ZENGRANGE

Zenwand – 71

Owner's Manual

ZENWAND-71

Owner's Manual

A Barcode Reading, Analysing and Printing System
for the HP-71 Handheld Computer

September 1985

CONTENTS

Chapter 1

INTRODUCING ZENWAND

Introduction

The Zengrange ZENWAND has been designed for the Hewlett Packard HP-71 Handheld Computer to provide a completely portable bar coding system using the Hewlett Packard HBCS-2200 Barcode Reader wand. The product has been further enhanced by a powerful, but uniquely friendly operating system, designed by Zengrange Ltd. Not only does this operating system cope with current requirements, but it is also capable of being further extended as new barcode decoders and printer types become available.

As delivered, ZENWAND decodes the following ten kinds of barcode:

- **Codabar** (USD-4)
- **Code 11** (USD-8)
- **Code 3-of-9** (USD-3)
- **International Article Number Codes**

includes:	**European Article Number:**	**EAN-13**
		EAN-8
	Universal Product Code:	**UPC-A**
		UPC-E
		UPC-E(1)

- **Industrial 2-of-5**
- **Interleaved 2-of-5** (USD-1)

Each barcode type read by ZENWAND is decoded by string functions, i.e. the barcode is returned as a text string. These functions permit the returned string values to be processed directly by the user's own application programs, stored in files in memory, transferred via HP's Interface Loop (HP-IL) to mass storage media or downloaded to host computer systems via IL-modems and IL-interfaces.

The ZENWAND system is further enhanced by the inclusion of functions for:

- automatic discrimination of barcode type,
- check digit verification,
- barcode analysing, and
- printing of barcodes on the HP-2225B ThinkJet Printer.

How to use this Manual

This manual describes how to install and operate the ZENWAND Barcode Reading, Analysing and Printing system. Because of the very nature of the product, in which usage depends upon an individual's own specific requirements, this manual has been written primarily as a reference for application programmers and not as a tutorial. The examples, with emphasis on programmatic use of ZENWAND commands, have been written for the knowledgeable user who is well versed in HP-71 BASIC language programming, the operation of the HP-71, and understands how to use barcode data and what it is for.

The bulk of the applications written for ZENWAND will probably take advantage of the versatility of the HP-71 Handheld Computer, its powerful BASIC language and communications ability with external HP-IL devices. Because system configurations are so variable, it has not been possible to include complete applications programs in this manual. Instead, short examples have been given to explain and illustrate the application of individual ZENWAND commands. Whilst we have assumed a detailed understanding of barcode reading technology, additional reference information has been included where this would aid the application programmer's task.

Although it may take some time to become fully proficient with the concepts and commands presented here, we believe that by experimenting with the commands and example routines even a newcomer to barcodes could very quickly design a program for a specific barcoding application.

The End User. It is unlikely that users will ever need to concern themselves with an understanding of barcode symbology or the keywords available in ZENWAND. The applications programmer will have produced a user friendly system that handles this for them. Users will therefore only need to read the following ZENWAND Owner's Manual sections:

Chapter 1: Introducing ZENWAND
Introduction
How to use this Manual
Installation Procedures

Chapter 3: Reading Barcodes with ZENWAND
Scanning a Barcode Label

Appendix A: Owner's Information
Maintenance
Replaceable Wand Parts
Warranty Information

Where Users are also involved in printing of barcode labels, they should also refer to:

Chapter 5: Printing Barcodes
Examples Using the HP-2225B ThinkJet Printer

Installation Procedures

CAUTIONS

* Do not place fingers, tools, or other foreign objects into any of the HP-71 ports. Such action could result in minor electrical shock hazard and interference with pacemaker devices worn by some persons. Serious damage to port contacts and internal circuitry could also result.

* Before installing or removing ZENWAND, or any other module, be sure to turn off the HP-71 computer by pressing [f],[OFF].

* If removing a RAM-Module to make a port available, you should execute:

  FREE PORT (*port number*)

  in order to free the port memory - This is necessary to reset internal pointers in the HP-71. Failure to do so may cause a loss of HP-71 memory contents when you remove the Memory Module. See also: *HP-71 Owner's Manual, Pages 105 to 106.*)

* ZENWAND can only be inserted one way into the HP-71. Do not try to force it into a port as this could damage contacts in either or both devices.

ZENWAND has been designed to fit into the front ports on the HP-71, into which you may also plug memory or application modules. For extra stability and trouble free placement of the wand cable, the plug has been shaped to fit into Port 2 (the second port from the front left-hand corner of the HP-71). You may, however, also install ZENWAND into Port 4 (the port closest to the front right-hand corner of the HP-71).

To insert ZENWAND, hold onto the plug casing and orient it so that the plug contacts are facing away from you with the wand cable leaving the plug on the left. Hold the HP-71 with the keyboard facing uppermost, and gently push the plug into Port 2 (the second slot from the left).

To remove ZENWAND, first observe the precautions above, then hold onto the ZENWAND plug and gently pull this straight out of the port. Do not attempt to remove ZENWAND by pulling the cable, but rather always pull by holding onto the plug. Install a port cover in the empty port to protect the contacts inside from dirt and debris.

Chapter 2

INTRODUCING BARCODES

Development of Barcodes

This explanation is not intended to be a full thesis on barcode technology, merely a very brief outline of the subject.

For the average person, their first encounter with barcodes will be noticing rows of bars and numbers on food packaging in the supermarket, and the possible use of these for automatic pricing of shopping at the checkout desk. However, with every product having its own unique "identity number", it is obvious that the system can be applied all the way through production and distribution for greater speed and accuracy in stock control, ordering and manufacture.

Although readily understood by the human brain, machine reading of characters is still difficult with the required speed and reliability essential in most applications. By generating an easily reproducible pattern of contrasts between dark bars and light spaces on a label or product, the computer can be programmed to see a specific pattern as representing a certain number. By programming the computer to respond to the ratio of bar and space widths, barcodes can even be reproduced at different magnifications, although this does imply quite strict control over the reproduction and printing.

Barcoding really began in 1973 in America with Universal Product Coding (UPC), which established a "numbers bank" from which manufacturers were allocated a "company number" to precede their own product identity numbers. In 1977 agreement was reached on the specification for a European Article Number (EAN) system within which individual European Countries could set up their own article numbering systems. Since its conception, EAN has been increasingly adopted by more countries throughout the world and is thus becoming the *de facto* international system for product numbering. It has also been ingeniously married to the long established International Standard Book Numbering system, a development (like the ISBN itself) pioneered by the United Kingdom book trade. Because both EAN and UPC systems are basically the same, the acronym IAN (International Article Number) is increasingly being adopted to refer to both UPC and EAN. In designing ZENWAND, and in writing this manual, we have followed that practice.

In addition to the International Article Number codes, other codes have been developed for industrial, book classification, and other specific manufacturer/product applications.

Because of the limitations of pure numeric coding systems, alphanumeric codes were also developed.

Barcode Symbology - An Introduction

Barcodes, in the case of ZENWAND, are read by the passing of ZENWAND over the barcode symbols. To see the symbols, ZENWAND emits a red light that is reflected from the surface of the barcode medium back into the ZENWAND. This pattern of reflected light is then decoded by ZENWAND into data bits according to the decoder active at that time. The encodation of binary (logic values '0' or '1') data bits into a bar and space pattern is known as the barcode symbology.

Nearly all barcodes are modular in construction. That is, each character is made up of a number of vertical strips or known data elements which are the same width and are known as modules. One common exception is CODABAR, where the data elements are of differing widths.

While there are numerous types of barcode systems employed in different areas of commerce and industry, the symbology used falls into two main categories;

- **MODULE WIDTH** encoding - most industrial codes
- **NON-RETURN-TO-ZERO** (NRZ) encoding - commercial codes.

In **module width encoding**, logical value zero is represented by a narrow bar or space; while data with a logic value of one is represented by a wide element whose width is typically two or three times that of the narrow element. hand (variable speed scan) scan.

The **NRZ encoding** technique is used in the International Article Numbering systems of Universal Product Code (UPC) and European Article Number (EAN). IAN codes encode binary data in the reflectivity of the bars and spaces, i.e. the logical '0' is represented as a reflective surface, and the logical '1' as a non-reflective surface. Note that there is no transition between bits unless the logic state changes. Thus a binary sequence of 1's or 0's may be represented by the width of a single printed element. **FIGURE 1** shows the characteristics of both module-width and NRZ encoding.

NRZ is not self clocking.

The process to convert a computer message into a barcode symbol is a simple four-stage process. The sequence begins with establishing the type of data to be represented and the number of characters in the message.

FUNCTION	MODULE WIDTH CODING	NRZ CODING
WAND OUTPUT		
LOGIC LOW "0"	REFLECTIVE (WHITE)	REFLECTIVE (WHITE)
LOGIC HIGH "1"	NON-REFLECTIVE (BLACK)	NON-REFLECTIVE (BLACK)
BINARY DATA ENCODATION		
LOGIC LOW "0"	NARROW ELEMENTS	REFLECTIVE (WHITE)
LOGIC HIGH "1"	WIDE ELEMENTS	NON-REFLECTIVE (BLACK)
MESSAGE/CHARACTER ENCODATION		
	SEQUENCE OF NARROW 0 & WIDE 1 ELEMENTS	WIDTH OF BLACK & WHITE ELEMENTS

EXAMPLE

CHARACTER = 110001

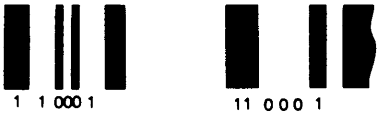


FIGURE 2.1: Barcode Conventions

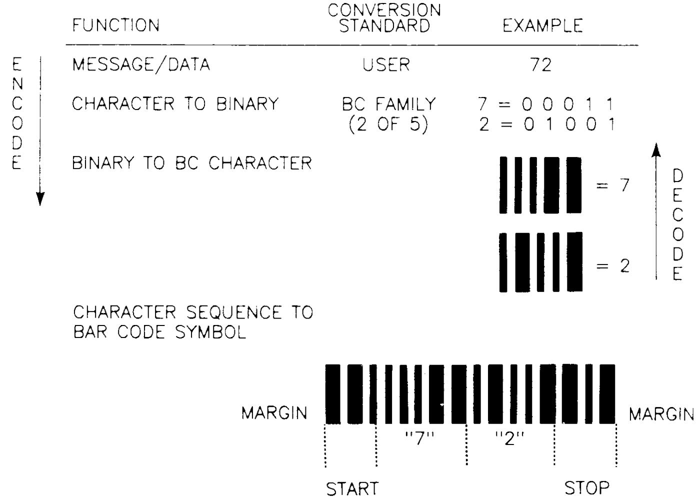
The second step is the translation of the human-readable information into a binary sequence. The number and value of the binary data bits are determined by the particular barcode symbology selected. **FIGURE 2.2** shows the character "72" being translated into the binary sequence prescribed by the **2 OF 5** barcode family.

The third step is the creation of the bar and space pattern that represents the binary word defined in step 2. In the Industrial 2 of 5 code, a narrow bar represents a logic 'zero' and a wide bar represents a logic 'one'

The last step is to format individual barcode characters into a symbol representing the whole message. This will consist of start/stop margins, start/stop character patterns, the data or message itself, and an optional checksum character. Start/Stop margins are the quiet zones, typically white, and are devoid of all printed characters or bars. This establishes a null or reference field against which barcode scanners can compare the reflected pattern. Start/Stop characters, on the other hand, are specific patterns that precede the first message character or follow the last message or checksum character. Checksums are often a summation of the preceding digits in the message and offer a check that the barcode symbol has correctly been scanned.

The physical length of the symbol is determined by:

- the number of characters,
- whether a checksum is included in the data,
- the number of printed modules needed to represent a character,
- the number of modules used in stop/start patterns, and
- the resolution-width of a narrow element.



IND 25
72

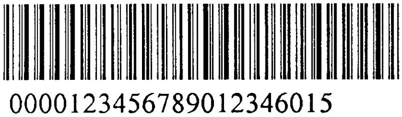
FIGURE 2.2: Symbol Encode-Decode Sequence

Types of Barcodes

CODE 11

CODE 11 is a discrete, numeric barcode similar to one of the 2 of 5 Code family (Matrix 2 of 5). Its symbology consists of 11 defined characters: numeric characters (0-9) and one special character (-), plus a special twelfth stop/start character.

Each character is encoded in five binary bits, three bars and two spaces. Unlike Matrix 2 of 5, only nine characters (1-8 and start/stop) have two wide elements out of five and the other three characters (0,9,-) have only one wide element out of five. This means that Code 11 is not self-checking and that implementing a checking algorithm would be very difficult and memory intensive to apply to each character. Although the theoretical density is 15 characters per inch, the lack of self checking, combined with a code structure that allows one printing defect in a character to result in a substitution error, means that one, or preferably two, checksum characters must be added, thus reducing the effective density. However, because of this discrete high density, Code 11 is often employed for barcode marking on printed circuit boards and other locations where space is at a premium.



CODE 39

CODE 39 (also called **CODE 3 OF 9)** is the most popular alphanumeric barcode in use. It employs 36 defined numeric and uppercase alphabetic characters (0-9,A-Z), plus seven special characters (-% .$/+), and a stop/start character (*). With Code 39, both bars and spaces are width-modulated to encode the logic values of the nine binary bits of data. A logic '1' is encoded as a wide element, while logic '0' is encoded as a narrow element. Individual characters are separated by an intercharacter space; thereby making this symbology a discrete type. See **FIGURE 2.3**.

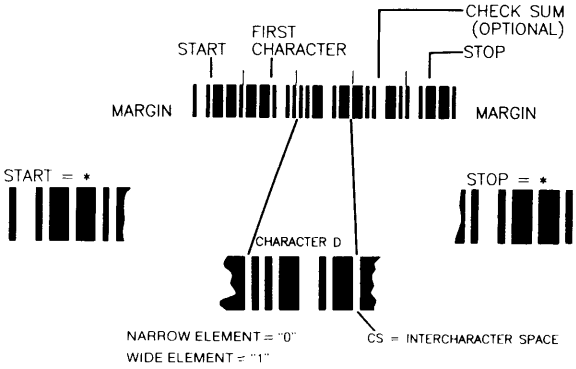
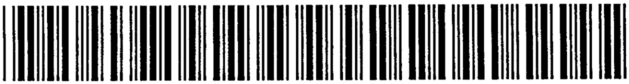


FIGURE 2.3. 3 of 9 Message/Character structure

By defining each character (using either bars and/or spaces), as consisting of three wide elements and six narrow elements, it is possible to define an easy self checking algorithm. Information density is determined by the number of modules per character, width resolution of the narrow element and the wide to narrow element ratio. Typically the following resolutions are used:

High resolution:	9.8 characters per inch
Medium "	5.2 c.p.i.
Low "	3 c.p.i.

An optional checksum character may also be added at the end of a Code 39 message to provide verification that the correct number and type of data is present.



SN-2411A00264

CODABAR

CODABAR is a discrete, width-modulated code that provides for encoding numeric data (0-9) plus six special characters. In addition, four different sets of stop/start characters can, for example, be used as the key to different databases. Although it is also sometimes called 'Code 2 of 7', this is incorrect, because Codabar is composed of both 2 of 7, and 3 of 7 symbology.

Codabar has seven binary bits of information encoded in the bars and spaces of each character. But, unlike other width-modulated codes, Codabar does not use common wide and narrow element widths to encode the logic 1's or 0's in the characters. Instead, a total of 18 different widths for bars and spaces are specified by the symbology. Although originally designed this way to cope with printing errors, it does also provide a constant character length, regardless of whether two or three wide elements are used in that particular character. See **FIGURE 2.4**.

