0. Дополнение переводчика.

Если формируемый Вами штрихкод включает контрольную цифру то внимательно читайте руководство по конкретному формату штрихкода - надо ли подавать на вход Zint код, включающий контрольную цифру, или без нее.

5. Использование API

Zint написан на языке C и имеет API для использования программами на языке C. Благодаря theunknownones, есть возможно использовать zint в виде компонента Delphi.
API библиотеки libzint очень похоже на тот, что используется пакетом GNU Barcode. Это позволяет легко переходить от GNU Barcode к Zint. Однако имена функций и параметров, используемые в Zint и GNU штрих-коде, различаются. Это позволяет спользовать одном приложении оба пакета без каких-либо конфликтов.

5.1 Создание и удаление штрих-кодов

Штрих-коды, которые создает Zint, находятся в структуре zint_symbol , определенной в zint.h. Эти штрих-коды создаются функцией ZBarcode_Create() и удаляются с помощью функции ZBarcode_Delete(). К примеру, данный код создает и затем удаляет штрих-код:

```
#include <stdio.h>
#include <zint.h>
int main()
{
      struct zint_symbol *my_symbol;
      my_symbol = ZBarcode_Create();
      if(my_symbol != NULL)
      {
      printf("Symbol successfully created!\n");
      }
      ZBarcode_Delete(my_symbol);
      return 0;
}
```

При компиляции этот код должен быть связан с библиотекой libzint с параметром -lzint:

```
gcc -o simple simple.c -lzint
```

5.2 Кодирование и сохранение в файл

Чтобы закодировать данные в штрих-код, используется функция ZBarcode_Encode(). Чтобы записать штрих-код в файл, используется функция ZBarcode_Print(). К примеру, данный код берет параметр из командной строки и выводит штрихкод в формате Code 128 в файл out.png (формат, соответственно, PNG) в текущем рабочем каталоге:

```
#include <stdio.h>
#include <zint.h>
int main(int argc, char **argv)
{
      struct zint_symbol *my_symbol;
      my_symbol = ZBarcode_Create();
      ZBarcode_Encode(my_symbol, argv[1], 0);
      ZBarcode_Print(my_symbol, 0);
      ZBarcode_Delete(my_symbol);
      return 0;
}
```

Это также может быть сделано в один этап с использованием функции ZBarcode_Encode_and_Print(), как показано в следующем примере:

```
#include <stdio.h>
#include <zint.h>
int main(int argc, char **argv)
{
      struct zint_symbol *my_symbol;
      my_symbol = ZBarcode_Create();
      ZBarcode_Encode_and_Print(my_symbol, argv[1], 0, 0);
      ZBarcode_Delete(my_symbol);
      return 0;
}
```

Строка на входе должна быть в кодировкеUnicode.

5.3 Функции кодирования и печати

Определены следующие функции для кодирования и печати штрих-кодов:

```
int ZBarcode_Encode(struct zint_symbol *symbol, unsigned char *input, int
length);
int ZBarcode_Encode_File(struct zint_symbol *symbol, char *filename);
int ZBarcode_Print(struct zint_symbol *symbol, int rotate_angle);
int ZBarcode_Encode_and_Print(struct zint_symbol *symbol, unsigned char *input,
int length, int rotate_angle);
int ZBarcode_Encode_File_and_Print(struct zint_symbol *symbol, char *filename,
int rotate_angle);
```

В этих фунциях параметр "length" может быт использован для установки длины вводной строки. Это позволяет кодировать NULL (ASCII 0) символы в тех штрихкодах, где это допустими. Значение 0 отключает эту возможность и Zint будет кодировать данные до первого символа NULL в строке ввода.

Параметр "rotate_angle" позволяет повернуть картинку во время ее вывода в формате PNG. Допустимые значения: 0, 90, 180 и 270.

Функции ZBarcode_Encode_File() and ZBarcode_Encode_File_and_Print() могут быть использованы, чтобы закодировать данные, получаемые из файла, имя файла указывается в строке "filename".

5.4 Буферизация штрих-кодов в память

В дополнение к сохранению штрих-кодов в файлы, Zint позволяет получить доступ к растровому изображению штрихкода в памяти.
Для этого используются следующие функции:

```
int ZBarcode_Buffer(struct zint_symbol *symbol, int rotate_angle);
int ZBarcide_Encode_and_Buffer(struct zint_symbol *symbol, unsigned char
*input,
int length, int rotate_angle);
int ZBarcode_Encode_File_and_Buffer(struct zint_symbol *symbol, char *filename,
int rotate_angle);
```

Параметры такие же, как и описано выше. Разница в том, что вместо сохранения картинки в файл, она помещается в массив (char). Указатель "bitmap" устанавливается на первой ячейке массива; значения "barcode_width" и "barcode_height" указывают на размер полученного изображения в пикселях. Вращение и параметры цвета могут быть использованы во время буферизации так же, как и при сохранении в PNG файл.

Растровые данные могут быть извлечены из массива с помощью метода, показанного в следующем примере. render_pixel() следует понимать, как функцию для рисования точек на экране, осуществляемая приложением:

```
int row, col, i = 0;
int red, blue, green;
for(row = 0; row < my_symbol->bitmap_height; row++) {
      for(column = 0; column < my_symbol->bitmap_width; column++) {
      red = my_symbol->bitmap[i];
      green = my_symbol->bitmap[i + 1];
      blue = my_symbol->bitmap[i + 2];
      render_pixel(row, column, red, green, blue);
      i += 3;
      }
}
```

5.5 Настройки

Пример выше, конечно очень прост, но позволяет ли формировать штриходы формата Code 128 и сохранять их в jut.png. Конечно, в реальном применении это все должно настраиваться. Это обеспечивается изменением значений элементов структуры zint_symbol в между этапами создания и кодирования. Структура zint_symbol состоит из следующих переменных:

Имя переменной	**Тип**	**Смысл**	**Значение по умолчанию**
symbology	integer	Вид штрихкода (см. раздел 5.7).	BARCODE_ CODE128
height	integer	Высота изображения (см. примечание)	50
whitespace_width	integer	Ширина пробела.	0
border_width	integer	Ширина границы.	0
output_options	integer	Параметры рамки и границ (см. раздел 5.8).	(none)
fgcolour	character string	Цвет элементов штрих-кода. (шестнадцатеричная RGB строка) Должен содержать 6 символов и заканчиваться '\0'.	"000000"
bgcolour	character string	Цвет фона штрих-кода. (шестнадцатеричная RGB строка) Должен содержать 6 символов и заканчиваться '\0'.	"ffffff"
outfile	character string	Содержит имя файла для вывода полученного изображения. Файл должен иметь расширение .png, .eps или .svg.	"out.png"
option_1	integer	Зависит от выбранного виды штрихкода	(automatic)
option_2	integer	Зависит от выбранного виды штрихкода	(automatic)
option_3	integer	Зависит от выбранного виды штрихкода	(automatic)
scale	float	Масштаб	1.0
input_mode	integer	Кодировка входных данных (см. раздел 5.9)	BINARY_MODE

primary	character string	Первичные данные для некоторых штрих-кодов.	NULL
text	unsigned character string	Понятный человеку текст, который обычно состоит из входных данный и одной или нескольких контрольных цифр. Используется кодировка UTF-8.	NULL
rows	integer	Количество строк в символе штрихкода (не путайте с количеством строк в изображении штрихкода)	(output only)
width	integer	Ширина штрихкода	(output only)
encoding_data	array of character strings	Представление закодированных данных.	(output only)
row_height	array of integers	Высоты строк штрихкода	(output only)
errtxt	character string	Сообщение об ошибке (при ее возникновении).	(output only)
bitmap	pointer to character array	Указатель на растровое изобрадение штрихкода.	(output only)
bitmap_width	integer	Ширина сохраненного растрового изображения (в пикселях).	(output only)
bitmap_height	integer	Высота сохраненного растрового изображения (в пикселях).	(output only)
Свойства ниже добавлены в версии 1.1. Доступны только в Ocvita Barcode ActiveX			
code_page	long	Кодовая страницы. 65001 - UTF-8 1251 -windows-1251 полный список см. ниже в разделе 5.A	65001
setting_1	long	Дополнительная настройка. См в разделе 5.A	0

Примечание. Это значение игнорируется в штрих-кодах Australia Post 4-State, PostNet, PLANET, USPS OneCode, RM4SCC, PDF417, Data Matrix, Maxicode, QR Code, GS1 DataBar-14 Stacked, PDF417 и MicroPDF417 – которые имеют фиксированную высоту.

Чтобы изменить эти значения, используют синтаксис, показанный на примере ниже Результат выполнения этого кода такой же, как и в предыдущем примере, но штрихкод будет выше и зеленого цвета.

```
#include <stdio.h>
#include <zint.h>
#include <string.h>
int main(int argc, char **argv)
{
      struct zint_symbol *my_symbol;
      my_symbol = ZBarcode_Create();
      strcpy(my_symbol->fgcolour, "00ff00");
      my_symbol->height = 400;
      ZBarcode_Encode_and_Print(my_symbol, argv[1], 0, 0);
      ZBarcode_Delete(my_symbol);
      return 0;
}
```

5.6 Обработка ошибок

В случае возникновения ошибок при кодировании, вызывающему приложению возвращается код ошибки (тип integer). Кроме того, подробное сообщение об ошибке помещается в переменную errtxt. Коды ошибок, которые могут быть возвращены Zint, приведены в следующей таблице:

Возвращаемое значение	Суть ошибки
WARN_INVALID_OPTION	Одно из значений в zint_struct было установлено не верно, но Zint сделал предположение, каким оно должно быть и в соответствие с этим сгенерировал штрих-код.
ERROR_TOO_LONG	Исходные данные слишком длинные или слишком короткие для выбранного вида штрихкода. Штрих-код создан не был.
ERROR_INVALID_DATA	Данные для кодирования включают в себя символы, недопустимые выбранном виде штрихода (например, буквы в формате EAN). Штрих-код создан не был.
ERROR_INVALID_CHECK	ISBN с неверно введенной контрольной цифрой. Штрих-код создан не был. (прим. перевод. - видимо не только ISBN)
ERROR_INVALID_OPTION	Одно из значений в zint_struct было установлено не верно, и Zint не смог предположить, каким ему следовало бы быть. Штрих-код создан не был.
ERROR_ENCODING_PROBLEM	Произошел сбой в процессе кодирования данных. Видимо ошибка в самом Zint. Пожалуйста, свяжитесь с разработчиком, если Вы столкнулись с этой ошибкой.
ERROR_FILE_ACCESS	Zint не удалось записать указанный файл. Обычно это связано с проблемой прав доступа к файлам.
ERROR_MEMORY	Zint не хватило памяти. (видимо в вашей системе ее совсем мало)

