The Embroidermodder Project

for Embroidermodder 2.0.0-alpha, libembroidery 1.0.0-alphaand all related projects

The Embroidermodder Team

May 30, 2022

https://embroidermodder.org

Contents

Chapter 1

Introduction

1.1 History

Chapter 2

The Graphical User Interface: Embroidermodder 2.0.0-alpha

(UNDER MAJOR RESTRUCTURING, PLEASE WAIT FOR VERSION 2)

```
![Build Linux/GNU Status](https://github.com/Embroidermodder/Embroidermodder/actions/workflo
![Build Mac OS Status](https://github.com/Embroidermodder/Embroidermodder/actions/workflows/
![Build Windows Status](https://github.com/Embroidermodder/Embroidermodder/actions/workflows

![Test Linux/GNU Status](https://github.com/Embroidermodder/Embroidermodder/actions/workflow
![Test Mac OS Status](https://github.com/Embroidermodder/Embroidermodder/actions/workflows/t
![Test Windows Status](https://github.com/Embroidermodder/Embroidermodder/actions/workflows/
```

Embroidermodder is a free machine embroidery application. The newest version, Embroidermodder 2 can:

- edit and create embroidery designs
- estimate the amount of thread and machine time needed to stitch a design
- convert embroidery files to a variety of formats
- upscale or downscale designs
- run on Windows, Mac and Linux

For more information, see [our website](http://embroidermodder.org).

Embroidermodder 2 is very much a work in progress since we're doing a ground up rewrite to an interface in Python using the GUI toolkit Tk. The reasoning for this is detailed in the issues tab.

For a more in-depth look at what we are developing read the developer notes[1]. This discusses recent changes in a less formal way than a changelog (since this software is in development) and covers what we are about to try.

To see what we're focussing on at the moment check this table.

Date	Event
April-June 2022	Finish the conversion to C/SDL2
July-August 2022	Finish all the targets in the Design, or assign them to 2.1.
September 2022	Bugfixing, Testing, QA. libembroidery 1.0 will be released, then updates will slow do
October 2022	Embroidermodder 2 is officially released.

[1] link to dev notes section

2.1 Build and Install

2.1.1 Desktop

First you must install the dependencies which aren't compiled into the source:

- `git`
- `cmake`
- `SDL2`
- `SDL2_image`
- `SDL2_ttf`
- A C compiler (we recommend `gcc` or `clang`)

on Debian Linux/GNU use:

```
$ sudo apt install git clang build-essential libsdl2-dev \
      libsdl2-images-dev libsdl2-ttf-dev
```

If you can't find a good fit for your system (on Windows use the section below), try compiling the included submodules with:

```
$ bash build_deps.sh
```

From here, on most sytems the command:

```
$ bash build.sh
```

will build the software. Currently this is the 2.0-alpha, which will have a build code of some kind.

2.2 Dependencies and Build

Chapter 3

The Low Level API: Libembroidery 1.0.0-alpha

Chapter 4

The Command Line Interface: `embroider`

4.1 Usage

Chapter 5

Mobile Support: MobileViewer and iMobileViewer

Chapter 6

Plans

Windows Specific Advice

This is one of many possible ways to build the software on Windows, this section is to help people who've not got a build environment to start with.

1. Download and install MSYS2 (follow their instructions): https://www.msys2.org/
2. Boot "Mintty" from the Start menu.
3. Use the commands:

```
$ pacman -S gcc cmake git bash mingw-w64-SDL2 mingw-w64-SDL2_image mingw-w64-SDL2_ttf
$ git clone https://github.com/Embroidermodder/Embroidermodder
$ cd Embroidermodder
$ bash build.sh
```

M

obile

These are currently unsupported (see iMobileViewer and Mobileviewer for iOS and Android respectively), but after the Desktop version is released we'll work on them.

The Mobile version will share some of the UI and all of the backend, so development of the Desktop version will help us make both.