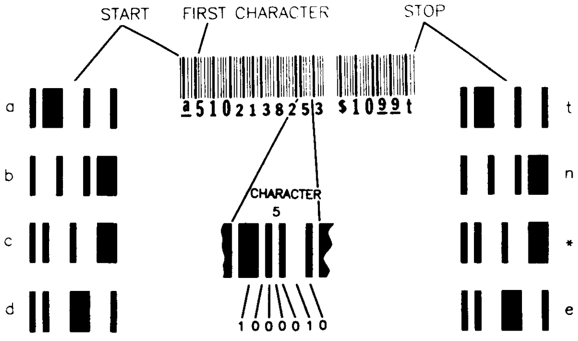
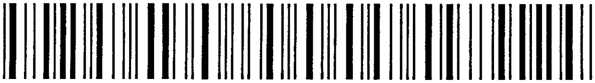


FIGURE 2.4: Typical Codabar Symbol

Codabar is commonly printed at high resolution, providing a density of 11 characters per inch. A lower resolution is sometimes also used, giving a density of 9 characters per inch.



a:10932001047:t

won't Read

2 of 5 Barcode family

The structure of the 2 of 5 barcode family is one of the simplest of the width-modulated industrial codes. The three best known members of this family; Industrial, Interleaved and Matrix have the following similarities:

- two wide elements per five-element character;
- black bars and white spaces;
- numeric characters (0-9);
- binary encoding: wide =1; narrow =0. Wide element is typically two to three times wider than narrow;
- non-character stop/start, bar/space pattern;
- even-parity character check;
- optional message checksum character;

All of these 2 of 5 codes use five binary elements to encode each character. Two of the elements are logic 1's and, depending upon symbology, are printed as wide bars and/or wide spaces. Consistent use of two wide elements assists with error checking. ZENWAND contains decoders for the two most common 2 of 5 codes; Industrial, and Interleaved.

IND 25

Industrial 2 OF 5 is the oldest member of the 2 of 5 family. Characters are represented by five printed black bar elements, separated by interelement spaces. These five elements create a discrete character used to encode the five binary bits that represent the message character. By employing intercharacter spaces each character becomes discrete - hence the reason that all symbologies employing intercharacter spacing are called "discrete codes". **FIGURE 2.5** shows the structure of IND 25 barcode.

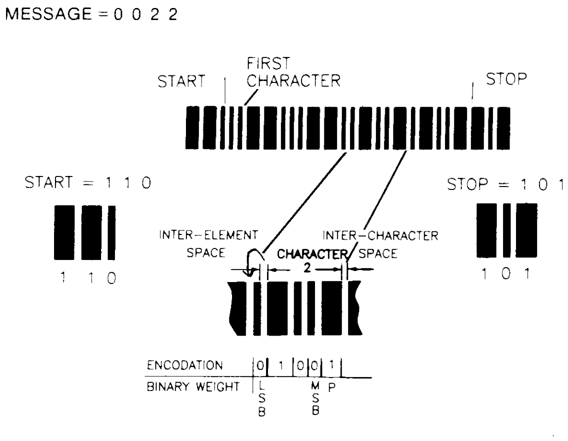


FIGURE 2.5: Industrial 2 of 5 Message/Character Structure

IND 25 characters consist of five bars (three narrow and two wide), four interelement spaces and one intercharacter space.

INT 25

Interleaved 2 OF 5 is of higher density than either IND- or MATRIX-25 symbology, because the intercharacter space is eliminated. (MATRIX-25 being half-way between the two, using encodation within the spaces, but retaining the intercharacter spaces.) INT-25 avoids the intercharacter space by interleaving characters encoded in the bars with characters encoded in the spaces (**FIGURE 2.6**). The first message character at the left is encoded into bars immediately following the start character; while the second message character is encoded into the spaces separating the bars in the first character, thereby eliminating the need for an inter-character space. Because of this it is also termed a "continuous" code. Because an even number of characters (including checksum) is needed, a leading, non-significant zero may need to be added.

By means of the interleaving, a more dense code is obtained; typically 40% more dense than IND 25 and 12% more than MATRIX.

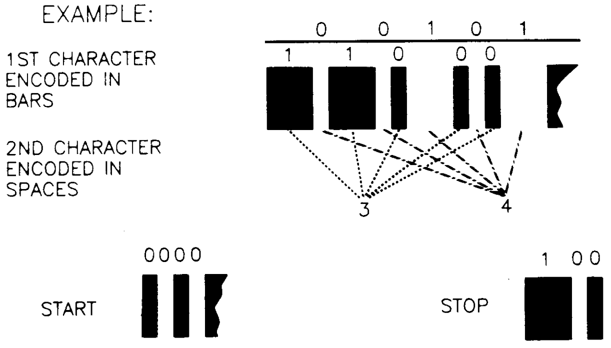
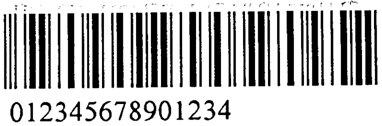

EXAMPLE:
0 0 1 0 1
1ST CHARACTER ENCODED IN BARS
1 1 0 0 0
2ND CHARACTER ENCODED IN SPACES
3
4
0000
START
1 00
STOP

FIGURE 2.6: Interleaved 2 of 5 Encodation



= 0
10001 = 1
01001 = 2
00101 = 4
10100 = 5
01100 = 6
10010 = 8
01010 = 9

IAN-Code

International Article Number (IAN) is the name increasingly being adopted to cover both the **Universal Product Code (UPC)** and the **European Article Number** (EAN). Because they are both basically the same, ZENWAND has been designed with a decoder common to both symbologies. The IAN symbologies were originally designed for the main requirements of product identification in computerised manufacturing, warehousing, distribution and retailing environments.

IAN codes are fixed length numeric only and may represent a 12 or 6 digit UPC number, or a 13 or 8 digit EAN number. They consist of four main elements; a prefix, the manufacturer's number, the item reference number and the check digit.

The common elements between EAN and UPC systems are the manufacturer's number (assigned by a national regulatory body from a numbers bank), the item reference (assigned by the holder of the manufacturer's number to identify individual products) and verification checksum.

The difference between the codes is in the prefix, where EAN has an extra digit. UPC was designed for the US-domestic market, so did not specify a country, and uses a single digit for product category (0=grocery, 2=pharmaceuticals, etc). But because EAN was introduced with the specific intention of it becoming an international standard, a two digit prefix was defined to identify the country of origin. Because of this difference many US-scanning devices will only read UPC-codes, while European devices, including ZENWAND, are quite happy with both.

The most common UPC codes, are the 12-digit **UPC-A** and the 6-digit **UPC-E**. In non-retail situations, the UPC-E system can also be found with the number system digit equal to zero. Such codes are commonly referred to as **UPC-E(1)**. The other UPC types; -B, -C, -D, are not even widely used in the USA. IAN codes also contain start/stop characters called guard bars and a special centre guard pattern.

The EAN codes are the 8-digit **EAN-8** or the 13-digit **EAN-13**

A difference that must be noted in dealing with ZENWAND and IAN codes, is that where most barcode reading systems return only 12 digits from the 13 digit encoded strings of UPC-A and EAN-13, ZENWAND returns the complete 13 character string. With UPC-A, the first digit will always be 0 as this is implied by the defined symbology.

EAN 13

EAN 13 encodes 13-digits (including a mandatory checksum), twelve of which are encoded explicitly and the leading digit implicitly. Because UPC-A is a subset of the EAN-13 code, there is compatability between the two. UPC-A codes are simply returned with an implicit leading digit of zero.

EAN 8

EAN 8 (also known as 'EAN S-version') is not really a zero-suppressed form of EAN-13, as UPC-E is of UPC-A code. Neither is it compatible with the UPC-E or UPC-E(1) ranges. It is however, an eight digit short form code consisting of a country flag (2 or 3-digits), 4 or 5 digits of data (e.g. a manufacturer and product item number) and lastly a one-digit checksum.

UPC-A

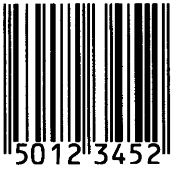
UPC-A is a 12-digit barcode with all digits explicitly encoded, making this the easier to decode. Because the leading digit is always '0', UPC-A is actually a special case, or subset, of EAN-13. The two are to all intents compatible. During the encodation process, different number sequences are used for the left and right halves of the barcode; thereby defining the centre, so that automated scanners can detect a forwards or backwards scan.

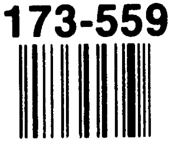
UPC-E

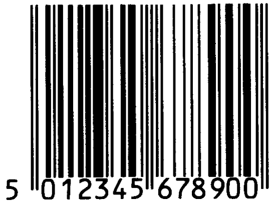
UPC-E is an 8-digit symbol, 6-digits obtained by zero-suppression of the UPC-A code, and the number system and check digits being encoded implicitly. Zero-suppression of UPC-E requires that there be sufficient zeros at specific locations in the 12-digit number and that the number system (the first digit) is '0'. It also requires that the check digit be calculated before zero suppression. Upon scanning, the 8-digits can be expanded by reinsertion of the zero-digits.

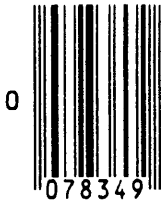
UPC-E(1)

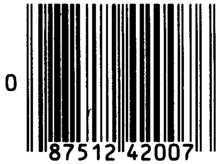
UPC-E(1) is basically the same as UPC-E, except that the number system digit, the first digit, has been given the value '1'. Number system 1 is really unassigned, but many companies are using this for non-retail applications. Unlike many barcode readers, ZENWAND does decode UPC-E(1).











Barcode Selection Summary

Various barcode systems have been introduced to industry; some have become standards, while many have fallen into disuse. Those presented here represent the vast majority of barcode usage today, and as the need for standardisation increases, more and more organisations will accept the shortfallings that these symbologies may have, against the benefits obtained by standardisation. For a person having to design, select and implement a barcoding system, the most dominant selection criteria will be:

1. The type of data to be encoded, and
2. The information density of the symbology.

If the data to be encoded is alphanumeric, then the most common choice is CODE 3 of 9. However, when only numeric data is to be encoded, one of the 2 of 5 family is normally preferable. For product numbering, the IAN range should be used.

The user's application will normally dictate the message length and the physical area available, thus determining the information density requirements. Within limitations, the density for a specific symbology can be changed by adjusting the module resolution and the wide-element to narrow-element ratio.

Once the symbology has been selected, the user must address the mechanics of barcode generation. This involves quite critical attention to both the printer and the medium on which the barcode is to be printed.

Characteristic	**3 of 9 Code**	**Industrial 2 of 5**	**Matrix 2 of 5**	**Interleaved 2 of 5**	**Codabar Code**	**Code 11**
Character Set	Alpha-numeric	Numeric	Numeric	Numeric	Numeric	Numeric
Number of Characters[1]	43	10	10	10	16	11
Number of Bits per Character	9	5	5	5	7	5
Number of Element Widths Used	2	2	2	2	18	3
Information in both Bars and Spaces	Yes	No	Yes	Yes	Yes	Yes
Discrete (Independent Characters)	Yes	Yes	Yes	No	Yes	Yes
Self-Checking	Yes	Yes	Yes	Yes	Yes	No
Checksum Character	Optional	Optional	Optional	Optional	None	Recom-mended

Note:
1. Not including start and stop characters.

TABLE : Summary of Barcode Characteristics

Chapter 3

READING BARCODES WITH ZENWAND

Because applications for ZENWAND vary so widely, both in requirements and system configurations available, ZENWAND has been designed for ultimate flexibility by providing all the commands necessary for users to write application programs tailored to their own specific requirements. Although the ZENWAND barcode system provides features to cover most applications, considerable thought has gone into making the system expandable by accepting additional barcode decoders that can be made available in plug-in modules and on mass storage media.

ZENWAND allows the user to read barcodes in two ways:

1. By providing functions that await a scan and then return a string representation of the label scanned.
2. With a "live wand" mode that can be used whenever the keyboard is active and the HP-71 is waiting for keyboard entry.

Scanning a Barcode Label

In most cases, to scan a barcode label, ZENWAND needs to be told the type of label to be decoded (automatic discrimination will be explained later). If ZENWAND can decode the scanned barcode, it will return the scanned data as a string and confirm the good scan with a high beep tone. If the scanned barcode cannot be decoded, or the current **time-out** period is exceeded, ZENWAND will signify a bad scan by returning a null string and emitting a low tone. For some barcode symbologies, if check digit verification (**CDIGIT ON**) is active, labels without check digits will not be decoded and will cause a bad scan error.

Here is the recommended procedure for scanning a barcode:

1) Connect ZENWAND to the HP-71, and turn the computer on.
2) Execute the keywords for the type of label you are scanning, e.g: **CODE39$** for Code 39, etc.
3) Hold the ZENWAND in a similar manner to how you would hold a pen and press the switch near the tip.
4) Position the tip of ZENWAND into the quiet zone (the white area either side of the label) and hold the Wand at an angle between 0 and 30 degrees from the vertical.

5) Scan the entire label smoothly, making sure that the speed of scan is between 7.6 cm/sec (3 in/sec) and 76 cm/sec (30 in/sec) and that the wand does not go outside the quiet zone boundary around the barcode. It is not necessary to bear down on the wand. Applying excess pressure during scanning reduces reliability and may smear or damage the barcode.

6) Release the switch on the ZENWAND to conserve battery power.

During running of an application program, step 2 will be handled by the program.

Time-Out. To conserve battery power, ZENWAND barcode scanning functions will time-out, if a scan has not begun within approximately 2.5 minutes (default setting) after the reading function began execution. Should the user not begin the scan before the time-out, then a null string is returned, a low tone (bad scan) will be emitted, and the Wand status indicator byte (**WANDSTAT**) will indicate a time-out error.

The time-out setting may be controlled by means of the **WANDTIME** command, e.g.:

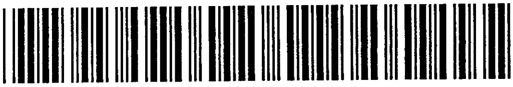
WANDTIME INF	Sets time-out to infinity
WANDTIME (A-TIME)	Sets to value calculated by the formula (A-TIME)
WANDTIME 30	Sets time-out to 30 seconds.

Using Audible and Visual Prompting

In using most applications programs, the user will need both audible and visual prompting to know when to scan a barcode label, etc.. The barcoding functions in ZENWAND itself do not supply prompting, this being the job of an applications program. The HP-71 commands **BEEP** (for audible prompts) and **DISP** (for visual prompts) provide this feedback during barcode reading. See the *HP-71 Owner's* and *Reference Manuals* for details of these commands. ZENWAND does, however, automatically provide audible confirmation of good or bad scanning.

Delay Settings. The prompting done by a program and the scanning done by the user must be synchronised. A subtlety of the **DISP** statement is its behaviour with the current **DELAY** setting. Whenever a **DISP** statement is encountered, program execution waits for the **DELAY** time before continuing. The following program is an example of bad use of DELAY, and the considerable frustration that this can give to the user while scanning labels:

10 DELAY 3	Sets delay of 3 seconds
20 DISP "Scan Label"	Prompts for input
30 A$=CODE39$	Decodes Code 39 label
40 IF A$="" THEN 20 ELSE DISP A$	Repeat if no input, otherwise display value.



CODE 3 OF 9

Because the user can begin scanning as soon as the "Scan Label" prompt appears, even though the 3 second delay is still in progress, the user may find that numerous scans are necessary before ZENWAND appears to work. Should the **DELAY** end midway through a scan, ZENWAND will respond with a **bad scan tone**. For the user, this may appear that the whole product is bad, but in fact the problem is one of bad design of the application program and will be cured by setting a short **DELAY** time, e.g. **DELAY 0**.

Beeps for Good or Bad Scans. After each completed scan of a label, ZENWAND responds with a scan tone. The frequency and duration of this indicates whether the label was successfully (good scan = **high short tone**), or unsuccessfully (bad scan = **low, longer tone**.) read. The tones can be suppressed by executing the HP-71 command **BEEP OFF** and the beep volume may be adjusted by setting or clearing flag **-25** (**SFLAG -25** or **CFLAG -25**).