Пример обработки сообщений об ошибках

```
#include <stdio.h>
#include <zint.h>
#include <string.h>
int main(int argc, char **argv)
{
      struct zint_symbol *my_symbol;
      int error = 0;
      my_symbol = ZBarcode_Create();
      strcpy(my_symbol->fgcolour, "nonsense");
      error = ZBarcode_Encode_and_Print(my_symbol, argv[1], 0, 0);
      if(error != 0)
      {
      /* some error occurred */
      printf("%s\n", my_symbol->errtxt);
      }
      if(error > WARN_INVALID_OPTION)
      {
      /* stop now */
      ZBarcode_Delete(my_symbol);
      return 1;
      }
      /* otherwise carry on with the rest of the application */
      ZBarcode_Delete(my_symbol);
      return 0;
```

```
}
```

Результатом будет сообщение об ошибке:

```
error: malformed foreground colour target
```

5.7 Указание формата

Формат штрихкода может быть задан число или именем, как показано в следующей таблице.

symbol->symbology= BARCODE_LOGMARS;

означает то же, что и

symbol->symbology = 50;

Значение	**Имя**	**Вид штрихкода (symbology)**
1	BARCODE_CODE11	Code 11
2	BARCODE_C25MATRIX	Standard Code 2 of 5
3	BARCODE_C25INTER	Interleaved 2 of 5
4	BARCODE_C25IATA	Code 2 of 5 IATA
6	BARCODE_C25LOGIC	Code 2 of 5 Data Logic
7	BARCODE_C25IND	Code 2 of 5 Industrial
8	BARCODE_CODE39	Code 3 of 9 (Code 39)
9	BARCODE_EXCODE39	Extended Code 3 of 9 (Code 39+)
13	BARCODE_EANX	EAN
16	BARCODE_EAN128	GS1-128
18	BARCODE_CODABAR	Codabar
20	BARCODE_CODE128	Code 128 (automatic subset switching)
21	BARCODE_DPLEIT	Deutshe Post Leitcode
22	BARCODE_DPIDENT	Deutshe Post Identcode
23	BARCODE_CODE16K	Code 16K
24	BARCODE_CODE49	Code 49
25	BARCODE_CODE93	Code 93
28	BARCODE_FLAT	Flattermarken
29	BARCODE_RSS14	GS1 DataBar-14
30	BARCODE_RSS_LTD	GS1 DataBar Limited
31	BARCODE_RSS_EXP	GS1 DataBar Expanded
32	BARCODE_TELEPEN	Telepen Alpha
34	BARCODE_UPCA	UPC A
37	BARCODE_UPCE	UPC E
40	BARCODE_POSTNET	PostNet
47	BARCODE_MSI_PLESSEY	MSI Plessey
49	BARCODE_FIM	FIM

50	BARCODE_LOGMARS	LOGMARS
51	BARCODE_PHARMA	Pharmacode One-Track
52	BARCODE_PZN	PZN
53	BARCODE_PHARMA_TWO	Pharmacode Two-Track
55	BARCODE_PDF417	PDF417
56	BARCODE_PDF417TRUNC	PDF417 Truncated
57	BARCODE_MAXICODE	Maxicode
58	BARCODE_QRCODE	QR Code
60	BARCODE_CODE128B	Code 128 (Subset B)
63	BARCODE_AUSPOST	Australia Post Standard Customer
66	BARCODE_AUSREPLY	Australia Post Reply Paid
67	BARCODE_AUSROUTE	Australia Post Routing
68	BARCODE_AUSREDIRECT	Australia Post Redirection
69	BARCODE_ISBNX	ISBN (EAN-13 with verification stage)
70	BARCODE_RM4SCC	Royal Mail 4 State (RM4SCC)
71	BARCODE_DATAMATRIX	Data Matrix
72	BARCODE_EAN14	EAN-14
75	BARCODE_NVE18	NVE-18
76	BARCODE_JAPANPOST	Japanese Post
77	BARCODE_KOREAPOST	Korea Post
79	BARCODE_RSS14STACK	GS1 DataBar-14 Stacked
80	BARCODE_RSS14STACK_OMNI	GS1 DataBar-14 Stacked Omnidirectional
81	BARCODE_RSS_EXPSTACK	GS1 DataBar Expanded Stacked
82	BARCODE_PLANET	PLANET
84	BARCODE_MICROPDF417	MicroPDF417
85	BARCODE_ONECODE	USPS OneCode
86	BARCODE_PLESSEY	Plessey Code
87	BARCODE_TELEPEN_NUM	Telepen Numeric
89	BARCODE_ITF14	ITF-14
90	BARCODE_KIX	Dutch Post KIX Code
92	BARCODE_AZTEC	Aztec Code
93	BARCODE_DAFT	DAFT Code
97	BARCODE_MICROQR	Micro QR Code
98	BARCODE_HIBC_128	HIBC Code 128
99	BARCODE_HIBC_39	HIBC Code 39
102	BARCODE_HIBC_DM	HIBC Data Matrix
104	BARCODE_HIBC_QR	HIBC QR Code
106	BARCODE_HIBC_PDF	HIBC PDF417
108	BARCODE_HIBC_MICPDF	HIBC MicroPDF417
112	BARCODE_HIBC_AZTEC	HIBC Aztec Code

128	BARCODE_AZRUNE	Aztec Runes
129	BARCODE_CODE32	Code 32
130	BARCODE_EANX_CC	Composite Symbol with EAN linear component
131	BARCODE_EAN128_CC	Composite Symbol with GS1-128 linear component
132	BARCODE_RSS14_CC	Composite Symbol with GS1 DataBar-14 linear component
133	BARCODE_RSS_LTD_CC	Composite Symbol with GS1 DataBar Limited component
134	BARCODE_RSS_EXP_CC	Composite Symbol with GS1 DataBar Extended component
135	BARCODE_UPCA_CC	Composite Symbol with UPC A linear component
136	BARCODE_UPCE_CC	Composite Symbol with UPC E linear component
137	BARCODE_RSS14STACK_CC	Composite Symbol with GS1 DataBar-14 Stacked component
138	BARCODE_RSS14_OMNI_CC	Composite Symbol with GS1 DataBar-14 Stacked Omnidirectional component
139	BARCODE_RSS_EXPSTACK_CC	Composite Symbol with GS1 DataBar Expanded Stacked component
140	BARCODE_CHANNEL	Channel Code
141	BARCODE_CODEONE	Code One
142	BARCODE_GRIDMATRIX	Grid Matrix

5.8 Добавление рамки и границы

Рамки и границы добавляются с помощью свойства output_options структуры zint_symbol. Надо просто присвоить переменной output_options значение из следующей таблицы:

Значение	Эффект
0	Нет ячеек и границ.
BARCODE_BIND	Граница сверху и снизу штрих-кода и между строк, если речь идет о составных символах.
BARCODE_BOX	Добавить рамку вокруг штрих-кода его полей.

Примечание. Кроме Code 16k и ITF-14.

5.9 Кодировка входных символов

Свойство input_mode позволяет установить способ кодирования исходных данных. Допустимые значения приведены в таблице ниже:

Значение	Результат
DATA_MODE	Используется полный диапазон ASCII (понимается как Latin-1 или двоичные данные).
UNICODE_MODE	Входные данные в формате UTF-8
GS1_MODE	Кодирует GS1 данные с использованием FNC1 символов.

5.10 Проверка доступности формата

В API имеется функция:

```
int ZBarcode_ValidID(int symbol_id);
```

Эта функция позволяет проверить: доступен ли данный формат. Возвращенное ненулевое значение указывает на то, что данный формат доступен. Например:

```
if(ZBarcode_ValidID(BARCODE_PDF417) != 0)
{
     printf("PDF417 available");
}
else
{
     printf("PDF417 not available");
}
```

5.A. Дополнительные свойства, реализованные в Ocvita Barcode

Кодовая страница (code_page) (реализовано в версии 1.1)

Переданное значение будет преобразовано в соответствующую кодировку. Для преобразования используется функция WideCharToMultiByte (фактически code_page это ее первый параметр).

Значение	.NET Name	Описание
037	IBM037	IBM EBCDIC US-Canada
437	IBM437	OEM United States
500	IBM500	IBM EBCDIC International
708	ASMO-708	Arabic (ASMO 708)
709		Arabic (ASMO-449+, BCON V4)
710		Arabic - Transparent Arabic
720	DOS-720	Arabic (Transparent ASMO); Arabic (DOS)
737	ibm737	OEM Greek (formerly 437G); Greek (DOS)
775	ibm775	OEM Baltic; Baltic (DOS)
850	ibm850	OEM Multilingual Latin 1; Western European (DOS)
852	ibm852	OEM Latin 2; Central European (DOS)
855	IBM855	OEM Cyrillic (primarily Russian)
857	ibm857	OEM Turkish; Turkish (DOS)
858	IBM00858	OEM Multilingual Latin 1 + Euro symbol
860	IBM860	OEM Portuguese; Portuguese (DOS)
861	ibm861	OEM Icelandic; Icelandic (DOS)
862	DOS-862	OEM Hebrew; Hebrew (DOS)
863	IBM863	OEM French Canadian; French Canadian (DOS)
864	IBM864	OEM Arabic; Arabic (864)
865	IBM865	OEM Nordic; Nordic (DOS)
866	cp866	OEM Russian; Cyrillic (DOS)
869	ibm869	OEM Modern Greek; Greek, Modern (DOS)
870	IBM870	IBM EBCDIC Multilingual/ROECE (Latin 2); IBM EBCDIC Multilingual Latin 2
874	windows-874	ANSI/OEM Thai (same as 28605, ISO 8859-15); Thai (Windows)
875	cp875	IBM EBCDIC Greek Modern