6.0.1 D

ocumentation

The documentation is in the form of the website (included in the 'docs/' directory) and the printed docs in the three files:

- docs/libembroidery_0.1_manual.pdf
- docs/embroidermodder_1.90.0_user_manual.pdf
- docs/embroidermodder_1.90.0_developer_notes.pdf

6.0.2 D

evelopment

If you wish to develop with us you can chat via the contact email on the [website](embroidermodder.org) or in the issues tab on the [github page](https://github.com/Embroidermodder/Embroidermodder/issues). People have been polite and friendly in these conversations and I (Robin) have really enjoyed them. If we do have any arguments please note we have a [Code of Conduct](CODE_OF_CONDUCT.md) so there is a consistent policy to enforce when dealing with these arguments.

The first thing you should try is building from source using the [build advice](link to build) above. Then read some of the [development notes](link to dev notes.md) to get the general layout of the source code and what we are currently planning.

Testing

To find unfixed errors run the tests by launching from the command line with:

```
$ embroidermodder --test
```

then dig through the output. It's currently not worth reporting the errors, since there are so many but if you can fix anything reported here you can submit a PR.

6.0.3 Overall Structure

6.0.4 Code Optimisations and Simplifications

Current

What Robin is currently doing.

Getting the code to pass PyLint, that involves getting all source files under 1000 lines, renaming all variables to be in snake case.

Changing the seperation of code between EM and libembroidery.

Translating the Qt widget framework to Tk.

Geometry

The geometry is stored, processed and altered via libembroidery. See the Python specific part of the documentation for libembroidery for this. What the code in Embroidermodder does is make the GUI widgets to change and view this information graphically.

For example if we create a circle with radius 10mm and center at (20mm, 30mm) then fill it with stitches the commands would be

```
from libembroidery import Pattern, Circle, Vector, satin
circle = Circle(Vector(20, 30), 10)
pattern = Pattern()
pattern.add_circle(circle, fill=satin)
pattern.to_stitches()
```

but the user would do this through a series of GUI actions:

1. Create new file 2. Click add circle 3. Use the Settings dialog to alter the radius and center 4. Use the fill tool on circle 5. Select satin from the drop down menu

So EM2 does the job of bridging that gap.

S

ettings Dialog

There are many codeblocks for changing out the colors in one go, for example:

```
self.mw.update_all_view_select_box_colors(
                self.accept["display_selectbox_left_color"],
                self.accept["display_selectbox_left_fill"],
                self.accept["display_selectbox_right_color"],
                self.accept["display_selectbox_right_fill"],
                self.preview["display_selectbox_alpha"])
```

This could be replaced with a simpler call

```
self.mw.update_all_view_select_box_colors(
                self.accept["display_selectbox_colors"],
                self.preview["display_selectbox_alpha"])
```

where we require that

```
self.accept["display_selectbox_colors"] == {
    "left_color": "#color",
    "left_fill": "#color",
    "right_color": "#color",
    "right_fill": "#color"
}
```

with `#color` being some valid hex code.

Kivy

Once the tkinter interface is up and running we can experiment with different frontends to improve the look of the application. For example, the MIT licensed KIVY would allow us to replace the mobile development in Swift and Java with all Python development:

`https://kivy.org/#home`

D

ata/Code Seperation

All the ”data” is in code files that are within the ‘config/‘ submodule. So this way we don’t have to deal with awkward data packaging, it’s just available as a single JSON style object called ‘settings‘ available with this import line:

```
from embroidermodder.config import settings
```

In order to pass PyLint style guides this will be split up and formatted into Python code but no processing beyond inlining the data into a single dict should be carried out here.

The Settings Dictionary

No more than 4 levels of indentation

Only strings, arrays, dicts and integers so matching the JSON standard. Ideally you should be able to copy/paste the data in and out and it would parse as JSON. Currently this fails because we have multi-line strings in Python syntax and inlining.

We may be able to extend the lisp support, which would deal with this. Or we can change multiline strings out for arrays of strings.

