the barracuda manual

https://github.com/robitex/barracuda

Roberto Giacomelli
email: giaconet.mailbox@gmail.com

Date 2020-02-04 — Version v0.0.10 — Beta stage

Abstract

Welcome to the barracuda software project devoted to barcode printing.

This manual shows you how to print barcodes in your TEX documents and how to export such graphic content to an external file, using barracuda.

barracuda is written in Lua programming language and is free software released under the GPL 2 License.

Contents

1 Getting started **2**
- 1.1 Introduction . . . 2
- 1.2 Manual Content . . . 2
- 1.3 Required knowledge and useful resources . . . 3
- 1.4 Running Barracuda . . . 3
  - 1.4.1 A Lua script . . . 3
  - 1.4.2 A LuaTEX source file . . . 4
  - 1.4.3 A LuaLATEX source file . . . 4
- 1.5 A more deep look . . . 4
- 1.6 Installing . . . 5
  - 1.6.1 Installing for Lua . . . 5
  - 1.6.2 Installing for TeX Live . . . 5

2 Barracuda LATEX Package **5**

3 Barcode parameters **6**
- 3.1 Encoder treename . . . 6
- 3.2 Barcode Reference . . . 6

4 Developer zone **6**
- 4.1 The Barracuda Framework . . . 6
- 4.2 Lua API reference . . . 6
- 4.3 ga specification . . . 6

5 Example and use cases **6**

1 Getting started

1.1 Introduction

Barcode symbols are usually a sequence of vertical lines representing encoded data that can be retrived with special laser scanner or more simpler with a smartphone running dedicated apps. Almost every store item has a label with a printed barcode for automatic identification purpose.

So far, `barracuda` supported symbologies are as the following:

- Code 39,
- Code 128,
- EAN family (ISBN, ISSN, EAN 8, EAN 13, and the add-ons EAN 2 and EAN 5),
- ITF 2of5, interleaved Two of Five.

The package provides different output graphic format. At the moment they are:

- PDF Portable Document Format (a modern TEX engine is required),
- SVG Scalable Vector Graphic.

The name `barracuda` is an assonance to the name Barcode. I started the project back in 2016 for getting barcode in my TEX generated PDF documents, studying the LuaTEX technology such as direct *pdfliteral node* creation.

At the moment `barracuda` is in *beta* stage. In this phase the Lua API can change respect to the result of development research.

1.2 Manual Content

The manual is divided into five part. In part 1.1 introduces the package and gives to the user a proof of concept to how to use it. The next parts present detailed information about option parameter of each barcode symbology and methods description to change the *module* width of a EAN-13 barcode. It's also detailed how the Lua code works internally and how to implement a barcode symbology not already included in the package.

The plan of the manual is (but some sections are not completed yet):

Part 1: Getting started

- general introduction → 2
- print your first barcode → 3
- installing `barracuda` on your system → 5

Part 2: LATEX packages

- `barracuda` LATEX package → 5

Part 3: Barcode Reference and Parameters

- encoder identification rule → 6
- barcode symbologies reference → 6

Part 4: Advanced Work with `barracuda`

- Lua framework description → 6
- API reference → 6
- `ga` specification → 6

Part 5: Real examples

- working example and use cases → 6

1.3 Required knowledge and useful resources

barracuda is a Lua package that can be executed by any Lua interpreter. To use it, it's necessary a minimal knowledge of Lua programming language and a certain ability with the terminal of your computer system in order to run command line task or make software installation.

It's also possible to run barracuda directly within a TEX source file, and compile it with a suitable typesetting engine like LuaTEX. In this case a minimal TEX system knowledge is required. As an example of this workflow you simply can look to this manual because itself is typesetted with LuaLaTEX, running barracuda to include barcodes as a vector graphic object.

A third way is to use the LATEX package barracuda.sty with its high level macros. A minimal knowledge of the LATEX format is obviously required.

Here is a collection of useful learning resources:

Lua: to learn Lua the main reference is the book called PIL, Programming in Lua from one of the language's Author Roberto Ierusalimschy.

LATEX: ...

LuaTEX: ...

1.4 Running Barracuda

The starting point to work with barracuda is always a plain text file with some code processed by a command line program with a Lua interpreter.

The paradigm of barracuda is the Object Oriented Programming. Generally speaking every object must be created with a function called *costructor* and every action must be run calling a *method* of it.

In this section you'll take a taste of barracuda coding in three different execution context: a Lua script, a LuaTEX document and a LATEX source file using the macro package barracuda.sty providing an high level interface to Lua code.

High level package like barracuda.sty make to write Lua code unnecessary. It will be always possible return to Lua code in order to resolve complex barcode requirements.

1.4.1 A Lua script

As a practical example to produce an EAN 13 barcode, open a text editor of your choice on an empty file and save it as first-run.lua with the content of the following two lines of code:

first-run.lua

```
local barracuda = require "barracuda"
barracuda:save("ean-13", "8006194056290", "my_barcode", "svg")
```

What you have done is to write a *script*. If you have installed a Lua interpreter along with barracuda, open a terminal and run it with the command:

```
$ lua first-run.lua
```

You will see in the same directory of your script, appearing a new file called my_barcode.svg with the drawing:

Coming back to the script first of all, it's necessary to load the library `barracuda` with the standard Lua function `require()` that returns an object–more precisely a reference to a table where are stored all the package machinery.