932	shift_jis	ANSI/OEM Japanese; Japanese (Shift-JIS)
936	gb2312	ANSI/OEM Simplified Chinese (PRC, Singapore); Chinese Simplified (GB2312)
949	ks_c_5601-1987	ANSI/OEM Korean (Unified Hangul Code)
950	big5	ANSI/OEM Traditional Chinese (Taiwan; Hong Kong SAR, PRC); Chinese Traditional (Big5)
1026	IBM1026	IBM EBCDIC Turkish (Latin 5)
1047	IBM01047	IBM EBCDIC Latin 1/Open System
1140	IBM01140	IBM EBCDIC US-Canada (037 + Euro symbol); IBM EBCDIC (US-Canada-Euro)
1141	IBM01141	IBM EBCDIC Germany (20273 + Euro symbol); IBM EBCDIC (Germany-Euro)
1142	IBM01142	IBM EBCDIC Denmark-Norway (20277 + Euro symbol); IBM EBCDIC (Denmark-Norway-Euro)
1143	IBM01143	IBM EBCDIC Finland-Sweden (20278 + Euro symbol); IBM EBCDIC (Finland-Sweden-Euro)
1144	IBM01144	IBM EBCDIC Italy (20280 + Euro symbol); IBM EBCDIC (Italy-Euro)
1145	IBM01145	IBM EBCDIC Latin America-Spain (20284 + Euro symbol); IBM EBCDIC (Spain-Euro)
1146	IBM01146	IBM EBCDIC United Kingdom (20285 + Euro symbol); IBM EBCDIC (UK-Euro)
1147	IBM01147	IBM EBCDIC France (20297 + Euro symbol); IBM EBCDIC (France-Euro)
1148	IBM01148	IBM EBCDIC International (500 + Euro symbol); IBM EBCDIC (International-Euro)
1149	IBM01149	IBM EBCDIC Icelandic (20871 + Euro symbol); IBM EBCDIC (Icelandic-Euro)
1200	utf-16	Unicode UTF-16, little endian byte order (BMP of ISO 10646); available only to managed applications
1201	unicodeFFFE	Unicode UTF-16, big endian byte order; available only to managed applications
1250	windows-1250	ANSI Central European; Central European (Windows)
1251	windows-1251	ANSI Cyrillic; Cyrillic (Windows)
1252	windows-1252	ANSI Latin 1; Western European (Windows)
1253	windows-1253	ANSI Greek; Greek (Windows)
1254	windows-1254	ANSI Turkish; Turkish (Windows)
1255	windows-1255	ANSI Hebrew; Hebrew (Windows)
1256	windows-1256	ANSI Arabic; Arabic (Windows)
1257	windows-1257	ANSI Baltic; Baltic (Windows)
1258	windows-1258	ANSI/OEM Vietnamese; Vietnamese (Windows)
1361	Johab	Korean (Johab)
10000	macintosh	MAC Roman; Western European (Mac)
10001	x-mac-japanese	Japanese (Mac)
10002	x-mac-chinesetrad	MAC Traditional Chinese (Big5); Chinese Traditional (Mac)
10003	x-mac-korean	Korean (Mac)
10004	x-mac-arabic	Arabic (Mac)
10005	x-mac-hebrew	Hebrew (Mac)
10006	x-mac-greek	Greek (Mac)
10007	x-mac-cyrillic	Cyrillic (Mac)
10008	x-mac-chinesesimp	MAC Simplified Chinese (GB 2312); Chinese Simplified (Mac)
10010	x-mac-romanian	Romanian (Mac)
10017	x-mac-ukrainian	Ukrainian (Mac)
10021	x-mac-thai	Thai (Mac)
10029	x-mac-ce	MAC Latin 2; Central European (Mac)
10079	x-mac-icelandic	Icelandic (Mac)

10081	x-mac-turkish	Turkish (Mac)
10082	x-mac-croatian	Croatian (Mac)
12000	utf-32	Unicode UTF-32, little endian byte order; available only to managed applications
12001	utf-32BE	Unicode UTF-32, big endian byte order; available only to managed applications
20000	x-Chinese_CNS	CNS Taiwan; Chinese Traditional (CNS)
20001	x-cp20001	TCA Taiwan
20002	x_Chinese-Eten	Eten Taiwan; Chinese Traditional (Eten)
20003	x-cp20003	IBM5550 Taiwan
20004	x-cp20004	TeleText Taiwan
20005	x-cp20005	Wang Taiwan
20105	x-IA5	IA5 (IRV International Alphabet No. 5, 7-bit); Western European (IA5)
20106	x-IA5-German	IA5 German (7-bit)
20107	x-IA5-Swedish	IA5 Swedish (7-bit)
20108	x-IA5-Norwegian	IA5 Norwegian (7-bit)
20127	us-ascii	US-ASCII (7-bit)
20261	x-cp20261	T.61
20269	x-cp20269	ISO 6937 Non-Spacing Accent
20273	IBM273	IBM EBCDIC Germany
20277	IBM277	IBM EBCDIC Denmark-Norway
20278	IBM278	IBM EBCDIC Finland-Sweden
20280	IBM280	IBM EBCDIC Italy
20284	IBM284	IBM EBCDIC Latin America-Spain
20285	IBM285	IBM EBCDIC United Kingdom
20290	IBM290	IBM EBCDIC Japanese Katakana Extended
20297	IBM297	IBM EBCDIC France
20420	IBM420	IBM EBCDIC Arabic
20423	IBM423	IBM EBCDIC Greek
20424	IBM424	IBM EBCDIC Hebrew
20833	x-EBCDIC-KoreanExtended	IBM EBCDIC Korean Extended
20838	IBM-Thai	IBM EBCDIC Thai
20866	koi8-r	Russian (KOI8-R); Cyrillic (KOI8-R)
20871	IBM871	IBM EBCDIC Icelandic
20880	IBM880	IBM EBCDIC Cyrillic Russian
20905	IBM905	IBM EBCDIC Turkish
20924	IBM00924	IBM EBCDIC Latin 1/Open System (1047 + Euro symbol)
20932	EUC-JP	Japanese (JIS 0208-1990 and 0121-1990)
20936	x-cp20936	Simplified Chinese (GB2312); Chinese Simplified (GB2312-80)
20949	x-cp20949	Korean Wansung
21025	cp1025	IBM EBCDIC Cyrillic Serbian-Bulgarian
21027		(deprecated)
21866	koi8-u	Ukrainian (KOI8-U); Cyrillic (KOI8-U)
28591	iso-8859-1	ISO 8859-1 Latin 1; Western European (ISO)
28592	iso-8859-2	ISO 8859-2 Central European; Central European (ISO)
28593	iso-8859-3	ISO 8859-3 Latin 3
28594	iso-8859-4	ISO 8859-4 Baltic
28595	iso-8859-5	ISO 8859-5 Cyrillic

28596	iso-8859-6	ISO 8859-6 Arabic
28597	iso-8859-7	ISO 8859-7 Greek
28598	iso-8859-8	ISO 8859-8 Hebrew; Hebrew (ISO-Visual)
28599	iso-8859-9	ISO 8859-9 Turkish
28603	iso-8859-13	ISO 8859-13 Estonian
28605	iso-8859-15	ISO 8859-15 Latin 9
29001	x-Europa	Europa 3
38598	iso-8859-8-i	ISO 8859-8 Hebrew; Hebrew (ISO-Logical)
50220	iso-2022-jp	ISO 2022 Japanese with no halfwidth Katakana; Japanese (JIS)
50221	csISO2022JP	ISO 2022 Japanese with halfwidth Katakana; Japanese (JIS-Allow 1 byte Kana)
50222	iso-2022-jp	ISO 2022 Japanese JIS X 0201-1989; Japanese (JIS-Allow 1 byte Kana - SO/SI)
50225	iso-2022-kr	ISO 2022 Korean
50227	x-cp50227	ISO 2022 Simplified Chinese; Chinese Simplified (ISO 2022)
50229		ISO 2022 Traditional Chinese
50930		EBCDIC Japanese (Katakana) Extended
50931		EBCDIC US-Canada and Japanese
50933		EBCDIC Korean Extended and Korean
50935		EBCDIC Simplified Chinese Extended and Simplified Chinese
50936		EBCDIC Simplified Chinese
50937		EBCDIC US-Canada and Traditional Chinese
50939		EBCDIC Japanese (Latin) Extended and Japanese
51932	euc-jp	EUC Japanese
51936	EUC-CN	EUC Simplified Chinese; Chinese Simplified (EUC)
51949	euc-kr	EUC Korean
51950		EUC Traditional Chinese
52936	hz-gb-2312	HZ-GB2312 Simplified Chinese; Chinese Simplified (HZ)
54936	GB18030	**Windows XP and later:** GB18030 Simplified Chinese (4 byte); Chinese Simplified (GB18030)
57002	x-iscii-de	ISCII Devanagari
57003	x-iscii-be	ISCII Bengali
57004	x-iscii-ta	ISCII Tamil
57005	x-iscii-te	ISCII Telugu
57006	x-iscii-as	ISCII Assamese
57007	x-iscii-or	ISCII Oriya
57008	x-iscii-ka	ISCII Kannada
57009	x-iscii-ma	ISCII Malayalam
57010	x-iscii-gu	ISCII Gujarati
57011	x-iscii-pa	ISCII Punjabi
65000	utf-7	Unicode (UTF-7)
65001	utf-8	Unicode (UTF-8) (по умолчанию)

Дополнительная настройка setting_1 (реализовано в версии 1.1)

Дополнительные настройки вывода штрихкодов. Подробнее указано в описании конкретных штрихкодов.

6. Форматы штрихкодов

6.1 Одномерные штрих-коды

Именно одномерные штрих-коды большинство людей ассоциирует с термином «штрих-код». Они состоят из ряда полосок и пробелов разной ширины.

6.1.1 Code 11

Разработанный Intermec в 1977 году Код 11 похож на Code 2 of 5 Matrix и используется в основном в области телекоммуникации. Могут кодироваться в строки любой длины. Допустимы цифры 0..9 и символ '-'. Используется одна контрольная цифра (алгоритм modulo-11).

6.1.2 Code 2 of 5

Код 2 из 5 представляет собой семейство одномерных штрих-кодов, 8 из которых поддерживается Zint. Обратите внимание, что эти штрих-коды в разных местах могут называться по разному, поэтому перед использованием этих штрих-кодов вам следует убедиться, что вы используете верный формат.

6.1.2.1 Standard Code 2 of 5

Стандартный Code 2 of 5, также известный как Code 2 of 5 Matrix, это код, обеспечивающий самоконтоль, используемый в промышленности и разработке фотографий. С помощью стандартного Code 2 of 5 можно кодировать числовые данные любой длины (цифры: 0 – 9).

6.1.2.2 IATA Code 2 of 5

Используется Международным агентством воздушного транспорта (IATA) в авиационных перевозках для обработки багажа. С помощью этого самоконтролируемого кода также кодировать числовые входные данные любой длины (цифры: 0 – 9). Контрольной цифры нет.

6.1.2.3 Industrial Code 2 of 5

Промышленный Code 2 of 5 также может кодировать числовые входные данные любой длины (цифры: 0 – 9). Контрольной цифры нет.

