. See “[ALIGN \(AN\)](#)” on page 9.

The implementation of scalable fonts makes using the MAG statement unnecessary. Although MAG works for such fonts, printout quality is improved by using a larger font size instead of magnifying a smaller one. The MAG statement is retained for backwards compatibility with previous versions of Fingerprint.

MAG also works with images, but a larger version of an image gives better printout quality than using MAG to enlarge a small image.

Note that the MAG statement cannot be used for bar code patterns (use BARHEIGHT and BARMAG statement for that purpose).

**Example** This example illustrates how the image “GLOBE.1” is printed both with its original size and magnified 4 times. Note the jagged edges of the curves in the enlarged image.

```

10 ALIGN 2
20 PRPOS 300,50
30 FONT "Swiss 721 BT"
40 PRTXT "Normal Size"
50 PRPOS 300,125
60 PRIMAGE "GLOBE.1"
70 PRPOS 300,300
80 PRTXT "Enlarged 4X"
90 PRPOS 300,375
100 MAG 4,4
110 PRIMAGE "GLOBE.1"
120 PRINTFEED
RUN

```

## MAKEASSOC

**Purpose** Statement for creating an association.

**Syntax** MAKEASSOC <sexp1>, <sexp2>, <sexp3>

where:

<sexp1> specifies the name of the association to be created (case-sensitive).

<sexp2> contains an argument list of parameter tuples according to the convention in <sexp3>.

<sexp3> should always be “HTTP” (case sensitive).

**Remarks** HTTP implies that the argument list in <sexp2> is encoded in “x-www-urlencoded.”

**Example** This example shows how a string, including three string names associated with three start values, is defined and one (time) will be changed:

```

10 QUERYSTRING$ =
"time=UNKNOWN&label=321&desc=DEF"
20 MAKEASSOC "QARRAY", QUERYSTRING$, "HTTP"
30 QTIME$ = GETASSOC$("QARRAY", "time")
40 QLABELS% = VAL(GETASSOC$("QARRAY", "label"))
50 QDESC$ = GETASSOC$("QARRAY", "desc")
60 PRINT "time="; QTIME$, "LABEL="; QLABELS%,
"DESCRIPTION="; QDESC$
70 SETASSOC "QARRAY", "time", time$
80 PRINT "time="; GETASSOC$("QARRAY", "time")
RUN

```

resulting in:

```
time=UNKNOWN LABEL=321 DESCRIPTION=DEF
time=153355
```

## MAP

**Purpose** Statement for changing the ASCII value of a character when received on the standard IN channel, or optionally on another specified communication channel.

**Syntax** MAP [<nexp1>,] <nexp2>, <nexp3>

where:

<nexp1> (optional) specifies a communication channel:

- 0 “console:”
- 1 “uart1:”
- 2 “uart2:”
- 3 “uart3:”
- 4 “centronics:”
- 5 “net1:”
- 6 “usb1:”
- 7 “uart4:”
- 8 “uart5:”

Default is Standard IN channel.

<nexp2> is the original ASCII decimal value.

<nexp3> is the new ASCII decimal value after mapping.

**Remarks** This statement modifies a character set (see NASC and NASCD statements) or filters out undesired characters. For example, to print a “Q” (ASCII 81 dec.) as the letter “Z” (ASCII 90 dec.) instead, enter the MAP statement as MAP 81,90.

The mapping interprets any ASCII 81 dec. value received on the standard IN channel as ASCII 90 dec., so that when you press “Q” on the keyboard of the host, the character “Z” is printed. However, pressing “Z” still produces a “Z” because that character has not been remapped.

To reset the mapping performed above, map the character back to its original ASCII value (MAP 81,81).

When the printer receives a character, it is processed with regard to possible MAP statements before it “enters” the Intermec Fingerprint firmware. That allows you to filter out undesired control characters, which may confuse the Intermec Fingerprint firmware, such as by mapping them as NUL (ASCII 0 decimal).

After processing, the selected character set (see NASC and NASCD statements) controls how characters are printed or displayed. If none of the character sets meets your demands completely, use MAP statements to modify the set that comes closest. Note that MAP statements are processed before any COMSET or ON KEY..GOSUB strings are checked.

NASC and NASCD statements are processed last.

Do not map any characters to ASCII values occupied by characters used in Fingerprint instructions, such as keywords, operators, %, \$, #, and certain punctuation marks. Mapping is reset to normal at power-up or reboot.

**Examples** You can check what characters the host produces by using a simple program. Pressing different keys on the host should produce the corresponding characters both on the label and on the screen of the host. If not, try another character set (see NASC). This example assumes that the keyboard produces ASCII 81 dec. and ASCII 90 dec. when you press the Q and Z keys respectively. Should any unexpected characters be printed on the labels or the screen, check the manuals of the host for information on what ASCII values are produced by the various keys and how the screen presents various ASCII values received from the printer.

```
10 FONT "Swiss 721 BT"
20 PRPOS 30,100
30 INPUT "Enter character";A$
40 PRTXT A$
50 PRINTFEED
```

By adding a MAP statement in line 5, you can test what happens. In this case we remap the character Q to be printed as Z. After printing, we map the character Q back to its original position.

```
5 MAP 81,90
10 FONT "Swiss 721 BT"
20 PRPOS 30,100
30 INPUT "Enter character";A$
40 PRTXT A$
50 PRINTFEED
60 MAP 81,81
```

Assume that a device connected to “uart2:” produces strings that always start with the control character STX (ASCII 2 decimal). STX can be filtered out by mapping it as NUL (ASCII 0 decimal):

```
10 MAP 2,2,0
```

Should “uart2:” be appointed the standard IN channel (see SETSTDIO), the first parameter can be omitted from the example above:

```
10 MAP 2,0
```

## MERGE



**Intermec recommends that you back up the current program before issuing a MERGE statement.**

**Purpose** Statement for merging a program in the printer’s current directory, or in another specified directory, with the program currently residing in the printer’s working memory.

**Syntax** MERGE<*scon*>

where <*scon*> is the name of the program to be merged with the program currently residing in the printer's working memory.

**Remarks** MERGE creates a copy of a program stored in the current directory (see CHDIR statement), or optionally in another directory, and blends its lines into the program currently residing in the printer's working memory.

If there are lines with the same numbers in both programs, the lines in the program currently residing in the working memory will be replaced by the corresponding lines in the MERGED program. This also applies to programs written without line numbers, since they will automatically be assigned hidden line numbers (10-20-30... etc.) at the execution of the IMMEDIATE ON statement.

In order to avoid overwriting any lines, you may SAVE a program without line numbers using a SAVE <*scon*>, L statement. When MERGED, it will be appended to the current program and assigned line numbers that start with the number of the last line of the current program plus 10.

MERGE makes it possible to store blocks of frequently used program instructions and include them in new programs. The printer's ROM memory contains a number of useful programs which can be MERGED into programs of your own creation.



**Be careful not to include any MERGE statement as a part of a program, or else the execution will stop after the MERGE statement has been executed.**

The EXECUTE statement offers an alternative method for combining Fingerprint programs.

**Examples** In this example, the program "XYZ.PRG" is merged with the current program. If there are identical line numbers in both programs, the lines from "XYZ.PRG" replace those in the current program.

```
MERGE "XYZ . PRG" (from current directory)
MERGE "/c/XYZ . PRG" (from permanent memory)
MERGE "tmp :XYZ . PRG" (from temporary memory)
MERGE "card1 :XYZ . PRG" (from memory card)
```

## **MIBVAR&**

**Purpose** Statement to read or write MIB variables for the Simple Network Management Protocol (SNMP).

**Syntax** MIBVAR& (<*nexp*>)

where <*nexp*> is the number of the variable to access. Range is 0 to 9.

**Remarks** SNMP MIB variables are normally used for network management tasks. These variables can be set from the outside through SNMP, and are available to Fingerprint programs through the MIBVAR& command. The SNMP variables must be of string type (maximum 128 characters) and are placed under the SNMP node “enterprises.1963.20.15.15.21.5.1.1., indexes 0-9.” The public community is read-only whereas the “pass” community is read-write. There are no read or write restrictions from Fingerprint.

Related instruction is ON MIBVAR& GOSUB.

**Example**

```
MIBVAR&(1) = "ON"
A$ = MIBVAR&(1)
PRINT A$
```

resulting in:

ON

## MID\$

**Purpose** Function returning a specified part of a string.

**Syntax** MID\$(<sexp>, <nexp1> [, <nexp2>] )

where:

<sexp> is the original string from which a specified part is to be returned.

<nexp1> specifies which character position in the original string is to be the first character in the part to be returned.

If this value is less than or equal to zero, then Error 44 (“Parameter out of range”) occurs.

If this value exceeds the length of the original string, an empty string is returned, but no error condition occurs.

[,<nexp2>] (optional) specifies the number of characters to be returned. If omitted, all characters from the start position specified by <nexp1> to the end of the string will be returned. If this value is less than zero, then error 44 (“Parameter out of range”) occurs.

**Remarks** If the value of <nexp1> does not exceed the length of the original string, but the sum of <nexp1> and <nexp2> exceeds the length of the original string, the remainder of the original string is returned.

**Example 1**

```
10 A$=MID$("INTERMEC PRINTERS", 6, 3)
20 PRINT A$
RUN
```

resulting in:

MEC

**Example 2**

```
10 A$="INTERMEC PRINTERS"
20 B%=10
30 C%=7
40 D$=MID$(A$,B%,C%)
50 PRINT D$
RUN
```

resulting in:

```
PRINTER
```

## MKDIR

**Purpose** Statement for creating a directory.

**Syntax** MKDIR<*sexp*>

where <*sexp*> specifies the directory to be created.

**Remarks** <*sexp*> can end with a slash (/) character, but / is not required. Only the device /c (or "c:") supports creating directories.

**Example**

```
NEW
MKDIR "DIR1"SAVE "DIR1/PROGRAM.PRG
FILES "/c/DIR1"
```

results in:

```
FILES on /c/DIR1
PROGRAM.PRG 2
2220032 bytes free 2 bytes used
```

## NAME DATE\$

**Purpose** Statement for formatting the month parameter in return strings of DATE\$("F") and DATEADD\$(...,"F").

**Syntax** NAME DATE\$ <*nexp*>, <*sexp*>

where:

<*nexp*> is the month number (1-12).

<*sexp*> is the desired name of the month.

**Remarks** This statement allows you to assign names to the different months in any form and language you like. The names will be returned instead of the corresponding numbers in connection with DATE\$("F") and DATEADD\$("F") instructions, provided that a FORMAT DATE\$ statement has been executed.

The number of characters assigned to represent months in the FORMAT DATE\$ statement decides how much of the names, as specified in the NAME DATE\$ statement, will be returned. The names are truncated from the left, as in this example:

```
FORMAT DATE$ "YY.MMM:DD"
NAME DATE$ 1, "JANUARY"
PRINT DATE$ ("F")
```

resulting in:

03.ARY.06

Usually, it is best to restrict the month parameter in the FORMAT DATE\$ statement to 2 or 3 characters (MM or MMM) and enter the names of the months in the NAME DATE\$ statement accordingly.

**Example** This example shows how to make the printer return dates in accordance with British standard:

```

10 DATE$="070115"
20 NAME DATE$ 1, "JAN"
30 NAME DATE$ 2, "FEB"
40 NAME DATE$ 3, "MAR"
50 NAME DATE$ 4, "APR"
60 NAME DATE$ 5, "MAY"
70 NAME DATE$ 6, "JUN"
80 NAME DATE$ 7, "JUL"
.
.
.
140 FORMAT DATE$ "MMM DD, YYYY"
150 PRINT DATE$ ("F")
RUN

```

resulting in:

JAN 15, 2007

## NAME WEEKDAY\$

**Purpose** Statement for formatting the day parameter in return strings of WEEKDAY\$.

**Syntax** NAME WEEKDAY\$ <nexp>, <sexp>

where:

<nexp> is the number of the weekday according to the WEEKDAY\$ function syntax (Monday = 1... Sunday = 7).

<sexp> is the name of the weekday. Default is the full English name in lowercase characters (monday, tuesday, etc.).

**Remarks** This statement allows you to assign names to the different weekdays in any form and language you like. The names are returned instead of the corresponding numbers in connection with WEEKDAY\$ function.

**Example** This example shows how to make the printer return the name of the weekday as an English 3-letter abbreviation:

```

10 FORMAT DATE$ ", MM/DD/YY"
20 DATE$="071201"
30 NAME WEEKDAY$ 1, "Mon"
40 NAME WEEKDAY$ 2, "Tue"
50 NAME WEEKDAY$ 3, "Wed"
60 NAME WEEKDAY$ 4, "Thu"
70 NAME WEEKDAY$ 5, "Fri"
80 NAME WEEKDAY$ 6, "Sat"
90 NAME WEEKDAY$ 7, "Sun"

```

```
100 PRINT WEEKDAY$ (DATE$) + DATE$ ("F")
RUN
```

resulting in:

Fri, 12/01/07

## NASC

**Purpose** Statement for selecting a single-byte character set, alternatively the multibyte character set UTF-8.

**Syntax** NASC<nexp> | <sexp>

where:

<nexp> is the reference number of a character set as described in the next table:

| Reference # | Character Set              | Reference # | Character Set                         |
|-------------|----------------------------|-------------|---------------------------------------|
| 1 (default) | Roman 8                    | -2          | ANSI<br>(same as 1252)                |
| 8           | UTF-8                      | 850         | MS-DOS Latin 1                        |
| 33          | French                     | 851         | MS-DOS Greek 1                        |
| 34          | Spanish                    | 852         | MS-DOS Latin 2                        |
| 39          | Italian                    | 855         | MS-DOS Cyrillic                       |
| 44          | English (UK)               | 857         | MS-DOS Turkish                        |
| 46          | Swedish                    | 1250        | Windows Latin 2<br>(Central Europe)   |
| 47          | Norwegian                  | 1251        | Windows Cyrillic<br>(Slavic)          |
| 49          | German                     | 1252        | Windows Latin 1<br>(ANSI, same as -2) |
| 81          | Japanese Latin<br>(romaji) | 1253        | Windows Greek                         |
| 351         | Portuguese                 | 1254        | Windows Latin 5<br>(Turkish)          |
| -1          | PCMAP                      | 1257        | Windows Baltic Rim                    |

<sexp> is either:  
“UTF-8” = UTF-8  
“file name” = NSC file character set.

**Remarks** For complete character set tables, see Chapter 4, “Character Sets and Fonts.”

By default, after processing any MAP statements, Fingerprint prints and, when applicable, displays all characters according to the Roman 8 character set. However, Fingerprint includes a number of other character sets useful for displaying information in other languages or countries, or for adapting the printer to the operating system of the host.

Thus, an ASCII code received by the printer may print or display a different character depending on the selected character set.

If none of the character sets available contains the desired character(s), use a MAP statement to reMAP the character set that comes closest to your needs. Note that MAP statements are processed before NASC statements.

You can also specify the multi-byte character set UTF-8. For more information on using UTF-8, see “[About the UTF-8 Character Set](#)” on page 271.

A NASC statement has these consequences:

- Font: The font used is the one last specified by the FONT command. If none has been specified, the default (Swiss 721 BT) is used.
- Text printing: Text on labels is printed according to the selected character set. However, parts of the label that have already been processed and stored in the print buffer (before the NASC statement is executed) are not affected. This allows for multiple character sets in a single label, which could be used to create multi-lingual labels.
- LCD Display: New messages in the display are affected by a NASC statement. However, a message that is already displayed is not updated automatically. The display can show all printable characters. In the Setup Mode, all characters are mapped according to US-ASCII standard.
- Communication: Data transmitted via any of the communication channels is not affected, since the data is defined as ASCII values and not as alphanumeric characters.

The active character set of the receiving unit determines the graphic presentation of the input data (for example, the screen of the host).

- Bar Code Printing: The pattern of the bars reflects the ASCII values of the input data and is not affected by a NASC statement. The bar code interpretation (the human readable characters below the bar pattern) is affected by a NASC statement. However, the interpretation of bar codes already processed and stored in the print buffer is not affected.

**Example** This example selects the Italian character set, prints the character corresponding to 123 dec. in that set, changes the set to Swedish, and prints the character corresponding to 123 dec.:

```
10 NASC 39
20 PRTXT CHR$ (123)
30 NASC 46
40 PRTXT CHR$ (123)
50 PRINTFEED
```

results in:

ää

## NASCD

**Purpose** Statement for selecting a double-byte character set, typically the multibyte character set UTF-8.

**Syntax** `NASCD <sexp>`

where `<sexp>` is the name of the character set or the file containing the character map. Default is “ ” (disables double-byte interpretation).

**Remarks** When a double-byte character set has been selected, the firmware usually treats all characters from ASCII 161 dec. to ASCII 254 dec (ASCII A1-FE hex) as the first part of a two-byte character. The next character byte received specifies the second part. However, the selected double-byte character set may specify some other ASCII value as the breaking point between single- and double-byte character sets.

There are various ways to produce double-byte characters from the host PC keyboard. By selecting the proper character set using a NASCD statement, the typed-in ASCII values are translated to the corresponding Unicode values so the desired glyph is printed. A font must have been selected using the FONTD command in order to print according to the selected double-byte character set.

UTF-8 character mapping is pre-installed in the firmware, and is selected with the command NASCD “UTF-8” (can also be selected with NASC). For more information, see [“About the UTF-8 Character Set” on page 271](#). Other double-byte character set tables and fonts are available from Intermec on special request.

**Example** The following text contains both single- and double-byte fonts. The double-byte font and its character set are stored in a Font Install Card:

```
10 NASC 46
20 FONT "Swiss 721 BT", 24, 10
30 FONTD "Song"
40 NASCD "card1:GB2312.NCD"
50 PRTXT CHR$(65);CHR$(164);CHR$(180)
60 PRINTFEED
RUN
```

This program yields a printed text line that starts with the Latin character A (ASCII 65 dec.) followed by the character in the Song font that corresponds to the address 164+180 dec. in the character set “GB2312.NCD.”

## NEW

**Purpose** Statement for clearing the printer’s working memory in order to allow a new program to be created.

**Syntax** `NEW`

**Remarks** The NEW statement deletes the program currently residing in the printer working memory, closes all files, and clears all variables and breakpoints.

If the current program has not been saved (see SAVE statement), it is lost and cannot be restored.

In the Intermec Direct Protocol, all counters are removed when a NEW statement is executed.

Note that clearing the printer working memory does not imply that the host screen is cleared too. The lines of the previous program remain onscreen until replaced by new lines.

## NORIMAGE (NI)

**Purpose** Statement for returning to normal printing after an INVIMAGE statement has been issued.

**Syntax** NORIMAGE | NI

**Remarks** Normal image is the default type of printing. Text and images are printed in black-on-white.

Using an INVIMAGE statement, the printing of text and images can be inverted. Such inverse printing is discontinued for all PRTXT and PRIMAGE statements that follow a NORIMAGE statement.

**Example** In this example, the first line is printed in inverted fashion and the second line in the normal fashion:

```

10 PRPOS 30,300
20 ALIGN 4
30 INVIMAGE
40 FONT "Swiss 721 BT"
50 PRTXT "INVERSE PRINTING"
60 PRPOS 30, 200
70 NORIMAGE
80 PRTXT "NORMAL PRINTING"
90 PRINTFEED
RUN

```

## ON BREAK GOSUB

**Purpose** Statement for branching to a subroutine, when break interrupt instruction is received.

**Syntax** ON.BREAK<nexp>GOSUB<ncon>|<line label>

where:

<nexp> is one of the following communication channels:

- 0 “console:”
- 1 “uart1:”
- 2 “uart2:”
- 3 “uart3:”
- 4 “centronics:”
- 7 “uart4:”
- 8 “uart5:”

*<ncon>|<line label>* is the number or label of the program line to be branched to.

**Remarks** This statement is closely related to BREAK and BREAK ON/OFF. When break interrupt is enabled (see BREAK ON) and the operator issues a break interrupt instruction (see BREAK), the execution of the currently running program is interrupted and branched to a specified line in a subroutine. Break from the console is not supported by the EasyCoder PD41.

**Examples** In this example, the printer emits a special signal when a break interrupt is issued from the printer keyboard:

```
10 ON BREAK 0 GOSUB 1000
20 GOTO 20
. . .
1000 FOR A%=1 TO 3
1010 SOUND 440,50
1020 SOUND 349,50
1030 NEXT A%
1040 END
```

The same example without line numbers looks like this:

```
IMMEDIATE OFF
ON BREAK 0 GOSUB QQQ
WWW: GOTO WWW
. .
QQQ: FOR A%=1 TO 3
SOUND 440,50
SOUND 349,50
NEXT A%
END
IMMEDIATE ON
```

## ON COMSET GOSUB

**Purpose** Statement for branching to a subroutine when the background reception of data on the specified communication channel is interrupted.

**Syntax** ON.COMSET<nexp1>GOSUB<nexp2>|<line label>

where:

*<nexp1>* is the communication channel:

- 0 “console:”
- 1 “uart1:”
- 2 “uart2:”
- 3 “uart3:”
- 4 “centronics:”
- 6 “usb1:”
- 7 “uart4:”
- 8 “uart5:”

*<nexp2>/<line label>* is number or label of the program line to be branched to.

**Remarks** This statement is closely related to COMSET, COMSTAT, COMSET ON, COMSET OFF, COM ERROR ON/OFF, and COMBUF\$. Use this statement to branch to a subroutine when:

- the end character is received.
- an attention string is received.
- the maximum number of characters is received.

These three parameters are set for the specified communication channel by a COMSET statement.

**Examples** In this example, the program branches to a subroutine to read the buffer of the communication channel:

```

1 REM Exit program with #STOP&
10 COMSET1,"#", "&","ZYX", "=", 50
20 ON COMSET 1 GOSUB 2000
30 COMSET 1 ON
40 IF A$ <> "STOP" THEN GOTO 40
50 COMSET 1 OFF
.....
.....
1000 END
2000 A$= COMBUF$(1)
2010 PRINT A$
2020 COMSET 1 ON
2030 RETURN

```

The same example written without line numbers looks like this:

```

IMMEDIATE OFF
REM Exit program with #STOP&
COMSET1,"#", "&","ZYX", "=", 50
ON COMSET 1 GOSUB QQQ
COMSET 1 ON
WWW: IF A$ <> "STOP" THEN GOTO WWW
COMSET 1 OFF
.....
.....
END
QQQ: A$=COMBUF$(1)
PRINT A$
COMSET 1 ON
RETURN
IMMEDIATE ON

```

## ON ERROR GOTO

**Purpose** Statement for branching to an error-handling subroutine when an error occurs.

**Syntax** ON.ERROR.GOTO<ncon>|<line label>

where <ncon> or <line label> is the number or label of the line to which the program should branch when an error condition occurs.

**Remarks** If any kind of error condition occurs after this statement has been encountered, the standard error-trapping routine is ignored and the program branches to the specified line, which should be the first line in an error handling subroutine.

If the line number is 0, the standard error-trapping routine is enabled and no error-branching within the current program is executed.

**Examples** If you try to run this example with the printhead raised (or if any other error occurs), a warning signal sounds and the error LED turns on.

```
10 LED 0 ON:LED 1 OFF
20 ON ERROR GOTO 1000
30 FONT "Swiss 721 BT"
40 PRTXT "HELLO"
50 PRINTFEED
60 END
.
.
.
1000 LED 0 OFF:LED 1 ON
1010 FOR A%=1 TO 3
1020 SOUND 440,50
1030 SOUND 359,50
1040 NEXT A%
1050 RESUME NEXT
```

The same example written without line numbers looks like this:

```
IMMEDIATE OFF
LED 0 ON:LED 1 OFF
ON ERROR GOTO QQQ
FONT "Swiss 721 BT"
PRTXT "HELLO"
PRINTFEED
END
.
.
.
QQQ: LED 0 OFF:LED 1 ON
FOR A%=1 TO 3
SOUND 440,50
SOUND 359,50
NEXT A%
RESUME NEXT
IMMEDIATE ON
```

## ON GOSUB

**Purpose** Statement for conditional branching to one of several subroutines.

**Syntax** ON<exp>GOSUB<con>|<line label>[,<con>|<line label>...]

where:

<exp> is a numeric expression that determines which line the program should branch to.

<con>|<line label> is the number or label of the line, or list of lines, to which the program should branch.

**Remarks** ON GOSUB is closely related to ON GOTO. The numeric expression may result in any positive value. The expression is truncated to an integer value before the statement is executed. If the resulting value is negative, 0, or larger than the number of subroutines, the statement is ignored.

The value of the numeric expression determines which of the subroutines the program should branch to. For example, if the value of the numeric expression is 2, the program will branch to the second subroutine in the list.

**Examples** In this example, different text strings appear on the screen depending on which key (1-3) you press on the host keyboard.

```
10 INPUT "PRESS KEY 1-3 ", A%
20 ON A% GOSUB 1000,2000,3000
30 END
1000 PRINT "You have pressed key 1"
1010 RETURN
2000 PRINT "You have pressed key 2"
2010 RETURN
3000 PRINT "You have pressed key 3"
3010 RETURN
```

The same example written without line numbers would look like this:

```
IMMEDIATE OFF
INPUT "PRESS KEY 1-3 ", A%
ON A% GOSUB QQQ,WWW,ZZZ
END
QQQ: PRINT "You have pressed key 1"
RETURN
WWW: PRINT "You have pressed key 2"
RETURN
ZZZ: PRINT "You have pressed key 3"
RETURN
IMMEDIATE ON
```

## ON GOTO

**Purpose** Statement for conditional branching to one of several lines.

**Syntax** ON<exp>GOTO<con>|<line label>[,<con>|<line label>...]

where:

<exp> is a numeric expression that determines which line the program should branch to.

<con>|<line label> is the number or label of the line, or list of lines, to which the program should branch.

**Remarks** This statement is closely related to the ON GOSUB statement. The numeric expression may result in any positive value. The expression is truncated to an integer value before the statement is executed. If the resulting value is negative, 0, or larger than the number of lines, the statement is ignored.

<exp> determines which of the lines the program should branch to. For example, if <exp> = 2, the program branches to the second line in the list.

**Examples** In this example, different text is printed on the screen depending on which of the keys (1-3) you press on the host keyboard.

```
10 INPUT "PRESS KEY 1-3 ", A%
20 ON A% GOTO 1000,2000,3000
30 END
1000 PRINT "You have pressed key 1"
1010 GOTO 30
2000 PRINT "You have pressed key 2"
2010 GOTO 30
3000 PRINT "You have pressed key 3"
3010 GOTO 30
```

The same example written without line numbers looks like this:

```
IMMEDIATE OFF
INPUT "PRESS KEY 1-3 ", A%
ON A% GOSUB QQQ,WWW,ZZZ
YYY: END
QQQ: PRINT "You have pressed key 1"
GOTO YYY
WWW: PRINT "You have pressed key 2"
GOTO YYY
ZZZ: PRINT "You have pressed key 3"
GOTO YYY
IMMEDIATE ON
```

## ON HTTP GOTO

**Purpose** Statement for branching to a subroutine when a request for an application CGI is received.

**Syntax** `ON__HTTP__GOTO<ncon>|<line label>`

where `<ncon>` or `<line label>` is the number or label of the line to which the program will branch when the CGI request is received.

**Remarks** This statement is used in connection with EasyLAN 100i and defines a Fingerprint subroutine that handles the CGI-request. Setting the line number or line label to 0 disables the handler.

When a request for an application CGI is received, the current execution point is pushed on to the stack and the execution commences in the handler with stdin and stdout redirected to or from the Web browser.

A related instruction is RESUME HTTP.

## ON KEY GOSUB

**Purpose** Statement for branching to a subroutine when a specified key on the printer's front panel is activated.

**Syntax** `ON__KEY(<nexp>) GOSUB<ncon>|<line label>`

where:

*<nexp>* is the ID number of one of the keys on the printer front panel.

*<ncon>/<line label>* is the number or label of the line to which the program will branch when the specified key is pressed down.

**Remarks** Except for the EasyCoder PD41, Fingerprint-compatible printers are fitted with a membrane-switch keyboard. Each key can be enabled individually using its ID number in a KEY ON statement. Then the key can be assigned, alone or in combination with the <Shift> key, to make the program branch to a subroutine using an ON KEY... GOSUB statement.

The <Shift> key adds 100 to the unshifted ID number of each key. See Appendix A for illustrations of the ID numbers for the printer keyboards.

The EasyCoder PD41 works in the same way, but has only a single key with ID number 17 when pressed, and 117 when released.

Note the difference between the ID numbers of the keys and the ASCII values they are able to produce (see for example BREAK).

BREAK takes precedence over any ON KEY statement, provided that break interrupt is not disabled for the “console:” by a BREAK 0 OFF statement.

**Examples** This example illustrates how activating the ◀/F1 key (ID = 10) makes the program branch to a subroutine containing the PRINTFEED statement. Note line 30 where the execution waits for the key to be pressed.

```

10 ON KEY (10) GOSUB 1000
20 KEY (10) ON
30 GOTO 30
.....
.....
.....
1000 FONT "Swiss 721 BT"
1010 PRPOS 30,100
1020 PRTXT "HELLO"
1030 PRINTFEED
1040 END
RUN

```

The same example can be written without line numbers this way:

```

IMMEDIATE OFF
ON KEY (10) GOSUB QQQ
KEY (10) ON
WWW: GOTO WWW
.....
.....
.....
QQQ: FONT "Swiss 721 BT"
PRPOS 30,100
PRTXT "HELLO"
PRINTFEED
END
IMMEDIATE ON
RUN

```

## **ON/OFF LINE**

**Purpose** Statement controlling the SELECT signal on the “centronics:” communication channel.

**Syntax** ON | OFF  $\rightarrow$  LINE<*nexp*>

where <*nexp*> specifies the communication channel:

- 4    “centronics:”
- 6    “usb1:”

**Remarks** Pin 13 in the Centronics/IEEE 1284 interface connector contains the SELECT signal:

- ON LINE 4 sets the SELECT signal high.
- OFF LINE 4 sets the SELECT signal low.

If no ON/OFF LINE statement is issued, the SELECT signal is high (the Centronics channel will be ON LINE). ON LINE/OFF LINE for the serial channel “usb1:” is implemented according to USB Device Class for Printing Devices v1.09, January 2000.

**Example** In this example, the “centronics:” communication channel is disabled, a new setup is performed on the printer by means of a setup file, and the channel is enabled:

```
10 OFF LINE 4
20 SETUP "New Setup.SYS"
30 ON LINE 4
. . .
. . .
. . .
```

## **ON MIBVAR& GOSUB**

**Purpose** Statement for branching to a subroutine when a specified SNMP MIB variable is changed.

**Syntax** ON.MIBVAR&(<*nexp*>) GOSUB<*ncon*> | <*line label*>

where:

<*nexp*> is the number of a MIBVAR& variable. Range is 0 to 9.

<*ncon*>/<*line label*> is the number or label of the line to which the program will branch when the specified MIB variable is modified.

**Remarks** This statement makes the Fingerprint program branch to a subroutine whenever the specified MIB variable is modified. This value is then maintained during the entire subroutine execution. External changes during the subroutine execution cause a new subroutine call at return. Related instruction is MIBVAR&.

**Example** This example prints the value of MIBVAR&(1) on a label every time its value is changed. Note that the program is idle (on line 20) as long as the MIBVAR& variable remains unchanged.

```

10 ON MIBVAR&(1) GOSUB 1000
20 GOTO 20
.....
.....
.....
1000 FONT "Swiss 721 BT"
1010 PRPOS 30,100
1020 A$ = MIBVAR&(1)
1030 PRTXT A$
1040 PRINTFEED
1050 RETURN
RUN

```

## OPEN

**Purpose** Opens a file or device (for example, a network connection), or creates a new file—for input, output, or append, allocating a buffer, and specifies the access mode.

**Syntax** OPEN<*sexp*> [FOR<*INPUT* | *OUTPUT* | *APPEND*>] AS [#]<*nexp1*> [*LEN*=<*nexp2*>]

where:

<*sexp*> is the file or device to be opened, or the file to be created. File names must not contain a colon character (:). A network connection can be specified.

# (optional) Indicates that whatever follows is a number.

<*nexp1*> is a designation number for the OPENed file or device.

<*nexp2*> (optional) is the length of the record in bytes. Default is 128.

**Remarks** An OPEN statement must be executed before a file or device can be used for input, output, and/or append. A maximum of 25 files and/or devices can be open at the same time.

### Sequential Access Mode

The access mode can optionally be specified as sequential INPUT, OUTPUT, or APPEND:

- INPUT: Sequential input from the file/device, replacing existing data. Existing files/devices only.
- OUTPUT: Sequential output to the file/device, replacing existing data.
- APPEND: Sequential output to the file/device, where new data will be appended without replacing existing data.

### **Random Access Mode**

If no access mode is specified in the statement, the file/device is opened for both input and output (RANDOM access mode). FIELD, LSET, RSET, PUT, and GET can only be used on records in files OPENed in the RANDOM access mode.

Please refer to the DEVICES statement for information on which devices can be opened for the different modes of access.

Use FILE to get lists of the files stored in the printer memory.

### **Network Connections**

Special syntax as follows can be used for the *<sexp>* expression that opens the Net1 device:

```
OPEN "net1:<host>[:port][,<timeout>]" AS #<nexp>
```

where:

*<host>* is an IP address, alternatively a DNS name.

*[:port]* is the port number. Range is 0 to 65535. Default is 9100.

*<timeout>* is the timeout value expressed in ticks (0.01 sec). Range is 10 to 6000. Default is 4500 (45 sec).

Only one Net1 connection can be opened at a time. Once the device is open, information can be sent or received according to the corresponding protocol.

**Examples** This example opens and writes to a file:

```
10 OPEN "TEST.TXT" FOR OUTPUT AS 1
20 PRINT#1:"AAAA"
30 PRINT#1,"!234"
40 CLOSE#1;
```

This example opens the file and prints it line by line:

```
10 OPEN "TEST.TXT" FOR INPUT AS #2
20 INPUT#2:A$
30 INPUT#3,B$
40 CLOSE#1;
50 PRINT A$
60 PRINT B$
RUN
```

You can allow sequential output to the printer's display (except for the EasyCoder PD41) using the OPEN statement this way. The method is the same as for files:

```
10 OPEN "console:" FOR OUTPUT AS #1
20 PRINT#1:PRINT#1
30 PRINT#1, "GONE TO LUNCH"
40 PRINT#1, "BACK SOON";
RUN
```

The text appears as:

**GONE TO LUNCH  
BACK SOON**

This example opens the file “PRICELIST” for random access with the reference number #8 and a record length of 254 bytes:

```
10 OPEN "PRICELIST" AS #8 LEN=254
```

The following example opens the Net1 device to connect to the NIST Internet Time Service (ITS) web site:

```
10 OPEN "net1:time.nist.gov:13" AS 1
20 PRINT #1, "time?"
30 QTICKS% = TICKS
40 WHILE TICKS-QTICKS% < 100
50 WEND
60 LINE INPUT #1, A$
70 PRINT A$
80 CLOSE 1
```

In the example above, row 10 specifies socket port 13. Rows 30-40 create a delay to allow the result to be received as A\$ within the specified time of 100 ticks.

## OPTIMIZE BATCH ON/OFF

**Purpose** Statement for enabling/disabling optimizing for batch printing.

**Syntax** OPTIMIZE "BATCH" ON | OFF

where ON|OFF enables or disables optimization. Default is Disabled(OFF). In Direct Protocol, default is Enabled(ON).

**Remarks** This facility is intended to speed up batch printing, the uninterrupted printing of large numbers of identical or similar labels.

The program execution will not wait for the printing of the label to be completed, but proceeds executing the next label image into the other of the two image buffers as soon as possible.

OPTIMIZE BATCH is automatically enabled (ON) during a batch (PRINTFEED more than one) if the following conditions are fulfilled:

1 LTS& OFF (default)

2 CUT OFF (default)

In Intermec Direct Protocol, the default value is enabled (ON).

**Examples** Run these two examples and note the differences in the printer's performance:

```
10 OPTIMIZE "BATCH" ON
20 FOR I%=1 TO 10
30 PRTXT I%
40 PRINT "Before printfeed"
50 PRINTFEED
60 PRINT "After printfeed"
70 NEXT
RUN

10 OPTIMIZE "BATCH" OFF
20 FOR I%=1 TO 10
```

```
30 PRTXT I%
40 PRINT "Before printfeed"
50 PRINTFEED
60 PRINT "After printfeed"
70 NEXT
RUN
```

## **PORTIN**

**Purpose** Reads the status of a port on a Serial/Industrial Interface Board.

**Syntax** PORTIN(<nexp>)

where <nexp> is the number of the port to be read:

IN ports (optical): 101-108 (301-308)  
OUT ports (relay): 201-204 (401-404)  
OUT ports (optical): 221-228 (421-428)

**Remarks** This function works with the Serial/Industrial Interface Board and can read the status of 8 IN ports with optocouplers, 8 OUT ports with optocouplers, and 4 OUT ports with relays. For information on how to set the OUT ports, see the PORTOUT statement.

A current can be led through an optocoupler in each IN port:

- If the current is on, the PORTIN function returns the value -1 (true).
- If the current is off, the PORTIN function returns the value 0 (false).

This feature allows the execution of Fingerprint to be controlled by various types of external sensors or non-digital switches.

The status of the OUT ports, as set by PORTOUT statements, can also be read by PORTIN functions.

Some printers (like the EasyCoder PM4i) can carry two Serial/Industrial Interface boards. In this case, the ports on the inner board (that is, the board closest to the CPU board) are specified by the low numbers (101-108, 201-204, and 221-228) while the ports on the outer board are specified by the high numbers (301-308, 401-404, and 421-428).

See the documentation of the Serial/Industrial Interface Board for more information.

**Example** The status of IN port 101 on a Serial/Industrial Interface Board decides when a label is to be printed. The printing is held until the current is switched off:

```
10 FONT "Swiss 721 BT"
20 PRTXT "POWER IS OFF"
30 IF PORTIN (101) THEN GOTO 30
40 PRINTFEED
50 END
```

## PORTOUT ON/OFF

**Purpose** Statement for setting one of four relay ports or one of eight optical ports on a Serial/Industrial Interface Board to either on or off.

**Syntax** PORTOUT (<nexp>) ON|OFF

where <nexp> is the number of the port to be set:

OUT ports (relay): 201-204 (401-404)

OUT ports (optical): 221-228 (421-428)

**Remarks** This statement works with the Serial/Industrial Interface Board and is able to control 8 IN ports with optocouplers, 8 OUT ports with optocouplers, and 4 OUT ports with relays. For information on how to read the status of the various ports, see the PORTIN function.

This feature allows the Fingerprint to control various external devices like gates, lamps, or conveyor belts.

Some printers (such as the EasyCoder PM4i) can carry two Serial/Industrial Interface boards. In this case, the ports on the inner board (that is, the board closest to the CPU board) are specified by the low numbers (201-204 and 221-228) while the ports on the outer board are specified by the high numbers (401-404 and 421-428).

See the documentation of the Serial/Industrial Interface Board for more information.

**Example** The relay of OUT port 201 on a Serial/Industrial Interface Board is Opened and then Closed like this:

```
.
.
.
1000 PORTOUT (201) ON
.
.
.
2000 PORTOUT (201) OFF
.
.
.
```

## PRBAR(PB)

**Purpose** Statement for providing input data to a bar code.

**Syntax** PRBAR | PB<<sexp>> | <nexp>>

where <<sexp>>|<nexp>> is the input data to the bar code generator.

**Remarks** The bar code must be defined by BARSET, BARTYPE, BARRATIO, BARHEIGHT, BARMAG, BARFONT, and/or BARFONT ON/OFF statements, or by the corresponding default values. Make sure that the type of input data (numeric or string) and the number of characters agree with the specification for the selected bar code type. For more information, see Chapter 5, ‘Bar Codes.’

**Example** Two different bar codes, one with numeric input data and one with string input data, can be generated this way. The input data could also have been entered in the form of variables:

```
10 BARFONT "Swiss 721 BT", 8 ON
20 PRPOS 50,400
30 ALIGN 7
40 BARSET "INT2OF5",2,1,3,120
50 PRBAR 45673
60 PRPOS 50,200
70 BARSET "CODE39",3,1,2,100
80 PRBAR "ABC"
90 PRINTFEED
RUN
```

## **PRBOX (PX)**

**Purpose** Statement for creating a box containing a single text line or a frame of multiple hyphenated text lines.

**Syntax** PRBOX | PX<nexp1>, <nexp2>, <nexp3> [, <sexp1> [, <nexp4> [, <nexp5> [, <sexp2> [, <sexp3>]]]]]

where:

<nexp1> is the box height in dots. Range is 1 to 6000.

<nexp2> is the box width in dots. Range is 1 to 6000.

<nexp3> is the line weight in dots. Range is 0 to 6000.

<sexp1> is the framed text to be written inside the box. Maximum of 20 lines and 300 characters per line. Single-byte fonts only.

<nexp4> is the horizontal distance between inner edge of the box line and the text frame. Range is -100 to 100 dots. Default is 0.

<nexp5> is the vertical distance between the inner edge of the box line and text frame, and between each line of text in the frame. Range is -100 to 100 dots. Default is the same value as <nexp4>.

<sexp2> is a line delimiter (maximum 9 characters), which replaces the default delimiter string CHR\$(10) or CHR\$(13). Each time this delimiter is encountered in the text string (<sexp1>), the rest of the text wraps to the next line.

<sexp3> is a control string for hyphen delimiter and replacement.

**Remarks** This statement creates a rectangular white box surrounded by a line with a certain thickness, or specifies a text frame that can contain up to 20 lines of hyphenated text. These two purposes can be combined so a text frame is surrounded by a black box.

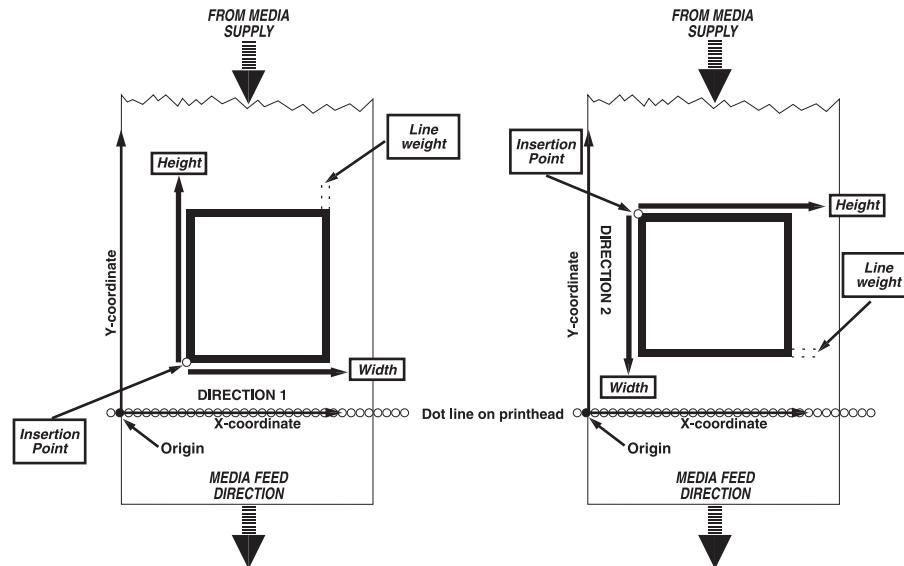
## Creating a Simple Box

To create a simple box, you only need to specify the first three parameters: height, width, and line weight (thickness). The box is drawn with its anchor point (see ALIGN) at the insertion point, as specified by the nearest preceding PRPOS statement. A box can be aligned left, right, or center along its baseline.

The print direction specifies how the box is rotated in relation to its anchor point.

The line weight (thickness) grows inward from the anchor point. The heavier the line, the less white area inside the box. Thus, it is possible to create a black area using a box with very heavy lines. For a simple box with no text field, the line weight must be  $> 0$ . The white area inside a box can be used for printing. Boxes, lines, and text may cross (also see XORMODE ON/OFF).

The next illustration shows how the height and width of the box are defined for different print directions.



## Creating a Multiline Text Field

The PRBOX statement can also create an area in which a field of wrapped and hyphenated text is printed. There is no need to specify each line of text separately as with the PRTXT statement.

The text field can be framed by the box (line weight  $> 0$ ), or the box can be invisible (line weight = 0). The maximum number of characters on each line is 300 and the maximum number of lines is 20.

The position of the text frame inside the box is affected by the direction (see DIR statement), the alignment (see ALIGN statement), and by two parameters in the PRBOX statement (`<nexp4>` and `<nexp5>`).

The direction rotates the box with its text field around the anchor point as specified by the alignment. The alignment specifies the anchor point of the box itself as left-, right-, or center-aligned (see ALIGN), and at the same time also determines how the field will be aligned inside the box (9 possible positions) and if the text lines will be left, right, or center justified.

In the following description, horizontal and vertical should be understood in relation to how the text is printed. That means that in directions 2 and 4, horizontal and vertical have opposite meanings than in directions 1 and 3.

The horizontal distance between the inner edge of the box line and the borders of the text field is specified by *<nexp4>*:

- For ALIGN 1, 4, or 7, *<nexp4>* determines the distance between the inner edge of the left side box line and the left-hand edge of the text field.
- For ALIGN 3, 6, or 9, *<nexp4>* determines the distance between the inner edge of the right side box line and the right-hand edge of the text field.
- For ALIGN 2, 5, or 8, *<nexp4>* is not necessary.

The vertical distance between the inner edge of the box line and the borders of the text field as well as the empty vertical space between the character cells of two adjacent lines (line spacing) is specified by *<nexp5>*:

- For ALIGN 1, 2, or 3, *<nexp5>* determines the distance between the inner edge of the bottom box line and the bottom edge of the text field as well as line spacing.
- For ALIGN 7, 8, or 9, *<nexp5>* determines the distance between the inner edge of the right side box line and the right-hand edge of the text field as well as line spacing.
- For ALIGN 4, 5, or 6, *<nexp5>* determines line spacing.

See the next illustration for examples of how the alignment affects the location of multi-line text.

If the text in *<sexp1>* is entered as a continuous string of characters without any spaces, linefeeds, or carriage returns, the text wraps to the next line when there is no room left for any more characters on a line.

If any combination of a carriage return (CR = ASCII 13 dec.) and a linefeed (LF = ASCII 10 dec.) is encountered, the remaining text wraps once to the next line.

Space characters (ASCII 32 dec.) also initiate a line wrap. If there is more than one space character, the wrapping is at the last space character that fits into the line in question.

You can replace the default line delimiters (CR, LF, and CR/LF) with another line delimiter specified in a string of up to 9 characters (*<sexp2>*). This delimiter is not printed, even if it is a printable character. Each time the delimiter is encountered, the text wraps to a new line.

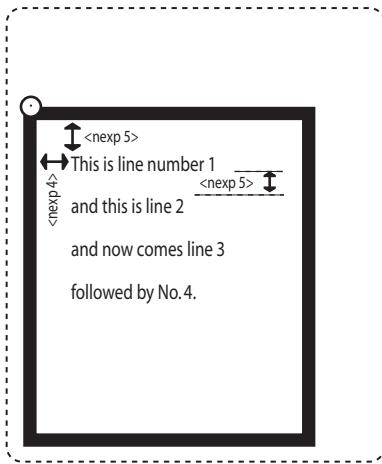
## **Hyphenation Support**

In *<sexp3>* you can modify the way hyphenation is performed using a special syntax described later in this section. You can put “invisible” hyphen delimiters in the text string at suitable wrap-around positions. The default delimiter is a hyphen sign (ASCII 45 dec.). You can use a string of up to nine characters instead, but be careful so the string is not confused with the text.

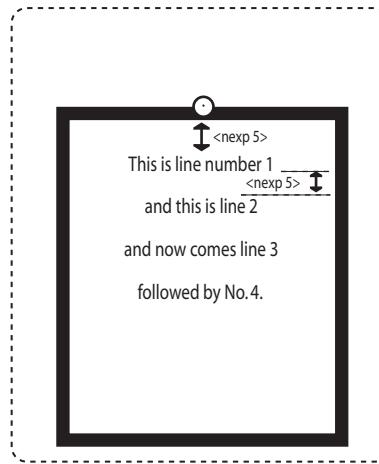
If a wrap-around is performed, the corresponding hyphen delimiter is printed as a hyphen sign (ASCII 45 dec.) by default, whereas hyphen delimiters not used for wrap-around will not be printed.

To print characters other than hyphens, you can specify a string of hyphen replacement characters. It is possible to use a string of up to 9 characters, but the shorter the string the less likely that a line will wrap outside the box.

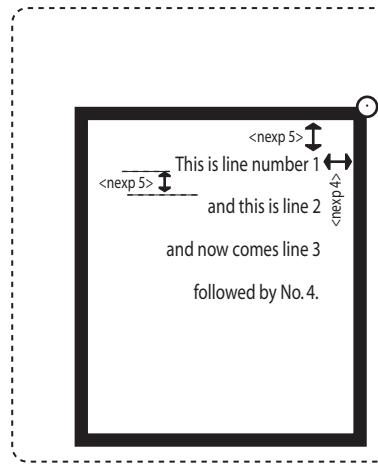
If you have a text string with long words and have not inserted all necessary line delimiters, a line-wrap may occur unexpectedly. You can optionally specify a hyphen delimiter for this case as well. Default is no delimiter.



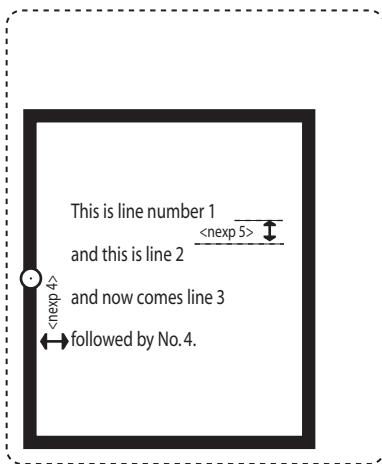
ALIGN 7



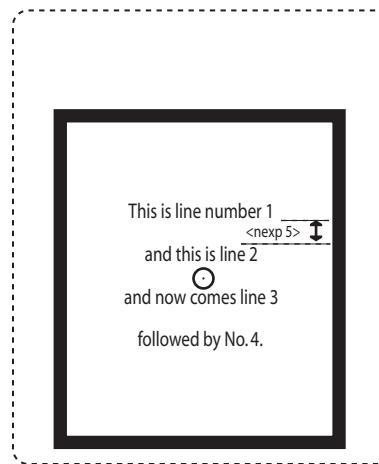
ALIGN 8



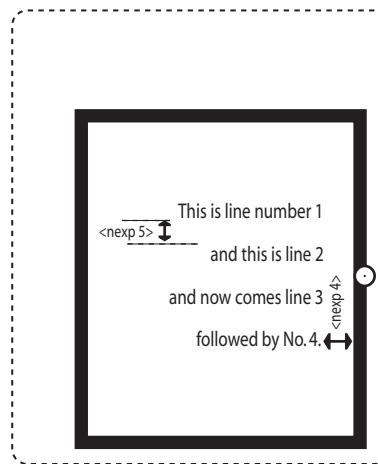
ALIGN 9



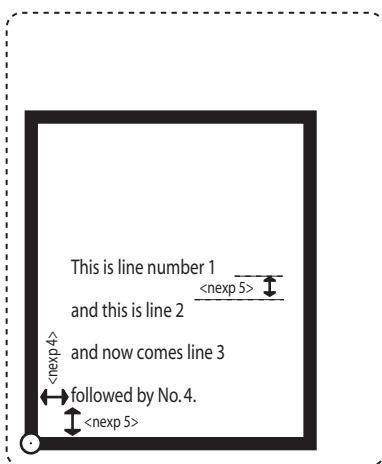
ALIGN 4



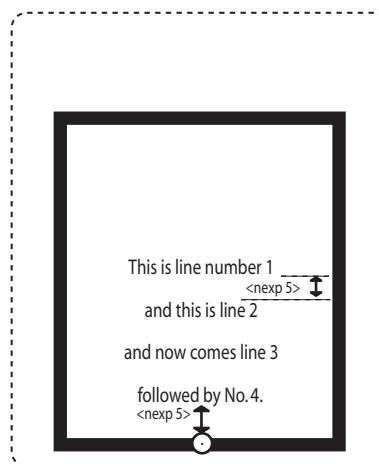
ALIGN 5



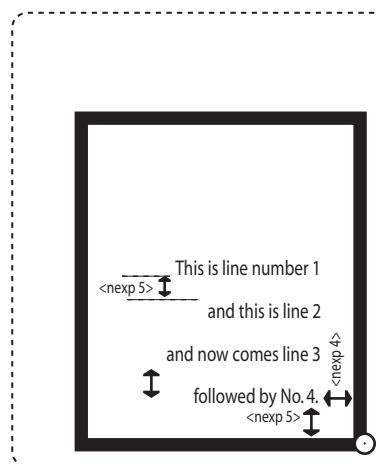
ALIGN 6



ALIGN 1



ALIGN 2



ALIGN 3

Specify the parameter *<sexp3>* in PRBOX using the following syntax:

*<sexp3>=<sexp3a> [space<sexp3b> [space<sexp3c>] ]*

where:

*<sexp3a>* is a soft hyphen delimiter. If the text does not have enough room on one line, the rest of the text is wrapped from the last space or from the position marked by the soft hyphen delimiter.

Exception: Two adjacent soft hyphen delimiters revoke each other. Default is a normal hyphen (-). Maximum length is 9 characters.

*space* is a string delimiter with the value CHR\$(32).

*<sexp3b>* is one or more characters that are printed at the end of a line which has been hyphenated according to a hyphen delimiter (see *<sexp3a>*). Default is a normal hyphen (-). Maximum length is 9 characters, but fewer is preferred.

*<sexp3c>* is a string of hyphen extension characters, used on single words too long to be printed on one line and have no hyphen delimiter specified. The hyphen extension character is printed at the right end of the line and the remainder of the word is printed on the next line. Default is no character. Maximum length is 9 characters.

If no *<sexp3>* is specified, the rules for hyphen delimiter and replacement are the same as for printing hyphens in text. Two adjacent hyphens are printed as one.

**Examples** This example draws a rectangle without any text:

```
10 PRPOS 50,50
20 PRBOX 200,400,5
30 PRINTFEED
RUN
```

This program illustrates a multi-line text field with line wrap, where “&S” is the soft hyphen delimiter:

```
10 DIR 1
20 ALIGN 8
30 R$="Hyphen&Sated words will be divid&Sed
into syll&Sbles."
40 NL$="NEWLINE"
50 S$="&S&Special Cases and EXTRAORDINARY long words."
60 T$=R$+NL$+S$
70 PRPOS 300,300
80 PRBOX 700,500,20,T$,25,1,NL$, "&S - +"
90 PRINTFEED
RUN
```

## PRBUF

**Purpose** Statement for receiving and printing bitmap image data using the PRBUF protocol.

**Syntax** PRBUF<nexp1>[,<nexp2>]<new line><image data>

where:

*<nexp1>* is the number of bytes of the image in PRBUF protocol.

*<nexp2>* (optional) is a timeout between characters in TICKS (0.01 sec). Default is ~12.7 sec/character.

*<new line>* is any combination of CR, CR/LF, or LF.

*<image data>* is the image according to the PRBUF protocol.

**Remarks** This statement is useful for receiving and printing bitmap images from, for example, a Windows printer driver. It is more effective and requires less memory than using a STORE IMAGE...PRIMAGE sequence. The bitmap image is printed directly and is not saved anywhere in the printer memory after the image buffer has been cleared.

At the PRBUF statement, the printer waits for image data to be received on the standard IN channel. Because PRBUF only works with binary transfers, XON/XOFF must be disabled. Optionally, you can set a timeout between characters (default is 12.7 seconds). When the specified number of bytes according to the PRBUF protocol has been received, the image data is processed directly into the printer's image buffer and printed without requiring any more Fingerprint instructions.

PRBUF does not work if *<nexp1>* bytes cannot be allocated. If memory is low, it is possible to download the bitmap image in two or more blocks. The field settings (alignment, clipping, direction, xor mode, inverse image, magnification, x-position, and y-position) are handled by the current protocol, but the basic rule is that x- and y-positions, field clipping, and xor mode are handled and the other attributes are ignored.

If PRPOS x,y, then the real print position is PRPOS x,y+1. For more information, see “[PRBUF Protocol](#)” on page 241.

The *<new line>* is not part of the statement, but any combination of carriage return (ASCII 13 dec,) and/or linefeed (ASCII 10 dec.) is allowed without interfering with the PRBUF protocol.