Lisp Expression Support

In order to safely support user contributed/shared data that can define, for example, double to double functions we need a consistent processor for these descriptions.

Embroidermodder uses a list processor (a subset of the language Lisp which is short for LISt Processor) to accomplish this.

For example the string:

```
(+ (* t 2) 5)
```

is equivalent to the expression:

```
2*t + 5
```

The benefit of not allowing this to simply be a Python expression is that it is safe against malicious use, or accidental misuse. The program can identify whether the output is of the appropriate form and give finitely many calculations before declaring the function to have run too long (stopping equations that hang).

To see examples of this see 'parser.py' and 'config/design_primatives.py'.

It's also worth noting that we don't use the simpler reverse Polish notation (RPN) approach because:

1. It's more compact to use Lisp because 'a b c + +' for example needs a new '+' sign for each new term as opposed to '(+ a b c)'.
2. It's easier to support expressions that are themselves function calls defined by the user (by adding support for 'defun' or 'lambda'.

S

VG Icons

To make the images easier to alter and restyle we could switch to svg icons. There's some code in the git history to help with this.

T

he Actions System

In order to simplify the development of a GUI that is flexible and easy to understand to new developers we have a custom action system that all user actions will go via an 'actuator' that takes a string argument. By using a string argument the undo history is just an array of strings.

The C 'action_hash_data' struct will contain: the icon used, the labels for the menus and tooltips and the function pointer for that action. There will be an accompanying argument for this function call, currently being drafted as 'action_call'. So when the user makes a function call it should contain information like the mouse position, whether special key is pressed etc.

A

ccessibility

Software can be more or less friendly to people with dylexia, partial sightedness, reduced mobility and those who don't speak English. Embroidermodder 2 has, in its design, the following features to help:

* icons for everything to reduce the amount of reading required * the system font is configurable: if you have a dyslexia-friendly font you can load it * the interface rescales to help with

partial-sightedness * the system language is configurable, unfortunately the docs will only be in English but we can try to supply lots of images of the interface to make it easier to understand as a second language * buttons are remappable: XBox controllers are known for being good for people with reduced mobility so remapping the buttons to whatever setup you have should help

Note that most of these features will be released with version 2.1, which is planned for around early 2023.

C

urrent work

1. Converting C++ to Python throughout.
2. OpenGL Rendering
   (a) "Real" rendering to see what the embroidery looks like. 2. Icons and toolbars. 3. Menu bar
3. Libembroidery interfacing: 1. Get all classes to use the proper libembroidery types within them. So 'Ellipse' has 'EmbEllipse' as public data within it. 2. Move calculations of rotation and scaling into 'EmbVector' calls. 1. Get undo history widget back (BUG). 2. Switch website to a CMake build. 3. GUI frontend for embroider features that aren't supported by embroidermodder: flag selector from a table 4. Update all formats without color to check for edr or rgb files. 5. EmbroideryFLOSS - Color picker that displays catalog numbers and names 6. Setting for reverse scrolling direction (for zoom, vertical pan) 7. Stitching simulation 8. User designed custom fill 9. Keyboard zooming, panning 10. Advanced printing 11. Libembroidery 1.0 12. Better integrated help: I don't think the help should backend to a html file somewhere on the user's system. A better system would be a custom widget within the program that's searchable. 13. New embroidermodder2.ico 16x16 logo that looks good at that scale. 14. saving dst, pes, jef 15. Settings dialog: notify when the user is switching tabs that the setting has been changed, adding apply button is what would make sense for this to happen. 16. Update language translations 17. Replace KDE4 thumbnailer. 18. Import raster image 19. Statistics from 1.0, needs histogram. 20. SNAP/ORTHO/POLAR 21. Cut/copy allow post-selection 22. Layout into config 23. Notify user of data loss if not saving to an object format. 24. Add which formats to work with to preferences. 25. Cannot open file with # in the name when opening multiple files but works with opening a single file. 26. Closing settings dialog with the X in the window saves settings rather than discarding them. 27. Otto theme icons: units, render, selectors, what's this icon doesn't scale 28. Layer manager and Layer switcher dock widget 29. test that all formats read data in correct scale (format details should match other programs). 30. Custom filter bug – doesn't save changes in some cases. 31. Get flake8, pylint and tests to pass. 32. Sphinx documentation from docstrings or similar.