6.1.2.4 Interleaved Code 2 of 5

С помощью этого самоконтролируемого форматы кодируют пары чисел, и поэтому можно кодировать только четное число цифр (0 – 9). Если введено нечетное число цифр, Zint добавляет в начала кода цифру 0. Контрольная цифра не добавляется.

6.1.2.5 Code 2 of 5 Data Logic

Data Logic также не включает контрольную цифру; с его помощью можно кодировать числовые входные данные любой длины (цифры: 0 – 9).

6.1.2.6 ITF-14

ITF-14, также известный как штрих-код UPC Shipping Container или Case Code основан на Interleaved Code 2 of 5 и требует входные данные из 13 цифр (цифры: 0 – 9). Рассчитывается контрольная цифра (modulo-10).

6.1.2.7 Deutsche Post Leitcode

Leitcode основан на Interleaved Code 2 из 5 и используется немецкой почтой. Кодируется строка из 13 цифр и включает контрольную цифру.

6.1.2.8 Deutsche Post Identcode

Identcode также основан на Interleaved Code 2 из 5 и также используется немецкой почтой. Кодируется строка из 11 цифр и включает контрольную цифру.

6.1.3 Universal Product Code (EN 797)

6.1.3.1 UPC-A

UPC-A используется в США в розничной торговле. Для формирования штрихкода укажите 11 цифр. Контрольная цифра вычисляется Zint. Кроме того, дополнительные коды EAN-2 и EAN-5 могут быть добавлены с помощью знака «+». К примеру, чтобы сформировать UPC-A штрих-код с данными 72527270270 и с надстройкой EAN-5, отображающей данные 12345, используйте команду:

zint --barcode=34 -d 72527270270+12345

или закодируйте строку данных, включив знак «+»:

```
my_symbol->symbology = BARCODE_UPCA;
error = ZBarcode_Encode_and_Print(my_symbol, "72527270270+12345");
```

6.1.3.2 UPC-E

UPC-E представляет собой сжатую версию UPC-A, разработанную для небольших упаковок. Для формирования штрихкода укажите 6 цифр. Контрольная цифра вычисляется Zint. Дополнительные коды EAN-2 и EAN-5 могут быть добавлены использованием знака «+», как и в UPC-A. Кроме того, Zint поддерживает Number System 1. Для этого укажите 7 цифр, начиная с 1. Например:

zint --barcode=37 -d 1123456

или

```
my_symbol->symbology = BARCODE_UPCE;
error = ZBarcode_Encode_and_Print(my_symbol, "1123456");
```

6.1.4 European Article Number (EN 797)

6.1.4.1 EAN-2, EAN-5, EAN-8 и EAN-13

EAN используется в розничной торговле в Европе и включает стандарты для кодов EAN-8 и EAN-13. Для их формирования укажите 7 или 12 значное число. (*Прим. перев.: если Вы используете Ocvita Barcode ActiveX, то можно указать также 8 или 13 значное число, т.е. включая контрольную цифру*). Также могут формироватьс дополнительные коды EAN-2, EAN-5. (2, 5 значные числа соответственно). Zint самостоят, какой формат будет использован, в соответствии с длиной входных данных. Для добавления дополнительных штрих-коды EAN-2 и EAN-5 используется знак «+», как и в формате UPC. Например:

zint --barcode=13 -d 54321

будет отдельно кодировать EAN-5, в то время как

zint --barcode=13 -d 7432365+54321

будет кодировать EAN-8 вместе с дополнительным EAN-5 кодом. Эти же результаты могут быть достигнуты при использовании API:

```
my_symbol->symbology = BARCODE_EANX;
error = ZBarcode_Encode_and_Print(my_symbol, "54321");
error = ZBarcode_Encode_and_Print(my_symbol, "7432365+54321");
```

Все штрих-коды EAN включают контрольные данные, которые добавляются Zint.

6.1.4.2 SBN, ISBN и ISBN-13

Штрих-коды EAN-13 (также известные как Bookland EAN-13) также могут быть сформированы из входных данных длиной 9 (SBN), 10 (ISBN) или 13 (ISBN-13) знаков. Входные данные должны включать контрольну. цифру (она будет проверена Zint) Кроме того, дополнительные штрих-коды EAN-2 и EAN-5 могут быть добавлены с помощью знака «+», как и в формате UPC.

6.1.5 Plessey

Также известный как Plessey Code, этот формат был разработан Plessey Company Ltd в Великобритании. С помощью этого штрих-кода можно кодировать данные любой длины, состоящие из цифр от 0 до 9 или букв от A до F и включающие контрольную цифру (CRC).

6.1.6 MSI Plessey

Разработанн MSE Data Corporation на основае Plessey. Может кодировать числовые данные любой длины. Доступны следующие варианты использования контрольной цифры (выбирается установкой свойства 'option_2'):

Значение 'option_2'	Контрольная цифра
0	Не используется
1	Modulo-10
2	Modulo-10 & Modulo-10
3	Modulo-11
4	Modulo-11 & Modulo-10

6.1.7 Telepen

6.1.7.1 Telepen Alpha

Telepen Alpha был разработан SB Electronic Systems Limited; с его помощью можно кодировать текстовые входные данные формата ASCII любой длины.
Telepen включает контрольную цифру modulo-127.

6.1.7.2 Telepen Numeric

Telepen Numeric позволяет сжимать числовые данные в штрих-код Telepen. Данные могут состоять из пар чисел или пар, состоящих из чисел, следующих за символом X. Например: 466333 и 466X33 – допустимые коды, а 46X333 – нет (пара цифр “X3” не допустима). Включает контрольную цифру modulo-127.

6.1.8 Code 39

6.1.8.1 Standard Code 39 (ISO 16388)

Стандартный Code 39 был разработан в 1974 году компанией Intermec. Входные данные могут быть любой длины и могут содержать следующие символы: цифры (0 – 9), буквы (A – Z), тире (-), точка (.),пробел, звездочка (*), доллар ($), косая черта (/) и плюс (+). Код не требует контрольных цифр, тем не менее контрольная цифра modulo-43 может быть добавлена, если это необходимо, с помощью установки option_2 = 1 или использования --ver=1.

6.1.8.2 Extended Code 39

Также известный как Code 39e и Code39+, этот формат расширяет возможности стандартного Code 39: поддерживает все символы ASCII. Код не требует контрольных цифр, тем не менее контрольная цифра modulo-43 может быть добавлена, если это необходимо, с помощью установки option_2 = 1 или использования --ver=1.

6.1.8.3 Code 93

Вариация Extended Code 39, Code 93 также поддерживает все символы ASCII. Две контрольные цифры добавляются Zint.

6.1.8.4 PZN

PZN представляет собой формат, основанный на Code 39 и используемый в фармацевтической промышленности в Германии. PZN кодирует 6-значное число и включает контрольную цифру (алгоритм modulo-10).

6.1.8.5 LOGMARS

LOGMARS (Logistics Applications of Automated Marking and Reading Symbols) - Логистические Приложения Автоматической Маркировки и Считывания Штрих-кодов) является разновидностью формата Code 39, используемой Министерством обороны США. С помощью LOGMARS может быть закодирован тот же набор символов, что и Standard Code 39 + добавляется контрольную цифру modulo-43.

6.1.8.6 Code 32

Разновидность Code 39, используемая в Министерстве здравоохранения Италии для кодирования фармацевтической продукции. Требует числовых входных данных длиной не более 8 цифр. Контрольная цифра добавляется Zint.

6.1.8.7 HIBC Code 39

Используется в медицинской промышленности. Аналогичен Standard Code 39, но добавляет знак «+» в начало и контрольная цифру modulo-49 в конец.

6.1.9 Codabar (EN 798)

Также известный как NW-7, Monarch, ABC Codabar, USD-4, Ames Code и Code 27, этот формат был разработан в 1972 компанией Monarch Marketing Systems для розничной торговли. В 1977 году Американская комиссия крови приняла Codabar в качестве стандартного формата идентификации крови. С помощью Codabar можно кодировать строку любой длины, начинающуюся и заканчивающуюся буквами A-D, между которыми могут находиться: цифры от 1 до 9, тире (-), доллар ($), двоеточие (:), косую черту (/), точку (.) и знак (+). Контрольные цифры не создаются.

6.1.10 Pharmacode

Pharmacode, разработанный Laetus, используется для идентификации фармацевтических препаратов. С помощью данного формата можно кодировать целые числа от 3 до 131070.

6.1.11 Code 128

6.1.11.1 Standard Code 128 (ISO 15417)

Один из самых распространенных форматов одномерных штрих-кодов, Code 128 был разработан в 1981 компанией Computer Identics. Этот формат полностью поддерживает ASCII текст и использует три варианта сжатия текста для уменьшения размера штрих-код. Zint автоматически выбирает режимы и добавляет контрольную цифру modulo-103. Code 128 является форматом, который Zint

использует по умолчанию. Кроме того, Zint поддерживает кодировку символов Latin-1 (не английских) в Code 128. Кодировка Latin-1 приведена в приложении A.

6.1.11.2 Code 128 Subset (подмножеств) B

Используется, если надо закодировать данные состоящие лишь из цифр но в режиме формате Code 128B (если установить просто Code 128, то Zint в таком случае будет использовать Code 128C).

6.1.11.3 GS1-128

Разновидность Code 128, также известная как UCC/EAN-128, этот формат определяется в Основной спецификации GS1 (GS1 General Specification). Идентификаторы применения должны быть введены в [квадратных скобках]. Они будут преобразованы в удобочитаемый текст в (круглых скобках). Это позволяет использовать круглые скобки к кодируемой строке данных. Данные с фиксированной длиной должны быть введены с соответствующей длиной для правильного кодирования (см. приложение C). GS1-128 не поддерживает расширенные символы ASCII. Контрольные цифры для данных GTIN (AI 01) не создаются и должны быть включены во входные данные. Ниже приведен пример правильных входных данных для GS1-128:

zint --barcode=16 -d "[01]98898765432106[3202]012345[15]991231"

6.1.11.4 EAN-14

Сокращенный вариант GS1-128, с помощью которого можно кодировать только GTIN данные. На вход подается строка из 13 цифр. Контрольная цифра GTIN и AI (01) добавляются Zint.

6.1.11.5 NVE-18

Разновидность Code 128 формата Nummber der Versandeinheit включает контрольные цифры как modulo-10, так и modulo-103. На вход подается строка из 17 цифр.

6.1.11.6 HIBC Code 128

Эта опция добавляет знак «+» в начало и контрольную цифру modulo-49 в конец стандартного Code 128 в соответствии с требованиями штрих-кодов, использующихся в медицинской промышленности.