String Functions to Read Barcodes

Barcode readings can be obtained by use of string functions provided by ZENWAND. These functions wait for a scan to occur, pass the data read by ZENWAND through a decoder, and then return a string representation of the barcode data for subsequent processing by the application program. Note that all user prompting must be supplied by the application program.

Because ZENWAND is capable of reading various different types of barcode, six separate string functions are provided:

CODE11$	Reads a Code 11 barcode
CODE39$	Reads a Code 39 barcode
CODABAR$	Reads a Codabar barcode
IND25$	Reads an Industrial 2 of 5 barcode
INT25$	Reads an Interleaved 2 of 5 barcode
IANCODE$	Reads an International Article Number barcode either: EAN-8 EAN-13 UPC-A UPC-E UPC-E(1)

All the barcode reading functions alter the values of **BARTYPE?** (the last decoder used) and **WANDSTAT** (Wand status indicator) according to the result of the scan and are therefore affected by settings for:

ENDSCAN	Selecting a key sequence to terminate a scan;
WANDTIME	Setting a value for time-out of Wand operation;
Attention Disable	Using the command **POKE** to disable the [ATTN] key.

Certain barcodes define optional check digit(s) as the last character(s). Verification of such check digits is optional, but if enabled (**CDIGIT ON**), then the barcode scanned must contain the required check digits to prevent a bad scan occurring. Barcode types with a check digit defined are indicated below. (See also: *Using Check Digit Verification*.)

A sample barcode of each type scanned by ZENWAND is given against the descriptions below. By changing program line 40 to suit the particular barcode type to be scanned, the following short program can be used to read all the example barcodes given.

```
10 DELAY 0                            Sets delay to zero seconds;
20 DISP "Scan Label..."               Prompts for input by scanning;
30 BEEP 1800,.5                       Beeps at 1800Hz, for 0.5 sec;
40 A$=CODE39$                         Reads and decodes a Code 39 label.
                                      Change: A$=string function to suit the
                                      desired decoder;
50 IF LEN(A$) THEN DISP A$ @ WAIT 1   If a good scan, then display value & wait;
60 GOTO 20                            Repeat scan.
```

Execution of a barcode reading function can be terminated by:

1) **Performing a scan.**

   Valid Scan: A successfully completed scan returns a string value representing the label data and causes the HP-71 to emit a good scan tone.

   Invalid Scan: An unsuccessfully completed scan, due to bad scanning technique, damaged barcode label, etc., will return a null string and emit the bad scan tone.

   Scanning a non-check digit label with CDIGIT ON: If check digits are defined for a particular barcode symbology, and the **CDIGIT ON** feature is set, then scanning a label with a bad or missing check digit(s) will return a null string and instigate a bad scan tone. If the selected barcode type does not define a check digit, then the setting of **CDIGIT ON/OFF** is ignored.

2) **Hitting [ATTN]** (Attention key)

   If the [ATTN] key has not been disabled, by means of the **POKE** command, then hitting [ATTN] will abort function execution, suspend any currently running program and return control to the BASIC operating system. However, once a scan has begun, pressing [ATTN] will not cause the scan to terminate.

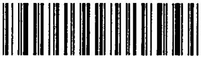
   If the [ATTN] key has been disabled, by means of the **POKE** command, then pressing the key has no effect. (See also: *ENDSCAN* for information regarding disabling the [ATTN]-key.)

3) **Waiting for the time-out period to expire**

   If a scan is not commenced within the time-out period (default 2.5 minutes) following function execution, then a "time-out" occurs, a null string is returned, and a bad scan tone is emitted.

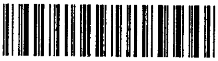
CODE11$

0123456789-



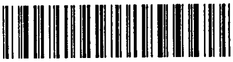
No check digit

0123456789-0



One check digit

0123456783-76



Two check digits

CODE11$ decodes a Code 11 type barcode in which the symbology allows for the definition of one or two optional checksum characters. Because Code 11 is prone to substitution errors, ZENWAND has been designed to verify two checks digits when the check digit verification feature (**CDIGIT ON**) has been set. When **CDIGIT ON** is set, the result of a successful scan, is for both of these digits to be stripped from the returned string. (See: *Using Check Digit Verification*). A scan will fail when the particular label scanned does not contain the proper check digits. If only one check digit is required to be validated, then the following short routine could be used:

```
10 CDIGIT  OFF                          Disable check digit verification;
20 B$=CODE11$                           Read barcode label;
30 IF  CDV11(B$) THEN                   Verify last check digit, if good
   B$=B$[1,LEN(B$)-1] ELSE DISP         then remove next digit. Otherwise
   "Check Digit Error" @ GOTO 20        display error message & re-scan.
                                        (See also: CDV-Functions.)
```

CODE 39
£sd6

CODE 39
ZENT

CODE 39
+$

CODE39$

£sd

ZEN

+$

CODE39$ decodes a Code 39 type barcode label. The symbology allows for the definition of an optional, one-digit checksum character. When **CDIGIT ON** is set, ZENWAND will automatically verify and delete that check digit from the returned string value.

The Code 3 of 9 barcode symbology has an extension, called Control 3 of 9, or Extended 3 of 9, that permits all 128 ASCII characters to be represented. This feature, included in ZENWAND, can be automatically controlled through the statement **EXPAND ON/OFF**. In default circumstances (**EXPAND OFF**), all Code 39 labels are automatically converted to full ASCII-representation. Manual, as opposed to automatic conversion, can be accomplished by using **CTRL39$**, or the inverse, by **NORM39$**.

CODABAR
b12+34-56/78e

CODABAR
a$2.95*

CODABAR
d01t

CODABAR$

b12+34-56/78e

a$2.95*

d01t

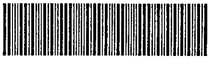
CODABAR$ decodes a barcode of the Codabar type. This type does not define check digits, as it is claimed to be self checking. By means of the four sets of stop/start characters, additional information can be encoded into the barcode, e.g. database type/number. Unlike other symbologies, these stop/start characters are returned as part of the scan. Although the barcode patterns for the four start characters are identical to those for the four stop characters, their position determines the returned characters. ZENWAND returns these as a/t, b/n, c/*, d/e.

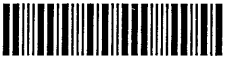
IND25
01234567895

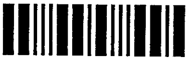
IND25
01236

IND25
01

IND25$


0123456789


0123

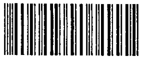

01

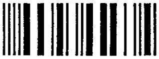
IND25$ decodes an Industrial 2 of 5 type barcode. The symbology defines one optional check digit as the last digit. If **CDIGIT ON** is set, ZENWAND verifies and deletes this check digit from the returned string. Because of its definition, a partial read of IND25 may be mistaken for a complete read. Most applications of IND25 use labels of a specific length in order to overcome this possibility. The HP-71 function **LEN** (return string length) can be used to eliminate this difficulty:

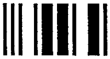
```
10 INPUT "Length ?";L @ DIM  B$[L]      Dimensions variable as input;
20 DELAY  0
30 ON ERROR GOTO 100                    Sets up error trapping for 'L' greater than
                                        dimensioned variable;
40 DISP  "Scan Label..."                Prompts to begin scanning;
50 B$=IND25$                            Read IND25 label into variable B$.
60 IF  LEN(B$)#L THEN 100 ELSE          Report good scan if length equal to 'L',
   DISP  "Good Read"                    else go to error sequence.
...
100 DISP  "Barcode Read Error" @        Otherwise report bad scan messages &
     GOTO  25                           re-scan.
```

INT25
00123456789

INT25$


0123456789


0123


01

INT25$ decodes an Interleaved 2 of 5 type barcode. The symbology specifies an optional one-digit checksum, which can be verified and deleted from the returned string by setting **CDIGIT ON**. Because INT25 exhibits the same misread tendency as IND25, the **LEN** function can again be used as an eliminator.

IANCODE$

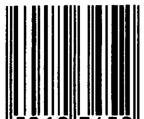
International Article Number Codes are automatically distinguished and decoded by ZENWAND into their subsets of: EAN-13, EAN-8, UPC-A, UPC-E, and UPC-E(1).

EAN-13

5 012345 678900

This 13-digit code includes a mandatory checksum as the last digit, which will always be validated by ZENWAND. If **CDIGIT OFF** is set, then the entire 13-digit string is returned. If **CDIGIT ON** is set, then the last digit is verified, and deleted from the returned string. Setting the EXPAND feature (**EXPAND ON**) has no effect on EAN-13 codes.

EAN-8



This is an eight-digit explicitly encoded type with a mandatory check digit as the last digit. The check digit is always validated by ZENWAND, but may be either returned, or eliminated from the returned string depending upon the setting of **CDIGIT**. **CDIGIT ON** deletes the check digit and returns 7 digits (or 12 with **EXPAND ON**), while **CDIGIT OFF** returns the complete 8-digits (or 13 with **EXPAND ON**). **EXPAND ON** acts by adding leading zeros to return a full form 13-digit code from a short-form EAN-8 code.

UPC-A

0
42000 06200

Although **UPC-A** is a 12-digit symbology, it is really a subset of EAN-13 with the 13th digit always '0' and therefore, mostly omitted. Care is needed with some UPC-A barcode readers as their decoders do not return this additional digit. ZENWAND will correctly decode UPC-A and EAN-13 codes and returns these as 13-digit IAN-Codes (with UPC-A having the leading digit set to '0'). Following the leading digit is a digit indicating the number system in use. The last digit is the mandatory checksum, which for UPC-A barcodes. is always validated by ZENWAND. By setting **CDIGIT ON**, this checksum digit can be removed and a 12 digit string returned. **CDIGIT OFF** returns the full IAN-Code of 13-digits. Reading UPC-A labels is not affected by the current setting of **EXPAND**.

UPC-E

UPC-E is a zero suppressed form of full UPC-A labels in which the number system ('0') and the mandatory checksum digit are implicitly encoded. When using UPC-E codes, it should be remembered that although most barcode readers return only 6-digits, ZENWAND returns the full eight. This enables greater control and manipulation of the returned string to take place under the application program. By setting the expand option (See: *Controlling the Barcode Decoders*), UPC-E labels will be expanded and returned in their non-zero-suppressed form of 13 digits. Because the check digit is mandatory, it will always be verified by ZENWAND. When **CDIGIT ON** is set, this checksum is omitted from the returned string and only 7 digits (or 12 with **EXPAND ON**) are returned. Setting **CDIGIT OFF** allows all 8 digits (or the full 13 when **EXPAND ON** is also set) to be returned.

UPC-E(1)

UPC-E(1) is a zero suppressed form of full UPC-A labels in which the number system (set to '1'), and the mandatory checksum digit are implicitly encoded. Although most barcode readers only return six-digits, when a UPC-E or UPC-E(1) label is scanned, ZENWAND returns the full eight-digits. By using **EXPAND ON**, the label can be returned in a non-zero-suppressed format of 13 digits. Because the check digit is mandatory, it will always be verified. It may be eliminated from the returned string by setting **CDIGIT ON**, in which case only 12 digits are returned.

Barcode Expansion Keywords

In addition to the six specific type functions, ZENWAND provides functions to control the manner in which the active decoder checks and decodes the scanned label.

EXPAND ON/OFF Selects automatic expansion for short form labels to full form, or automatic conversion between normal 3 of 9 and full ASCII-characters;

CTRL39$ Manually converts a normal Code 39 string to its full ASCII, or Control 3 of 9 representation;

NORM39$ Manually converts a full ASCII or Control 3 of 9 string to its normal Code 3 of 9 representation;

IANEXP$ Expands a short form EAN or UPC label;

IANSUP$ Suppresses zeros in a long form UPC label.

EXPAND ON/OFF

This affects only certain barcode types (CODE 39 and IANCODEs of those supplied with ZENWAND). Generally, if **EXPAND ON** is set, then the barcode read will be automatically expanded to a 'fuller' format.

For **CODE39$**, if **EXPAND ON** is set, labels are returned in their normal, expanded format. This is equivalent to automatically executing the function **NORM39$**. When **EXPAND OFF** has been set, a barcode label will be automatically contracted to full ASCII-character, or Control-39 representation (i.e. the function **CTRL39$** is automatically executed).

For **IANCODE$**, if **EXPAND ON** is set and short form labels of either EAN-8, UPC-E or UPC-E(1) are scanned, they will be returned in their expanded full 13-digit IANCODE format. This is the equivalent of automatically executing **IANEXP$**. If **EXPAND OFF** is set, then a short form IAN label will be returned as its normal 8-digit representation. The setting of **EXPAND** has no effect on decoding UPC-A or EAN-13 labels.

CTRL39$

The function, **CTRL39$**, converts a Normal Code 3 of 9 string to a Control 3 of 9 string and thereby allows the full ASCII 128-character set to be represented in the barcode. An "Invalid Arg" (invalid argument) error will be reported if characters are not defined in the Code 3 of 9 set. The equivalent result to that obtained by **CTRL39$** can also be obtained by setting **EXPAND OFF**. The character conversions from Code 3 of 9 to Control 3 of 9 are shown in **FIGURE 3.1.**

NORM39$

The function, **NORM39$**, converts a Control 3 of 9 string (with characters of the full ASCII 128-character set) into its Normal Code 3 of 9 representation. Should the barcode label contain characters not in the range 0 to 127, then an **"Invalid Arg"** (invalid argument) error will be reported. The equivalent result to that obtained by **NORM39$** can also be obtained by setting **EXPAND ON**. See also **FIGURE 3.1**.

ASCII	CODE 39	ASCII	CODE 39	ASCII	CODE 39	ASCII	CODE 39
NUL	%U	SP	Space	@	%V	`	%W
SOH	$A	!	/A	A	A	a	+A
STX	$B	"	/B	B	B	b	+B
ETX	$C	#	/C	C	C	c	+C
EOT	$D	$	/D	D	D	d	+D
ENQ	$E	%	/E	E	E	e	+E
ACK	$F	&	/F	F	F	f	+F
BEL	$G	'	/G	G	G	g	+G
BS	$H	(	/H	H	H	h	+H
HT	$I	)	/I	I	I	i	+I
LF	$J	*	/J	J	J	j	+J
VT	$K	+	/K	K	K	k	+K
FF	$L	,	/L	L	L	l	+L
CR	$M	—	—	M	M	m	+M
SO	$N	.	.	N	N	n	+N
SI	$O	/	/O	O	O	o	+O
DLE	$P	0	0	P	P	p	+P
DC1	$Q	1	1	Q	Q	q	+Q
DC2	$R	2	2	R	R	r	+R
DC3	$S	3	3	S	S	s	+S
DC4	$T	4	4	T	T	t	+T
NAK	$U	5	5	U	U	u	+U
SYN	$V	6	6	V	V	v	+V
ETB	$W	7	7	W	W	w	+W
CAN	$X	8	8	X	X	x	+X
EM	$Y	9	9	Y	Y	y	+Y
SUB	$Z	:	/Z	Z	Z	z	+Z
ESC	%A	;	%F	[	%K	{	%P
FS	%B	<	%G	\	%L	¦	%Q
GS	%C	=	%H	]	%M	}	%R
RS	%D	>	%I	∧	%N	~	%S
US	%E	?	%J	_	%O	DEL	%T, %X, %Y, %Z

Note: Character pairs /M and /N decode as a minus sign and a period, respectively. Character pairs /P through /Y decode as 0 through 9.

