The EAN barcodes by TEX

Petr Olšák

Abstract

In this article, we describe the algorithm for the transformation from the EAN 13 code (13-digit number) to the barcode (the sequence of bars and spaces) and we show the implementation of this algorithm to the macro language of TEX. The drawing of the bars is realized by the TEX primitive `\vrule`. Some data from the standard for the EAN barcodes (tolerances and so on) are presented too. The corresponding TEX macro is available on CTAN in `tex-archive/macros/generic/ean`.

– – ∗ – –

I have prepared my first book about TEX written in Czech (Olsak, 1995). My interest in preparing the book didn't end with sending the manuscript or the type matter to the publisher because the publisher is our CSTUG (the Czechoslovak TEX users group). I made the cover design of the book, I worked on the distribution problems like getting the ISBN, and so on.

When I got the ISBN (International Standard Book Number), I converted it to EAN 13 (European Article Numbering) and I took concern about the barcode for this number, because it is commonly used on book covers. I found out that it would be very expensive to have commercial firms make the barcode. On the other hand, using TEX to produce the barcode is a very natural application of this program because of its high accuracy and its algorithmic macro language. To find the description of the conversion algorithm with 13 digits as the input and the barcode metrics as the output was the only problem. This algorithm is described in (Benadikov et al., 1994).

The transformation from ISBN to EAN is simple. The ISBN is a 10-digit number. The dashes between digits divide the ISBN into the fields "country-publisher-number-checksum" and (essentially) can be ignored. First we write three new constant digits (978) at the front of the ISBN. Next we compute a new checksum digit (the last one). The algorithm for computing the ISBN checksum is different from the one for computing the EAN checksum. For the EAN, first we need to compute the sum of digits on the even positions. Let the sum be e. Next we compute the sum of digits on the odd positions (without the checksum digit). Let the sum be o. We evaluate the expression $3 \times e + o$. The difference between the result and the next modulo-10 number is the checksum digit. For example, *The TEXbook* book hard cover (Knuth, 1986) has its ISBN 0-201-13447-0 (0: country USA; 201: publisher Addison Wesley; 13447: internal book number assigned by publisher; 0: the checksum digit). We write the three constant digits at the front and remove the checksum digit to obtain 978020113447?. Now $e = 7+0+0+1+4+7 = 19$ and $o = 9+8+2+1+3+4 = 27$. The difference between $3 \times 19 + 27 = 84$ and 90 is 6 and this is the checksum digit. We can divide the result by dashes into 6 digit fields (only for easier reading) and the result is 9-780201-134476.

The transformation from the EAN number to the barcode metric is more complicated. The leftmost digit (the 13th position) is 9 for books, but it is different for other kinds of goods. This digit doesn't have its own field in the barcode but it influences the algorithm for the transformation of the following digits to their fields. The widths of the bars or white spaces between the bars are multiples of the basic so-called "module X". The size of module X varies for different SC standards (see below), but the basic size is 0.33 mm. Each digit from positions 12 to 1 is transformed to a field of module width 7X. This field by definition contains two bars and two white spaces. The "start mark" of module width 3X (1X bar, 1X space and 1X bar) is appended before the digit from the 12th position. The same "stop mark" (module width 3X) is appended after the last digit and a so-called "separator mark" of module width 5X (1X space, 1X bar, 1X space, 1X bar and 1X space) is placed between digits on the 7th and 6th positions. The "mark" bars are 5X longer than the bars from the digits.

It is easy to see that the total length of the EAN barcode is 95X. We also have to consider the 11X white space to the left of the code and the 7X white space to the right of the code. These are the minimal white "margins", which are important for the barcodes on a color background. The total number of bars is 30.

Each digit is transformed into two bars in its 7X size field according to one of the tables (A, B and C) shown in Table 1.

Zero in the table stands for the white module (of size 1X) and one means the black module. For example, the digit 4 is converted to 1X space, 1X bar, 3X space and 2X bar by table A and to 2X space, 3X bar, 1X space and 1X bar by table B. Notice that all tables convert the digit into exactly two spaces and two bars and that the converted field starts with the space if table A or B is used, and with the bar if table C is used.