6.1.12 GS1 DataBar (ISO 24724)

Также известен как RSS (Reduced Space Symbology - символика сокращенного размера). Эти штрих-коды начиная с 2010 года должны постепенно заменять штрихкоды GS1-128. Если штрих-код GS1 DataBar должен печататься с 2D-компонентом, как указано в ISO 24723, следует установить option_1 = 2 или использовать option --mode=2 в командной строке. (См.раздел 6.3 настоящего руководства, чтобы узнать, как создать штрих-код DataBar с 2D-компонентом)

6.1.12.1 DataBar-14 and DataBar-14 Truncated

Также известен как RSS-14. С помощью этого штрих-кода можно закодировать 13-значное число. Контрольная цифра и идентификатор применения (01) добавляются Zint. Обратите внимание, что для полного соответствия стандарту высота штрих-кода должна быть больше или равна 33 модулям. Для сокращенного DataBar-14 установленная высота штрих-кода должна быть не менее 13.

6.1.12.2 DataBar Limited

Также известен как RSS Limited. Также как и DataBar-14 , данный формат позволяет кодировать 13-значное число. Однако первой цифрой должно быть 0 или 1 (т.е. числа в диапазоне от 0 до 1999999999999). Контрольная цифра и идентификатор применения (01) также добавляются Zint.

6.1.12.3 DataBar Expanded

Также известен как RSS Expanded. Длина входных данных переменная. Позволяет кодировать данные нескольких идентификаторов применения в один штрих-код. В исходных данных идентификаторы применения должны быть заключены в [квадратные скобки]. В последствие они будут преобразованы в (круглые скобки), и в таком виде будут отбражены в тексте выводимом рядом со штрих-кодом. В кодируемых данных могут присутствовать круглые скобки. Данные GTIN (идентификатор применения 01) должны также включать в себя контрольную цифру, так как они не вычисляются Zint при кодировании данного формата. Данные с фиксированной длиной должны быть введены с соответствующей длиной для правильного кодирования (см. приложение C). Ниже приведен пример допустимых входных данных для расширенного DataBar:

zint --barcode=31 -d "[01]98898765432106[3202]012345[15]991231"

6.1.13 Корейский почтовый штрих-код

Корейский почтовый штрих-код обычно используется для кодирования 6-значного числа и включает одну контрольную цифру.

6.1.14 Channel Code

Сильно сжатый штрих-код для числовых данных. Число каналов в штрих-коде может быть от 3 до 8 и определяется установленным значением option_2. Также оно может быть определено длиной входных данных, например, трехсимвольная входная строка по умолчанию создает 4-канальный код. Допустимое максимальное значение зависит от числа используемых каналов, как это показано в таблице ниже:

Каналов (Channels)	**Minimum Value**	**Maximum Value**
3	00	26
4	000	292
5	0000	3493
6	00000	44072
7	000000	576688
8	0000000	7742862

Обратите внимание, что 7- и 8-канальные коды требуют большого количества вычислений, поэтому эти коды формируются относительно медленно.

6.2 Многорядные штрих-коды

6.2.1 Формирование обычных штрихкодов в несколько рядов.

Самый простой способ поместить в штрихкод больше информации - просто использовать несколько одномерных штрихкодов один над другим. (Это используется в основном в автомобильной промышленности). Например

zint -d 'This' -d 'That'

создаст два штрих-кода Code 128, один над другим. Тот же результат можно достигнуть с помощью API, используя функцию ZBarcode_Encode() более одного раза на штрих-код. Например:

```
my_symbol->symbology = BARCODE_CODE128;
error = ZBarcode_Encode(my_symbol, "This");
error = ZBarcode_Encode(my_symbol, "That");
error = ZBarcode_Print(my_symbol);
```

6.2.2 Code 16k (EN 12323)

Более сложный способ заключается в назначении индексов разным рядам , которые указывают считывателю штрих-кодов, в каком порядке коды должны быть считаны. Этот принцип используется в Code 16k, который является блоком из кодов Code128 и может включать до 16 рядов. Максимальный объем данных: 77 букв или 154 цифры. Code 16k включает две контрольные цифры modulo-107. Кроме того данный формат поддерживает расширенный набор символов ASCII, кодируя его тем же образом, что Code 128.

6.2.3 PDF417 (ISO 15438)

Формат PDF417 широко используется в почтовой индустрии; с его помощью можно закодировать огромное количество данных на небольшом пространстве. В соответствии с ISO, Zint поддерживает кодирование максимум до 925 кодовых слов, что (при 0 уровне коррекции ошибок) дает максимальный - 1850 букв или 2710 цифр. Ширина генерируемого штрих-кода PDF417 может быть указана в командной строке с помощью переключения параметра --cols, следующего за цифрой от 1 до 30. Количество контрольных цифр может быть задано с помощью параметра --security switch, следующего за цифрой от 0 до 8, количество кодовых цифр равно 2^(значение+1). При использовании API эти значения указываются в option_2 и option_1 соответственно. По умолчанию количество контрольной информации зависит от объема кодируемых данных. Поддержка международных символов осуществляется с помощью набора символов Latin-1, как описано в Приложении A. Отдельный идентификатор формата может быть использован для кодирования данных Health Industry Barcode (HIBC), при этом в начало добавляется знак «+», а также контрольную (алгоритм modulo-49).

6.2.4 Компактный PDF417

Также известен как сокращенный PDF417. Параметры те же, что и для обычного PDF417.

6.2.5 MicroPDF417 (ISO 24728)

Разновидность стандартного PDF417, MicroPDF417 предназначен для случаев, когда размер штрих-кода должен быть минимальным. Доступно 34 предопределенных размера штрих-кодов с количеством столбцов от 1 до 4 и строчек от 4 до 44. Штрих-код MicroPDF417 максимального размера может содержать 250 буквенно-числовых символа или 366 цифр. Уровень коррекции ошибок зависит от размера штрих-кода. Количество используемых столбцов может быть определено с помощью параметра --cols или option_2, как и в PDF417. Специальный идентификатор формата может быть использован для кодирования данных Health Industry Barcode (HIBC) при этом в начало добавляется знак «+», а также контрольную (алгоритм modulo-49).

6.2.6 GS1 DataBar-14 Stacked (ISO 24724)

Составная разновидность штрих-кода GS1 DataBar-14, использующая такие же входные данные (см. раздел 6.1.12.1). Высота этого штрих-кода фиксирована. Данные кодируются в двухстрочный штрих-код с центральной поисковой структурой. Этот штрихкод может быть создан с дополнительным двумерным компонентом.

6.2.7 GS1 DataBar-14 Stacked Всенаправленный

Составная разновидность штрих-кода GS1 DataBar-14, требующая такие же входные данные (см. раздел 6.1.12.1). Данные кодируются в двухстрочный штрих-код с центральной поисковой структурой. Чтобы создать составной штрих-код, этот штрих-код может быть сгенерирован с двумерным компонентом.

6.2.8 GS1 DataBar Expanded Stacked (ISO 24724)

Составная вариация штрих-кода GS1 DataBar Expanded для маленьких упаковок. Входные данные такие же, как для GS1 DataBar Расширенный (см. раздел 6.1.12.3). Кроме того, ширина штрих-кода

может быть изменена с помощью параметра –cols или option_2. В этом случае количество столбцов зависит от числа пар символов на каждой строке штрих-кода. Для символов с 2D-компонентом количество столбцов должны быть не менее 2. Этот штрих-код может быть сгенерирован с двумерным компонентом.

6.2.9 Code 49

Code 49, разработанный в 1987 году в Intermec, это нечто среднее между UPC и Code 39. Code 49 является одним из ранних многорядных штрих-кодов, и именно он во многом повлиял на дизайн Code 16K, разработанного несколькими годами позже. Он поддерживает полные 7-битные входные данные ASCII длиной до 49 символов или 81 цифр. Также поддерживается кодирование данных GS1.

6.3 Составные штрихкоды (ISO 24723)

Составные штрихкоды используют несколько компонентов, чтобы дать более полную информацию о продукте. Допустимое содержание составного штрих-кода определяется в соответствии с GS1 General Specification. Составные штрих-коды состоят из линейного компонента, которым могут быть штрих-коды EAN, UPC, GS1-128 или GS1 DataBar, 2D-компонента, основанного на PDF417 или MicroPDF417, и разделительной структуры. Тип используемого линейного компонента задается с помощью параметра –b или --barcode= или настройки symbol->symbology, как и обычно. Допустимые значения приведены ниже:

Значение	Имя	Формат
130	BARCODE_EANX_CC	Составной штрих-код с линейным компонентом EAN
131	BARCODE_EAN128_CC	Составной штрих-код с линейным компонентом GS1-128
132	BARCODE_RSS14_CC	Составной штрих-код с линейным компонентом GS1 DataBar-14
133	BARCODE_RSS_LTD_CC	Составной штрих-код с компонентом GS1 DataBar Ограниченный
134	BARCODE_RSS_EXP_CC	Составной штрих-код с компонентом GS1 DataBar Расширенный
135	BARCODE_UPCA_CC	Составной штрих-код с линейным компонентом UPC A
136	BARCODE_UPCE_CC	Составной штрих-код с линейным компонентом UPC E
137	BARCODE_RSS14STACK_CC	Составной штрих-код с компонентом GS1 DataBar-14 Составной
138	BARCODE_RSS14_OMNI_CC	Составной штрих-код с компонентом GS1 DataBar-14 Составной Всенаправленный
139	BARCODE_RSS_EXPSTACK_CC	Составной штрих-код с компонентом GS1 DataBar-14 Расширенный Составной

Данные для кодирования в линейный компонент составного штрих-кода должны быть введены в основную строку с данными для 2D-компонента, вводимыми в обычном порядке. Для этого в командной строке используйте --primary= command. К примеру:

zint -b 130 --mode=1 --primary=331234567890 -d "[99]1234-abcd"

Это создает линейный компонент EAN-13 с данными "331234567890" и 2D CC-A (см. ниже) компонент с данными "(99)1234-abcd". Такие же результаты могут быть достигнуты с помощью API, как показано ниже:

```
my_symbol->symbology = 130;my_symbol->option_1 = 1;
strcpy(my_symbol->primary, "331234567890");
ZBarcode_Encode_and_Print(my_symbol, "[99]1234-abcd");
```

Дополнительные данные EAN-2 и EAN-5 могут быть использованы со штрих-кодами EAN и UPC с помощью знака «+», как описано в разделах 6.1.3 и 5.1.4.
2D-компонент составного штрих-кода может использовать одну из трех следующих систем: CC-A, CC-B и CC-C, как описано ниже. Тип 2D-компонента может быть автоматически выбран Zint в зависимости от длины входной строки. Также три режима могут быть заданы с помощью функции --mode= prompt, перед которой ставятся цифры 1, 2 или 3, соотвественно, или с помощью переменной option_1, как показано выше.