**Example** This example shows how the printer is instructed to receive and print 1,424 bytes of image data according to the PRBUF protocol:

```
PRBUF 1424 .
<binary image data>
```

## PRIMAGE (PM)

**Purpose** Statement for selecting an image stored in the printer's memory.

**Syntax** PRIMAGE | PM*<sexp>*

where *<sexp>* is the full name of the desired image including its extension.

**Remarks** An image is positioned according to the preceding PRPOS, DIR, and ALIGN statements. It can be magnified by means of a MAG statement. For the best printout quality, create and download a larger version of the image instead of magnifying a smaller one. All images provided by Intermec have an extension which indicates for which directions the image is intended:

- Extension .1 indicates print directions 1 and 3.
- Extension .2 indicates print directions 2 and 4.

Even if Fingerprint firmware does not require such an extension, Intermec strongly recommends that you follow the same convention when creating your own images so that selecting the correct image is easier.

**Example** This example illustrates printing a label containing an image printed “upside down”:

```
10 PRPOS 200,200
20 DIR 3
30 ALIGN 5
40 PRIMAGE "GLOBE.1"
50 PRINTFEED
RUN
```

## PRINT (?)

**Purpose** Statement for printing data to the standard OUT channel.

**Syntax** PRINT | ? [ <<expr> | <expr>> [ <, | ; ><<expr> | <expr>> . . . ] [ ; ] ]

where <<expr> | <expr>> are string or numeric expressions, which will be printed to the standard OUT channel.

**Remarks** If no expressions are specified after the PRINT statement, it yields a blank line. If one or more expressions are listed, the expressions are processed and the resulting values are presented on the standard OUT channel (for example, on the screen of the host). The shorthand form of PRINT is a question mark.

Do not confuse the PRINT statement with the PRINTFEED statement.

Each line is divided into zones of 10 character positions each. These zones can be used for positioning the values:

- A comma sign (,) between the expressions causes the next value to be printed at the beginning of the next zone.
- A semicolon sign (;) between the expressions causes the next value to be printed immediately after the last value.
- A plus sign (+) between two string expressions causes the next value to be printed immediately after the last value. Plus signs cannot be used between numeric expressions.
- If the list of expressions is terminated by a semicolon, the next PRINT statement is added on the same line. Otherwise, a carriage return is performed at the end of the line. If the printed line is wider than the

screen, the software automatically wraps to a new line and continues printing.

Printed numbers are always followed by a space character. Printed negative numbers are preceded by a minus sign.

**Example**

```
10 LET X% = 10
20 LET A$ = "A"
30 PRINT X%; X%+1, X%+5; X%-25
40 PRINT A$+A$; A$, A$
50 PRINT X%;
60 ? "PIECES"
RUN
```

resulting in:

```
10 11 15 -15
AAA A
10 PIECES
```

## **PRINT KEY ON/OFF**

**Purpose** Statement for enabling or disabling printing of a label by pressing the Print key.

**Syntax** PRINT KEY ON|OFF

Default is PRINT KEY OFF.

**Remarks** In Immediate Mode and Intermec Direct Protocol, the **Print** key can be enabled to issue printing commands corresponding to PRINTFEED statements. This implies that each time the **Print** key is pressed, one single label, ticket, tag, or portion of continuous stock is printed and fed out.

Note that this statement cannot be entered in Programming Mode (use KEY ON and ON KEY GOSUB statements instead).

**Example** This example shows how the **Print** key is enabled and a label is printed in Direct Protocol. Abbreviated instructions are used whenever available:

```
INPUT ON ↴
PRINT KEY ON ↴
PP 100,100 ↴
FT "Swiss 721 BT" ↴
PT "TEST LABEL" ↴
```

[Press the **Print** key]

```
INPUT OFF ↴
```

## **PRINT#**

**Purpose** Statement for printing data to a specified OPENed device or sequential file.

**Syntax** PRINT#<nexp1> [, <<nexp2> | <sexp1>> [<, | ; ><<nexp3> | <sexp2>>... ] [ ; ] ]

where:

*<nexp1>* is the number assigned to the file or device when it was OPENed.

*<<nexp2-n>|<sexp1-n>>* are the string or numeric expressions which will be printed to the specified file or device.

**Remarks** Expressions can be separated by commas or semicolons according to the same rules as for the PRINT statement. The expressions must be separated properly so they can be read back when needed, or be presented correctly on the printer display.

PRINT# can only be used to print to sequential files, not to random files. When sending data to the printer display ("console:"), PRINT# works the same way as PRINT does on the standard OUT channel. For example, the display can be cleared by sending PRINT#<nexp> twice (see line 20 in the example below).

**Example** The display on the printer can show two lines of 16 characters each. Before sending any text, the device must be OPENed (line 10) and both lines on the display must be cleared (line 20). Note the trailing semicolon on line 40:

```
10 OPEN "console:" FOR OUTPUT AS #1
20 PRINT# 1:PRINT# 1
30 PRINT# 1,"OUT OF LABELS"
40 PRINT# 1,"PLEASE RELOAD! ";
50 CLOSE# 1
RUN
```

Since the last line was appended by a semicolon, there is no carriage return and the text appears in the printer display as:

**OUT OF LABELS  
PLEASE RELOAD!**

An alternative method is to send all the data to the display in a single PRINT# statement. Characters 1 through 16 are displayed on the upper line and characters 17 through 33 are displayed on the lower line, although character 17 is ignored.

The double-headed arrows in line 30 represent space characters. Note the trailing semicolon in line 30.

```
10 OPEN "console:" FOR OUTPUT AS #1
20 PRINT# 1: PRINT# 1
30 PRINT# 1,"OUT↔OF↔LABELS↔↔↔↔PLEASE↔RELOAD! ";
40 CLOSE# 1
RUN
```

## PRINTFEED (PF)

**Purpose** Statement for printing and feeding out one or a specified number of labels, tickets, tags, or portions of strip, depending on the printer setup.

**Syntax** PRINTFEED|PF [*<nexp1>*] | [-1,*<nexp2>*]

where:

- <exp1> specifies number of copies to be printed.
- 1,<exp2> specifies that <exp2> number of identical copies of the last printed label should be reprinted. This function cannot be used with Intermec Direct Protocol.

**Remarks** Each time a PRINTFEED statement without any appending value is executed, one new label, ticket, tag, or portion of continuous stock is printed.

Optionally, the PRINTFEED statement can be appended by a numeric expression specifying the number of copies to print. In the Intermec Direct Protocol, possible counter, time, and date values are updated between copies printed using a predefined layout. Do not include any PRINTFEED statements in layouts in the Intermec Direct Protocol.

If the number of copies is >1 and LTS& and CUT are disabled (= LTS& OFF and CUT OFF), the BATCH optimizing strategy is automatically enabled, corresponding to an OPTIMIZE BATCH ON statement.

When these conditions are no longer fulfilled, BATCH optimizing strategy is automatically disabled, corresponding to an OPTIMIZE BATCH OFF statement.

It is also possible to reprint a specified number of copies of the last printed label, for example after an out-of-media condition (also see PRSTAT). The execution of a PRINTFEED statement resets the following statements to their default values:

|         |            |                |
|---------|------------|----------------|
| ALIGN   | BARRATIO   | INVIMAGE       |
| BARFONT | MAG        | BARFONT ON/OFF |
| BARSET  | PRPOS      | BARHEIGHT      |
| DIR     | XORMODE ON | BARMAG         |
| FONT    | FONTD      |                |

Fields defined by statements executed before the PRINTFEED statement are not affected. Note that, when using a PRINTFEED in a loop, all formatting parameters are reset to default each time the PRINTFEED statement is executed and must therefore be included inside the loop.

The length of media to be fed out at execution of a PRINTFEED statement is determined by the choice of media type in the printer setup (label stock with gaps, tickets with gaps, fixed length strip, or variable length strip) and globally by the start and stop adjustment setup (positive or negative). See the printer user's guide for more information. The length of media to be fed out can be further modified by an additional positive or negative FORMFEED statement, either before or after the PRINTFEED statement.

**Examples** This example prints a single label with one line of text:

```
10 FONT "Swiss 721 BT"
20 PRTXT "Hello!"
```

```
30 PRINTFEED
RUN
```

This example prints five identical labels with one line of text:

```
10 FONT "Swiss 721 BT"
20 PRTXT "Hello!"
30 PRINTFEED 5
RUN
```

This example prints five labels using a FOR...NEXT loop. Note that formatting parameters are placed inside the loop:

```
10 FOR A%=1 TO 5
20 FONT "Swiss 721 BT"
30 PRPOS 200, 100
40 DIR 3
50 ALIGN 5
60 PRTXT "Hello!"
70 PRINTFEED
80 NEXT A%
RUN
```

This example prints five labels in the Intermec Direct Protocol, illustrating how the TICKS value is updated between labels, provided a predefined layout is used (1 TICK = 0.01 sec):

```
INPUT ON .
FORMAT INPUT "#", "@", "&" .
LAYOUT INPUT "tmp:LABEL1" .
FT "Swiss 721 BT" .
PP 100,100 .
PT TICKS .
PP 100,200 .
PT VAR1$.
LAYOUT END .
LAYOUT RUN "tmp:LABEL1" .
#See how time flies&@ .
PF 5 .
INPUT OFF .
```

## PRINTONE

**Purpose** Statement for printing characters, specified by their ASCII values, to the standard OUT channel.

**Syntax** PRINTONE<nexp>[<, | ;><nexp>...] [ ; ]

where <nexp> is the ASCII decimal value of a character, which will be printed to the standard OUT channel.

**Remarks** When certain characters cannot be produced by the host computer, they can be substituted by the corresponding ASCII decimal values using the PRINTONE statement. The characters are printed, according to the currently selected character set (see NASC statement), to the standard OUT channel (usually the screen of the host).

PRINTONE is very similar to the PRINT statement and the use of commas and semicolons follows the same rules.

**Example** PRINTONE 80;82;73;67;69;58,36;52;57;46;57;53  
results in:  
PRICE: \$49.95

## **PRINTONE#**

**Purpose** Statement for printing characters specified by their ASCII values to a device or sequential file.

**Syntax** PRINTONE#<nexp1>[,<nexp2>[<, | ;><nexp3>...][;]]  
where:

<nexp1> is the number assigned to the file or device when it was OPENed.

<nexp2-n> is the ASCII decimal value of the character to be printed to the specified file or device.

**Remarks** This statement is useful when the host cannot produce certain characters. The ASCII values entered produce characters according to the currently selected character set (see the NASC statement for more information). The ASCII values can be separated by commas or semicolons according to the same rules as for the PRINT# statement.

PRINTONE# can only be used to print to sequential files, not to random files. When sending data to the printer display, PRINTONE# works in a way similar to PRINT#. The display can be cleared by sending PRINT#<nexp> twice (see line 20 in the example below).

**Example** The printer display is able to show two lines of 16 characters each. Before sending any text, the device must be OPENed and the display cleared. Note the trailing semicolon sign on line 40.

```
10 OPEN "console:" FOR OUTPUT AS #1
20 PRINT# 1:PRINT# 1
30 PRINTONE# 1,80;82;69;83;83
40 PRINTONE# 1,69;78;84;69;82;
50 CLOSE #1
RUN
```

Since the last line was appended by a semicolon, there is no carriage return. The text appears on the printer display as:

```
PRESS
ENTER
```

## **PRLINE (PL)**

**Purpose** Statement for creating a line.

**Syntax** PRLINE | PL<nexp1>,<nexp2>  
where:

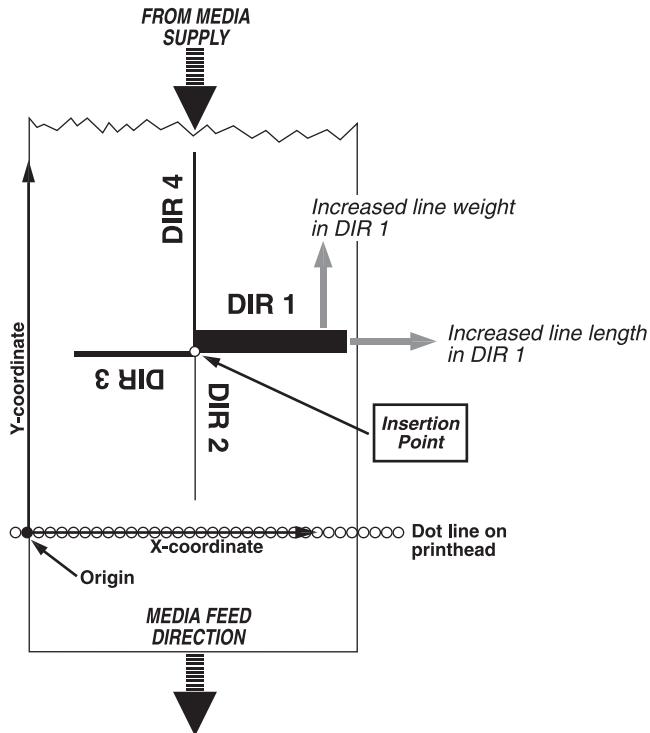
<nexp1> is the length of the line in dots. Maximum is 6000.

*<nexp2>* is the line weight in dots. Maximum is 6000.

**Remarks** The line is drawn from the insertion point and away according to the nearest preceding DIR and ALIGN statements (that is, the line runs in parallel with any text printed in the selected direction).

A line can be ALIGNED left, right or center. The anchor points are situated at the bottom of the line, which means that as the line weight increases, the line grows upward in relation to the selected direction.

In the illustration below, all lines are aligned left. Lines may cross (see XORMODE ON/OFF statement).



**Example** This example draws a 2.5 cm (1 in) long and 10 dots thick line across the media in an 8 dots/mm printer:

```
10 PRPOS 50,100
20 PRLINE 200,10
30 PRINTFEED
RUN
```

## PRPOS (PP)

**Purpose** Statement for specifying the insertion point for a line of text, a bar code, an image, a box, or a line.

**Syntax** PRPOS | PP*<nexp1>*,*<nexp2>*

where:

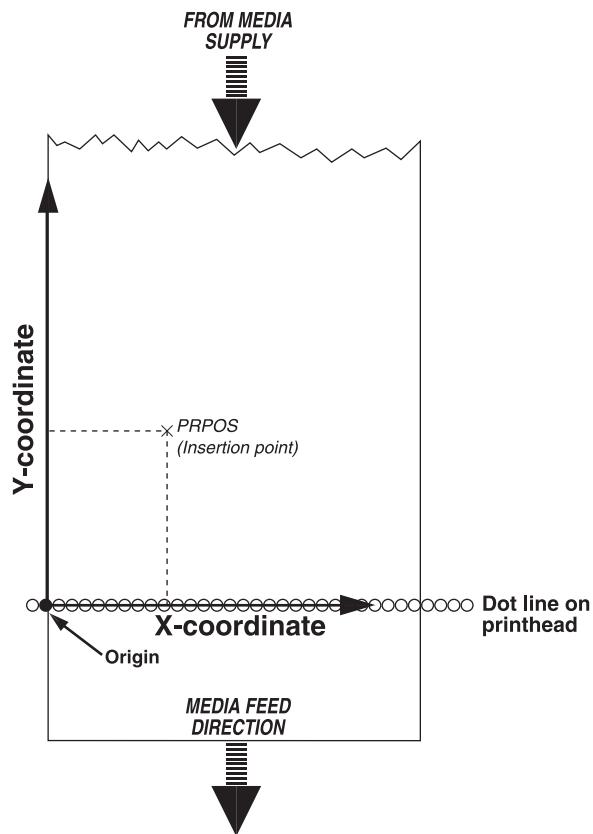
*<nexp1>* is the X-coordinate (number of dots from the origin).

*<nexp2>* is the Y-coordinate (number of dots from the origin).

Default for both is 0. Reset to default by executing a PRINTFEED.

**Remarks** When the printer is set up, a “print window” is created. This involves specifying the location of the origin along the X-axis, setting the maximum print width along the X-axis from the origin, and setting the maximum print length along the Y-axis from the origin.

The X-coordinate goes across the media path and the Y-coordinate along the media feed direction, as illustrated below. They are set in relation to the origin on the printhead, not in relation to the media. Thus, the position where an object actually will be printed depends on the relation between printhead and media at the moment when the printing starts.



The insertion point must be selected so the field in question fits inside the print window. This implies that the print direction, the size of the field (including “invisible” parts of an image), the alignment, and other formatting instructions must be considered. A field that does not fit entirely inside the print window causes error 1003 (“Field out of label”) except when a CLIP ON statement is issued.

To find the present insertion point, use the PRSTAT function.

**Examples** This example prints a line of text:

```

10 FONT "Swiss 721 BT"
20 PRPOS 30,200
30 PRTXT "HELLO"
40 PRINTFEED
RUN

```

Each text line is normally positioned separately by its own PRPOS statement.

If no position is given for a printable statement, it is printed immediately after the preceding printable statement.

```
10 FONT "Swiss 721 BT"
20 PRPOS 30,200
30 PRTXT "SUMMER"
40 PRTXT "TIME"
50 PRINTFEED
RUN
```

resulting in a label with the text:

```
SUMMERTIME
```

A program for fixed line-spacing of text may be composed this way (another way is to use the extended PRBOX statement):

```
10 FONT"Swiss 721 BT"
20 X%=30 : Y%=500
30 INPUT A$
40 PRPOS X%,Y%
50 PRTXT A$
60 Y%=Y%-50
70 IF Y%>=50 GOTO 30
80 PRINTFEED
90 END
RUN
```

Enter the text for each line after the question mark shown on the screen of the host. The Y-coordinate is decremented by 50 dots for each new line until it reaches the value 50, which means that ten lines are printed.

## PRSTAT

**Purpose** Returns the current printer status or the current position of the insertion point.

**Syntax** PRSTAT [ (<nexp>) ]

where:

- <nexp> = 1      returns the X-position for the insertion point at DIR 1&3.
- <nexp> = 2      returns the Y-position for the insertion point at DIR 2&4.
- <nexp> = 3      returns the X-position of the corner with the lowest coordinates of the last object.
- <nexp> = 4      returns the Y-position of the corner with the lowest coordinates of the last object.
- <nexp> = 5      returns the width along the X-axis of the last object.
- <nexp> = 6      returns the height along the Y-axis of the last object.
- <nexp> = 7      returns the print job identifier
- <nexp> = 8      returns the print job state (see table below).

- <nexp> = 9      returns the print job error code.  
<nexp> = 10     returns the remaining number of copies to be printed in a batch print job.

**Remarks****PRSTAT**

Returns a numeric expression, which is the sum of the values given by the following conditions, at the moment when the PRSTAT function is executed:

| <b>Value</b> | <b>Condition</b>                                              |
|--------------|---------------------------------------------------------------|
| 0            | OK                                                            |
| 1            | Printhead lifted                                              |
| 2            | Label not removed (see note)                                  |
| 4            | Label stop sensor (LSS) detects no label                      |
| 8            | Printer out of transfer ribbon (TTR) or ribbon installed (DT) |
| 16           | Printhead voltage too high                                    |
| 32           | Printer is feeding                                            |
| 64           | Reserved                                                      |
| 128          | Printer out of media                                          |

**Note:** Always returns 0 in printers not fitted with a label taken sensor.



If two error conditions occur at the same time (for example, if the printhead is lifted and the printer is out of transfer ribbon), the sum will be  $(1+8) = 9$ . Every combination of errors will result in a unique sum. You can use it to branch to a subroutine such as notifying the operator or interrupting the program. The label stop sensor detects no label if a gap or black mark is in front of the sensor, as well as when the printer is out of media.

**PRSTAT(1) & PRSTAT(2)**

Returns the current position of the insertion point relative to either the X or the Y position, depending on the selected print direction. This is useful for measuring the length of a text string or a bar code.

**PRSTAT(3)through PRSTAT(6)**

Returns the position and size of the last object regardless of RENDER ON/OFF. Their values are not updated by the execution of a PRBUF statement.

**PRSTAT(7)-PRSTAT(10)**

Detects if a print job has been interrupted, so steps can be taken to reprint missing copies (see PRINTFEED).

PRSTAT (7) returns a print job identifier that is automatically assigned to the print job by the firmware.

PRSTAT (8) returns the state of the print job as a numeric expression, which is the sum of the values given by the following conditions:

| Value | Condition                                                         |
|-------|-------------------------------------------------------------------|
| 0     | Print cycle not set up for printing, perhaps due to out-of-ribbon |
| 1     | The previous print cycle never ended (timeout)                    |
| 2     | Print cycle has started                                           |
| 4     | All lines successfully printed                                    |
| 8     | Printing truncated (media shorter than print image)               |
| 16    | Printhead strobing error or label length exceeded                 |
| 32    | Ribbon low                                                        |

PRSTAT (8) = 6 or 22 indicates a successfully printed label (in the latter case error “next label not found” may have been detected).

PRSTAT (9) returns the error code (see Chapter 7, “Error Codes”) detected by the print engine during printfeed. It is used together with PRSTAT(8) to determine the error cause when using OPTIMIZE “BATCH” ON.

PRSTAT (10) returns the number of copies that remain to be printed in an interrupted batch print job.

**Examples** This examples shows how two error conditions are checked:

```

10 A% = PRSTAT
20 IF A% AND 1 THEN GOSUB 1000
30 IF A% AND 128 THEN GOSUB 1010
40 END
.....
1000 PRINT "Printhead is lifted":RETURN
1010 PRINT "Printer out of media":RETURN
RUN

```

This example illustrates how you can check the length of a text:

```

10 PRPOS 100,100: FONT "Swiss 721 BT"
20 PRTXT "ABCDEFGHIJKLM"
30 PRINT PRSTAT(1)
RUN

```

results in:

519

## PRTXT (PT)

**Purpose** Statement for providing the input data for a text field.

**Syntax** PRTXT | PT<<nexp>|<sexp>> [ ;<<nexp>|<sexp>>... ] [ ; ]

where <<nexp>|<sexp>> specifies one line of text (maximum 300 characters).

**Remarks** A text field consists of one line of text. The text field must be defined by FONT or FONTD and may be further defined and positioned by DIR, ALIGN, MAG, PRPOS, INVIMAGE, or NORIMAGE statements or their respective default values.

Two or more expressions can be combined to form a text line. They must be separated by semicolons (;) and are printed adjacently. Plus signs can also be used for the same purpose, but only between string expressions.

String constants must be enclosed by quotation marks, whereas numeric constants or any kind of variables must not.

**Examples** This example prints a line of text:

```
10 FONT "Swiss 721 BT"
20 PRPOS 30,300
30 PRTXT "How do you do?"
40 PRINTFEED
RUN
```

Several string constants and string variables can be combined into one line of text by the use of plus signs or semicolons:

```
10 FONT "Swiss 721 BT"
20 PRPOS 30,300
30 PRTXT "SUN"; "SHINE"
40 A$="MOON"
50 B$="LIGHT"
60 PRPOS 30,200
70 PRTXT A$+B$
80 PRINTFEED
RUN
```

resulting in the text:

```
SUNSHINE
MOONLIGHT
```

Numeric constants and numeric variables can be combined by the use of semicolons, but plus signs cannot be used in connection with numeric expressions:

```
10 FONT "Swiss 721 BT"
20 PRPOS 30,300
30 PRTXT 123;456
40 A%=222
50 B%=555
60 PRPOS 30,200
70 PRTXT A%;B%
80 PRINTFEED
RUN
```

resulting in the text:

```
123456
222555
```

Numeric and string expressions can be mixed on the same line, for example:

```
10 FONT "Swiss 721 BT"
20 PRPOS 30,300
30 A$="December"
40 B%=27
50 PRTXT A$; " ";B%; " ";"2007"
80 PRINTFEED
RUN
```

resulting in the text:

December 27 2007

Two program lines of text are printed on the same line if the first program line is appended by a semicolon:

```
10 FONT "Swiss 721 BT"
20 PRPOS 30,300
30 PRTXT "HAPPY"+" ";
40 PRTXT "BIRTHDAY"
50 PRINTFEED
RUN
```

resulting in the text:

HAPPY BIRTHDAY

## PUT

**Purpose** Statement for writing a given record from the random buffer to a given random file.

**Syntax** PUT [#] <nexp1>,<nexp2>

where:

# indicates that whatever follows is a number. Optional.

<nexp1> is the number assigned to the file when it was OPENed.

<nexp2> is the number of the record. Must be  $\geq 1$ .

**Remarks** Use LSET or RSET statements to place data in the random buffer before issuing the PUT statement.

**Example**

```
10 OPEN "PHONELIST" AS #8 LEN=26
20 FIELD#8,8 AS F1$, 8 AS F2$, 10 AS F3$
30 SNAME$="SMITH"
40 CNAME$="JOHN"
50 PHONE$="12345630"
60 LSET F1$=SNAME$
70 LSET F2$=CNAME$
80 RSET F3$=PHONE$
90 PUT #8,1
100 CLOSE#8
RUN
SAVE "PROGRAM 1.PRG "
NEW
10 OPEN "PHONELIST" AS #8 LEN=26
20 FIELD#8,8 AS F1$, 8 AS F2$, 10 AS F3$
30 GET #8,1
40 PRINT F1$,F2$,F3$
RUN
```

resulting in:

SMITH...JOHN.....12345630

## RANDOM

**Purpose** Function generating a random integer within a specified interval.

**Syntax** RANDOM (<nexp1>, <nexp2>)

where:

<nexp1> is the first integer in the interval.

<nexp2> is the last integer in the interval.

**Remarks** The randomly generated integer will be:

- equal to or greater than <nexp1>.
- equal to or less than <nexp2>.

**Example** This example produces ten random integers between 1 and 100:

```
10 FOR I%=1 TO 10
20 A% = RANDOM (1,100)
30 PRINT A%
40 NEXT I%
RUN
```

resulting in:

```
31
45
82
1
13
16
41
77
20
70
```

## RANDOMIZE

**Purpose** Statement for reseeding the random number generator, optionally with a specified value.

**Syntax** RANDOMIZE [<nexp>]

where <nexp> is the integer (0 to 99999999) with which the random number generator will be reseeded.

**Remarks** If no value is specified, a message appears asking you to enter a value between 0 and 99,999,999.

**Examples** In this example, a prompt appears, asking you to specify a value to reseed the generator:

```
10 RANDOMIZE
20 A%=RANDOM1,100)
30 PRINT A%
RUN
```

```
Random Number Seed (0 to 99999999) ?
Enter 555
```

resulting in:

36

In this example, the reseeding integer is specified, so there is no prompt:

```
10 RANDOMIZE 556
20 A%=RANDOM(1,100)
30 PRINT A%
RUN
```

resulting in:

68

A higher degree of randomization is obtained when the random integer generator is reseeded with another random integer, in this example provided by a TICKS function:

```
10 A%=TICKS
20 RANDOMIZE A%
30 B%=RANDOM(1,100)
40 PRINT B%
RUN
```

resulting in:

42

## READY

**Purpose** Statement for ordering a ready signal, for example XON, CTS/RTS or PE, to be transmitted from the printer on the specified communication channel.

**Syntax** READY [<nexp>]

where <nexp> optionally specifies a communication channel:

- 1 “uart1:”
- 2 “uart2:”
- 3 “uart3:”
- 4 “centronics:”
- 6 “usb1:”
- 8 “uart4:”
- 9 “uart5:”

**Remarks** The selected communication protocol usually contains a “ready” signal, which tells the host computer that the printer is ready to receive more data. The READY statement allows you to order a ready signal to be transmitted on the specified communication channel. If no channel is specified, the signal will be transmitted on the standard OUT channel (see SETSTDIO statement).

The READY signal is used to revoke a previously transmitted BUSY signal. However, the printer may still be unable to receive more data due to other reasons such as a full receive buffer.

For the “centronics:” communication channel, BUSY/READY controls the PE (paper end) signal on pin 12 according to an error-trapping routine (READY = PE low).

- Example** This example allows the printer to receive more data on “uart2:” after the process of printing a label is completed.

```
10 FONT "Swiss 721 BT"
20 PRTEXT "HELLO!"
30 BUSY2
40 PRINTFEED
50 READY2
RUN
```



**Note:** Running this example may require the optional interface board.

## **REBOOT**

**Purpose** Statement for restarting the printer.

**Syntax** REBOOT

**Remarks** This statement has exactly the same effect as cycling power to the printer.

## **REDIRECT OUT**

**Purpose** Statement for redirecting the output data to a created file.

**Syntax** REDIRECT .OUT [<sexp>]

where <sexp> (optional) is the name of the file to be created and where the output will be stored.

**Remarks** Normally the output data is transmitted on the standard output channel (see SETSTDIO statement). In most cases, this means the screen of the host. However, by means of a REDIRECT OUT <sexp> statement, a file can be created to which the output will be redirected. That implies that no data is echoed back to the host. Normal operation, with the output being transmitted on the standard output channel again, is resumed when a REDIRECT OUT statement without any appending file name is executed.

- Example** In this example, a file (“LIST.DAT”) is created to which the names of the files in the printer permanent memory are redirected. The redirection is then terminated (line No. 30) and the file is OPENed for input.

```
10 REDIRECT OUT "LIST.DAT"
20 FILES "/c"
30 REDIRECT OUT
40 OPEN "LIST.DAT" FOR INPUT AS #1
.
```

· · · · ·  
· · · · ·

## REM (')

- Purpose** Statement for adding headlines and comments to the program without including them in the execution.
- Syntax** REM | ' <remark>  
where <remark> is a string of text inserted in the program as a comment (maximum of 32,767 characters per line).
- Remarks** A REM statement may either be entered on a program line of its own or be inserted at the end of a line containing another instruction. In the latter case, REM must be preceded by a colon (":REM").  
A shorthand form for REM is an apostrophe (ASCII 39 dec.). It is possible to branch to a line of REM statement. Execution continues at the first executable line after the REM line. Use REM statements carefully, because they slow down execution and transfer of data and also take up valuable memory space.

**Example** This example includes both types of REM statements:

```
10 'Label format No. 1
20 FONT "Swiss 721 BT"
30 PRPOS 30,100
40 DIR 1 :REM Print across web
50 ALIGN 4 :REM Aligned left/baseline
60 MAG 2,2 :'Double height and width
70 PRTXT "HELLO"
80 PRINTFEED
RUN
```

## REMOVE IMAGE

- Purpose** Statement for removing a specified image from the printer memory.
- Syntax** REMOVE<sub>↔</sub> IMAGE <sexp>  
where <sexp> is the full name, including extensions, of the image to be removed.
- Remarks** Use this statement to delete obsolete or faulty images from the printer memory to save valuable space.  
Note that there is a distinction between images and image files (compare with IMAGES and FILES statements). Use REMOVE IMAGE to delete images downloaded by means of a STORE statement (see STORE and STORE IMAGE). Image files downloaded using other methods (such as a TRANSFER KERMIT statement) should be removed using a KILL statement the same way as other files.  
REMOVE IMAGE cannot be undone, so use this statement carefully.

**Example**    10 REMOVE IMAGE "LOGOTYPE.1"  
              RUN

## RENDER ON/OFF

**Purpose** Statement for enabling/disabling rendering of text, bar code, image, box, and line fields.

**Syntax**    RENDER ON|OFF

where ON (default) enables rendering and OFF disables rendering.

**Remarks** These statements are intended to get information regarding size and position of a field without actually rendering it; the field is not printed when the program is executed. The information on the field is retrieved using PRSTAT functions.

When rendering is disabled:

- PRTXT, PRBAR, PRIMAGE, PRLINE, and PRBOX statements give no result when a PRINTFEED statement is executed.
- statements other than PRPOS do not update the insertion point.
- field numbers (see FIELDNO) are not updated.
- statements such as CLIP ON/OFF, XORMODE ON/OFF, or BARSET retain their usual meanings.

PRBUF renders a field regardless of RENDER ON/OFF.

RENDER ON enables field rendering after a RENDER OFF statement. Duplicate statements have no effect; if a RENDER OFF statement has been executed, another RENDER OFF statement is ignored. The same applies to RENDER ON.

**Example** This example retrieves information on the size of a text field which was not rendered. Note that the actual result may vary depending on font, font size, and printer type.

```
10 RENDER OFF
20 PRTXT "Render off"
30 PRINT "Width:",PRSTAT(5), "Height:",
PRSTAT(6)
40 RENDER ON
50 PRINTFEED
RUN
```

resulting in:

```
Width: 153 Height: 46
No field to print in line 50
Ok
```

## RENUM

**Purpose** Statement for renumbering the lines of the program currently residing in the printer's working memory.

**Syntax** RENUM [<ncon1> [, [<ncon2> [, <ncon3>]]]

where:

<ncon1> is the first line number of the new sequence. Default is 10.

<ncon2> is the line in the current program at which renumbering is to start. Default is 1.

<ncon3> is the desired increment between line numbers in the new sequence. Default is 10.

**Remarks** This statement is useful for providing space for more program lines when expanding an existing program, and for renumbering programs written without line numbers (for example, after an unnumbered programs is LISTed, LOADED, or MERGED). Line references following GOTO statements are renumbered accordingly. Use a LIST statement to print the new numbers on the screen.

**Example** This example renames a program:

```
10 FONT "Swiss 721 BT"
20 PRPOS 30,100
30 PRTXT "HELLO"
40 A%=A%+1
50 PRINTFEED
60 IF A%<3 GOTO 40
70 END
RENUM 100,20,50
LIST
```

resulting in:

```
10 FONT "Swiss 721 BT"
100 PRPOS 30,100
150 PRTXT "HELLO"
200 A%=A%+1
250 PRINTFEED
300 IF A%<3 GOTO 200
350 END
```

Note that the line number in the GOTO statement on line 300 has changed. Line 10 is not renumbered, since line 20 was specified as the starting point. The new increment is 50.

## REPRINT ON/OFF

**Purpose** Statement for enabling/disabling reprinting of a label in the Direct Protocol.

**Syntax** REPRINT <ON|OFF>

where ON (default) enables reprinting and OFF disables reprinting.

**Remarks** REPRINT OFF also affects and overrides the behavior of the PRINT KEY ON statement. If REPRINT OFF is entered before PRINT KEY ON, no error message is shown but PRINT KEY is set to ON and an empty label is printed when the **Print** key is pressed.

If REPRINT is set to OFF, there is no way to reprint an old print job. If a PRINTFEED statement is sent to the printer after a print job has been completed, a blank label is fed out and an error 1006 (“No field to print”) occurs. However, the REPRINT OFF statement does not clear the print buffer, which only occurs after a PRINTFEED statement has been executed (as seen in the second example below).

Leaving and re-entering the Direct Protocol does not reset the REPRINT status. A reboot resets the REPRINT status to its default value (ON). A REPRINT OFF statement prevents automatic reprinting after the error has been cleared for the following errors:

- 1005: “Out of paper”
- 1022: “Head lifted”
- 1031: “Next label not found”
- 1058: “Transfer ribbon is installed”

Error 1027 (“Out of transfer ribbon”) does not generate the display prompt “Continue-Reprint” if REPRINT is set to OFF.

**Examples** This example disables reprinting:

|             |                           |
|-------------|---------------------------|
| INPUT ON    | Enter the Direct Protocol |
| REPRINT OFF | Disable reprinting        |

This example shows a special case, when the first and second PRINTFEED statements generate a printed label. After the second PRINTFEED, the print buffer is cleared and a “No field to print” error occurs.

|                |                                             |
|----------------|---------------------------------------------|
| INPUT ON       | Enter the Direct Protocol                   |
| PRTXT "PRINT1" | Print the text “Print1”                     |
| PRINTFEED      | Yields a print label                        |
| REPRINT OFF    | Disable reprinting                          |
| PRINTFEED      | Yields a printed label, clears print buffer |
| PRINTFEED      | Yields a blank label, generates an error    |

## RESUME

**Purpose** Statement for resuming program execution after an error-handling subroutine has been executed.

**Syntax** RESUME [<< ncon >> | < line\_label > | < NEXT > | < 0 >> | < HTTP >]

where <ncon>|<line label> is the number or label of the line to which the program should return.

**Remarks** RESUME must only be used in connection with error-handling subroutines (see ON ERROR GOTO).

There are five ways of using RESUME:

- RESUME or RESUME 0: Execution is resumed at the statement where the error occurred.
- RESUME NEXT: Execution is resumed at the statement immediately following the one that caused the error.
- RESUME *<ncon>*: Execution is resumed at the specified line.
- RESUME *<line label>*: Execution is resumed at the specified line label.
- RESUME *<HTTP>*: Execution is resumed at the point where it was branched by an ON HTTP GOTO statement. Stdin and stdout are restored to their original values.

**Examples** This short program is the basis for two examples of alternative subroutines:

```
10 ON ERROR GOTO 1000
20 FONT "Swiss 721 BT"
30 PRPOS 100,100
40 PRTXT "HELLO"
50 PRPOS 100, 300
60 PRIMAGE "GLOBE.1"
70 PRINTFEED
80 END
```

### Alternate Subroutine 1

A font is selected automatically and execution resumes from the line after the error occurred. If another error than the specified error condition occurs, the execution is terminated.

```
1000 IF ERR=15 THEN FONT "Swiss 721 BT":RESUME NEXT
1010 RESUME 80
```

### Alternate Subroutine 2

An error message is printed and the execution goes on from the line following the error.

```
1000 IF ERR=15 THEN PRINT "Font not found"
1010 RESUME NEXT
```

## RETURN

**Purpose** Statement for returning to the main program after having branched to a subroutine because of a GOSUB statement.

**Syntax** RETURN [*<ncon>* | *<line label>*]

where *<ncon>* or *<line label>* is the number or label of a line in the main program to return to.

**Remarks** When the statement RETURN is encountered during the execution of a subroutine, the execution returns to the main program. Execution continues from the statement immediately following the most recently executed GOSUB or from an optionally specified line.

If a RETURN statement is encountered without a GOSUB statement having been previously executed, error 28 (“Return without Gosub”) occurs.

**Example**

```
10 PRINT "This is the main program"
20 GOSUB 1000
30 PRINT "You're back in the main program"
40 END
1000 PRINT "This is subroutine 1"
1010 GOSUB 2000
1020 PRINT "You're back in subroutine 1"
1030 RETURN
2000 PRINT "This is subroutine 2"
2010 GOSUB 3000
2020 PRINT "You're back in subroutine 2"
2030 RETURN
3000 PRINT "This is subroutine 3"
3010 PRINT "You're leaving subroutine 3"
3020 RETURN
RUN
```

resulting in:

```
This is the main program
This is subroutine 1
This is subroutine 2
This is subroutine 3
You're leaving subroutine 3
You're back in subroutine 2
You're back in subroutine 1
You're back in the main program
```

## REWINDCONTROL

**Purpose** Function to control the internal rewind motor in EasyCoder PX-series printers.

**Syntax** REWINDCONTROL <nexp>

where <nexp> controls the rewind motor as follows:

- Positive values specify how long the internal rewinder pulls the liner before the paper feed motor starts. Maximum value is 5000 dots.
- Negative values specify the distance (in dots) the paper is fed before the internal rewind motor starts pulling liner. A value of -1 turns the rewind motor off. Maximum value is label length.

**Remarks** EasyCoder PX-series printers may have two separate motors, one for feeding paper and one for rewinding liner. REWINDCONTROL allows for control of the internal rewind motor.

A positive value stretches the liner before printing to ensure better label dispensing. However, a positive value might cause the stepper motor to stall for certain liner surfaces. In such cases, contact your local Intermec support organization.

A negative value allows the label to be rewound with the liner using the self-strip mode. A value of -1 turns the internal rewind motor off, which makes for more silent operation when the printer runs without self-strip. This command cannot be used when REWINDVOID ON is enabled.

- Examples** REWINDCONTROL 200  
REWINDCONTROL -100

## REWINDVOID

- Purpose** Function to rewind VOID labels with the liner in EasyCoder PX-series printers with RFID enabled.
- Syntax** REWINDVOID OFF|ON [<nexp>]  
where <nexp> specifies the distance (in dots) the paper is fed before the internal rewind motor starts pulling liner. Default is 100. Maximum value is label length.
- Remarks** EasyCoder PX-series printers may have two separate motors, one for feeding paper and one for rewinding liner. This function causes a VOID tag/label to be rewound with the liner paper when using the printer in selfstrip mode, preventing a label being from being dispensed (such as to an applicator). The value for <nexp> must be tested for each specific type of media and label length.
- Example** REWINDVOID ON 200

## RIGHT\$

- Purpose** Returning a specified number of characters from a given string, starting from the end (extreme right end) of the string.
- Syntax** RIGHT\$ (<sexp>, <nexp>)  
where:  
<sexp> is the string from which the characters will be returned.  
<nexp> specifies the number of characters to be returned.
- Remarks** This function is the complementary function for LEFT\$, which returns the characters starting from the beginning (extreme left side) of the string. If the number of characters to be returned is greater than the number of characters in the string, then the entire string is returned. If the number of characters is set to zero, a null string is returned.

**Example 1** PRINT RIGHT\$ ("THERMAL\_PRINTER", 7)

results in:

PRINTER

**Example 2** 10 A\$="THERMAL\_PRINTER":B\$ = "LABEL"

20 PRINT RIGHT\$(B\$, 5);RIGHT\$(A\$, 8);"S"

RUN

resulting in:

LABEL\_PRINTERS

## RSET

**Purpose** Statement for placing data right-justified into a field in a random file buffer.

**Syntax** RSET<svvar>=<sexp>

where:

<svvar> is the string variable assigned to the field by a FIELD statement.

<sexp> holds the input data.

**Remarks** After OPENing a file and formatting it using a FIELD statement, you can enter data into the random file buffer using the RSET and LSET statements (LSET left-justifies the data).

The input data can only be stored in the buffer as string expressions. Numeric expressions must be converted to strings by the STR\$ function before an LSET or RSET statement is executed.

If the length of the input data is less than the field, the data is right justified and the remaining number of bytes are printed as space characters.

If the length of the input data exceeds the length of the field, the input data is truncated on the left side.

**Example**

```
10 OPEN "PHONELIST" AS #8 LEN=26
20 FIELD#8,8 AS F1$, 8 AS F2$, 10 AS F3$
30 SNAME$="SMITH"
40 CNAME$="JOHN"
50 PHONE$="12345630"
60 LSET F1$=SNAME$
70 LSET F2$=CNAME$
80 RSET F3$=PHONE$
90 PUT #8,1
100 CLOSE#8
RUN
SAVE "PROGRAM 1.PRG "
NEW
10 OPEN "PHONELIST" AS #8 LEN=26
20 FIELD#8,8 AS F1$, 8 AS F2$, 10 AS F3$
30 GET #8,1
40 PRINT F1$,F2$,F3$
RUN
```

resulting in:

```
SMITH...JOHN.....12345630
```

## RUN

**Purpose** Statement for starting the execution of a program.

**Syntax** RUN [<< scon >> | <ncon>>]

where:

<scon> (optional) specifies an existing program to run.

<ncon> (optional) specifies the number of a line in the current program where the execution will start.

**Remarks** The RUN statement starts the execution of the program currently residing in the printer working memory, or of a specified program residing elsewhere. The execution begins at the line with the lowest number, or from a specified line in the current program.

If the program is stored somewhere besides the current directory (see CHDIR statement) and has not been LOADED, its designation must be preceded by a reference to that device (such as “/c”, “tmp:”, “/rom”, or “card1:”).

Never use RUN on a numbered line or in a line without number in the Programming Mode, or error 40 (“Run statement in program”) occurs.

A RUN statement executed in the Intermec Direct Protocol makes the printer switch to Immediate Mode and has the same effect as an INPUT OFF statement.

**Examples** To execute the current program from its first line:

```
RUN
```

To execute the current program starting from line 40:

```
RUN 40
```

To execute the program “TEST.PRG” from its first line:

```
RUN "TEST"
```

To execute the program “TEST.PRG” from its first line.

```
RUN "TEST.PRG"
```

To execute the program “FILELIST.PRG”, which is stored in the read-only memory, from its first line:

```
RUN "/rom/FILELIST.PRG"
```

## **SAVE**

**Purpose** Statement for saving a file in the printer's memory or in a memory card.

**Syntax** `SAVE<scon> [, P | L]`

where:

`<scon>` is the name of the file. To save the file to somewhere other than the current directory, start this string with the desired directory. Maximum of 30 characters including extension, or 26 characters without an extension.

`P` (optional) protects the file.

`L` (optional) saves the file without line numbers.

**Remarks** When a file is SAVED, its file name can be a maximum of 30 characters including an extension. By default, the program automatically appends the name with the extension .PRG and converts all lowercase characters to uppercase. The name must not contain any quotation marks ("").

If you start the file name with a period character (.), it is not removed at a soft formatting operation (see FORMAT). Such a file is also listed differently (see FILES).

Files can only be SAVED in the printer permanent memory ("c"), its temporary memory ("tmp."), or in an optional CompactFlash memory card ("card1:"). If a file with the selected name already exists in the selected directory, that file is deleted and replaced by the new file without warning. You can continue to work with a file after saving it until a NEW, LOAD, KILL, or REBOOT instruction is issued.

A protected file (SAVE <filename>,P) is encrypted at saving and cannot be LISTed after being LOADED. Program lines cannot be removed, changed, or added. Once a file has been protected, it cannot be deprotected again.

Therefore, you should save an unprotected copy, should a programming error be detected later on. If you are going to use an electronic key to prevent unauthorized access to a file, you should protect it.

A SAVED program can be MERGED with the program currently residing in the printer working memory. To prevent automatically assigned line numbers from interfering with the line numbers in the current program, you can choose to SAVE the program without line numbers (SAVE <filename>,L). See MERGE instruction.

**Examples** `SAVE "Label14"`

saves the file as "LABEL 14.PRG" in current directory.

`SAVE "/c/Label14", P`

saves and protects the file "LABEL14.PRG" in the permanent memory.

`SAVE "card1:Label14", L`

saves "LABEL14.PRG" without line numbers on a CompactFlash card.

## SET FAULTY DOT

- Purpose** Statement for marking one or several dots on the printhead as faulty, or marking all faulty dots as correct.
- Syntax** SET<sub>↔</sub>FAULTY<sub>↔</sub>DOT<exp1>[,<expn>...]
- where <exp1> is the number of the dot to be marked as faulty. Successive executions add more faulty dots. <exp1> = -1 marks all dots as correct (default).
- Remarks** This statement is closely related to the HEAD function and the BARADJUST statement. You can check the printhead for possible faulty dots by means of the HEAD function and mark them as faulty, using the SET FAULTY DOT statement. Using the BARADJUST statement, you can allow the firmware to automatically reposition horizontal bar codes sideways so as to place the faulty dots between the bars, preserving the readability.
- This command works on the EasyCoder PD41, but the HEAD command cannot be used to identify faulty dots.
- Once a dot has been marked faulty by a SET FAULTY DOT statement, it remains marked as faulty until all dots are marked as correct by a SET FAULTY DOT -1 statement.
- Note that the HEAD function makes it possible to mark all faulty dots using a single instruction instead of specifying each faulty dot using SET FAULTY DOT.

- Example** This example illustrates how a bar code is repositioned by means of BARADJUST when a number of dots are marked as faulty by a SET FAULTY DOTS statement. Type RUN and send various numbers of faulty dots from the host a few times to see how the bar code moves sideways across the label.

```

10 INPUT "No. of faulty dots"; A%
20 FOR B% = 1 TO A%
30 C% = C% + 1
40 SET FAULTY DOT C%
50 NEXT
60 D% = A%+2
70 BARADJUST D%, D%
80 PRPOS 0, 30
90 BARTYPE "CODE39"
100 PRBAR "ABC"
110 SET FAULTY DOT -1
120 PRINTFEED
RUN

```

## SETASSOC

- Purpose** Statement for setting a value for a tuple in a string association.
- Syntax** SETASSOC <exp1>, <exp2>, <exp3>

where:

- <sexp1> is the name of the association (case-sensitive).
- <sexp2> is the name of the tuple.
- <sexp3> is the value of the tuple.

**Remarks** An association is an array of tuples, where each tuple consists of a name and a value.

**Example** This example shows how a string, including three string names associated with three start values, is defined and one of them (time) is changed:

```
10 QUERYSTRING$="time=UNKNOWN&label=321&desc=DEF"
20 MAKEASSOC "QARRAY",QUERYSTRING$, "HTTP"
30 QTIME$=GETASSOC$ ("QARRAY", "time")
40 QLABELS%=VAL(GETASSOC$ ("QARRAY", "label"))
50 QDESC$=GETASSOC$ ("QARRAY", "desc")
60 PRINT"time=" ; QTIME$, "LABEL=" ; QLABELS% ,
"DESCRIPTION=" ; QDESC$
70 SETASSOC "QARRAY", "time", time$
80 PRINT "time=" ; GETASSOC$ ("QARRAY", "time")
RUN
```

resulting in:

```
time=UNKNOWN LABEL=321 DESCRIPTION=DEF
time=153355
```

## **SETPFSVAR**

**Purpose** Statement for registering variables to be saved at power failure.

**Syntax** SETPFSVAR<sexp> [ , <nexp>]

where:

- <sexp> is the name of a numeric or string variable (uppercase characters only).
- <nexp> is the size in bytes of a string variable. Maximum is 230.

**Remarks** When a program is loaded, it is copied to and executed in the printer temporary memory (“tmp:”). Should an unexpected power failure occur, the printer tries to save as much data as possible in the short time available before all power is lost. To minimize the risk of losing important variable data at a power failure, you can register numeric and string variables to be saved. There are 2176 bytes (including overhead) available for this purpose.

However, should the power failure occur while the printer is printing, there will be no power left to save the current variables.

When you register a string variable, you must also specify its size in bytes. The variable name is limited to a length of 20 characters.

**Note:** The EasyCoder PD41 lacks power-fail signal, and thus it cannot save variables at unexpected power failures.



Related instructions are GETPFSVAR, DELETEPFSVAR, and LISTPFSVAR.

**Examples** Example with string variable:

```
100 IF QA$="" THEN QA$="Hello":QA%=LEN(QA$)
110 SETPFSVAR "QA$", QA%
```

Example with numeric variable:

```
200 SETPFSVAR "QCPS%"
```

## SETSTDIO

**Purpose** Statement for selecting standard IN and OUT communication channel.

**Syntax** SETSTDIO<nexp1>[,<nexp2>]

where:

<nexp1> is the desired input/output channel:

|     |                               |
|-----|-------------------------------|
| 100 | autohunting enabled (default) |
| 0   | "console:"                    |
| 1   | "uart1:"                      |
| 2   | "uart2:"                      |
| 3   | "uart3:"                      |
| 4   | "centronics:"                 |
| 5   | "net1:"                       |
| 6   | "usb1:"                       |
| 7   | "uart4:"                      |
| 8   | "uart5"                       |

<nexp2> (optional) specifies a different output channel:

|   |            |
|---|------------|
| 0 | "console:" |
| 1 | "uart1:"   |
| 2 | "uart2:"   |
| 3 | "uart3:"   |
| 5 | "net1:"    |
| 6 | "usb1:"    |
| 7 | "uart4:"   |
| 8 | "uart5"    |

**Remarks** The printer is controlled from its host via a communication channel. By default, autohunting is selected, meaning that all available channels are continuously scanned for input. When data is received on a given channel, that channel is designated as the standard input/output channel. If no data is received on the present standard input channel within a 2-second timeout period, the firmware scans all other existing channels (except "console:") looking for input data. The channel where input data is first found is then designated the new STDIN/STDOUT channel. The same procedure is repeated infinitely as long as autohunting is enabled.

These restrictions apply to autohunting:

- If “centronics:” is used as input channel and autohunting is enabled, “uart1:” is selected stdio channel.
- Autohunting does not work with “console:”.
- Autohunting does not work with COMSET, INKEY\$, INPUT\$, or LINE INPUT.
- When using autohunting in the programming mode, the detected port stays as STDIO until the program breaks.

It is also possible to specify a certain channel as permanent STDIN and/or STDOUT channel. If only one channel is specified, it serves as both the standard input and standard output channel. Alternatively, a different channel can be selected for the standard output channel.

For programming, Intermec recommends using “uart1:” for both the standard input and standard output channels. If another channel is selected, use the same serial channel for both input and output. The “centronics:” channel can only be used for input to the printer and is not suited for programming.

**Example** This example selects the “uart2:” communication channel as the standard input and output channel:

```
10 SETSTDIO 2
. . . .
. . . .
```

This example enables autohunting for input and “uart1:” for output:

```
10 SETSTDIO 100,1
. . . .
. . . .
```

## **SETUP**

**Purpose** Statement for entering Setup Mode or for changing the setup.

**Syntax 1** `SETUP`

If no parameter is specified, the printer enters Setup Mode. This statement has no effect on the EasyCoder PD41.

**Syntax 2** `SETUP<sexp>`

where `<sexp>` is the name of an existing setup file that is used to change the entire current printer setup, or a string used to change a single parameter in the current printer setup.

**Syntax 3** `SETUP <sexp1>,<sexp2>`

where:

`<sexp1>` is the name of a setup section (EasyLAN User’s Guide).

`<sexp2>` is the name of a file that will be used to change the specified setup section.

**Syntax 4**    SETUP <sexp1>, <sexp2>, <sexp3>

where:

<sexp1> is the name of a setup section (see the EasyLAN User's Guide).  
Not implemented for "prt".

<sexp2> is the name of the setup object (see the EasyLAN User's Guide).

<sexp3> specifies the new value (see the EasyLAN User's Guide).

**Remarks**    The methods of manual setup via the printer keyboard are described in the User's Guides for the various printer models. You can also change the setup as a part of the program execution, or change the setup remotely from the host, by using setup files and setup strings.

A setup file may contain new values for one or several setup parameters, whereas a setup string only can change a single parameter. Creating a setup file requires several operations. Setup strings can be created in a single operation, making them suitable for use with the Intermec Direct Protocol.

By default, setup parameters are saved as a file in the printer permanent memory. However, you can use SYSVAR (35) to prevent changes from being saved. See SYSVAR for more information.

When a SETUP<sexp> statement is encountered, the setup is changed accordingly, and then the program execution is resumed. Note that some printing instructions (ALIGN, DIR, FONT, and PRPOS) may be changed when test labels are printed.