FIGURE 3.1: Normal 3 of 9 to Control 3 of 9 Character Conversions

IANEXP$

Expands a short form EAN (EAN-8) or UPC (UPC-E or UPC-E(1)) label into a full 13-character IAN format by adding leading zeros for EAN, or inserting zeros for UPC. It must be remembered that EAN-8 is NOT a zero-suppressed form of EAN-13. They are totally different numbering systems. The equivalent result to that obtained by **IANEXP$** can also be obtained by setting **EXPAND ON**. An "Invalid Arg" error will be reported if either the check digit is incorrect, the string contains a character not defined by the IAN-code symbologies, or the string length is not 8.

IANSUP$

Suppresses zeros in a long form UPC label provided that the label possesses the correct number of zero digits at specific locations. EAN-13 labels are not suppressible. An "Invalid Arg" error will be reported if either the string is not representable in UPC-E, the string does not consist of 13-numeric characters, the first character in the string is neither '0' or '1', or an invalid check digit is found.

Using Check Digit Verification

Wherever the barcode symbology defines either optional or mandatory check digits, ZENWAND provides means of either automatically (**CDIGIT**) or manually (**CDV**-functions) verifying check digits.

CDIGIT ON/OFF

With **CDIGIT ON** any barcode read will be automatically checked for the correct check digit(s). Should the check digit be found to be bad, then this will be indicated by the ZENWAND status indicator (**WANDSTAT**), a null string will be returned, and a bad scan tone will be emitted. If the check digit is good, then it will be deleted from the returned string. Where more than one check digit, or different computation methods are defined for a barcode symbology, reference should be made to the documentation for that particular decoder. For symbologies without defined check digits, the setting of **CDIGIT** is ignored.

CDV-Functions

For many reasons, not all barcodes can be scanned successfully, and it may be necessary to enter values from the keyboard. **CDIGIT** cannot automatically check such entries, so manual verification is necessary. In addition, with some applications, it may be necessary to verify the check digits, but to require them still to be returned with the string. Because setting **CDIGIT ON** will verify and automatically delete these digits from the returned string, it once again cannot be used. To overcome such difficulties as these, ZENWAND provides the following functions to manually access the **CDIGIT** coding. (It should be noted that Codabar and the IAN-Codes are not represented, although IAN-Codes can still be verified by means of the **CDV25D** function.)

CDV11 (*string*) Verifies a Code 11 string containing only one check digit;

CDV11K (*string*) Verifies a Code 11 string containing two check digits;

CDV25D (*string*) Verifies an Industrial 2 of 5 string. It can also be used to verify IAN-Code check digits;

CDV25I (*string*) Verifies an Interleaved 2 of 5 string. Provided that the string contains an even number of digits, **CDV25I** can also be used for verifying IAN-Codes;

CDV39 (*string*) Verifies a Code 3 of 9 string. **CDV39** will only accept a normal Code 3 of 9 data string. A full ASCII (Control 3 of 9) string will cause an error.

The **CDV**-functions are Boolean functions, requiring string arguments, that calculate the check digit for the passed barcode string, then attempt to verify it. A numeric integer will be returned to indicate the success of verification. A integer of '0' is returned upon verification failure, or an integer of '1' if verification was successful. Should any of the characters within the string not belong to the character set of that particular barcode symbology, then an "**Invalid Arg**" error is reported.

The following sequence shows the method of keyboard entry and check digit verification:

```
10 CDIGIT OFF                      Disables check digit verify & deletion;
20 DISP "Scan Label..."            Prompts for scan;
30 A$=INT25$                       Reads an Interleaved 2 of 5 label;
40 IF NOT LEN(A$) THEN 30          If bad scan, then retry;
50 IF NOT CDV25I(A$) THEN          If good scan, but bad check digit, then
   BEEP 200,.5 @ GOTO 30           beep and retry;
60 DISP A$                         Upon good scan, display barcode data.
```

Controlling ZENWAND Operation

The operation of the ZENWAND can be controlled by means of the following functions and statements:

CDIGIT ON/OFF	Selects automatic check digit verification and deletion from the returned string;
EXPAND ON/OFF	Selects automatic expansion of the label into full-ASCII or long-form label according active barcode decoder;
WANDTIME	Sets the time-out period for a barcode reading function;
ENDSCAN	Allows the user to select a key, including shifted keys, to terminate a barcode scanning function.
WANDSTAT	Returns the ZENWAND status showing result of scan, mode settings, etc.;
Wand Wake-up	When **LIVEWAND** mode is set, or the HP-71 was turned off by a programmatic **BYE**, pressing the ZENWAND switch turns on the HP-71.

WANDTIME

WANDTIME allows the user to alter the time-out for a barcode reading function from the default setting of 2.5 minutes to a user chosen value. If a barcode reading function is executed, but no scan is made before the end of the current time-out period, then ZENWAND will return a null string, emit a bad scan beep, and change the status of **WANDSTAT** to indicate that a time-out has occurred.

Example usage of **WANDTIME**:

WANDTIME 5	Sets time-out to 5 seconds;
WANDTIME ON	Disables time-out (equivalent to setting **WANDTIME INF**);
WANDTIME OFF	Resets time-out to default (2.5min)
WANDTIME (A-TIME)	Sets time-out to number of seconds calculated by **(A-TIME)**;
WANDTIME INF	Sets time-out to infinity.

The resolution of **WANDTIME** is limited to $^1/_{32}$ of a second. **WANDTIME 0** is the equivalent of setting **WANDTIME OFF** and resets the default value of 2.5 minutes. **WANDTIME** values equal to or greater than 2048 seconds (34.13 minutes) is equivalent to disabling the time-out by setting **WANDTIME INF** or **WANDTIME ON**.

ENDSCAN

A barcode reading function can be terminated by any of the means detailed under *Chapter 3: String Functions to Read Barçodes.* In addition, the applications programmer may specify a key that will terminate and exit a scan function. With the exception of the [f] and [g] shift keys themselves, any HP-71 key can be selected, whether it be unshifted, [f] or [g] shifted. During the execution of a barcode scanning function, pressing the selected key will terminate the scan, cause the bad scan beep, return a null string and change Wand status (See also: **WANDSTAT**) to indicate the scan was terminated by the **ENDSCAN** key. **ENDSCAN** takes a string expression representing the key location, for example:

ENDSCAN "fQ"	[f],[Q] terminates scanning.
ENDSCAN "#38"	[ENDLINE] terminates scanning.
ENDSCAN "="	[=] key terminates scanning.
ENDSCAN ""	Resets **ENDSCAN** setting

To permit the [ON] key to be used as the terminator with the **ENDSCAN** statement, [ATTN] must first be disabled by the **POKE** command:

```
POKE "2F441","F" @ ENDSCAN "#43"
```

WANDSTAT

The status of ZENWAND is stored as a one byte integer (a value between 0 and 255) that is returned by the **WANDSTAT** function.

The upper four bits indicate system settings and status, while the lower four bits indicate the status of the last scan performed by ZENWAND.

The four system bits indicate:

bits	decimal value	status
bit 7 :	128	**LIVEWAND** mode status: 1=ON, 0=OFF;
bit 6 :	64	If ZENWAND switch down, value = 1;
bit 5 :	32	**EXPAND** status: 1=ON, 0=OFF;
bit 4 :	16	**CDIGIT** status: 1=ON, 0=OFF.

The lower four bits indicate the following according to the value returned (Note that no other values are possible):

bits: 3 2 1 0	decimal value	status
0 0 0 0	0	good read, forward scan;
0 1 0 0	4	good read, reverse scan;
1 0 0 0	8	bad read, unable to decode;
1 0 1 0	10	bad read, check digit verify failed;
1 1 0 0	12	bad read, time-out;
1 1 0 1	13	bad read, terminated by **ENDSCAN** key;
1 1 1 1	15	No read, aborted by [ATTN]-key.

The reporting of the scan direction allows the applications programmer to design barcode reading systems that differentiate between the scanning direction. For example, forward scanning could add a number to a file, while reverse scanning could delete that number from the file.

All bad scans return a null string except in the case of aborting by pressing the [ATTN] key. In this case, the function itself is exited and normal [ATTN] key processing is done by the operating system, e.g. a running program will be suspended. If the [ATTN] key has previously been disabled by the **POKE** command, then it must be enabled again; or it will be treated like any other key pressed during a scanning operation. To enable [ATTN], use:

```
POKE "2F441","0"
```

The user can ascertain the status of a particular feature reported by **WANDSTAT** in two ways. By using the **BIT**-function from the HP-IL Module (e.g. **BIT(WANDSTAT,7)**), or by a routine such as:

```
10 B=7                                      Sets variable to bit number to be tested;
20 I=WANDSTAT                               Assigns WANDSTAT value to 'I';
30 DISP "Livewand "; @ IF FNC(I,B)          Displays status of Livewand;
  THEN DISP "ON" ELSE DISP "OFF"
...
100 DEF FNC(I,B)=                           User defined function to compute status
    MOD(I,2^(B+1))=>2^B                     of the bit number to be tested.
```

WAND WAKE-UP

ZENWAND allows the user to wake-up the HP-71 Computer from a deep sleep state (the HP-71 has been turned off) by pressing the ZENWAND switch. The **Wand Wake-up** feature is active whenever the HP-71 was turned off by the execution of a programmatic **BYE** command, or when **LIVEWAND ON** mode has been set. (See also: *Controlling HP-71 Operation - Using ZENWAND for Keyboard Entry.*)

This feature is further enhanced by combining a programmatic **BYE** with the **ON WAND GOSUB** or **GOTO** feature. (See also: *Controlling HP-71 Operation - Controlling Program Execution.*) When the user switches the HP-71 on by pressing the ZENWAND switch, program execution will automatically branch as indicated by the **GOSUB/GOTO** statements. If instead, the user switches on by pressing the 71's [ON] key, then program execution will continue as normal. The following example illustrates the basic principle:

```
100 ON WAND GOTO 200                 Set ON  WAND mode for Wand Wake-up;
110 BYE                              Turns off the HP-71;
120 DISP "[ON] Key Activation"       [ON]-key wake-up sequence;
130 WAIT 1 @ GOTO 110
...
200 DISP "ZENWAND Activation"        ZENWAND switch wake-up sequence.
210 WAIT 1 @ GOTO 110
```

Automatic Discrimination of Barcode Types

Although most applications programs written for ZENWAND will only utilise a particular barcode type, it may be necessary for the program to accept and automatically discriminate between various possible barcode types. Such a situation may apply where a common application program is used within one company, but different offices or departments have different requirements, e.g. alphanumeric, high density, etc. In these circumstances, employing a common program, with standardised prompting and input procedures, considerably aids user friendliness and reduces potential errors. The following ZENWAND functions provide the programmer with the means to automatically discriminate between and report barcode types scanned.

BARCODE$

This function will automatically read any barcode type provided that there is a decoder present that can interpret that particular barcode. When **BARCODE$** is executed, ZENWAND will firstly search its own barcode decoders, before searching any external decoders that may be present in plug-in modules, or RAM-files. The ZENWAND Barcoding System offers considerable advantage over other traditional products by allowing additional barcode types to be handled by external decoders that can be distributed on mass-storage media. Zengrange Ltd will be making further decoders available at a later date and can also design special decoders for custom applications.

In some instances, the type of barcode used on a particular label may be ambiguous, in that two different decoders may be able to decode that particular label. In such instances, ZENWAND will use the first decoder found during the search. This problem will not occur with the internal ZENWAND decoders, but may be encountered with external decoders for barcodes of similar symbologies.

The settings for the check digit verification function (**CDIGIT ON/OFF**) and the expansion function (**EXPAND ON/OFF**) are also valid during execution of **BARCODE$**. The settings conform to the rules defined under the specific barcode type functions and respond as though that function was called directly. For example: If a type Code 11 label is scanned, then **BARCODE$** will use the **EXPAND** and/or **CDIGIT** settings as if the label was scanned using the **CODE11$** function.

Because **BARCODE$** is auto-discriminating, it must analyse the scanned label and search to find the correct decoder to use. This means that it is slower than reading a barcode by calling the decoder directly. For applications where it is necessary to use auto-discrimination, it may be advantageous to use **BARCODE$** only on the first label of each barcode type to be scanned. The actual decoder used during that scan can then be determined by **BARTYPE?**, declared to be the **ACTIVE** decoder and subsequent barcodes read by means of **ACTIVE$**. (See also: *BARTYPE?* and *Section C - Using ZENWAND for Keyboard Entry.*)

BARTYPE?

When using auto-discrimination, it may be necessary for the program to take various subsequent actions depending upon the type of barcode read by **BARCODE$**. The barcode type read can be determined by the ZENWAND function, **BARTYPE?** which returns a value indicating the last barcode decoder used. The returned value is the XFN number (eXternal FunctioN) of the decoder. If the last scan was aborted, or the subject of a bad scan, then the XFN-number is set to zero; thereby providing an additional means of trapping bad scans. Correct scans, for decoders built into ZENWAND, will be indicated by XFN-number ranging from 245003 to 245009. As a general rule, XFN-values follow the format:

i i i f f f

where: iii is the ID of the LEX-file or module, and
fff is the function number within that file.

(See also: *Appendix D, Reference Information - Keyword Index* for a complete list of ZENWAND XFN-Numbers.)

The following routine shows a typical example of using **BARCODE$** and **BARTYPE?**. It also uses three new keywords that will be described in detail later. Briefly their function is as follows:

MSG$(*message number*) — Returns text string of message number indicated.

ACTIVE *barcode type* — Selects the barcode type decoder to be used for scanning.

ACTIVE$ — Reads a barcode of the type previously declared to be **ACTIVE**.

Example Routine:

```
10 DISP "Scan Barcode..."
20 A$=BARCODE$                          Reads barcode into variable using
                                        automatic discrimination
30 IF  LEN(A$) THEN                     If a good scan (length of barcode was
     A=BARTYPE? ELSE 20                 non-zero), then A = type of code read. If
                                        a bad scan, then re-scan barcode.
40 DISP  "Barcode Type: "&MSG$(A)       Displays barcode type scanned.
50 ACTIVE  A                            Selects the type of barcode just scanned
                                        to be the active type for future
                                        processing.

...
1000 B$=ACTIVE$ @ DISP  B$              Reads a barcode label of the ACTIVE
                                        barcode type into variable B$ and
                                        displays this.
```

Analysing Barcodes

In addition to being able to scan and decode barcodes, ZENWAND is also able to analyse any pattern of black and white bars by means of the function WAND$. Applications of **WAND$** are generally rather specialised; such as analysing the printing quality of barcode labels to determine their suitability and readability for a particular application.

WAND$

Executing **WAND$** causes ZENWAND to wait for a scan to take place. After completion, a character string is returned representing the amount of time that ZENWAND spent over each bar or space element in the barcode label. To create the string, ZENWAND first finds the narrowest element that it has scanned (this being the shortest time element), and assigns it the value 16 (i.e. **CHR$(16)**). A total string is then built up based on the relationship of other elements to the narrowest element. Because of this relationship, resolution will be limited to 1/16th of the narrowest element scanned.

For example, a label containing bars and spaces of a constant size, where 0's represent spaces and 1's bars, might be:

1 1 0 1 0 1 0 0 1 0 1 1 1 0 1 0 1

With perfect printing and a constant scanning speed this should return a string of the following characters:

Bars:	11	0	1	0	1	00	1	0	111	0	1	0	1
CHR$(x):	32	16	16	16	16	32	16	16	48	16	16	16	16

Note that where there is no change between elements, those elements are added together. Because of the deficiencies of most barcode printing processes, the character string returned will most likely show typical inaccuracies, such as:

CHR$(x):	36	18	18	17	19	34	17	16	52	16	17	18	16

These differences can be used to verify the quality of the barcode printing process being used.

WAND$ is subject to all the normal setting of **WANDTIME**, **ENDSCAN**, etc. Settings for check digit verification (**CDIGIT**) and code expansion (**EXPAND**) have no effect upon the scan, nor the resulting string. If the scan was good then a forward scan is always reported by **WANDSTAT** and the **BARTYPE?** value is always set to 0.