The digits in positions 6 to 1 are transformed by table C under any circumstance. The digits in positions 12 to 7 are transformed by table A or B.

	tab. A	tab. B	tab. C
0	0001101	0100111	1110010
1	0011001	0110011	1100110
2	0010011	0011011	1101100
3	0111101	0100001	1000010
4	0100011	0011101	1011100
5	0110001	0111001	1010000
6	0101111	0000101	1010000
7	0111011	0010001	1000100
8	0110111	0001001	1001000
9	0001011	0010111	1110100

Table 1: Tables A, B and C

13th digit	12	11	10	9	8	7
0	A	A	A	A	A	A
1	A	A	B	A	B	B
2	A	A	B	B	A	B
3	A	A	B	B	B	A
4	A	B	A	A	B	B
5	A	B	B	A	A	B
6	A	B	B	B	A	A
7	A	B	A	B	A	B
8	A	B	A	B	B	A
9	A	B	B	A	B	A

Table 2: Dependence on the 13th digit

The choice depends on the value of the digit in the 13th position, as described in Table 2.

For example, if the digit in position 13 is 9 (our case for books), the digits in positions 12, 9 and 7 are transformed by table A and the ones in positions 11, 10 and 8 are transformed by table B.

Now we can show the TEX macro. First we load the special OCRb font for printing out the EAN code in human-readable form. This printout is appended to the barcode. The METAFONT sources of these fonts are available on CTAN; they were created by Norbert Schwarz. I had to make one little correction in these sources: the command "`mode_setup;`" was added to the beginning of the file `ocrbmac.mf`. Starting at line 4 some "variables" are declared.

```
 1 \message{The EAN-13 barcodes macro. Copyright (C) Petr Olsak, 1995}
 2 \font\ocrb=ocrb9       % for EAN in ‘‘number form’’
 3 \font\ocrbsmall=ocrb7  % for ISBN
 4 \newcount\numlines \newcount\nummodules  % number of bars and of modules.
 5 \newcount\numdigit \newcount\evensum \newcount\oddsum  % internal variables
 6 \newdimen\X            % the module size X,
 7 \newdimen\bcorr        % the bar correction (see below).
 8 \newdimen\workdimen \newdimen\barheight   % internal variables
```

The main macro `\EAN` (line 11) converts the 13-digit EAN number to the internal 60-digit number `\internalcode`. Each digit of the `\internalcode` represents the multiple of the X module size for either the white space or the bar. The order of digits is the same as the order of spaces and bars in the code. The odd positions in the `\internalcode` (from the left) stand for the white spaces and the even ones for the bars.

The usage of the macro is `\EAN 9-780201-134476`, for example. The presence of the "-" signs has no significance. The macro reads 13 digits and saves them in `\firstdigit`, `\frontdigits` and `\enddigits`. At this point, the macro converts the input into `\internalcode` using macros `\settables`, `\usetabAB`, `\insertseparator` and `\usetabC`. The `\testchecksum` macro (line 25) checks for the correctness of the last (check-sum) digit of the EAN.

```
 9 \def\internalcode{0111}             % Begin mark at start
10 \def\frontdigits{}                  % 12--7 digit of EAN
11 \def\EAN{\begingroup\EANscan}
12 \def\EANscan#1{\if#1-\let\next=\EANscan \else
13     \advance\numdigit by1
14     \ifnum\numdigit<13
15       \ifodd\numdigit \advance\oddsum by #1 \else \advance\evensum by #1 \fi
16       \let\next=\EANscan
17       \ifnum\numdigit=1 \settables#1\def\firstdigit{#1}\else
18       \ifnum\numdigit<8 \usetabAB#1\edef\frontdigits{\frontdigits#1}\else
19       \ifnum\numdigit=8 \insertseparator \A \usetabC #1\def\enddigits{#1}%
20      \else            \usetabC#1\edef\enddigits{\enddigits#1}%
21      \fi\fi\fi
22    \else \testchecksum#1\usetabC#1\edef\enddigits{\enddigits#1}%
23      \let\next=\EANclose
24    \fi\fi \next}
25 \def\testchecksum#1{\multiply\evensum by3 \advance\evensum by\oddsum
26    \oddsum=\evensum
27    \divide\oddsum by10 \multiply\oddsum by10 \advance\oddsum by10
28    \advance\oddsum by-\evensum \ifnum\oddsum=10 \oddsum=0 \fi
29    \ifnum#1=\oddsum \else
30    \errmessage{The checksum digit has to be \the\oddsum, no #1 !}\fi}
```

At the time of the `\EANclose` expansion (line 31), we close the `\internalcode` by the `\insertendmark`; next we write to the `log` the EAN number in the 13-digit form and in the internal 60-digit representation. The last action is to "run" the macro `\EANbox`, which makes the box with the bar-code. The input parameter to this macro is the 60-digit `\internalcode`.