Setup files or setup strings have a special syntax for each parameter that must be followed exactly:

- Variable numeric input data is indicated by "n"—"nnnn", string data by "ssss", and alternative data by bold characters separated by vertical bars (|).
- Compulsory space characters are indicated by double-headed arrows (↔). Do not type these. Note that some parameters listed below may only apply to a certain printer model or an optional device.

```
"SER-COM, UART1 | UART2 | UART3 | UART4 | UART5, BAUDRATE,
300 | 600 | 1200 | 2400 | 4800 | 9600 | 19200 | 38400 | 57600 | 115200"
"SER-COM, UART1 | UART2 | UART3 | UART4 | UART5, CHAR.LENGTH, 7 | 8"
"SER-COM, UART1 | UART2 | UART3 | UART4 | UART5, PARITY, NONE | EVEN | ODD | MARK | SPACE"
"SER-COM, UART1 | UART2 | UART3 | UART4 | UART5, STOPBITS, 1 | 2"
"SER-COM, UART1 | UART2 | UART3 | UART4 | UART5, FLOWCONTROL, RTS/CTS, ENABLE | DISABLE"
"SER-COM, UART1 | UART2 | UART3 | UART4 | UART5, FLOWCONTROL, ENQ/ACK, ENABLE | DISABLE"
"SER-COM, UART1 | UART2 | UART3 | UART4 | UART5, FLOWCONTROL, XON/
XOFF, DATA.FROM.HOST, ENABLE | DISABLE"
"SER-COM, UART1 | UART2 | UART3 | UART4 | UART5, FLOWCONTROL, XON/
XOFF, DATA.TO.HOST, ENABLE | DISABLE"
"SER-COM, UART2, PROT.ADDR, ENABLE | DISABLE"
"SER-COM, UART1 | UART2 | UART3 | UART4 | UART5, NEW.LINE, CR/LF | LF | CR"
"SER-COM, UART1 | UART2 | UART3 | UART4 | UART5, REC.BUF, nnnnn"
"SER-COM, UART1 | UART2 | UART3 | UART4 | UART5, TRANS.BUF, nnnnn"
"SER-COM, UART2, PROTOCOL.ADDR., nn"
"NET-COM, NET1, NEW.LINE, CR/LF | LF | CR"
```

```
"NETWORK, IP.SELECTION, DHCP+BOOTP | MANUAL | DHCP | BOOTP"
"NETWORK, IP.ADDRESS, nnn.nnn.nnn.nnn"
"NETWORK, NETMASK, nnn.nnn.n.n"
"NETWORK, DEFAULT.ROUTER, nnn.nnn.nnn.nnn"
"NETWORK, NAME.SERVER, nnn.nnn.n.n"
"RFID, ON|OFF"
"RFID, ON, TAGADJUST, nnnn" (negative value allowed)
"RFID, ON, RETRIES, nn"
"RFID, ON, VOIDTEXT, sssss"
"FEEDADJ, STARTADJ, nnnn" (negative value allowed)
"FEEDADJ, STOPADJ, nnnn" (negative value allowed)
"MEDIA, MEDIA.SIZE, XSTART, nnnn"
"MEDIA, MEDIA.SIZE, WIDTH, nnnn"
"MEDIA, MEDIA.SIZE, LENGTH, nnnnn"
"MEDIA, MEDIA.TYPE, LABEL. (w.GAPS) | TICKET. (w.MARK) | TICKET. (w.GAPS) | FIX.LENGTH.ST
RIP|VAR.LENGTH STRIP"
"MEDIA, PAPER.TYPE, TRANSFER|DIRECT.THERMAL"
"MEDIA, PAPER.TYPE, DIRECT.THERMAL, LABEL.CONSTANT, nnn"
"MEDIA, PAPER.TYPE, DIRECT.THERMAL, LABEL.FACTOR, nnn"
"MEDIA, PAPER.TYPE, TRANSFER, RIBBON.CONSTANT, nnn"
"MEDIA, PAPER.TYPE, TRANSFER, RIBBON.FACTOR, nnn"
"MEDIA, PAPER.TYPE, TRANSFER, LABEL.OFFSET, nnn"
"MEDIA, PAPER.TYPE, TRANSFER, LOW.DIAMETER, nnn"
"MEDIA, CONTRAST, -10% | -8% | -6% | -4% | -2% | +0% | +2% | +4% | +6% | +8% | +10%"
"MEDIA, TESTFEED.MODE, FAST|SLOW"
"MEDIA, LEN(SLOW MODE), nn"
"MEDIA, PAPER, LOW.DIAMETER, nnn"
"PRINT.DEFS, PRINT.SPEED, nnn"
"PRINT.DEFS, LTS.VALUE, nn"
```

**Example 1** This example enables a key for branching to Setup Mode:

```
10 ON KEY(18) GOSUB 1000
20 KEY(18) ON
.....
1000 SETUP
1010 RETURN
```

**Example 2** This example shows how a new file is OPENed for output and each parameter in the setup is changed by means of PRINT# statements. Then the file is CLOSEd. Any lines, except the first and the last line in the example, may be omitted. Finally, the printer setup is changed using this file.

```
10 OPEN "/tmp/SETUP.SYS" FOR OUTPUT AS #1
20 PRINT#1, "SER-COM,UART1,BAUDRATE,19200"
30 PRINT#1, "SER-COM,UART1,CHAR LENGTH,7"
40 PRINT#1, "SER-COM,UART1,PARITY,EVEN"
50 PRINT#1, "SER-COM,UART1,STOPBITS,2"
60 PRINT#1, "SER-COM,UART1,FLOWCONTROL,RTS/CTS,ENABLE"
70 PRINT#1, "SER-COM,UART1,FLOWCONTROL,ENQ/ACK,ENABLE"
80 PRINT#1, "SER-COM,UART1,FLOWCONTROL,XON/XOFF,DATA FROM
HOST,ENABLE"
90 PRINT#1, "SER-COM,UART1,FLOWCONTROL,XON/XOFF,DATA TO
HOST,ENABLE"
100 PRINT#1, "SER-COM,UART1,NEW LINE,CR"
110 PRINT#1, "SER-COM,UART1,REC BUF,800"
120 PRINT#1, "SER-COM,UART1,TRANS BUF,800"
130 PRINT#1, "FEEDADJ,STARTADJ,-135"
```

```

140 PRINT#1, "FEEDADJ,STOPADJ, -36"
150 PRINT#1, "MEDIA,MEDIA SIZE,XSTART,50"
160 PRINT#1, "MEDIA,MEDIA SIZE,WIDTH,1000"
170 PRINT#1, "MEDIA,MEDIA SIZE,LENGTH,2000"
180 PRINT#1, "MEDIA,MEDIA TYPE,LABEL (w GAPS)"
190 PRINT#1, "MEDIA,PAPER TYPE,TRANSFER"
200 PRINT#1, "MEDIA,PAPER TYPE,TRANSFER,RIBBON CONSTANT,110"
210 PRINT#1, "MEDIA,PAPER TYPE,TRANSFER,RIBBON FACTOR,25"
220 PRINT#1, "MEDIA,PAPER TYPE,TRANSFER,LABEL OFFSET,00"
230 PRINT#1, "MEDIA,PAPER TYPE,TRANSFER,LOW DIAMETER,30"
230 PRINT#1, "MEDIA,CONTRAST,-4%"
240 PRINT#1, "PRINT DEFS,PRINT SPEED,200"
250 CLOSE
260 SETUP "/tmp/SETUP.SYS"

```

**Example 3** This example shows how a setup parameter is changed in the Immediate Mode (or Intermec Direct Protocol) using a setup string.

```
SETUP"MEDIA,MEDIA TYPE,VAR LENGTH STRIP" .
```

This method can also be used in Programming Mode, as in this example:

```
10 SETUP"MEDIA,MEDIA TYPE,VAR LENGTH STRIP"
```

## SETUP GET

**Purpose** Statement for getting the current setting for a single setup object.

**Syntax** `SETUP GET<sexp1>,<sexp2>,<sexp3>`

where:

<sexp1> specifies the setup section.

<sexp2> specifies the setup object.

<sexp3> stores the result.

**Remarks** Refer to the EasyLAN User's Guide for a list of setup sections and objects.

**Examples** `SETUP GET "lan1","RTEL_PR1",A$`  
`SETUP GET "prt","MEDIA,MEDIA TYPE", B$`  
`SETUP GET "alerts","lts",C$`

## SETUP KEY

**Purpose** Statement for enabling or disabling the access to Setup Mode from the printer keypad.

**Syntax** `SETUP KEY ON|OFF`

where ON enables entering Setup mode through the keypad, and OFF disables entering Setup mode through the keypad.

**Remarks** You can prevent operators or non-authorized personnel from changing printer settings from the printer keypad. SETUP KEY OFF allows the administrator to disable the SETUP key on the printer keypad, effectively preventing changes to the printer setup. SETUP KEY ON re-enables the SETUP key and access to the setup mode.

The setup mode can always be entered via a SETUP command.

## **SETUP WRITE**

**Purpose** Statement for creating a file containing the current printer setup, or for returning the setup file on a specified communication channel.

**Syntax** `SETUP WRITE [<sexp1> ,<sexp2>]`

where:

`<sexp1>` (optional) specifies the setup section.

`<sexp2>` is the name of a file or device to which the current printer setup is written.

**Remarks** The SETUP WRITE statement is useful when you want to return to the printer's current setup at a later time. You can make a copy of the current setup using `SETUP WRITE <filename>`, change the setup using a `SETUP <filename>` statement, and return to the original setup when needed by issuing a new `SETUP <filename>` statement containing the name of the copy you created.

Intermec recommends that you create the file in the printer's temporary memory ("tmp:"), as in `SETUP WRITE "tmp:OLDSETUP"`. Once the file has been created in "tmp:", the file can be copied to the printer's permanent memory ("/c") so it is not lost at power off.

SETUP WRITE can be used to transmit the current printer setup on a serial communication channel, as in this example:

```
SETUP WRITE "uart1:".
```

Setup sections are used in connection with EasyLAN. Refer to the EasyLAN User's Guide for a list of setup sections.

SETUP WRITE returns the printer setup in this example, which shows a standard EasyCoder PF4i printer:

```
SETUP WRITE "uart1:"
```

resulting in:

```
SER-COM, UART1, BAUDRATE, 9600
SER-COM, UART1, CHAR LENGTH, 8
SER-COM, UART1, PARITY, NONE
SER-COM, UART1, STOPBITS, 1
SER-COM, UART1, FLOWCONTROL, RTS/CTS, DISABLE
SER-COM, UART1, FLOWCONTROL, ENQ/ACK, DISABLE
SER-COM, UART1, FLOWCONTROL, XON/XOFF, DATA FROM HOST, DISABLE
SER-COM, UART1, FLOWCONTROL, XON/XOFF, DATA TO HOST, DISABLE
SER-COM, UART1, NEW LINE, CR/LF
```

```

SER-COM, UART1, REC BUF, 1024
SER-COM, UART1, TRANS BUF, 1024
FEEDADJ, STARTADJ, 0
FEEDADJ, STOPADJ, 0
MEDIA, MEDIA SIZE, XSTART, 24
MEDIA, MEDIA SIZE, WIDTH, 832
MEDIA, MEDIA SIZE, LENGTH, 1200
MEDIA, MEDIA TYPE, LABEL (w GAPS)
MEDIA, PAPER TYPE, TRANSFER
MEDIA, PAPER TYPE, DIRECT THERMAL, LABEL CONSTANT, 85
MEDIA, PAPER TYPE, DIRECT THERMAL, LABEL FACTOR, 40
MEDIA, PAPER TYPE, TRANSFER, RIBBON CONSTANT, 95
MEDIA, PAPER TYPE, TRANSFER, RIBBON FACTOR, 25
MEDIA, PAPER TYPE, TRANSFER, LABEL OFFSET, 0
MEDIA, PAPER TYPE, TRANSFER, RIBBON SENSOR, 14
MEDIA, PAPER TYPE, TRANSFER, LOW DIAMETER, 36
MEDIA, CONTRAST, +0%
#MEDIA, TESTFEED, 26 28 0 10
MEDIA, TESFEED MODE, FAST
MEDIA, LEN (SLOW MODE), 0
PRINT DEFS, HEAD RESIST, 702
PRINT DEFS, PRINT SPEED, 100

```

Note that when a SETUP WRITE file is used to change the setup, the printer's present TESTFEED adjustment is not affected.

- Examples** In this example, the current setup is saved in the printer temporary memory under the name "SETUP1.SYS". Then the start adjustment is changed to "200" by the creation of a new setup file named "SETUP2. SYS." Finally, the created setup file is used to change the printer setup.

```

10 SETUP WRITE "tmp:SETUP1.SYS"
20 OPEN "tmp:SETUP2.SYS" FOR OUTPUT AS #1
30 PRINT#1, "FEEDADJ, STARTADJ, 200"
40 CLOSE
50 SETUP "tmp:SETUP2.SYS"

```

In this example, the setup section "prt" is returned on the serial channel "uart1:":

```
SETUP WRITE "prt", "uart1:"
```

## SGN

- Purpose** Returns the sign (positive, zero, or negative) of a specified numeric expression.

**Syntax** SGN(<nexp>)

where <nexp> is the numeric expression from which the sign will be returned.

- Remarks** The sign is returned as:

|                  |          |
|------------------|----------|
| SGN(<nexp>) = -1 | Negative |
| SGN(<nexp>) = 0  | Zero     |
| SGN(<nexp>) = 1  | Positive |

**Examples** Positive numeric expression:

```
10 A%=(5+5)
20 PRINT SGN(A%)
RUN
```

results in:

1

Negative numeric expression:

```
10 A%=(5-10)
20 PRINT SGN(A%)
RUN
```

results in:

-1

Zero numeric expression:

```
10 A%=(5-5)
20 PRINT SGN(A%)
RUN
```

results in:

0

## SORT

**Purpose** Statement for sorting a one-dimensional array.

**Syntax** SORT<<ivar>>|<svar>>,<nexp1>,<nexp2>,<nexp3>

where:

<<ivar>>|<svar>> is the array to be sorted.

<nexp1> is the number of the first element.

<nexp2> is the number of the last element.

<nexp3> In a string array, the value specifies the position according to which the array will be sorted:

> 0: Ascending sorting

< 0: Descending sorting

= 0: Illegal value

**Remarks** A numeric or string array can be sorted, in its entirety or within a specified range of elements, in ASCII value order.

<nexp3> is used differently for numeric and string arrays. The sign always specifies ascending or descending order. For numeric arrays, the value is of no consequence, but for string arrays, the value specifies in which character position the elements will be sorted. <nexp3> = 0 results in error 41 (“Parameter out of range”).

**Example** In this example, one numeric and one string array are sorted in descending order. The string array is sorted in ascending order according to the third character position in each string:

```

10 ARRAY% (0) = 1001
20 ARRAY% (1) = 1002
30 ARRAY% (2) = 1003
40 ARRAY% (3) = 1004
50 ARRAY$ (0) = "ALPHA"
60 ARRAY$ (1) = "BETA"
70 ARRAY$ (2) = "GAMMA"
80 ARRAY$ (3) = "DELTA"
90 SORT ARRAY%, 0, 3, -1
100 SORT ARRAY$, 0, 3, 3
110 FOR I% = 0 TO 3
120 PRINT ARRAY% (I%), ARRAY$ (I%)
130 NEXT
RUN

```

resulting in:

```

1004 DELTA
1003 GAMMA
1002 ALPHA
1001 BETA

```

## SOUND

**Purpose** Statement for making the printer beeper produce a sound specified in regard of frequency and duration.

**Syntax** SOUND<exp1>, <exp2>

where:

<exp1> is the frequency of the sound in Hz.

<exp2> is the duration of the sound in periods of 0.020 seconds each (maximum of 15,0000 = 5 minutes).

**Remarks** This statement allows you to include significant sound signals in your programs. For example, a sound signal can notify the operator that various errors have occurred.

A sound with approximately the specified frequency is produced for the specified duration. If the program encounters a new SOUND statement, it is not executed until the previous sound reaches the specified duration. The EasyCoder PD41 makes no sound, but a time delay equal to the specified duration occurs.

The SOUND statement even allows you to make melodies, although the musical quality may be somewhat limited. For more information on setting specific musical pitches, see “[KEY BEEP](#)” on page 104.

**Example** The tune “Colonel Bogey” starts like this:

```
10 SOUND 392,10
20 SOUND 330,15
30 SOUND 330,10
40 SOUND 349,10
50 SOUND 392,10
60 SOUND 659,18
70 SOUND 659,18
80 SOUND 523,25
```

## **SPACE\$**

**Purpose** Returns a specified number of space characters.

**Syntax** SPACE\$ (<nexp>)

where <nexp> is the number of space characters to return.

**Remarks** This function is useful for more complicated spacing, such as in a table.

**Examples** This example prints two left-justified columns on the screen:

```
10 FOR Q%=1 TO 6
20 VERBOFF:INPUT "",A$
30 VERBON:PRINT A$;
40 VERBOFF:INPUT "",B$
50 VERBON
60 C$=SPACE$ (25-LEN (A$))
70 PRINT C$+B$
80 NEXT Q%
90 END
RUN
```

Enter:

```
January ↴
February ↴
March ↴
April ↴
May ↴
June ↴
July ↴
August ↴
September ↴
October ↴
November ↴
December ↴
```

resulting in:

|           |          |
|-----------|----------|
| January   | February |
| March     | April    |
| May       | June     |
| July      | August   |
| September | October  |
| November  | December |

## SPLIT

**Purpose** Splits a string into an array according to the position of a specified separator character and returns the number of elements in the array.

**Syntax** `SPLIT(<sexp1>, <sexp2>, <nexp>)`

where:

`<sexp1>` is the string to be split.

`<sexp2>` is the string array in which the parts of the split string should be put.

`<nexp>` specifies the ASCII value for the separator.

**Remarks** The string is divided by a specified separating character which may found an infinite number of times in the string. Each part of the string becomes an element in the string array, but the separator character itself will not be included in the array.

Should the string be split into more than four elements, error 57 (“Subscript out of range”) occurs. To avoid this error, issue a DIM statement to create a larger array before the string is split.

Note that SPLIT does not count if the last substring is empty. If this is valid for your data, see Example 2 for one way to handle this issue.

**Example 1** In this example a string is divided into five parts by the separator character # (ASCII 35 decimal). The result is an array of five elements numbered 0-4 as specified by a DIM statement. Finally, the number of elements is also printed on the screen.

```

10 A$="ONE#TWO#THREE#FOUR#FIVE"
20 B$="ARRAY$"
30 DIM ARRAY$(5)
40 C%=SPLIT(A$,B$,35)
50 PRINT ARRAY$(0)
60 PRINT ARRAY$(1)
70 PRINT ARRAY$(2)
80 PRINT ARRAY$(3)
90 PRINT ARRAY$(4)
100 PRINT C%
RUN

```

resulting in:

```

ONE
TWO
THREE
FOUR
FIVE
5

```

**Example 2**

```

10 A$="A#B#C#D" : GOSUB ZSUB
20 A$=A#B#C" : GOSUB ZSUB
30 END
40 ZSUB:
50 DIM ARRAY$ (4): REM Maximum numbers of substrings

```

```
60 C% = SPLIT(A$,"ARRAY$",35)
70 IF RIGHT$(A$,1)= CHR$(35) THEN
80 ARRAY$(C%)="" : C%=C%+1: REM Insert empty array entry.
90 END IF
100 FOR I%=0 to C%: PRINT ARRAY$(I%): NEXT: PRINT C%; "
total member count "
110 RETURN
```

## STOP

**Purpose** Statement for terminating execution of a program and returns the printer to Immediate Mode.

**Syntax** STOP

**Remarks** When a STOP statement is encountered, the following message is returned to the Debug STDOUT channel (default is “uart1:”):

Break in line <line number>

You can resume execution where it was stopped by means of a CONT statement, or at a specified program line using a GOTO statement in the Immediate Mode.

STOP is usually used in conjunction with CONT for debugging. When execution is stopped, you can examine or change the values of variables using direct mode statements. You may then use CONT to resume execution.

CONT is invalid if the program has been edited during the break.

Related instructions are CONT and DBSTDIO.

**Example**

```
10 A%=100
20 B%=50
30 IF A%=B% THEN GOTO QQQ ELSE STOP
40 GOTO 30
50 QQQ:PRINT "Equal"
Ok
RUN
Break in line 30
Ok
PRINT A%
100
Ok
PRINT B%
50
Ok
B%=100
OK
CONT
Equal
Ok
```

## STORE IMAGE

**Purpose** Sets up parameters for storing an image in the printer memory.

**Syntax** STORE<sub>↔</sub> IMAGE [RLL] [KILL] <sexp1>, <nexp1>, <nexp2>, [<nexp3>], <sexp2>

where:

- [RLL] (optional) indicates RLL compression.
- [KILL] (optional) erases the image from the temporary memory at startup (recommended).
- <sexp1> is the name of the image. Maximum 30 characters including extension.
- <nexp1> is the width of the image in bits (= dots).
- <nexp2> is the height of the image in bits (= dots).
- [<nexp3>] is the size of the images in bytes (RLL only).
- <sexp2> is the name of the protocol: “INTELHEX”  
“UBI00”  
“UBI01”  
“UBI02”  
“UBI03”  
“UBI10”

- Remarks** The name of the protocol must be entered in one sequence (for example, “INTELHEX”). Upper or lower case letters can be used. For more information on protocols, see Chapter 3, “About Image Transfer Protocols.”
- STORE IMAGE RLL is used when the image to be received is compressed into RLL format. In this case the size of the image must be included in the list of parameters (<nexp3>).
- STORE IMAGE KILL implies that the image will be stored in the printer temporary memory, which is erased at power off or REBOOT. Intermec strongly recommends this option to improve performance.
- To store the image permanently, copy it from the temporary memory (“tmp:”) to the permanent memory (“/c”) after the download is completed.
- A STORE IMAGE statement must precede any STORE INPUT statement.

- Example** This example shows how an Intelhex file is received via the standard input channel and stored in the printer temporary memory:

```

10 STORE OFF
20 INPUT "Name:", N$
30 INPUT "Width:", W%
40 INPUT "Height:", H%
50 INPUT "Protocol:", P$
60 STORE IMAGE N$, W%, H%, P$
70 INPUT "", F$
80 STORE F$
90 IF MID$(F$, 8, 2) <> "01" THEN GOTO 70
100 STORE OFF

```

## **STORE INPUT**

**Purpose** Statement for receiving and storing protocol frames of image data in the printer memory.

**Syntax** STORE<sub>↔</sub> INPUT<nexp1>[,<nexp2>]

where:

<nexp1> is the timeout in ticks (0.01 sec) before the next character is received.

<nexp2> (optional) is the number assigned to a device when it was OPENed for INPUT. Default is the standard IN channel.

**Remarks** The STORE INPUT statement receives and stores a protocol frame of image data as specified by preceding INPUT and STORE IMAGE statements.

It also performs an end frame check. (STORE INPUT substitutes the old STORE statement (not documented here)).

STORE INPUT works differently for various types of protocol:

- INTELHEX: Receives and stores frames until timeout or end frame is received.
- UBI00-03: Receives and stores frames until timeout or required number of bytes are received.
- UBI10: Receives and stores frames until timeout or end frame is received.

**Examples** This example shows how an Intelhex file is stored using the STORE IMAGE statement. The number of input parameters may vary depending on the type of protocol (see STORE INPUT statement).

```
10 STORE OFF
20 INPUT "Name:", N$
30 INPUT "Width:", W%
40 INPUT "Height:", H%
50 INPUT "Protocol:", P$
60 STORE IMAGE N$, W%, H%, P$
70 STORE INPUT 100
80 STORE OFF
```

To receive the input from another channel than std IN channel, the device must be OPENed for INPUT and a reference must be included in the STORE INPUT statement.

```
10 STORE OFF
20 OPEN "uart2:" FOR INPUT AS #9
30 INPUT "Name:", N$
40 INPUT "Width:", W%
50 INPUT "Height:", H%
60 INPUT "Protocol:", P$
70 STORE IMAGE N$, W%, H%, P$
80 STORE INPUT 100,9
```

```
90 CLOSE #9
100 STORE OFF
```

## STORE OFF

- Purpose** Terminates the storing of an image and resets the storing parameters.
- Syntax** STORE<sub>↔</sub> OFF
- Remarks** After having stored all protocol frames of an image, storing must be terminated by a STORE OFF statement. Even if you want to store another image, you must still issue a STORE OFF statement before the parameters for the new image can be set up using a new STORE IMAGE statement. Intermec recommends that you always start an image storing procedure by issuing a STORE OFF statement to clear the parameters of any existing STORE IMAGE statement.
- Example** This example shows how an Intelhex file is received via the standard IN channel and stored in the printer memory:

```
10 STORE OFF
20 INPUT "Name:", N$
30 INPUT "Width:", W%
40 INPUT "Height:", H%
50 INPUT "Protocol:", P$
60 STORE IMAGE N$, W%, H%, P$
70 STORE INPUT 100
80 STORE OFF
```

## STR\$

- Purpose** Returns the string representation of a numeric expression.
- Syntax** STR\$(*nexp*)  
where *nexp* is the numeric expression from which the string representation is returned.
- Remarks** This is the complementary function for the VAL function.
- Example** In this example, the value of the numeric variable A% is converted to string representation and assigned to the string variable A\$:

```
10 A%=123
20 A$=STR$(A%)
30 PRINT A%+A%
40 PRINT A$+A$
RUN
```

resulting in:

```
246
123123
```

## **STRINGS**

- Purpose** Repeatedly returns the character of a specified ASCII value, or the first character in a specified string.
- Syntax** STRING\$(<nexp1>, <<nexp2> | <sexp>>)
- where:
- <nexp1> is the number of times the specified character should be repeated.
  - <nexp2> is the ASCII decimal code of the character to be repeated.
  - <sexp> is the string expression from which the first character will be repeated.
- Remarks** The character to be repeated is specified either by its ASCII decimal code according to the selected character set (see NASC), or as the first character in a specified string expression.
- Example** This example shows both ways of using STRING\$. The asterisk character (\*) is ASCII 42 decimal:

```
10 A$="*INTERMEC*"
20 LEADING$ = STRING$(10,42)
30 TRAILING$ = STRING$(10,A$)
40 PRINT LEADING$; A$; TRAILING$
RUN
```

resulting in:

```
*****INTERMEC*****
```

## **SYSHEALTH**

- Purpose** Variable for setting or getting the system health and controlling the Intermec Ready-to-Work Indicator on the printer front panel.
- Syntax 1** ***Setting the system health***  
SYSHEALTH=<nexp>
- where <nexp> sets the Fingerprint application's view of the system health and controls the Intermec Ready-to-Work Indicator:
- 1 Indicator off
  - 2 Indicator blinking
  - 3 Indicator on (default)
- Syntax 2** ***Getting the system health***  
<nvar>=SYSHEALTH
- where <nvar> returns the current system health status as shown by the Ready-to-Work Indicator:
- 1 Indicator off
  - 2 Indicator blinking
  - 3 Indicator on

- Remarks** The readiness of the printer, individually or as a part of a solution, is indicated by the blue Intermec Ready-to-Work Indicator. If the indicator blinks or is switched off, the printer is not ready. Further information can be obtained in the printer display by pressing **F5/i** (not available on the EasyCoder PD41). In case of several errors or similar conditions occurring simultaneously, only the most significant error is displayed. Once this error has been cleared, the next remaining error is displayed.
- Provided the printer is connected to a network, all conditions that prevent printing are reported to the EasyADC Console. The EasyADC Console is software that allows a supervisor to monitor all connected devices that have an Intermec Ready-to-Work Indicator, including handheld computers, access points, and printers.
- The SYSHEALTH variable offers the opportunity to add more functionality to the Ready-to-Work Indicator than is offered by the printer's system. However, it does not override the standard indicator handling; that is, the worst case is always reported regardless if it is a system error or an application error.

- Example** This example shows how the Ready-to-Work Indicator can be made to show a “Connection refused” condition:

```

10 ON ERROR GOTO 1000
50 TRANSFER NET
"ftp://wrong.server.com/file", "c/myfile"
60 PRINT "XXX"
100 END
1000 IF ERR=1833 THEN SYSHEALTH=2 ELSE SYSHEALTH=3
1010 RETURN

```

You can find out the health of the system this way:

```

A%=SYSHEALTH
PRINT A%

```

## SYSHEALTH\$

- Purpose** Function for returning the error causing the current system health status.

- Syntax** <svar>=SYSHEALTH\$

where <svar> returns the error causing the current status, such as “Head lifted.”

- Remarks** If SYSHEALTH = 3, SYSHEALTH\$ returns “Operational.”

- Example** A\$=SYSHEALTH\$
PRINT A\$

resulting in:

Out of paper

## SYSVAR

**Purpose** System array for reading or setting system variables.

**Syntax** SYSVAR (<nexp>)

where <nexp> is the reference number of the system variable as described in the next table.

### SYSVAR Variables

| No.    | Description                                                | Notes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------|------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 - 13 | Reserved, obsolete or not implemented.                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 14     | Read errors since power on                                 | Reads number of errors detected since last power up.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 15     | Read errors since last SYSVAR(15)                          | Reads number of errors detected since last executed SYSVAR(15).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 16     | Read number of bytes received at execution of STORE INPUT  | Reads the number of bytes received after the execution of a STORE INPUT statement. Reset by the execution of a STORE IMAGE statement.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 17     | Read number of frames received at execution of STORE INPUT | Reads the number of frames received after the execution of a STORE INPUT statement. Reset by the execution of a STORE IMAGE statement.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 18     | Read or Set verbosity level                                | <p>In the Immediate and Programming Modes, all levels are enabled by default.</p> <p>In the Intermec Direct Protocol, all levels are disabled by default.</p> <p>Different verbosity levels can be selected:</p> <ul style="list-style-type: none"> <li>SYSVAR (18) = -1: All levels enabled (= VERBON)</li> <li>SYSVAR (18) = 0: No verbosity (= VERBOFF)</li> <li>SYSVAR (18) = 1: Echo received characters</li> <li>SYSVAR (18) = 2: “Ok” after correct command lines</li> <li>SYSVAR (18) = 4: Echo input characters from comm. port</li> <li>SYSVAR (18) = 8: Error after failed lines</li> </ul> <p>The levels can be combined, so for example SYSVAR(18)=3 means both “Echo received characters” and “Ok after correct command line.”</p> <p>The presently selected verbosity level can also be read and is returned as a numeric value, for example by PRINT SYSVAR(18).</p> |

**SYSVAR Variables (continued)**

| No. | Description                                        | Notes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-----|----------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 19  | Read or Set type of error message                  | <p>Four types of error messages can be selected:</p> <p>SYSVAR(19) = 1: &lt;string&gt; in line &lt;line&gt; (default)<br/>for example “Invalid font in line 10”</p> <p>SYSVAR(19) = 2: Error &lt;number&gt; in line &lt;line&gt;:<br/>&lt;string&gt;, for example “Error 19 in line 10: Invalid font”</p> <p>SYSVAR(19) = 3: E&lt;number&gt;, for example “E19”</p> <p>SYSVAR(19) = 4: Error &lt;number&gt; in line &lt;line&gt;, for example “Error 19 in line 10”</p> <p>The presently selected type of error message can also be read and is returned as a numeric value (1-4), for example by PRINT SYSVAR(19).</p> |
| 20  | Read direct or transfer mode                       | <p>SYSVAR(20) allows you to read if the printer is set up for direct thermal printing or thermal transfer printing, which is decided by your choice of paper type in the printer’s setup.</p> <p>The printer returns:</p> <p>0 = Direct thermal printing<br/>1 = Thermal transfer printing</p>                                                                                                                                                                                                                                                                                                                          |
| 21  | Read printhead density (dots/mm)                   | SYSVAR(21) allows you to read the density of the printer’s printhead, expressed as number of dots per millimeter.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 22  | Read number of printhead dots                      | SYSVAR(22) allows you to read the number of dots in the printer’s printhead.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 23  | Read status of transfer ribbon sensor              | <p>SYSVAR(23) allows you to read the status of the transfer ribbon sensor in thermal transfer printers.</p> <p>The printer returns:</p> <p>0 = No ribbon detected<br/>1 = Ribbon detected</p>                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 24  | Read if startup has occurred since last SYSVAR(24) | <p>This system variable is important when using the Intermec Direct Protocol.</p> <p>At power up, all data not saved as programs, files, fonts or images will be deleted, and most instructions will be reset to their respective default values. SYSVAR(24) allows the host to poll the printer to see if a power up has occurred, for example because of a power failure and, if so, download new data and new instructions.</p> <p>The printer returns:</p> <p>0 = No power up since last SYSVAR(24)<br/>1 = Power up has occurred since last SYSVAR(24)</p>                                                         |

**SYSVAR Variables (continued)**

| No. | Description                                               | Notes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----|-----------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 25  | Obsolete                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 26  | Read ribbon low condition                                 | Allows you to read the status of the ribbon low sensor, assuming that the printer is fitted with a thermal transfer mechanism. In the Setup Mode (Media/Paper Type/Transfer/Low Diameter), you can specify a diameter in mm of the ribbon supply roll, when SYSVAR(26) will switch from 0 to 1.<br><br>The printer returns:<br>0 = Ribbon not low<br>1 = Ribbon low<br><br>By default, the Low Diameter is set to 0, which disables the ribbon low function. However, if the Low Diameter is set to a higher value than 0 and SYSVAR(26) returns 1, the error condition 1083 “Ribbon Low” occurs at every tenth PRINTFEED operation. Further actions must be taken care of by the running Fingerprint program.                        |
| 27  | Set condition for label reprinting at out-of-ribbon error | When printing a batch of labels using thermal transfer printing (OPTIMIZE “BATCH” ON or PRINTFEED<n>), a label is deemed erroneous and thus eligible for reprinting if the ribbon has been empty for a distance longer than specified in dots by SYSVAR(27). Default is 0. Non-negative integers only.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 28  | Set or read media feed data erase at printhead lift       | The firmware keeps track of all labels (or similar) between the label stop sensor and the dot line of the printhead. If the printhead is lifted, there is a large risk that the media is moved, so the media feed will not work correctly before those labels have been fed out. This parameter allows you to decide or read whether these data should be cleared or not when the printhead is lifted.<br><br>SYSVAR(28) = 0 Media feed data are not cleared at headlift<br>SYSVAR(28) = 1 Media feed data are cleared at headlift (default)<br>SYSVAR(28) = 2 Media feed data are cleared at headlift and the firmware looks for the first gap or mark and adjusts the media feed using the same data as before the head was lifted. |

**SYSVAR Variables (continued)**

| No. | Description                                                     | Notes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-----|-----------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 29  | Read DSR condition on “uart2:”                                  | This parameter allows you to read the DSR (Data Send Ready) condition on the serial channel “uart2:”. The printer returns:<br>0 = No<br>1 = Yes                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 30  | Read DSR condition on “uart3:”                                  | This parameter allows you to read the DSR (Data Send Ready) condition on the serial channel “uart3:”.<br>The printer returns:<br>0 = No<br>1 = Yes                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 31  | Read last sent ACK, NAK, or CAN character in the MUSE protocol. | This parameter allows you to read the last control character sent from the MUSE protocol (special applications).<br>The printer returns one of the following alternatives:<br>NUL<br>ACK<br>NAK<br>CAN                                                                                                                                                                                                                                                                                                                                                                                              |
| 32  | Read odometer value                                             | Returns the length of media feed past the printhead (the PD41 odometer cannot distinguish prinheads, so it counts length for the platform).<br>Resolution: 10 meters.                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 33  | Read DSR condition on “uart1:”                                  | Allows you to read the DSR (Data Send Ready) condition on “uart1:”. Not available on the PD41.<br>The printer returns:<br>0 = No<br>1 = Yes                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34  | Read or Set positioning mode for TrueType characters            | This parameter allows you to select one of three modes for the positioning of TrueType characters and also to read for which mode the printer is set.<br>The modes are:<br>0 = Standard mode (default) <ul style="list-style-type: none"><li>• This mode was introduced with Intermec Fingerprint 7.2.</li></ul> 1 = Compatible mode <ul style="list-style-type: none"><li>• This mode is compatible with Intermec Fingerprint 7.xx earlier than version 7.2.</li></ul> 2 = Adjusted mode <ul style="list-style-type: none"><li>• This mode was introduced with Intermec Fingerprint 7.2.</li></ul> |

**SYSVAR Variables (continued)**

| No. | Description                          | Notes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----|--------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 35  | Setup saving (non-volatile/volatile) | This parameter allows you to decide whether a change in the printer's setup is to be saved as a file (that is effective after a reboot or power down) or not be saved (volatile). You can also read for which alternative the printer is set. Note that the SYSVAR(35) setting at the moment when the new setup is entered decides whether it will be saved or not.<br><br>The alternatives are:<br>0 = Setup saved to file (non-volatile) Default<br>1 = Setup not saved to file (volatile)                           |
| 36  | Print changes of program modes       | This parameter is used with the Fingerprint debugger and controls whether changes of program modes should be printed to the Debug Standard Out port (see DBSTDIO).<br><br>The options are:<br>0 = Disable printout (default)<br>1 = Enable printout                                                                                                                                                                                                                                                                    |
| 37  | Set minimum gap length               | The media may have perforations or marks that are not intended to be interpreted as gaps or black marks by the LSS. Using this SYSVAR parameters, it is possible to make the LSS ignore gaps or marks that are shorter than a specified value. (In this context, long and short are related to the media feed direction.) The minimum gap length is specified in dots within a range of 1-32. Default value is 1 mm (0.039 inches). Note that SYSVAR(37) affects PRINTFEED and FORMFEED. For TESTFEED, see SYSVAR(38). |
| 38  | Obsolete                             | This parameter is obsolete and has no effect, even if it does not cause an error if used.                                                                                                                                                                                                                                                                                                                                                                                                                              |

**SYSVAR Variables (continued)**

| No. | Description                                         | Notes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-----|-----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 39  | Enable/disable slack compensation-Force startadjust | <p>Slack compensation:</p> <p>Label slack compensation is a method of eliminating slack in the belts after having fed the media back. At a negative FORMFEED, the printer will pull back the media slightly more than specified by the FORMFEED statement and then feed the media forward the same distance. If slack compensation is enabled, and FORMFEED -100 is specified, the printer will pull back the media for example -112 dots and then feed the media forward +12 dots to take out the slack.</p> <p>Force Startadjust:</p> <p>Performing a positive FORMFEED will normally only feed out paper even if startadjust is set to a negative value. Using narrow labels and liner rewinding may cause the printout position to differ on the next printed label after a FORMFEED. Forcing a startadjust when doing FORMFEED improves printout precision in such cases.</p> <p>The options are:</p> <p>0 = Disable slack compensation and disable force startadjust.</p> <p>1 = Enable slack compensation (default).</p> <p>2 = Force startadjust at FORMFEED.</p> <p>3 = Force startadjust at FORMFEED and enable slack compensation.</p> |
| 40  | Not implemented                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 41  | “Next label not found” at predefined feed length    | The automatic detection of the error condition “Next label not found” (error 1031) by the label stop sensor can be overridden by specifying a fixed length in dots. The length should preferably correspond to at least the distance between the tops of two consecutive labels. During printing, error 1031 occurs if the media does not come loose from the core (media glued to core) or if a label is missing on the liner. Especially useful for short labels (10–40 mm/0.4–1.5 inches long). Default value is 0.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 42  | Stop media feed in the middle of label gaps         | <p>0 = The media feed stops so the middle of a 3 mm (0.12 in) gap becomes aligned with the tear bar when using labels (w gaps). This is the default setting.</p> <p>1 = The media feed stops so the middle of the gap becomes aligned with the tear bar, regardless of gap size.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

**SYSVAR Variables (continued)**

| No. | Description                                                                                    | Notes                                                                                                                                                                                                                                                                                                                                                                                                          |
|-----|------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 43  | Enable/disable file name conversion                                                            | File name conversion means that lowercase characters will be converted to uppercase and the extension .PRG will be added if an extension is missing.<br>0 = File name conversion is enabled (default)<br>1 = File name conversion is disabled.                                                                                                                                                                 |
| 44  | Enable/disable filtering of NUL characters in background communication.                        | SYSVAR(44) controls the filtering of NUL characters in background communication (see COMBUFS\$).<br>0 = Enables filtering (default)<br>1 = Disables filtering                                                                                                                                                                                                                                                  |
| 45  | Read printhead resolution                                                                      | SYSVAR(45) returns the resolution of the printhead expressed in dots per inch (dpi).                                                                                                                                                                                                                                                                                                                           |
| 46  | Read status of paper low sensor                                                                | 0 = Indicates that the diameter of the media supply is larger than specified in the Setup Mode.<br>1 = Indicates that the detected diameter of the media supply roll is equal or less than the diameter specified in the Setup Mode (Media/Paper/Low Diameter). The error condition 1084 "Paper low" will occur.<br>This error does not stop the printing, but interrupts any program that does not handle it. |
| 47  | Enable/disable use of startadjust-stopadjust together with positive and/or negative formfeeds. | 0 = Use of startadjust/stopadjust OR negative positive Formfeed values (default).<br>1 = Enable the use of startadjust/stopadjust values together with negative/positive Formfeed values.                                                                                                                                                                                                                      |
| 48  | Enable/disable bidirectional direct protocol                                                   | 0 = Disables use of direct commands (default)<br>1 = Scans stdIN channel for direct commands. Can only be set in the Direct Protocol.                                                                                                                                                                                                                                                                          |
| 49  | Set temporarily lower speed after negative startadjust.                                        | This is used to avoid hard stretches at the beginning of a label with a full roll. Specify the percentage of the negative startadjust value that should be fed slower (70 mm/s). For example SYSVAR(49)=150 will cause the printer to run slower for 150% of the negative startadjust value, before speeding up.                                                                                               |
| 50  | Set lower speed for given length after lowering printhead.                                     | To avoid hard stretches after loading new media roll, this can lower the speed (to 70 mm/s) for the specified length after lowering the printhead. The value is given in dots. Default is 0.                                                                                                                                                                                                                   |
| 51  | Set enabled limit for function of SYSVAR(49) and (50).                                         | Set the value in meters that values for SYSVAR(49) and SYSVAR(50) will be in effect after lowering the printhead.                                                                                                                                                                                                                                                                                              |
| 53  | Set or read the highest allowed outer diameter (in mm) of the ribbon supply.                   | Range is 40 to 1000 mm. Default is 83.<br>Note that a larger value causes out of ribbon detection to require a longer feed.                                                                                                                                                                                                                                                                                    |

**Examples** Reading the value of a system variable, in this case the transfer ribbon sensor:

```
PRINT SYSVAR (23)
```

Setting the value of a system variable. In this case verbosity is disabled:

```
SYSVAR(18) = 0
```

## TAGFIELD

**Purpose** Statement defining a field in the RFID tag memory area available for RFID operations.

**Syntax** TAGFIELD [<sexp1>,] <sexp2> [, <sexp3> [, <sexp4>]]

where:

<sexp1> is the optional name of the field to make available.

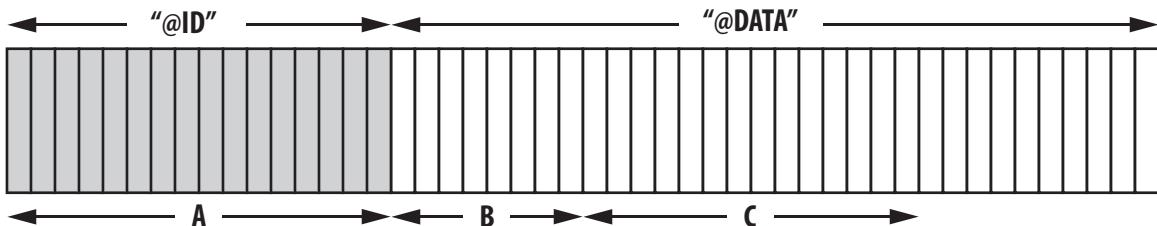
<sexp2> is the name of the segment to make available. Allowed values are described in the next table.

| Tag Type                       | Values for <sexp2>                                                                                                                              |
|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| EPC Class 1 Version 1          | “@ID”: Tag identification segment for EPC data. Default.                                                                                        |
| ISO 18000-6B or EPC UCode 1.19 | “@ID”: Tag identification segment (default)<br>“@DATA”: Optional data segment<br>“@ALL”: Tag's complete memory structure                        |
| EPC Class 1 Gen 2              | “@RESERVED”: Access and kill password segment.<br>“@EPC”: EPC specific data (default)<br>“@TID”: Tag identifier.<br>“@USER”: User defined data. |

<sexp3> is the field starting byte in the chosen segment. Default varies according to RFID tag standard and segment in use.

<sexp4> is the length of the field in bytes. Default varies according to the RFID tag standard and EPCGlobal tag format in use.

**Remarks** The TAGFIELD command is used to specify the field available for subsequent RFID commands, such as TAGWRITE or TAGREAD. The memory structure of RFID depends on the tag type. Class 1 tags only allow the “@ID” segment, where EPC data is stored. The ISO18000-6B, or EPC UCode 1.19 tags contains two segments, “@ID” and “@DATA”. Gen 2 RFID tags have four segments, “@RESERVED”, “@EPC”, “@TID”, and “@USER”. For more information, see Chapter 6, “RFID Tag Formats.”



The *<nexp3>* and *<nexp4>* parameters specify a subset of a segment as the currently available field. It is possible to name the field to use with *<sexp1>*. This name may not start with an @-character. A possible representation of tag segments can be seen in the illustration.

The default values for *<nexp1>* and *<nexp2>* may be modified by a subsequent TAGFORMAT command (see Chapter 6). Values for *<nexp3>* and *<nexp4>* do not normally need to be set when writing EPC information, as these are automatically adjusted for the EPC tag format specified. The only exception is ISO 18000-6B tags. For more information, see Chapter 6, “RFID Tag Formats.”

The TAGFIELD command resets the TAGFORMAT to default. Related commands are TAGFORMAT, TAGREAD and TAGWRITE.

**Examples** The fields A, B, and C in the illustration above are defined by:

```
A: TAGFIELD "@ID"
B: TAGFIELD "@DATA", 0, 8
C: TAGFIELD "C", "@DATA", 8, 14
```

The field name “C” can then be reused:

```
C: TAGFIELD "C"
```

## TAGFORMAT

**Purpose** Statement specifying the format of the data to be read from or written to an RFID tag.

**Syntax** TAGFORMAT*<sexp>*

where *<sexp>* is the format of the data. Available formats are:

|         |                                                                                                                                                                                          |
|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| “ASCII” | 8-bit ASCII string.                                                                                                                                                                      |
| “HEX”   | (Default) Hexadecimal string. Values 0-9 and a-f allowed. Hex characters must be entered in pairs.                                                                                       |
| “NUM”   | Integer. Valid range is 0 to 2147483647. Always uses 4 bytes to represent data, unless a smaller field has been defined, and the number fits in that field. Not allowed for Class1 tags. |

The following formats are supported by EPCglobal tags only:

|            |                                                                   |
|------------|-------------------------------------------------------------------|
| “SGTIN-64” | “Filter”, “Company Prefix”, “Item Reference”, “Serial Number”     |
| “SGTIN-96” | “Filter”, “Company Prefix”, “Item Reference”, “Serial Number”     |
| “SSCC-64”  | “Filter”, “Company Prefix”, “Serial Reference”                    |
| “SSCC-96”  | “Filter”, “Company Prefix”, “Serial Reference”                    |
| “SGLN-64”  | “Filter”, “Company Prefix”, “Location Reference”, “Serial Number” |

|             |                                                                              |
|-------------|------------------------------------------------------------------------------|
| “SGLN-96”   | “Filter”, “Company Prefix”, “Location Reference”, “Serial Number”            |
| “GRAI-64”   | “Filter”, “Company Prefix”, “Asset Type”, “Serial Number”                    |
| “GRAI-96”   | “Filter”, “Company Prefix”, “Asset Type”, “Serial Number”                    |
| “GIAI-64”   | “Filter”, “Company Prefix”, “Individual Asset Reference”                     |
| “GIAI-96”   | “Filter”, “Company Prefix”, “Individual Asset Reference”                     |
| “GID-96”    | “General Manager Number”, “Object Class”, “Serial Number”                    |
| “USDOD-64”  | “Filter”, “Government Managed Identifier”, “Serial Number”                   |
| “USDOD-96”  | “Filter”, “Government Managed Identifier”, “Serial Number”                   |
| “EPC-HEX64” | Hex values for all bits on a 64-bit tag's memory area.                       |
| “EPC-HEX96” | Hex values for all bits on a 96-bit tag's memory area.                       |
| “EPC-URN”   | Uniform Resource Name string. Standardized format for entry of EPC identity. |

**Remarks** The TAGFORMAT command specifies the format of the data to be read from or written to a tag with a subsequent TAGREAD or TAGWRITE command. The format applies to the field defined by the most recent TAGFIELD command.

More information on tag formats can be found in Chapter 6.

TAGFORMAT is reset to default by a TAGFIELD command. Related commands are TAGFIELD, TAGREAD and TAGWRITE.

#### Examples

```

10 FILTER$ = "3"
20 PREFIX$ = "0614141"
30 ITEM$ = "100734"
40 SERIAL$ = "2"
50 TAGFIELD "@EPC"
60 TAGFORMAT "SGTIN-96"
70 TAGWRITE FILTER$, PREFIX$, ITEM$, SERIAL$

80 TAGFIELD "@ID"
90 TAGFORMAT "EPC-URN"
100 TAGREAD MyURI$
110 PRINT MyURI$
RUN

```

resulting in:

```

urn:epc:sgtin-96:3.0614141.100734.2

10 TAGFIELD "@USER",10,4
20 TAGFORMAT "ASCII"
30 TAGWRITE "RFID"
40 TAGREAD A$
50 PRINT A$
RUN

```

resulting in:

RFID

## **TAGPROTECT**

**Purpose** Statement used to protect tag data from being overwritten.

**Syntax** TAGPROTECT<*sexp1*>

where <*sexp1*> specifies the tag protection level. TAGPROTECT is supported only by EPCglobal Class 1, ISO 18000-6B, and EPC UCode v1.19 tags. Available options are:

“ON” Tag is protected.

“OFF” Tag remains unprotected.

**Remarks** The TAGPROTECT command protects the tag from being overwritten by an RFID operation. The protection is permanent. The command applies to the latest defined field and is executed at the next TAGWRITE operation, since TAGPROTECT does not align the tag over the antenna.

Related commands are TAGFIELD, TAGFORMAT and TAGWRITE.

**Examples**

```
10 TAGFIELD "@ID"
20 TAGFORMAT "SSCC-64"
30 TAGPROTECT "ON"
40 TAGWRITE "1", "12345", "123456"
```

## **TAGREAD**

**Purpose** Statement used to read an RFID tag field.

**Syntax** TAGREAD<*nvar*>|<*svar1*>[,<*svar2*>,<*svar3*>,<*svar4*>]

where:

<*nvar*> is the numeric variable that stores the data. Only used with the “NUM” field in TAGFORMAT.

<*svarN*> is the string variable that stores the data. The number of arguments depends on TAGFORMAT.

**Remarks** The TAGREAD command reads the data from the field specified by the latest TAGFIELD command into the variable <*nvar*> or variables <*svarN*>.

The format of the data is defined by the latest TAGFORMAT command. If you state a numeric variable <*nvar*>, an error is returned unless the data is in the “NUM” format.

Related commands are TAGFIELD, TAGFORMAT and TAGWRITE.

**Examples** Using numeric variables:

```
10 TAGFIELD "@DATA", 2, 2
20 TAGFORMAT "NUM"
30 TAGWRITE 19562
40 TAGFIELD "@DATA", 2, 2
```

```
50 TAGFORMAT "NUM"
60 TAGREAD A%
RUN
```

results in:

```
19562
```

Reading a SGTIN-96 tag from a Gen 2 tag:

```
10 TAGFIELD "@EPC"
20 TAGFORMAT "SGTIN-96"
30 TAGREAD FILTER$, PREFIX$, ITEM$, SERIAL$
40 PRINT FILTER$, PREFIX$, ITEM$, SERIAL$
RUN
```

results in:

```
3 0614141 100734 2
```

## TAGWRITE

**Purpose** Statement used to write to an RFID tag field.

**Syntax** TAGWRITE< nvar > | < svar1 > [, < svar2 >, < svar3 >, < svar4 >]

where:

< nvar > is the numeric variable to be written. Only used with the “NUM” field in TAGFORMAT.

< svarN > is the string variable to be written. The number of arguments depends on TAGFORMAT.

**Remarks** TAGWRITE writes the data from < nvar > or < svar1 > to the field specified by the latest TAGFIELD command. The format of the data is specified by the latest TAGFORMAT command. If the data written is shorter than the field, the field is padded with zeroes.

The exception to this rule occurs in some EPC tag formats where the exact number of digits must be entered, even if it means adding non-significant digits (for example, writing 00234 instead of 234). If the data is too long to fit in the specified field, an error is returned. For more information, see Chapter 6, “RFID Tag Formats.”

Related commands are FORMAT\$, TAGFIELD, TAGFORMAT and TAGREAD.

**Examples** Writing an SGTIN-96 to a Gen 2 tag:

```
10 TAGFIELD "@EPC"
20 TAGFORMAT "SGTIN-96"
30 TAGWRITE "3", "0614141", "100734", "2"
```

Two examples of writing the same SSCC-64 info to an ISO18000-6B tag:

```
10 TAGFIELD "@DATA", 10, 8
20 TAGFORMAT "SSCC-64"
30 TAGWRITE "1", "12345", "123456"
```

```
10 TAGFIELD "@DATA",10,8
20 TAGFORMAT "EPC-URN"
30 TAGWRITE
"urn:epc:tag:sscc-64:1.12345.123456"
```

## TESTFEED

**Purpose** Statement for adjusting the label stop, ribbon end/low and paper low sensors, and RFID module while running the media and ribbon feed mechanisms.

**Syntax** TESTFEED [<nexp>]

where <nexp> (optional) is a feed length in dots.

**Remarks** A TESTFEED statement feeds <nexp> dots while calibrating the label stop/black mark sensor (LSS) for the characteristics of the media presently loaded in the printer. The statement is needed to detect media, gaps, black marks, and out-of-paper conditions and should be done for all media types.

If an RFID module is installed, and RFID ON is set, TESTFEED attempts to identify the RFID tag. This TESTFEED is always performed in SLOW MODE.

If <nexp> is omitted, it is automatically set to 1.5 times the media length specified in the setup. For the TESTFEED to be successful, at least one gap or black mark must pass the LSS. Best results for “Ticket w Mark” are obtained with a <nexp> value of 1200 or any other reasonable number.

When a TESTFEED is executed, the ribbon end/low and paper low sensors are also calibrated (if installed). However, this does not apply when the testfeed is ordered using the testfeed option in Setup Mode.

In the Immediate Mode, a TESTFEED is performed when the <Shift> and <Feed> keys are pressed simultaneously.

In the setup, TESTFEED MODE can be set to SLOW. This might be necessary when using media with pre-printed lines. This is done with a SETUP command or through the printer Setup Mode. When in SLOW mode, TESTFEED samples the media length plus 10 mm. Alternately, the length sampled can be set using the MEDIA,LEN (SLOW MODE) option, the minimum being the number of dots corresponding to 10 mm. This value is ignored when TESTFEED MODE is set to FAST.

Since the TESTFEED is essential for a proper media load, some facility for issuing a TESTFEED statement should be included in all custom-made label-printing programs as seen in the example.

**Example** This program performs a TESTFEED statement when the **Shift** and **Feed** keys are pressed simultaneously on the printer keypad:

```
10 ON KEY (119) GOSUB QTESTFEED
20 KEY (119) ON
30 QLOOP:
40 GOTO QLOOP
.
```

```
1000 QTESTFEED:
1010 TESTFEED
1020 RETURN
```

## TICKS

**Purpose** Returns the elapsed time since the last power up in the printer, expressed in number of “TICKS” (1 TICK = 0.01 sec).

**Syntax** TICKS

**Remarks** TICKS allows you to measure time more exactly than the TIME\$ variable, which cannot handle time units smaller than 1 second. The TICKS counter is reset to zero at power up.

**Example**

```
10 A%=TICKS
20 PRINT A%
RUN
```

resulting in:

```
1081287
```

In this case, the time which has passed since the printer was started is 10812.87 seconds, or 3 hours 12.87 seconds.

## TIME\$

**Purpose** Variable for setting or returning the current time.

**Syntax** **Setting the time**

```
TIME$=<sexp>
```

where <sexp> sets the current time by a 6-digit number specifying Hour, Minute and Second.

**Reading the time**

```
<svar>=TIME$ [(<sexp>)]
```

where:

<svar> returns the current time according to the printer clock.

<sexp> (optional) is a flag “F”, indicating that the time will be returned according to the format specified by FORMAT TIME\$.

**Remarks** This variable works best if a real-time clock circuit (RTC) is fitted on the printer CPU board. The RTC is battery backed-up and keeps track of the time even if the power is turned off or lost.

If no RTC is installed, the internal clock is used. After startup, an error occurs when trying to read the date or time before the internal clock has been manually set using either a DATE\$ or a TIME\$ variable.

If only the date is set, the internal clock starts at 00:00:00, and if only the time is set, the internal clock starts at Jan 01 1980. After setting the internal clock, you can use the DATE\$ and TIME\$ variables the same way as when an RTC is fitted, until a power off or REBOOT causes the date and time values to be lost.

By default, time is always entered and returned as HHMMSS, where:

HH = Hours: Two digits (00-23)

MM = Minutes: Two digits (00-59)

SS = Seconds: Two digit (00-59)

Time is entered as a 24-hour cycle (for example, 8 P.M. is entered as "200000").

The clock is reset at the exact moment when the appending carriage return character is received (when you press the Return key in Immediate Mode and Intermec Direct Protocol), or when the instruction is executed (Programming Mode).

The format for how the printer returns time from a TIME\$("F") variable can be changed using a FORMAT TIME\$ statement.

**Example** This example sets and reads the time, then prints it on the screen of the host:

```
10 TIME$ = "154300"
20 FORMAT TIME$ "HH.MM"
30 PRINT "Time is "+TIME$("F")
RUN
```

resulting in:

```
Time is 15.43
```

## TIMEADD\$

**Purpose** Function returning a new time after a number of seconds have been added to or subtracted from the current time or from a specified time.

**Syntax** TIMEADD\$ ( [<sexp1>, ] <nexp> [ , <sexp2> ] )

where:

<sexp1> is any time given according to the TIME\$ format, which a certain number of seconds should be added to or subtracted from.

<nexp> is the number of seconds to be added to (or subtracted from) the current time, or (optional) the time specified by <sexp1>.

<sexp2> is an optional flag "F", indicating that the time will be returned according to the format specified by FORMAT TIME\$.

**Remarks** The original time (<sexp1>) should always be entered according to the TIME\$ format (HHMMSS), where:

HH = Hours: Two digits (00-23)

MM = Minutes: Two digits (00-59)

SS = Seconds: Two digits (00-59)

Time is entered as a 24-hour cycle (for example, 8 P.M. is entered as "200000").