When printing the returned character string, users should be aware that characters between 0 and 31, or over 126 are not standardised in the ASCII-defined set. Many printers treat such characters as non-printing control codes (such as Form Feeds, Character Font Changes, etc); while others use these character numbers to represent special symbols. These may, or may not be the same as the character set displayed in the HP-71 Display. Decimal values of the string characters can be printed by means of the HP-71 mainframe function NUM. For example, the following routine will print a list of bar and space widths for a scanned barcode label.

```
10 OPTION BASE 0 @ DELAY 0,0         Sets option base and delay rates;
20 DIM A$[200], B$(1)[5]             Dimension variables;
30 B$(1)="Bar " @ B$(0)="Space"      Sets up printout labels;
40 DISP "Scan Label..."              Prompt to begin scanning;
50 A$=WAND$                          Scan barcode pattern;
60 FOR I=1 TO LEN(A$)                Loop for number of label characters;
70 PRINT B$(FP(I/2)&" width: ";      Print output label, then numeric value
   NUM(A$[I])                        of each character;
80 NEXT I                            Loop back for next character.
90 END
```

Chapter 4

CONTROLLING HP-71 OPERATION

In addition to barcode reading and analysing functions, ZENWAND also provides functions that allow the programmer to directly control the operation of the HP-71.

These functions fall into two main categories:

Using ZENWAND for Keyboard Entry

LIVEWAND

This mode allows both ZENWAND and the HP-71's keyboard to be used as input devices whenever the HP-71 is expecting input from the keyboard. Input from the keyboard is expected when an **INPUT** or **LINPUT** statement has been executed, or when the HP-71 has entered a normal state of light sleep. (Light sleep is the state that the HP-71 powers down into whilst awaiting new instructions. Deep sleep is entered when the HP-71 has been turned off.)

When using ZENWAND to input data, a terminating key is simulated at the end of the scan; equivalent to keying-in data from the keyboard, then pressing the [ENDLINE]-key. With **LIVEWAND** mode active, this can be very useful in conjunction with the **INPUT** statement because the user then has an option to either key in the data and press the [ENDLINE]-key, or to scan a barcode. A typical use of **LIVEWAND** mode would be to cater for circumstances where, because of damage, a barcode label cannot be read. If the scan fails, the user can just key in the data values from the keyboard and terminate each label by pressing [ENDLINE].

LIVEWAND mode is activated by the statement **LIVEWAND ON**, and disabled by the statement **LIVEWAND OFF**. To use **LIVEWAND** mode, a particular barcode type must have been declared as active by means of the **ACTIVE** statement. If none has been declared, then **LIVEWAND** defaults to Code 3 of 9 as the active barcode type. A typical example of using **LIVEWAND** mode would be as follows:

10 ACTIVE "CODABAR$"	Declares Codabar to be the **ACTIVE** barcode type.
20 LIVEWAND ON	Activates **LIVEWAND** mode.
30 DISP "Scan or Key Label...."	Prompts for input from keyboard or by scanning with ZENWAND.
40 INPUT A$	Awaits scan, or keyboard input followed by [ENDLINE].

When ZENWAND is being used for **LIVEWAND** mode input, the settings of both check digit verification (**CDIGIT**) and code expansion (**EXPAND**) are observed, in as far as these apply to the particular barcode type selected as active. In addition, **LIVEWAND** mode also follows the rules for wand status indication (**WANDSTAT**) and reporting the last barcode type scanned by ZENWAND (**BARTYPE?**).

Because **LIVEWAND** mode requires that the decoder be pre-defined, it cannot be classified as auto-discriminating. However, **LIVEWAND** mode can be made to appear to be auto-discriminating, by employing:

ACTIVE "BARCODE$"

When **LIVEWAND** mode is set, ZENWAND can also be used to wake up the HP-71 from deep sleep, i.e. when the computer has been turned off. By combining **LIVEWAND ON** with the wand interrupt statement (**ON WAND GOTO** or **ON WAND GOSUB**), the application program can be made to automatically execute upon activation of the ZENWAND switch. (See also: *Chapter 3, Controlling ZENWAND Operation - Wand Wake-up.*)

Normal HP-71 time-out settings, valid for **INPUT** and **LINPUT**, are maintained when **LIVEWAND ON** has been executed.

ACTIVE

ACTIVE selects the active barcode type for use with **LIVEWAND** mode. It also allows the applications programmer to call a barcode reading function in an indirect manner by defining the barcode type decoder to be used when scanning with **ACTIVE$**.

Setting the **ACTIVE** barcode decoder type can be accomplished in one of two possible ways:

- by indicating the XFN-number of the decoder to use, or
- by indicating the name of barcode decoder as a string.

For example:

ACTIVE 245003	Set to XFN-number 245003 (CODABAR)
ACTIVE 245 * 1000 + 5	Set to XFN-number 245005 (CODE39)
ACTIVE BARTYPE?	Set to that of last decoder used. (Determined by XFN-number returned by **BARTYPE?**.)
ACTIVE "CODE11$"	Set decoder to CODE11 barcodes.
ACTIVE MSG$(BARTYPE?)&"$"	Set to that of the last decoder used. Determined by the **MSG$** (message string) of the XFN-number returned by **BARTYPE?**, converted into a string function name.

When indicating the barcode type by an XFN-number, this may be either a valid internal (built into ZENWAND) or external (contained in a RAM-file or plug-in module) decoder. Invalid XFN-numbers return "**Invalid Arg**" errors.

When indicating the barcode type by means of a string expression, the string must represent the name of a valid internal or external decoder. An "**Invalid Arg**" error is returned if the function does not exist, or if the function is not a barcode decoder.

ACTIVE$

ACTIVE$ is a barcode string function that enables indirect reading of a barcode. The type of barcode to be read must have been previously declared with the **ACTIVE** statement.

In reading the declared barcode type, **ACTIVE$** conforms to the definitions and settings for check digit verification (**CDIGIT**) and code expansion (**EXPAND**), in as far as they may apply to that particular barcode type. E.g. Using **ACTIVE$** to indirectly read a Code 3 of 9 barcode (having declared **CODE39$** as **ACTIVE**) is equivalent to directly reading the barcode using the function **CODE39$** and all settings for **CDIGIT** and **EXPAND** will be observed.

During an application program it may be necessary to set the active barcode type for future scans according to the type last used. The following sequence can be used to achieve this:

10 A$=BARCODE$	Await scan of first label using auto-discrimination of barcode type.
20 ACTIVE MSG$(BARTYPE?)	Determine decoder used and declare this as the **ACTIVE** barcode type.
30 A$=ACTIVE$	Await scans of further barcodes of type declared to be **ACTIVE**.

EXAMPLE PROGRAM:

The following short program illustrates the use of **ACTIVE**, **ACTIVE$** and **LIVEWAND** mode to scan five barcode labels, place the returned values into a data file, trap bad scans and allow three repeat scans before prompting for keyboard entry of the label values.

10 DIM A$[30],T,B	Dimension maximum lengths.
20 DELAY 0,0	Set delay.
30 ASSIGN #1 TO LABELS	Open channel to data file "LABELS", or create the file if it doesn't exist;
40 DISP "Scan Test Label..."	
50 A$=BARCODE$	Scan using auto discrimination;
60 T=BARTYPE?	Assigns 'T' the XFN-number of decoder used in line 50;
70 IF NOT T THEN 50 ELSE ACTIVE T	Traps error in reading & if a good scan, makes barcode decoder (used for the test label) ACTIVE;
80 FOR I=1 TO 5 @ B=0	Set up loop for 5 labels & set variable for bad scan count to zero.
90 DISP "Scan ";MSG$(T);" Label:";I	Prompts for particular barcode type and number (1 to 5) to be scanned;
100 A$=ACTIVE$	Await scan of barcode label.
110 IF LEN(A$) THEN 160	If good scan, go to print# routine.
120 B=B+1 @ IF B<3 THEN 90	Increment bad scan variable. If <3 repeat.

```
130 LIVEWAND ON                      If 3 bad scans obtained, set LIVEWAND
                                     mode for keyboard entry;
140 LINPUT "Key-in  Label: ";A$      Prompt for keyboard input of label
150 IF  NOT  LEN(A$) THEN  140       If not keyboard input, reprompt;
150 LIVEWAND OFF                     Cancel LIVEWAND mode;
160 PRINT #1;A$                      Print# label data to file;
170 NEXT I                           Loop to scan next label;
180 DISP "Done"                      Finished.
```

Controlling Program Execution

ON WAND

In addition to allowing control of the keyboard during scanning, ZENWAND also allows the programmer to control the flow of program execution in response to pressing the Wand switch. This interrupt handling is much the same as the "ON *Event*" interrupts available in the HP-71 mainframe functions **ON TIMER, OFF TIMER** and **ON ERROR, OFF ERROR.**

ZENWAND provides program branching, following Wand interrupts, in two forms:

ON WAND **GOTO** *line# or label* — Continue program execution at the specified line number or program label.

ON WAND **GOSUB** *line# or label* — Branch to a subroutine beginning at the specified line number or program label, and return after completion of subroutine.

It is important to note that **ON WAND** interrupts are checked after each statement, and not just at the end of a program line. For comparison, the HP-IL Module function **ON INTR** only checks for HP-IL interrupts at the end of each program line, while the HP-71 mainframe function **ON TIMER** checks for timer interrupts after each statement.

Similar to other HP-71 "**ON** *Event*" statements, the **ON WAND** branch is local to a particular program environment. **ON WAND** branches are therefore saved upon calling another environment and reinstated upon returning from that environment. (See also: *HP-71 Owner's Manual, Section 12 - Subprogram Environments, Pages 210 to 214.*)

In addition to being able to interrupt the flow of a running application program, **ON WAND** can be used to automatically branch to and begin program execution at a specific line/subroutine upon waking up the HP-71 by pressing the ZENWAND switch. (See: *Chapter 3 - Controlling ZENWAND Operation, Wand Wake-up.*)

OFF WAND

The **ON WAND GOTO/GOSUB** function operates locally within a subprogram environment. Executing the **OFF WAND** function therefore inhibits program branching only within that one program environment. Further **ON WAND** commands in other calling or called subprograms are not influenced by **OFF WAND**. (See also: *HP-71 Owner's Manual, Section 12 - Subprogram Environments, Pages 210 to 214.*)

Chapter 5

PRINTING BARCODES WITH ZENWAND

In many barcoding applications it is difficult to justify either the cost of commercially printed barcodes, or the cost of specialised printing equipment to produce such barcodes in-house. To overcome this difficulty, ZENWAND has been designed to allow users to print their own barcodes on micro-computer printers such as the HP-2225B ThinkJet Printer commonly found in many offices.

Because ZENWAND has been designed around the HP-71B to provide a completely portable barcoding system, communicating with the outside world via HP-IL (the Hewlett Packard Interface Loop), barcode printing currently utilises only the facilities available on the portable HP-IL ThinkJet Printer (HP-2225B). However, the barcode printing routines within ZENWAND have been designed to accept additional printer drivers in the form of RAM files distributed on mass storage media, or contained in custom plug-in application modules. In the future, Zengrange Ltd will be making further Printer Drivers available for printers, such as the HP-LaserJet Printer (HP-2686A), that will offer higher density and resolution printing of barcodes.

Barcode printing with ZENWAND can be accomplished in two forms:

- by means of a user-friendly application program, **BCPRINT**, that provides all the input prompting, validation and error handling necessary to ensure trouble free barcode production (See: *The BCPRINT Program* and *Using BCPRINT to Print Barcodes*);

- by means of the user's own program that provides all the prompting, validation and error handling of data input before passing these values as parameters to the barcode printing subprogram, **BCP**, by means of the HP-71 **CALL** statement. **BCP** can be used where the user needs to print a large number of barcodes, or perhaps an obscure sequence, without having to repeatedly answer the extensive prompting available with **BCPRINT**. (See: *The BCP Subprogram.*)

The BCPRINT Program

BCPRINT is the ZENWAND user-interface program that automatically handles barcode printing. Before being able to print barcodes, **BCPRINT** does require the following information:

- The assigned name of the HP-IL **Output Device** on which the barcode is to be printed,
- The name of the file in memory to use as the **Driver** for the selected Output Device,
- The name of the **barcode type** to print,
- The **resolution** at which the barcode is to be printed,
- The **data** to be printed as barcode.

The advantage of employing **drivers** for output devices on the HP-IL is that **BCPRINT** commands are easily translated into the specific control and escape code sequences required by the output device. By this means, it is also possible to handle devices other than those directly available on HP-IL. E.g. An HP-2686A LaserJet Printer could be used as the output device by connecting via an HP-IL/RS232C Interface and providing the necessary output driver.

When a particular printer on HP-IL, with its associated device driver, is selected as the output device, this generally determines the default settings and capabilities available for barcode printing. For example, the type of printer will determine the highest resolution and density of barcode that can be printed on that particular printer. Because some barcode types can only be printed at particular resolutions, it may also determine which barcode types can be printed.

The default printer related settings for BCPRINT are:

- the output device is the currently assigned PRINTER IS device;
- the barcode type is Code 3 of 9;
- the resolution/density of barcode printing is the lowest that the current output device can print.

In addition, there are various other data that the user may optionally choose to provide:

- a descriptive label to be printed with the barcode;
- the height of the barcode;
- whether or not check digit(s) are to be computed;
- special extensions dependent upon barcode type.

The default settings for these options are:

- the barcode data is also to be printed as the barcode label;
- the barcode height of 10mm is to be used;
- no check digits are to be printed (unless the barcode type selected contains mandatory check digits).

Using BCPRINT to Print Barcodes

Barcode printing is executed by typing:

RUN BCPRINT or **CALL BCPRINT**

BCPRINT firstly performs a set-up sequence to check that the Interface Loop is intact, and whether or not a printer is assigned. The following warning messages may be displayed:

"**Loop Broken**"	The HP-IL is either physically broken or an HP-IL device has been switched off. The BCPRINT program terminates at this stage;
"**No Printer Assigned**"	No HP-IL device has been assigned as the current PRINTER IS device.

If a printer has been assigned as the current PRINTER IS device, and that printer is one currently supported by ZENWAND, then execution branches immediately to the main command level prompt:

BCPrint: Ready

Should a printer either not be assigned, or the assigned device not be an HP-2225B (ThinkJet) printer, the program branches to a printer set up routine, where the user is requested to input the name of the printer device ("**Printer ?**") and/or the name of the device driver file ("**Driver ?**") present in memory. The printer name may be input as the device specifier or the loop name given to that device, e.g:

Printer ? HP2225B	Assigns the device with a device-ID of HP2225B as the barcode printer;

Printer **?** 2 — Assigns device at position 2 on the loop as the barcode printer;

Printer **?** :PRINTER — Assigns the first printer class device found on the loop.

Input of an illegal response to the "**Printer?**" prompt will cause the warning "**Invalid Printer**" followed by a return to the prompt for re-entry of the printer name. Where the program requests the entry of a device driver name, by prompting with "**Driver?**", entry of illegal data will cause reprompting with either:

"**Driver ?**" — The specified file name is either non-existent, or not a valid device driver;

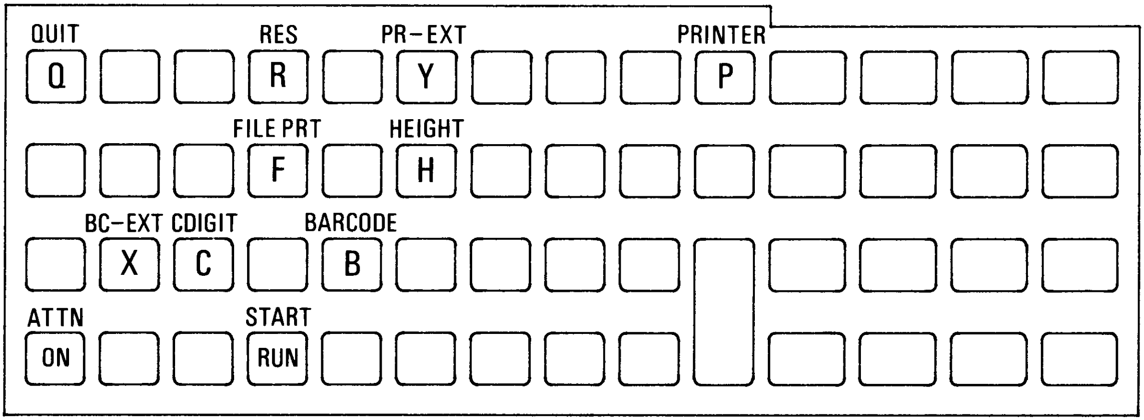
"**Invalid Device**" then "**Printer ?**" — Signifies that the device driver was found, but that a printer for that driver was not found;

"**No Driver**" — No file name for a device driver was input.