For more details read on into the Design section.

S

ample Files

Various sample embroidery design files can be found in the embroidermodder2/samples folder.

6.0.5 D

esign

These are key bits of reasoning behind why the software is built the way it is.

C

AD command review

— ID — Name — Arguments — Description — ——————————— — 0 — newfile — none — Create a new EmbPattern with a new tab in the GUI. — — 1 — openfile — ‘char *fname;‘ — Open an EmbPattern with the supplied filename ‘fname‘. — — 2 — savefile — ‘char *fname;‘ — Save the current loaded EmbPattern to the supplied filname ‘fname‘. — — 1 — scale — selected objects, 1 float — Scale all selected objects by the number supplied, without selection scales the entire design — — 2 — circle — mouse co-ords — Adds a circle to the design based on the supplied numbers, converts to stitches on save for stitch only formats. — — 3 — offset — mouse co-ords — Shifts the selected objects by the amount given by the mouse co-ordinates. — — 4 — extend — — — — 5 — trim — — — — 6 — BreakAtPoint — — — — 7 — Break2Points — — — — 8 — Fillet — — — — 9 — star — — — — 10 — singlelinetext — — — — 11 — Chamfer — — — — 12 — split — — — — 13 — area — — — — 14 — time — — — — 15 — pickadd — — — — 16 — zoomfactor — — — — 17 — product — — — — 18 — program — — — — 19 — zoomwindow — — — — 20 — divide — — — — 21 — find — — — — 22 — record — — — — 23 — playback — — — — 24 — rotate — — — — 25 — rgb — — — — 26 — move — — — — 27 — grid — — — — 28 — griphot — — — — 29 — gripcolor — — — — 30 — gripcool — — — — 31 — gripsize — — — — 32 — highlight — — — — 33 — units — — — — 34 — locatepoint — — — — 35 — distance — — — — 36 — arc — — — — 37 — ellipse — — — — 38 — array — — — — 39 — point — — — — 40 — polyline — — — — 41 — polygon — — — — 42 — rectangle — — — — 43 — line — — — — 44 — arc (rt) — — — — 45 — dolphin — — — — 46 — heart — — —

R

emoved Elements

So I’ve had a few pieces of web infrastructure fail me recently and I think it’s worth noting. An issue that affects us is an issue that can effect people who use our software.

Q

t and dependencies

Downloading and installing Qt has been a pain for some users (46Gb on possibly slow connections).

I’m switching to FreeGLUT 3 (which is a whole other conversation) which means we can ship it with the source code package meaning only a basic build environment is necessary to build it.

S

ocial Platform

Github is giving me a server offline (500) error and is still giving a bad ping.

So... all the issues and project boards etc. being on Github is all well and good assuming that we have our own copies. But we don’t if Github goes down or some other major player takes over the space and we have to move (again, since this started on SourceForge).

This file is a backup for that which is why I’m repeating myself between them.

P

andoc Documentation

The documentation is, well better in that it's housed in the main repository, but I'm not a fan of the "write once build many" approach as it means trying to weigh up how 3 versions are going to render.

Can we treat the website being a duplicate of the docs a non-starter? I'd be happier with tex/pdf only and (I know this is counter-intuitive) one per project.

O

penGL

OpenGL rendering within the application. This will allow for Realistic Visualization - Bump Mapping/OpenGL/Gradients?

This should backend to a C renderer or something.

C

onfiguration Data Ideas

embroidermodder should boot from the command line regardless of whether it is or is not installed (this helps with testing and running on machines without root). Therefore, it can create an initiation file but it won't rely on its existence to boot: ' /.embroidermodder/config.json'.