6.3.1 CC-A

Используется разновидность MicroPDF417, оптимизированную для небольшого размера. Размер 2D-компонента и уровень коррекции ошибок определяется размером кодируемых данных и типом используемого линейного компонента. С помощью CC-A можно кодировать до 56 цифр или буквенно-числовую строку меньшей длины. Чтобы выбрать CC-A, установите --mode=1.

6.3.2 CC-B

Используется MicroPDF417 для кодирования 2D-компонента. Размер 2D-компонента и уровень коррекции ошибок определяется размером кодируемых данных и типом используемого линейного компонента. С помощью CC-B можно кодировать до 338 цифр или буквенно-числовую строку меньшей длины. Чтобы выбрать CC-A, установите --mode=2.

6.3.3 CC-C

Используется PDF417 и только в сочетании с линейным компонентом GS1-128. С помощью CC-C можно кодировать до 2361 цифр или буквенно-числовую строку меньшей длины. Чтобы выбрать CC-A, установите --mode=3.

6.4 Двухрядные штрих-коды

6.4.1 Двухрядный Pharmacode

Двухрядный Pharmacode, разработанный Laetus, является альтернативой обычного Pharmacode OneTrack, используемого для идентификации лекарственных средств. С помощью этого штрих-кода можно кодировать целые числа от 4 до 64570080.

6.4.2 PostNet

Штрих-код PostNet до 2009 года использовался почтовой службой США для кодирования индексов почтовых отправлений. PostNet использует числовые входные данные и включает одну контрольную цифру modulo-10. Zint будет кодировать штрих-коды PostNet любой длины, однако стандартными длинами, используемой USPS, были PostNet6 (5-значные входные данные ZIP), PostNet10 (5-значные ZIP + 4-значные пользовательские данные) и PostNet12 (5-значные ZIP + 6-значные пользовательские данные).

6.4.3 PLANET

Штрих-код PLANET до 2009 года использовался почтовой службой США для кодирования данных о маршрутизации почтовых отправлений. PLANET использует числовые входные данные и включает одну контрольную цифру modulo-10. Zint будет кодировать штрих-коды PLANET любой длины, однако стандартными длинами, используемой USPS, были Planet12 (11-значные входные данные) и Planet14 (13-значные входные данные).

6.5 4х рядные почтовые коды

6.5.1 4х рядные код Australia Post (австралийской почты)

6.5.1.1 Клиентские штрих-коды

Стандартный Австралийский Почтовый Клиентский Штрих-код, Клиентский Штрих-код 2 и Клиентский Штрих-код 2 представляют собой, соответственно, 37-, 52- и 67-полосные форматы, разработанные Австралийской Почтой для печати идентификаторов пунктов доставки и клиентской информации на почтовых отправлениях. Допустимые символы: 0-9, A-Z, a-z, пробел и решетка. Код Контроля Формата (FCC) добавляется Zint и не должен присутствовать во входных данных. Данные коррекции ошибок (Reed-Solomon) создаются Zint. Режим кодирования определяется длиной входных данных в соответствии с формулой, приведенной в таблице ниже:

Длина входных данных	Требуемый формат входных данных	Длина штрих-кода	Код Контроля Данных (FCC)	Кодирование таблицы
8	99999999	37полос	11	None
13	99999999AAAAA	52 полосы	59	C
16	9999999999999999	52 полосы	59	N
18	99999999AAAAAAAAAA	67 полос	62	C
23	999999999999999999999	67 полос	62	N

6.5.1.2 Штрих-код Reply Paid (оплаченного овета)

Данный штрих-код является разновидностью 4х рядногоAustralia Post Barcode (FCC 45), требующей 8-значные входные данные идентификаторов пунктов доставки.

6.5.1.3 Штрих-код Routing (маршрутизации)

Данный штрих-код является разновидностью 4х рядногоAustralia Post Barcode (FCC 87), требующей 8-значные входные данные идентификаторов пунктов доставки.

6.5.1.4 Штрих-код Redirect (переадресации)

Данный штрих-код является разновидностью 4х рядногоAustralia Post Barcode (FCC 92), требующей 8-значные входные данные идентификаторов пунктов доставки.

6.5.2 Dutch Post KIX Code (Нидерландский Почтовый Код KIX)

Этот формат используется Royal Dutch TPG Post (Нидерланды) для почтового индекса и автоматической сортировки почты. Входные данные могут состоять из цифр (0 -9) и латинских букв (A-Z) и должны включать 11 символов. Контрольные цифры не добавляются.

6.5.3 Royal Mail (королевский почтовый) 4-State Country Code (RM4SCC)

Формат RM4SCC используется Королевской Почтой Великобритании для кодирования почтовых индексов и клиентских данных на почтовых отправлениях. Входные данные могут состоять из цифр (0 -9) и латинских букв (A-Z) и обычно включают почтовый индекс места доставки, следующий за номером дома. Например, "W1J0TR01" для дома № 1 на Пиккадилли-Серкус в Лондоне. Контрольные цифры генерируются Zint.

6.5.4 USPS OneCode

Также известная как Intilligent Maik штрих-код. Используется Почтовой Службой США, система OneCode в 2009 заменила PostNet и PLANET форматы. OneCode – это фиксированной длины (65-полосный) штрих-код, объединяющая маршрутную и клиентскую информацию в один символ. Входные данные состоят из 20-значного кода отслеживания, следующего за знаком тире (-), следующего за zip-кодом пункта доставки, который может состоять из 0, 5, 9 или 11 цифр. Так, все нижеуказанные входные данные являются допустимыми:
"01234567094987654321"
"01234567094987654321-01234"
"01234567094987654321-012345678"
"01234567094987654321-01234567891"

6.5.5 Японский Почтовый Код

Используются для кодирования адресных данных на почтовых отправлениях. Допустимыми символами являются 0-9, A-Z и тире (-). Контрольная цифра modulo 19 добавляется.

6.6 Двухмерный штрих-коды

6.6.1 Data Matrix (ISO 16022)

Также известный как Semacode формат DataMatrix был разработан Acuity CiMatrix в партнерстве с Министерством Обороны США и NASA в1989. С помощью данного штрих-кода можно кодировать большой объем данных на маленькой площади. Data Matrix может кодировать все символы набора Latin-1 и GS1 данные. Размер генерируемого штрих-кода может быть отрегулирован с помощью параметра --vers= или настройки option_2, как это показано в таблице ниже. Индивидуальный идентификатор формата может быть использован для кодирования данных Health Industry Barcode (HIBC), который добавляет в начало знак «+» и контрольную цифру modulo-49 к закодированным данным. Обратите внимание, что этот штрих-код поддерживает только кодирование ECC200; более старые форматы уже удалены из Zint.

Входные данные	Размер штрих-кода	Входные данные	Размер штрих-кода
1	10 x 10	16	64 x 64
2	12 x 12	17	72 x 72
3	14 x 14	18	80 x 80
4	16 x 16	19	88 x 88
5	18 x 18	20	96 x 96
6	20 x 20	21	104 x 104
7	22 x 22	22	120 x 120
8	24 x 24	23	132 x 132
9	26 x 26	24	144 x 144
10	32 x 32	25	8 x 18
11	36 x 36	26	8 x 32
12	40 x 40	27	12 x 26
13	44 x 44	28	12 x 36
14	48 x 48	29	16 x 36
15	52 x 52	30	16 x 48

Для штрих-кодов Data Matrix доступна настройка, при которой будут формироваться только квадратные. Чтобы использовать эту функцию, в командной строке установите параметр --square, или option_3 = DM_SQUARE.при использовании API.
При использоваании Ocvita Barcode для формирования только квадратных штрихкодов укажите 100 в поле option_3.

Начиная с версии 1.1 Ocvita Barcode позволяет принудительно выбирать вид кодирования, указав соответствующее значение в поле setting_1

Setting_1	Вид кодирования
0	Автовыбор. Примечание: в этом режиме в пределах одного символа может быть использовано несколько указанных ниже видов кодирования. Например первые 20 байт кодируются как ASCII, 40 байт далее как BASE256 и т.п. Виды кодирования выбираются исходя из содержимого для получения оптимального размера символа. В Ocvita Barcode 1.0 использовался только такой режим.
1	ASCII
2	C40
3	TEXT
4	X12
5	EDIFACT
6	BASE256

6.6.2 QR Code (ISO 18004)

QR Code , также известен как Quick Response. Формат был разработан Denso. Четыре уровня коррекции ошибок могут быть установлены с помощью опции security= или настройки option_1, как показано в таблице ниже.

Входные данные	**Уровень коррекции ошибок**	**Объем коррекции ошибок**	**Возможность восстановления**
1	L (по умолчанию)	Приблизительно 20% штрих-кода	Приблизительно 7%
2	M	Приблизительно 37% штрих-кода	Приблизительно 15%
3	Q	Приблизительно 55% штрих-кода	Приблизительно 25%
4	H	Приблизительно 65% штрих-кода	Приблизительно 30%

Размер штрих-кода может быть установлен с помощью опции vers= или установки значения option_2 от 1 до 40 для требуемой версии QR Code. Размеры создаваемых символов приведены в таблице ниже.

Значение	**Размер**	**Значение**	**Размер**
1	21 x 21	21	101 x 101
2	25 x 25	22	105 x 105
3	29 x 29	23	109 x 109
4	33 x 33	24	113 x 113
5	37 x 37	25	117 x 117
6	41 x 41	26	121 x 121
7	45 x 45	27	125 x 125
8	49 x 49	28	129 x 129
9	53 x 53	29	133 x 133
10	57 x 57	30	137 x 137
11	61 x 61	31	141 x 141
12	65 x 65	32	145 x 145
13	69 x 69	33	149 x 149
14	73 x 73	34	153 x 153
15	77 x 77	35	157 x 157
16	81 x 81	36	161 x 161
17	85 x 85	37	165 x 165
18	89 x 89	38	169 x 169
19	93 x 93	39	173 x 173

20	97 x 97	40	177 x 177

Максимальный объем штрих-кода QR Code составляет 7089 цифр, 4296 буквенно-числовых символов или 2953 байта данных (в версии 40). QR Code также может быть использован для кодирования данных GS1. С помощью данного штрих-кода можно кодировать символы Latin-1 и Kanji, являющихся частями схемы кодирования Shift-JIS. Обратите внимание, что изначально входные данные должны быть введены как UTF-8 поток; позже Zint автоматически переведет их в Shift-JIS. Специальный идентификатор формата может быть использован для кодирования данных Health Industry Barcode (HIBC), который добавляет в начало знак «+» и контрольную цифру modulo-49 к закодированным данным.