The number of seconds to be added or subtracted from the original time should be specified as a positive or negative numeric expression respectively.

If no "F" flag is included in the TIMEADD\$ function, the result is returned according to the TIME\$ format.

If the TIMEADD\$ function includes an "F" flag, the result is returned in the format specified by FORMAT TIME\$.

#### **Example 1**

```
10 A%=30
20 B$=TIMEADD$ ("133050",A%)
30 PRINT B$
RUN
```

resulting in:

133120

#### **Example 2**

```
10 TIME$="133050"
20 FORMAT TIME$ "hh.mm.ss pp"
30 A% = -40
40 PRINT TIMEADD$(A%, "F")
RUN
```

resulting in:

01.30.10 pm

## **TIMEDIFF**

**Purpose** Returns the difference between two specified times in number of seconds.

**Syntax** TIMEDIFF (<sexp1>, <sexp2>)

where:

<sexp1> is the first point in time (time 1).

<sexp2> is the second point in time (time 2).

**Remarks** To get the result as a positive value, the two points of time should be entered with the earlier moment (time 1) first and the later moment (time 2) last, as seen the first example.

If the later moment (time 2) is entered first, the resulting value is negative, as seen in the second example.

The time should be entered according to the format HHMMSS, where:

HH = Hours: Two digits (00-23)

MM = Minutes: Two digits (00-59)

SS = Seconds: Two digits (00-59)

Time is entered as a 24-hour cycle (for example 8 P.M. is entered as “200000”).

The resulting difference in seconds is returned.

**Examples** PRINT TIMEDIFF ("133050", "133120")

results in:

30

PRINT TIMEDIFF ("133120", "133050")

results in:

-30

## TRANSFER KERMIT

**Purpose** Statement for transferring data files using KERMIT communication protocol.



**Note:** Tests have shown that Microsoft Windows Terminal versions 3.0 and 3.1 can send files to the printer but are unable to receive a file from the printer.

**Syntax** TRANSFER\_K [ERMIT] <sexp1> [, <sexp2> [, <sexp3> [, <sexp4>]]]

where:

<sexp1> specifies the direction of the transmission by the expression “S” (= send) or “R” (= receive).

<sexp2> (optional) is the name of the file transmitted from the printer. Default is “KERMIT.FILE”.

<sexp3> (optional) specifies the input device as “uart1:”, “uart2:”, “uart3:”, “uart4:”, or “uart5:”. Default is the std IN channel.

<sexp4> (optional) specifies the output device as “uart1:”, “uart2:”, “uart3:”, “uart4:”, or “uart5:”. Default is the std OUT channel.

**Remarks** Kermit is a protocol for serial binary transfer of a complete file, and is included in HyperTerminal and other communication programs. For more information, consult the application program documentation.

TRANSFER KERMIT can only handle a single file at a time. When transmitting files from the printer to the host, carefully observe possible restrictions on the number of characters in the file name that may be imposed by the host operating system.

When receiving a file, you must start the transmission within 30 seconds of completing the TRANSFER KERMIT “R” statement. The printer stores the file in the current directory (“/c”, “tmp:”, or “card1:”). Files cannot be received into “/rom”. If a file in the current directory has the same name as the one to be transferred, the existing file will be replaced by the new file.

Thus, you need to keep track of the files already stored in the current directory (see FILES statement). Give the new file a name not already used by an existing file, unless you want to replace the existing file. Downloaded fonts and images are auto-installed.

**Examples** This example sets up the printer to receive a file on the standard IN channel:

```
TRANSFER KERMIT "R"
```

This example sends “FILE1.TXT” from the printer to the host on a channel other than the standard OUT channel:

```
TRANSFER K "S", "FILE1.TXT", "uart2:", "uart2:"
```

## TRANSFER NET

**Purpose** Statement for transferring files to and from the printer using FTP.

**Syntax** TRANSFER N [ET] <sexp1>, <sexp2> [, <sexp3>]

where:

<sexp1> is the source. If the source is a local file, this file is sent from the printer to the destination specified by <sexp2>. If the source is a URI, this file is fetched from the server, sent to the printer, and stored at the location specified by <sexp2>.

<sexp2> is the destination of the file transfer.

<sexp3> (optional) is an account secret.

**Remarks**

**Limitations**  
TRANSFER NET is not a complete FTP client. It only supports file transfer to and from the printer in binary format. Only one file can be transferred per command. File transfer between two local or two remote files is not supported.

### Local files

A local file is a path to an existing file (when sending from the printer) or to the file that is created (when sending to the printer). If a local file already exists when sending a file to the printer, the existing file is replaced if it is not write-protected. A read-protected file is not sent. If the destination is a local directory, the sent file will get the same name as the source file.

### URIs

Enter a URI in the format:

```
ftp:// [<user>:<password>@] <server> [:port] /<path>
```

Entries inside square brackets [...] are optional. The following default values are used:

- user: anonymous
- password: nopass@<ip address>
- port: 21

If the destination is a URI specifying a directory, the sent file gets the same name as the source file.

**Account Secret**

If the user does not want to reveal his/her username and password in plain text in a Fingerprint program, the account secret option can be used. The account secret holds the secret information and cannot be read by any user. To create an account secret, use the external command RUN “secret.” Listing and deleting accounts secrets are reserved for admin.

**Fonts and Images**

Downloaded fonts and images are auto-installed.

- Example** This example shows how the file README.uploads is fetched from the sunet ftp server and stored as UPLOAD.TXT in the current directory. The default user, password and port number are used.

```
TRANSFER NET "ftp://ftp.sunet.se/README.uploads",
"UPLOAD.TXT"
```

## TRANSFER STATUS

**Purpose** Statement for checking the last TRANSFER KERMIT or TRANSFER ZMODEM operation.

**Syntax** TRANSFER\_\_S [TATUS] <nvar>, <svar>

where:

<nvar> is a five-element one-dimensional numeric array where the elements return:

- 0 Number of packets. (Kermit only)
- 1 Number of NAK's. (Kermit only)
- 2 ASCII value of last status character. (Kermit only)
- 3 Last error. (Kermit and ZMODEM)
- 4 Block check type used. (Kermit only)

<svar> is a two-element one-dimensional string array where the elements return:

- 0 Type of protocol. (“KERMIT” or “ZMODEM”)
- 1 Last file name received.

**Remarks** After a file transfer using the Kermit or ZMODEM protocol has been performed (see TRANSFER KERMIT and TRANSFER ZMODEM statements), you can check how the transfer was performed. Note that the numeric array requires the use of a DIM statement, since the array contains more than four elements.

**Example** 10 TRANSFER KERMIT "R"

```
20 DIM A%(4)
```

```
30 TRANSFER STATUS A%, B$
```

```
40 PRINT A%(0), A%(1), A%(2), A%(3), A%(4)
```

```
50 PRINT B$(0), B$(1)
```

```
.....
```

```
.....
```

## TRANSFER ZMODEM

**Purpose** Statement for transferring data files using ZMODEM communication protocol.

**Syntax** TRANSFER *\_Z [MODEM] <sexp1> [, <sexp2> [, <sexp3> [, <sexp4>]] ]*

where:

<*sexp1*> specifies the direction of the transmission by the expression “S” (= send) or “R” (= receive).

<*sexp2*> (optional) is the name of the file transmitted from the printer (default “ZMODEM.FILE”).

<*sexp3*> (optional) specifies the input device as “uart1:”, “uart2:”, “uart3:”, “uart4:”, or “uart5:”. Default is the std IN channel.

<*sexp4*> (optional) specifies the output device as “uart1:”, “uart2:”, “uart3:”, “uart4:”, or “uart5:”. Default is the std OUT channel.

**Remarks** ZMODEM is a protocol for serial transfer of a complete file. For more information on the ZMODEM protocol, please refer to [www.omen.com](http://www.omen.com). Related instructions are the external commands RZ (receive data using the ZMODEM protocol) and SZ (send data using the ZMODEM protocol).

TRANSFER ZMODEM handles a single file at a time.

When transmitting files from the printer to the host, carefully observe possible restrictions on the number of characters in the file name that may be imposed by the operating system of the host. When receiving a file, you must start the transmission within 30 seconds of completing the TRANSFER ZMODEM “R” statement. The printer stores the file in the current directory “/c”, “tmp:”, or “card1:”. Files cannot be received into “/rom”. If a file in the current directory has the same name as the one to be transferred, the existing file will be replaced by the new file.

Thus, you need to keep track of the files already stored in the current directory (see FILES statement). Before transfer, give the new file a name not already occupied by an existing file, unless you want to replace the existing file. If you use TRANSFER ZMODEM to download a font or image file, the font or image is automatically installed after the downloading is completed with no need for a reboot.

**Examples** This example sets up the printer to receive a file on the standard IN channel:

```
TRANSFER ZMODEM "R"
```

This example sends the file “FILE1.TXT” from the printer to the host on a channel other than the standard OUT channel:

```
TRANSFER Z "S", "FILE1.TXT", "uart2:", "uart2:"
```

## TRANSFER\$

- Purpose** Executes a transfer from source to destination as specified by a TRANSFERSET statement.
- Syntax** TRANSFER\$ (<nexp>)  
where <nexp> is the character time-out in ticks (10 ms).
- Remarks** The TRANSFER\$ function executes the transfer from source to destination as specified by the TRANSFERSET statement. It also checks the transfer and breaks it if no character has been transmitted before the specified time-out has expired, or if any break character (as specified by the break character string in the TRANSFERSET statement) is encountered.
- If the transmission was interrupted because a character in the break set was encountered, that character is returned.
- If the transmission was interrupted because of a time-out error, an empty string is returned.
- If the transmission was interrupted because of the reception of a character on any other communication channel than the source (as specified by TRANSFERSET statement), an empty string is returned.
- Example** In this example, the transfer is executed by the TRANSFER\$ function in line 60, and possible interruptions are indicated by a break character or empty string (" ") in the string variable C\$.

```
10 OPEN "LABEL1.PRG" FOR INPUT AS #1
20 OPEN "UART1:" FOR OUTPUT AS #2
30 A$=CHR$(13)
40 B$=CHR$(10)
50 TRANSFERSET #1, #2, A$+B$
60 C$=TRANSFER$(100)
.....
.....
.....
```

## TRANSFERSET

- Purpose** Statement for entering setup for the TRANSFER\$ function.
- Syntax** TRANSFERSET [#] <nexp1>, [#] <nexp2>, <sexp> [, <nexp3>]  
where:
- # (optional) number sign.
- <nexp1> is the number of the source (the file or device OPENed for input).
- <nexp2> is the number of the destination file (the file or device OPENed for output or append).
- <sexp> is a set of break characters.

*<nexp3>* optionally enables or disables break on any other channel than the source:

*<nexp>* = 0 Break disabled  
*<nexp>* \_, 0 Break enabled

Default is the standard I/O with no break characters. Break on any other channel is enabled.

**Remarks** This statement sets up the transfer of data from a file or device OPENed for input to another file or device OPENed for output or append. The transfer is interrupted if any character in a string of break characters specified in this statement is encountered (optionally on another specified channel). The actual transfer is executed by means of a TRANSFER\$ function that also returns the break character that caused any possible interruption.

**Example** In this example, the data transfer from a file in the current directory to an external device connected to the communication port "uart1:" is interrupted as soon as a carriage return or a line feed character is encountered in the file.

```
10 OPEN "LABEL1.PRG" FOR INPUT AS #1
20 OPEN "uart1:" FOR OUTPUT AS #2
30 A$=CHR$(13)
40 B$=CHR$(10)
50 TRANSFERSET #1, #2, A$+B$
60 C$=TRANSFER$(100)
.....
.....
.....
```

## TRON/TROFF

**Purpose** Statements enabling/disabling tracing of the program execution.

**Syntax** TRON | TROFF

where TRON enables tracing and TROFF disables tracing (default).

**Remarks** This statement is useful for debugging purposes. When tracing is enabled, each line number of the program is displayed on the screen within parentheses as the execution goes on.

Tracing is disabled when a TROFF statement is executed.

**Example**

```
10 PRINT "HELLO"
20 INPUT"Enter Text"; A$
30 PRINT A$
TRON
RUN
```

resulting in:

```
(10) HELLO
(20) Enter test? (Operator enters "WORLD")
(30) WORLD
```

## **VAL**

**Purpose** Returns the numeric representation of a string expression.

**Syntax** `VAL (<sexp>)`

where `<sexp>` is the string expression from which the numeric representation is returned.

**Remarks** VAL is the complementary function for STR\$. VAL ignores space characters from the argument string to determine the result.

If the first character in the string expression is anything else but a digit, a plus sign, or a minus sign, the VAL function returns the value 0.

**Example** In this example, the values of the string variables A\$ and B\$ are read and assigned to the numeric variables A% and B%:

```
10 A$="123, MAIN STREET"
20 A%=VAL (A$)
30 B$="PHONE 123456"
40 B%=VAL (B$)
50 PRINT A$
60 PRINT A%
70 PRINT B$
80 PRINT B%
RUN
```

resulting in:

```
123, MAIN STREET
123
PHONE 123456
0
```

## **VERBON/VERBOFF**

**Purpose** Statements for specifying the verbosity level of the communication from the printer on the standard OUT channel (serial communication only).

**Syntax** `VERBON | VERBOFF`

where VERBON enables all verbosity levels (default) and VERBOFF disables all verbosity levels.

**Remarks** **VERBON**

By default, when a character is received on the standard IN channel (see SETSTDIO statement), the corresponding character is echoed back on the standard OUT channel. As the serial channel “uart1:” is by default selected as the standard IN and OUT channel, when you enter a character on the keyboard of the host the same character appears on the screen after being transmitted to the printer and back.

When an instruction is successfully executed, “Ok” appears on the screen. Otherwise an error message is returned. Since this requires two-way communication, verbosity has no meaning when using the parallel “centronics:” communication protocol.

VERBON corresponds to SYSVAR(18) = -1.

Other verbosity levels can be selected using SYSVAR(18), and the type of error message can be selected using SYSVAR (19).

### **VERBOFF**

All responses are suppressed, which means that no characters or error messages are echoed back. VERBOFF statements do not affect question marks or prompts displayed as a result of an INPUT statement.

Instructions like DEVICES, FILES, FONTS, IMAGES, LIST, and PRINT also work normally.

VERBOFF corresponds to SYSVAR(18) = 0.

**Example** This example shows how VERBOFF suppresses the printing of INPUT data in lines 20 and 40 during the actual typing on the host, and VERBON allow printing of the resulting string variables on the screen:

```
10 FOR Q%=1 TO 6
20 VERBOFF:INPUT "", A$
30 VERBON:PRINT A$;
40 VERBOFF:INPUT "", B$
50 VERBON
60 C$=SPACE$(25-LEN(A$))
70 PRINT C$+B$
80 NEXT Q%
90 END
```

## **VERSION\$**

**Purpose** Returns the firmware version, printer family, or type of CPU board.

**Syntax** VERSION\$ [ (<nexp>) ]

where <nexp> (optional) is the type of information to be returned:

- 0 Version of firmware (default)
- 1 Printer family
- 2 Type of CPU board

**Remarks** The name of the firmware depends on whether or not the printer is running in Immediate or Programming Mode, or in Intermec Direct Protocol. The printer family is returned as one of the following:

```
PD41
PF2i
PF4i
PM4i
PX4i
PX6i
```

The CPU board type is returned as a string of text such as:

```
hardware version 4.0
```

**Example 1** PRINT VERSION\$(0)

returns for example:

```
Fingerprint 8.70.0
```

**Example 2** PRINT VERSION\$(1)

returns for example:

```
PF4i
```

**Example 3** PRINT VERSION\$(2)

returns for example:

```
hardware version 4.0
```

## **WEEKDAY**

**Purpose** Returns the weekday of a specified date.

**Syntax** WEEKDAY(<sexp>)

where <sexp> is the date in DATE\$ format from which the weekday will be returned.

**Remarks** This function returns the weekday as a numeric constant:

- 1 Monday
- 2 Tuesday
- 3 Wednesday
- 4 Thursday
- 5 Friday
- 6 Saturday
- 7 Sunday

The date should be entered according to the syntax for the DATE\$ variable that is in the following order:

YY = Year: Last two digits (for example 2007 = 07)

MM = Month: Two digits (01-12)

DD = Day: Two digits (01-28|29|30|31)

Example: December 1, 2007 is entered as "071201".

The built-in calendar corrects illegal values for the years 1980-2048, for example the illegal date 031232 will be corrected to 040101.

**Example** In this example the weekday for the current date is printed on the screen of the host (another way is to use NAME WEEKDAY\$ statement and WEEKDAY\$ function):

```
10 B$=DATE$
20 A% = WEEKDAY (B$)
30 IF A% = 1 THEN PRINT "MONDAY"
40 IF A% = 2 THEN PRINT "TUESDAY"
50 IF A% = 3 THEN PRINT "WEDNESDAY"
60 IF A% = 4 THEN PRINT "THURSDAY"
70 IF A% = 5 THEN PRINT "FRIDAY"
80 IF A% = 6 THEN PRINT "SATURDAY"
```

```
90 IF A% = 7 THEN PRINT "SUNDAY"
```

```
RUN
```

resulting in:

```
THURSDAY
```

## WEEKDAY\$

**Purpose** Returns the name of the weekday from a specified date.

**Syntax** WEEKDAY\$ (<sexp>)

where <sexp> is the date for which the name of the weekday is returned.

**Remarks** This function returns the name of the weekday from a list of names specified by a NAME WEEKDAY\$ statement or, if the name is missing, the full English name in lowercase characters (for example, “friday”).

The date should be entered according to the syntax for the DATE\$ variable:

YY = Year: Last two digits (for example 2007 = 07)

MM = Month: Two digits (01-12)

DD = Day: Two digits (01-28|29|30|31)

Example: December 1, 2007 is entered as “071201”.

The built-in calendar corrects illegal values for the years 1980-2048, for example the illegal date 031232 will be corrected to 040101.

**Example** This example shows how to make the printer return the name of the weekday as a three-letter English abbreviation in connection with a formatted date:

```
10 FORMAT DATE$ " , MM/DD/YY"
20 DATE$="071201"
30 NAME WEEKDAY$ 1, "Mon"
40 NAME WEEKDAY$ 2, "Tue"
50 NAME WEEKDAY$ 3, "Wed"
60 NAME WEEKDAY$ 4, "Thu"
70 NAME WEEKDAY$ 5, "Fri"
80 NAME WEEKDAY$ 6, "Sat"
90 NAME WEEKDAY$ 7, "Sun"
100 PRINT WEEKDAY$ (DATE$) + DATE$ ("F")
RUN
```

resulting in:

```
MON, 12/01/07
```

## WEEKNUMBER

**Purpose** Returns the number of the week for a specified date.

**Syntax** WEEKNUMBER (<*sexp*> [, <*nexp*> ])

where:

<*sexp*> is the date for which the week number will be returned. Range is 1 to 53.

<*nexp*> specifies the calculating function as described in the next table:

| < <i>nexp</i> > = Week #1 starts on: |                                                                                                                                                                                                                                                                                                                                                 |
|--------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 (default)                          | Per ISO 8601 (European standard): <ul style="list-style-type: none"> <li>week #1 will start on the last Monday at or before the New Year, if January 1 occurs on a Monday, Tuesday, Wednesday, or Thursday.</li> <li>week #1 will start on the first Monday after the New Year, if January 1 occurs on a Friday, Saturday, or Sunday</li> </ul> |
| 1                                    | Sunday in the first week with 7 days in the actual year                                                                                                                                                                                                                                                                                         |
| 2                                    | January 1st, with each following week starting on a Sunday                                                                                                                                                                                                                                                                                      |
| 3                                    | Monday in the first week with 7 days in the actual year                                                                                                                                                                                                                                                                                         |
| 4                                    | January 1st, with each following week starting on a Monday                                                                                                                                                                                                                                                                                      |
| 5                                    | Tuesday in the first week with 7 days in the actual year                                                                                                                                                                                                                                                                                        |
| 6                                    | January 1st, with each following week starting on a Tuesday                                                                                                                                                                                                                                                                                     |
| 7                                    | Wednesday in the first week with 7 days in the actual year                                                                                                                                                                                                                                                                                      |
| 8                                    | January 1st, with each following week starting on a Wednesday                                                                                                                                                                                                                                                                                   |
| 9                                    | Thursday in the first week with 7 days in the actual year                                                                                                                                                                                                                                                                                       |
| 10                                   | January 1st, with each following week starting on a Thursday                                                                                                                                                                                                                                                                                    |
| 11                                   | Friday in the first week with 7 days in the actual year                                                                                                                                                                                                                                                                                         |
| 12                                   | January 1st, with each following week starting on a Friday                                                                                                                                                                                                                                                                                      |
| 13                                   | Saturday in the first week with 7 days in the actual year                                                                                                                                                                                                                                                                                       |
| 14                                   | January 1st, with each following week starting on a Saturday                                                                                                                                                                                                                                                                                    |

### Remarks

The date should be entered according to the syntax for the DATE\$ variable that is in the following order:

YY = Year: Last two digits (for example 2007 = 07)

MM = Month: Two digits (01-12)

DD = Day: Two digits (01-28|29|30|31)

Example: December 1, 2007 is entered as “071201”.

The built-in calendar corrects illegal values for the years 1980-2048. For example, the incorrect date 031232 is corrected to 040101.

**Examples** This example returns the week number of December 29, 2007 using calculating function 2:

```
PRINT WEEKNUMBER ("071229", 2)
```

resulting in:

53

## WHILE...WEND

**Purpose** Statement for executing a series of statements in a loop providing a given condition is true.

**Syntax**

```
WHILE <nexp>
 <stmt> [. . . <stmt>]
WEND
```

where:

<nexp> is a numeric expression that is either TRUE (-1) or FALSE (0).

<stmt> is a statement, or a list of statements on separate lines, that are executed provided <nexp> is TRUE.

**Remarks** If <nexp> is TRUE, all following statements are executed successively until a WEND statement is encountered. The program execution then goes back to the WHILE statement and repeats the process, provided <nexp> still is TRUE.

If <nexp> is FALSE, the execution resumes at the statement following the WEND statement.

WHILE...WEND statements can be nested. Each WEND matches the most recent WHILE statement.

**Example** In this example, the WHILE...WEND loop is executed only if the character "Y" (ASCII 89 dec.) is entered on the host keyboard.

```
10 B%=0
20 WHILE B%<>89
30 INPUT "Want to exit? Press Y=Yes or N=No ",A$
40 B%=ASC(A$)
50 WEND
60 PRINT "The answer is Yes"
70 PRINT "You will exit the program"
80 END
RUN
```

resulting in:

```
Want to exit? Press Y=Yes or N=No N
Want to exit? Press Y=Yes or N=No Y
The answer is Yes
You will exit the program
```

## XORMODE ON/OFF

**Purpose** Statement for enabling or disabling the xor/flip mode of Fingerprint in connection with graphical operations.

**Syntax** XORMODE ON|OFF

**Remarks** When XORMODE is set ON, dots are reversed (as opposed to set) by all graphical operations except bar codes. For example, if two black lines cross, the intersection is white. If XORMODE is OFF, the intersection is black.

Default is XORMODE OFF. XORMODE is automatically set to default when a PRINTFEED statement is executed or a Fingerprint program has been successfully run.

**Example** The following program illustrates the difference between XORMODE ON and XORMODE OFF. The two lines to the left are drawn with XORMODE disabled and the lines to the right with XORMODE enabled.

```
10 XORMODE OFF
20 PRPOS 0,50
30 PRLINE 300,30
40 DIR 4
50 PRPOS 100,0
60 PRLINE 200,30
70 XORMODE ON
80 DIR 1
90 PRPOS 400,50
100 PRLINE 300,30
110 DIR 4
120 PRPOS 500,0
130 PRLINE 200,30
140 PRINTFEED
RUN
```

## External Command; Account Secret

**Purpose** Creating an Account Secret for use with the TRANSFER NET statement.

**Syntax 1** secret [-t]<application> <name> <string> (create a secret)

**Syntax 2** For user /admin/ two more functions are available:

```
secret -rm <application> <name> (delete a secret)
secret -l (list all secrets)
```

where:

|               |                                                     |
|---------------|-----------------------------------------------------|
| -t            | (optional) Temporary and is removed at next reboot. |
| -rm           | Remove                                              |
| -l            | List                                                |
| <application> | ftp                                                 |
| <name>        | is the name of secret.                              |
| <string>      | is the secret string.                               |

**Remarks** The user may not want to have his/her username and password in plain text in a Fingerprint program. Instead of writing the account info in the URI, the <account secret> parameter of TRANSFER NET can be used. The account secret holds the secret information and cannot be read by any user.

For the application ftp, <string> should have the following structure:

<user>[:passwd]@<server>

where <server> is:

<server name>|<server name>:<port number>|\*

If <server> is set to “\*”, the server name supplied in the URI is used. This means that this account secret can be used with any server.

Account info (user, password, server name and port number) can be stated in both the account secret and the URI. Any parameter supplied in the account secret has precedence over parameters supplied in the URI. So, for example, if the URI states that port 25 should be used and the account secret says port 21, then port 21 is used.

**Example 1** This example creates a temporary account secret, my\_account, and uses it to send the file “beta1.bin” in directory “/tmp” to the server TheServer. The sent file gets the name beta1.bin and is put in the home directory of myusername on TheServer.

First of all, set SYSVAR(43) to 1 to avoid file name conversion:

```
SYSVAR(43)=1
RUN "secret -t ftp my_account myusername:mypassword@*"
TRANSFER NET "/tmp/beta1.bin", "ftp://TheServer/", "my_account"
```

**Example 2** This example creates a permanent account secret, mrbill, and use it to send the file MY.TXT in the current directory to the server MrBill. The file YOUR.TXT will be put in the directory /absolute/path/.

```
RUN "secret ftp mrbill mrbill_username:mrbill_password@"
MrBill"
TRANSFER NET "MY.TXT", "ftp://MrBill//absolute/path/
YOUR.TXT", "mrbill"
```

The server name in the URI will not be used since there is a server name in the account secret. Hence the two following command lines will have the same effect:

```
TRANSFER NET "MY.TXT", ftp://What_ever//absolute/path/
YOUR.TXT, "mrbill"
```

and

```
TRANSFER NET "MY.TXT", "ftp:///absolute/path/
YOUR.TXT", "mrbill"
```

## External Command; ZMODEM

**Purpose** External commands for receiving and sending data using the ZMODEM protocol.

**Syntax 1** RUN "rz [<switches>] [<filename>]" (receive data)

where:

- |                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>&lt;switches&gt;</i> | -c Forces no crash recovery, even if sender requests ZCRESUM (resume interrupted file transfer).<br>-e Print last error to std OUT channel.<br>-l [<logfile>] Send verbose output to logfile. Default logfile name is "tmp:.zmodemlog".<br>-r If ZMCLOB is not set and the file already exists, replace file if the transfer is successful.<br>-v [<level>] Set verbosity level. Level is a decimal number. Default level is 1.<br>-u Translate file name to uppercase. If a filename is given as parameter, no translation is done. |
| <i>&lt;filename&gt;</i> | (optional) is the name to which the file will be saved.                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

**Syntax 2** RUN "sz [<switches>] [<filename>]" (send data)

where:

- |                         |                                                                                                                                                                              |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>&lt;switches&gt;</i> | -l [<logfile>] Send verbose output to logfile. Default logfile name is "tmp:.zmodemlog".<br>-v [<level>] Set verbosity level. Level is a decimal number. Default level is 1. |
| <i>&lt;filename&gt;</i> | is the name of the file.                                                                                                                                                     |

**Remarks** Note that rz and sz must be entered in lowercase characters.

If a file name is given in the rz statement, this name overrides the name given by the transmitting unit.

For more information on the ZMODEM protocol, please refer to <http://www.omen.com>. Related instruction is TRANSFER ZMODEM.



## **3 About Image Transfer Protocols**

This chapter describes the various image transfer protocols used in Intermec Fingerprint v8.71.

## About Image Protocols

The following five image transfer file protocols are used in connection with the STORE IMAGE statement and use a common format for the image data.

### **Intelhex**

Intel hex [Intel Hexadecimal Intellic 8/MDS (I\_hex) file format] is a well known standard format for transfer of bitmap images. Please refer to the standard literature on the subject.

Note that:

- Hex digits in Intelhex frames must be uppercase.
- Null frames may be omitted.
- Frames can be received in any order.
- Maximum file size is 64 KB.

### **UBI00**

Each frame contains:

*<data bytes>*

where *<data bytes>* are binary images. Modulo 2 bytes.

### **UBI01**

Each frame of data contains:

*<data bytes><checksum>*

where:

*<data bytes>* are binary images. Modulo 2 bytes.

*<checksum>* is the Modulo 65536 byte-wise sum of what is defined in protocol of “data bytes.” 2-byte binary, MSB, LSB.

### **UBI02**

Each frame of data contains:

*<number of data bytes><data bytes><checksum>*

where:

*<number of data bytes>* is a 2-byte binary. MSB, LSB.

*<data bytes>* are binary images. Modulo 2 bytes.

*<checksum>* is the Modulo 65536 byte-wise sum of what is defined in protocol of “number of data bytes” and “data bytes.” 2-byte binary, MSB, LSB.

## UBI03

Each frame of data contains:

*<start of frame id.><number of data bytes><data bytes><checksum>*

where:

|                                     |                                                                                                                                                               |
|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>&lt;start of frame id.&gt;</i>   | is 1 byte (ASCII 42 dec. = “*”).                                                                                                                              |
| <i>&lt;number of data bytes&gt;</i> | is 2 bytes binary. MSB, LSB.                                                                                                                                  |
| <i>&lt;data bytes&gt;</i>           | are binary images. Modulo 2 bytes.                                                                                                                            |
| <i>&lt;checksum&gt;</i>             | is the Modulo 65536 byte-wise sum of what is defined in protocol of “start of frame id” and “number of data bytes” and “data bytes.” 2 byte binary. MSB, LSB. |

## Image Format

The following image format is valid for Intelhex, UBI00, UBI01, UBI02, and UBI03 image transfer protocols, but not for the UBI10 protocol, which is a combined image transfer protocol and format.

A bitmap picture can be encoded:

- as a plain bit representation.
- with a Run Length Limited (RLL) algorithm.

Pictures can be magnified by the printer up to four times independently in both x and y directions.

The pictures can be rotated 180 degrees by the printer, effectively printing the picture upside-down. To print a bitmap in all four directions you have to define two bitmaps, one straight and one rotated 90 degrees. To comply with the Intermec Fingerprint convention, use the extension .1 for the straight bitmap and extension .2 for the rotated one.

Bitmap pictures, in both encoding schemes, are printed with the lowest address first - that is, the first row of defined data is the first thing out. If you misinterpret this, the only result is that your picture will come out upside-down.

## Encoding a Bitmap As a Bit Representation

The bitmap picture is encoded word oriented (16 bits), low byte first. The bits in each byte are read lsb first (bit 0). In this example, X's represent black dots in the final printout. The pattern shown is 22 bits wide and 28 rows high:

- Note the bit order in each byte and the word fill to nearest word (16 bit).
- To the right is a hex representation of the pattern, as it would appear in a memory dump.
- To get the pattern to appear as printed on this page with direction one, the last row (row 27) should have the lowest address.

|    | byte 3  byte 2  byte 1  byte 0   |                |
|----|----------------------------------|----------------|
|    | 76543210765432107654321076543210 |                |
| 0  | .....XXXXXXXXXXXXXXXXXXXXXX      | ff, ff, 3f, 00 |
| 1  | .....X.....X.....X.....X         | 01, 00, 20, 00 |
| 2  | .....X.....XX.....X.....X        | 61, 00, 20, 00 |
| 3  | .....X.....X.X.....X.....X       | a1, 00, 20, 00 |
| 4  | .....X.....X..X.....X.....X      | 21, 01, 20, 00 |
| 5  | .....X.....X..X..X.....X.....X   | 21, 02, 20, 00 |
| 6  | .....X.....X....X.....X.....X    | 21, 04, 20, 00 |
| 7  | .....X.....X....X.....X.....X    | 21, 08, 20, 00 |
| 8  | .....X.....X....X.....X.....X    | 21, 10, 20, 00 |
| 9  | .....X.....X.....X.....X.....X   | 21, 20, 20, 00 |
| 10 | .....X.....X.....X.....X.....X   | 21, 40, 20, 00 |
| 11 | .....X.....X.....X.....X.....X   | 21, 80, 20, 00 |
| 12 | .....X....X.....X.....X.....X    | 21, 00, 21, 00 |
| 13 | .....X..X.....X.....X.....X      | 21, 00, 22, 00 |
| 14 | .....X..X.....X.....X.....X      | 21, 00, 24, 00 |
| 15 | .....X.X.....X.....X.....X       | 21, 00, 28, 00 |
| 16 | .....X.XXXXXXXXXXXXXXXXXXXXXX    | fd, ff, 2f, 00 |
| 17 | .....X.....X.....X.....X         | 21, 00, 20, 00 |
| 18 | .....X.....X.....X.....X         | 21, 00, 20, 00 |
| 19 | .....X.....X.....X.....X         | 21, 00, 20, 00 |
| 20 | .....X.....X.....X.....X         | 21, 00, 20, 00 |
| 21 | .....X.....X.....X.....X         | 21, 00, 20, 00 |
| 22 | .....X.....X.....X.....X         | 21, 00, 20, 00 |
| 23 | .....X.....X.....X.....X         | 21, 00, 20, 00 |
| 24 | .....X.....X.....X.....X         | 21, 00, 20, 00 |
| 25 | .....X.....XXXXXX..X             | f9, 03, 20, 00 |
| 26 | .....X.....X.....X.....X         | 01, 00, 20, 00 |
| 27 | .....XXXXXXXXXXXXXXXXXXXXXX      | ff, ff, 3f, 00 |

## **Encoding a Bitmap With an RLL Algorithm**

RLL encoding is a very efficient way of compressing big bitmaps with relatively big black and/or white areas.

The RLL encoded picture is encoded byte oriented (8 bits). Each byte represents the number of consecutive black or white dots. The sum of bytes for each row must equal the width of the pattern. The first byte represents white dots, the second black and so on. The last byte must alter the color back to white. If the first dot is black just enter a zero first. Valid range for dot fields is 0 to 127 (0 to 7f hex). To get a row longer than 127, concatenate two rows with zero (for example, to get a row of 240 dots, enter 128,0,112).

The next step in our RLL encoding ageratum is to compress identical rows. To do this, add a byte to both ends of the dot row. The valid range for these bytes is -1 to -128 (ff to 80 hex). In this example of RLL encoding, X's represent black dots in the final print out. The pattern shown is 22 bits wide and 32 rows high:

- Notice the reverse byte order. Count dots from right.
- To the right is a decimal representation of the pattern.

To get the pattern to appear as printed on this page with direction one, the last row (row 27) should have the lowest address. Rows 18 through 24 are repeated by the data in row 17.

|    |                        |                      |
|----|------------------------|----------------------|
| 0  | XXXXXXXXXXXXXXXXXXXXXX | 0,22,0               |
| 1  | X.....X.....X          | 0,1,20,1,0           |
| 2  | X.....XX....X          | 0,1,4,2,14,1,0       |
| 3  | X.....X.X....X         | 0,1,4,1,1,1,13,1,0   |
| 4  | X.....X..X....X        | 0,1,4,1,2,1,12,1,0   |
| 5  | X.....X..X....X        | 0,1,4,1,3,1,11,1,0   |
| 6  | X.....X....X....X      | 0,1,4,1,4,1,10,1,0   |
| 7  | X.....X....X....X      | 0,1,4,1,5,1,9,1,0    |
| 8  | X.....X....X....X      | 0,1,4,1,6,1,8,1,0    |
| 9  | X.....X....X....X      | 0,1,4,1,7,1,7,1,0    |
| 10 | X.....X....X....X      | 0,1,4,1,8,1,6,1,0    |
| 11 | X.....X....X....X      | 0,1,4,1,9,1,5,1,0    |
| 12 | X....X.....X....X      | 0,1,4,1,10,1,4,1,0   |
| 13 | X...X.....X....X       | 0,1,4,1,11,1,3,1,0   |
| 14 | X..X.....X....X        | 0,1,4,1,12,1,2,1,0   |
| 15 | X.X.....X....X         | 0,1,4,1,13,1,1,1,0   |
| 16 | X.XXXXXXXXXXXXXXXX.X   | 0,1,1,18,1,1,0       |
| 17 | X.....X....X....X      | -8,0,1,4,1,15,1,0,-8 |
| 18 | X.....X....X....X      |                      |
| 19 | X.....X....X....X      |                      |
| 20 | X.....X....X....X      |                      |
| 21 | X.....X....X....X      |                      |
| 22 | X.....X....X....X      |                      |
| 23 | X.....X....X....X      |                      |
| 24 | X.....X....X....X      |                      |
| 25 | X.....XXXXXX..X        | 0,1,2,5,13,1,0       |
| 26 | X.....X.....X          | 0,1,20,1,0           |
| 27 | XXXXXXXXXXXXXXXXXXXXXX | 0,22,0               |
| 28 | .....X.....            |                      |
| 29 | .....X.....            |                      |
| 30 | .....X.....            |                      |
| 31 | .....X.....            |                      |
| 32 | .....XXXXXXX.....      | 7,9,6                |

## About the UBI10 Protocol

UBI10 is a combined protocol/file format for image transfer, as opposed to Intelhex and UBI00-UBI03 protocols described earlier in this chapter.

### Protocol Description

```

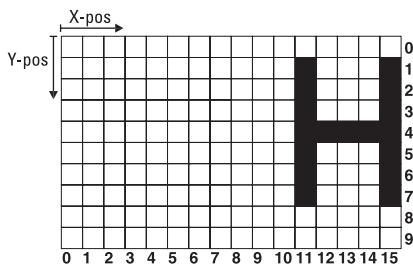
!BG
!X<pos>A .
!Y<pos>A .
!X<pos>A | !Y<pos>A !SB<bytes>W<data>
!X<pos>A | !Y<pos>A !SB<bytes>W<data>
!X<pos>A | !Y<pos>A !SB<bytes>W<data>
. . . .
!X<pos>A | !Y<pos>A !SB<bytes>W<data>!EG .
!PRINT .

```

### Frame Definitions

The width of the image in the STORE IMAGE statement should be given as a multiple of 16 bits.

| Parameter         | Description                                                                                                                                                                                                                                                                                                                                                                           |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| !BG               | Begin graphics.<br>Always appended by a carriage return character.                                                                                                                                                                                                                                                                                                                    |
| !X<pos>A          | Set absolute x position <pos><br>The value must be divisible by 8.<br>Default value is 0.<br>Once set, it will affect all consecutive y-positions in the image, until a new x-position is set.<br>Appended by a carriage return character, unless followed by a !SB<bytes>W<data> string on the same line.                                                                            |
| !Y<pos>A          | Set absolute y position <pos>.<br>Default value is 0. Appended by a carriage return character, unless followed by a !SB<bytes>W<data> string on the same line.                                                                                                                                                                                                                        |
| !SB<bytes>W<data> | Send one line of bitmap with <bytes> number of bytes. <data> is bitmap bytes.<br>Can be preceded by a new x- and/or y-position.<br>If appended by a carriage return character, next !SB set of data will be positioned at the current y-position incremented by 1.<br>If no appending carriage return character is used, a new y-position must be specified for next !SB set of data. |
| !EG               | End graphics<br>Always appended by a carriage return character.                                                                                                                                                                                                                                                                                                                       |
| !PRINT            | End page (end frame).<br>Always appended by a carriage return character.                                                                                                                                                                                                                                                                                                              |



The image illustrated above contains 2 bytes (= 16 bits) in each horizontal line. By setting the absolute start position to  $x = 8$ , you can start counting from the start of the second byte,  $x = 8$  in the matrix above. The first 3 bits (x-positions) are white, then comes one black bit followed by three white bits, and finally one black bit. Expressed in 0:s and 1:s, where 0 represents a white bit and 1 a black bit, the pattern is 00010001. This binary number can be expressed as 11 hex. The same pattern is repeated for each y-position from  $y = 1$  through  $y = 7$  with the exception of position  $y = 4$ , where all bits are black except for the leading three (that is, the pattern is 00011111, which can be expressed as 1F hex). Use this hexadecimal value as input data as shown in the example below.

### Example

In this example, the image shown above is transmitted to the printer. Do not use XON/XOFF (11 hex/13 hex) protocol, since these characters may coincide with input data. Use RTS/CTS instead. Do not strip LF.

```

10 STORE OFF
20 OPEN "uart1:" FOR INPUT AS #1
30 QNAME$="H.1"
40 QWIDTH%=16
50 QHEIGHT%=10
60 QPRO$="UBI10"
70 STORE IMAGE QNAME$,QWIDTH%,QHEIGHT%,QPROT$
80 STORE INPUT 900,4: 'Timeout 9 sec.
90 CLOSE#1
100 STORE OFF
RUN

```

The input string in line 80 should contain the following data. Carriage returns (.) after each !SB set of data increments the y-position by 1 in consecutive order. It may also be sent as a continuous string.

|                   |                                      |
|-------------------|--------------------------------------|
| !BG               | (Begin graphic)                      |
| !X8A              | (Set x-position)                     |
| !Y1A!SB1W<11 hex> | (Set y-position + data for $y = 1$ ) |
| !SB1W<11 hex>     | (Data for $y = 2$ )                  |
| !SB1W<11 hex>     | (Data for $y = 3$ )                  |
| !SB1W<1F hex>     | (Data for $y = 4$ )                  |
| !SB1W<11 hex>     | (Data for $y = 5$ )                  |
| !SB1W<11 hex>     | (Data for $y = 6$ )                  |
| !SB1W<11 hex>!EG  | (Data for $y = 7$ + end graphics)    |
| !PRINT            | (End frame)                          |

## PRBUF Protocol

The PRBUF Protocol is designed for downloading bitmap print image data directly from an application program, such as a Windows printer driver, directly to the printer's image buffer in connection with the PRBUF statement. The protocol consists of a two-byte header and a number of data bytes as described next.

### 2-Byte Header

Byte 1 is always the @-sign (Commercial at; Unicode 0x0040) and indicates start of the protocol header.

Byte 2 is:

|       |                             |
|-------|-----------------------------|
| 0     | Reserved (bitmap format)    |
| 1     | Reserved (RLL image format) |
| 2     | RLL buffer format           |
| 3-255 | Reserved                    |

### RLL Buffer Format

The RLL buffer format is optimized for use by Windows drivers. Because the performance of the host usually exceeds the performance of the printer, it is preferable to do most of the processing in the host before sending the job down to the printer.

- Data bytes 1 and 2 specify the pixel width (unsigned) of data in BIG Endian format for one line.
- Data bytes 3 and 4 specify the pixel height (unsigned) of the buffer when it is expanded BIG Endian.
- Data bytes 5 to  $nn$  specify the bitmap in RLL format. Example of RLL buffer protocol header, 515x212 pixels hexdump:

```
40 02 02 03 00 d4
```

### RLL Format

The RLL format is good for black and white pixel runs. It compresses data in both dimensions. It works well with one-dimensional bar codes, but grayscales grow in size instead of shrinking. The format is symmetric so that all pixel runs start and end with a white pixel and with line repetitions whenever applicable. This makes it possible to turn the format upside down.

## Specification of the RLL Format

```

<begin><toggling pixelruns><end>
-- total width of RLL pattern --
<begin> ::= <linereps>|<small white pixelrun>
<end> ::= : <begin>|<empty>
<toggling pixelruns> ::= <whiteAndBlack pixelruns>|
 <blackAndWhite pixelruns>|
<whiteAndBlack pixelruns> ::= <whiteAndBlack pixelruns><black pixel
 run>|<blackAndWhite pixelruns>|
<blackAndWhite pixelruns> ::= <black pixelrun><white pixelrun>
<linereps> ::= ((-1)-(-128))*-1 number of equal lines
<small white pixelrun> ::= 0-127, number of white pixels
<black pixelrun> ::= 0-255, number of black pixels
<white pixelrun> ::= 0-255, number of white pixels
<empty> ::= empty, extreme if the entire line fits in
 one pixelrun.

```

If there is no line repetition, there does not have to be any line repeat. If the pixelrun is out of range, it must be split into several runs.

Example of RLL format for an eight bit pattern:

|                                                                                                                           |                                                                                                                                                        |
|---------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> -**-*-*-* 1,1,1,1,1,1,1,1,0 *-**-*-* 0,1,1,1,1,1,1,1,1 --**-*-* 2,2,2,2,0 ***-*-*-- -2,0,2,2,2,2,-2 *-*-*-*- </pre> | Note the last 0 to end with a<br>white pixelrun of 0 pixels<br>repetition, stopped with a white<br>pixelrun of 0 pixels line and pixel<br>repetitions. |
|---------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|

Example of coding a black square of 800 dots to valid RLL format:

```

-128,0,255,0,255,0,255,0,35,0,-128
-128,0,255,0,255,0,255,0,35,0,-128
-128,0,255,0,255,0,255,0,35,0,-128
-128,0,255,0,255,0,255,0,35,0,-128
-128,0,255,0,255,0,255,0,35,0,-128
-128,0,255,0,255,0,255,0,35,0,-128
-32,0,255,0,255,0,255,0,35,0,-32

```



# **4** Character Sets and Fonts

This chapter includes illustrations of the fonts and character sets supported by Fingerprint, and describes scalable fonts and using the UTF-8 character set.

# Introduction

The following information applies to all single-byte character sets:

- Characters between ASCII 00 decimal and ASCII 31 decimal are unprintable control characters as listed below.
- Characters between ASCII 32 decimal and ASCII 127 decimal can always be printed, regardless of 7-bit or 8-bit communication protocol, provided that the selected font contains those characters.
- Characters between ASCII 128 decimal and ASCII 255 decimal can only be printed if the selected font contains those characters and an 8-bit communication protocol is used. If you use 7-bit communication, select another national character set (see NASC statement) or use a MAP statement to remap a character set.
- If a character which does not exist in the selected font is used, an error condition occurs.

## ***Non-Printable Control Characters (ASCII 00-31 dec)***

| ASCII | Character | Meaning               | ASCII | Character | Meaning                   |
|-------|-----------|-----------------------|-------|-----------|---------------------------|
| 00    | NUL       | Null                  | 16    | DLE       | Data link escape          |
| 01    | SOH       | Start of heading      | 17    | DC1       | Device control one        |
| 02    | STX       | Start of text         | 18    | DC2       | Device control two        |
| 03    | ETX       | End of text           | 19    | DC3       | Device control three      |
| 04    | EOT       | End of transmission   | 20    | DC4       | Device control four       |
| 05    | ENQ       | Enquiry               | 21    | NAK       | Negative acknowledge      |
| 06    | ACK       | Acknowledge           | 22    | SYN       | Synchronous idle          |
| 07    | BEL       | Bell                  | 23    | ETB       | End of transmission block |
| 08    | BS        | Backspace             | 24    | CAN       | Cancel                    |
| 09    | HT        | Horizontal tabulation | 25    | EM        | End of medium             |
| 10    | LF        | Line feed             | 26    | SUB       | Substitute                |
| 11    | VT        | Vertical tabulation   | 27    | ESC       | Escape                    |
| 12    | FF        | Form feed             | 28    | FS        | File separator            |
| 13    | CR        | Carriage return       | 29    | GS        | Group separator           |
| 14    | SO        | Shift out             | 30    | RS        | Record separator          |
| 15    | SI        | Shift in              | 31    | US        | Unit separator            |

**Roman 8 Character Set NASC 1**

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 8 | 9 |
|-----|---|---|---|---|---|---|----|---|---|---|
| 30  |   |   |   | ! | " | # | \$ | % | & | ' |
| 40  | ( | ) | * | + | , | - | .  | / | 0 | 1 |
| 50  | 2 | 3 | 4 | 5 | 6 | 7 | 8  | 9 | : | ; |
| 60  | < | = | > | ? | @ | A | B  | C | D | E |
| 70  | F | G | H | I | J | K | L  | M | N | O |
| 80  | P | Q | R | S | T | U | V  | W | X | Y |
| 90  | Z | [ | \ | ] | ^ | _ | `  | a | b | c |
| 100 | d | e | f | g | h | i | j  | k | l | m |
| 110 | n | o | p | q | r | s | t  | u | v | w |
| 120 | x | y | z | { |   | } | ~  |   | € | □ |
| 130 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 140 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 150 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 160 | À | Â | È | Ê | Ë | Î | Ï  | ' | ` |   |
| 170 | ^ | “ | ~ | Ù | Û | £ | –  | Ý | ý | ° |
| 180 | Ç | ç | Ñ | ñ | i | ¿ | ¤  | £ | ¥ | § |
| 190 | f | ¢ | â | ê | ô | û | á  | é | ó | ú |
| 200 | à | è | ò | ù | ä | ë | ö  | ü | Å | î |
| 210 | Ø | Æ | å | í | ø | æ | Ä  | ì | Ö | Ü |
| 220 | É | ï | ß | Ô | Á | Ã | ã  | Ð | ð | í |
| 230 | Ì | Ó | Ò | Õ | õ | Š | š  | Ú | ÿ | ÿ |
| 240 | Þ | þ | · | μ | ¶ | ¾ | —  | ¼ | ½ | ¤ |
| 250 | º | « | ■ | » | ± |   | □  |   |   |   |

**French Character Set NASC 33**

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 8 | 9 |
|-----|---|---|---|---|---|---|----|---|---|---|
| 30  |   |   |   | ! | " | £ | \$ | % | & | ' |
| 40  | ( | ) | * | + | , | - | .  | / | 0 | 1 |
| 50  | 2 | 3 | 4 | 5 | 6 | 7 | 8  | 9 | : | ; |
| 60  | < | = | > | ? | @ | A | B  | C | D | E |
| 70  | F | G | H | I | J | K | L  | M | N | O |
| 80  | P | Q | R | S | T | U | V  | W | X | Y |
| 90  | Z | ° | ç | § | ^ | _ | μ  | a | b | c |
| 100 | d | e | f | g | h | i | j  | k | l | m |
| 110 | n | o | p | q | r | s | t  | u | v | w |
| 120 | x | y | z | é | ù | è | “  |   | € | □ |
| 130 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 140 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 150 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 160 | À | Â | È | Ê | Ë | Î | Ï  | ’ | ‘ | ` |
| 170 | ^ | ” | ~ | Ù | Û | £ | –  | Ý | ý | ° |
| 180 | Ç | ç | Ñ | ñ | i | ¿ | ¤  | £ | ¥ | § |
| 190 | f | ¢ | â | ê | ô | û | á  | é | ó | ú |
| 200 | à | è | ò | ù | ä | ë | ö  | ü | Å | î |
| 210 | Ø | Æ | å | í | ø | æ | Ä  | ì | Ö | Ü |
| 220 | É | ï | ß | Ô | Á | Ã | ã  | Đ | ð | í |
| 230 | Ì | Ó | Ò | Õ | õ | Š | š  | Ú | ÿ | ÿ |
| 240 | Þ | þ | · | μ | ¶ | ¾ | —  | ¼ | ½ | ¤ |
| 250 | º | « | ■ | » | ± | □ |    |   |   |   |

**Spanish Character Set NASC 34**

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 8 | 9 |
|-----|---|---|---|---|---|---|----|---|---|---|
| 30  |   |   |   | ! | " | £ | \$ | % | & | ' |
| 40  | ( | ) | * | + | , | - | .  | / | 0 | 1 |
| 50  | 2 | 3 | 4 | 5 | 6 | 7 | 8  | 9 | : | ; |
| 60  | < | = | > | ? | § | A | B  | C | D | E |
| 70  | F | G | H | I | J | K | L  | M | N | O |
| 80  | P | Q | R | S | T | U | V  | W | X | Y |
| 90  | Z | i | Ñ | ¿ | ^ | _ | `  | a | b | c |
| 100 | d | e | f | g | h | i | j  | k | l | m |
| 110 | n | o | p | q | r | s | t  | u | v | w |
| 120 | x | y | z | ° | ñ | ç | ~  |   | € | □ |
| 130 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 140 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 150 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 160 | À | Â | È | Ê | Ë | Î | Ï  | ' | ` |   |
| 170 | ^ | " | ~ | Ù | Û | £ | -  | Ý | ý | ° |
| 180 | Ç | ç | Ñ | ñ | i | ¿ | ¤  | £ | ¥ | § |
| 190 | f | ¢ | â | ê | ô | û | á  | é | ó | ú |
| 200 | à | è | ò | ù | ä | ë | ö  | ü | Å | î |
| 210 | Ø | Æ | å | í | ø | æ | Ä  | ì | Ö | Ü |
| 220 | É | ï | ß | Ô | Á | Ã | ã  | Ð | ð | í |
| 230 | Ì | Ó | Ò | Õ | õ | Š | š  | Ú | ÿ | ÿ |
| 240 | Þ | þ | · | μ | ¶ | ¾ | —  | ¼ | ½ | ¤ |
| 250 | º | « | ■ | » | ± | □ |    |   |   |   |

**Italian Character Set NASC 39**

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 8 | 9 |
|-----|---|---|---|---|---|---|----|---|---|---|
| 30  |   |   |   | ! | " | £ | \$ | % | & | ' |
| 40  | ( | ) | * | + | , | - | .  | / | 0 | 1 |
| 50  | 2 | 3 | 4 | 5 | 6 | 7 | 8  | 9 | : | ; |
| 60  | < | = | > | ? | § | A | B  | C | D | E |
| 70  | F | G | H | I | J | K | L  | M | N | O |
| 80  | P | Q | R | S | T | U | V  | W | X | Y |
| 90  | Z | ° | ç | é | ^ | _ | ù  | a | b | c |
| 100 | d | e | f | g | h | i | j  | k | l | m |
| 110 | n | o | p | q | r | s | t  | u | v | w |
| 120 | x | y | z | à | ò | è | ì  |   | € | □ |
| 130 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 140 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 150 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 160 | À | Â | È | Ê | Ë | Î | Ï  | ’ | ‘ | ` |
| 170 | ^ | ” | ~ | Ù | Û | £ | –  | Ý | ý | ° |
| 180 | Ç | ç | Ñ | ñ | i | ¿ | ¤  | £ | ¥ | § |
| 190 | f | ¢ | â | ê | ô | û | á  | é | ó | ú |
| 200 | à | è | ò | ù | ä | ë | ö  | ü | Å | î |
| 210 | Ø | Æ | å | í | ø | æ | Ä  | ì | Ö | Ü |
| 220 | É | ï | ß | Ô | Á | Ã | ã  | Ð | ð | í |
| 230 | Ì | Ó | Ò | Õ | õ | Š | š  | Ú | ÿ | ÿ |
| 240 | Þ | þ | · | μ | ¶ | ¾ | —  | ¼ | ½ | ¤ |
| 250 | º | « | ■ | » | ± | □ |    |   |   |   |

**English (UK) Character Set NASC 44**

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 8 | 9 |
|-----|---|---|---|---|---|---|----|---|---|---|
| 30  |   |   |   | ! | " | £ | \$ | % | & | ' |
| 40  | ( | ) | * | + | , | - | .  | / | 0 | 1 |
| 50  | 2 | 3 | 4 | 5 | 6 | 7 | 8  | 9 | : | ; |
| 60  | < | = | > | ? | @ | A | B  | C | D | E |
| 70  | F | G | H | I | J | K | L  | M | N | O |
| 80  | P | Q | R | S | T | U | V  | W | X | Y |
| 90  | Z | [ | \ | ] | ^ | _ | `  | a | b | c |
| 100 | d | e | f | g | h | i | j  | k | l | m |
| 110 | n | o | p | q | r | s | t  | u | v | w |
| 120 | x | y | z | { |   | } | -  |   | € | □ |
| 130 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 140 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 150 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 160 | À | Â | È | Ê | Ë | Î | Ï  | ' | ` |   |
| 170 | ^ | “ | ~ | Ù | Û | £ | -  | Ý | ý | ° |
| 180 | Ç | ç | Ñ | ñ | i | ¿ | ¤  | £ | ¥ | § |
| 190 | f | ¢ | â | ê | ô | û | á  | é | ó | ú |
| 200 | à | è | ò | ù | ä | ë | ö  | ü | Å | î |
| 210 | Ø | Æ | å | í | ø | æ | Ä  | ì | Ö | Ü |
| 220 | É | ï | ß | Ô | Á | Ã | ã  | Ð | ð | í |
| 230 | Ì | Ó | Ò | Õ | õ | Š | š  | Ú | ÿ | ÿ |
| 240 | Þ | þ | · | μ | ¶ | ¾ | —  | ¼ | ½ | ¤ |
| 250 | º | « | ■ | » | ± | □ |    |   |   |   |