Whilst the "**Printer?**" prompt is displayed, the user can terminate the running program by clearing the default response (press [f] [-LINE]) and then pressing [ENDLINE].

Following assignment, the program confirms printer selection by returning to the main command loop with the confirmation message remaining in the display:

Printer=*Printer Name*



The "**BCPrint: Ready**" Prompt is the main command level prompt, from which the user can select various options according to the key pressed. The main command level keys are shown on the keyboard opposite and provide selection for:

[P] Allows selection of the output device to use as the printer (defaults to the current PRINTER IS device);

[B] Allows selection of the barcode type to print. Defaults to Code 3 of 9;

[F] Allows printing of an HP-71 file in barcode form (not currently available);

[H] Allows the barcode height to be specified between heights of 5 and 150mm. The default height is 10mm;

[R] Where the barcode type and the printer device allow different resolutions, the user can specify [L]ow, [M]edium or [H]igh resolution. The default is the lowest for that particular barcode type and printer;

[C] Where the barcode symbology specifies optional check digit(s), the user can select the digit(s) inclusion and the number of digits. For most barcode symbologies, only the options [Y]es or [N]o will be allowed. Default setting is [N]o;

[X] Allows selection of "barcode extensions" where the current barcode type permits such. This is provided by a subprogram call in the barcode header file, and is currently supported only by Code 3 of 9 to permit selection of Control 3 of 9 (press [Y]es) or Normal 3 of 9 (press [N]o);

[Y] Allows selection of "printer extensions" where the current output device provides special features. (Not currently available with the HP2225B ThinkJet Printer ;

[RUN] Allows the user to specify a descriptive label and the data, or a range of data to be barcoded;

[Q] Quits the barcode printing program, displays "**BCPrint: Done**" and returns to the BASIC operating system;

[ATTN] Cancels the selection confirmation message and displays the main command level prompt, "**BCPrint: Ready**".

Under circumstances where the currently specified barcode type or printer does not define some of the above functions, e.g. barcode extensions, then pressing that particular option key causes no action to be taken.

In response to pressing any of the above keys, BCPRINT expects input from the user. Where the prompt contains a blinking cursor, BCPRINT requires input of numeric/alphanumeric characters followed by pressing [ENDLINE], e.g.:

Barcode ? CODE39

Where no cursor is displayed by the prompt, BCPRINT expects a [Y]es, or [N]o response, e.g.:

Checkdigit ?

Pressing any other key at this time simply displays the current setting for that option.

Where one of a number of special keys may be pressed to select particular options, BCPRINT displays the options available in parentheses, e.g.:

Resolution ? (LMH)

Pressing any one of the [L], [M] or [H] keys selects that setting.

The BCPRINT Options Keys:

[P] **Selecting the printer output device:** Pressing the [P] key allows the user to change the printer device on which barcodes will be printed. The BCPRINT program will respond with:

Printer ? *Current Printer Device*

The user should respond by entering the device specifier of the device to use and press [ENDLINE]. If a valid printer device name was used, and a driver with that printer device name exists, BCPRINT will respond with:

Printer=*Printer Name*

However, should a device driver of that name not exist, BCPRINT will request the name of the printer driver file:

Driver ? *Current Device Driver*

The user should respond with the file name of the device driver. For a valid driver name, BCPRINT will confirm the selection of the specified printer by responding with:

Printer=*Printer Name*

Following selections by pressing one of the option keys, BCPRINT returns to the main command loop with the confirmation message remaining in the display. At this stage BCPRINT is waiting for the user to select the next option.

[B] **Selecting the Barcode Type to Print:** Pressing the [B] key allows the user to change the type of barcode to be printed by BCPRINT. The program will respond with:

Barcode ? *Current Barcode Type*

The user should respond by entering the barcode type to print and press [ENDLINE]. Currently BCPRINT has only the following barcode types available:

CODE39	Selects printing of Code 3 of 9;
IND25	Selects printing of Industrial 2 of 5 barcodes;
INT25	Selects printing of Interleaved 2 of 5 barcodes.

If an invalid barcode type name was entered, BCPRINT will display a warning "**Invalid Barcode**" before returning to the previous input prompt.

For a valid driver name, BCPRINT will confirm selection of the specified printer by responding with:

Barcode=*Barcode Type*

Following selection, BCPRINT returns to the main command loop with the confirmation message remaining in the display. At this stage BCPRINT is waiting for the user to select the next option.

Printing Code 3 of 9 with Check Digits. Because ZENWAND was designed to allow future expansion of barcode printing, the **BCPRINT** routine must be modified when trying to print a Code 3 of 9 barcode containing check digits. This alteration is only applicable to Code 3 of 9 (with check digits) and involves creating the following program line in user memory:

```
10 SUB CHK39(B$) @ CALL CDC39(B$) @ END  SUB
```

This program line may be present in any program in memory and, if present, will provide the correct check digit encoding necessary for printing Code 3 of 9 barcodes. If the line is not found, ZENWAND will respond with the message:

"Sub Not Found"

[F] **Selects printing of HP-71 Files in Barcode Format:** This option is not currently supported, but will be available at a later date as an extension to the ZENWAND system. Pressing the [F] key causes a file search for a *file driver* that will allow the user to specify the HP-71 file to be printed by BCPRINT. Because this driver is not present, the BCPRINT program currently returns to the "**BCPrint: Ready**" prompt.

[H] **Selects the height of barcode to print:** Pressing the [H] key allows the user to select the height of barcode to be printed by BCPRINT. The BCPRINT program will respond with:

Height ? *Current Barcode Height*

The default setting for height is 10mm. The user can respond with values, between 5 and 150mm, and press [ENDLINE].

Following selection of the height, BCPRINT returns to the main command loop with the confirmation message:

Height=*Current Height*

remaining in the display. At this stage BCPRINT is waiting for the user to select the next option.

[R] **Selecting the Barcode Resolution for Printing:** Pressing the [R] key allows the user to specify the barcode resolution to use. The BCPRINT program will respond with:

Resolution ? (LMH)

Where the barcode type and the selected printer device permit printing at differing barcode resolutions, the user may respond by pressing either the [L]ow, [M]edium or [H]igh keys. The default setting for resolution is the lowest possible for that particular barcode type and printer.

Following selection, BCPRINT returns to the main command loop with the confirmation message:

Resolution=*Resolution chosen*

remaining in the display. At this stage BCPRINT will wait for the user to select the next option.

BCPRINT will only validate the resolution specified, and display a warning message ("**Resolution Too High**"), once the printing sequence has begun. Should the user have chosen an invalid resolution, execution of BCPRINT will be terminated immediately.

[C] **Selects the inclusion of Check Digits:** Pressing the [C] key allows the user to specify the inclusion, and number of check digits in the printed barcode. The default setting for the supplied barcode types is N(o), unless the barcode type selected defines mandatory check digit(s), or the setting has previously been changed. Additional responses may be allowed for external printer drivers and encoders. Refer to the supplied documentation for further information.

When [C] is pressed, BCPRINT will respond with:

Checkdigit ? *Current Setting*

For most barcode types, only the [Y]es, or [N]o keys are operable. However, where the barcode symbology defines multiple check digits, the user may also respond with the relevant number key.

Following selection, BCPRINT returns to the main command loop with the confirmation message:

Checkdigit=*Current Setting*

remaining in the display. At this stage BCPRINT is waiting for the user to select the next option.

When inclusion of check digits has been selected, and Code 3 of 9 barcodes are to be printed, the program line:

```
10 SUB CHK39(B$) @ CALL CDC39(B$) @ END  SUB
```

must always be in memory. (See also key option: [B] - *Selecting the Barcode Type to Print for further details.*)

[X] **Selects the Barcode Extension:** Where the previously selected barcode type defines such, pressing the [X]-key will allow the user to specify the use of barcode extensions. If no extensions are defined for a particular barcode type, then pressing the [X]-key has no action.

For the barcode types currently supported, only Code 3 of 9 permits the use of barcode extensions. When CODE39 has been selected as the barcode type to print, pressing the [X]-key allows the user to select between Control 3 of 9, or Normal 3 of 9 barcode printing. BCPRINT therefore prompts with:

Control 39 ?

The user should respond by keying either [N]o to indicate Normal 39 barcodes or [Y]es to select Control 3 of 9.

When specifying other barcodes, the user should refer to the external decoder documentation for information on the barcode extensions available for that barcode type.

[Y] **Selects Printer Extensions:** Where a printer and its printer driver specify extensions, these may be selected by pressing the [Y]-key. For the currently supported HP-2225B ThinkJet printer, no printer extensions are available. BCPRINT therefore takes no action when the [Y]-key is pressed.

The user should refer to the documentation provided with the printer driver for information on the extensions available with that printer.

[RUN] **Begin Barcode Printing Sequence:** Pressing the [RUN]-key begins the printing process by requesting the user to input the barcode data to be printed.

The BCPRINT program allows the user to specify the following data items:

Start ?: The first barcode data to be printed. The data input, whether numeric or alphanumeric, must be defined by the particular barcode symbology selected. Input of invalid, undefined data will cause warnings:

"Illegal Char '*x*'" or **"Invalid Arg"**

The user may terminate barcode printing at this stage by clearing any data entry in the display, press [f][-LINE], and pressing [ENDLINE].

The length of barcode data that may be input is determined by the particular printer device used, the barcode symbology selected, and whether additional extensions or check digits and resolution have been specified.

Label ?: A descriptive label to use for the starting barcode. A default label, of the starting barcode data, is presented to the user for selection. BCPRINT requires that a descriptive label is always input. If by chance a label is not included, the barcode will still be printed, but printing may terminate with an error or warning message. Label data may be numeric or alphanumeric.

End ?: The data of the last of an automatically incremented sequence of barcodes. This allows the user to print an incremental series of barcodes with minimal user supervision. Logically, this is only applicable where the starting barcode data and the descriptive label are capable of being incremented by virtue of being numeric and identical. Selection of the ending barcode data defaults to that of the starting barcode.

Following completion of data entry, BCPRINT begins verification of the data input, then prints the barcode or sequence of barcodes specified. During this period, the HP-71 display shows the message:

"Working...".

Following completion of printing, BCPRINT returns to the main command loop with the message:

"BCPrint: Ready"

in the display. At this stage, BCPRINT waits for the user to select the next option.

[Q] **Quits the BCPRINT Program:** From the main command level prompt, the user may terminate the program by pressing the [Q]-key. BCPRINT displays the message:

BCPrint: Done

[ATTN] **Displays the BCPrint: Ready Prompt:** From the main command level, whenever the display shows one or other of the BCPRINT confirmation messages, the main prompt, "**BCPrint: Ready**", may be returned to the display by pressing the [ATTN]-key.

The BCP Subprogram

BCP is the main subprogram that prints the barcode data. Unlike **BCPRINT**, **BCP** has been designed to be controlled from a user's own supervisory program. It is particularly useful for applications where the user needs to customise data input prompting, needs to produce a large number of barcodes, or needs to print an obscure sequence of incremented barcodes that cannot be handled by **BCPRINT**. Because **BCP** has been designed to be called from another program it assumes that all the input parameters passed to it have been validated for correctness and that all error handling will be undertaken by the user's application program.

The **BCP** subprogram is **CALL**ed by the applications program as follows:

```
CALL BCP(B$,P$,T$,B1$,H,L1$,B9$,R,C1$,X1$,X$)
```

The parameters passed to **BCP** represent values for:

B$ The name of the barcode encoder file to use, e.g. CODE39, IND25, INT25, etc.);

P$ The device specifier of the loop device on which barcodes are to be printed. This is the same name you would normally type in response to the **BCPRINT** prompt "**Printer ?**";

T$ The name of the file to use as the printer driver;

B1$ The data of the starting barcode to be printed;

H The height of the barcode in millimeters;

L1$ The descriptive label of the first barcode;

B9$ The data of the ending barcode. This provides for automatic incrementation of numeric barcodes in a manner similar to that of **BCPRINT**;

R The desired resolution of the barcode entered as an integer. For those barcode types currently defined, Low=1, Medium=2 and High=3. In the future, external encoders may be defined with other resolutions. For these, value 1 will always represent the first option, with additional options each incrementing by 1;

C1$ The name of the check digit calculation routine (e.g. CDC39, CDC25D, CDC25I, etc.). Passing a null string, causes no check digit to be computed;

X1$ Barcode and printer extension information. Currently this should always be left as a null string;

X$ A variable to return the error or warning messages, should **BCP** fail to run to completion. X$ must be dimensioned to be a string =>22 characters long. Note that warning and error messages are not displayed by **BCP**, but are stored by the variable X$ for subsequent processing by the user's own application program.

Printing Code 3 of 9 with Check Digits. Because ZENWAND was designed to allow future expansion of barcode printing, an alteration must be made to the **BCP** subprogram to print a Code 3 of 9 barcode containing check digits. This is only applicable to Code 3 of 9 (with check digits) and involves creating the following program line in user memory:

```
10 SUB CHK39(B$) @ CALL CDC39(B$) @ END  SUB
```

This program line may be present in any program in memory and, if present, will provide the correct check digit encoding necessary for printing Code 3 of 9 barcodes. If the line is not found, ZENWAND will respond with the message:

"Sub Not Found"

Control 3 of 9 Expansion: To implement printing of Control 3 of 9 expansion using the **BCP** subprogram, the user must either; perform the barcode data conversion manually by inserting the necessary control characters, or use the automatic conversion obtained by passing the data string through the ZENWAND function **NORM39$**.

For example: Suppose the character string **"David"** is to be printed as barcode. The user may either key in this string as:

B1$="D+a+v+i+d"

or, pass the string **"David"** through the **NORM39$** function:

10 Input "Start Barcode: ";B1$	Prompts for input of barcode data string;
20 B1$=NORM39$(B1$)	Converts string **"David"** into string consisting of conversion characters **"D+a+v+i+d"**

For additional information regarding the Control 3 of 9 conversion characters, refer to the explanations of **CTRL39$** and **NORM39$**. (See: *Chapter 3. Reading Barcodes With ZENWAND - Controlling the Barcode Decoders.*)

When printing barcodes by calling the BCP subprogram, the user should ensure that all validation of input is taken care of **before** passing the values to **BCP**. Errors will in most instances terminate the program, but may cause other unexpected results, or the program may not exit after completion of printing. Mostly the error or warning message can subsequently be examined by means of the **DISP X$** statement. However, because BCP uses binary subprograms, to compute check digits, etc., there is a risk that if these fail by virtue of being passed bad data, the BASIC operating system will terminate execution without trapping and notifying the error message.

Examples Using the HP-2225B ThinkJet Printer

When used with the HP-ThinkJet printer, ZENWAND provides a completely portable, battery powered barcoding system. Users should refer to the *HP-IL Interface Owner's Manual* and the *HP-ThinkJet Printer Reference Manual* for specific information regarding interconnecting and operating the HP-82401A HP-IL Module and HP-2225B ThinkJet Printer.