How was the `\internalcode` made? The macros starting at line 35 answer this question. These macros are the tables mentioned above rewritten in the macro language of TeX.

There is no need to define table C in the macro, because table C is the exact "inverse" of table A. When we insert the separator (line 52 of the macro), the odd number of digits (namely 5) is appended to the `\internalcode`. This implies that the parity of the black/white order is changed. Using table `\A` is therefore sufficient for the transformation of the digits in positions 6 to 1 (see line 19).

```
31 \def\EANclose{\insertendmark
32    \wlog{EAN: \firstdigit\space\frontdigits\space\enddigits}%
33    \wlog{EANinternal: \internalcode}%
34    \expandafter\EANbox\internalcode..\endgroup}
35 \def\A{\def\0{3211}\def\1{2221}\def\2{2122}\def\3{1411}\def\4{1132}%
36   \def\5{1231}\def\6{1114}\def\7{1312}\def\8{1213}\def\9{3112}}
37 \def\B{\def\0{1123}\def\1{1222}\def\2{2212}\def\3{1141}\def\4{2311}%
38   \def\5{1321}\def\6{4111}\def\7{2131}\def\8{3121}\def\9{2113}}
39 \def\settables#1{\ifnum#1=0 \def\tabs{\A\A\A\A\A\A}\fi
40                  \ifnum#1=1 \def\tabs{\A\A\B\A\B\B}\fi
41                  \ifnum#1=2 \def\tabs{\A\A\B\B\A\B}\fi
42                  \ifnum#1=3 \def\tabs{\A\A\B\B\B\A}\fi
43                  \ifnum#1=4 \def\tabs{\A\B\A\A\B\B}\fi
44                  \ifnum#1=5 \def\tabs{\A\B\B\A\A\B}\fi
45                  \ifnum#1=6 \def\tabs{\A\B\B\B\A\A}\fi
46                  \ifnum#1=7 \def\tabs{\A\B\A\B\A\B}\fi
47                  \ifnum#1=8 \def\tabs{\A\B\A\B\B\A}\fi
48                  \ifnum#1=9 \def\tabs{\A\B\B\A\B\A}\fi}
49 \def\usetabAB#1{\expandafter\scantab\tabs\end \usetabC #1}
50 \def\scantab#1#2\end{#1\def\tabs{#2}} % The tab #1 is activated and removed
51 \def\usetabC#1{\edef\internalcode{\internalcode\csname#1\endcsname}}
52 \def\insertseparator{\edef\internalcode{\internalcode 11111}}
53 \def\insertendmark{\edef\internalcode{\internalcode 111}}
```

Now comes the most important part of our macro: creating the bars using the TeX primitive `\vrule`. The internal macro `\EANbox` (line 54) does this job. This macro reads the 60-digit `\internalcode` (ended by two dots) as its parameter. It scans two digits per step from the parameter (first digit: the white space; second digit: the black bar) and puts in the appropriate kerns and rules. Each kern/rule pair is corrected by a so-called "bar correction". The standard recommends making each rule thinner than what is exactly implied by the multiple of the X size. This recommendation is due to the ink behavior during the actual printing. For example, for offset process technology, it is recommended to reduce the bar width by 0.020 mm. If the bar width is reduced, the white space must be enlarged by the same amount in order to preserve the global distance between bars.