**Swedish Character Set NASC 46**

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|-----|---|---|---|---|---|---|---|---|---|---|
| 30  |   |   |   | ! | " | # | ¤ | % | & | ' |
| 40  | ( | ) | * | + | , | - | . | / | 0 | 1 |
| 50  | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; |
| 60  | < | = | > | ? | É | A | B | C | D | E |
| 70  | F | G | H | I | J | K | L | M | N | O |
| 80  | P | Q | R | S | T | U | V | W | X | Y |
| 90  | Z | Ä | Ö | Å | Ü |   | é | a | b | c |
| 100 | d | e | f | g | h | i | j | k | l | m |
| 110 | n | o | p | q | r | s | t | u | v | w |
| 120 | x | y | z | ä | ö | å | ü |   | € | □ |
| 130 | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ |
| 140 | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ |
| 150 | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ |
| 160 | À | Â | È | Ê | Ë | Î | Ï | ' | ` |   |
| 170 | ^ | “ | ~ | Ù | Û | £ | – | Ý | ý | ° |
| 180 | Ç | ç | Ñ | ñ | i | ¿ | ¤ | £ | ¥ | § |
| 190 | f | ¢ | â | ê | ô | û | á | é | ó | ú |
| 200 | à | è | ò | ù | ä | ë | ö | ü | Å | î |
| 210 | Ø | Æ | å | í | ø | æ | Ä | ì | Ö | Ü |
| 220 | É | ï | Þ | Ô | Á | Ã | ã | Ð | ð | í |
| 230 | Ì | Ó | Ò | Õ | õ | Š | š | Ú | ÿ | ÿ |
| 240 | Þ | þ | · | μ | ¶ | ¾ | — | ¼ | ½ | ¤ |
| 250 | º | « | ■ | » | ± | □ |   |   |   |   |

**Norwegian Character Set NASC 47**

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 8 | 9 |
|-----|---|---|---|---|---|---|----|---|---|---|
| 30  |   |   |   | ! | " | # | \$ | % | & | ' |
| 40  | ( | ) | * | + | , | - | .  | / | 0 | 1 |
| 50  | 2 | 3 | 4 | 5 | 6 | 7 | 8  | 9 | : | ; |
| 60  | < | = | > | ? | @ | A | B  | C | D | E |
| 70  | F | G | H | I | J | K | L  | M | N | O |
| 80  | P | Q | R | S | T | U | V  | W | X | Y |
| 90  | Z | Æ | Ø | Å | ^ | _ | `  | a | b | c |
| 100 | d | e | f | g | h | i | j  | k | l | m |
| 110 | n | o | p | q | r | s | t  | u | v | w |
| 120 | x | y | z | æ | ø | å | —  |   | € | □ |
| 130 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 140 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 150 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 160 | À | Â | È | Ê | Ë | Î | Ï  | ’ | ‘ | ` |
| 170 | ^ | ” | ~ | Ù | Û | £ | —  | Ý | ý | ° |
| 180 | Ç | ç | Ñ | ñ | i | ¿ | ¤  | £ | ¥ | § |
| 190 | f | ¢ | â | ê | ô | û | á  | é | ó | ú |
| 200 | à | è | ò | ù | ä | ë | ö  | ü | Å | î |
| 210 | Ø | Æ | å | í | ø | æ | Ä  | ì | Ö | Ü |
| 220 | É | ï | Þ | Ô | Á | Ã | ã  | Ð | ð | í |
| 230 | Ì | Ó | Ò | Õ | õ | Š | š  | Ú | ÿ | ÿ |
| 240 | Þ | þ | · | μ | ¶ | ¾ | —  | ¼ | ½ | ¤ |
| 250 | º | « | ■ | » | ± | □ |    |   |   |   |

**German Character Set NASC 49**

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 8 | 9 |
|-----|---|---|---|---|---|---|----|---|---|---|
| 30  |   |   |   | ! | " | # | \$ | % | & | ' |
| 40  | ( | ) | * | + | , | - | .  | / | 0 | 1 |
| 50  | 2 | 3 | 4 | 5 | 6 | 7 | 8  | 9 | : | ; |
| 60  | < | = | > | ? | § | A | B  | C | D | E |
| 70  | F | G | H | I | J | K | L  | M | N | O |
| 80  | P | Q | R | S | T | U | V  | W | X | Y |
| 90  | Z | Ä | Ö | Ü | ^ | _ | `  | a | b | c |
| 100 | d | e | f | g | h | i | j  | k | l | m |
| 110 | n | o | p | q | r | s | t  | u | v | w |
| 120 | x | y | z | ä | ö | ü | ß  |   | € | □ |
| 130 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 140 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 150 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 160 | À | Â | È | Ê | Ë | Î | Ï  | ' | ` | ` |
| 170 | ^ | “ | ~ | Ù | Û | £ | –  | Ý | ý | ° |
| 180 | Ç | ç | Ñ | ñ | i | ¿ | ¤  | £ | ¥ | § |
| 190 | f | ¢ | â | ê | ô | û | á  | é | ó | ú |
| 200 | à | è | ò | ù | ä | ë | ö  | ü | Å | î |
| 210 | Ø | Æ | å | í | ø | æ | Ä  | ì | Ö | Ü |
| 220 | É | ï | Þ | Ô | Á | Ã | ã  | Ð | ð | í |
| 230 | Ì | Ó | Ò | Õ | õ | Š | š  | Ú | ÿ | ÿ |
| 240 | Þ | þ | · | μ | ¶ | ¾ | —  | ¼ | ½ | ¤ |
| 250 | º | « | ■ | » | ± | □ |    |   |   |   |

**Japanese Latin Character Set NASC 81**

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 8 | 9 |
|-----|---|---|---|---|---|---|----|---|---|---|
| 30  |   |   |   | ! | " | # | \$ | % | & | ' |
| 40  | ( | ) | * | + | , | - | .  | / | 0 | 1 |
| 50  | 2 | 3 | 4 | 5 | 6 | 7 | 8  | 9 | : | ; |
| 60  | < | = | > | ? | @ | A | B  | C | D | E |
| 70  | F | G | H | I | J | K | L  | M | N | O |
| 80  | P | Q | R | S | T | U | V  | W | X | Y |
| 90  | Z | [ | ¥ | ] | ^ | _ | `  | a | b | c |
| 100 | d | e | f | g | h | i | j  | k | l | m |
| 110 | n | o | p | q | r | s | t  | u | v | w |
| 120 | x | y | z | { |   | } | ~  |   | € | □ |
| 130 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 140 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 150 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 160 | À | Â | È | Ê | Ë | Î | Ï  | ’ | ‘ | ` |
| 170 | ^ | ” | ~ | Ù | Û | £ | –  | Ý | ý | ° |
| 180 | Ç | ç | Ñ | ñ | i | ¿ | ¤  | £ | ¥ | § |
| 190 | f | ¢ | â | ê | ô | û | á  | é | ó | ú |
| 200 | à | è | ò | ù | ä | ë | ö  | ü | Å | î |
| 210 | Ø | Æ | å | í | ø | æ | Ä  | ì | Ö | Ü |
| 220 | É | ï | ß | Ô | Á | Ã | ã  | Đ | ð | í |
| 230 | Ì | Ó | Ò | Õ | õ | Š | š  | Ú | ÿ | ÿ |
| 240 | Þ | þ | · | μ | ¶ | ¾ | —  | ¼ | ½ | ¤ |
| 250 | º | « | ■ | » | ± | □ |    |   |   |   |

**Portuguese Character Set NASC 351**

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 8 | 9 |
|-----|---|---|---|---|---|---|----|---|---|---|
| 30  |   |   |   | ! | " | # | \$ | % | & | ' |
| 40  | ( | ) | * | + | , | - | .  | / | 0 | 1 |
| 50  | 2 | 3 | 4 | 5 | 6 | 7 | 8  | 9 | : | ; |
| 60  | < | = | > | ? | § | A | B  | C | D | E |
| 70  | F | G | H | I | J | K | L  | M | N | O |
| 80  | P | Q | R | S | T | U | V  | W | X | Y |
| 90  | Z | Ã | Ç | Õ | ^ | _ | `  | a | b | c |
| 100 | d | e | f | g | h | i | j  | k | l | m |
| 110 | n | o | p | q | r | s | t  | u | v | w |
| 120 | x | y | z | ã | ç | õ | °  |   | € | □ |
| 130 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 140 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 150 | □ | □ | □ | □ | □ | □ | □  | □ | □ | □ |
| 160 | À | Â | È | Ê | Ë | Î | Ï  | ' | ` | ` |
| 170 | ^ | " | ~ | Ù | Û | £ | -  | Ý | ý | ° |
| 180 | Ç | ç | Ñ | ñ | i | ¿ | ¤  | £ | ¥ | § |
| 190 | f | ¢ | â | ê | ô | û | á  | é | ó | ú |
| 200 | à | è | ò | ù | ä | ë | ö  | ü | Å | î |
| 210 | Ø | Æ | å | í | ø | æ | Ä  | ì | Ö | Ü |
| 220 | É | ï | ß | Ô | Á | Ã | â  | Ð | ð | í |
| 230 | Ì | Ó | Ò | Õ | õ | Š | š  | Ú | ÿ | ÿ |
| 240 | Þ | þ | · | μ | ¶ | ¾ | —  | ¼ | ½ | ¤ |
| 250 | º | « | ■ | » | ± | □ |    |   |   |   |

**PCMAP Character Set NASC -1**

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 8 | 9 |
|-----|---|---|---|---|---|---|----|---|---|---|
| 30  |   |   |   | ! | " | # | \$ | % | & | ' |
| 40  | ( | ) | * | + | , | - | .  | / | 0 | 1 |
| 50  | 2 | 3 | 4 | 5 | 6 | 7 | 8  | 9 | : | ; |
| 60  | < | = | > | ? | @ | A | B  | C | D | E |
| 70  | F | G | H | I | J | K | L  | M | N | O |
| 80  | P | Q | R | S | T | U | V  | W | X | Y |
| 90  | Z | [ | \ | ] | ^ | _ | `  | a | b | c |
| 100 | d | e | f | g | h | i | j  | k | l | m |
| 110 | n | o | p | q | r | s | t  | u | v | w |
| 120 | x | y | z | { |   | } | ~  | Ç | Ü |   |
| 130 | é | â | ä | à | å | ç | ê  | ë | è | ï |
| 140 | î | ì | Ä | Å | É | æ | Æ  | ô | ö | ò |
| 150 | û | ù | ÿ | Ö | Ü | ¢ | £  | ¥ | § | f |
| 160 | á | í | ó | ú | ñ | Ñ | ¤  | ¤ | ¤ | ` |
| 170 | ^ | ½ | ¼ | i | « | » | —  | Ý | ý | ° |
| 180 | Ç | ç | Ñ | ñ | i | ¿ | ¤  | £ | ¥ | § |
| 190 | f | ¢ | â | ê | ô | û | á  | é | ó | ú |
| 200 | à | è | ò | ù | ä | ë | ö  | ü | Å | î |
| 210 | Ø | Æ | å | í | ø | æ | Ä  | ì | Ö | Ü |
| 220 | É | ï | Þ | Ô | Á | Ã | ã  | Ð | ð | í |
| 230 | Ì | Ó | Ò | Õ | õ | Š | š  | Ú | Ý | ÿ |
| 240 | Þ | þ | · | μ | ¶ | ¾ | —  | ¼ | ½ | ¤ |
| 250 | ¤ | « | ■ | » | ± | □ |    |   |   |   |

**ANSI Character Set NASC -2**

|     | 0 | 1 | 2 | 3   | 4 | 5 | 6  | 7  | 8 | 9 |
|-----|---|---|---|-----|---|---|----|----|---|---|
| 30  |   |   |   | !   | " | # | \$ | %  | & | ' |
| 40  | ( | ) | * | +   | , | - | .  | /  | 0 | 1 |
| 50  | 2 | 3 | 4 | 5   | 6 | 7 | 8  | 9  | : | ; |
| 60  | < | = | > | ?   | @ | A | B  | C  | D | E |
| 70  | F | G | H | I   | J | K | L  | M  | N | O |
| 80  | P | Q | R | S   | T | U | V  | W  | X | Y |
| 90  | Z | [ | \ | ]   | ^ | _ | `  | a  | b | c |
| 100 | d | e | f | g   | h | i | j  | k  | l | m |
| 110 | n | o | p | q   | r | s | t  | u  | v | w |
| 120 | x | y | z | {   |   | } | ~  |    | € | □ |
| 130 | , | f | „ | ... | † | ‡ | ^  | %o | Š | „ |
| 140 | Œ | □ | Ž | □   | □ | ‘ | ,  | “  | ” | • |
| 150 | — | — | ~ | ™   | š | › | œ  | □  | ž | ÿ |
| 160 | í | ¢ | £ | ¤   | ¥ | — | §  | “  | © |   |
| 170 | á | « | ¬ | -   | ® | — | °  | ±  | 2 | 3 |
| 180 | ’ | μ | ¶ | .   | „ | ¹ | º  | »  | ¼ | ½ |
| 190 | ¾ | ¿ | À | Á   | Â | Ã | Ä  | Å  | Æ | Ç |
| 200 | È | É | Ê | Ë   | Ì | Í | Î  | Ï  | Ð | Ñ |
| 210 | Ò | Ó | Ô | Õ   | Ö | × | Ø  | Ù  | Ú | Û |
| 220 | Ü | Ý | Þ | ß   | à | á | â  | ã  | ä | å |
| 230 | æ | ç | è | é   | ê | ë | ì  | í  | î | ï |
| 240 | ð | ñ | ò | ó   | ô | õ | ö  | ÷  | ø | ù |
| 250 | ú | û | ü | ý   | þ | ÿ |    |    |   |   |

**MS-DOS Latin 1 Character Set NASC 850**

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6   | 7    | 8     | 9  |
|-----|---|---|---|---|---|---|-----|------|-------|----|
| 30  |   |   |   | ! | " | # | \$  | %    | &     | '  |
| 40  | ( | ) | * | + | , | - | .   | /    | 0     | 1  |
| 50  | 2 | 3 | 4 | 5 | 6 | 7 | 8   | 9    | :     | ;  |
| 60  | < | = | > | ? | @ | A | B   | C    | D     | E  |
| 70  | F | G | H | I | J | K | L   | M    | N     | O  |
| 80  | P | Q | R | S | T | U | V   | W    | X     | Y  |
| 90  | Z | [ | \ | ] | ^ | _ | `   | a    | b     | c  |
| 100 | d | e | f | g | h | i | j   | k    | l     | m  |
| 110 | n | o | p | q | r | s | t   | u    | v     | w  |
| 120 | x | y | z | { |   | } | ~   | Ç    | Ü     |    |
| 130 | é | â | ä | à | å | ç | ê   | ë    | è     | ï  |
| 140 | î | ì | Ä | Å | É | æ | Æ   | ô    | ö     | ò  |
| 150 | û | ù | ÿ | Ö | Ü | ø | £   | Ø    | x     | f  |
| 160 | á | í | ó | ú | ñ | Ñ | ã   | º    | ¿     | ®  |
| 170 | ¬ | ½ | ¼ | í | « | » | ■■■ | ■■■■ | ■■■■■ |    |
| 180 | ¬ | Á | Â | À | © | ⌐ |     | ⌐    | ⌐     | ¢  |
| 190 | ¥ | ¬ | ¬ | ¬ | ¬ | ¬ | —   | +—   | ã     | Ã  |
| 200 | ⌐ | ⌐ | ⌐ | ⌐ | ⌐ | ⌐ | ⌐   | ⌐    | ⌐     | Đ  |
| 210 | Ê | Ë | È | Í | Î | Ï | └   | └    | └     | █  |
| 220 | ▀ | ▀ | ▀ | ▀ | Ó | Þ | Ô   | Ò    | õ     | Õ  |
| 230 | μ | þ | þ | Ú | Û | Ù | ý   | Ý    | —     | ’  |
| 240 | - | ± | = | ¾ | ¶ | § | ÷   | ,    | °     | .. |
| 250 | . | 1 | 3 | 2 | ▀ |   |     |      |       |    |

**MS-DOS Greek 1 Character Set NASC 851**

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 8 | 9  |
|-----|---|---|---|---|---|---|----|---|---|----|
| 30  |   |   |   | ! | " | # | \$ | % | & | '  |
| 40  | ( | ) | * | + | , | - | .  | / | 0 | 1  |
| 50  | 2 | 3 | 4 | 5 | 6 | 7 | 8  | 9 | : | ;  |
| 60  | < | = | > | ? | @ | A | B  | C | D | E  |
| 70  | F | G | H | I | J | K | L  | M | N | O  |
| 80  | P | Q | R | S | T | U | V  | W | X | Y  |
| 90  | Z | [ | \ | ] | ^ | _ | `  | a | b | c  |
| 100 | d | e | f | g | h | i | j  | k | l | m  |
| 110 | n | o | p | q | r | s | t  | u | v | w  |
| 120 | x | y | z | { |   | } | ~  | Ç | Ü |    |
| 130 | é | â | ä | à | Á | ç | ê  | ë | è | ï  |
| 140 | Î | Ê | Ä | À | Í | Ó | Ô  | Ö | Ù | Ý  |
| 150 | Û | Ù | Ω | Ö | Ü | á | ξ  | é | ñ | í  |
| 160 | ő | ő | ó | ú | á | ß | Δ  | Ε | Ζ |    |
| 170 | Η | ½ | Θ | Ι | « | » | ▀  | ▀ | ▀ | ▀  |
| 180 | └ | Κ | Λ | Μ | Ν | ┘ | ┘  | ┘ | ┘ | ┘  |
| 190 | Ο | ┐ | └ | ─ | ─ | ─ | ─  | Π | Ρ |    |
| 200 | └ | Γ | ─ | ─ | ─ | ─ | ─  | Σ | Τ | Υ  |
| 210 | Φ | Χ | Ψ | Ω | α | β | γ  | └ | └ | █  |
| 220 | █ | δ | ε | █ | ζ | η | θ  | ι | κ | λ  |
| 230 | μ | ν | ξ | ο | π | ρ | σ  | ς | τ | ΄  |
| 240 | - | ± | υ | φ | χ | § | ψ  | , | ° | .. |
| 250 | ω | ü | ü | ώ | █ |   |    |   |   |    |

**MS-DOS Latin 2 Character Set NASC 852**

|     | 0 | 1 | 2  | 3 | 4 | 5 | 6  | 7 | 8 | 9 |
|-----|---|---|----|---|---|---|----|---|---|---|
| 30  |   |   |    | ! | " | # | \$ | % | & | ' |
| 40  | ( | ) | *  | + | , | - | .  | / | 0 | 1 |
| 50  | 2 | 3 | 4  | 5 | 6 | 7 | 8  | 9 | : | ; |
| 60  | < | = | >  | ? | @ | A | B  | C | D | E |
| 70  | F | G | H  | I | J | K | L  | M | N | O |
| 80  | P | Q | R  | S | T | U | V  | W | X | Y |
| 90  | Z | [ | \  | ] | ^ | _ | `  | a | b | c |
| 100 | d | e | f  | g | h | i | j  | k | l | m |
| 110 | n | o | p  | q | r | s | t  | u | v | w |
| 120 | x | y | z  | { |   | } | ~  | Ç | Ü |   |
| 130 | é | â | ä  | ú | ć | ç | ł  | ë | Ő | ő |
| 140 | í | ž | Ä  | Ć | É | Í | ó  | ö | Ľ |   |
| 150 | ł | ś | ś  | ö | ü | ł | ł  | ż | x | č |
| 160 | á | í | ó  | ú | ł | ą | ž  | ž | ę | ę |
| 170 | ń | ż | č  | ş | « | » | █  | █ | █ |   |
| 180 | – | Á | Â  | Ě | Ş | – | –  | – | – | Ž |
| 190 | ž | – | –  | – | – | – | –  | – | – | ă |
| 200 | Ł | Ę | Ł  | Ł | Ł | Ł | Ł  | Ł | Ł | Đ |
| 210 | Đ | Ę | d' | Ñ | í | î | ě  | ł | – | đ |
| 220 | █ | Ł | Ü  | █ | Ó | ß | Ô  | Ń | ń | ň |
| 230 | Š | š | Ŕ  | Ú | ŕ | Ű | ý  | Ý | ł | ‘ |
| 240 | – | ” | ‘  | ” | ” | § | ÷  | ” | ” | ” |
| 250 | · | ú | Ř  | ř | █ |   |    |   |   |   |

**MS-DOS Cyrillic Character Set NASC 855**

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 8 | 9 |
|-----|---|---|---|---|---|---|----|---|---|---|
| 30  |   |   |   | ! | " | # | \$ | % | & | ' |
| 40  | ( | ) | * | + | , | - | .  | / | 0 | 1 |
| 50  | 2 | 3 | 4 | 5 | 6 | 7 | 8  | 9 | : | ; |
| 60  | < | = | > | ? | @ | А | В  | С | Д | Е |
| 70  | Ғ | ҂ | ҃ | ҄ | ҅ | ҆ | ҇  | ҈ | ҉ | Ҋ |
| 80  | Ҍ | ҍ | Ҏ | ҏ | Ґ | ґ | Ғ  | ғ | Ҕ | ҕ |
| 90  | Җ | [ | \ | ] | ^ | _ | `  | а | б | с |
| 100 | Ҙ | ҙ | Қ | қ | Ҝ | ҝ | Ҟ  | ҟ | Ҡ | Ң |
| 110 | Ҋ | ҋ | Ҍ | ҍ | Ҏ | ҏ | Ґ  | ґ | Ғ | ғ |
| 120 | Ҳ | Ҕ | ҕ | Җ | җ | ҈ | ҉  | Ҋ | ҋ | Ҍ |
| 130 | Ҋ | ҋ | Ҍ | ҍ | Ҏ | ҏ | Ґ  | ґ | Ғ | ғ |
| 140 | Ҋ | ҋ | Ҍ | ҍ | Ҏ | ҏ | Ґ  | ґ | Ғ | ғ |
| 150 | Ҋ | ҋ | Ҍ | ҍ | Ҏ | ҏ | Ґ  | ґ | Ғ | ғ |
| 160 | Ҋ | ҋ | Ҍ | ҍ | Ҏ | ҏ | Ґ  | ґ | Ғ | ғ |
| 170 | Ҋ | ҋ | Ҍ | ҍ | Ҏ | ҏ | Ґ  | ґ | Ғ | ғ |
| 180 | Ҋ | ҋ | Ҍ | ҍ | Ҏ | ҏ | Ґ  | ґ | Ғ | ғ |
| 190 | Ҋ | ҋ | Ҍ | ҍ | Ҏ | ҏ | Ґ  | ґ | Ғ | ғ |
| 200 | Ҋ | ҋ | Ҍ | ҍ | Ҏ | ҏ | Ґ  | ґ | Ғ | ғ |
| 210 | Ҋ | ҋ | Ҍ | ҍ | Ҏ | ҏ | Ґ  | ґ | Ғ | ғ |
| 220 | Ҋ | ҋ | Ҍ | ҍ | Ҏ | ҏ | Ґ  | ґ | Ғ | ғ |
| 230 | Ҋ | ҋ | Ҍ | ҍ | Ҏ | ҏ | Ґ  | ґ | Ғ | ғ |
| 240 | Ҋ | ҋ | Ҍ | ҍ | Ҏ | ҏ | Ґ  | ґ | Ғ | ғ |
| 250 | Ҋ | ҋ | Ҍ | ҍ | Ҏ | ҏ | Ґ  | ґ | Ғ | ғ |

**MS-DOS Turkish Character Set NASC 857**

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6   | 7   | 8   | 9 |
|-----|---|---|---|---|---|---|-----|-----|-----|---|
| 30  |   |   |   | ! | " | # | \$  | %   | &   | ' |
| 40  | ( | ) | * | + | , | - | .   | /   | 0   | 1 |
| 50  | 2 | 3 | 4 | 5 | 6 | 7 | 8   | 9   | :   | ; |
| 60  | < | = | > | ? | @ | A | B   | C   | D   | E |
| 70  | F | G | H | I | J | K | L   | M   | N   | O |
| 80  | P | Q | R | S | T | U | V   | W   | X   | Y |
| 90  | Z | [ | \ | ] | ^ | _ | `   | a   | b   | c |
| 100 | d | e | f | g | h | i | j   | k   | l   | m |
| 110 | n | o | p | q | r | s | t   | u   | v   | w |
| 120 | x | y | z | { |   | } | ~   | Ç   | Ü   |   |
| 130 | é | â | ä | à | å | ç | ê   | ë   | è   | ï |
| 140 | î | í | Ä | Å | É | æ | Æ   | ô   | ö   | ò |
| 150 | û | ù | İ | Ö | Ü | ø | £   | Ø   | Ş   | ş |
| 160 | á | í | ó | ú | ñ | Ñ | Ğ   | ğ   | ¿   | ® |
| 170 | ¬ | ½ | ¼ | í | « | » | ■■■ | ☒☒☒ | ☒☒☒ |   |
| 180 | ¬ | Á | Â | À | © | ⌐ |     | ⌐   | ⌐   | ¢ |
| 190 | ¥ | ¬ | ¬ | ¬ | ¬ | ¬ | —   | +   | ã   | Ã |
| 200 | ⌐ | ⌐ | ⌐ | ⌐ | ⌐ | ⌐ | ⌐   | ⌐   | º   | ª |
| 210 | Ê | Ë | È | □ | Í | Î | Ï   | └   | └   | █ |
| 220 | ▀ | ▀ | ▀ | ▀ | Ó | Þ | Ô   | Ò   | õ   | Õ |
| 230 | μ | □ | × | Ú | Û | Ù | ì   | ÿ   | —   | ' |
| 240 | - | ± | □ | ¾ | ¶ | § | ÷   | ,   | °   | ” |
| 250 | . | 1 | 3 | 2 | ▀ |   |     |     |     |   |

**Windows Latin 2 Character Set NASC 1250**

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7  | 8 | 9 |
|-----|---|---|---|---|---|---|----|----|---|---|
| 30  |   |   |   | ! | " | # | \$ | %  | & | ' |
| 40  | ( | ) | * | + | , | - | .  | /  | 0 | 1 |
| 50  | 2 | 3 | 4 | 5 | 6 | 7 | 8  | 9  | : | ; |
| 60  | < | = | > | ? | @ | A | B  | C  | D | E |
| 70  | F | G | H | I | J | K | L  | M  | N | O |
| 80  | P | Q | R | S | T | U | V  | W  | X | Y |
| 90  | Z | [ | \ | ] | ^ | _ | `  | a  | b | c |
| 100 | d | e | f | g | h | i | j  | k  | l | m |
| 110 | n | o | p | q | r | s | t  | u  | v | w |
| 120 | x | y | z | { |   | } | ~  |    | € | □ |
| 130 | , | □ | „ | … | † | ‡ | □  | %o | Š | „ |
| 140 | Ś | Ł | Ž | Ź | □ | ‘ | ,  | “  | ” | • |
| 150 | — | — | □ | ™ | ſ | › | ś  | ł  | ż | ź |
| 160 | „ | „ | Ł | ¤ | À | Í | §  | ”  | © |   |
| 170 | Ş | « | — | - | ® | Ž | °  | ±  | ‘ | ł |
| 180 | ’ | μ | ¶ | · | ¸ | ą | ş  | »  | Ł | ” |
| 190 | ł | ż | Ŕ | Á | Â | Ã | Ä  | Ĺ  | Ć | Ç |
| 200 | Č | É | È | Ë | Ě | Í | Î  | Đ  | Ð | Ń |
| 210 | Ñ | Ó | Ô | Õ | Ö | × | Ř  | Ü  | Ú | Ű |
| 220 | Ü | Ý | Ț | Þ | ŕ | á | â  | ă  | ä | í |
| 230 | ć | ç | č | é | ę | ë | ě  | í  | î | đ |
| 240 | đ | ń | ň | ó | ô | ö | ö  | ÷  | ř | ů |
| 250 | ú | ű | ü | ý | ť | ‘ | ‘  | ‘  | ‘ | ‘ |

**Windows Cyrillic Character Set NASC 1251**

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 8 | 9 |
|-----|---|---|---|---|---|---|----|---|---|---|
| 30  |   |   |   | ! | " | # | \$ | % | & | ' |
| 40  | ( | ) | * | + | , | - | .  | / | 0 | 1 |
| 50  | 2 | 3 | 4 | 5 | 6 | 7 | 8  | 9 | : | ; |
| 60  | < | = | > | ? | @ | А | В  | С | Д | Е |
| 70  | Ғ | ҂ | ҃ | ҄ | ҅ | ҆ | ҇  | ҈ | ҉ | Ҋ |
| 80  | Ҍ | ҍ | Ҏ | ҏ | Ґ | ґ | Ғ  | ғ | Ҕ | ҕ |
| 90  | Җ | [ | \ | ] | ^ | _ | `  | а | б | с |
| 100 | Ҙ | ҙ | Қ | қ | Ҝ | ҝ | Ҟ  | ҟ | Ҡ | Ң |
| 110 | Ҋ | ҋ | Ҍ | ҍ | Ҏ | ҏ | Ґ  | Ғ | ғ | Ҕ |
| 120 | Ҳ | Җ | җ | { |   | } | ~  | Җ | җ | Ҕ |
| 130 | , | ҁ | ҂ | ҃ | ҄ | ҅ | ҆  | ҈ | ҉ | Ҋ |
| 140 | Ҋ | ҁ | ҂ | ҃ | ҄ | ҅ | ҆  | ҈ | ҉ | Ҋ |
| 150 | — | — | □ | ™ | љ | › | њ  | ќ | ћ | Џ |
| 160 | Ӧ | Ӧ | Ӧ | Ӧ | Ӧ | Ӧ | Ӧ  | Ӧ | Ӧ | Ӧ |
| 170 | Ҽ | ՞ | ՞ | ՞ | Ր | ՞ | ՞  | ՞ | ՞ | ՞ |
| 180 | Ր | Ր | Ր | Ր | Ր | Ր | Ր  | Ր | Ր | Ր |
| 190 | Տ | Ե | Բ | Վ | Գ | Ճ | Ե  | Ժ | Յ | Յ |
| 200 | Ի | Ի | Կ | Լ | Մ | Ն | Օ  | Պ | Ր | Ը |
| 210 | Տ | Ս | Փ | Խ | Ծ | Չ | Շ  | Ռ | Ե | Յ |
| 220 | Ե | Յ | Յ | Յ | Յ | Յ | Յ  | Յ | Յ | Յ |
| 230 | Ժ | Յ | Ի | Յ | Կ | Լ | Մ  | Ն | Ո | Ո |
| 240 | Ր | Ծ | Տ | Ս | Փ | Խ | Ծ  | Ռ | Ե | Յ |
| 250 | ՚ | ՚ | ՚ | ՚ | ՚ | ՚ | ՚  | ՚ | ՚ | ՚ |

**Windows Latin 1 Character Set NASC 1252**

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7   | 8 | 9 |
|-----|---|---|---|---|---|---|----|-----|---|---|
| 30  |   |   |   | ! | " | # | \$ | %   | & | ' |
| 40  | ( | ) | * | + | , | - | .  | /   | 0 | 1 |
| 50  | 2 | 3 | 4 | 5 | 6 | 7 | 8  | 9   | : | ; |
| 60  | < | = | > | ? | @ | A | B  | C   | D | E |
| 70  | F | G | H | I | J | K | L  | M   | N | O |
| 80  | P | Q | R | S | T | U | V  | W   | X | Y |
| 90  | Z | [ | \ | ] | ^ | _ | `  | a   | b | c |
| 100 | d | e | f | g | h | i | j  | k   | l | m |
| 110 | n | o | p | q | r | s | t  | u   | v | w |
| 120 | x | y | z | { |   | } | ~  |     | € | □ |
| 130 | , | f | „ | … | † | ‡ | ^  | %oo | Š | „ |
| 140 | Œ | □ | Ž | □ | □ | ‘ | ,  | “   | ” | • |
| 150 | — | — | ~ | ™ | š | › | œ  | □   | ž | ÿ |
| 160 | ı | ¢ | £ | ¤ | ¥ | ¦ | §  | “   | © |   |
| 170 | ¤ | « | — | - | ® | — | °  | ±   | 2 | 3 |
| 180 | ’ | μ | ¶ | · | · | ¹ | º  | »   | ¼ | ½ |
| 190 | ¾ | ¿ | À | Á | Â | Ã | Ä  | Å   | Æ | Ç |
| 200 | È | É | Ê | Ë | Ì | Í | Î  | Ï   | Ð | Ñ |
| 210 | Ò | Ó | Ô | Õ | Ö | × | Ø  | Ù   | Ú | Û |
| 220 | Ü | Ý | Þ | ß | à | á | â  | ã   | ä | å |
| 230 | æ | ç | è | é | ê | ë | ì  | í   | î | ï |
| 240 | ð | ñ | ò | ó | ô | õ | ö  | ÷   | ø | ù |
| 250 | ú | û | ü | ý | þ | ÿ |    |     |   |   |

**Windows Greek Character Set NASC 1253**

|     | 0  | 1  | 2  | 3   | 4  | 5  | 6  | 7  | 8  | 9 |
|-----|----|----|----|-----|----|----|----|----|----|---|
| 30  |    |    |    | !   | "  | #  | \$ | %  | &  | ' |
| 40  | (  | )  | *  | +   | ,  | -  | .  | /  | 0  | 1 |
| 50  | 2  | 3  | 4  | 5   | 6  | 7  | 8  | 9  | :  | ; |
| 60  | <  | =  | >  | ?   | @  | A  | B  | C  | D  | E |
| 70  | F  | G  | H  | I   | J  | K  | L  | M  | N  | O |
| 80  | P  | Q  | R  | S   | T  | U  | V  | W  | X  | Y |
| 90  | Z  | [  | \  | ]   | ^  | _  | `  | a  | b  | c |
| 100 | d  | e  | f  | g   | h  | i  | j  | k  | l  | m |
| 110 | n  | o  | p  | q   | r  | s  | t  | u  | v  | w |
| 120 | x  | y  | z  | {   |    | }  | ~  |    | €  | □ |
| 130 | ,  | f  | „  | ... | †  | ‡  | □  | %o | □  | „ |
| 140 | □  | □  | □  | □   | □  | ‘  | ,  | “  | ”  | • |
| 150 | —  | —  | □  | ™   | □  | ›  | □  | □  | □  | □ |
| 160 | „  | ‘A | £  | ¤   | ¥  | ¡  | §  | „  | ©  |   |
| 170 | □  | «  | ¬  | -   | ®  | —  | °  | ±  | 2  | 3 |
| 180 | ’  | μ  | ¶  | ·   | ‘E | ‘H | ‘I | »  | ‘O | ½ |
| 190 | ‘Y | ‘Ω | ‘ł | A   | B  | Γ  | Δ  | E  | Z  | H |
| 200 | Θ  | I  | K  | Λ   | M  | N  | Ξ  | O  | Π  | P |
| 210 | □  | Σ  | T  | Y   | Φ  | X  | Ψ  | Ω  | Ï  | Ŷ |
| 220 | á  | é  | ń  | í   | ü  | á  | þ  | ý  | đ  | ë |
| 230 | ζ  | η  | θ  | ı   | κ  | λ  | μ  | ν  | ξ  | օ |
| 240 | π  | ρ  | ς  | σ   | τ  | υ  | φ  | χ  | ψ  | ω |
| 250 | ő  | ü  | ó  | ú   | ώ  | □  |    |    |    |   |

**Windows Latin 5 Character Set NASC 1254**

|     | 0  | 1 | 2 | 3 | 4 | 5 | 6  | 7   | 8 | 9 |
|-----|----|---|---|---|---|---|----|-----|---|---|
| 30  |    |   |   | ! | " | # | \$ | %   | & | ' |
| 40  | (  | ) | * | + | , | - | .  | /   | 0 | 1 |
| 50  | 2  | 3 | 4 | 5 | 6 | 7 | 8  | 9   | : | ; |
| 60  | <  | = | > | ? | @ | A | B  | C   | D | E |
| 70  | F  | G | H | I | J | K | L  | M   | N | O |
| 80  | P  | Q | R | S | T | U | V  | W   | X | Y |
| 90  | Z  | [ | \ | ] | ^ | _ | `  | a   | b | c |
| 100 | d  | e | f | g | h | i | j  | k   | l | m |
| 110 | n  | o | p | q | r | s | t  | u   | v | w |
| 120 | x  | y | z | { |   | } | ~  |     | € | □ |
| 130 | ,  | f | „ | … | † | ‡ | ^  | %oo | Š | „ |
| 140 | Œ  | □ | □ | □ | □ | ‘ | ,  | “   | ” | • |
| 150 | —  | — | — | ™ | š | › | œ  | □   | □ | ÿ |
| 160 | ı  | ¢ | £ | ¤ | ¥ | — | §  | “   | ” | © |
| 170 | ¤  | « | — | - | ® | — | °  | ±   | 2 | 3 |
| 180 | ’  | μ | ¶ | · | · | ¹ | º  | »   | ¼ | ½ |
| 190 | ¾  | ¿ | À | Á | Â | Ã | Ä  | Å   | Æ | Ç |
| 200 | È  | É | Ê | Ë | Ì | Í | Î  | Ï   | Ð | Ñ |
| 210 | Ò  | Ó | Ô | Õ | Ö | × | Ø  | Ù   | Ú | Û |
| 220 | Ü  | ı | Ş | ß | à | á | â  | ã   | ä | å |
| 230 | æ  | ç | è | é | ê | ë | ì  | í   | î | ï |
| 240 | gó | ñ | ò | ó | ô | õ | ö  | ÷   | ø | ù |
| 250 | ú  | û | ü | ı | ş | ÿ |    |     |   |   |

**Windows Baltic Rim Character Set NASC 1257**

|     | 0 | 1  | 2 | 3 | 4 | 5 | 6  | 7  | 8 | 9 |
|-----|---|----|---|---|---|---|----|----|---|---|
| 30  |   |    |   | ! | " | # | \$ | %  | & | ' |
| 40  | ( | )  | * | + | , | - | .  | /  | 0 | 1 |
| 50  | 2 | 3  | 4 | 5 | 6 | 7 | 8  | 9  | : | ; |
| 60  | < | =  | > | ? | @ | A | B  | C  | D | E |
| 70  | F | G  | H | I | J | K | L  | M  | N | O |
| 80  | P | Q  | R | S | T | U | V  | W  | X | Y |
| 90  | Z | [  | \ | ] | ^ | _ | `  | a  | b | c |
| 100 | d | e  | f | g | h | i | j  | k  | l | m |
| 110 | n | o  | p | q | r | s | t  | u  | v | w |
| 120 | x | y  | z | { |   | } | ~  |    | € | □ |
| 130 | , | □  | „ | … | † | ‡ | □  | %o | □ | „ |
| 140 | □ | .. | ▼ | „ | ‘ | ’ | ,  | “  | ” | • |
| 150 | — | —  | □ | ™ | □ | › | □  | —  | ‘ | □ |
| 160 | □ | ¢  | £ | ¤ | □ | — | §  | Ø  | © |   |
| 170 | Ŗ | «  | — | - | ® | Æ | °  | ±  | 2 | 3 |
| 180 | ’ | μ  | ¶ | · | ø | ¹ | r  | »  | ¼ | ½ |
| 190 | ¾ | æ  | Ӓ | Ӆ | Ӓ | Ć | Ä  | Å  | Ӗ | Ӗ |
| 200 | Ҫ | É  | Ž | Ѐ | Ӯ | Ӯ | Ӯ  | Ӯ  | Ӯ | Ӯ |
| 210 | Ҥ | Ó  | Ӯ | Ӯ | Ӯ | Ӯ | Ӯ  | Ӯ  | Ӯ | Ӯ |
| 220 | Ӯ | Ż  | Ž | ڦ | ڦ | ڦ | ڦ  | ڦ  | ڦ | ڦ |
| 230 | ę | ē  | č | é | ž | è | ǵ  | ķ  | ī | ł |
| 240 | š | ń  | ñ | ó | ó | ő | ö  | ÷  | ü | ł |
| 250 | ś | ū  | ü | ż | ż | · |    |    |   |   |

**OCR-A BT Character Set**

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|-----|---|---|---|---|---|---|---|---|---|---|
| 30  |   |   |   | ! | " | # | ÷ | % | & | ' |
| 40  | ( | ) | * | + | - | . | / | Ø | Ł |   |
| 50  | Ł | Ę | ₃ | ₄ | ₅ | ₆ | ₇ | ₈ | ₉ | : |
| 60  | < | = | > | ? | ⓐ | Ⓐ | Ⓑ | Ⓒ | Ⓓ | Ⓔ |
| 70  | F | G | H | I | J | K | L | M | N | Ø |
| 80  | P | Q | R | S | T | U | V | W | X | Y |
| 90  | Z | Œ | \ | Ǯ | ^ | □ | □ | a | b | c |
| 100 | d | e | f | g | h | i | j | k | l | m |
| 110 | n | o | p | q | r | s | t | u | v | w |
| 120 | x | y | z | { |   | } | □ | □ | □ | □ |
| 130 | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ |
| 140 | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ |
| 150 | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ |
| 160 | □ | ſ | □ | £ | □ | ¥ | □ | □ | □ | ' |
| 170 | □ | < | □ | □ | □ | □ | □ | □ | · | . |
| 180 | □ | □ | □ | □ | □ | □ | □ | > | □ | □ |
| 190 | □ | □ | ƿ | □ | — | □ | Ä | Å | Æ | □ |
| 200 | □ | □ | □ | □ | □ | □ | □ | □ | - | Ñ |
| 210 | □ | ■ | □ | □ | ö | □ | ø | □ | □ | □ |
| 220 | Ü | □ | Ń | □ | Ƴ | □ | □ | □ | □ | □ |
| 230 | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ |
| 240 | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ |
| 250 | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ |

**OCR-B 10 Pitch BT Character Set**

|     | 0 | 1 | 2 | 3 | 4 | 5  | 6  | 7  | 8 | 9 |
|-----|---|---|---|---|---|----|----|----|---|---|
| 30  |   |   | ! | " | # | \$ | %  | &  | ' |   |
| 40  | ( | ) | * | + | , | -  | .  | /  | 0 | 1 |
| 50  | 2 | 3 | 4 | 5 | 6 | 7  | 8  | 9  | : | ; |
| 60  | < | = | > | ? | @ | A  | B  | C  | D | E |
| 70  | F | G | H | I | J | K  | L  | M  | N | O |
| 80  | P | Q | R | S | T | U  | V  | W  | X | Y |
| 90  | Z | [ | \ | ] | ^ | _  | `  | a  | b | c |
| 100 | d | e | f | g | h | i  | j  | k  | l | m |
| 110 | n | o | p | q | r | s  | t  | u  | v | w |
| 120 | x | y | z | { |   | }  | ~  |    |   |   |
| 130 |   |   |   |   |   |    |    |    |   |   |
| 140 |   |   |   |   |   |    |    |    |   |   |
| 150 |   |   |   |   |   |    |    |    |   |   |
| 160 |   |   |   | £ | ¤ | ¥  |    | §  | " | ' |
| 170 | " |   |   |   |   |    |    |    | † |   |
| 180 | ' | m |   | - | , | "  |    |    |   |   |
| 190 |   |   |   | ' | — | ^  | Ä  | Å  | Æ |   |
| 200 |   |   |   |   |   |    | IJ | ij |   | Ñ |
| 210 |   | ■ |   | - |   | ö  | ø  |    |   |   |
| 220 | Ü |   |   | ß |   |    |    |    |   | ä |
| 230 | æ |   |   |   |   |    |    |    |   |   |
| 240 |   |   |   |   |   |    |    |    | ø |   |
| 250 |   |   |   |   |   |    |    |    |   |   |

**DingDings SWA Character Set**

|     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|-----|---|---|---|---|---|---|---|---|---|---|
| 30  |   |   | ✂ | ✂ | ✂ | ✂ | ☎ | ☎ | ⌚ | ☰ |
| 40  | ✈ | ✉ | 👉 | 👉 | ✌ | ✌ | ✍ | ✍ | ✍ | ✍ |
| 50  | 🔑 | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✚ | ✚ | ✚ |
| 60  | ❖ | ✝ | ✝ | ✝ | ✝ | ✡ | ❖ | ❖ | ❖ | ❖ |
| 70  | ◆ | ❖ | ★ | ★ | ★ | ★ | ★ | ★ | ★ | ★ |
| 80  | ★ | * | * | * | * | * | * | * | * | * |
| 90  | * | * | * | * | * | * | * | * | ☀ | * |
| 100 | ❄ | ❄ | ❄ | ✿ | ✿ | ✿ | ✿ | ● | ○ |   |
| 110 | ■ | □ | □ | □ | □ | ▲ | ▼ | ◆ | ❖ | ☽ |
| 120 |   |   | █ | • | , | “ | ” | ( | ) |   |
| 130 | ( | ) | { | } | < | > | { | } | [ | ] |
| 140 | { | } | □ | □ | □ | □ | □ | □ | □ | □ |
| 150 | □ | □ | □ | □ | □ | □ | □ | □ | □ | □ |
| 160 | ♪ | : | ♪ | ♥ | ♪ | ♪ | ♪ | ♣ | ♦ |   |
| 170 | ♥ | ♠ | ① | ② | ③ | ④ | ⑤ | ⑥ | ⑦ | ⑧ |
| 180 | ⑨ | ⑩ | ① | ② | ③ | ④ | ⑤ | ⑥ | ⑦ | ⑧ |
| 190 | ⑨ | ⑩ | ① | ② | ③ | ④ | ⑤ | ⑥ | ⑦ | ⑧ |
| 200 | ⑨ | ⑩ | ① | ② | ③ | ④ | ⑤ | ⑥ | ⑦ | ⑧ |
| 210 | ⑨ | ⑩ | → | → | ↔ | ↑ | ↓ | → | ↗ | → |
| 220 | → | → | → | → | → | → | → | → | → | → |
| 230 | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ |
| 240 | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ |
| 250 | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ | ➡ |

## About the UTF-8 Character Set

UTF-8 (Universal character set Transformation Format-8) was devised to be a character set that encodes all Unicode characters and maintains compatibility with the US-ASCII (0 to 127 dec.) range of characters. The UTF-8 character set is available in Fingerprint v8.50 and later. UTF-8 data is encoded with 1, 2, 3 or 4 bytes, depending on the character number range. The table below shows the UTF-8 binary sequences corresponding to the Unicode character number.

| <b>Unicode character number range</b> |                                                                                                                                                                                                                                                 | <b>UTF-8 Byte sequence</b>                                                                                                                                                                                                                         |
|---------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Hex                                   | Binary                                                                                                                                                                                                                                          | Binary                                                                                                                                                                                                                                             |
| 0000-                                 | x <sub>7</sub> x <sub>6</sub> x <sub>5</sub> x <sub>4</sub> x <sub>3</sub> x <sub>2</sub> x <sub>1</sub>                                                                                                                                        | One byte:<br><b>0x<sub>7</sub>x<sub>6</sub>x<sub>5</sub>x<sub>4</sub>x<sub>3</sub>x<sub>2</sub>x<sub>1</sub></b>                                                                                                                                   |
| 007F                                  |                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                    |
| 0080-                                 | y <sub>5</sub> y <sub>4</sub> y <sub>3</sub> y <sub>2</sub> y <sub>1</sub> x <sub>6</sub> x <sub>5</sub> x <sub>4</sub> x <sub>3</sub> x <sub>2</sub> x <sub>1</sub>                                                                            | Two bytes:<br><b>110y<sub>5</sub>y<sub>4</sub>y<sub>3</sub>y<sub>2</sub>y<sub>1</sub> 10x<sub>6</sub>x<sub>5</sub>x<sub>4</sub>x<sub>3</sub>x<sub>2</sub>x<sub>1</sub></b>                                                                         |
| 07FF                                  |                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                    |
| 0800-                                 | z <sub>4</sub> z <sub>3</sub> z <sub>2</sub> z <sub>1</sub> y <sub>6</sub> y <sub>5</sub> y <sub>4</sub> y <sub>3</sub> y <sub>2</sub> y <sub>1</sub> x <sub>6</sub> x <sub>5</sub> x <sub>4</sub> x <sub>3</sub> x <sub>2</sub> x <sub>1</sub> | Three bytes:<br><b>1110z<sub>4</sub>z<sub>3</sub>z<sub>2</sub>z<sub>1</sub> 10y<sub>6</sub>y<sub>5</sub>y<sub>4</sub>y<sub>3</sub>y<sub>2</sub>y<sub>1</sub> 10 x<sub>6</sub>x<sub>5</sub>x<sub>4</sub>x<sub>3</sub>x<sub>2</sub>x<sub>1</sub></b> |
| FFFF                                  |                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                    |
| 010000-                               |                                                                                                                                                                                                                                                 | Four bytes:                                                                                                                                                                                                                                        |
| 10FFFF                                |                                                                                                                                                                                                                                                 | Not currently supported.                                                                                                                                                                                                                           |



**Note:** Characters requiring a UTF-8 sequence of 4 bytes are not supported.

Follow the next procedure to convert a Unicode character code in hex format to the UTF-8 byte decimal value necessary to print the characters.

### To convert a hex format Unicode character code to a decimal value

- 1 Determine the Unicode hex value for the character. For example, the hex value for the Cyrillic capital letter ZHE (Ж) is 0416.
- 2 Based on the hex value, determine the number of bytes required for UTF-8 encoding:

| <b>Hex value of character</b> | <b>Number of bytes required</b> |
|-------------------------------|---------------------------------|
| 0000 to 007F                  | One                             |
| 0080 to 07FF                  | Two                             |
| 0800 to FFFF                  | Three                           |

Using the same example, a hex value of 0416 requires two bytes for UTF-8 encoding.

- 3 Convert the hex value to binary. Using the same example, a hex value of 0416 equals the binary value 10000010110.
- 4 Identify x, y, and z bits as applicable. Start with the least significant digits to the right and pad with zeros to the left if necessary.

In this example, the first five digits of the binary value 10000010110 correspond to the  $y$  bits, and the remaining six digits correspond to the  $x$  bits. No padding zeros are necessary.

The first byte is 11010000.

The second byte is 10010110.

- 5** Convert the bytes to decimal format. Using this example, the byte value 11010000 equals a decimal value of 208, and the byte value 10010110 equals a decimal value of 150.

Now that you have determined the decimal value for the Unicode character, you can use the values in a print command:

```
prtxt chr$(208)+chr$(150)
```

The UTF-8 character set is invoked using NASC 8, NASC “UTF-8” or NASCD “UTF-8”:

- When selecting UTF-8 with the NASC command, the font must be selected with the FONT command. Disable UTF-8 encoding by choosing a different character set with the NASC command.
- Use FONTD to select the desired font if you invoke UTF-8 with the NASCD command. After you select UTF-8 with the NASCD command, you must actively disable it with NASCD “” before returning to a single-byte character set.



**Note:** In order to avoid confusion between active character sets and fonts, it is recommended that you use only the NASC and FONT commands when working with UTF-8 unless you have experience with the NASCD and FONTD commands.

When using UTF-8, it is important that the font contains the desired characters. Complete lists of characters included in pre-installed fonts can be found in the *Intermec Fingerprint, Font Reference Manual*. The default font, Swiss 721 BT, contains the largest number of glyphs of the pre-installed fonts. More information on adding fonts to the printer can be found under the FONT command in Chapter 2. Unicode character numbers can be found at the web site of the Unicode organization ([www.unicode.org](http://www.unicode.org)). It is not recommended to have UTF-8 enabled when printing bar codes since bar code data will use the UTF-8 byte sequence as input, while the human readable uses the UTF-8 mapped character number.



**Note:** FONT and FONTD commands are reset to their defaults after a PRINTFEED (or CLL) command. NASC and NASCD commands are not reset to default after a PRINTFEED (or CLL) command.

## Example

This example prints the Hiragana Letter Small A character (Unicode hex 3041), corresponding to the UTF-8 sequence 227 dec. + 129 dec. + 129 dec., in the Song font. This is followed by the Cyrillic Capital Letter ZHE (Unicode hex 0416) in the Swiss 721 BT font.

```

10 NASC "UTF-8"
20 FONT "Song"
30 PRTXT CHR$(227)+CHR$(129)+CHR$(129)
40 PRTXT " = Hiragana Letter Small A"
50 PRPOS 0,35
60 FONT "Swiss 721 BT"
70 PRTXT CHR$(208)+CHR$(150)
80 PRTXT " = Cyrillic Capital Letter ZHE"
90 PRINTFEED

```

## Scalable Fonts

The printer comes with 15 pre-installed scalable fonts. The printout samples on the next page are in 10 point size with no slant and 100% width. The quality of these samples does not exactly correspond to the printout quality from your printer, which is affected by printhead density, printing method, type of media and ribbon, and a number of other factors.

|                             |                                                                                                                     |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------|
| Century Schoolbook BT       | THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG<br>the quick brown fox jumps over the lazy dog 1234567890               |
| DingDings SWA               | *★❖ *❖★❖★ +❖★❖★ ◆❖★❖ ❖★❖★❖ ★❖★❖ *★❖ ★❖★❖ *❖★❖ ◆❖★❖                                                                  |
| Dutch 801 Roman BT          | THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG<br>the quick brown fox jumps over the lazy dog 1234567890               |
| Dutch 801 Bold BT           | <b>THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG</b><br><b>the quick brown fox jumps over the lazy dog 1234567890</b> |
| Futura Light BT             | THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG<br>the quick brown fox jumps over the lazy dog 1234567890               |
| Letter Gothic 12 Pitch BT   | THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG<br>the quick brown fox jumps over the lazy dog 1234567890               |
| Monospace 821 BT            | THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG<br>the quick brown fox jumps over the lazy dog 12345                    |
| Monospace 821 Bold BT       | <b>THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG</b><br><b>the quick brown fox jumps over the lazy dog 12345</b>      |
| OCR-A BT                    | THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG<br>the quick brown fox jumps over the lazy dog 1234                     |
| OCR-B 10 Pitch BT           | THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG<br>the quick brown fox jumps over the lazy dog 123456                   |
| Prestige 12 Pitch Bold BT   | THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG<br>the quick brown fox jumps over the lazy dog 123456                   |
| Swiss 721 BT                | THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG<br>the quick brown fox jumps over the lazy dog 1234567890               |
| Swiss 721 Bold BT           | <b>THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG</b><br><b>the quick brown fox jumps over the lazy dog 1234567890</b> |
| Swiss 721 Bold Condensed BT | <b>THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG</b><br><b>the quick brown fox jumps over the lazy dog 1234567890</b> |
| Zurich Extra Condensed Bold | THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG<br>the quick brown fox jumps over the lazy dog                          |

## Bitmap Fonts

It is possible to use fonts in the “old” Intermec .ATF bitmap font format. This feature improves compatibility with custom-made programs originally created in Fingerprint v6.xx or earlier versions.

Downloading an .ATF font (for example XX030RSN.ATF) to the printer produces three fonts in the memory; one without any extension (for example XX030RSN), one with the extension .1 (for example XX030RSN.1), and one with the extension .2 (for example XX030RSN.2). When using bitmap fonts in Fingerprint v8.xx, the relation between print direction and extension is of no consequence.

It is recommended to exclude the font height parameter in the FONT and BARFONT statements and use MAG to enlarge the font. Slant does not work at all with bitmap fonts.

## Font Aliases

The standard font names in this version of Intermec Fingerprint are longer than in earlier Fingerprint versions and may be cumbersome to use. They are also incompatible with the LAYOUT statement, which restricts the font name to 10 characters.

However, it is possible to create a file containing a list of font aliases. The file should be named exactly as shown here (note the leading period character that specifies it as a system file):

```
"c:..FONTALIAS"
```

The format of the file should be:

```
"<Alias name #1>","<Name of font>"[,size[,<slant>[,<width>]]]
"<Alias name #2>","<Name of font>"[,size[,<slant>[,<width>]]]
"<Alias name #3>","<Name of font>"[,size[,<slant>[,<width>]]]
```

The file can contain as many fontname aliases as required. The default size is 12 points, the default slant is 0°, and the default width is 100 (%).

A font alias can be used as any other font, but its size, slant, and width cannot be changed.



**Note:** The printer must be rebooted before the font alias can be used.

Examples:

```
"BODYTEXT","Century Schoolbook BT",10,0,80
"HEADLINE","Swiss 721 Bold BT",18,0,110
"WARNING","Swiss 721 BT",12
```

For more information on fonts and character sets, refer to the *Intermec Fingerprint v8.xx, Font Reference Manual*.





# 5 Bar Codes

This chapter lists information on bar code symbologies supported by Fingerprint, including settings, default values, and other setup information.