EXAMPLE 1: Printing with BCPRINT

In the following example, an HP-71B has been set up with ZENWAND plugged-in and two HP2225B ThinkJet Printers on the Interface Loop. The **BCPRINT** program will be used to print a Code 3 of 9 barcode, with check digit, on the second ThinkJet.

Firstly, because check digits are to be included in the barcode, key in the following program:

EDIT BAR39	Sets up new program called BAR39;
10 SUB CHK39(B$) @ CALL CDC39(B$) **@ END SUB**	Sets up the subprogram call to allow the use of a check digit with the Code 3 of 9 barcode.

The next stage is to run the BCPrint routine. In response to the prompts, the user should perform the indicated keystrokes. Unless the keystrokes are shown in square brackets, e.g. [Y], the [ENDLINE]-key should be pressed after the keystrokes shown.

Keystrokes	Display
RUN BCPRINT	BCPrint: Ready

Now assign the second ThinkJet Printer as the device to print to:

[P]	Printer ? HP2225B
HP2225B(2)	Printer=HP2225B(2)

Select the barcode type to be printed:

[B]	Barcode ? CODE39

Because the default setting indicated is that desired, just press the [ENDLINE]-key.

[ENDLINE]	Barcode=CODE39

Specify low resolution printing:

[R]	Resolution ? (LMH)
[L]	Resolution=L

Select inclusion of check digits in barcode:

[C]	Checkdigit ?
[Y]	Checkdigit=Y

Specify a barcode height of 15mm:

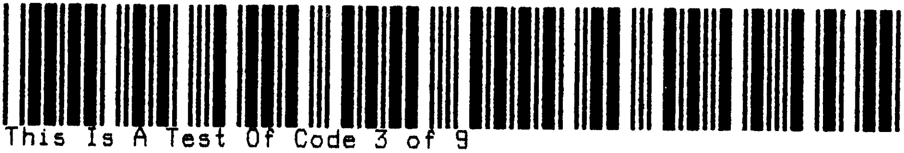
[H]	Height ? 10
15	Height=15

The program is now ready to accept input of the barcode data to be be printed. For example, in order to print the text "**CODE 3 OF 9**" as barcode, perform the following keystrokes:

[RUN]	Start ?
CODE 3 OF 9	Label ? CODE 3 OF 9
This Is A Test Of Code 3 of 9	Working...

The program will now proceed to print the barcode on the second ThinkJet Printer, before beeping to signify completion and returning to the main command level prompt "**BCPrint: Ready**".

CODE 39



CODE 3 OF 9D

EXAMPLE 2: **Printing with the BCP Subprogram:**

In the following example, for the purposes of illustration, we have used the concept of identity numbers for personnel within a large organisation. BCP will therefore be used to print identity barcodes giving the person's ID-number with the person's name printed under the barcode as the descriptive label. To do this, ID-number data will be extracted from a data file stored in the HP-71B's memory and passed to the **BCP** subprogram for printing as Interleaved 2 of 5 barcode, with an included check digit.

Firstly, a data file must be created with records containing information about:

- the person's company identity number. E.g. 10932000001. Where this is perhaps in the format of: Company No.10, Department No.932, and Personnel Numbers 000001 to 999999;
- the person's name and initials (up to 15 characters);
- any additional data such as date of birth.

10 CREATE DATA PERSONS, 5, 35	Create data file called PERSONS, with 5 records each of 35 bytes long to contain one real number and a string variable of 15 characters (plus the required header bytes). See: *HP-71 Reference Manual* for details about using **CREATE**;
20 ASSIGN #1 TO PERSONS	Assign channel number 1 to PERSONS.
30 FOR X=0 TO 4	Sets up input loop;
40 INPUT "Dept:10932, ID:";I @ I=10932000000 + I	Prompts for ID-No. and convert to full format ID-Number;
50 LINPUT "Name: ";N$	Prompts for input of person's name;
60 PRINT #1,X;I,N$	Prints details to data file;
70 NEXT X	Loop to get next input.

Run this program and input the following record information:

ID:	431	Name:	Cawsey, G.D.
	432		Wales, F.C.
	433		Webb, C.P.
	434		Meyers, N.
	435		Badger, T.T.S.

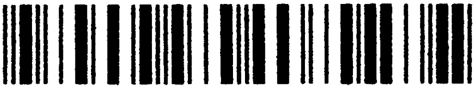
Now key in the following program to print the barcode identity tags:

EDIT BCFILE	Sets up new file;
10 CALL BCFILE @ SUB BCFILE	Runs BCFILE without destroying variables within calling program;
20 DIM B$[8],C1$[8],P$[8],X$[22], B1$[18],F$[20],L1$[15]	Dimensions string variables;
30 INTEGER H,I,R @ REAL B	Dimensions numeric variables;
40 B$="INT25" @ P$="HP2225B" @ C1$="CDC25I" @ H=10 @ R=1	Selects INT25 barcode, ThinkJet printer & driver, check digit ON, height of 10mm & Low resolution;
50 INPUT "Datafile ?";F$ @ IF NOT LEN(F$) THEN 130	Prompts for name of data file & traps erroneous input;
60 ON ERROR GOTO 110 @ ASSIGN #1 TO F$@ I=0	Sets up error trap in case file is non-existent;
70 READ #1,I;B,L1$ @ B1$=STR$(B)	Reads barcode data and descriptive label from data file, and assign ending barcode to starting;
80 CALL BCP(B$,P$,P$,B1$,H,L1$, B1$,R,C1$,"",X$)	Calls BCP subprogram with parameters;
90 IF NOT LEN(X$) THEN I=I+1 @ GOTO 70	If no warning message, loop to get next values from data file.
100 DISP X$ @ GOTO 120	Displays message returned from BCP should subprogram execution be terminated.
110 OFF ERROR @ IF ERRN=54 THEN 50 ELSE DISP ERRM$	If "End of File", go to line 50, else display error message.
120 BEEP 800,.2	
130 END SUB	

The program **BCFILE** can now be **RUN**, or **CALL**ed, to print the barcode identity labels. In response to the prompt **"Datafile ?"**, input the name assigned to the data file (**PERSONS**).

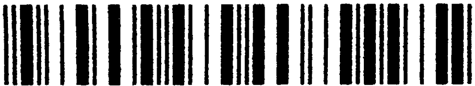
"AUTO"

INT25
109320004313



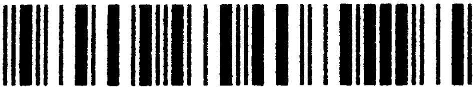
Cawsey, G.D.

INT25
109320004320



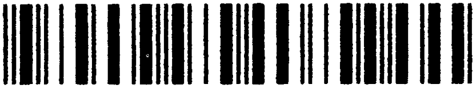
Wales, F.C.

INT25
109320004337



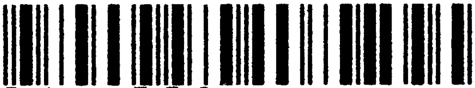
Webb, C.P.

INT25
109320004344



Meyers, N.

INT25
109320004351



Badger, T.T.S.

Chapter 6

ADDITIONAL SUPPORT FEATURES

Support Functions

MSG$

The MSG$ function returns error and system messages from ZENWAND. It may also be used to return messages from the HP-71 mainframe, plug-in modules, or LEX-files present in user memory (RAM). A list of ZENWAND warnings and messages is given in *Appendix D - Warnings and Messages Index.*

The syntax for MSG$ is:

MSG$(*numeric expression*)

The numeric expression passed to MSG$, is an integer with format:

iii fff

where:	iii	is the LEX-ID of the mainframe, plug-in module or LEX-file containing the message, and
	fff	is the particular message number.

ZENWAND possesses the LEX-ID '245' and contains messages '0' through '43'. See also: *Appendix D, Reference Information - Warning and Message Index.*

The LEX-ID of the HP-71 mainframe is '000' and contains warning and error messages in the range '0' through '097'.

The following illustrates the use of MSG$ to display both ZENWAND and HP-71 mainframe messages:

DISP MSG$(245027)	Displays ZENWAND message number 245027: **"Stopped by User"**
A$=MSG$(245*1000+4*10)	Returns to variable A$ the ZENWAND message 245040: **"Working..."**

DISP MSG$(65) Displays the HP-71 mainframe message 000065: **"Line Too Long"**

A=60 @ A$=MSG$(A) Returns to variable A$ the HP-71 mainframe message number 000060: **"Illegal Access"**.

KEYWAIT$

The **KEYWAIT$** function causes the HP-71 to enter a low power state (light sleep), whilst awaiting a key press. Following the pressing of a key, KEYWAIT$ will return the name of the pressed key as a string. For example:

pressing [G] returns "**G**". Note that uppercase G is returned even though the HP-71 may be in lowercase character mode at that time;

pressing [g][G] returns "**g**" Note that the [g]-key is the case shift key. Pressing the [g]-shifted key always returns the lowercase character, even though the HP-71 may be in uppercase character mode at that time;

pressing [f] [S] returns "**fS**" Note that the "f..." indicates that this is an [f]-shifted key.

pressing [g] [ENDLINE] returns "**#150**" The #-symbol indicates that this is a key code, i.e. the system key **CMDS**.

For further details regarding key names and key code numbers see the *HP-71 Owner's Manual - Section 7: Customising the HP-71, Pages 120 to 124.*

Binary Subprograms

In addition to the major ZENWAND keywords and printing program, ZENWAND also contains five binary subprograms to compute check digits of barcode strings. For speed these subprograms have been written in machine language and can be **CALL**ed by the applications programmer from within a BASIC language operating environment.

Each of the five subprograms takes a string parameter, referencing the barcode string, and after processing, returns that string parameter.

The five binary subprograms are:

SUB CDC11(X$) Computes one check digit for the string parameter 'X$', where this is a Code 11 string.

SUB CDC11K(X$) Computes two check digits for a Code 11 string.

SUB CDC25D(X$) Computes check digit for an Industrial 2 of 5 string.

SUB CDC25I(X$) Computes check digit for an Interleaved 2 of 5 string.

SUB CDC39(X$) Computes check digit for a Code 3 of 9 string.

Each subprogram functions by calculating the check digit(s) for the string that is passed to it and then returns that string with the resulting check digit(s) added to it. Should the subprogram detect illegal characters in the passed string, i.e. characters not defined in that particular barcode symbology, then it will return the original string unchanged. The applications programmer can detect this error by comparing the original with the result.

The subprogram **CDC25I** differs from **CDC25D** in that, if the string passed has an even number of characters, then a leading zero digit is added to fill out the resulting string into an even number of characters as defined and required by the symbology.

The subprogram **CDC25D** can also be used to compute the check digit for IAN barcodes.

Appendix A

OWNER'S INFORMATION

Maintenance

With the exception of the Wand's replaceable sapphire tip, the ZENWAND Barcoding System does not contain any user serviceable parts and should not require maintenance during extended normal use. During use, there are several precautions that you should observe to ensure trouble free operation.

CAUTIONS

* Do not place fingers, tools, or other foreign objects into any of the HP-71 ports. Such action could result in minor electrical shock hazard and interference with pacemaker devices worn by some persons. Serious damage to port contacts and internal circuitry could also result.

* Before installing or removing ZENWAND, or any other module, be sure to turn off the HP-71 computer by pressing [f],[OFF].

* If removing a RAM-Module to make a port available, you should execute:

  FREE PORT (*port number***)**

  in order to free the port memory - This is necessary to reset internal pointers in the HP-71. Failure to do so may cause a loss of HP-71 memory contents when you remove the Memory Module. See also: *HP-71 Owner's Manual, Pages 105 to 106.*)

* ZENWAND can only be inserted one way into the HP-71. Do not try to force it into a port as this could damage contacts in either or both devices.

* Protect the HP-71's ports from dust by keeping a port cap installed in any empty port.

* When coiling the ZENWAND lead, avoid coiling the lead too tightly. Also protect the ZENWAND connector from damage and contamination by dust and dirt.

Replaceable Wand Parts

The only user replaceable part in the ZENWAND system is the sapphire wand tip. The sapphire wand tip has been designed to provide superior wear resistance and increased scanning ease in applications where the conventional open tip design would suffer dirt and debris clogging. Should the sapphire tip ever become damaged through misuse or accident, it may be unscrewed by finger pressure. A replacement sapphire tip may be ordered from Hewlett Packard. Users should contact their local Hewlett-Packard Sales Offices for information on ordering spare parts.

The Hewlett-Packard Part Number for the sapphire tip is:

HBCS - 2999

Limited 90-day Warranty

The ZENWAND Barcoding System has been developed by Zengrange Ltd and manufactured to the highest possible standards by Zengrange Ltd and Hewlett Packard. With the exception of software content, ZENWAND is warranted by Zengrange Ltd against defects in materials and workmanship affecting electronic and mechanical performance for a period of 90 days from the date of original purchase. If given as a gift, the warranty is transferred to a new owner for the remainder of that period, provided that proof of purchase date is supplied. During the warranty period, Zengrange Ltd will replace or, at our option, repair a product that proves to be defective, provided that it is returned, shipping prepaid, together with proof of purchase to Zengrange Ltd.

Zengrange Ltd makes no expressed or implied warranty with regard to the software or program material offered, nor to merchantability or fitness of the material for any particular purpose. Software material is made available to the user on an 'as is' basis, with the entire risk as to quality and performance resting with the user. Whilst every effort has been made to eliminate deficiencies, the user (and not Zengrange Ltd, nor any other party) shall bear the entire cost of all necessary correction and all incidential or consequential damages.

The ZENWAND Barcoding System is sold on the basis of specifications as at manufacture. Zengrange Ltd shall be under no obligation to modify or update ZENWAND once manufactured.

Warranty for Consumer Transactions in the United Kingdom

This warranty shall not affect the statutory rights of a consumer whose rights as Buyer and the obligations of Seller are determined by statute.

Limitations of Warranty

The Zengrange warranty does not, and shall not apply if ZENWAND has been damaged by accident, misuse or if attempts have been made to modify the device. No other expressed or implied warranty is given. The repair, or at our option, replacement of ZENWAND is your exclusive remedy.

Shipping for Service

In the unlikely event that ZENWAND proves to be defective, return the device, postage prepaid, to:

Zengrange Ltd,
Greenfield Road,
Leeds,
LS9 8DB,
England.

When returning ZENWAND, be sure to include the following items:

- A Sales Receipt, or other proof of purchase date (if the warranty period has not expired).
- A description of the problem, detailing when and how the problem occurs.

Whether or not ZENWAND is still under the warranty or not, it is the responsibility of the owner to ensure that the device is securely packaged to prevent damage in transit (this is not covered by the Zengrange Ltd. warranty) and that shipping costs to Zengrange Ltd are paid.

Technical Assistance

The keystroke procedures, program material and operating instructions provided for using ZENWAND are supplied with the assumption that the user has a working knowledge of the concepts, terminology, technology and equipment used. The technical assistance of Zengrange Ltd., is limited to explanations of the operating procedures used in the manual.

Zengrange Ltd can provide assistance with additional barcode decoders, encoders and printer device drivers. See also: *Appendix C - Extending the ZENWAND Barcode System.*

Appendix B

COMPATIBILITY BETWEEN ZENWAND AND THE HP-75D

The HP-75 Portable Computer

Barcode reading facilities are also available for the Hewlett-Packard HP-75D Portable Computer via the HP-82718A Expansion Pod, or the HP-82725A Bar Code Reader Module.

Note: Barcode reading is only possible on the HP-75 model 'D', not on the HP-75C. The HP-75C does not possess the necessary rear panel socket for connection of a Hewlett-Packard Barcode Reading Wand.

The facilities available with the HP-82718A Expansion Pod are a similar, although restricted, implimentation of those available with the HP-82725A Bar Code Reader Module for the HP-75D.