6.6.3 Микро QR Code (ISO 18004)

Микро QR Code - это миниатюрная версия штрих кода QR Code, предназначенная для коротких сообщений. Для этого штрих-кода доступны те же уровни коррекции, что и для стандартной версии (выше). С помощью данного штрих-кода также можно кодировать символы форматов Latin-1 и Kanji, являющихся частями схемы кодирования Shift-JIS. Обратите внимание, что изначально входные данные должны быть в кодировке UTF-8, позже Zint автоматически переведет их в Shift-JIS. Предпочтительный размер штрих-кода может быть выбран с помощью опции vers= или настройки option_2, хотя Zint может установить и другой размер, если это будет необходимо. Возможные размеры приведены в таблице ниже:

Входные данные	Версия	Размер штрих-кода
1	M1	11 x 11
2	M2	13 x 13
3	M3	15 x 15
4	M4	17 x 17

6.6.4 Maxicode (ISO 16023)

Штрих-код Maxicode, разработанный UPS, представляет собой сетку из шестиугольников, окружающих символ «бычий глаз». Данный формат предназначен для идентификации посылок. Штрих-коды Maxicode могут быть закодированы в одном из пяти доступных режимов.
В режимах 2 и 3 код состоит из двух частей, называющихся первичное и вторичное сообщения. Первичное сообщение состоит из структурированной области данных, включающей в себя различную информацию об отправленном пакете; вторичное сообщение обычно состоит из адресных данных в специальной структуре. Формат первичного сообщения, требуемый Zint, указан в таблице следующей таблице:

Символы	Значение
1-9	Почтовый индекс, который может содержать до 9 цифр (в режиме 2) и до 6 буквенно-числовых символов (в режиме 3).Оставшиеся неиспользованные места для данных должны быть заполнены пробелами (ASCII 32).
10-12	Трехзначный код страны в соответствии с ISO 3166 (см. приложение B).
13-15	Трехзначный служебный код, зависящий от почтового курьера.

Первичное сообщение может быть определено в командной строке с помощью изменения ключа --primary=. Для вторичного сообщения используется стандартный способ ввода данных. К примеру:

zint -o test.eps -b 57 --primary='999999999840012' -d 'Secondary Message Here'

При использовании API первичное сообщение должно быть помещено в строку symbol->primary. Вторичное вводится тем же способом, как описано в разделе 5.2. Zint проанализирует первичное сообщение и определит, какой из режимов (2 или 3) подходит больше.
Режимы 4, 5 и 6 можно установить с помощью ключа --mode= или настройки option_1. Эти режимы не требуют первичного сообщения. Например:

zint -o test.eps -b 57 --mode=4 -d 'A MaxiCode Message in Mode 4'

Режим 6 зарезервирован для обслуживания аппаратной части сканера и не должен использоваться для кодирования пользовательских данных.
Все режимы поддерживают расширенный набор символов ASCII и сжатие чисел. Максимальная длина текста, который может быть вмещен в штрих-код Maxicode, зависит от типов символов, используемых в этом тексте. Примеры максимальных длин данных приведены в таблице ниже:

Режим	Максимальное количество прописных букв	Максимальное количество цифр	Число кодовых слов коррекции ошибок
2 (только вторичное сообщение)	84	126	50
3(только вторичное сообщение)	84	126	50
4	93	135	50
5	77	110	66
6	93	135	50

6.6.5 Aztec Code (ISO 24778)

Aztec Code, изобретенный Эндрю Лонгакром из Welch Allyn Inc в 1995, это матричный штрих-код с характерной структурой поиска «бычий глаз». Zint может генерировать сжатый (compact) Aztec Code (иногда называемый малым (small) Aztec Code) также, как полный (full-range), и по умолчанию автоматически выбирает тип и размер штрих-кода в зависимости от длины кодируемых данных. Кодовые слова коррекции ошибок обычно занимают по крайней мере 23% штрих-кода. Для изменения этих настроек используйте два следующих способа:
Размер штрих-кода может быть задан с помощью опции --ver= или указанием значения option_2 от 1 до 36 в соответствии с таблицей ниже. Штрих-коды, отмеченные в таблице звездочкой (*), являются сжатыми, то есть имеют в центре поисковую структуру меньшего размера.

Входные данные	Размер штрих-кода	Входные данные	Размер штрих-кода
1	15 x 15*	19	79 x 79
2	19 x 19*	20	83 x 83
3	23 x 23*	21	87 x 87
4	27 x 27*	22	91 x 91
5	19 x 19	23	95 x 95
6	23 x 23	24	101 x 101
7	27 x 27	25	105 x 105
8	31 x 31	26	109 x 109
9	37 x 37	27	113 x 113
10	41 x 41	28	117 x 117
11	45 x 45	29	121 x 121
12	49 x 49	30	125 x 125
13	53 x 53	31	131 x 131
14	57 x 57	32	135 x 135
15	61 x 61	33	139 x 139
16	67 x 67	34	143 x 143
17	71 x 71	35	147 x 147
18	75 x 75	36	151 x 151

Обратите внимание, что при указании для штрихкода конкретного размера, объем коррекции ошибок зависит от длины входных данных. Zint может уменьшить объем коррекции ошибок до 3 кодовых слов.

Альтернативный вариант - задать объем коррекции ошибок с помощью опции --mode= или установки значения option_1 из следующей таблицы:

Режим	Объем коррекции ошибок
1	>10% + 3 кодовых слова
2	>23% + 3 кодовых слова
3	>36% + 3 кодовых слова
4	>50% + 3 кодовых слова

Для одного и того же штрих-кода невозможно выбрать и размер и объем коррекции ошибок. Если вы зададите оба параметра, выбор объема коррекции ошибок будет проигнорирован.
Aztec Code имеет возможность кодирования данных расширенного ASCII формата длиной до приблизительно 3823 цифр или 3067 букв, или 1914 байта. Специальный идентификатор формата может быть использован для кодирования данных Health Industry Barcode (HIBC), который добавляет в начало знак «+» и контрольную цифру modulo-49 к закодированным данным.

6.6.6 Aztec Runes

Сокращенная разновидность сжатого Aztec Code, используемая для кодирования целых чисел от 0 до 255. Включает коррекцию ошибок Reed-Solomon. Определен в ISO / IEC 24778 Приложение A.

6.6.7 Code One

Матричный штрих-код, разработанный Тедом Уильямсом в 1992, кодирующий данные тем же образом, что и Data Matrix. С помощью Code One можно кодировать набор символов Latin-1 и данные GS1. Существует два типа Code One – штрих-код переменной высоты, имеющий примерно квадратную форму (версии A – H), и штрих-код фиксированной длины (версии S и T). Они могут быть выбраны с помощью опции --vers= или установки значения option_2, как показано в таблице ниже:

Значение	Версия	Размер	Максимальное число цифр	Максимальное число букв
1	A	16 x 18	22	13
2	B	22 x 22	44	27
3	C	40 x 42	104	64
4	D	40 x 42	217	135
5	E	52 x 54	435	271
6	F	70 x 76	886	553
7	G	104 x 98	1755	1096
8	H	148 x 134	3550	2218
9	S	8X высота	18	-
10	T	16X высота	90	55

С помощью штрих-кодов версии S могут быть закодированы лишь числовые данные. Ширина штрих-кодов версий S и T определяется длиной входных данных.

6.6.8 Grid Matrix

Один из новейших форматов кодирования, поддерживаемый Zint. Grid Matrix в 2008 году стал AIM стандартом. Данный штрих-код позволяет кодировать символы Latin-1и китайские иероглифы стандартного набора GB 2312 в шахматном порядке. Входные данные должны быть введены как UTF-8 поток; перевод в GB 2312 Zint осуществляет автоматически. Размер штрих-кода и объем коррекции ошибок могут быть заданы. Если указать значения обоих параметров, Zint постарается наилучшим образом удовлетворить обоим условиям. Размер штрих-кода может быть выбран с помощью опции ver= или указания значения option_2, объем коррекции ошибок – с помощью опции security= или или указания значения option_1 согласно следующей таблице:

Значение	Размер

1	18 x 18
2	30 x 30
3	42 x 42
4	54 x 54
5	66 x 66
6	78 x 78
7	90x 90
8	102 x 102
9	114 x 114
10	126 x 126
11	138 x 138
12	150 x 150
13	162 x 162

6.7 Другие маркировки, сходные с штрих-кодами

6.7.1. Знак Лицевой Идентификации (Facing Identification Mark - FIM)

Формат FIM используется Почтовой Службой США для автоматизированной обработки почты. Этим штрихкодом можно кодировать одну букву из диапазона A – D, как показано в таблице ниже:

Кодируемая буквы	Использование
A	Используется для ответных писем и почтовых отправлении, оплаченных по тарифу, с заранее напечатанным штрих-кодом PostNet.
B	Используется для деловых писем без заранее напечатанного zip-кода.
C	Используется для деловых писем с заранее напечатанным zip-кодом.
D	Используется для IBI (Information Based Indicia) посылок.

6.7.2 Flattermarken

Flattermarken, используется в типографиях для определения последовательности страниц. По сути не является настоящим штрих-кодом и требует точной информации о местоположении знака на странице. Flattermarken позволяет кодировать числа любой длины и не содержит контрольных цифр.

6.7.3 DAFT Code

Метод создания кодов 4х рядных (4-state) кодов, где входные данные предоставляются внешней программой. Входные данные должны состоять из букв 'D', 'A', 'F' и 'T', которые являются ссылками на использование нижней линии, верхней линии, обеих линий (и нижней, и верхней) и не одной из линии (ни нижней, ни верхней) соответственно. Все прочие символы игнорируются.