## Supported Symbologies

Fingerprint supports the bar code symbologies listed in the next table. Use the designation assigned to each symbology when using Fingerprint commands.

### **Bar Code Symbologies Supported by Fingerprint**

| <b>Bar Code</b>                         | <b>Designation</b> |
|-----------------------------------------|--------------------|
| Aztec                                   | “AZTEC”            |
| Codabar                                 | “CODABAR”          |
| Code 11                                 | “CODE11”           |
| Code 16K                                | “CODE16K”          |
| Code 39‘                                | “CODE39”           |
| Code 39 full ASCII                      | “CODE 39A”         |
| Code 39 w. checksum                     | “CODE39C”          |
| Code 49                                 | “CODE49”           |
| Code 93                                 | “CODE93”           |
| Code 128                                | “CODE128”          |
| Code 128 subset A                       | “CODE128A”         |
| Code 128 subset B                       | “CODE128B”         |
| Code 128 subset C                       | “CODE128C”         |
| Data Matrix                             | “DATAMATRIX”       |
| DUN-14/16                               | “DUN”              |
| EAN-8                                   | “EAN8”             |
| EAN-8 Composite with CC-A or CC-B       | “EAN8_CC”          |
| EAN-13                                  | “EAN13”            |
| EAN-13 Composite with CC-A or CC-B      | “EAN13_CC”         |
| EAN 128                                 | “EAN128”           |
| EAN 128 subset A                        | “EAN128A”          |
| EAN 128 subset B                        | “EAN128B”          |
| EAN 128 subset C                        | “EAN128C”          |
| EAN.UCC 128 Composite with CC-A or CC-B | “EAN128_CCCAB”     |
| EAN. UCC 128 Composite with CC-CC       | “EAN128_CCC”       |
| Five-Character Supplemental Code        | “ADDON5”           |
| Industrial 2 of 5                       | “C2OF5IND”         |
| Industrial 2 of 5 w. checksum           | “C2OF5INDC”        |
| Interleaved 2 of 5                      | “INT2OF5”          |
| Interleaved 2 of 5 w. checksum          | “INT2OF5C”         |
| Interleaved 2 of 5 A                    | “INT2OF5A”         |
| Matrix 2 of 5                           | “CODE128A”         |
| MaxiCode                                | “MAXICODE”         |
| MicroPDF417                             | “MICROPDF417”      |
| MSI (modified Plessey)                  | “MSI”              |
| Planet                                  | “PLANET”           |
| PDF 417                                 | “PDF417”           |
| Plessey                                 | “PLESSEY”          |

***Bar Code Symbolologies Supported by Fingerprint (cont'd.)***

|                                        |            |
|----------------------------------------|------------|
| Postnet                                | "POSTNET"  |
| QR Code                                | "QRCODE"   |
| RSS-14                                 | "RSS14"    |
| RSS-14 Truncated                       | "RSS14T"   |
| RSS-14 Stacked                         | "RSS14S"   |
| RSS-14 Stacked Omnidirectional         | "RSS14SO"  |
| RSS-14 Limited                         | "RSS14L"   |
| RSS-14 Expanded                        | "RSS14E"   |
| RSS-14 Expanded Stacked                | "RSS14ES"  |
| Straight 2 of 5                        | "C2OF5"    |
| Two-Character Supplemental Code        | "ADDON2"   |
| UCC-128 Serial Shipping Container Code | "UCC128"   |
| UPC-5 digits Add-On Code               | "SCCADDON" |
| UPC-A                                  | "UPCA"     |
| UPC-A Composite with CC-A or CC-B      | "UPCA_CC"  |
| UPC-D1                                 | "UPCD1"    |
| UPC-D2                                 | "UPCD2"    |
| UPC-D3                                 | "UPCD3"    |
| UPC-D4                                 | "UPCD4"    |
| UPC-D5                                 | "UPCD5"    |
| UPC-E                                  | "UPCE"     |
| UPC-E Composite with CC-A or CC-B      | "UPCE_CC"  |
| UPC Shipping Container Code            | "UPCSCC"   |

## **General Rules for Bar Code Printing**

The printer contains a number of bar code generators, which can produce highly readable bar codes in four different directions.

However, a general rule which applies to all thermal printers is that it is more difficult to print a bar code with the bars across the media path (ladder style) than along the media path (picket fence style.) Therefore, to ensure a highly readable printout, we recommend that you do not use narrow bars less than 3 dots, when printing bar codes with the bars across the media path (ladder style).

No such restrictions apply for bar codes with the bars along the media path (picket fence style).

Print speed also affects the printout quality of bar codes:

- Generally, a lower print speed gives a better quality, especially for ladder style bar codes and at low ambient temperatures.
- Do not use a higher print speed than necessary and consider the overall print cycle time.
- In some instances, a lower print speed may actually give better overall performance.

Intermec recommends that you do your own tests with your unique applications to find the best compromise between printout quality, performance, and media.

Specifications for bar code symbologies can be obtained from these organizations:

- EAN International (<http://www.ean-int.org>)
- UCC - The Uniform Code Council, Inc. (<http://www.uc-council.org>)
- AIM International, Inc. (<http://www.aimi.org>)
- ANSI - American National Standard Institute (<http://www.ansi.org>)

## One- and Two-Dimensional Bar Codes

This section lists the most commonly used one- and two-dimensional bar code symbologies and their associated parameters, ranges, and settings. This information is intended to help you enter acceptable parameters or input data. For more information, please refer to the standard literature on the subject of bar codes.

For simple bar codes, like EAN and UPC codes, it is recommended to use separate instructions, whereas for more complex bar codes like MicroPDF417 or QR Code, the BARSET statement is required.

The following bar codes are described:

- Aztec
- Code 39
- Code 128
- Data Matrix
- EAN-8
- EAN-13
- EAN 128
- Interleaved 2 of 5
- MaxiCode
- MicroPDF417
- PDF417
- QR Code
- RSS-14
- RSS-14 Truncated
- RSS-14 Stacked
- RSS-14 Stacked Omnidirectional
- RSS-14 Limited
- RSS-14 Expanded
- RSS-14 Expanded Stacked
- UPC-A
- UPC-E

For more information on add-on codes, see “[About AddOn Codes](#)” on [page 308](#).

For more information on composite bar codes, see “[About Composite Bar Codes](#)” on [page 309](#).

## Aztec

### *Separate Commands*

| Command   | Ranges and Settings  |
|-----------|----------------------|
| BARTYPE   | “AZTEC”              |
| BARRATIO  | Not applicable.      |
| BARMAG    | < 128. Default is 2. |
| BARHEIGHT | Not applicable.      |
| BARFONT   | Not applicable.      |

### *BARSET Data for Aztec*

| BARSET    | Description                                             | Ranges and Settings                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-----------|---------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [#<ncon>] | (Optional) Specifies the start parameter in the syntax. |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <sexp>    | Bar code type.                                          | “AZTEC”                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <nexp1-7> | Not applicable.                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <nexp8>   | Error correction or fixed symbol.                       | 0 (default): 23%+3 codewords. Dynamic symbol size with fixed error correction.<br>1-99: Static error correction level (in percent). Dynamic symbol size.<br>101-104: Compact Format symbol, 1 to 4 layers (+100). Error correction level depends on spare bits in chosen symbol size. Static symbol size.<br>201-232: Full Range symbol, 1-32 layers (+200). Error correction level depends on spare bits in chosen symbol size. Static symbol size.<br>300: Simple Aztec rune. |
| <nexp9>   | Menu symbol.                                            | 0 (default): No menu symbol.<br>1: Menu symbol.                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <nexp10>  | Symbol append.                                          | 1 (default): Symbol append OFF.<br>2-26: Append # of symbols. ID field used if present.                                                                                                                                                                                                                                                                                                                                                                                         |
| <nexp11>  | Enables or disables ECI support.                        | 0 (default): ECI support disabled.<br>1: ECI support enabled.                                                                                                                                                                                                                                                                                                                                                                                                                   |

### *ID Field Formatting in PRBAR (Symbol Append ON)*

| Parameter            | Ranges and Settings  | Notes                                                                                 |
|----------------------|----------------------|---------------------------------------------------------------------------------------|
| Number of characters | 0 to 24 (0x00-0xFF). | 0x20 not allowed.                                                                     |
| Input characters     | ASCII 0x41-0x5a      | Most efficient if all uppercase characters. Do not use space character (ASCII(0x20)). |

**Remarks**

When Aztec is selected and ECI support is enabled ( $<next11> = 1$ ), the strings “\<nnnnnn>” and “\\” are interpreted as follows:

\<nnnnnn> The backslash indicates that the character set should be changed, and *nnnnnn* is the number of the chosen character set. The string is decoded as an ECI character followed by one to three digit numbers in the bar code. All ECI characters are printed in ASCII encodation, meaning that this string itself can be encoded in any mode. The algorithm switches to ASCII mode to encode the ECI characters, then encodes the other characters in the most efficient modes.

\\ Two backslashes are decoded as a single backslash character (“\\”) in the bar code. All double backslashes are printed in the most efficient modes as specified and are decoded the same way when ECI support is not enabled.

Error 1112 (ECI syntax error) occurs if ECI mode is enabled and the PRBAR input string is invalid.

**Code 39****Separate Commands**

| <b>Command</b> | <b>Ranges and Settings</b>                                                                                                                 |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:       | “CODE39”<br>“CODE39A”<br>“CODE39C”                                                                                                         |
| BARRATIO:      | 2:1-3:1. Default is 3:1.                                                                                                                   |
| BARMAG:        | If the narrow element is less than 4 dots wide, then the ratio must be larger than 2.25:1 (9:4). Otherwise, no restrictions. Default is 2. |
| BARHEIGHT:     | No restrictions. Default is 100.                                                                                                           |
| BARFONT:       | No restrictions.                                                                                                                           |

**Input Data for Code 39**

| <b>Parameter</b>     | <b>Ranges and Settings</b> | <b>Notes</b>            |
|----------------------|----------------------------|-------------------------|
| Number of characters | Unlimited                  |                         |
| Check digit          | None                       |                         |
| Digits               | 0 to 9                     |                         |
| Uppercase letters    | A to Z                     | No national characters. |
| Lowercase letters    | Not supported              |                         |
| Punctuation marks    | - . space \$ / + %         |                         |
| Start character      | *                          | Added automatically.    |
| Stop character       | *                          | Added automatically.    |

**Remarks**

Code39A allows for the entire ASCII set to be encoded (128 characters). Code39C allows characters that are not in the input set to be sent into the symbol without error. The character will not be printed.

**Code 128*****Separate Commands***

| Command    | Ranges and Settings                                 |
|------------|-----------------------------------------------------|
| BARTYPE:   | “CODE128”<br>“CODE128A”<br>“CODE128B”<br>“CODE128C” |
| BARRATIO:  | Fixed. BARRATIO statement ignored.                  |
| BARMAG:    | = 2 (default).                                      |
| BARHEIGHT: | No restrictions. Default is 100.                    |
| BARFONT    | No restrictions.                                    |

***BARSET Data for Code 128***

| BARSET     | Description                                                         | Ranges and Settings                                                                                                                         |
|------------|---------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| [#<ncon>]  | (Optional) Specifies the start parameter in the syntax.             |                                                                                                                                             |
| <sexp>     | Bar code type.                                                      | “CODE128”<br>“CODE128A”<br>“CODE128B”<br>“CODE128C”                                                                                         |
| <nexp1-2>  | Not applicable.                                                     |                                                                                                                                             |
| <nexp3>    | Barmag/Enlargement.                                                 | <128. Default is 2.                                                                                                                         |
| <nexp4>    | Barheight.                                                          | No restriction. Default is 100.                                                                                                             |
| <nexp5>    | Not applicable.                                                     |                                                                                                                                             |
| <nexp6>    | Character exclusion in bar code.<br>Does not affect human readable. | 0: Disables parentheses and spaces in the bar code data.<br>Non-zero: Bar code and human readable field will include exactly the same data. |
| <nexp7-10> | Not applicable.                                                     |                                                                                                                                             |

***Input Data for Code 128***

| Parameter             | Ranges and Settings                                                                          | Notes                               |
|-----------------------|----------------------------------------------------------------------------------------------|-------------------------------------|
| Number of characters  | Unlimited                                                                                    |                                     |
| Check digit:          | One                                                                                          | Added automatically.                |
| Input characters:     | ASCII 0 to 127                                                                               | According to Roman 8 character set. |
| Functional characters | FNC1 ASCII 128 dec.)<br>FNC2 ASCII 129 dec.)<br>FNC3 ASCII 130 dec.)<br>FNC4 ASCII 131 dec.) | See Note 1.                         |
| Start characters      | See Note 2.                                                                                  |                                     |
| Code characters       | See Notes 1 and 2.                                                                           |                                     |
| Shift characters      | See Notes 1 & 2.                                                                             |                                     |
| Stop characters       | Always.                                                                                      | Added automatically.                |



**Note 1:** Function characters FNC1-4, code characters, and shift characters require either an 8-bit communication protocol, remapping to an ASCII value between 0-127 dec., or the use of an CHR\$ function.

FNC2-4 are not allowed in Subset C.



**Note 2:** Code 128 has automatic selection of start character and character subset (that is, selects optimal start character and handles shift and changes of subset depending on the content of the input data).

**Remarks**

Code 128A, Code 128B, and Code 128C select subsets A, B, and C respectively. The last character in the bar code name signifies both the start character and the chosen subset.

The selected subset can be changed anywhere in the input string, either for a single character using a Shift character (not for Subset C), or for the remainder of the input string using a Code character (all subsets). The Shift and Code characters consist of a combination of two characters:

The Shift and Code characters consist of a combination of two characters:

- Two left-pointing double angle quotation marks («) specify a Shift character. Shift character: «« (« = ASCII 171 dec.)
- One left-pointing double angle quotation mark («) specifies a Code character. It should be followed by an uppercase letter that specifies the subset: Code character: « + A|B|C (« = ASCII 171 dec.)

## Data Matrix

### *Separate Commands*

| Command    | Ranges and Settings                                                                                                                                                                                                                                                                                                                             |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:   | “DATAMATRIX”                                                                                                                                                                                                                                                                                                                                    |
| BARRATIO:  | Fixed. Values will be interpreted as BARMAG.                                                                                                                                                                                                                                                                                                    |
| BARMAG:    | <128. Default is 2.                                                                                                                                                                                                                                                                                                                             |
| BARHEIGHT: | Not applicable.                                                                                                                                                                                                                                                                                                                                 |
| BARFONT    | Set to ON to print the human readable: <ul style="list-style-type: none"> <li>• If ECI support is disabled, the human readable is printed as is.</li> <li>• If ECI support is enabled, the human readable is printed without the ECI-related characters. For example, a bar code string of “\000009abc\123” is printed as “abc\123”.</li> </ul> |

### ***BARSET Data for Data Matrix***

| BARSET     | Instruction                                             | Ranges and Settings                                 |
|------------|---------------------------------------------------------|-----------------------------------------------------|
| [#<ncon>]  | (Optional) Specifies the start parameter in the syntax. |                                                     |
| <sexp>     | Bar code type                                           |                                                     |
| <nexp1-2>  | Not applicable                                          |                                                     |
| <nexp3>    | Barmag/Enlargement                                      | < 128. Default is 2.                                |
| <nexp4-10> | Not applicable                                          |                                                     |
| <nexp11>   | Enables or disables ECI support.                        | 0: Disables ECI support.<br>1: Enables ECI support. |

### ***Input Data for Data Matrix***

| Parameter            | Ranges and Settings    | Notes                |
|----------------------|------------------------|----------------------|
| Number of characters | 2,335 ASCII characters |                      |
| Check digit          | Yes                    | Added automatically. |
| Input characters     | ASCII 0-255 decimal    |                      |



**Note 1:** ECC 200 type is used for this Data Matrix symbol. ECC 200 data may be encoded using any of the encodation schemes ASCII, C40, Text, X12, EDIFACT or Base 256. ASCII encodation is the default scheme. All other encodation schemes are invoked from ASCII encodation and return to this scheme.



**Note 2:** The number of characters determines the size of the symbol. For example, the data “123456” generates a 10 row × 10 column Data Matrix symbol, and 72 digits generate a 24 row × 24 column Data Matrix symbol. Fewer characters can be used for all symbols (10 × 10, 12 × 12, and so on up to 144 × 144) if the data includes non-numeric characters.

### Remarks

When Data Matrix is selected and ECI support is enabled ( $<nextp11> = 1$ ), the strings “\<nnnnnn>” and “\\” are interpreted as follows:

- \<nnnnnn> The backslash indicates that the character set should be changed, and *nnnnnn* is the number of the chosen character set. The string is decoded as an ECI character followed by one to three digit numbers in the bar code. All ECI characters are printed in ASCII encodation, meaning that this string itself can be encoded in any mode. The algorithm switches to ASCII mode to encode the ECI characters, then encodes the other characters in the most efficient modes.
- \\" Two backslashes are decoded as a single backslash character (“\\”) in the bar code. All double backslashes are printed in the most efficient modes as specified and are decoded the same way when ECI support is not enabled.

Error 1112 (ECI syntax error) occurs if ECI mode is enabled and the PRBAR input string is invalid.

**EAN-8*****Separate Commands***

| <b>Command</b> | <b>Ranges and Settings</b>                                  |
|----------------|-------------------------------------------------------------|
| BARTYPE:       | “EAN8”                                                      |
| BARRATIO:      | Fixed ratio. BARRATIO statement ignored.                    |
| BARMAG:        | Maximum 8. Default is 2.                                    |
| BARHEIGHT:     | No restrictions. Default is 100.                            |
| BARFONT        | Barfont generated automatically. BARFONT statement ignored. |

***Input Data for EAN-8***

| <b>Parameter</b>     | <b>Ranges and Settings</b> | <b>Notes</b>         |
|----------------------|----------------------------|----------------------|
| Number of characters | 7                          |                      |
| Check digit:         | One                        | Added automatically. |
| Digits               | 0 to 9                     |                      |
| Uppercase letters:   | No                         |                      |
| Lowercase letters:   | No                         |                      |
| Punctuation marks:   | No                         |                      |
| Start character:     | No                         |                      |
| Stop character:      | No                         |                      |

**EAN-13*****Separate Commands***

| <b>Command</b> | <b>Ranges and Settings</b>                                                          |
|----------------|-------------------------------------------------------------------------------------|
| BARTYPE:       | “EAN13”                                                                             |
| BARRATIO:      | Fixed ratio. BARRATIO statement ignored.                                            |
| BARMAG:        | Maximum 8. Default is 2.                                                            |
| BARHEIGHT:     | No restrictions. Default is 100.                                                    |
| BARFONT        | Barfont generated automatically. BARFONT statement ignored.<br>BARFONT ON/OFF work. |

***Input Data for EAN-13***

| <b>Parameter</b>     | <b>Ranges and Settings</b> | <b>Notes</b>         |
|----------------------|----------------------------|----------------------|
| Number of characters | 12                         |                      |
| Check digit          | One                        | Added automatically. |
| Digits               | 0 to 9                     |                      |
| Uppercase letters    | No                         |                      |
| Lowercase letters    | No                         |                      |
| Punctuation marks    | No                         |                      |
| Start character      | No                         |                      |
| Stop character       | No                         |                      |

## EAN-128

This bar code is identical to Code 128 with the exception that the initial FNC1 function character is generated automatically.

### Separate Commands

| Command    | Ranges and Settings                             |
|------------|-------------------------------------------------|
| BARTYPE:   | “EAN128”<br>“EAN128A”<br>“EAN128B”<br>“EAN128C” |
| BARRATIO:  | Fixed ratio. BARRATIO statement ignored.        |
| BARMAG:    | = 2 (default).                                  |
| BARHEIGHT: | No restrictions. Default is 100.                |
| BARFONT    | No restrictions.                                |

### BARSET Data for EAN-128

| BARSET     | Description                                                         | Ranges and Settings                                                                                                                         |
|------------|---------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| [#<ncon>]  | (Optional) Specifies the start parameter in the syntax.             |                                                                                                                                             |
| <sexp>     | Bar code type.                                                      | “EAN128”<br>“EAN128A”<br>“EAN128B”<br>“EAN128C”                                                                                             |
| <nexp1-2>  | Not applicable                                                      |                                                                                                                                             |
| <nexp3>    | Barmag/Enlargement.                                                 | <128. Default is 2.                                                                                                                         |
| <nexp4>    | Barheight.                                                          |                                                                                                                                             |
| <nexp5>    | Not applicable                                                      |                                                                                                                                             |
| <nexp6>    | Character exclusion in bar code.<br>Does not affect human readable. | 0: Disables parentheses and spaces in the bar code data.<br>Non-zero: Bar code and human readable field will include exactly the same data. |
| <nexp7-10> | Not applicable                                                      |                                                                                                                                             |

### Input Data for EAN-128

| Parameter            | Ranges and Settings             | Notes                               |
|----------------------|---------------------------------|-------------------------------------|
| Number of characters | Unlimited                       |                                     |
| Check digit          | Trailing symbol check character | Added automatically.                |
| Input characters     | ASCII 0-127                     | According to Roman 8 character set. |
| Start characters     | See Note 2.                     |                                     |
| Code characters      | See Notes 1 & 2                 |                                     |
| Shift characters     | See Notes 1 & 2                 |                                     |
| Stop characters:     | Always                          | Added automatically.                |



**Note 1:** Code characters and shift characters require either an 8-bit communication protocol, remapping to an ASCII value between 0-127 dec., or the use of a CHR\$ function.



**Note 2:** EAN 128 has automatic selection of start character and character subset (that is, selects optimal start character and handles shift and changes of subset depending on the content of the input data).

EAN 128A, EAN 128B, and EAN 128C select subsets A, B, and C respectively. The last character in the bar code name signifies both the start character and the chosen subset.

The selected subset can be changed anywhere in the input string, either for a single character using a Shift character (not for Subset C), or for the remainder of the input string using a Code character (all subsets).

The Shift and Code characters consist of a combination of two characters:

- Two left-pointing double angle quotation marks («) specify a Shift character.  
Shift character: «« (« = ASCII 171 dec.)
- One left-pointing double angle quotation mark («) specifies a Code character. It should be followed by an uppercase letter that specifies the subset:  
Code character: « + A|B|C (« = ASCII 171 dec.)

## Interleaved 2 of 5

Interleaved 2 of 5 is a numeric code where input digits are encoded in pairs. If an odd number of digits is entered, a leading zero is added automatically.

### Separate Commands

| Command    | Ranges and Settings              |
|------------|----------------------------------|
| BARTYPE:   | “INT2OF5”                        |
| BARRATIO:  | 2:1-3:1/ Default: 3:1            |
| BARMAG:    | No restrictions. Default is 2.   |
| BARHEIGHT: | No restrictions. Default is 100. |
| BARFONT    | No restrictions.                 |

***Input Data for Interleaved 2 of 5***

| Parameter            | Ranges and Settings | Notes               |
|----------------------|---------------------|---------------------|
| Number of characters | Unlimited           |                     |
| Check digit          | No                  |                     |
| Digits               | 0 to 9              |                     |
| Uppercase letters    | No                  |                     |
| Lowercase letters    | No                  |                     |
| Punctuation marks    | No                  |                     |
| Start character      | Yes                 | Added automatically |
| Stop character       | Yes                 | Added automatically |

**MaxiCode*****Separate Commands***

| Command    | Ranges and Settings            |
|------------|--------------------------------|
| BARTYPE:   | “MAXICODE”                     |
| BARRATIO:  | Not applicable. Input ignored. |
| BARMAG:    | Not applicable. Input ignored. |
| BARHEIGHT: | Not applicable. Input ignored. |
| BARFONT    | Not applicable. Input ignored. |

**Remarks**

MaxiCode requires 8 fields of data separated by an LF character (entered as CHR\$(10)):

- Maximum number of characters: 84 (Modes 2 and 3) or 138 (Mode 4).
- Check character is automatic (Reed-Solomon algorithm).
- Data type: ASCII 0 to 255.

For all modes, each field must be present in final in data string and must contain valid data as described in the next table:

***MaxiCode Data Field Information***

| Field | Mode    | Data Type and Range                                                 | Notes                                                                                   |
|-------|---------|---------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| 1     | 2, 3    | Mode 2 - 5 numeric characters<br>Mode 3 - 6 alphanumeric characters |                                                                                         |
| 2     | 2       | 4 numeric characters [0-9999]                                       |                                                                                         |
| 3     | 2, 3    | 3 numeric characters [0-999]                                        | Country code.                                                                           |
| 4     | 2, 3    | 3 numeric characters [0-999]                                        | Service class.                                                                          |
| 5     | 2, 3, 4 | User-defined message                                                |                                                                                         |
| 6     |         | 1 numeric character [2 3 4]                                         | Mode selector.                                                                          |
| 7     |         | 1 numeric character [1-8]                                           | Position in structured append.<br>See the next section, “About Structured Append Mode.” |
| 8     |         | 1 numeric character                                                 | Total number of symbols in structure.                                                   |

## About Structured Append Mode

F8 in structured append is the trigger for structured append mode:

- If F8 >1, the two first code words in secondary message will be a pad followed by position code word. F8 has higher precedence than F7.
- If F7 >1 when F8 = 1, the two first code words do not signal structured append.
- If F8 >1, F7 may be >F8 without error and structured append codeword will signal given values.

## Primary and Secondary Message Data Elements

The primary and secondary message data elements are described in the next tables.



**Note:** All primary message data elements are required for Mode 2 (structured data message for U.S. destinations) and Mode 3 (structured data message for international destinations).

### Primary Message Data Elements

| Element                      | Mode and Length                                 | Sample             |
|------------------------------|-------------------------------------------------|--------------------|
| Zip code + 4-digit extension | Mode 2 (Numeric): 9<br>Mode 3 (Alphanumeric): 6 | 152392802<br>B1050 |
| Country code                 | Mode 2 (Numeric): 3<br>Mode 3 (Numeric): 3      | 840                |
| Service class                | Mode 2 (Numeric): 3<br>Mode 3 (Numeric): 3      | 001                |

### Secondary Message Data Elements

| Element           | Mode and Length                                                                         | Sample                                                        |
|-------------------|-----------------------------------------------------------------------------------------|---------------------------------------------------------------|
| Secondary message | Mode 2 (Alphanumeric): 84<br>Mode 3 (Alphanumeric): 84<br>Mode 4 (Standard symbol): 138 | This is the secondary message.                                |
| Header (optional) | Mode 2 (Numeric): 9<br>Mode 3 (Numeric): 9                                              | [ ) <sup>R</sup> <sub>S</sub> 01 <sup>G</sup> <sub>S</sub> 97 |

### Example

This example creates a MaxiCode symbol based on the following information:

|                      |            |
|----------------------|------------|
| Zip Code:            | 84170      |
| Zip Code Extension:  | 1280       |
| Country Code:        | 840        |
| Class of Service:    | 001        |
| Message Header:      | []><RS>01  |
| Year:                | 07         |
| Tracking Number:     | 1Z12345675 |
| SCAC:                | UPSN       |
| UPS Shipper Number:  | 12345E     |
| Julian Day of Pickup | 089        |
| Shipment ID:         | 1324567    |
| Package:             | 1/1        |
| Weight:              | 10.1       |
| Address Validation:  | Y          |
| Ship to Street       | 1 Main ST  |
| Ship to City:        | PITTSBURGH |
| Ship to State:       | PA         |

The syntax for this information is:

```

10 PRPOS 100,100
20 DIR 1
30 ALIGN 1
40 a$="84170"+CHR$(10)+"1280"+CHR$(10)+"840"+
 CHR$(10)+"001"+CHR$(10)+"]">"+CHR$(30)+
 "01"+CHR$(29)+"07"+"1Z12345675"+CHR$(29)+"
 "UPSN"+CHR$(29)+"12345E"+CHR$(29)+"089"+
 +CHR$(29)+"1234567"+CHR$(29)+"1/1"+CHR$(29)+"
 +"10.1"+CHR$(29)+"Y"+CHR$(29)+"1 MAIN ST"
50 b$= CHR$(29)+"PITTSBURGH"+CHR$(29)+"PA"
 +CHR$(29)+CHR$(30)+CHR$(4)+CHR$(10)+"2"+
 +CHR$(10)+"1"+CHR$(10)+"1"
60 BARTYPE "MAXICODE"
70 PRBAR a$;b$
80 PRINTFEED
RUN

```

### MicroPDF417

MicroPDF417 is a multi-row symbology based on PDF417. A limited set of symbol sizes is available where each size has a fixed level of error correction. Up to 250 alphanumeric characters or 366 numeric digits can be encoded in a symbol.



**Note:** Enhanced applications such as Extended Channel Interpretation (ECI), structured append, reader initialization, Code 128 emulation, and macro characters are not supported.

***BARSET Data for MicroPDF417***

| <b>BARSET</b> | <b>Instruction</b>                                      | <b>Ranges and Settings</b>             |
|---------------|---------------------------------------------------------|----------------------------------------|
| [#<ncon>]     | (Optional) Specifies the start parameter in the syntax. |                                        |
| <sexp>        | Bar code type                                           | "MICROPDF417"                          |
| <nexp1-2>     | Not applicable                                          |                                        |
| <nexp3>       | Element width in dots                                   | 1 to 21. Default is 2.                 |
| <nexp4>       | Element height in dots                                  | 1 to 127. Default is 100.              |
| <nexp5-7>     | Not applicable                                          |                                        |
| <nexp8>       | Number of rows                                          | 0, 4 to 44. Default is 0 (=automatic). |
| <nexp9>       | Number of columns                                       | 0 to 4. Default is 0 (=automatic).     |
| <nexp10>      | Not applicable                                          |                                        |

***Setting the Number of Rows and Columns***

The symbol size is defined by specifying the number of rows and columns. Not all combinations of rows and columns are allowed. The next table illustrates the valid combinations.

***Valid Row and Column Combinations for MicroPDF417***

| <b>Total Columns</b> | <b>Valid Number of Rows for Total Columns</b> |    |    |    |    |    |    |    |    |    |    |   |
|----------------------|-----------------------------------------------|----|----|----|----|----|----|----|----|----|----|---|
| 1                    | 11                                            | 14 | 17 | 20 | 24 | 28 | -  | -  | -  | -  | -  | - |
| 2                    | 8                                             | 11 | 14 | 17 | 20 | 23 | 26 | -  | -  | -  | -  | - |
| 3                    | 6                                             | 8  | 10 | 12 | 15 | 20 | 26 | 32 | 38 | 44 | -  | - |
| 4                    | 4                                             | 6  | 8  | 10 | 12 | 15 | 20 | 26 | 32 | 38 | 44 | - |

If the number of rows is set to a value that does not match the valid values for the given number of columns, the printer automatically chooses a larger number from the list of valid values.

***Automatic Selection***

The number of columns and rows can be set automatically by the printer. If the number of columns is set to 0, the printer sets the number of columns and rows automatically, regardless of the number of rows specified. The printer tries to fit the given data into a symbol with as few columns as possible. If the number of columns is non-zero and the number of rows is set to 0, the printer automatically sets the number of rows to the lowest number required to encode the given data.

***Examples***

This example shows how a MicroPDF417 bar code is specified using the BARTYPE and BARSET statements.

Bar width: 2 dots

Bar height: 8 dots

Number of rows: 26

Number of columns: 3

```
BARTYPE "MICROPDF417"
BARSET #4,2,8,1,1,1,26,3
```



**Note:** The bar width and bar height can also be set using BARMAG and BARHEIGHT respectively.

The number of columns and rows are set using the Fingerprint statement BARSET. Parameters number 9 and 10 are the number of rows and columns respectively. Examples A and B below set the number of rows to 12 and the number of columns to 3. The type of bar code is set to MicroPDF417. Not all parameters of the BARSET command are applicable to the MicroPDF417 implementation. The parameters ignored by the implementation are set to ‘1’ in example B (large bar ratio, small bar ratio, security level, aspect height, aspect width).

### Example A (Direct Protocol)

```
BARTYPE "MICROPDF417"
BARSET #9, 12
BARSET #10, 3
```

### Example B (Direct Protocol)

```
BARSET "MICROPDF417",1,1,2,8,1,1,1,12,3
```

The example code below prints a small MicroPDF417 bar code containing the string “MicroPDF417.” The number of rows and columns is set by the printer based on the input string since the number of columns is set to 0.

```
10 BARSET "MICROPDF417",1,1,4,8,1,1,1,0,0
20 PRPOS 50, 50
30 PRBAR "MICROPDF417"
40 PRINTFEED
```

## PDF417

### Separate Commands

| Command    | Ranges and Settings                      |
|------------|------------------------------------------|
| BARTYPE:   | “PDF417”                                 |
| BARRATIO:  | Fixed. Values are interpreted as BARMAG. |
| BARMAG:    | 2 (default) to 128.                      |
| BARHEIGHT: | No restrictions. Default is 100.         |
| BARFONT    | Not applicable                           |

***BARSET Data for PDF417***

| <b>BARSET</b> | <b>Description</b>                                      | <b>Ranges and Settings</b>                                 |
|---------------|---------------------------------------------------------|------------------------------------------------------------|
| [#<ncon>]     | (Optional) Specifies the start parameter in the syntax. |                                                            |
| <sexp>        | Bar code type                                           | “PDF417”                                                   |
| <nexp1>       | Light bar ratio                                         | No restrictions. Default is 3.                             |
| <nexp2>       | Narrow bar ratio                                        | No restrictions. Default is 1.                             |
| <nexp3>       | Barmag/Enlargement                                      | < 128. Default is 2.                                       |
| <nexp4>       | Barheight                                               | < 500. Default is 100.                                     |
| <nexp5>       | Security level                                          | 1-5. Default is 2.                                         |
| <nexp6>       | Aspect height ratio                                     | No restrictions. Default is 3.                             |
| <nexp7>       | Aspect width ratio                                      | No restrictions. Default is 1.                             |
| <nexp8>       | Number of rows                                          | No restrictions. Default is 0.                             |
| <nexp9>       | Number of columns                                       | No restrictions. Default is 0.                             |
| <nexp10>      | Truncate                                                | 0 or not equal to 0. The default 0 prints a normal symbol. |

***Input Data for PDF417***

| <b>Parameter</b>     | <b>Ranges or Settings</b>               | <b>Notes</b>                      |
|----------------------|-----------------------------------------|-----------------------------------|
| Number of characters | 1,800 ASCII characters or 2,700 digits. | Depending of level of compactness |
| Check digit          | Yes                                     | Added automatically.              |
| Input characters     | ASCII 0-255 decimal                     |                                   |

**Example**

This example shows PDF417 in GM label as per ANSI B-14, with the following data:

| Parameter                 | Data or Setting      | Description            |
|---------------------------|----------------------|------------------------|
| Data Identifier/Separator | Data                 | Field Name             |
| ]>{RS}                    |                      | Message Header         |
| 06{GS}                    |                      | Format Header          |
| P                         | 12345678             | Part Number            |
| {GS}                      |                      | Group Separator        |
| Q                         | 160                  | Quantity               |
| {GS}                      |                      |                        |
| 1J                        | UN123456789A2B4C6D8E | License Plate          |
| {GS}                      |                      |                        |
| 20L                       | LA6-987              | Material Handling Code |
| {GS}                      |                      |                        |
| 21L                       | S4321ZES             | Plant/Dock Code        |
| {GS}                      |                      |                        |
| K                         | GM1234               | PO Number              |
| {GS}                      |                      |                        |
| 15K                       | G1155                | Kanban Number          |
| {GS}                      |                      |                        |
| B                         | KLT3214              | Container Type         |
| {GS}                      |                      |                        |
| 7Q                        | 10GT                 | Gross Weight           |
| {RS}                      |                      | Record Separator       |
| {EOT}                     |                      | Message Trailer        |

The syntax is as follows:

```

10 PRPOS 16,1180
20 DIR 4
30 ALIGN 9
40 BARSET "PDF417",1,1,2,6,5,1,2,0,5,0
50 PRBAR " [) >" +CHR$(30) +"06"+CHR$(29) +
 "P12345678" +CHR$(29) +"Q160" +CHR$(29) +
 "1JUN123456789A2B4C6D8E" +CHR$(29) +
 "20LA6-987" +CHR$(29) +"21L54321 ZES" +
 +CHR$(29) +"KGM1234" +CHR$(29) +"15KG1155" +
 +CHR$(29) +"BKLT3214" +CHR$(29) +"7Q10GT" +
 +CHR$(30) +CHR$(4)
60 PRINTFEED
RUN

```

## QR Code

### **BARSET Data for QR Code**

| <b>BARSET</b> | <b>Description</b>                                      | <b>Ranges and Settings</b>             |
|---------------|---------------------------------------------------------|----------------------------------------|
| [#<ncon>]     | (Optional) Specifies the start parameter in the syntax. |                                        |
| <sexp>        | Bar code type                                           | "QRCODE"                               |
| <nexp1-2>     | Not applicable                                          |                                        |
| <nexp3>       | Element size in dots                                    | 1 to 27. Default is 2.                 |
| <nexp4>       | QR Code model                                           | 1 (default) or 2 (recommended)         |
| <nexp5>       | Security level                                          | 1: L<br>2: M (default)<br>3: Q<br>4: H |
| <nexp6>       | Mask pattern selection                                  | Not currently supported. Reserved.     |
| <nexp7-10>    | Not applicable                                          |                                        |

### ***Input Data Capacity - QR Code***

| <b>Security Level</b> | <b>Correction Level</b> | <b>Numeric</b> | <b>Alphanumeric</b> | <b>8-bit Byte Data</b> | <b>Kanji</b> |
|-----------------------|-------------------------|----------------|---------------------|------------------------|--------------|
| L                     | 7%                      | Model 1: 1176  | 707                 | 486                    | 299          |
|                       |                         | Model 2: 7089  | 4296                | 2953                   | 1817         |
| M                     | 15%                     | Model 1: 877   | 531                 | 365                    | 225          |
|                       |                         | Model 2: 5596  | 3391                | 2331                   | 1435         |
| Q                     | 25%                     | Model 1: 738   | 447                 | 307                    | 189          |
|                       |                         | Model 2: 3993  | 2420                | 1663                   | 1024         |
| H                     | 30%                     | Model 1: 498   | 302                 | 207                    | 127          |
|                       |                         | Model 2: 3057  | 1842                | 1273                   | 784          |

The unit is number of characters. Mixed mode is supported for all combinations except or combinations containing both 8-bit byte and Kanji data. The type of data is set automatically by the implementation based on the input characters.

### **Example**

This example shows how BARSET is used in two different ways to create a QR Code Model 2 bar code with element size 4 and security code M:

```
BARSET "QRCode", 1, 1, 4, 2, 2
```

or

```
BARSET #4, "QRCode", 4, 201, 2
```

## RSS-14

### *Separate Commands*

| Command    | Ranges and Settings                                                                                                            |
|------------|--------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:   | “RSS14”                                                                                                                        |
| BARRATIO:  | Not applicable.                                                                                                                |
| BARMAG:    | Specifies the x-dimension width in dots of the most narrow element for both the linear bar code and the 2D code. Default is 2. |
| BARHEIGHT: | Specifies the height of the linear bar code. Default is 100. See the Remarks for more information.                             |
| BARFONT    | Not applicable.                                                                                                                |

### ***BARSET Data for RSS-14***

| BARSET     | Description                                             | Ranges and Settings  |
|------------|---------------------------------------------------------|----------------------|
| [#<ncon>]  | (Optional) Specifies the start parameter in the syntax. |                      |
| <sexp>     | Bar code type                                           | “RSS14”              |
| <nexp1>    | Placeholder                                             | Value insignificant. |
| <nexp2>    | Placeholder                                             | Value insignificant. |
| <nexp3>    | Barmag/Enlargement                                      | See BARMAG.          |
| <nexp4>    | Barheight                                               | See BARHEIGHT.       |
| <nexp5>    | Placeholder                                             | Value insignificant. |
| <nexp6>    | Placeholder                                             | Value insignificant. |
| <nexp7>    | Placeholder                                             | Value insignificant. |
| <nexp8-10> | Not applicable                                          |                      |

### ***Input Data for RSS-14***

| Parameter            | Ranges and Settings | Notes                                                                                                       |
|----------------------|---------------------|-------------------------------------------------------------------------------------------------------------|
| Number of characters | 13 digits           | If less than 13 digits are entered, leading zeroes are automatically added so the string is 13 digits long. |

### **Remarks**

There are restrictions in the standard for the minimum size for each RSS bar code, even if it is possible to print an RSS bar code in any height. The height should relate to the magnification. RSS Stacked differs because the bar code rows do not have the same height. BARHEIGHT or BARSET<nexp4> specifies the height of the lower row, and the height of the upper row is automatically calculated from the height of the lower row.

For RSS-14, the width is 96X and the minimum height is 33X, where X is the width of the most narrow element as specified by BARMAG or BARSET<nexp3>.

## RSS-14 Truncated

### *Separate Commands*

| Command    | Ranges and Settings                                                                                                            |
|------------|--------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:   | “RSS14T”                                                                                                                       |
| BARRATIO:  | Not applicable.                                                                                                                |
| BARMAG:    | Specifies the x-dimension width in dots of the most narrow element for both the linear bar code and the 2D code. Default is 2. |
| BARHEIGHT: | Specifies the height of the linear bar code. Default is 100. See the Remarks for more information.                             |
| BARFONT    | No restriction.                                                                                                                |

### ***BARSET Data for RSS-14T***

| BARSET     | Description                                             | Range and Settings   |
|------------|---------------------------------------------------------|----------------------|
| [#<ncon>]  | (Optional) Specifies the start parameter in the syntax. |                      |
| <sexp>     | Bar code type                                           | “RSS14T”             |
| <nexp1>    | Placeholder                                             | Value insignificant. |
| <nexp2>    | Placeholder                                             | Value insignificant. |
| <nexp3>    | Barmag/Enlargement                                      | See BARMAG.          |
| <nexp4>    | Barheight                                               | See BARHEIGHT.       |
| <nexp5>    | Placeholder                                             | Value insignificant. |
| <nexp6>    | Placeholder                                             | Value insignificant. |
| <nexp7>    | Placeholder                                             | Value insignificant. |
| <nexp8-10> | Not applicable                                          |                      |

### ***Input Data for RSS-14T***

| Parameter            | Ranges and Settings | Notes                                                                                                                |
|----------------------|---------------------|----------------------------------------------------------------------------------------------------------------------|
| Number of characters | 13 digits           | If less than 13 digits are entered, leading zeroes will automatically be added so the string will be 13 digits long. |

### **Remarks**

There are restrictions in the standard for the minimum size for each RSS bar code, even if it is possible to print an RSS bar code in any height. The height should relate to the magnification. RSS Stacked differs, because the bar code rows do not have the same height. BARHEIGHT or BARSET<*nexp4*> specifies the height of the lower row, and the height of the upper row is automatically calculated from the height of the lower row.

For RSS-14 Truncated, the width is 96X and the minimum height is 13X, where X is the width of the most narrow element as specified by BARMAG or BARSET<*nexp3*>.

## RSS-14 Stacked

### *Separate Commands*

| Command    | Ranges and Settings                                                                                                            |
|------------|--------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:   | “RSS14S”                                                                                                                       |
| BARRATIO:  | Not applicable.                                                                                                                |
| BARMAG:    | Specifies the x-dimension width in dots of the most narrow element for both the linear bar code and the 2D code. Default is 2. |
| BARHEIGHT: | Specifies the height of the linear bar code. Default is 100. See the Remarks for more information.                             |
| BARFONT    | No restrictions.                                                                                                               |

### ***BARSET Data for RSS-14S***

| BARSET     | Description                                             | Ranges and Settings                                                                                                                                                                                                                                                                         |
|------------|---------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [#<ncon>]  | (Optional) Specifies the start parameter in the syntax. |                                                                                                                                                                                                                                                                                             |
| <sexp>     | Bar code type                                           | “RSS14S”                                                                                                                                                                                                                                                                                    |
| <nexp1-2>  | Placeholder                                             | Value insignificant.                                                                                                                                                                                                                                                                        |
| <nexp3>    | Barmag/Enlargement                                      | See BARMAG.                                                                                                                                                                                                                                                                                 |
| <nexp4>    | Barheight                                               | See BARHEIGHT.                                                                                                                                                                                                                                                                              |
| <nexp5-6>  | Placeholder                                             | Value insignificant.                                                                                                                                                                                                                                                                        |
| <nexp7>    | Placeholder                                             | The height is the same for the pattern row between the 2D bar code and the linear bar code as for the pattern rows between the linear rows. If too low a height is entered, the height will be changed to the smallest legal height for the selected magnification. Minimum 1X, maximum 2X. |
| <nexp8-10> | Not applicable                                          |                                                                                                                                                                                                                                                                                             |

### ***Input Data for RSS-14S***

| Parameter            | Ranges and Settings | Notes                                                                                                       |
|----------------------|---------------------|-------------------------------------------------------------------------------------------------------------|
| Number of characters | 13 digits           | If less than 13 digits are entered, leading zeroes are added automatically so the string is 13 digits long. |

### **Remarks**

There are restrictions in the standard for the minimum size for each RSS bar code, even if it is possible to print an RSS bar code in any height. The height should relate to the magnification. RSS Stacked differs, because the bar code rows do not have the same height. BARHEIGHT or BARSET<nexp4> specifies the height of the lower row and height of the upper row is automatically calculated from the height of the lower row.

For RSS-14S, the width is 50X and the minimum height is 13X (upper 5X + lower 7X + separator 1X min.), where X is the width of the most narrow element as specified by BARMAG or BARSET<nexp3>.

**Example**

This example creates an RSS14S bar code with the following characteristics and with recommended minimum height selected:

Place holder (nexp1): 1  
 Place holder (nexp2): 1  
 Most narrow element width in dots: 3  
 Height in dots: 21  
 Place holder (nexp5): 1  
 Place holder (nexp6): 1  
 Separator pattern row height: 4  
 Data: 1234567890123

```
BARSET "RSS14S",1,1,3,21,1,1,4
PRBAR "1234567890123"
```

**RSS-14 Stacked Omnidirectional*****Separate Commands***

| Command    | Ranges and Settings                                                                                                            |
|------------|--------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:   | "RSS14SO"                                                                                                                      |
| BARRATIO:  | Not applicable.                                                                                                                |
| BARMAG:    | Specifies the x-dimension width in dots of the most narrow element for both the linear bar code and the 2D code. Default is 2. |
| BARHEIGHT: | Specifies the height of the linear bar code. Default is 100. See the Remarks for more information.                             |
| BARFONT    | No restrictions.                                                                                                               |

***BARSET Data for RSS-14SO***

| BARSET     | Description                                             | Ranges and Settings                                                                                                                                                                                                                                                                         |
|------------|---------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [#<ncon>]  | (Optional) Specifies the start parameter in the syntax. |                                                                                                                                                                                                                                                                                             |
| <nexp>     | Bar code type                                           | "RSS14SO"                                                                                                                                                                                                                                                                                   |
| <nexp1>    | Placeholder                                             | Value insignificant.                                                                                                                                                                                                                                                                        |
| <nexp2>    | Placeholder                                             | Value insignificant.                                                                                                                                                                                                                                                                        |
| <nexp3>    | Barmag/Enlargement                                      | See BARMAG.                                                                                                                                                                                                                                                                                 |
| <nexp4>    | Barheight                                               | See BARHEIGHT.                                                                                                                                                                                                                                                                              |
| <nexp5>    | Placeholder                                             | Value insignificant.                                                                                                                                                                                                                                                                        |
| <nexp6>    | Placeholder                                             | Value insignificant.                                                                                                                                                                                                                                                                        |
| <nexp7>    | Placeholder                                             | The height is the same for the pattern row between the 2D bar code and the linear bar code as for the pattern rows between the linear rows. If too low a height is entered, the height will be changed to the smallest legal height for the selected magnification. Minimum 1X, maximum 2X. |
| <nexp8-10> | Not applicable                                          |                                                                                                                                                                                                                                                                                             |

***Input Data for RSS-14SO***

| Parameter            | Ranges and Settings | Notes                                                                                                       |
|----------------------|---------------------|-------------------------------------------------------------------------------------------------------------|
| Number of characters | 13 digits           | If less than 13 digits are entered, leading zeroes are added automatically so the string is 13 digits long. |

**Remarks**

There are restrictions in the standard for the minimum size for each RSS bar code, even if it is possible to print an RSS bar code in any height. The height should relate to the magnification. RSS Stacked differs, because the bar code rows do not have the same height. BARHEIGHT or BARSET<exp4> specifies the height of the lower row and height of the upper row is automatically calculated from the height of the lower row.

For RSS-14SO, the width is 50X and the minimum height is 69X (upper 33X + lower 33X + separator 3\*1X min.).

**RSS-14 Limited*****Separate Commands***

| Command    | Ranges and Settings                                                                                                            |
|------------|--------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:   | “RSS14L”                                                                                                                       |
| BARRATIO:  | Not applicable.                                                                                                                |
| BARMAG:    | Specifies the x-dimension width in dots of the most narrow element for both the linear bar code and the 2D code. Default is 2. |
| BARHEIGHT: | Specifies the height of the linear bar code. Default is 100. See the Remarks for more information.                             |
| BARFONT    | No restrictions.                                                                                                               |

***BARSET Data for RSS-14 Limited***

| BARSET     | Description                                            | Ranges and Settings  |
|------------|--------------------------------------------------------|----------------------|
| [#<ncon>]  | (Optional) Specifies the start parameter in the syntax |                      |
| <sexp>     | Bar code type                                          | “RSS14L”             |
| <nexp1-2>  | Placeholder                                            | Value insignificant. |
| <nexp3>    | Barmag/Enlargement                                     | See BARMAG.          |
| <nexp4>    | Barheight                                              | See BARHEIGHT.       |
| <nexp5-7>  | Placeholder                                            | Value insignificant. |
| <nexp8-10> | Not applicable                                         |                      |

***Input Data for RSS-14 Limited***

| Parameter            | Ranges and Settings | Notes                                                                                                       |
|----------------------|---------------------|-------------------------------------------------------------------------------------------------------------|
| Number of characters | 13 digits           | If less than 13 digits are entered, leading zeroes are automatically added so the string is 13 digits long. |

**Remarks**

There are restrictions in the standard for the minimum size for each RSS bar code, even if it is possible to print an RSS bar code in any height. The height should relate to the magnification. RSS Stacked differs, because the bar code rows do not have the same height. BARHEIGHT or BARSET<exp4> specifies the height of the lower row, and the height of the upper row is automatically calculated from the height of the lower row.

For RSS-14L, the width is 71X and the minimum height is 10X, where X is the width of the most narrow element as specified by BARMAG or BARSET<exp3>.

**RSS-14 Expanded****Separate Commands**

| <b>Command</b> | <b>Ranges and Settings</b>                                                                                                     |
|----------------|--------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:       | “RSS14E”                                                                                                                       |
| BARRATIO:      | Not applicable.                                                                                                                |
| BARMAG:        | Specifies the x-dimension width in dots of the most narrow element for both the linear bar code and the 2D code. Default is 2. |
| BARHEIGHT:     | Specifies the height of the linear bar code. Default is 100. See the Remarks for more information.                             |
| BARFONT        | No restrictions.                                                                                                               |

**BARSET Data for RSS-14 Expanded**

| <b>BARSET</b> | <b>Description</b>                                      | <b>Ranges and Settings</b> |
|---------------|---------------------------------------------------------|----------------------------|
| [#<ncon>]     | (Optional) Specifies the start parameter in the syntax. |                            |
| <sexp>        | Bar code type                                           | “RSS14L”                   |
| <exp1>        | Placeholder                                             | Value insignificant.       |
| <exp2>        | Placeholder                                             | Value insignificant.       |
| <exp3>        | Barmag/Enlargement                                      | See BARMAG.                |
| <exp4>        | Barheight                                               | See BARHEIGHT.             |
| <exp5>        | Placeholder                                             | Value insignificant.       |
| <exp6>        | Placeholder                                             | Value insignificant.       |
| <exp7>        | Placeholder                                             | Value insignificant.       |
| <exp8-10>     | Not applicable                                          |                            |

**Input Data for RSS-14 Expanded**

| <b>Parameter</b>     | <b>Ranges and Settings</b>                    | <b>Notes</b>                                                                                             |
|----------------------|-----------------------------------------------|----------------------------------------------------------------------------------------------------------|
| Number of characters | Max. 71 numeric or 41 alphanumeric characters | Allowed characters:<br>0-9, A-Z, a-z, ! " % & ' ( ) * + , - . / : ; < = > ? _ space<br>FNC1 [CHR\$(128)] |

**Remarks**

Use RSS-14 Expanded for intelligent encoding of the input data. RSS-14E bar codes can be created with different encoding methods and compressed data fields. To understand how to create intelligent bar codes with RSS-14E, see the most current AIM specification.

**RSS-14 Expanded Stacked*****Separate Commands***

| Command    | Ranges and Settings                                                                                                            |
|------------|--------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:   | “RSS14ES”                                                                                                                      |
| BARRATIO:  | Not applicable.                                                                                                                |
| BARMAG:    | Specifies the x-dimension width in dots of the most narrow element for both the linear bar code and the 2D code. Default is 2. |
| BARHEIGHT: | Specifies the height of the linear bar code. Default is 100. See the Remarks for more information.                             |
| BARFONT    | No restrictions.                                                                                                               |

***BARSET Data for RSS-14 Expanded Stacked***

| BARSET    | Description                                            | Notes                                                                                                                                                                                                    |
|-----------|--------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [#<ncon>] | (Optional) Specifies the start parameter in the syntax |                                                                                                                                                                                                          |
| <sexp>    | Bar code type                                          | “RSS14ES”                                                                                                                                                                                                |
| <nexp1>   | Placeholder                                            | Value insignificant.                                                                                                                                                                                     |
| <nexp2>   | Placeholder                                            | Value insignificant.                                                                                                                                                                                     |
| <nexp3>   | Barmag/Enlargement                                     | See BARMAG.                                                                                                                                                                                              |
| <nexp4>   | Barheight                                              | See BARHEIGHT.                                                                                                                                                                                           |
| <nexp5>   | Segments per row                                       | 2-22 (default 2). Multiples of 2 only.                                                                                                                                                                   |
| <nexp6>   | Placeholder                                            | Value insignificant.                                                                                                                                                                                     |
| <nexp7>   | Height of a separator row                              | The height for the pattern rows between the linear rows. If too low a height is entered, the height will be changed to the smallest legal height for the selected magnification. Minimum 1X, maximum 2X. |
| <nexp8>   | Not applicable                                         |                                                                                                                                                                                                          |
| <nexp9>   | Not applicable                                         |                                                                                                                                                                                                          |
| <nexp10>  | Not applicable                                         |                                                                                                                                                                                                          |

***Input Data for RSS-14 Expanded Stacked***

| Parameter            | Ranges and Settings                | Notes                                                                                           |
|----------------------|------------------------------------|-------------------------------------------------------------------------------------------------|
| Number of characters | Max. 71 numeric or 41 alphanumeric | Allowed characters: 0-9 A-Z a-z ! " % & ' ( ) * + , - . / : ; < = > ? _ space FNC1 [CHR\$(128)] |

### Remarks

There are restrictions in the standard for the minimum size for each RSS bar code, even if it is possible to print an RSS bar code in any height. The height should relate to the magnification. RSS Stacked differs because the bar code rows do not have the same height. BARHEIGHT or BARSET<nexp4> specifies the height of the lower row and height of the upper row is automatically calculated from the height of the lower row.

For RSS-14ES, the width depends on input. The minimum height is 34X per row + 3\*X per separator, where X is the width of the most narrow element as specified by BARMAG or BARSET<nexp3>.

RSS-14ES can be used for intelligent encoding of the input data. RSS-14ES bar codes can be created with different encoding methods and compressed data fields. To understand how to create intelligent bar codes with RSS-14ES, see the most current AIM specification.