The bars 1, 2, 15, 16, 29 and 30 have nonzero depth (5X) because these are the lines from the start, the separator and the stop marks. The height of the bars is 69.24 X in the normal case but it may be reduced, if the ISBN is appended to the top of the code. If the `\barheight` is zero, than the implicit height is used. Otherwise the `\barheight` is used. This feature gives the user the possibility to set the bar height individually.

```
54 \def\EANbox{\vbox\bgroup\offinterlineskip
55   \setbox0=\hbox\bgroup \kern11\X\EANrepeat}
56 \def\EANrepeat#1#2{\if#1.\let\next=\EANfinal \else\let\next=\EANrepeat
57   \advance\numlines by1
58   \advance\nummodules by#1 \advance\nummodules by#2
59   \workdimen=#1\X \advance\workdimen by \bcorr \kern\workdimen
60   \workdimen=#2\X \advance\workdimen by-\bcorr \vrule width\workdimen
61      \ifdim\barheight=0pt height 69.24242424\X \else height\barheight \fi
62      \ifnum\numlines=1    depth5\X\else  % the start mark
63      \ifnum\numlines=2    depth5\X\else
64      \ifnum\numlines=15   depth5\X\else  % the separator mark
65      \ifnum\numlines=16   depth5\X\else
66      \ifnum\numlines=29   depth5\X\else  % the end mark
67      \ifnum\numlines=30   depth5\X\else depth0pt \fi\fi\fi\fi\fi\fi
68   \fi\next}
```

The `\EANfinal` macro checks for the correctness of the scanned `\internalcode`. The number of the digits must be 60 and the sum of digits must be 95 (since 95X modules is the total). If the check fails, the `\internalerr` macro is activated. However this situation should never occur. This error indicates that some internal tables are wrong and/or the consistency of the macro is broken.

The `\vbox` is completed by `\EANfinal`. The natural depth of the internal `\hbox` with the bars is 5X because that is the depth of the mark rules. We overwrite this depth by zero and append the human-readable EAN number using the font `\ocrb`.

If the user writes the ISBN number using the macro `\ISBN` (`\ISBN 0-201-13447-0` for example), these data are appended to the top of the barcode and the height of the bars is reduced.

Finally, lines 81 and 82 define the X module size and the bar correction.

```
69 \def\EANfinal{\testconsistence
70   \kern7\X\egroup
71   \hbox{\ocrbsmall \kern10\X \ISBNnum}\kern1\X
72   \dp0=0pt \box0 \kern-1\X
73   \hbox{\ocrb\kern2\X\firstdigit\kern5\X \frontdigits\kern5\X \enddigits}
74   \egroup \global\barheight=0pt \gdef\ISBNnum{}}
75 \def\testconsistence{\ifnum\numlines=30\else\internalerr\fi
76   \ifnum\nummodules=95\else\internalerr\fi}
77 \def\internalerr{\errmassage{Sorry, my internal tables are wrong, may be.}}
78 \barheight=0pt
79 \def\ISBNnum{}
80 \def\ISBN #1 {\def\ISBNnum{ISBN #1}\barheight=45.151515\X\relax}
81 \X=.33mm       % Basic size 100%, SC2 code
82 \bcorr=.020mm  % Bar-correction for offset process
83 \endinput
```

If the macro was stored in file `ean13.tex` then it can be run with plain TEX as shown below:

```
\input ean13
\nopagenumbers
\ISBN 80-901950-0-8  \EAN 978-80-901950-0-4
\end
```

The output looks like:

ISBN 80-901950-0-8

9 788090 195004

X size	standard	scaled	size incl. margins
0.264	SC0	0.800	29.83 × 21.00
0.270	SC0	0.818	30.58 × 21.53
0.281	SC0	0.850	31.70 × 22.32
0.297	SC1	0.900	33.56 × 23.63
0.313	SC1	0.950	35.43 × 24.94
0.330	SC2	1.000	37.29 × 26.26
0.346	SC2	1.050	39.15 × 27.58
0.363	SC3	1.100	41.02 × 28.29
0.379	SC3	1.150	42.88 × 30.20
0.396	SC4	1.200	44.75 × 31.51
0.412	SC4	1.250	46.61 × 32.82
0.429	SC5	1.300	48.48 × 34.14
0.445	SC5	1.350	50.34 × 35.45
0.462	SC5	1.400	52.21 × 36.76
0.478	SC5	1.450	54.07 × 38.08
0.495	SC6	1.500	55.94 × 39.39
0.511	SC6	1.550	57.80 × 40.70
0.528	SC7	1.600	59.66 × 42.01
0.544	SC7	1.650	61.53 × 43.33
0.561	SC7	1.700	63.39 × 44.64
0.577	SC7	1.750	65.26 × 45.96
0.594	SC8	1.800	67.12 × 47.26
0.610	SC8	1.850	68.99 × 48.58
0.627	SC8	1.900	70.85 × 49.90
0.643	SC8	1.950	72.72 × 51.20
0.660	SC9	2.000	74.58 × 52.52
0.700	SC10	2.120	79.05 × 55.67

Table 3: Various sizes of the X module

The macro also works with LaTeX (both LaTeX 2.09 and LaTeX 2$_\varepsilon$).