7.3 Информация о версиях

v0.1 – (под названием *Zebar*) позволяет создавать штрих-коды UPC-A. UPC-E, EAN-8, EAN-13, Interlaced 2 of 5, Codabar, Code 39, Расширенный Code 39 and Code 93, а также дополнительные коды EAN-2 и EAN-5 без четности. 13/11/2006

v0.2 – добавлены штрих-коды Code 128 (используемый по умолчанию), Code 11, Code 2 of 5, Add-on codes EAN-2 and EAN-5 parity and MSI/Plessey без контрольной цифры 12/12/2006

v0.3 – добавлены проверочные опции Mod 10 и 2 x Mod 10 к штрих-коду MSI/Plessey; штрих-коды Telepen ASCII и Telepen числовой, Postnet, RM4SCC. Исправление ошибок. Добавлен связующий параметр (bind option). 30/12/2006

v0.4 – добавлены многорядные штрих-коды (до 16 кодов) и Code16k. 15/1/2007

v0.5 – добавлены штрих-коды Australia Post 4-State и Pharmacode (1- и 2-разрядный). Теперь коды 4-state генерируются с правильным соотношением высоты и ширины. 28/2/2007

v0.6 – добавлен Plessey и некоторые производные коды ((EAN-128, Code 128 subset B, Auspost Reply, Auspost Routing, Auspost Redirect, ITF-14). Код снова почищен: форматы разделены на большее число файлов, все таблицы поиска поставлены в массивы (что позволило уменьшить их число для каждого штрих-кода, особенно в Code 39e и Code 93). Теперь ошибки выдаются на stderr. Добавлена проверка входных данных. В дистрибутив входят HTML вместо отдельного файла README. Вывод в формате PNG. Выводимые данные могут быть цветными. Пользователи могут выбирать ширину полей и границы. Исправлена ошибка в EAN add-on. Добавлены настройки полей и высоты. Проект переименован в Zint, чтобы избежать конфликтов с существующей торговой маркой. Входные данные могут содержать esc 1/4/2007

v1.0 - устранена проблема с esc символами. Поддержка PDF417. Это завершает список возможностей, которые автор изначально хотел (плюс еще несколько) реализовать, поэтому присвоен номер версии 1.0. 20/4/2007

v1.1 – добавлено больше производных кодов ((Code 2 of 5 Matrix, IATA and Data Logic, Truncated PDF417, Deutsche Post Leitcode и Identcode, Pharmazentralnummer, Planet) и Flattermarken. Исправлены ошибки в 2 of 5 Code. 26/4/2007
v1.2 – поддерживает Data Matrix (в связи с absorption IEC16022 кода Стефаном Шмидтом и др.) Добавлены инверсные цвета, FIM, MSI/Plessey Modulo 11 и Modulo 11/10. Исправлено вычисление контрольной цифры для Code 16k. 28/5/2007

v1.3 – поддерживает USPS OneCode и LOGMARS. Вся полезная информация собрана в один документ (User Manual). 13/6/2007

v1.4 – добавлена поддержка NVE-18. Исправлены некоторые проблемы с компиляцией и контролем входных данных. Обработка параметров командной строки теперь использует getopt(); параметры были изменены. Добавлен параметр --font. 20/6/2007

v1.5 – структурирован API. Исправлены ошибки с EAN-13, PDF417 и LOGMARS. Добавлен вывод в EPS. Добавлена поддержка QR Code, использующая libqrencode. Исправлена ошибка проверки ISBN. Документация переведена в формат HTML. Введены в действие надлежащие процедуры обработки ошибок. Удален параметр --font. Теперь кодирование осуществляется с помощтю структуры zint_symbol. Добавлен параметр и оптимальная поддержка QR Code для makefile. Устранены небольшие проблемы с штрих-кодами 4-State. Сокращено количество исходных файлов. Добавлена поддержка MicroPDF417. 12/8/2007.

v1.5.1 – добавлено форматирование EAN и UPC символов в EPS в соответствии EN 797:1996. Все снова перепроверено и по необходимости изменено и исправлено в соответствии с ISO 16388 и ISO 15417, в том числе добавлена поддержка Latin-1. Изменены настройки изображения, используемые по умолчанию; добавлена автоматическая граница (ITF). Исправлена ошибка в USPS OneCode. Немного почищен модуль Code 39, добавлены параметры Mod 43. 3/9/2007

v1.5.2 – для Code 16k добавлена поддержка расширенного набора символов ASCII. Исправлена ошибка в Code 128. Путем интеграции кода Джоном Льеном добавлена поддержка Maxicode. 26/9/2007

v1.5.3 – сделаны серьезные поправки к Maxicode путем удаление и переписывания большей части программы Льена. Теперь Maxicode поддерживает расширенный набор символов ASCII и режимы 4, 5 и 6. 10/10/2007

v1.5.4 – добавлена поддержка GS1 DataBar. 26/11/2007

v1.5.5 – добавлена поддержка композитных символов. Исправлены ошибки обработки GS1-128 и PDF417/MicroPDF417. Переход на лицензию GPL версии 3. 9/3/2008

v1.6 - Data Matrix, Maxicode и Australia Post теперь используют общие функции Reed-Solomon – это также исправляет ошибку в коррекции ошибок штрих-кода Maxicode. Больше не используется код Льена. Добавлен вывод в PNG штрих-кодов Maxicode. Удалены некоторые бесполезные штрих-коды. QR обновлена до *libqrencode* v2.0.0. 22/4/2008

v1.6.1 – крупная реструктуризация кода, генерирующего PNG: с данного времени он корректно создает штрих-коды UPCA и EAN и вставляет в изображение удобочитаемый человеческий текст. Также исправлен ряд неприятных ошибок: зацикливание в Code 128 и неполадки с контрольными цифрами в PostNet и Planet. 8/7/2008

v1.6.2 – добавлена поддержка KIX Code и возможность вращения PNG изображений. Исправлена ошибка с поддержкой расширенного набора символов ASCII в Code 128 и Code 16k. 28/7/2008

v2.0b - добавлена поддержка штрих-кодов Aztec Code, Codablock-F, код 32, EAN-14 и DAFT. Номера форматов теперь аналогичны Tbarcode v8. Исправлена ошибка (зацикливание в EAN-128). 29/9/2008

v2.0b2 – много исправлении. ((Code 11, Code 128, EAN-128, Aztec Code, Codablock-F, Code 16k, Postnet, PLANET, NVE-18, PZN, Data Matrix, Maxicode and QR Code)

v2.0 – внесены коррективы в Aztec Code и протестирован вывод с bcTester. Добавлены Aztec Runes, Micro QR Code и Data Matrix ECC 000-140. Обновлена информация электронной почты. 18/11/2008

v2.1 – восстановлен штрих-код Korea Post. Согласовывающий параметры bind и box; обработка Unicode перемещена на сервер, в штрих-код Data Matrix добавлены опция input_mode и параметры размера. Добавлена обработка NULL символов в штрих-коды Codablock-F, Code 128, Code 16k, Extended Code 39, Code 93, Telepen, Maxicode, Data Matrix ECC 200, PDF417 и MicroPDF417. В Code 16k, Codablock-F and Aztec Code добавлены поддержка GS1. Добавлен масштаб и направление на stdout параметров. Перестроены алгоритмы кодирования штрих-кода Data Matrix EC 200 для поддержки кодирования NULL и GS1 данных. 31/1/2009

v2.1.1 – незначительные исправления в Data Matrix. Также добавлены опции HIBC. 10/2/2009

v2.1.2 – добавлена возможность вывода данных в изображения формата SVG. Улучшена поддержка японских символов, включая возможность Unicode >Shift-JIS. Исправления в Data Matrix (недостающие символы в конце строки) и Codablock-F (контрольные цифры K1/K2 и ряд показателей выше 6 разряда). 1/3/2009

v2.1.3 - множество улучшений в графическом интерфейсе QZint, теперь переименованном в "Zint Barcode Studio 0.2". Добавлены штрих-коды Japanese Postal, Code 49 и Channel и исправлены ошибки в Data Matrix (при двоичном режиме сжатия данные обрезаются), Aztec Code (ошибка при автоматическом изменении размера после удаления кодовых слов "all 0" и "all 1") и Code 128 (повреждение символов расширенного ASCII набора). 19/5/2009

v2.1.4 – множество улучшений стабильности, включая устранения переполнения буферов в Code 39, LOGMARS, PZN, Aztec Code и Composite CC-A. Добавление файлов для компиляции на платформе MS Windows – успешно протестировано на XP и Vista. 19/6/2009

v2.2 – добавлен Code One; и поддержка GS1 в Code 49. Программный файл gui переименован в zint-qt и обновлен до версии 1.0. Сделан ряд незначительных поправок в штрих-кодах Code 39, ITF-14, Aztec Code, Code 128 и Code 16K. В графическом интерфейсе введена кнопка «rest». Включает .spec файлы от Radist. 18/7/2009

v2.2.1 - исправлены ошибки в кодировании данных в Aztec Code, Data Matrix, USPS One Code и PDF417, ошибки создания штрих-кода в MicroPDF417 и 2D-компонентах композитных символов. Также устранены проблемы с форматирование текста в Qt, и сделано несколько исправлении в выводе изображения в PNG в MSVC. 6/8/2009

v2.2.2 - бета-версия предварительного просмотра новой API структуры. Улучшение поддержки NULL символов за счет удаления значения "nullchar". Добавлены загрузка из файла и последовательные диалоги в Barcode Studio. 29/9/2009

v2.3 – исправлены проблемы в Micro QR Code и поддержка QR Code реорганизована устранением зависимости от libqrencode. Улучшена поддержка Kanji символов в QR Code и Micro QR Code, которые теперь автоматически распознают их и преобразуют в Shift-JIS. Добавлен формат Grid Matrix, также поддерживающий Kanji символы и автоматически преобразующий их в GB 2312. Удален no_qr параметр компиляции. Номер версии Barcode Studio увеличен в соответствии с номером версии библиотеки. 23/11/2009

v2.3.1 – удален Codablock-F. Функция масштаба переработана так, что человеческий читаемый текст и штрих-код Maxicode могут быть последовательно увеличены. Исправлены ошибки кодирования в Code 128/Code 16k и Data Matrix ECC 050. В CLI добавлен параметр --notext. 7/3/2010

v2.3.2 – благодаря тщательному изучению кода Пабло Ордуна из проекта PIRAmIDE исправлены многие ошибки в GS1 DataBar Extended. Благодаря Монике Свэнсон из Source Technologies исправлены ошибки в Maxicode. Кроме того, внесены незначительные исправления в ISBN и Aztec Code, и некоторые небольшие нововведения, таких как параметр –square в CLI. 29/5/2010

v2.4 – благодаря Сэму Лауну и Джиму Эвинсу встроены дополнения к API для интеграции с glabels. Благодаря Исмаэль Лусено добавлена оптимизация кодов и получение входные данные с stdin. Восстановлен символ переключения входных данных. Упрощение Barcode Studio. 13/9/2010

v2.4.1 & 2.4.2 – целый ряд исправлении, включая коррекцию ECC для Code-1 и добавление пакетной обработки в командную строку.