1. Switch colors to be stored as 6 digit hexcodes with a `#`.
2. We've got close to a hand implemented ini read/write setup in `settings.py`.

Distribution

When we release the new pip wheel we should also package:

- `.tar.gz` and `.zip` source archive.
- Debian package
- RPM package

Only do this once per minor version number.

S

cripting Overhaul

Originally Embroidermodder had a terminal widget, this is why we removed it.

¿ ROBIN: I think supporting scripting within Embroidermodder doesn't make sense. ¿ ¿ All features that use scripting can be part of libembroidery instead. ¿ Users who are capable of using scripting won't need it, they can alter their embroidery files in CSV format, or import pyembroidery to get access. ¿ It makes maintaining the code a lot more complicated, especially if we move away from Qt. ¿ Users who don't want the scripting feature will likely be confused by it, since we say that's what libembroidery, embroider and pyembroidery are for. ¿ ¿ How about a simpler "call user shell" feature? Similar to texmaker we just call system on a batch or shell script supplied by the user and it processes the file directly then the software reloads the file. Then we aren't parsing it directly. ¿ ¿I don't want to change this without Josh's support because it's a fairly major change. ¿ ¿¿ JOSH: I totally agree. ¿¿ ¿¿ I like the idea of scripting just so people that know how to code could write their own designs without needing to fully

build the app. Scripting would be a very advanced feature that most users would be confused by. Libembroidery would be a good fit for advanced features.

Now we are using Python (again, sort of) this would be a lot more natural, perhaps we could boot the software without blocking the shell so they can interact? TODO: Screenshot a working draft to demonstrate.

6.0.6 P

erennial Jobs

1. Check for memory leaks 2. Write new tests for new code. 3. Get Embroidermodder onto the current version of libembroidery-python. 4. PEP8 compliance. 5. Better documentation with more photos/screencaps.

D

eveloping for Android

https://developer.android.com/studio/projects/add-native-code

apt install google-android-ndk-installer cmake lldb gradle

6.1 L

ibembroidery v1.0-alpha Manual

(Under construction, please wait for v1.0 release.)

6.1.1 1

.1. What is libembroidery?

libembroidery is a low-level library for reading, writing, and altering digital embroidery files in C.

libembroidery is the underlying library that is used by [Embroidermodder 2](http://embroidermodder.org and is developed by [The Embroidermodder Team](link to the-embroidermodder-team). It handles over 45 different embroidery specific formats as well as several non-embroidery specific vector formats.

It also includes a CLI called `embroider` that allows for better automation of changes to embroidery files and will be more up-to date than the Embroidermodder 2 GUI.

If you want to find a simple fix to contribute see the *Development* section of the manual.

6.1.2 1

.2. Table of Contents

1

.2.1. License

Libembroidery is distributed under the permissive zlib licence, see the LICENCE file. This applies to all the source code in this directory.

1

.2.2. The Embroidermodder Project

The *Embroidermodder 2* project is a collection of small software utilities for manipulating, converting and creating embroidery files in all major embroidery machine formats. The program *Embroidermodder 2* itself is a larger graphical user interface (GUI) which is at the heart of the project.

This manual, the website ('embroidermodder.org'), mobile embroidery format viewers and tools ('iMobileViewer', 'MobileViewer'), the core library of functions ('libembroidery') and CLI ('embroider') are all tools to make the standard user experience of working with an embroidery machine better without expensive software which is locked to specific manufacturers and formats. But ultimately we hope that the core *Embroidermodder 2* is a practical, ever-present tool in larger workshops, small cottage industry workshops and personal hobbyist's bedrooms.

Embroidermodder 2 is licensed under the zlib license and we aim to keep all of our tools open source and free of charge. If you would like to support the project check out our Open Collective group. If you would like to help, please join us on GitHub. This document is written as developer training as well helping new users (see the last sections) so this is the place to learn how to start changing the code.