At the end of this article we compare the tolerances described in the standard, the TeX accuracy and the possibilities of some output devices.

The X module size can vary. The macro above makes EAN barcodes for the basic X module size of 0.33 mm. This size is described in the SC2 variant of the standard as the basic 100% size code. However the standard also allows some other sizes of the X module. One can change the parameter `\X` to obtain the other size of EAN code. Of course, then the size of the OCRb font must be changed too.

The allowed sizes of the X module are described in Table 3.

The small sizes of the X module are recommended for high quality output devices while the large sizes of X allow the possibility to make the barcodes even on a low resolution output device.

Depending on the width of the X module, the standard specifies three tolerance parameters. The parameter a specifies the tolerance for the bar width, the parameter b specifies the tolerance for the distance between edges (either left or right ones) of two consecutive bars, and the parameter c specifies the tolerance for the width of the field for one digit (therefore for width 7X). The Table 4 describes these tolerances in micrometers (μm). I don't know why the table column "X size" doesn't match with the column "X size" of the previous table. Sorry, standards are mysterious.

X size	$\pm a$	$\pm b$	$\pm c$
0.26	32	38	75
0.28	52	41	81
0.30	72	44	87
0.32	92	47	93
0.33	101	49	96
0.34	105	50	99
0.36	115	53	104
0.38	124	56	110
0.40	134	59	116
0.42	143	62	122
0.44	152	65	128
0.46	162	68	133
0.48	171	71	139
0.50	181	73	145
0.52	190	76	151
0.54	199	79	157
0.56	209	82	162
0.58	218	85	168
0.60	228	88	174
0.62	237	91	180
0.64	246	94	186

Table 4: The tolerances

Now we can compare. Consider the basic 100% size (the X module is 0.33 mm). The tolerance for the width of the bar is 101 μm, the TeX (in)accuracy is 0.0054 μm, the pixel size of the phototypesetter at 2400 dpi is approximately 10 μm and the recommended bar correction for the offset process is 20 μm. If we use the phototypesetter at 1200 dpi, the inaccuracy of its output is comparable to the bar correction for the offset process.

Depending on the inner `dvi` driver algorithm the high TeX accuracy may be lost and the tolerance parameters may be overcome. The `dvi` driver algorithms include one of two possible approaches to the "rounding" problem. The first approach is to position and round each rule from `dvi` individually. In the second approach, the `dvi` driver works only with rounded values (one pixel = one unit) *before* making the queue of kern, rule, kern, rule... In this case, the roundoff error can accumulate and the parameter *c* can be overcome. But it seems to me that the barcode scanners can read the code better if the metrics of the consecutive bars and spaces are preserved instead of the global width.

As I observed, the `dvi` drivers usually round the rule width *up* to the pixel units and never *down*. The consequence of this feature is that spaces tend to be one pixel smaller than the rules of (presumably) the same width. Therefore I recommend to add one half of the pixel size to the bar correction, namely to the `\bcorr` register.

I have heard that EAN barcodes are successfully read from stickers printed by matrix printers with a very low resolution at module X size of 0.33 mm or comparably small. That would imply that the tolerances of the barcode scanners are usually much higher than those required by the standard.

References

Adriana Benadiková, Štefan Mada and Stanislav Weinlich. *Čárové kódy, automatická identifikace (Barcodes, the Automatic Identification).* Grada 1994, 272 pp., ISBN 80-85623-66-8.

Donald Knuth. *The TEXbook*, volume A of *Computers and Typesetting.* Addison-Wesley, Reading, MA, USA, 1986. ix+483 pp. Hard cover ISBN 0-201-13447-0.

Petr Olšák. *Typografický systém TEX (Typesetting System TEX).* CSTUG 1995, 270 pp., ISBN 80-901950-0-8.

◇ Petr Olšák
Department of Mathematics
Czech Technical University in Prague
Czech Republic
`olsak@math.feld.cvut.cz`