### Example

This example is of RSS-14 Expanded Stacked for a variable weight item (0,001 kilogram increments) and with recommended minimum height selected:

|                                        |                                                               |
|----------------------------------------|---------------------------------------------------------------|
| Start parameter:                       | #4                                                            |
| Most narrow element width in dots:     | 2                                                             |
| Height in dots:                        | 68                                                            |
| Segments per row:                      | 4                                                             |
| Place holder <nexp6>:                  | 1                                                             |
| Separator pattern row height:          | 2                                                             |
| DataL                                  | 01900123456789003103001750                                    |
| Explanation of data                    | 01 9001234567890 0 3103 001750                                |
| Application Identifier (AI):           | 01                                                            |
| AI 01 ID:                              | 9001234556780<br>(In this method, the first digit must be 9.) |
| Digit:                                 | 0                                                             |
| Application Identifier (AI):           | 3103                                                          |
| AI3103 variable weight element string: | 001750                                                        |

```
BARTYPE "RSS14ES"
BARSET #4,2,68,4,1,2
PRBAR "01900123456789003103001750"
```

**UPC-A*****Separate Commands***

| Command    | Ranges and Settings                                                                 |
|------------|-------------------------------------------------------------------------------------|
| BARTYPE:   | "UPCA"                                                                              |
| BARRATIO:  | Fixed ratio. BARRATIO statement ignored.                                            |
| BARMAG:    | Maximum 8. Default is 2.                                                            |
| BARHEIGHT: | No restrictions. Default is 100.                                                    |
| BARFONT    | Barfont generated automatically. BARFONT statement ignored.<br>BARFONT ON/OFF work. |

***Input Data for UPC-A***

| Parameter             | Ranges and Settings | Notes                |
|-----------------------|---------------------|----------------------|
| Number of characters: | 11                  |                      |
| Check digit:          | One                 | Added automatically. |
| Digits:               | 0-9                 |                      |
| Uppercase letters:    | No                  |                      |
| Lowercase letters:    | No                  |                      |
| Punctuation marks:    | No                  |                      |
| Start character:      | No                  |                      |
| Stop character:       | No                  |                      |

**UPC-E*****Separate Commands***

| Command    | Ranges and Settings                                                                 |
|------------|-------------------------------------------------------------------------------------|
| BARTYPE:   | "UPCE"                                                                              |
| BARRATIO:  | Fixed ratio. BARRATIO statement ignored.                                            |
| BARMAG:    | Maximum 8. Default is 2.                                                            |
| BARHEIGHT: | No restriction. Default is 100.                                                     |
| BARFONT    | Barfont generated automatically. BARFONT statement ignored.<br>BARFONT ON/OFF work. |

***Input Data for UPC-E***

| Parameters           | Ranges and Settings | Notes                |
|----------------------|---------------------|----------------------|
| Number of characters | 6                   |                      |
| Check digit          | One                 | Added automatically. |
| Digits               | 0-9                 |                      |
| Uppercase letters    | No                  |                      |
| Lowercase letters    | No                  |                      |
| Punctuation marks    | No                  |                      |
| Start character:     | No                  |                      |
| Stop character:      | No                  |                      |

## About AddOn Codes

The EAN and UPC bar code standards allow for two- and five-character supplemental bar codes to be printed together with the normal code.

As of Fingerprint v8.40 it is possible to print these bar codes simply by adding the desired characters to the PRBAR statement, separated by a period (.). The placement of the addon bar code is done automatically.

### Examples

The AddOn characters are added to the PRBAR statement by use of a period (.) character as separator:

```
10 PRPOS 100, 100
20 BARSET "EAN8"
30 ALIGN 1
40 DIR 1
50 BARFONT ON
60 PRBAR "1234567.12345"
70 PRINTFEED
RUN
```

resulting in:



The AddOn code can be used in combination with any EAN and UPC bar code set. The following example prints a composite bar code, with a two character AddOn code. For more information on composite bar codes, see the next section.

```
10 PRPOS 100,100
20 BARSET "EAN8_CC"
30 ALIGN 1
40 DIR 1
50 BARFONT ON
60 PRBAR "1234567.12|987654321"
70 PRINTFEED
RUN
```

resulting in:



## About Composite Bar Codes

An EAN.UCC Composite symbol consists of a linear component and an adjacent 2D Composite Component.

The Composite symbol always includes a linear component so that the primary identification is readable by all scanning technologies. 2D imagers can use the linear component as a finder pattern for the adjacent 2D Composite Component.



EAN.UCC Composite Symbology is dependent on several other symbologies. Some minor adjustment of implementation of PDF417 and MicroPDF417, nine new symbologies and a logical way of combining nine linear and three 2D symbologies in different composite combinations have been implemented.

## About the Linear Components

The linear bar code component encodes the item's primary identification in one of the following barcode types:

- UCC/EAN-128
- EAN/UPC: 8 or 13
- UPC-A or E
- EAN/I{C 8, 13, UPC-A or E
- Any RSS-family symbology that includes a separator character in between the data for the linear component and the 2D component of the input string.

## About the Composite Components

The 2D composite barcode component encodes supplementary data, such as a batch number or expiration date. There are three types:

- CC-A, a variant of MicroPDF417, designed for efficient encoding of supplemental application identifier data.
- CC-B, a MicroPDF417 symbol with codeword 920 in the first data codeword position as a linkage flag denoting EAN.UCC Composite Symbology data compaction.

- CC-C, a PDF417 symbol with a codeword 920 in the first data codeword position as a linkage flag denoting EAN.UCC Composite Symbology data compaction.

The 2D Composite Component, abbreviated as CC, is chosen based on the selected linear component and on the amount of complementary data to be encoded. The choice of linear symbol determines the name of the composite symbol, such as EAN-13 Composite symbol or UCC/EAN-128 Composite symbol.

## About the Human Readable Section

For all combinations of the EAN.UCC Composite Symbology, the human readable section is an optional presentation of the information in either or both bar codes. The linear human readable section is presented below the linear component, within the same rules as for the single bar code. The human readable for the 2D composite component part is presented under the human readable for the linear bar code.

EAN/UPC bar codes have fixed fonts for the human readable bar code interpretation.

## Supported Component Combinations

Based upon the width of the linear component, a choice of “best fit” 2D Composite Component is specified. The table below lists all of the permissible combinations.

| Linear Component               | CC-A/CC-B | CC-C | Number of columns       |
|--------------------------------|-----------|------|-------------------------|
| UPC-A & EAN-13                 | Yes/Yes   | No   | 4                       |
| EAN-8                          | Yes/Yes   | No   | 3                       |
| UPC-E                          | Yes/Yes   | No   | 2                       |
| UCC/EAN-128                    | Yes/Yes   | Yes  | 4 variable width (1-30) |
| RSS-14                         | Yes/Yes   | No   | 4                       |
| RSS-14 Stacked                 | Yes/Yes   | No   | 2                       |
| RSS-14 Stacked omnidirectional | Yes/Yes   | No   | 2                       |
| RSS Limited                    | Yes/Yes   | No   | 3                       |
| RSS Expanded                   | Yes/Yes   | No   | 4                       |
| RSS Expanded Stack             | Yes/Yes   | No   | 4                       |

### UPC-A Composite Symbol

There is no linkage flag in the UPC-A symbol to indicate the presence of an associated 2D Composite Component. UPC-A linear component may only be linked to four-column CC-A or CC-B components.

### EAN-13 Composite Symbol

There is no linkage flag in the EAN-13 symbol to indicate the presence of an associated 2D Composite Component. EAN-13 linear component may only be linked to four-column CC-A or CC-B components.

### EAN-8 Composite Symbol

There is no linkage flag in the EAN-8 symbol to indicate the presence of an associated 2D Composite Component. EAN-8 linear component may only be linked to three-column CC-A or CC-B components.

### UPC-E Composite Symbol

There is no linkage flag in the UPC-E symbol to indicate the presence of an associated 2D Composite Component. UPC-E linear component may only be linked to two-column CC-A or CC-B components.

### UCC/EAN-128 Composite Symbol

UCC/EAN-128 is a Code 128 with a FNC1 character in the first position after the start character. When UCC/EAN-128 is the linear component of an EAN.UCC Composite Symbol, the bar code shall have a Code Set character (see the table below) as a linkage flag in the last symbol character position before the check character. The printer will add the linkage flag automatically.

| Active Code Set | Link Flag Character,<br>CC-A or CC-B | Link Flag Character,<br>CC-C |
|-----------------|--------------------------------------|------------------------------|
| A               | CODE B                               | CODE C                       |
| B               | CODE C                               | CODE A                       |
| C               | CODE A                               | CODE B                       |

UCC/EAN-128 Composite Symbol may be combined with any of the 2D bar code components created for the EAN.UCC Symbology. The choice between using CC-A/B or CC-C is made by the user. This table describes how an UCC/EAN-128 Composite symbol is built.

| Parameter           | Notes                                                                                                                             |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| Start character     | Added by the printer.                                                                                                             |
| FNC1                | Added by the printer.                                                                                                             |
| Data                | Data is added by the user. If it is necessary to change Code Set the printers add Code- or Shift-characters to change the subset. |
| Link flag character | Added by the printer.                                                                                                             |
| Check digit         | Calculated and added by the printer.                                                                                              |
| Stop character      | Added by the printer.                                                                                                             |

It is possible to choose to filter out spaces, parentheses, and carriage returns for the bar code data and display them in the human readable field. Carriage returns make it possible to display the human readable field in multiple rows.

### UCC/EAN-128 With a 2D Composite Component (CC-A or CC-B)

The length of the data decides if a CC-A or a CC-B will be used. Only a 4 columns wide CC-A or a CC-B is used.

### **UCC/EAN-128 With a 2D Composite Component (CC-C)**

The number of columns of the CC-C is selectable by the user.

The separator pattern is a complement of the linear symbol. The patterns above the UCC/EAN-128 component's quiet zones are light.

The 2D bar code is placed above the separator pattern on the following position: The first interior space module of the CC-C component is aligned with the second module of the linear components star character.

### **RSS Composite Symbol**

When RSS symbols are used as a Composite Component, the encoded value includes a linkage flag indicating the presence of an adjacent 2D Composite Component.

### **RSS-14 Composite Symbol**

RSS-14 linear component may only be linked to four-column CC-A or CC-B components.

### **RSS-14 Truncated Composite Symbol**

RSS-14 Truncated linear component may only be linked to four-column CC-A or CC-B components.

### **RSS-14 Stacked Composite Symbol**

RSS-14 Stacked linear component may only be linked to two-column CCA or CC-B components.

### **RSS-14 Stacked Omnidirectional Composite Symbol**

RSS-14 Stacked Omnidirectional linear component may only be linked to two-column CC-A or CC-B components.

### **RSS Limited Composite Symbol**

RSS-14 Limited linear component may only be linked to three-column CC-A or CC-B components.

### **RSS Expanded Composite Symbol**

RSS-14 Expanded linear component may only be linked to four-column CC-A or CC-B components.

### **RSS Expanded Stacked Composite Symbol**

RSS-14 Expanded Stacked linear component may only be linked to four column CC-A or CC-B components. If linked to a 2D Composite Component the top row of the linear component must contain at least four symbol characters.

## **About CC-ABC Composite Symbols**

CC-A, CC-B and CC-C are the 2D Composite Components in EAN.UCC Composite Symbology. Each is almost the same bar code as the PDF417 (CC-C) and MicroPDF417 (CC-A and CC-B).

**CC-A**

CC-A is a variant of MicroPDF417, and the smallest of the 2D Composite Components. CC-A can encode up to 56 digits and has 3 to 12 rows and 2 to 4 columns. CC-A is a variant of MicroPDF417 with a unique combination of row address patterns.

**CC-B**

CC-B is a variant of MicroPDF417 symbol uniquely identified by the codeword 920 as the first codeword in the symbol. Encoding systems select CC-B when the data to be encoded exceeds the capacity for CC-A. CC-B can encode up to 338 digits and has from 10 to 44 rows and 2 to 4 columns.

**CC-C**

CC-C bar code is the same as a PDF417 except for:

- reserved code word: CC-C has codeword 920 in the second symbol character position (immediately following the Symbol Length Descriptor.)
- high-level data encodation: after the first data codeword 920 the second data codeword is a latch to Byte Compaction mode.
- structure append: not used in CC-C
- reader initialization: codeword 921 does not appear in CC-C.
- no quiet zones are required above and under the bar code.
- reference decode algorithm: an incorrect composite 2D bar code.
- error correction: CC-C meets or exceeds the minimum error correction level recommended for PDF417.
- symbology identifiers: special for CC-C.

Composite Component escape mechanism: After the first Byte mode codeword 901 or 924 in the 2D Composite component, if another code word greater than 899 occurs, the composite “symbol” is logically terminated at that point. The remaining code words are encoded or decoded according to special rules.

**Example**

This example corresponds to the illustration at the start of this section:

```
BARFONT ON
BARSET "EAN128_CCC",1,1,5,100,0,0,0,0,0,1
PRBAR "(01)93812345678901|(10)ABCD123456(410)389
8765432108"
PRINTFEED
```

## EAN8 Composite With CC-A or CC-B

### Separate Commands

| Command    | Ranges and Settings                                                                                                                                                                                               |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:   | “EAN8_CC”                                                                                                                                                                                                         |
| BARRATIO:  | Not applicable.                                                                                                                                                                                                   |
| BARMAG:    | Specifies the x-dimension width in dots of the most narrow element for both the linear bar code and the 2D code. Default is 2. Maximum is 8.                                                                      |
| BARHEIGHT: | Specifies the height of the linear bar code. Default is 100.                                                                                                                                                      |
| BARFONT    | ON: The human readable for the linear bar will be printed under the linear bar code. If BARSET<nexp10> is set to 1, the human readable for the 2D bar code is printed also.<br>OFF: No human readable is printed. |

### BARSET Data for EAN-8 Composite With CC-A or CC-B

| BARSET    | Instruction                                                          | Ranges and Settings                                                                               |
|-----------|----------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| [#<ncon>] | (Optional) Specifies the start parameter.                            |                                                                                                   |
| <sexp>    | Bar code type                                                        | “EAN8_CC”                                                                                         |
| <nexp1-2> | Placeholder                                                          | Value insignificant. 0 is not allowed.                                                            |
| <nexp3>   | Barmag/Enlargement                                                   | See BARMAG.                                                                                       |
| <nexp4>   | Barheight                                                            | See BARHEIGHT.                                                                                    |
| <nexp5>   | Placeholder                                                          | Value insignificant.                                                                              |
| <nexp6>   | Character exclusion in bar code (does not affect human readables)    | Always 1 (default). The bar code and the human readable field will include exactly the same data. |
| <nexp7>   | Height of separator pattern row in dots                              | Value insignificant.                                                                              |
| <nexp8>   | Height of each row in 2D bar code in dots                            | Value must be 1-2 × BARMAG. For example, if BARMAG = 5, 5-10 is OK.                               |
| <nexp9>   | Separator character between data to linear bar code and 2D bar code. | ASCII 2-255 dec. Default is ASCII 124 dec. ASCII 48-57 dec. not allowed.                          |
| <nexp10>  | 2D human readables On/Off                                            | 0: No human readable 2D field.<br>1: Print human readable 2D field.                               |

### Input Data for EAN-8 Composite With CC-A or CC-B

| Parameter             | Ranges and Settings | Notes                |
|-----------------------|---------------------|----------------------|
| Number of characters: | 7                   |                      |
| Check digit:          | 1                   | Added automatically. |
| Digits:               | 0-9                 |                      |
| Uppercase letters:    | No                  |                      |
| Lowercase letters:    | No                  |                      |
| Punctuation marks:    | No                  |                      |
| Start characters:     | No                  |                      |
| Stop characters:      | No                  |                      |

## EAN13 Composite With CC-A or CC-B

### Separate Commands

| Command    | Ranges and Settings                                                                                                                                                                                               |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:   | “EAN13_CC”                                                                                                                                                                                                        |
| BARRATIO:  | Not applicable.                                                                                                                                                                                                   |
| BARMAG:    | Specifies the x-dimension width in dots of the most narrow element for both the linear bar code and the 2D code. Default is 2. Maximum is 8.                                                                      |
| BARHEIGHT: | Specifies the height of the linear bar code. Default is 100.                                                                                                                                                      |
| BARFONT    | ON: The human readable for the linear bar will be printed under the linear bar code. If BARSET<nexp10> is set to 1, the human readable for the 2D bar code is printed also.<br>OFF: No human readable is printed. |

### BARSET Data for EAN13 Composite With CC-A or CC-B

| BARSET    | Instruction                                                          | Notes                                                                                             |
|-----------|----------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| [#<ncon>] | (Optional) Specifies the start parameter                             |                                                                                                   |
| <sexp>    | Bar code type                                                        | “EAN13_CC”                                                                                        |
| <nexp1-2> | Placeholder                                                          | Value insignificant. 0 is not allowed.                                                            |
| <nexp3>   | Barmag/Enlargement                                                   | See BARMAG.                                                                                       |
| <nexp4>   | Barheight                                                            | See BARHEIGHT.                                                                                    |
| <nexp5>   | Placeholder                                                          | Value insignificant.                                                                              |
| <nexp6>   | Character exclusion in bar code (does not affect human readables)    | Always 1 (default). The bar code and the human readable field will include exactly the same data. |
| <nexp7>   | Height of separator pattern row in dots                              | Value must be 1-2 × BARMAG. For example, if BARMAG = 5, 5-10 is OK.                               |
| <nexp8>   | Height of each row in 2D bar code in dots                            | Default: 0 (= 3 × BARMAG)                                                                         |
| <nexp9>   | Separator character between data to linear bar code and 2D bar code. | ASCII 2-255 dec. Default: ASCII 124 dec. ASCII 48-57 dec. not allowed.                            |
| <nexp10>  | 2D human readables On/Off                                            | 0: No human readable 2D field.<br>1: Print human readable 2D field.                               |

### Input Data for EAN13 Composite With CC-A or CC-B

| Parameter             | Ranges and Settings | Notes                |
|-----------------------|---------------------|----------------------|
| Number of characters: | 12                  |                      |
| Check digit:          | 1                   | Added automatically. |
| Digits:               | 0-9                 |                      |
| Uppercase letters:    | No                  |                      |
| Lowercase letters:    | No                  |                      |
| Punctuation marks:    | No                  |                      |
| Start characters:     | No                  |                      |
| Stop characters:      | No                  |                      |

## UPC-E Composite With CC-A or CC-B

### Separate Commands

| Command    | Ranges and Settings                                                                                                                                                                                      |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:   | “UPCE_CC”                                                                                                                                                                                                |
| BARRATIO:  | Not applicable.                                                                                                                                                                                          |
| BARMAG:    | Specifies the x-dimension width in dots of the most narrow element for both the linear bar code and the 2D code. Default is 2. Maximum is 8.                                                             |
| BARHEIGHT: | Specifies the height of the linear bar code. Default is 100.                                                                                                                                             |
| BARFONT    | ON: The human readable for the linear bar will be printed under the linear bar code. If BARSET<nexp10> = 1, the human readable for the 2D bar code is printed also.<br>OFF: No human readable is printed |

### BARSET Data for UPC-E Composite With CC-A or CC-B

| BARSET    | Instruction                                                          | Notes                                                                  |
|-----------|----------------------------------------------------------------------|------------------------------------------------------------------------|
| [#<ncon>] | (Optional) Specifies the start parameter.                            |                                                                        |
| <sexp>    | Bar code type                                                        | “UPCE_CC”                                                              |
| <nexp1-2> | Placeholder                                                          | Value insignificant. 0 is not allowed.                                 |
| <nexp3>   | Barmag/Enlargement                                                   | See BARMAG.                                                            |
| <nexp4>   | Barheight                                                            | See BARHEIGHT.                                                         |
| <nexp5>   | Placeholder                                                          | Value insignificant.                                                   |
| <nexp6>   | Character exclusion in bar code (does not affect human readables)    | Always 1. The bar code and human readable field include the same data. |
| <nexp7>   | Height of separator pattern row in dots                              | Value must be 1-2 × BARMAG. For example, if BARMAG = 5, 5-10 is OK.    |
| <nexp8>   | Height of each row in 2D bar code in dots                            | Default: 0 (= 3 × BARMAG)                                              |
| <nexp9>   | Separator character between data to linear bar code and 2D bar code. | ASCII 2-255 dec. Default: ASCII 124 dec. ASCII 48-57 dec. not allowed. |
| <nexp10>  | 2D human readables On/Off                                            | 0: No human readable 2D field.<br>1: Print human readable 2D field.    |

### Input Data for UPC-E Composite With CC-A or CC-B

| Parameter             | Ranges and Settings | Notes                |
|-----------------------|---------------------|----------------------|
| Number of characters: | 6                   |                      |
| Check digit:          | 1                   | Added automatically. |
| Digits:               | 0-9                 |                      |
| Uppercase letters:    | No                  |                      |
| Lowercase letters:    | No                  |                      |
| Punctuation marks:    | No                  |                      |
| Start characters:     | No                  |                      |
| Stop characters:      | No                  |                      |

## UPC-A Composite With CC-A or CC-B

### Separate Commands

| Command    | Ranges and Settings                                                                                                                                                                                               |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:   | “UPCA_CC”                                                                                                                                                                                                         |
| BARRATIO:  | Not applicable.                                                                                                                                                                                                   |
| BARMAG:    | Specifies the x-dimension width in dots of the most narrow element for both the linear bar code and the 2D code. Default is 2. Maximum is 8.                                                                      |
| BARHEIGHT: | Specifies the height of the linear bar code. Default is 100.                                                                                                                                                      |
| BARFONT    | ON: The human readable for the linear bar will be printed under the linear bar code. If BARSET<nexp10> is set to 1, the human readable for the 2D bar code is printed also.<br>OFF: No human readable is printed. |

### BARSET Data for UPC-A Composite With CC-A or CC-B

| BARSET    | Instruction                                                          | Notes                                                                                             |
|-----------|----------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| [#<ncon>] | (Optional) Specifies the start parameter.                            |                                                                                                   |
| <sexp>    | Bar code type                                                        | “UPCA_CC”                                                                                         |
| <nexp1-2> | Placeholder                                                          | Value insignificant. 0 is not allowed.                                                            |
| <nexp3>   | Barmag/Enlargement                                                   | See BARMAG.                                                                                       |
| <nexp4>   | Barheight                                                            | See BARHEIGHT.                                                                                    |
| <nexp5>   | Placeholder                                                          | Value insignificant.                                                                              |
| <nexp6>   | Character exclusion in bar code (does not affect human readables)    | Always 1 (default). The bar code and the human readable field will include exactly the same data. |
| <nexp7>   | Height of separator pattern row in dots                              | Value must be 1-2 × BARMAG. For example, if BARMAG = 5, 5-10 is OK.                               |
| <nexp8>   | Height of each row in 2D bar code in dots                            | Default: 0 (= 3 × BARMAG)                                                                         |
| <nexp9>   | Separator character between data to linear bar code and 2D bar code. | ASCII 2-255 dec. Default: ASCII 124 dec. ASCII 48-57 dec. not allowed.                            |
| <nexp10>  | 2D human readables On/Off                                            | 0: No human readable 2D field.<br>1: Print human readable 2D field.                               |

### Input Data for UPC-A Composite With CC-A or CC-B

| Parameter          | Ranges and Settings | Notes                |
|--------------------|---------------------|----------------------|
| No. of characters: | 11                  |                      |
| Check digit:       | 1                   | Added automatically. |
| Digits:            | 0-9                 |                      |
| Uppercase letters: | No                  |                      |
| Lowercase letters: | No                  |                      |
| Punctuation marks: | No                  |                      |
| Start characters:  | No                  |                      |
| Stop characters:   | No                  |                      |

## EAN.UCC 128 Composite With CC-C

### Separate Commands

| Command    | Ranges and Settings                                                                                                                                                                                               |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:   | “EAN128_CCC”                                                                                                                                                                                                      |
| BARRATIO:  | Not applicable.                                                                                                                                                                                                   |
| BARMAG:    | Specifies the x-dimension width in dots of the most narrow element for both the linear bar code and the 2D code. Default is 2.                                                                                    |
| BARHEIGHT: | Specifies the height of the linear bar code. Default is 100.                                                                                                                                                      |
| BARFONT    | ON: The human readable for the linear bar will be printed under the linear bar code. If BARSET<nexp10> is set to 1, the human readable for the 2D bar code is printed also.<br>OFF: No human readable is printed. |

### BARSET Data for EAN.UCC 128 Composite With CC-C

| BARSET    | Instruction                                                          | Notes                                                                                                                                                                                                                                                                                            |
|-----------|----------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [#<ncon>] | (Optional) Specifies the start parameter.                            |                                                                                                                                                                                                                                                                                                  |
| <sexp>    | Bar code type                                                        | “EAN128_CCC”                                                                                                                                                                                                                                                                                     |
| <nexp1-2> | Placeholder                                                          | Value insignificant. 0 is not allowed.                                                                                                                                                                                                                                                           |
| <nexp3>   | Barmag/Enlargement                                                   | See BARMAG.                                                                                                                                                                                                                                                                                      |
| <nexp4>   | Barheight                                                            | See BARHEIGHT.                                                                                                                                                                                                                                                                                   |
| <nexp5>   | Number of columns/row in 2D bar code                                 | 0 (default) = select the largest number of columns for the 2D code to fit within the linear bar code including its quiet zones.                                                                                                                                                                  |
| <nexp6>   | Character exclusion in bar code (does not affect human readables)    | 0: Bar code will not include space, comma (,), or CR ([CHR\$(13)]) characters, but they are printed in the human readable field.<br>Carriage returns make it possible to print human readable in multiple rows.<br>1 (default): Bar code and human readable field include exactly the same data. |
| <nexp7>   | Height of separator pattern row in dots                              | Value must be 1-2 × BARMAG. For example, if BARMAG = 5, 5-10 is OK.                                                                                                                                                                                                                              |
| <nexp8>   | Height of each row in 2D bar code in dots                            | Default: 0 (= 3 × BARMAG)                                                                                                                                                                                                                                                                        |
| <nexp9>   | Separator character between data to linear bar code and 2D bar code. | ASCII 2-255 dec. Default: ASCII 124 dec. ASCII 48-57 dec. not allowed.                                                                                                                                                                                                                           |
| <nexp10>  | 2D human readables On/Off                                            | 0: No human readable 2D field.<br>1: Print human readable 2D field.                                                                                                                                                                                                                              |

***Input Data for EAN.UCC 128 Composite With CC-C***

| Parameter             | Ranges and Settings             | Notes                               |
|-----------------------|---------------------------------|-------------------------------------|
| Number of characters: | Unlimited                       |                                     |
| Check digit:          | Trailing symbol check character | Added automatically.                |
| Input characters      | ASCII 0-127                     | According to Roman 8 character set. |
| Start characters:     | See note 2.                     |                                     |
| Code characters       | See note 1 & 2                  |                                     |
| Shift characters:     | Always                          |                                     |
| Stop characters:      | Always                          | Added automatically.                |

**Remarks**

EAN 128 is identical to Code 128 with the exception that the initial FNC1 function character is generated automatically.



**Note 1:** Code characters and shift characters require either an 8-bit communication protocol, remapping to an ASCII value between 0-127 dec., or the use of an CHR\$ function.



**Note 2:** EAN 128 has automatic selection of start character and character subset (that is, selects optimal start character and handles shift and changes of subset depending on the content of the input data), whereas EAN 128A, EAN 128B, and EAN 128C selects subset A, B, and C respectively. The last character in the bar code name signifies both the start character and the chosen subset.

The selected subset can be changed anywhere in the input string, either for a single character using a Shift character (not for Subset C), or for the remainder of the input string using a Code character (all subsets).

The Shift and Code characters consist of a combination of two characters:

- Two left-pointing double angle quotation marks («) specify a Shift character.  
Shift character: «« (« = ASCII 171 dec.)
- One left-pointing double angle quotation mark («) specifies a Code character. It should be followed by an uppercase letter that specifies the subset:  
Code character: « + A|B|C (« = ASCII 171 dec.)

## EAN.UCC 128 Composite With CC-A or CC-B

### Separate Commands

| Command    | Ranges and Settings                                                                                                                                                                                              |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:   | “EAN128_CCAB”                                                                                                                                                                                                    |
| BARRATIO:  | Not applicable.                                                                                                                                                                                                  |
| BARMAG:    | Specifies the x-dimension width in dots of the most narrow element for both the linear bar code and the 2D code. Default is 2.                                                                                   |
| BARHEIGHT: | Specifies the height of the linear bar code1. Default is 100.                                                                                                                                                    |
| BARFONT    | ON: The human readable for the linear bar will be printed under the linear bar code. If BARSET<exp10> is set to 1, the human readable for the 2D bar code is printed also.<br>OFF: No human readable is printed. |

### BARSET Data for EAN.UCC 128 Composite With CC-A or CC-B

| BARSET    | Description                                                          | Ranges and Settings                                                                                                                                                                                                                                                                                        |
|-----------|----------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [#<ncon>] | (Optional) Specifies the start parameter.                            |                                                                                                                                                                                                                                                                                                            |
| <sexp>    | Bar code type                                                        | “EAN128_CCAB”                                                                                                                                                                                                                                                                                              |
| <nexp1-2> | Placeholder                                                          | Value insignificant. 0 is not allowed.                                                                                                                                                                                                                                                                     |
| <nexp3>   | Barmag/Enlargement                                                   | See BARMAG.                                                                                                                                                                                                                                                                                                |
| <nexp4>   | Barheight                                                            | See BARHEIGHT.                                                                                                                                                                                                                                                                                             |
| <nexp5>   | Placeholder                                                          | Value insignificant.                                                                                                                                                                                                                                                                                       |
| <nexp6>   | Character exclusion in bar code (does not affect human readables)    | 0: Bar code will not include space, comma (,), or CR ([CHR\$(13)]) characters, but they are printed in the human readable field. Carriage returns make it possible to print human readable in multiple rows.<br>1 (default): The bar code and the human readable field will include exactly the same data. |
| <nexp7>   | Height of separator pattern row in dots                              | Value must be 1-2 × BARMAG. For example, if BARMAG = 5, 5-10 is OK.                                                                                                                                                                                                                                        |
| <nexp8>   | Height of each row in 2D bar code in dots                            | Default: 0 (= 3 × BARMAG)                                                                                                                                                                                                                                                                                  |
| <nexp9>   | Separator character between data to linear bar code and 2D bar code. | ASCII 2-255 dec. Default is ASCII 124 dec. ASCII 48-57 dec. not allowed.                                                                                                                                                                                                                                   |
| <nexp10>  | 2D human readables On/Off                                            | 0: No human readable 2D field.<br>1: Print human readable 2D field.                                                                                                                                                                                                                                        |

***Input Data for EAN.UCC 128 Composite With CC-A or CC-B***

| Parameter             | Ranges and Settings             | Notes                               |
|-----------------------|---------------------------------|-------------------------------------|
| Number of characters: | Unlimited                       |                                     |
| Check digit:          | Trailing symbol check character | Added automatically.                |
| Input characters      | ASCII 0-127                     | According to Roman 8 character set. |
| Start characters:     | See note 2.                     |                                     |
| Code characters       | See notes 1 and 2.              |                                     |
| Shift characters:     | See notes 1 and 2.              |                                     |
| Stop characters:      | Always                          | Added automatically.                |

**Remarks**

EAN128 is identical to Code 128, except that the initial FNC1 function character is generated automatically.



**Note 1:** Code characters and shift characters require either an 8-bit communication protocol, remapping to an ASCII value between 0-127 dec., or the use of an CHR\$ function.



**Note 2:** EAN 128 has automatic selection of start character and character subset (that is, selects optimal start character and handles shift and changes of subset depending on the content of the input data), whereas EAN 128A, EAN 128B, and EAN 128C selects subset A, B, and C respectively. The last character in the bar code name signifies both the start character and the chosen subset.

The selected subset can be changed anywhere in the input string, either for a single character using a Shift character (not for Subset C), or for the remainder of the input string using a Code character (all subsets). The Shift and Code characters consist of a combination of two characters:

- Two left-pointing double angle quotation marks («) specify a Shift character.  
Shift character: «« (« = ASCII 171 dec.)
- One left-pointing double angle quotation mark («) specifies a Code character. It should be followed by an uppercase letter that specifies the subset:  
Code character: « + A|B|C (« = ASCII 171 dec.)

## RSS-14 (Composite)

### *Separate Commands*

| Command    | Ranges and Settings                                                                                                            |
|------------|--------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:   | “RSS14”                                                                                                                        |
| BARRATIO:  | Not applicable.                                                                                                                |
| BARMAG:    | Specifies the x-dimension width in dots of the most narrow element for both the linear bar code and the 2D code. Default is 2. |
| BARHEIGHT: | Specifies the height of the linear bar code. Default is 100. See the Remarks for more information.                             |
| BARFONT    | No restrictions.                                                                                                               |

### ***BARSET Data for RSS-14 Composite***

| BARSET    | Description                                                                | Notes                                                                                                                                                                                                                                                                                                                               |
|-----------|----------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [#<ncon>] | (Optional) Specifies the start parameter in the syntax.                    |                                                                                                                                                                                                                                                                                                                                     |
| <sexp>    | Bar code type                                                              | “RSS14”                                                                                                                                                                                                                                                                                                                             |
| <nexp1-2> | Placeholder                                                                | Value insignificant. 0 is not allowed.                                                                                                                                                                                                                                                                                              |
| <nexp3>   | Barmag/Enlargement                                                         | See BARMAG.                                                                                                                                                                                                                                                                                                                         |
| <nexp4>   | Barheight                                                                  | See BARHEIGHT.                                                                                                                                                                                                                                                                                                                      |
| <nexp5>   | Placeholder                                                                | Value insignificant.                                                                                                                                                                                                                                                                                                                |
| <nexp6>   | Character exclusion in bar code<br>(does not affect human<br>readables)    | 0: Bar code will not include space,<br>comma (,), or CR ([CHR\$(13)])<br>characters, but they are printed in the<br>human readable field. Carriage returns<br>make it possible to print human<br>readable in multiple rows.<br><br>1 (default): The bar code and the<br>human readable field will include<br>exactly the same data. |
| <nexp7>   | Height of separator pattern row<br>in dots                                 | Value must be 1-2 × BARMAG. For<br>example, if BARMAG = 5, 5-10 is OK.                                                                                                                                                                                                                                                              |
| <nexp8>   | Height of each row in 2D bar<br>code in dots                               | Default: 0 (= 3 × BARMAG)                                                                                                                                                                                                                                                                                                           |
| <nexp9>   | Separator character between<br>data to linear bar code and 2D<br>bar code. | ASCII 2-255 dec. Default: ASCII 124<br>dec. ASCII 48-57 dec. not allowed.                                                                                                                                                                                                                                                           |
| <nexp10>  | 2D human readables On/Off                                                  | 0: No human readable 2D field.<br>1: Print human readable 2D field.                                                                                                                                                                                                                                                                 |

***Input Data***

| Parameter        | Ranges and Settings | Notes                                                                                                      |
|------------------|---------------------|------------------------------------------------------------------------------------------------------------|
| Input characters | 13 digits           | If less than 13 digits are entered, leading zeros are automatically added so the string is 13 digits long. |

**Remarks**

There are restrictions in the standard for the minimum size for each RSS bar code, even if it is possible to print an RSS bar code in any height. The height should relate to the magnification. RSS Stacked differs because the bar code rows do not have the same height. BARHEIGHT or BARSET<exp4> specifies the height of the lower row and height of the upper row is automatically calculated from the height of the lower row.

For RSS-14, the width is 96X and the minimum height is 33X, where X is the width of the most narrow element as specified by BARMAG or BARSET<exp3>.

**RSS-14 Truncated (Composite)*****Separate Commands***

| Command    | Ranges and Settings                                                                                                            |
|------------|--------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:   | "RSS14T"                                                                                                                       |
| BARRATIO:  | Not applicable.                                                                                                                |
| BARMAG:    | Specifies the x-dimension width in dots of the most narrow element for both the linear bar code and the 2D code. Default is 2. |
| BARHEIGHT: | Specifies the height of the linear bar code. Default is 100. See the Remarks for more information.                             |
| BARFONT    | No restrictions.                                                                                                               |

**BARSET Data for RSS-14T (Composite)**

| <b>BARSET</b> | <b>Description</b>                                                   | <b>Ranges and Settings</b>                                                                                                                                                                                                                                                                                                                                                                   |
|---------------|----------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [#<ncon>]     | (Optional) Specifies the start parameter in the syntax.              |                                                                                                                                                                                                                                                                                                                                                                                              |
| <sexp>        | Bar code type                                                        | “RSS14T”                                                                                                                                                                                                                                                                                                                                                                                     |
| <nexp1-2>     | Placeholder                                                          | Value insignificant. 0 is not allowed.                                                                                                                                                                                                                                                                                                                                                       |
| <nexp3>       | Barmag/Enlargement                                                   | See BARMAG.                                                                                                                                                                                                                                                                                                                                                                                  |
| <nexp4>       | Barheight                                                            | See BARHEIGHT.                                                                                                                                                                                                                                                                                                                                                                               |
| <nexp5>       | Placeholder                                                          | Value insignificant.                                                                                                                                                                                                                                                                                                                                                                         |
| <nexp6>       | Character exclusion in bar code (does not affect human readables)    | 0: Bar code will not include space, comma (,), or CR ([CHR\$(13)]) characters, but they are printed in the human readable field.<br>Carriage returns make it possible to print human readable in multiple rows.<br><br>1 (default): The bar code and the human readable field will include exactly the same data.<br><br>Value must be 1-2 × BARMAG. For example, if BARMAG = 5, 5-10 is OK. |
| <nexp7>       | Height of separator pattern row in dots                              | Default: 0 (= 3 × BARMAG)                                                                                                                                                                                                                                                                                                                                                                    |
| <nexp8>       | Height of each row in 2D bar code in dots                            | ASCII 2-255 dec. Default: ASCII 124 dec.<br>ASCII 48-57 dec. not allowed.                                                                                                                                                                                                                                                                                                                    |
| <nexp9>       | Separator character between data to linear bar code and 2D bar code. |                                                                                                                                                                                                                                                                                                                                                                                              |
| <nexp10>      | 2D human readables On/<br>Off                                        | 0: No human readable 2D field.<br>1: Print human readable 2D field.                                                                                                                                                                                                                                                                                                                          |

**Input Data for RSS-14T (Composite)**

| <b>Parameter</b> | <b>Ranges and Settings</b> | <b>Notes</b>                                                                                               |
|------------------|----------------------------|------------------------------------------------------------------------------------------------------------|
| Input characters | 13 digits                  | If less than 13 digits are entered, leading zeros are automatically added so the string is 13 digits long. |

**RSS-14 Stacked (Composite)****Separate Commands**

| <b>Command</b> | <b>Ranges and Settings</b>                                                                                                     |
|----------------|--------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:       | “RSS14S”                                                                                                                       |
| BARRATIO:      | Not applicable.                                                                                                                |
| BARMAG:        | Specifies the x-dimension width in dots of the most narrow element for both the linear bar code and the 2D code. Default is 2. |
| BARHEIGHT:     | Specifies the height of the linear bar code. Default is 100. See the Remarks for more information.                             |
| BARFONT        | No restrictions.                                                                                                               |

**BARSET Data for RSS-14S (Composite)**

| <b>BARSET</b> | <b>Description</b>                                                   | <b>Ranges and Settings</b>                                                                                                                                                                                                                                                                            |
|---------------|----------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [#<ncon>]     | (Optional) Specifies the start parameter in the syntax.              |                                                                                                                                                                                                                                                                                                       |
| <sexp>        | Bar code type                                                        | “RSS14S”                                                                                                                                                                                                                                                                                              |
| <nexp1-2>     | Placeholder                                                          | Value insignificant. 0 is not allowed.                                                                                                                                                                                                                                                                |
| <nexp3>       | Barmag/Enlargement                                                   | See BARMAG.                                                                                                                                                                                                                                                                                           |
| <nexp4>       | Barheight                                                            | See BARHEIGHT.                                                                                                                                                                                                                                                                                        |
| <nexp5>       | Placeholder                                                          | Value insignificant.                                                                                                                                                                                                                                                                                  |
| <nexp6>       | Character exclusion in bar code (does not affect human readables)    | 0: Bar code will not include space, comma (,), or CR ([CHR\$(13)]) characters, but they are printed in the human readable field. Carriage returns make it possible to print human readable in multiple rows.<br>1 (default): The bar code and the human readable field include exactly the same data. |
| <nexp7>       | Height of separator pattern row in dots                              | Value must be 1-2 × BARMAG. For example, if BARMAG = 5, 5-10 is OK.                                                                                                                                                                                                                                   |
| <nexp8>       | Height of each row in 2D bar code in dots                            | Default: 0 (= 3 × BARMAG)                                                                                                                                                                                                                                                                             |
| <nexp9>       | Separator character between data to linear bar code and 2D bar code. | ASCII 2-255 dec. Default: ASCII 124 dec. ASCII 48-57 dec. not allowed.                                                                                                                                                                                                                                |
| <nexp10>      | 2D human readables On/Off                                            | 0: No human readable 2D field.<br>1: Print human readable 2D field.                                                                                                                                                                                                                                   |

**Input Data for RSS-14S (Composite)**

| <b>Parameter</b> | <b>Ranges and Settings</b> | <b>Notes</b>                                                                                               |
|------------------|----------------------------|------------------------------------------------------------------------------------------------------------|
| Input characters | 13 digits                  | If less than 13 digits are entered, leading zeros are automatically added so the string is 13 digits long. |

**Remarks**

There are restrictions in the standard for the minimum size for each RSS bar code, even if it is possible to print an RSS bar code in any height. The height should relate to the magnification. RSS Stacked differs because the bar code rows do not have the same height. BARHEIGHT or BARSET<nexp4> specifies the height of the lower row and height of the upper row is automatically calculated from the height of the lower row.

For RSS-14S, the width is 50X and the minimum height is 13X (upper 5X + lower 7X + separator 1X min.), where X is the width of the most narrow element as specified by BARMAG or BARSET<nexp3>.

**Example**

Example of an RSS14S bar code with the minimum recommended height selected and with these characteristics:

|                                        |                         |
|----------------------------------------|-------------------------|
| Place holder (nexp1):                  | 1                       |
| Place holder (nexp2):                  | 1                       |
| Most narrow element width in dots:     | 3                       |
| Height in dots:                        | 21                      |
| Place holder (nexp5):                  | 1                       |
| Character exclusion (nexp6):           | 1                       |
| Separator pattern row height (nexp7):  | 4                       |
| Height of rows in 2D bar code (nexp8): | 4                       |
| Separator character (nexp9):           | 124                     |
| 2D human readables (nexp10):           | 1                       |
| Data:                                  | 1234567890123 987654321 |

```
BARSET "RSS14S",1,1,3,21,1,1,4,4,124,1
PRBAR "1234567890123|987654321"
```

**RSS-14 Stacked Omnidirectional (Composite)****Separate Commands**

| Command    | Ranges and Settings                                                                                                            |
|------------|--------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:   | “RSS14SO”                                                                                                                      |
| BARRATIO:  | Not applicable.                                                                                                                |
| BARMAG:    | Specifies the x-dimension width in dots of the most narrow element for both the linear bar code and the 2D code. Default is 2. |
| BARHEIGHT: | Specifies the height of the linear bar code. Default is 100. See the Remarks for more information.                             |
| BARFONT    | No restrictions.                                                                                                               |

**BARSET Data for RSS-14SO (Composite)**

| BARSET    | Description                                             | Notes                                  |
|-----------|---------------------------------------------------------|----------------------------------------|
| [#<ncon>] | (Optional) Specifies the start parameter in the syntax. |                                        |
| <sexp>    | Bar code type                                           | “RSS14SO”                              |
| <nexp1-2> | Placeholder                                             | Value insignificant. 0 is not allowed. |
| <nexp3>   | Barmag/Enlargement                                      | See BARMAG.                            |
| <nexp4>   | Barheight                                               | See BARHEIGHT.                         |
| <nexp5>   | Placeholder                                             | Value insignificant.                   |

| BARSET   | Description                                                          | Notes                                                                                                                                                                                                                                                                                                      |
|----------|----------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <nexp6>  | Character exclusion in bar code (does not affect human readables)    | 0: Bar code will not include space, comma (,), or CR ([CHR\$(13)]) characters, but they are printed in the human readable field. Carriage returns make it possible to print human readable in multiple rows.<br>1 (default): The bar code and the human readable field will include exactly the same data. |
| <nexp7>  | Height of separator pattern row in dots                              | Value must be 1-2 × BARMAG. For example, if BARMAG = 5, 5-10 is OK.                                                                                                                                                                                                                                        |
| <nexp8>  | Height of each row in 2D bar code in dots                            | Default: 0 (= 3 × BARMAG)                                                                                                                                                                                                                                                                                  |
| <nexp9>  | Separator character between data to linear bar code and 2D bar code. | ASCII 2-255 dec. Default: ASCII 124 dec. ASCII 48-57 dec. not allowed.                                                                                                                                                                                                                                     |
| <nexp10> | 2D human readables On/Off                                            | 0: No human readable 2D field.<br>1: Print human readable 2D field.                                                                                                                                                                                                                                        |

### ***Input Data for RSS-14SO (Composite)***

| Parameter        | Ranges and Settings | Notes                                                                                                      |
|------------------|---------------------|------------------------------------------------------------------------------------------------------------|
| Input characters | 13 digits           | If less than 13 digits are entered, leading zeros are automatically added so the string is 13 digits long. |

### **Remarks**

There are restrictions in the standard for the minimum size for each RSS bar code, even if it is possible to print an RSS bar code in any height. The height should relate to the magnification. RSS Stacked differs because the bar code rows do not have the same height. BARHEIGHT or BARSET<nexp4> specifies the height of the lower row and height of the upper row is automatically calculated from the height of the lower row.

For RSS-14SO, the width is 50X and the minimum height is 69X (upper 33X + lower 33X + separator 3\*1X min.), where X is the width of the most narrow element as specified by BARMAG or BARSET<nexp3>.

## **RSS-14 Limited (Composite)**

### ***Separate Commands***

| Command    | Ranges and Settings                                                                                                            |
|------------|--------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:   | “RSS14L”                                                                                                                       |
| BARRATIO:  | Not applicable.                                                                                                                |
| BARMAG:    | Specifies the x-dimension width in dots of the most narrow element for both the linear bar code and the 2D code. Default is 2. |
| BARHEIGHT: | Specifies the height of the linear bar code. Default is 100. See the Remarks for more information.                             |
| BARFONT    | No restrictions.                                                                                                               |

***BARSET Data for RSS-14L (Composite)***

| <b>BARSET</b> | <b>Description</b>                                                   | <b>Ranges and Settings</b>                                                                                                                                                                                                                                                                                 |
|---------------|----------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [#<ncon>]     | (Optional) Specifies the start parameter in the syntax.              |                                                                                                                                                                                                                                                                                                            |
| <sexp>        | Bar code type                                                        | “RSS14SL”                                                                                                                                                                                                                                                                                                  |
| <nexp1-2>     | Placeholder                                                          | Value insignificant. 0 is not allowed.                                                                                                                                                                                                                                                                     |
| <nexp3>       | Barmag/Enlargement                                                   | See BARMAG.                                                                                                                                                                                                                                                                                                |
| <nexp4>       | Barheight                                                            | See BARHEIGHT.                                                                                                                                                                                                                                                                                             |
| <nexp5>       | Placeholder                                                          | Value insignificant.                                                                                                                                                                                                                                                                                       |
| <nexp6>       | Character exclusion in bar code (does not affect human readables)    | 0: Bar code will not include space, comma (,), or CR ([CHR\$(13)]) characters, but they are printed in the human readable field. Carriage returns make it possible to print human readable in multiple rows.<br>1 (default): The bar code and the human readable field will include exactly the same data. |
| <nexp7>       | Height of separator pattern row in dots                              | Value must be 1-2 × BARMAG. For example, if BARMAG = 5, 5-10 is OK.                                                                                                                                                                                                                                        |
| <nexp8>       | Height of each row in 2D bar code in dots                            | Default: 0 (= 3 × BARMAG)                                                                                                                                                                                                                                                                                  |
| <nexp9>       | Separator character between data to linear bar code and 2D bar code. | ASCII 2-255 dec. Default: ASCII 124 dec. ASCII 48-57 dec. not allowed.                                                                                                                                                                                                                                     |
| <nexp10>      | 2D human readables On/Off                                            | 0: No human readable 2D field.<br>1: Print human readable 2D field.                                                                                                                                                                                                                                        |

***Input Data for RSS-14L (Composite)***

| <b>Parameter</b> | <b>Ranges and Settings</b> | <b>Notes</b>                                                                                               |
|------------------|----------------------------|------------------------------------------------------------------------------------------------------------|
| Input characters | 13 digits                  | If less than 13 digits are entered, leading zeros are automatically added so the string is 13 digits long. |

**Remarks**

There are restrictions in the standard for the minimum size for each RSS bar code, even if it is possible to print an RSS bar code in any height. The height should relate to the magnification. RSS Stacked differs because the bar code rows do not have the same height. BARHEIGHT or BARSET<nexp4> specifies the height of the lower row and height of the upper row is automatically calculated from the height of the lower row.

For RSS-14L, the width is 71X and the minimum height is 10X, where X is the width of the most narrow element as specified by BARMAG or BARSET<nexp3>.

## RSS-14 Expanded (Composite)

### Separate Commands

| Command    | Ranges and Settings                                                                                                            |
|------------|--------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:   | “RSS14E”                                                                                                                       |
| BARRATIO:  | Not applicable.                                                                                                                |
| BARMAG:    | Specifies the x-dimension width in dots of the most narrow element for both the linear bar code and the 2D code. Default is 2. |
| BARHEIGHT: | Specifies the height of the linear bar code. Default is 100. See the Remarks for more information.                             |
| BARFONT    | No restrictions.                                                                                                               |

### BARSET Data for RSS-14E (Composite)

| BARSET    | Description                                                                | Ranges and Settings                                                                                                                                                                                                                                                                                                                 |
|-----------|----------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [#<ncon>] | (Optional) Specifies the start parameter in the syntax.                    |                                                                                                                                                                                                                                                                                                                                     |
| <sexp>    | Bar code type                                                              | “RSS14SE”                                                                                                                                                                                                                                                                                                                           |
| <nexp1-2> | Placeholder                                                                | Value insignificant. 0 is not allowed.                                                                                                                                                                                                                                                                                              |
| <nexp3>   | Barmag/Enlargement                                                         | See BARMAG.                                                                                                                                                                                                                                                                                                                         |
| <nexp4>   | Barheight                                                                  | See BARHEIGHT.                                                                                                                                                                                                                                                                                                                      |
| <nexp5>   | Placeholder                                                                | Value insignificant.                                                                                                                                                                                                                                                                                                                |
| <nexp6>   | Character exclusion in bar code<br>(does not affect human<br>readables)    | 0: Bar code will not include space,<br>comma (,), or CR ([CHR\$(13)])<br>characters, but they are printed in the<br>human readable field. Carriage returns<br>make it possible to print human<br>readable in multiple rows.<br><br>1 (default): The bar code and the<br>human readable field will include<br>exactly the same data. |
| <nexp7>   | Height of separator pattern row<br>in dots                                 | Value must be 1-2 × BARMAG. For<br>example, if BARMAG = 5, 5-10 is OK.                                                                                                                                                                                                                                                              |
| <nexp8>   | Height of each row in 2D bar<br>code in dots                               | Default: 0 (= 3 × BARMAG)                                                                                                                                                                                                                                                                                                           |
| <nexp9>   | Separator character between data<br>to linear bar code and 2D bar<br>code. | ASCII 2-255 dec. Default: ASCII 124<br>dec. ASCII 48-57 dec. not allowed.                                                                                                                                                                                                                                                           |
| <nexp10>  | 2D human readables On/Off                                                  | 0 = No human readable 2D field.<br>1 = Print human readable 2D field.                                                                                                                                                                                                                                                               |

### Remarks

There are restrictions in the standard for the minimum size for each RSS bar code, even if it is possible to print an RSS bar code in any height. The height should relate to the magnification. RSS Stacked differs because the bar code rows do not have the same height. BARHEIGHT or BARSET<nexp4> specifies the height of the lower row and height of the upper row is automatically calculated from the height of the lower row.

For RSS-14E, the width is depending on input and the minimum height is 33X, where X is the width of the most narrow element as specified by BARMAG or BARSET<nexp3>.

## RSS-14 Expanded Stacked (Composite)

### Separate Commands

| Command    | Ranges and Settings                                                                                                            |
|------------|--------------------------------------------------------------------------------------------------------------------------------|
| BARTYPE:   | "RSS14ES"                                                                                                                      |
| BARRATIO:  | Not applicable.                                                                                                                |
| BARMAG:    | Specifies the x-dimension width in dots of the most narrow element for both the linear bar code and the 2D code. Default is 2. |
| BARHEIGHT: | Specifies the height of the linear bar code. Default is 100. See the Remarks for more information.                             |
| BARFONT    | No restrictions.                                                                                                               |

### BARSET Data for RSS-14ES

| BARSET    | Description                                                          | Ranges and Settings                                                                                                                                                                                                                                                                         |
|-----------|----------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [#<ncon>] | (Optional) Specifies the start parameter in the syntax.              |                                                                                                                                                                                                                                                                                             |
| <nexp>    | Bar code type                                                        | "RSS14ES"                                                                                                                                                                                                                                                                                   |
| <nexp1-2> | Placeholder                                                          | Value insignificant. 0 is not allowed.                                                                                                                                                                                                                                                      |
| <nexp3>   | Barmag/Enlargement                                                   | See BARMAG.                                                                                                                                                                                                                                                                                 |
| <nexp4>   | Barheight                                                            | See BARHEIGHT.                                                                                                                                                                                                                                                                              |
| <nexp5>   | Segments per row                                                     | 2-22 (default 2). Multiples of 2 only.                                                                                                                                                                                                                                                      |
| <nexp6>   | Placeholder                                                          | Value insignificant.                                                                                                                                                                                                                                                                        |
| <nexp7>   | Height of separator row                                              | The height is the same for the pattern row between the 2D bar code and the linear bar code as for the pattern rows between the linear rows. If too low a height is entered, the height will be changed to the smallest legal height for the selected magnification. Minimum 1X, maximum 2X. |
| <nexp8>   | Height of each row in 2D bar code in dots                            | Default: 0 (= 3 × BARMAG)                                                                                                                                                                                                                                                                   |
| <nexp9>   | Separator character between data to linear bar code and 2D bar code. | ASCII 2-255 dec. Default: ASCII 124 dec. ASCII 48-57 dec. not allowed.                                                                                                                                                                                                                      |
| <nexp10>  | 2D human readables On/Off                                            | 0 = No human readable 2D field.<br>1 = Print human readable 2D field.                                                                                                                                                                                                                       |

### Input Data for RSS-14ES (Composite)

| Parameter        | Ranges and Settings                           | Notes                                                                                           |
|------------------|-----------------------------------------------|-------------------------------------------------------------------------------------------------|
| Input characters | Max. 71 numeric or 41 alphanumeric characters | Allowed characters: 0-9 A-Z a-z ! " % & ' ( ) * + , - . / : ; < = > ? _ space FNC1 [CHR\$(128)] |

## Remarks

There are restrictions in the standard for the minimum size for each RSS bar code, even if it is possible to print an RSS bar code in any height. The height should relate to the magnification. RSS Stacked differs because the bar code rows do not have the same height. BARHEIGHT or BARSET<*nexp4*> specifies the height of the lower row and height of the upper row is automatically calculated from the height of the lower row.

For RSS-14ES, the width is depending on input and the minimum height is 34X per row + 3\*1X per separator, where X is the width of the most narrow element as specified by BARMAG or BARSET<*nexp3*>.

RSS-14ES can be used for intelligent encoding of the input data. They can be created with different encoding methods and compressed data fields. To understand how to create intelligent bar codes with RSS-14ES, see the most current AIM specifications.

## Example

Example of RSS14 Expanded Stacked for a variable weight item (0,001 kilogram increments) with recommended minimum height selected:

|                                                 |                                       |
|-------------------------------------------------|---------------------------------------|
| Start parameter:                                | #4                                    |
| Most narrow element width in dots               | 2                                     |
| Height in dots:                                 | 68                                    |
| Segments per row ( <i>nexp5</i> ):              | 2                                     |
| Character exclusion ( <i>nexp6</i> ):           | 1                                     |
| Separator pattern row height ( <i>nexp7</i> ):  | 4                                     |
| Height of rows in 2D bar code ( <i>nexp8</i> ): | 4                                     |
| Separator character ( <i>nexp9</i> ):           | 124                                   |
| 2D human readables ( <i>nexp10</i> ):           | 1                                     |
| Data:                                           | 01900123456789003103001750 9876543210 |

|                                         |                                                                              |
|-----------------------------------------|------------------------------------------------------------------------------|
| Explanation of data:                    | 01 9001234567890 0 3103 001750                                               |
| Application Identifier (AI):            | 01                                                                           |
| AI 01 item ID:                          | 9001234567890<br>(In this method, the first digit must be 9.)                |
| Digit:                                  | 0. The value is insignificant but a digit must be entered as a place holder. |
| Application Identifier (AI):            | 3103                                                                         |
| AI 3103 variable weight element string: | 001750                                                                       |
| 2D data string:                         | 9876543210                                                                   |

```
BARTYPE "RSS14ES"
BARSET #4,2,68,2,1,4,4,124,1
PRBAR "01900123456789003103001750|9876543210"
```

## **Setup Bar Codes**

Intermec Fingerprint v8.xx-compatible EasyCoder printers can optionally be fitted with an EasySet bar code wand or a scanner (see the printer's User's Guide). By reading a special bar code containing encoded data for one or several setup parameters, the printer's setup can easily be changed, even by a person without any knowledge of Intermec Fingerprint, the Direct Protocol, or their supporting software.