With the second line of code, an EAN 13 barcode is saved as `my_barcode.svg` using the method `save()` of the `barracuda` object. The `save()` method takes in order the barcode symbology identifier called *treename*, an argument as a string or as a whole number that represents data to be encoded, the output file name and the optional output format. With a fifth optional argument we can pass options to the barcode encoder as a Lua table.

Each encoder has an own identifier called treename explained at section 3.1. In short, in `barracuda` we can build more encoders of the same symbology with different parameters.

1.4.2 A LuaTEX source file

`barracuda` can also runs with LuaTEX and any others Lua powered TEX engines. The source file is a bit difference respect to the previuos script: the Lua code lives inside the argument of a `\directlua` primitive, moreover we must use an horizontal box register as output destination.

```
% !TeX program = LuaTeX
\newbox\mybox
\directlua{
        local require "barracuda"
        barracuda:hbox("ean-13", "8006194056290", "mybox")
}\leavevmode\box\mybox
\bye
```

The method `hbox()` works only with LuaTEX. It takes three[1] arguments: encoder *treename*, encoding data as a string, the TEX horizontal box name.

1.4.3 A LuaLATEX source file

LATEX working minimal example would be:

```
% !TeX program = LuaLaTeX
\documentclass{article}
\usepackage{barracuda}
\begin{document}
\barracuda{ean-13}{8006194056290}
\end{document}
```

1.5 A more deep look

`barracuda` is designed to be modular and flexible. For example it is possible to draw different barcodes on the same canvas or tune barcode parameters.

The main workflow to draw a barcode object reveals more details on internal structure. In fact, to draw an EAN 13 barcode we must do at least the following steps:

1. load the library,
2. get a reference to the `Barcode` abstract class,
3. build an `ean` encoder of the variant `13`,
4. build an EAN 13 symbol passing data to a costructor,
5. get a reference to a new canvas object,
6. draw barcode on the canvas object,

[1] A fourth argment is optional as a table with user defined barcode parameters.

7. get a reference of the driver object,
8. print the graphic material saving an external `svg` file.

Following that step by step procedure the corresponding code is translated in the next listing:

```
-- lua script
local barracuda = require "barracuda" -- step 1
local barcode = barracuda:barcode() -- step 2

local ean13, err_enc = barcode:new_encoder("ean-13") -- step 3
assert(ean13, err_enc)

local symb, err_symb = ean13:from_string("8006194056290") -- step 4
assert(symb, err_symb)

local canvas = barracuda:new_canvas() -- step 5
symb:append_ga(canvas) -- step 6

local driver = barracuda:get_driver() -- step 7
local ok, err_out = driver:save("svg", canvas, "my_barcode") -- step 8
assert(ok, err_out)
```

Late the manual will give objects and methods references at section 4.2.

1.6 Installing

1.6.1 Installing for Lua

Manually copy `src` folder content to a suitable directory of your system that is reachable to the system Lua interpreter.

1.6.2 Installing for TeX Live

If you have TeX Live installed from CTAN or from DVD TeX Collection, before any modification to your system check if the package is already installed looking for *installed* key in the output of the command:

```
$ tlmgr show barracuda
```

If ‘barracuda‘ is not present, run the command:

```
$ tlmgr install barracuda
```

If you have installed TeX Live via Linux OS repository try your distribution's package management system running a software update.

It's also possible to install the package manually:

1. Grab the sources from CTAN or `https://github.com/robitex/barracuda`.
2. Unzip it at the root of one or your TDS trees (local or personal).
3. You may need to update some filename database after this, see your TEX distribution's manual for details.

2 Barracuda LATEX Package

The LATEX package delivered with `barracuda` is still under an early stage of development. The only macro available is `\barracuda{encoder}{data}`. A simple example is the following source file for LuaLATEX:

```
% !TeX program = LuaLaTeX
\documentclass{article}
\usepackage{barracuda}
\begin{document}
\leavevmode
\barracuda{code39}{123ABC}\\
\barracuda{code128}{123ABC}
\end{document}
```

Every macro `\barracuda` typesets a barcode symbol with the encoder defined in the first argument, encoding data defined by the second.

3 Barcode parameters

3.1 Encoder treename

TODO

3.2 Barcode Reference

TODO

4 Developer zone

4.1 The Barracuda Framework

The `barracuda` package framework consists in indipendent modules: a barcode class hierarchy encoding a text into a barcode symbology; a geometrical library called `libgeo` representing several graphic objects; an encoding library for the `ga` format (graphic assembler) and several driver to *print* a ga stream into a file or a TEX hbox register.

To implement a barcode encoder you have to write a component called *encoder* defining every parameters and implementing the encoder builder, while a driver must understand ga opcode stream and print the corresponding graphic object.

Every barcode encoder come with a set of parameters, some of them can be reserved and can't be edit after the encoder was build. So, you can create many instances of the same encoder for a single barcode type, with its own parameter set.

The basic idea is getting faster encoders, for which the user may set up paramenters at any level: barcode abstract class, encoder globally, down to a single symbol object.

The Barcode class is completely indipendent from the ouput driver and viceversa.

4.2 Lua API reference

TODO

4.3 ga specification

TODO

5 Example and use cases

TODO