Barcode applications written for the HP-75D can also be run on the HP-71. However, there are some minor differences between the BASIC programming languages of the HP-71 and HP-75D. In addition, there are some differences between the functionality of some ZENWAND keywords and the equivalent HP-75 Keywords. For information on conversion of programs between HP-75 BASIC and HP-71 BASIC, see the *HP-71 Owner's Manual - Transforming Files, Page 160.* Major differences in keyword functionality between the two computers are detailed below.

When converting an applications program from the HP-75D to the HP-71, it is likely that the programmer will wish to take full advantage of the added barcode features and enhancements available with ZENWAND, but not included with the HP-75D. To take full advantage of ZENWAND features, such as extended scanning control, adaption of the applications program will be necessary.

Compatability

UPC and EAN Barcodes.

ZENWAND uses just one function, **IANCODE$**, for reading Article Number Codes **UPC** and **EAN**. Program calls to the functions UPCOD$ and EANCOD$ found in an HP-75D program should be changed to **IANCODE$** for running with ZENWAND on the HP-71.

Strings returned by **IANCODE$** will differ from those returned by UPCOD$ on the HP-75D. If an UPC-A label is scanned, ZENWAND returns 13-characters, while the HP-75D returns only 12. If a UPC-E label is scanned, ZENWAND returns 8-characters and the HP-75D only returns 6. See also the explanation of **IANCODE$** given in: *Chapter 3: Reading Barcodes with ZENWAND.*

ZENWAND will decode UPC-E(1) labels, the HP-75D will not.

Special care should be taken when converting HP-75 applications (containing the function UPCOD$) to use ZENWAND-71, since the strings returned by ZENWAND are longer. This may require that strings be dimensioned differently and/or that data file record sizes are altered.

With ZENWAND, short form IAN labels (EAN-8, UPC-E and UPC-E(1)) can be expanded to the full 13-digit representation either automatically, by setting **EXPAND ON**, or manually, by means of **IANEXP$**.

Codabar

Using ZENWAND to scan a Codabar label returns 't', 'n', '*', or 'e' as the stop characters. The HP-75D returns 'a', 'b', 'c' or 'd'.

Code 3 OF 9

As default, when ZENWAND reads a Code 3 of 9 barcode it automatically converts that label into its full ASCII equivalent using the defined Control 3 of 9 conversion standards. However, if **EXPAND ON** is set, then this automatic conversion is inhibited and the returned strings will be the same as those returned by the HP-75D.

General Compatability

The HP-75D can only decode barcodes of maximum 42-characters length. Barcode lengths, when using ZENWAND are only constrained by the amount of available memory in the HP-71. The amount of available memory can be ascertained by means of the **MEM** function. An approximate guide to the maximum barcode length is found by the calculation: **MEM/15.**

Whilst a barcode reading function is executing, the HP-75D will exit the function and return a null string if the Wand switch is pressed, but no barcode is scanned. This can be undesirable because of the problems occasionally associated with key bounce. ZENWAND will not determine that there has been a bad scan unless an attempt to scan a label has been made, thus tapping the ZENWAND switch will not cause a bad scan error.

Check Digits

The functionality of the ZENWAND statement **CDIGIT ON** is slightly different to that of the same statement on the HP-75D. With **CDIGIT ON**, ZENWAND will validate and delete check digit(s) from the scanned barcode. With **CDIGIT ON** on the HP-75D, the check digits are just validated.

With **CDIGIT ON**, if a Code 11 string is scanned, ZENWAND will validate and delete *two* check digits. The HP-75 just validates for *one* check digit when **CDIGIT ON** has been set.

Additional features provided by ZENWAND:

1. Auto-discrimination of barcodes using **BARCODE$** together with automatic identification of barcode type scanned using **BARTYPE?**.
2. Indirect accessing of a barcode decoder by using **ACTIVE$**.
3. Barcode analysing using **WAND$**.
4. Optional one or two check digits for Code 11 barcodes using **CDV11** and **CDV11K**.
5. Optional use of Control 3 of 9 (also called Expanded 3 of 9) using **CTRL39$** and **NORM39$**.
6. Manipulation of IAN codes using **IANEXP$** and **IANSUP$**.
7. Use of a Wand status indicator for determining current system settings and last scan status (**WANDSTAT**).
8. A Livewand mode for using the wand as a remote keyboard device (**LIVEWAND ON/OFF** and **ACTIVE**).
9. Automatic expansion of short form IAN codes and Code 3 of 9 using **EXPAND ON/OFF**.
10. Setting a special key to terminate a barcode reading scan using **ENDSCAN**.
11. Program branching upon receiving a wand interrupt signal (**ON WAND GOSUB/GOTO**).
12. Altering the time-out period for a scan using **WANDTIME**.
13. Check digit computation subprograms

Appendix C

EXTENDING THE ZENWAND BARCODE SYSTEM

The ZENWAND barcode reading, analysing and printing system has been designed in such a way that extensions to the current system can easily be implimented. The repertoire of barcode decoders can be extended, barcode encoders and printer drivers can be added. All of these extensions can be contained in a custom plug-in module, EPROM, or can be provided on mass storage media for copying to a user memory (RAM) based file.

External Barcode Decoders

Additional barcode decoders can be provided in a LEX-file (Language **EX**tension) that interfaces to the main WandRom-file. ZENWAND automatically searches for external decoders, so full integration into the ZENWAND system is possible; e.g. functions such as **BARCODE$**, **ACTIVE**, **BARTYPE?**, etc., will all behave as if the external decoder was a part of the main ZENWAND system. External decoders may be used to read either other industry standard barcode types, such as MATRIX 2 of 5, PLESSEY-Code, etc., or even custom designed barcode types.

Additional Barcode Encoders

The barcode printing system may be enhanced and extended by adding special format DATA-files that specify barcode symbologies other than those currently supplied. The printing system will automatically incorporate and interface with external barcode encoders if they are present. Such barcode encoders may be used to print other industry standard barcode types or custom designed barcode symbologies.

Additional Printer Drivers

Printer drivers are special format files that provide the necessary graphics translation and communications interface between **BCPRINT** (or the **BCP**-subprogram) and the new printing device. Printer drivers may be added for printers other than the HP-2225B ThinkJet Printer, such as the HP-2686A LaserJet Printer or the HP82906A Dot Matrix Printer. Where the resolution and print density permits, the printer may also allow other barcode types to be printed. The printing system will automatically integrate external printer drivers.

If users have a specific requirement for an extension to the ZENWAND system, they should contact Zengrange Ltd, who can supply the additional features.

Appendix D

REFERENCE INFORMATION

Keyword Index

BARCODE READING FUNCTIONS

Return result of scanning operation as a string.

XFN No	Keyword	Page	Description
245001	ACTIVE$	41	Scan & decode barcode of ACTIVE type
245002	BARCODE$	34	Scan & decode barcode using auto-discrimination.
245003	CODABAR$	22	Scan & decode Codabar barcode.
245004	CODE11$	21	Scan & decode Code 11 barcode.
245005	CODE39$	22	Scan & decode Code 39 barcode.
245006	IANCODE$	24	Scan & decode an International Article Number barcode.
245007	IND25$	23	Scan & decode an Industrial 2 of 5 barcode.
245008	INT25$	23	Scan & decode an Interleaved 2 of 5 barcode.
245009	WAND$	36	Scan & decode any barcode pattern into bar/space timing information.

VERIFICATION FUNCTIONS

Takes string parameter of barcode to be verified, computes check digit and returns an integer of '1' if correct, or '0' if incorrect.

XFN No	Keyword	Page	Description
245011	CDV11	29	Verify a Code 11 barcode string that contains one check digit.
245012	CDV11K	29	Verify a Code 11 barcode string that contains two check digits.
245013	CDV25D	29	Verify an Industrial 2 of 5 barcode string.
245014	CDV25I	29	Verify an Interleaved 2 of 5 barcode string.
245015	CVD39	29	Verify a Code 3 of 9 type barcode string

BARCODE SUPPORT FUNCTIONS

XFN No	Keyword	Page	Description
245010	BARTYPE?	34	Return the XFN-Number of the last barcode type decoder used.
245016	CTRL39$	27	Convert a normal Code 3 of 9 string into its full ASCII, or Control 3 of 9, representation.
245017	IANEXP$	28	Expand an EAN-8, UPC-E or E(1) label into EAN-13 or UPC-A respectively.
245018	IANSUP$	28	Zero-suppress an UPC-A label.
245019	NORM39$	27	Convert a full ASCII, or Control 3 of 9, string into normal Code 3 of 9 representation.
245020	WANDSTAT	31	Return the Wand status indicator byte.

SUPPORT STATEMENTS

	Keyword	Page	Description
245021	ACTIVE	40	Selects barcode type for use by LIVEWAND and ACTIVE$.
245022	CDIGIT ON/OFF	28	Controls automatic check digit verification and deletion for barcode reading functions.
245023	ENDSCAN	31	Specified a key that, when pressed, will terminate a barcode scanning operation.
245024	EXPAND ON/OFF	26	Enables automatic expansion of barcode. E.g. Converts between short form IAN and full 13-digit IAN, or contracts/expands Code 3 of 9 between Normal and Control 3 of 9 data.
245025	LIVEWAND ON/OFF	39	Allows keyboard to be used as an alternative input device to ZENWAND.
245029	WANDTIME	30	Controls time-out setting for barcode reading functions.
245027	ON WAND GOSUB/GOTO	43	Enables program branching upon receipt of interrupt from the Wand.
245026	OFF WAND	44	Disables program branching within the current environment.

ADDITIONAL SUPPORT FUNCTIONS

XFN No	Keyword	Page	Description
82001	KEYWAIT$	64	Causes HP-71 to enter low power state whilst awaiting key to be pressed. Returns name of pressed key as a string.
82003	MSG$	63	Returns error, system and warning messages from LEX-files from the HP-71, plug-in modules, or in user memory.

CHECK DIGIT COMPUTATION

Compute check digit(s) for passed string. If data was valid, returns string inclusive of check digit(s). If invalid data found in string returns unchanged original string.

SUB PROGRAMS

XFN No	Keyword	Page	Description
	CDC11	65	Computes **one** check digit for a Code 11 string.
	CDC11K	65	Compute **two** check digits for a Code 11 string.
	CDC25D	65	Compute check digit for an Industrial 2 of 5 string.
	CDC25I	65	Compute check digit for an Interleaved 2 of 5 string.
	CDC39	65	Compute check digit for a Code 3 of 9 string.

Warning and Message Index

The following warnings and system messages are reported by ZENWAND. ZENWAND messages are contained in a LEX-file with the ID '245' and message numbers ranging from '0' to '43'. In addition, ZENWAND makes extensive use of HP-71 mainframe messages. These are fully detailed in the *HP-71 Reference Manual - Errors, Warnings and System Messages, Pages 378 to 392.*

Number	Message	Condition	
245000	**WAND**	Name of ZENWAND Device	
245001	**BCPrint:**	*Prompt:*	
245002	**#PBFHRCXYQ**	*Options:*	Available BCPrint key selections
245003	**CODABAR**	}	
245004	**CODE11**	} *Barcode*	
245005	**CODE39**	} *type*	
245006	**IANCODE**	} *messages*	
245007	**IND25**	}	
245008	**INT25**	}	
245009	**Barcode Too Long**	*Warning*:	Barcode too long to print.
245010	**BCPrint: Ready**	*Prompt*:	BCPrint main command level.
245011	**BCPrint: Done**	*Message*:	BCPrint program terminated.
245012	**Encode Failure**	*Warning*:	BCPrint unable to encode characters in the barcode string.
245013	**Illegal Char '**	*Warning*:	Character not defined by chosen barcode symbology was found in string for BCPrint.
245014	**Invalid Barcode**	*Warning*:	Barcode encoder specified not present.
245015	**Invalid Device**	*Warning*:	Device specified is not capable of printing barcodes with the chosen Printer Driver.
245016	**Invalid Height**	*Warning*:	Specified barcode height exceeds limits of <5 and >150 mm.
245017	**Invalid Printer**	*Warning*:	Specified device is not a valid PRINTER IS device.
245018	**No Barcode**	*Warning*:	No header file defining the barcode symbology for the barcode type selected is present in memory.
245019	**No Data**	*Warning*:	Specified file defining the barcode encodation pattern is not present in memory.
245020	**No Driver**	*Warning*:	No Device Driver of the name given is present in memory.

245021	**No HPIL**	*Warning*:	No HP-IL module is connected to the HP-71.
245022	**No Printer Assigned**	*Warning*:	No HP-IL device has been assigned as a PRINTER IS device.
245023	**No WAND**	*Warning*:	No ZENWAND Barcode Reading Device is connected to the HP-71.
245024	**Printer Error**	*Warning*:	An error condition has occurred with the Printer device.
245025	**Resolution Too High**	*Warning*:	Attempted to print barcode at a resolution outside capability of assigned device.
245026	**Start/Stop**	}	
245027	**Stopped by User**	}	
245028	**Barcode**	}	
245029	**Checkdigit**	}	
245030	**Control 39**	}	
245031	**Driver**	}	
245032	**End**	}	
245033	**Height**	} *Building*	
245034	**Label**	} *segments*	
245035	**No**	} *for*	
245036	**Printer**	} *ZENWAND*	
245037	**Resolution**	} *prompts*	
245038	**Start**	}	
245039	**Stop**	}	
245040	**Working...**	}	
245041	**YN**	}	
245042	**0123456789**	}	
245043	**Copyright (c) Zengrange, 1985**		

Memory Requirements

This section describes the HP-71 main user-memory requirements of the ZENWAND Barcode System. Programmers should be aware of this additional requirement when designing large application programs.

General Requirement: ZENWAND takes **36** bytes of main memory when plugged into the HP-71. This includes both memory required by the HP-71 to configure the plug-in device, and also memory buffer space set aside by ZENWAND for its own purposes.

LIVEWAND mode: When using LIVEWAND mode, further memory is also required by ZENWAND. **24** bytes are temporarily used whilst the HP-71 is awaiting LIVEWAND input from the Wand. An additional **one** byte per character will be used for storage of LIVEWAND mode data input. The 24-bytes required by LIVEWAND mode are restored to main memory once the scanning operation is complete, but the one byte per character input during LIVEWAND scanning will only be restored upon scanning another barcode in LIVEWAND mode, or following a reset of the HP-71.

Bibliography and References

For General Information on Barcodes:

1. "Elements of a Bar Code System". HP-Application Note 1013.

   Hewlett Packard Components, 640 Page Mill Road, Palo Alto, California, CA 94304, USA.

2. "Bar Code Symbology" by David Allias.

   Intermec, 4405 Russell Road, P.O.Box 360602, Lynnwood, WA 98046-9702, USA.

For Information on Specific Barcode Types:

Code 3 of 9 **Interleaved 2 of 5** **Code 11** **Codabar**	Automatic Identification Manufacturers Inc., 1326 Freeport Road, Pittsburgh, PA 15238, USA.
Code 3 of 9 **Interleaved 2 of 5** **Codabar**	American National Standards Institute (ANSI), 1430 Broadway, New York, USA.
UPC	Uniform Product Code Council Inc., 7051 Corporate Way, Suite 201, Dayton, Ohio 45459, USA.
EAN	International Article Numbering Association, Rue des Colonies 28, Bte 8-B-1000, Bruxelles, Belgium.

For Information on using ZENWAND for Barcode Systems:

Zengrange Ltd, Greenfield Road, Leeds, England, LS9 8DB.

Hewlett-Packard, PCD, 1000 N.E. Circle Blvd., Corvallis, OR 97330, USA.

Hewlett-Packard France, 5 Avenue Raymond Chanas, F-38320 Eybens, Grenoble,
France

Hewlett-Packard UK (Pinewood), GB-Nine Mile Ride, Wokingham, Berks, RG11 3LL,
England.

SUBJECT INDEX

Leeds Booklet Printing Company
LS5 3AE England
0532 744022 .