You could print such bar codes with your printer and paste them on a board in the vicinity of the printer. When, for example, the operator needs to switch to another type of media, he or she will only have to pick up the EasySet wand or scanner and read the appropriate bar code.

The only bar code that can be used for this purpose is a Code 128 containing the function character FNC3 (ASCII 130 dec). If the FNC character is missing, the printer will regard the bar code as containing ordinary ASCII input to the "wand:" device.

Please refer to the *EasySet Bar Code Wand Setup* manual for syntax and parameter descriptions.



# **6** **RFID Tag Formats**

This chapter includes information on supported RFID tag formats and media. For more information on supported tags and media, contact your Intermec representative.

## About EPC Tag Formats

EPC (Electronic Product Code) are derived from EAN.UCC standards. The concept is to program RFID tags with the same information from EAN.UCC encoding schemes normally printed on bar codes. For an example, see “[EPC Tag Writing Example](#)” on page 340.

The tag format information in the tables in this section include only the input data necessary to program the tags with Fingerprint. More information is in fact contained by the tags, such as “header” and “partition”:

- The header is uniquely defined by the tag format and not included in the input.
- The partition does not need to be entered, as it is defined by the length of other fields (for example, “company prefix” and “item reference”) and is automatically calculated by Fingerprint.

It is therefore very important to enter the correct number of digits in each field, even if they may contain leading zeroes.

All values entered for EPC formats must be integers entered as strings. However, United States Department of Defense (USDOD) formats do not strictly conform to EPC norms and may differ slightly.

More information on EPC and USDOD tag formats can be obtained from:

- [www.epcglobalinc.org](http://www.epcglobalinc.org).
- [www.dodrfid.org](http://www.dodrfid.org).

## About the Uniform Resource Identifier (URI)

A standardized method to write to a tag is provided by the EPC standard to simplify encoding of tags. The URI (Uniform Resource Identifier) representations for EPC tag formats contain just the fields necessary to distinguish objects from each other, and are included in the tables above. In order to use the URIs, the tag format “EPC-URN” must be previously specified by the TAGFORMAT command. EPC is a namespace in the Uniform Resource Name (URN) encoding scheme.

### Example

```
10 TAGFIELD "@EPC"
20 TAGFORMAT "EPC-URN"
30 TAGWRITE "urn:epc:tag:sscc-96:0.12345678.987654321"
```

## Tag Formats

The following list conforms to revision 1.27 of the EPC Tag Data Standards Version 1.1. The Gen 2 standard does not mention the use of 64-bit tags, although they are supported by Fingerprint:

- SGTIN: Serialized Global Trade Identification Number
- SSCC: Serial Shipping Container Code
- SGLN: Serialized Global Location Number
- GIAI: Global Individual Asset Identifier
- GRAI: Global Returnable Asset Identifier
- USDOD: United States Department of Defense

### SGTIN-64

| Field                              | Number of Digits | Range        |
|------------------------------------|------------------|--------------|
| Filter                             | Not applicable   | 0-7          |
| Company Prefix Index (CoPrefIndex) | Not applicable   | 0-16 383     |
| Item Reference (ItemRef)           | Not applicable   | 0-1 048 575  |
| Serial Number (SerialNo)           | Not applicable   | 0-33 554 431 |

#### URI Representation

"urn:epc:tag:sgtin-64:Filter.CoPrefIndex.ItemRef.SerialNo"

### SGTIN-96

| Field                              | Number of Digits | Range             |
|------------------------------------|------------------|-------------------|
| Filter                             | Not applicable   | 0-7               |
| Company Prefix Index (CoPrefIndex) | 6-121            | Not applicable    |
| Item Reference (ItemRef)           | 7-11             | Not applicable    |
| Serial Number (SerialNo)           | Not applicable   | 0-274 877 906 943 |

#### URI Representation

"urn:epc:tag:sgtin-96:Filter.CoPrefIndex.ItemRef.SerialNo"

#### Notes

Number of digits in Company Prefix and Item Reference must total 13

**SSCC-64**

| Field                              | Number of Digits | Range            |
|------------------------------------|------------------|------------------|
| Filter                             | Not applicable   | 0-7              |
| Company Prefix Index (CoPrefIndex) | Not applicable   | 0-16 383         |
| Serial Number (SerialNo)           | Not applicable   | 0-99 999 999 999 |

**URI Representation**

"urn:epc:tag:sscc-64:Filter.CoPrefIndex.SerialNo"

**SSCC-96**

| Field                        | Number of Digits | Range          |
|------------------------------|------------------|----------------|
| Filter                       | Not applicable   | 0-7            |
| Company Prefix (CoPref)      | 6-122            | Not applicable |
| Serial Reference (SerialRef) | 11-52            | Not applicable |

**URI Representation**

"urn:epc:tag:sscc-96:Filter.CoPref.SerialRef"

**Notes**

Number of digits in Company Prefix and Serial Reference must total 17.

**SGLN-64**

| Field                              | Number of Digits | Range     |
|------------------------------------|------------------|-----------|
| Filter                             | Not applicable   | 0-7       |
| Company Prefix Index (CoPrefIndex) | Not applicable   | 0-16 383  |
| Location Reference (LocRef)        | Not applicable   | 0-999 999 |
| Serial Number (SerialNo)           | Not applicable   | 0-524 287 |

**URI Representation**

"urn:epc:tag:schl-64:Filter.CoPreIndex.LocRefe.SerialNo"

**Notes**

The serial number can be left out when writing to a tag, as EAN.UCC specifications do not yet allow the use of a serial number in SGLN tags.

**SGLN-96**

| Field                              | Number of Digits | Range               |
|------------------------------------|------------------|---------------------|
| Filter                             | Not applicable   | 0-7                 |
| Company Prefix Index (CoPrefIndex) | 6-124            | Not applicable      |
| Location Reference (LocRef)        | 6-04,5           | Not applicable      |
| Serial Number (SerialNo)           | Not applicable   | 022 199 023 255 551 |

**URI Representation**

"urn:epc:tag:sgln-96:Filter.CoPrefIndex.LocRef.SerialNo"

**Notes**

Total number of digits in Company Prefix Index and Location Reference must be 12.

An empty Location Reference must be entered as an empty string ("").

The serial number can be left out when writing to a tag, as EAN/UCC specifications do not yet allow the use of serial numbers in SGLN tags.

**GRAI-64**

| Field                              | Number of Digits | Range     |
|------------------------------------|------------------|-----------|
| Filter                             | Not applicable   | 0-7       |
| Company Prefix Index (CoPrefIndex) | Not applicable   | 0-16 383  |
| Asset Type (Asset)                 | Not applicable   | 0-999 999 |
| Serial Number (SerialNo)           | Not applicable   | 0-524 287 |
| URI Representation                 |                  |           |

**URI Representation**

"urn:epc:tag:grai-64:Filter.CoPreIndex.Asset.SerialNo"

**GRAI-96**

| Field                              | Number of Digits | Range             |
|------------------------------------|------------------|-------------------|
| Filter                             | Not applicable   | 0-7               |
| Company Prefix Index (CoPrefIndex) | 6-127            | Not applicable    |
| Asset Type (Asset)                 | 6-07             | Not applicable    |
| Serial Number (SerialNo)           | Not applicable   | 0-274 887 906 943 |

**URI Representation**

"urn:epc:tag:sgln-96:Filter.CoPrefIndex.Asset.SerialNo"

**Notes**

Total number of digits in Company Prefix and Asset Type must be 12.

An empty Asset Type must be entered as an empty string ("").

**GIAI-64**

| Field                                    | Number of Digits | Range             |
|------------------------------------------|------------------|-------------------|
| Filter                                   | Not applicable   | 0-7               |
| Company Prefix Index (CoPrefIndex)       | Not applicable   | 0-16 383          |
| Individual Asset Reference (IndAssetRef) | Not applicable   | 0-549 755 813 887 |

**URI Representation**

"urn:epc:tag:gaii-64:Filter.CoPrefIndex.IndAssetRef"

**GIAI-96**

| Field                                 | Number of Digits | Range          |
|---------------------------------------|------------------|----------------|
| Filter                                | Not applicable   | 0-7            |
| Company Prefix Index (CoPrefIndex)    | 6-129            | Not applicable |
| Individual Asset Reference (IndAsset) | 18-129           | Not applicable |
| URI Representation                    |                  |                |
| Special Note:                         |                  |                |

**URI Representation**

"urn:epc:tag:gaii 96:Filter.CoPrefIndex.IndAsset.SerialNo"

**Notes**

Total number of digits in Company Prefix and Individual Asset Reference must be 24.

**GID-96**

| Field                         | Number of Digits | Range             |
|-------------------------------|------------------|-------------------|
| General Manager Number (GMNo) | Not applicable   | 0-268 435 455     |
| Object Class (Object)         | Not applicable   | 0-16 777 215      |
| Serial Number (SerialNo)      | Not applicable   | 0-549 755 813 887 |

**URI Representation**

"urn:epc:tag:gid-96:GMNo.Object.SerialNo"

**USDOD-64**

| Field                                | Number of Digits     | Range                            |
|--------------------------------------|----------------------|----------------------------------|
| Filter                               | Not applicable       | 0-3                              |
| Government Managed Identifier (GMID) | Exactly 5 characters | 0-9, SPACE, and A-Z (not I or O) |
| Serial Number (SerialNo)             | Not applicable       | 0-16 777 215                     |

**URI Representation**

"urn:epc:tag:usdod-64:Filter.GMID.SerialNo"

## USDOD-96

| Field                                | Number of Digits     | Range                               |
|--------------------------------------|----------------------|-------------------------------------|
| Filter                               | Not applicable       | 0-15                                |
| Government Managed Identifier (GMID) | Exactly 5 characters | 0-9, SPACE, and A-Z<br>(not I or O) |
| Serial Number (SerialNo)             | Not applicable       | 0-68 719 476 735                    |

### URI Representation

"urn:epc:tag:usdod-64:Filter.GMID.SerialNo"

## Other EPC Tag Input Methods

Although Intermec recommends that you use the syntax described in the previous section to write to an EPC tag, it is possible to write the full length of the 64-bit or 96-bit chip manually in hex format. You can use the tag formats "EPC-HEX64" and "EPC-HEX96" to manually write to the tag.

### EPC-HEX64 Format

Number of bytes: 8

Allowed values: 0 to 9, A to F in hex format

#### Example

```
10 TAGFIELD "@ID"
20 TAGFORMAT "EPC-HEX64"
30 TAGWRITE "1122334455667788"
```

### EPC-HEX96 Format

Number of bytes: 12

Allowed values: 0 to 9, A to F in hex format

#### Example

```
10 TAGFIELD "@ID"
20 TAGFORMAT "EPC-HEX96"
30 TAGWRITE "11223344556677889900AABB"
```

## EPC Tag Writing Example

The following example shows how to include EAN.UCC encoding schemes (as currently used in printed bar codes) in writing to an RFID tag. This example uses a Serialized Global Trade Item Number (SGTIN) formed by the GTIN 10614141007346 and serial number 2. The GTIN is formed by an indicator, company prefix and item reference.

GTIN Indicator: 1  
 Company Prefix: 0614141  
 Item reference: 00734  
 Check digit: 6  
 Serial Number: 2

This would normally be encoded as an EAN128 bar code:



| Callout | Description             | Value in example |
|---------|-------------------------|------------------|
| A       | Application identifiers | (01)             |
| B       | GTIN indicator          | 1                |
| C       | Company prefix index    | 0614141          |
| D       | Item reference          | 00734            |
| E       | Check digit             | 6                |
| F       | Application identifier  | (21)             |
| G       | Serial number           | 2                |

For the correct syntax, the application identifiers (AI) are dropped because this information is included in the header. The check digit is dropped. The GTIN indicator is repositioned at the leftmost position of the item reference, to form a new item reference. A filter value of 3 is chosen for this example (signifying Standard Trade Item Group, though filter definitions are non-normative at this time). This gives us the necessary values to program an SGTIN-96 tag, in this order:

Filter: 3  
 Company Prefix Index: 0614141  
 Item reference: 100734  
 Serial number: 2

We could use the following Fingerprint programs to program an SGTIN-96 tag with this information, The first example assumes a Gen 2 tag, and the second an ISO 18000-6B chip:

**Example 1: EPCglobal Class 1 Gen 2 Tag**

```

10 FILTER$ = "3"
20 PREFIX$ = "0614141"
30 ITEM$ = "100734"
40 SERIAL$ = "2"
50 TAGFIELD "@EPC"
60 TAGFORMAT "SGTIN-96"
70 TAGWRITE FILTER$, PREFIX$, ITEM$, SERIAL$

```

**Example 2: ISO 18000-6B Tag**

```

10 TAGFIELD "@DATA",10,12
20 TAGFORMAT "EPC-URN"
30 TAGWRITE "urn:epc:tag:sgtin-96:3.0614141.100734.2"

```

**About Tag Memory Allocation Standards**

Tag standards currently supported by Fingerprint firmware are:

- EPC UCode 1.19.
- ISO 18000-6B.
- EPC Class 1.
- EPCglobal Class 1 Gen 2.

These tags have different memory structures, and thus may need different input parameters for the TAGFIELD and TAGFORMAT statement. The tag memory structure for these supported tags are shown in the following tables.

**UCode EPC 1.19**

| <b>Byte</b> |      | <b>@ID</b> | <b>@DATA</b> | <b>@ALL</b> | <b>Read/Write</b> | <b>Comment</b>          |
|-------------|------|------------|--------------|-------------|-------------------|-------------------------|
| 0           | ID   | 0          |              | 0           | Read only         | EPCGlobal<br>(HEX EF04) |
| 1           |      | 1          |              | 1           |                   |                         |
| 2           |      | 2          |              | 2           |                   |                         |
| 3           |      | 3          |              | 3           |                   |                         |
| 4           |      | 4          |              | 4           |                   |                         |
| 5           |      | 5          |              | 5           |                   |                         |
| 6           |      | 6          |              | 6           |                   |                         |
| 7           |      | 7          |              | 7           |                   |                         |
| 8           | Data |            | 0            | 8           | Read/Write        | User Data               |
| 9           |      |            | 1            | 9           |                   |                         |
| 10          |      |            | 2            | 10          |                   |                         |
| 11          |      |            | 3            | 11          |                   |                         |
| 12          |      |            | 4            | 12          |                   |                         |
| 13          |      |            | 5            | 13          |                   |                         |
| 14          |      |            | 6            | 14          |                   |                         |
| 15          |      |            | 7            | 15          |                   |                         |
| 16          | ID   | 8          |              | 16          | Read/Write        | EPC Data                |
| 17          |      | 9          |              | 17          |                   |                         |
| 18          |      | 10         |              | 18          |                   |                         |
| 19          |      | 11         |              | 19          |                   |                         |
| 20          |      | 12         |              | 20          |                   |                         |
| 21          |      | 13         |              | 21          |                   |                         |
| 22          |      | 14         |              | 22          |                   |                         |
| 23          |      | 15         |              | 23          |                   |                         |
| 24          | Data |            | 8            | 24          | Read/Write        | User Data               |
| 25          |      |            | 9            | 25          |                   |                         |
| 26          |      |            | 10           | 26          |                   |                         |
| ...         |      |            | ...          | ...         |                   |                         |
| ...         |      |            | ...          | ...         |                   |                         |
| ...         |      |            | ...          | ...         |                   |                         |
| ...         |      |            | ...          | ...         |                   |                         |
| 47          |      |            | 31           | 47          |                   |                         |

When writing EPC data to the @ID segment of a UCode EPC 1.19 tag, Fingerprint ignores input parameters (and other defaults) for the TAGFIELD command, and automatically starts writing at byte 2 of the @ID segment. When writing to the @DATA segment, valid start byte and field length parameters must be entered in the TAGFIELD instruction.



**Note:** No manual mapping of the EPC data to the UCode EPC 1.19 memory structure is necessary, as this is done automatically by Fingerprint firmware.

## ISO 18000-6B (UCode HSL) Tag Memory Allocation

| Byte             |      | @ID              | @DATA            | @ALL             | Read/Write | Comment                                                                                            |
|------------------|------|------------------|------------------|------------------|------------|----------------------------------------------------------------------------------------------------|
| 0                | ID   | 0                |                  | 0                | Read only  | Unique ID                                                                                          |
| 1                |      | 1                |                  | 1                |            |                                                                                                    |
| 2                |      | 2                |                  | 2                |            |                                                                                                    |
| 3                |      | 3                |                  | 3                |            |                                                                                                    |
| 4                |      | 4                |                  | 4                |            |                                                                                                    |
| 5                |      | 5                |                  | 5                |            |                                                                                                    |
| 6                |      | 6                |                  | 6                |            |                                                                                                    |
| 7                |      | 7                |                  | 7                |            |                                                                                                    |
| 8                | Data | 0                | 8                | 8                | Read only  | Tag Manufacturer                                                                                   |
| 9                |      | 1                | 9                | 9                |            |                                                                                                    |
| 10               |      | 2                | 10               | 10               | Read Only  | Tag Hardware Type                                                                                  |
| 11               |      | 3                | 11               | 11               |            |                                                                                                    |
| 12               |      | 4                | 12               | 12               | Read/Write | Tag Memory Layout                                                                                  |
| 13               |      | 5                | 13               | 13               |            |                                                                                                    |
| 14               |      | 6                | 14               | 14               |            |                                                                                                    |
| 15               |      | 7                | 15               | 15               |            |                                                                                                    |
| 16               |      | 8                | 16               | 16               |            |                                                                                                    |
| 17               |      | 9                | 17               | 17               |            |                                                                                                    |
| 18               |      | 10               | 18               | 18               | Read/Write | User Data<br><br>Definition and<br>format of<br>User Data<br>determined by<br>Tag Memory<br>Layout |
| 19               |      | 11               | 19               | 19               |            |                                                                                                    |
| 20               |      | 12               | 20               | 20               |            |                                                                                                    |
| ...              |      | ...              | ...              | ...              |            |                                                                                                    |
| ...              |      | ...              | ...              | ...              |            |                                                                                                    |
| ...              |      | ...              | ...              | ...              |            |                                                                                                    |
| ...              |      | ...              | ...              | ...              |            |                                                                                                    |
| ...              |      | ...              | ...              | ...              |            |                                                                                                    |
| ...              |      | ...              | ...              | ...              |            |                                                                                                    |
| 223 <sub>1</sub> |      | 215 <sub>1</sub> | 223 <sub>1</sub> | 223 <sub>1</sub> |            |                                                                                                    |

<sup>1</sup> The actual byte length of the tag may vary.

When writing to an ISO 18000-6B tag, EPC data must be written to the @DATA field, and valid start byte and field length parameters must be entered in the TAGFIELD instruction.

### Example

```

10 FILTER$ = "3"
20 PREFIX$ = "0614141"
30 ITEM$ = "100734"
40 SERIAL$ = "2"
50 TAGFIELD "@DATA", 10, 12
60 TAGFORMAT "SGTIN-96"
70 TAGWRITE FILTER$, PREFIX$, ITEM$, SERIAL$
```

## Class 1 (EPC Class 1 Version 1) Tag Memory Allocation

| <b>Byte</b> | <b>Row</b> | <b>Pointer value (bit)</b> | <b>@ID</b> | <b>EPC 64</b>                                   | <b>EPC 96</b>                                   |
|-------------|------------|----------------------------|------------|-------------------------------------------------|-------------------------------------------------|
| 0           | 0          | 0                          |            | Tag CRC - no Read nor Write                     |                                                 |
| 1           |            |                            |            |                                                 |                                                 |
| 2           | 1          | 16                         | 0          | Tag ID most significant bytes (EPC data) - R/W  |                                                 |
| 3           |            |                            | 1          |                                                 |                                                 |
| 4           | 2          | 32                         | 2          | Tag ID (EPC data) - R/W                         |                                                 |
| 5           |            |                            | 3          |                                                 |                                                 |
| 6           | 3          | 48                         | 4          | Tag ID (EPC data)- R/W                          |                                                 |
| 7           |            |                            | 5          |                                                 |                                                 |
| 8           | 4          | 64                         | 6          | Tag ID least significant bytes (EPC data) - R/W | Tag ID (EPC data)- R/W                          |
| 9           |            |                            | 7          |                                                 |                                                 |
| 10          | 5          | 80                         | 8          | Kill Passcode - R/W                             | Tag Id (EPC data) - R/W                         |
| 11          |            |                            | 9          | Lock bits --/-                                  |                                                 |
| 12          | 6          | 96                         | 10         |                                                 | Tag ID least significant bytes (EPC data) - R/W |
| 13          |            |                            | 11         |                                                 |                                                 |
| 14          | 7          | 112                        | 12         |                                                 | Kill Passcode - R/W                             |
| 15          |            |                            |            |                                                 | Lock bits --/-                                  |

For Class 1 tags, EPC data is stored in the @ID segment, bytes 0-7 for 64-bit and bytes 1-11 for 96-bit tags. The values for the parameters <nexp2> and <nexp3> for the TAGFIELD command do not need to be entered when writing EPC data, as these are assigned automatically.

The Kill passcode can be addressed as byte 12 in EPC-96 (byte 8 in EPC-64). It must however be addressed independently with a TAGFIELD, "@ID", 12, 1 command.

The Class 1 standard defines a tag CRC (cyclic redundancy check) for tag ID bits, and if incorrect, the tag will not be detected by a TAGREAD operation.

Erasing all tag information is done by the following command:

```
TAGFIELD "@ID", 0, 0
TAGFORMAT "HEX"
TAGWRITE ""
```

## EPCglobal Class 1 Gen 2 Tag Memory Allocation

| Field    | @Reserved      | @EPC | @TID | @USER         | Content               | Read/Write |  |  |
|----------|----------------|------|------|---------------|-----------------------|------------|--|--|
| RESERVED | 0              |      |      |               | Kill Password         | R/W        |  |  |
|          | 1              |      |      |               | Access Password       | R/W        |  |  |
|          | 2              |      |      |               |                       |            |  |  |
|          | 3              |      |      |               | Optional              | R/W        |  |  |
|          | 4              |      |      |               |                       |            |  |  |
|          | 5              |      |      |               | CRC-16                | R/-        |  |  |
|          | 6              |      |      |               |                       |            |  |  |
|          | 7              |      |      |               | Protocol Control Bits | R/W        |  |  |
|          | 8              |      |      |               |                       |            |  |  |
|          | ...            |      |      |               | EPC data              | R/W        |  |  |
|          | n              |      |      |               |                       |            |  |  |
| EPC      | 0              |      |      |               |                       |            |  |  |
|          | 1              |      |      |               |                       |            |  |  |
|          | 2              |      |      |               |                       |            |  |  |
|          | 3              |      |      |               |                       |            |  |  |
|          | 4              |      |      |               |                       |            |  |  |
|          | 5              |      |      |               |                       |            |  |  |
|          | 6              |      |      |               |                       |            |  |  |
|          | 7              |      |      |               |                       |            |  |  |
|          | 8              |      |      |               |                       |            |  |  |
|          | 9              |      |      |               |                       |            |  |  |
|          | 10             |      |      |               |                       |            |  |  |
|          | 11             |      |      |               |                       |            |  |  |
|          | 12             |      |      |               |                       |            |  |  |
|          | 13             |      |      |               |                       |            |  |  |
|          | 14             |      |      |               |                       |            |  |  |
| TID      | 15             |      |      |               |                       |            |  |  |
|          | 16             |      |      |               |                       |            |  |  |
|          | ...            |      |      |               |                       |            |  |  |
|          | n <sub>2</sub> |      |      |               |                       |            |  |  |
|          | 0              |      |      | Optional data | R/W                   |            |  |  |
| USER     | 1              |      |      |               |                       |            |  |  |
|          | 2              |      |      |               |                       |            |  |  |
|          | 3              |      |      |               |                       |            |  |  |
|          | ...            |      |      |               |                       |            |  |  |
|          | n <sub>3</sub> |      |      |               |                       |            |  |  |
| USER     | 0              |      |      |               | Optional User Data    | R/W        |  |  |
|          | ...            |      |      |               |                       |            |  |  |
|          | n <sub>4</sub> |      |      |               |                       |            |  |  |

The @RESERVED field contains kill and access passwords. These are not always implemented, and the tag acts as if they were zero-valued passwords.

The @EPC segment is specifically designed to hold EPC values, and is at least 96 bits long. When writing EPC information to the @EPC segment, values for the <nexp2> and <nexp3> values in the TAGFIELD command need not be defined.

For EPCglobal applications, the @TID segment contains special tag and vendor-specific data. This segment may contain an identifier for ISO/IEC 15963, but this is not an EPCglobal application.

The @USER bank allow for user-specific data storage. The organization of user memory for EPCglobal applications is vendor-defined.

## About Non-Standard Tag Formats

Fingerprint allows users to program RFID tags with other data than EPC tag data. The available tag formats are “NUM”, “HEX”, and “ASCII”, representing numeric, hexadecimal, and ASCII data respectively. When reading tags, the data is assumed to be in the format last specified by the TAGFORMAT command. The last example below demonstrates how the ASCII characters written as “RFID” will be read as “52464944” if the last format is “HEX”.

### Examples

```
10 TAGFIELD "@ID", 2, 4
20 TAGFORMAT "NUM"
30 TAGWRITE 1234

10 TAGFIELD "@DATA", 10, 12
20 TAGFORMAT "HEX"
30 TAGWRITE "11223344556677889900AABB"

10 TAGFIELD "@DATA", 10, 4
20 TAGFORMAT "ASCII"
30 TAGWRITE "RFID"
40 TAGFORMAT "HEX"
50 TAGREAD A$
60 PRINT A$
RUN
```

resulting in:

52464944



# **7** Error Codes

This section lists the error code messages you may see when using Fingerprint to send instructions to the printer.

## Error Code Messages

| <b>Code</b> | <b>Message/Explanation</b>           |
|-------------|--------------------------------------|
| 0           | No error.                            |
| 1           | Syntax error.                        |
| 2           | Unbalanced parentheses.              |
| 3           | Feature not implemented.             |
| 4           | Evaluation syntax error.             |
| 5           | Unrecognized token.                  |
| 6           | Tokenized line too long.             |
| 7           | Evaluation stack overflow.           |
| 8           | Error in exectab.                    |
| 9           | Undefined token.                     |
| 10          | Non-executing token.                 |
| 11          | Evaluation stack underflow.          |
| 12          | Type mismatch.                       |
| 13          | Line not found.                      |
| 14          | Division with zero.                  |
| 15          | Font not found.                      |
| 16          | Bar code device not found.           |
| 17          | Bar code type not implemented.       |
| 18          | Disk full.                           |
| 19          | Error in file name.                  |
| 20          | Input line too long.                 |
| 21          | Error stack overflow.                |
| 22          | RESUME without error.                |
| 23          | Image not found.                     |
| 24          | Overflow in temporary string buffer. |
| 25          | Wrong number of parameters.          |
| 26          | Parameter too large.                 |
| 27          | Parameter too small.                 |
| 28          | RETURN without GOSUB                 |
| 29          | Error in startup file.               |
| 30          | Assign to a read-only variable.      |
| 31          | Illegal file number.                 |
| 32          | File is already open.                |
| 33          | Too many files open.                 |
| 34          | File is not open.                    |
| 37          | Cutter device not found.             |
| 38          | User break.                          |

| <b>Code</b> | <b>Message/Explanation</b>        |
|-------------|-----------------------------------|
| 39          | Illegal line number.              |
| 40          | Run statement in program.         |
| 41          | Parameter out of range.           |
| 42          | Illegal bar code ratio.           |
| 43          | Memory overflow.                  |
| 44          | File is write protected.          |
| 45          | Unknown store option              |
| 46          | Store already in progress.        |
| 47          | Unknown store protocol.           |
| 48          | No store defined.                 |
| 49          | NEXT without FOR.                 |
| 50          | Bad store record header.          |
| 51          | Bad store address.                |
| 52          | Bad store record.                 |
| 53          | Bad store checksum.               |
| 54          | Bad store record end.             |
| 55          | Remove in ROM.                    |
| 56          | Illegal communication channel.    |
| 57          | Subscript out of range.           |
| 58          | Field overflow.                   |
| 59          | Bad record number.                |
| 60          | Too many strings.                 |
| 61          | Error in setup file.              |
| 62          | File is list protected.           |
| 63          | ENTER function.                   |
| 64          | FOR without NEXT                  |
| 65          | Evaluation overflow.              |
| 66          | Bad optimizing type.              |
| 67          | Error from communication channel. |
| 68          | Unknown execution identity.       |
| 69          | Not allowed in immediate mode.    |
| 70          | Line label not found.             |
| 71          | Line label already defined.       |
| 72          | IF without ENDIF.                 |
| 73          | ENDIF without IF.                 |
| 74          | ELSE without ENDIF.               |
| 75          | ELSE without IF.                  |
| 76          | WHILE without WEND.               |
| 77          | WEND without WHILE.               |

| <b>Code</b> | <b>Message/Explanation</b>           |
|-------------|--------------------------------------|
| 78          | Not allowed in execution mode.       |
| 79          | Not allowed in a layout.             |
| 80          | Download timeout.                    |
| 81          | Exit to system.                      |
| 82          | Invalid cont environment.            |
| 83          | ETX Timeout.                         |
| 1001        | Not implemented.                     |
| 1002        | Memory too small.                    |
| 1003        | Field out of label.                  |
| 1004        | Wrong font to chosen direction.      |
| 1005        | Out of paper.                        |
| 1006        | No field to print.                   |
| 1007        | Lss too high.                        |
| 1008        | Lss too low.                         |
| 1009        | Invalid parameter.                   |
| 1010        | Hardware error.                      |
| 1011        | I/O error.                           |
| 1012        | Too many files as opened.            |
| 1013        | Device not found.                    |
| 1014        | File not found.                      |
| 1015        | File is read-only.                   |
| 1016        | Illegal argument.                    |
| 1017        | Results are too large.               |
| 1018        | Bad file descriptor.                 |
| 1019        | Invalid font.                        |
| 1020        | Invalid image.                       |
| 1021        | Too large argument for MAG.          |
| 1022        | Head lifted.                         |
| 1023        | Incomplete label.                    |
| 1024        | File too large.                      |
| 1025        | File does not exist.                 |
| 1026        | Label pending.                       |
| 1027        | Out of transfer ribbon.              |
| 1028        | Paper type is not selected.          |
| 1029        | Printhead voltage too high.          |
| 1030        | Character is missing in chosen font. |
| 1031        | Next label not found.                |
| 1032        | File name too long.                  |
| 1033        | Too many files are open.             |

| <b>Code</b> | <b>Message/Explanation</b>                     |
|-------------|------------------------------------------------|
| 1034        | Not a directory.                               |
| 1035        | File pointer is not inside the file.           |
| 1036        | Subscript out of range.                        |
| 1037        | No acknowledge received within specified time. |
| 1038        | Communication checksum error.                  |
| 1039        | Not mounted.                                   |
| 1040        | Unknown file operating system.                 |
| 1041        | Error in fos structure.                        |
| 1042        | Internal error in mcs.                         |
| 1043        | Timer table full.                              |
| 1044        | Low battery in memory card                     |
| 1045        | Media was removed.                             |
| 1046        | Memory checksum error.                         |
| 1047        | Interrupted system call.                       |
| 1051        | Dot resistance measure out of limits.          |
| 1052        | Error in printhead.                            |
| 1053        | Unable to complete a dot measurement.          |
| 1054        | Error when trying to write to device.          |
| 1055        | Error when trying to read from device.         |
| 1056        | O_BIT open error.                              |
| 1057        | File exists.                                   |
| 1058        | Transfer ribbon is installed.                  |
| 1059        | Cutter does not respond.                       |
| 1061        | Wrong type of media.                           |
| 1062        | Not allowed.                                   |
| 1067        | Is a directory.                                |
| 1073        | Directory not empty.                           |
| 1076        | Permission denied.                             |
| 1077        | Broken pipe.                                   |
| 1081        | Timer expired.                                 |
| 1082        | Unsupported protocol.                          |
| 1083        | Ribbon low.                                    |
| 1084        | Paper low.                                     |
| 1085        | Connection timed out.                          |
| 1086        | Secret not found.                              |
| 1087        | Paper jam.                                     |
| 1088        | Printhead too hot.                             |
| 1101        | Illegal character in bar code.                 |
| 1102        | Illegal bar code font.                         |

| <b>Code</b> | <b>Message/Explanation</b>                 |
|-------------|--------------------------------------------|
| 1103        | Too many characters in bar code.           |
| 1104        | Bar code too large.                        |
| 1105        | Bar code parameter error.                  |
| 1106        | Wrong number of characters.                |
| 1107        | Illegal bar code size.                     |
| 1108        | Number of rows out of range.               |
| 1109        | Number of columns out of range.            |
| 1112        | ECI syntax error.                          |
| 1201        | Insufficient font data loaded.             |
| 1202        | Transformation matrix out of range.        |
| 1203        | Font format error.                         |
| 1204        | Specifications not compatible with output. |
| 1205        | Intelligent transform not supported.       |
| 1206        | Unsupported output mode requested.         |
| 1207        | Extended font not supported.               |
| 1208        | Font specifications not set.               |
| 1209        | Track kerning data not available.          |
| 1210        | Pair kerning data not available.           |
| 1211        | Other Speedo error.                        |
| 1212        | No bitmap or outline device.               |
| 1212        | No bitmap or outline device.               |
| 1213        | Speedo error six.                          |
| 1214        | Squeeze or clip not supported.             |
| 1215        | Character data not available.              |
| 1216        | Unknown font.                              |
| 1217        | Font format is not supported.              |
| 1218        | Correct mapping table is not found.        |
| 1219        | Font is in the wrong direction.            |
| 1220        | Error in external map table.               |
| 1221        | Map table was not found.                   |
| 1222        | Double byte map table is missing.          |
| 1223        | Single byte map table is missing.          |
| 1224        | Character map function is missing.         |
| 1225        | Double byte font is not selected.          |
| 1301        | Index outside collection bounds.           |
| 1302        | Collection could not be expanded.          |
| 1303        | Parameter is not a collection.             |
| 1304        | Item not a member of the collection.       |
| 1305        | No compare function, or returns faulty.    |

| <b>Code</b> | <b>Message/Explanation</b>                               |
|-------------|----------------------------------------------------------|
| 1306        | Tried to insert a duplicate item.                        |
| 1320        | No RFID support installed.                               |
| 1321        | No tag found.                                            |
| 1322        | Access outside tag memory.                               |
| 1323        | Access too long for block.                               |
| 1324        | RFID inactive.                                           |
| 1601        | Reference Font Not Found.                                |
| 1602        | Error in Wand-Device.                                    |
| 1603        | Error in Slave Processor.                                |
| 1604        | Print Shift Error.                                       |
| 1605        | No Hardware Lock.                                        |
| 1606        | Testfeed not done.                                       |
| 1607        | General Print Error.                                     |
| 1608        | Access Denied.                                           |
| 1609        | Specified Feed Length Exceeded.                          |
| 1610        | Illegal Character Map File.                              |
| 1701        | Cutter not back in position after cut.                   |
| 1702        | Cutter has not reached upper position: unsuccessful cut. |
| 1703        | Cutter not back in position after unsuccessful cut.      |
| 1704        | Cutter open.                                             |
| 1710        | Power supply Generic Error.                              |
| 1711        | Power supply Pending.                                    |
| 1712        | Power Status OK.                                         |
| 1713        | Power supply Power Fail.                                 |
| 1714        | Power supply Over Volt V24.                              |
| 1715        | Power supply Under Volt V24.                             |
| 1716        | Power supply Over Volt VSTM.                             |
| 1717        | Power supply Under Volt VSTM.                            |
| 1718        | Power supply Over Temperature.                           |
| 1719        | Power supply Error.                                      |
| 1820        | No route to host.                                        |
| 1821        | Disc quota exceeded.                                     |
| 1833        | Connection refused.                                      |





# A Printer Keyboard Layouts

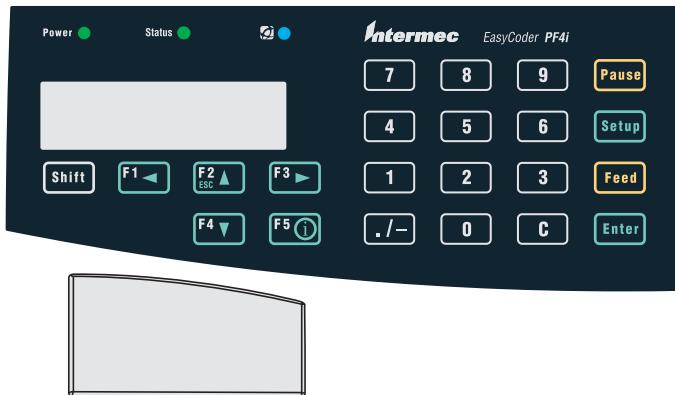
This appendix includes illustrations of the keyboards for Intermec EasyCoder printers that support Fingerprint v8.71 (10.0 for PD41).

Each group of illustrations shows the actual keyboard and keymaps for ID numbers (used with the ON KEY GOSUB command), position numbers, and ASCII values. You can remap the printer keyboard with the KEYBMAP\$ command.

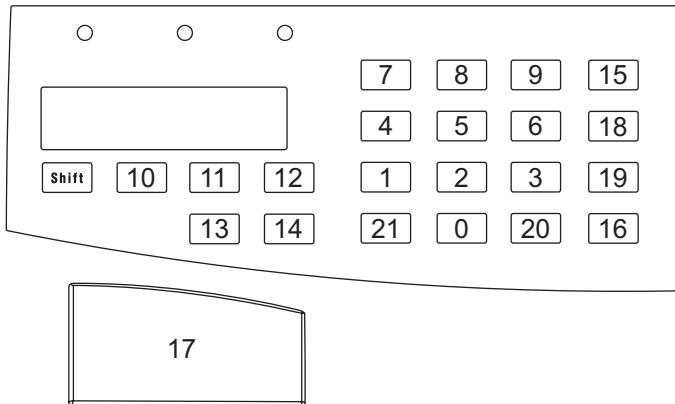
| Printer                 | Illustrations are on: |
|-------------------------|-----------------------|
| PF2i, PF4i, PF4ci       | 356                   |
| PM4i                    | 358                   |
| PX4i, PX6i              | 359                   |
| PX4i, PX6i Alphanumeric | 361                   |

## **PF2i, PF4i, PF4ci**

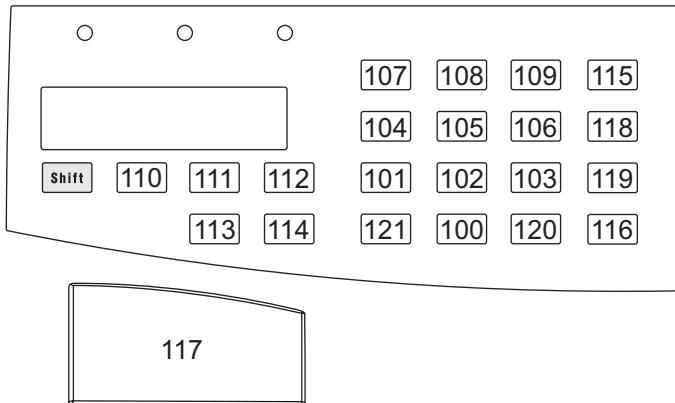
### **PF-Series Keyboard**



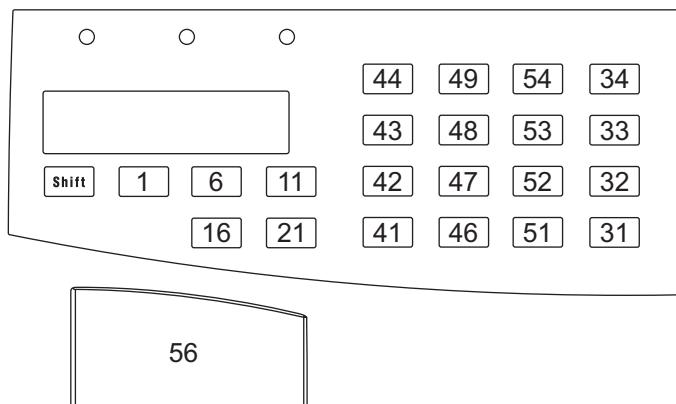
### **ID Numbers (Default)**



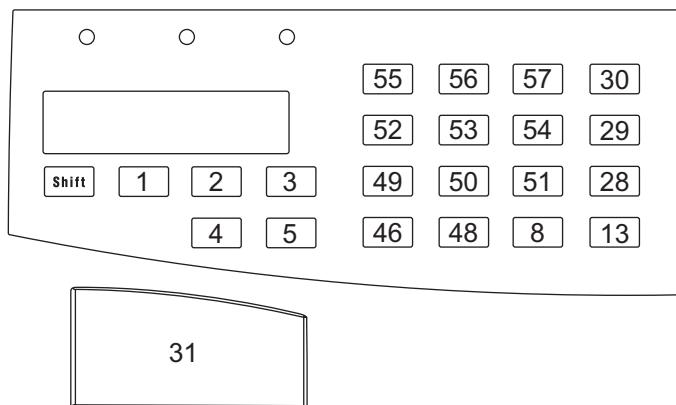
### **ID Numbers (+ Shift)**



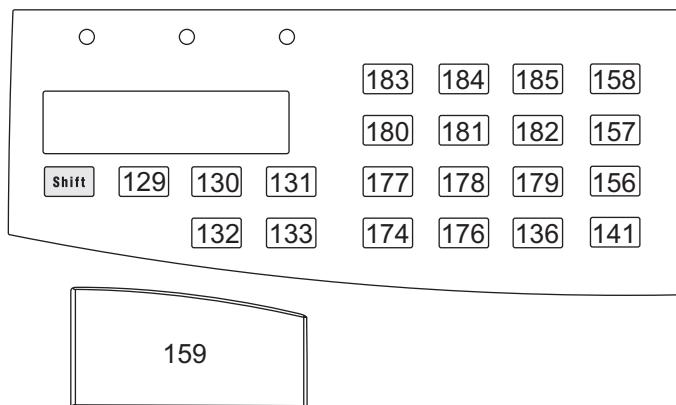
**Position Numbers**



**ASCII Values (Default)**

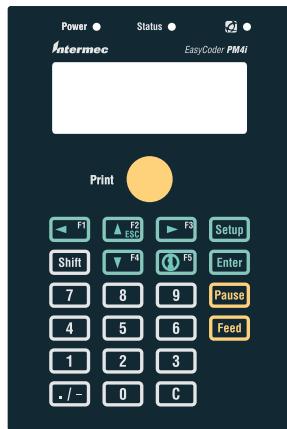


**ASCII Values (+ Shift)**



## **PM4i**

### **PM-Series Keyboard**



#### **ID Numbers (Default)**

17

|       |    |    |    |
|-------|----|----|----|
| 10    | 11 | 12 | 18 |
| Shift | 13 | 14 | 16 |
| 7     | 8  | 9  | 15 |
| 4     | 5  | 6  | 19 |
| 1     | 2  | 3  |    |
| 21    | 0  | 20 |    |

#### **ID Numbers (+ Shift)**

117

|       |     |     |     |
|-------|-----|-----|-----|
| 110   | 111 | 112 | 118 |
| Shift | 113 | 114 | 116 |
| 107   | 108 | 109 | 115 |
| 104   | 105 | 106 | 119 |
| 101   | 102 | 103 |     |
| 121   | 100 | 120 |     |

#### **Position Numbers**

56

|       |    |    |    |
|-------|----|----|----|
| 1     | 6  | 11 | 33 |
| Shift | 16 | 21 | 31 |
| 44    | 49 | 54 | 34 |
| 43    | 48 | 53 | 32 |
| 42    | 47 | 52 |    |
| 41    | 46 | 51 |    |

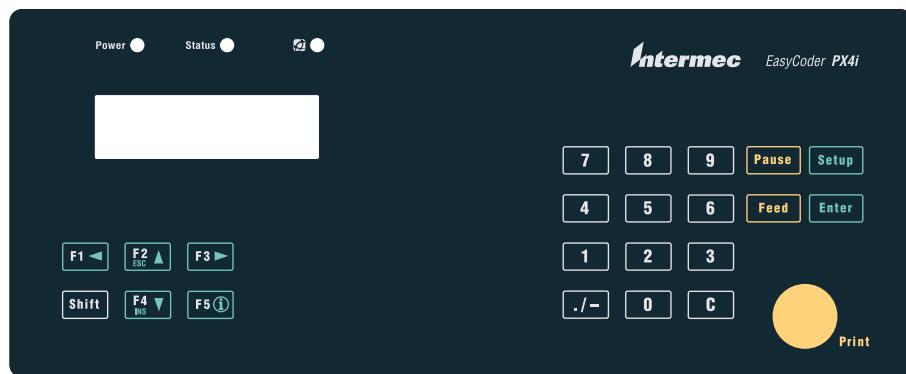
**ASCII Values (Default)**

|       |    |    |    |
|-------|----|----|----|
| 31    |    |    |    |
| 1     | 2  | 3  | 29 |
| Shift | 4  | 5  | 13 |
| 55    | 56 | 57 | 30 |
| 52    | 53 | 54 | 28 |
| 49    | 50 | 51 |    |
| 46    | 48 | 8  |    |

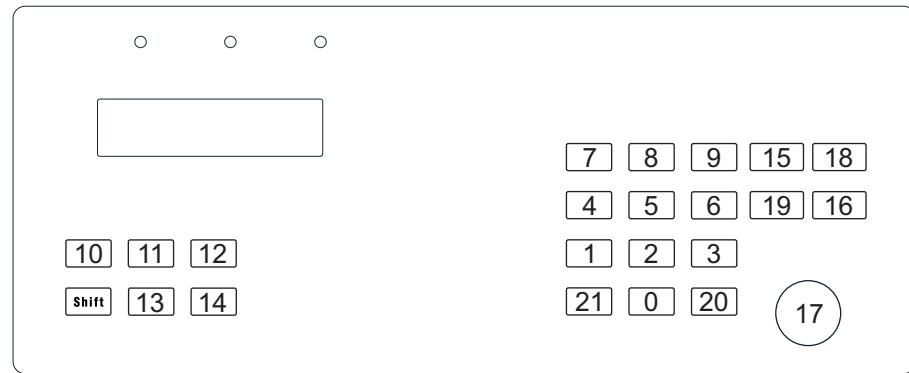
**ASCII Values (+ Shift)**

|       |     |     |     |
|-------|-----|-----|-----|
| 117   |     |     |     |
| 110   | 111 | 112 | 118 |
| Shift | 113 | 114 | 116 |
| 107   | 108 | 109 | 115 |
| 104   | 105 | 106 | 119 |
| 101   | 102 | 103 |     |
| 121   | 100 | 120 |     |

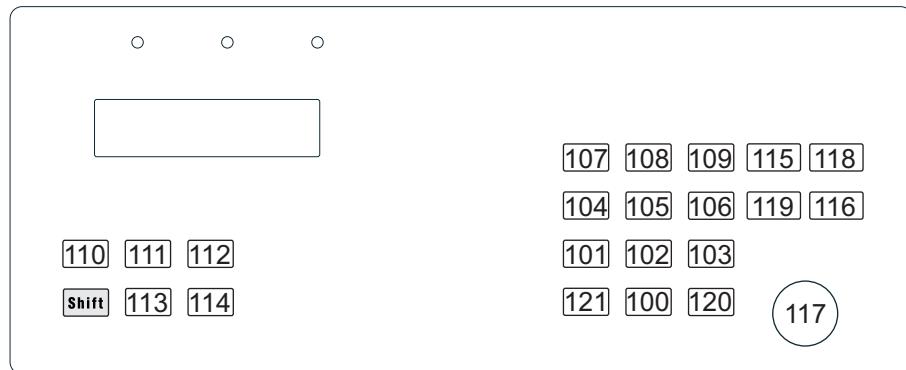
## PX4i, PX6i Standard Keyboard



**ID Numbers (Default)**



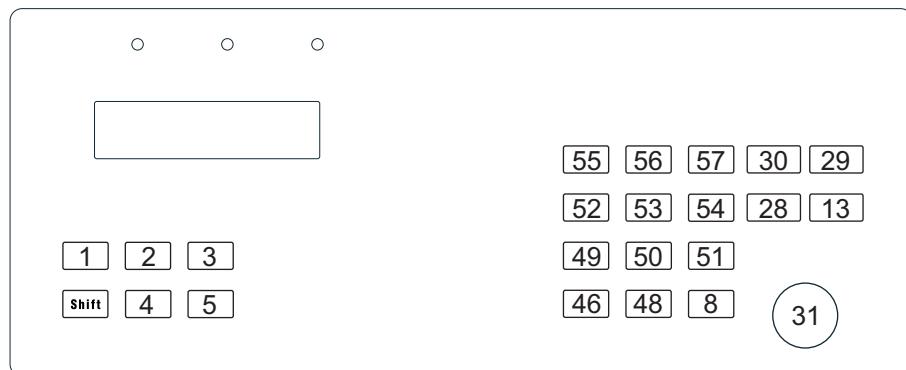
***ID Numbers (+ Shift)***

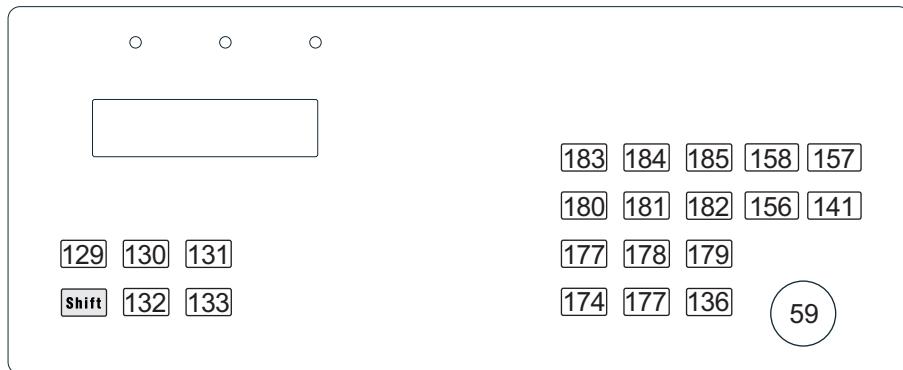
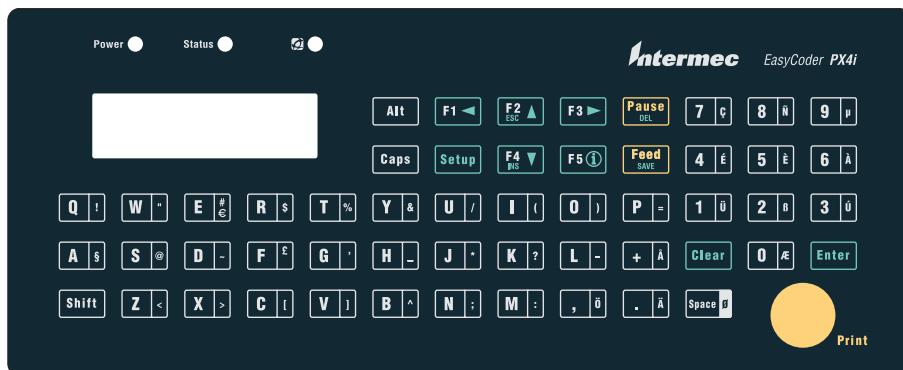
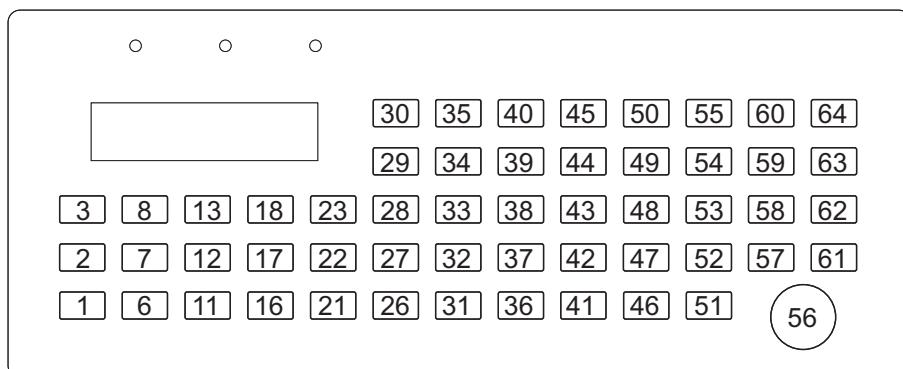


***Position Numbers***



***ASCII Values (Default)***



**ASCII Values (+ Shift)****PX4i, PX6i Alphanumeric Keyboard****Position Numbers****About ID Numbers and ASCII Values**

The PX4i and PX6i alphanumeric keyboard does not automatically map to ASCII or ID values. You need to manually map the keys to desired ASCII values.

ID numbers cannot be explicitly mapped, but follow the corresponding ASCII values. ID numbers exist only for the 22 keys (and their respective shifted keys) which are common to both the standard and alphanumeric keypads.

The alphanumeric keyboard must be remapped after every reboot according to the user's requirements. Use KEYBMAP\$ to assign ASCII values to each corresponding position number.

The example code maps the keyboard to the values shown on the actual keyboard according to the ANSI character set used by most terminal programs.

In order to map (or remap) the keyboard, you must identify the ASCII values for each character you want available. The KEYBMAP\$ command requires a string containing all 64 keyboard characters. You must map the complete keyboard for two (three in PX printers with alphanumeric keyboard) cases: no Shift key pressed, and Shift key pressed (plus Alt key pressed for PX printers).

This example code maps the alphanumeric keyboard to the actual alphanumeric keyboard appearance.

```
10 REM PX Alphanumeric keyboard mapping.
20 REM Character Set = ANSI (NASC -2)
30 REM One keyboard column per program row
40 NASC -2
90 REM Unshifted keys
100 A$ = CHR$(128)+"aq" + STRING$(2,0)
110 A$ = A$ + "zsw" + STRING$(2,0)
120 A$ = A$ + "xde" + STRING$(2,0)
130 A$ = A$ + "cfr" + STRING$(2,0)
140 A$ = A$ + "vgt" + STRING$(2,0)
150 A$ = A$ + "bhy" + CHR$(6)+CHR$(7)
160 A$ = A$ + "nju" + CHR$(29) + CHR$(1)
170 A$ = A$ + "mki" + CHR$(4) + CHR$(2)
180 A$ = A$ + ",lo" + CHR$(5) + CHR$(3)
190 A$ = A$ + ".+p" + CHR$(28)+CHR$(30)
200 A$ = A$ + " " + CHR$(8) + "147"
210 A$ = A$ + CHR$(31)+"0258"
220 A$ = A$ + CHR$(13) + "369"
290 REM Shifted keys
300 B$ = CHR$(0)+"AQ" + STRING$(2,0)
310 B$ = B$ + "ZSW" + STRING$(2,0)
320 B$ = B$ + "XDE" + STRING$(2,0)
```

```
330 B$ = B$ + "CFR" + STRING$(2,0)
340 B$ = B$ + "VGT" + STRING$(2,0)
350 B$ = B$ + "BHY" + CHR$(0)+CHR$(0)
360 B$ = B$ + "NJU" + CHR$(157) + CHR$(129)
370 B$ = B$ + "MKI" + CHR$(132) + CHR$(130)
380 B$ = B$ + ",LO" + CHR$(133) + CHR$(131)
390 B$ = B$ + ".+P" + CHR$(156)+CHR$(158)
400 B$ = B$ + " " + CHR$(136) + "147"
410 B$ = B$ + CHR$(159)+"0258"
420 B$ = B$ + CHR$(141) + "369"
```

## About the PD41

The EasyCoder PD41 keyboard has only one button, with Id. number 17 when pressed and 117 when released, giving the corresponding ASCII values of 16 and 144. The keyboard map string is only six characters long (compared to 64 for PF/PM/PX printers), with the Print button assigned position 1.





**Worldwide Headquarters**  
6001 36th Avenue West  
Everett, Washington 98203  
U.S.A.

**tel** 425.348.2600

**fax** 425.355.9551

[www.intermec.com](http://www.intermec.com)

Intermec Fingerprint v8.71 Programmer's Reference Manual



P/N 937-005-001