1

.2.3. The Embroidermodder Team

The Embroidermodder Team is the collection of people who've submitted patches, artwork and documentation to our three projects. The team was established by Jonathan Greig and Josh Varga. For a full list of members please see the [Embroidermodder github page](https://github.com/Embroid
where it is actively maintained.

6.1.3 1

.3. Build

libembroidery and EmbroiderModder 2 use CMake builds so if you are building the project to use as a library we recommend you run:

"' git clone https://github.com/Embroidermodder/libembroidery cd libembroidery cmake . cmake –build . cmake –install . "'

This builds both the static and shared versions of the library as well as the command line program 'embroider'.

1

.3.1. Debug

If you wish to help with development, [run this debug script](`https://embroidermodder.org/libembroidery_debugger.sh`) and send us the error log. Note: this is maintained outside of the repository because it gives us a oneliner for new systems with a stable URL:

```
curl https://embroidermodder.org/libembroidery_debugger.sh | sh
```

While we will attempt to maintain good results from this script as part of normal development it should be the first point of failure on any system we haven't tested or format we understand less.

6.1.4 Usage

For basic use, we recommend you build as above, then run without arguments:

```
$ embroider
```

which will print out this advice on how to use these tools without digging straight into the rest of this manual.

```
EMBROIDER
    A command line program for machine embroidery.
    Copyright 2013-2021 The Embroidermodder Team
    Licensed under the terms of the zlib license.

    https://github.com/Embroidermodder/libembroidery
    https://embroidermodder.org

Usage: embroider [OPTIONS] fileToRead...

Conversion:
-t, -to         Convert all files given to the format specified
                by the arguments to the flag, for example:
                    $ embroider -t dst input.pes
                would convert \"input.pes\" to \"input.dst\"
                in the same directory the program runs in.

                The accepted input formats are (TO BE DETERMINED).
                The accepted output formats are (TO BE DETERMINED).

Output:
-h, -help       Print this message.
-f, -format     Print help on the formats that
                embroider can deal with.
-q, -quiet      Only print fatal errors.
-V, -verbose    Print everything that has reporting.
-v, -version    Print the version.

Graphics:
-c, -circle     Add a circle defined by the arguments
                given to the current pattern.
-e, -ellipse    Add a circle defined by the arguments
                given to the current pattern.
-l, -line       Add a line defined by the arguments
                given to the current pattern.
-P, -polyline   Add a polyline.
-p, -polygon    Add a polygon.
-s, -satin      Fill the current geometry with satin
                stitches according
                to the defined algorithm.
-S, -stitch     Add a stitch defined by the arguments
                given to the current pattern.
```

```
Quality Assurance:
    -test       Run the test suite.
```

For each of the flags described here we will go into greater detail in this manual.

To Flag

Circle Flag

Ellipse Flag

1

.4.4. Line Flag

1

.4.5. Polyline Flag

1

.4.6. Polygon Flag

1

.4.7. Satin Flag

1

.4.8. Stitch Flag

1

.4.9.

1

.4.10. Basic Test Suite

The flag '–test' runs the tests that take the least time and have the most utility. If you're submitting a patch for review, please run:

```
$ embroider --test | tail -n 1
```

You'll be presented with an overall PASS or FAIL for your build, if your build fails you can try and trace the error with:

```
$ valgrind embroider --verbose --test
```

or

```
$ gdb --args embroider --verbose --test
```

depending on your preferred debugging approach. Passing this test will be required for us to accept your patch.

Full Test Suite

The flag '–full-test-suite' runs all the tests that have been written. Since this results in a lot of output the details are both to stdout and to a text file called 'test_matrix.txt'.

Patches that strictly improve the results in the 'test_matrix.txt' over the current version will likely be accepted and it'll be a good place to go digging for contributions. (Note: strictly improve means that the testing result for each test is as good a result, if not better. Sacrificing one critera for another would require some design work before we would consider it.)

6.1.5 Ideas

Rendering system

There are two forms of render that will be produced.

1. A raster format as ppm so we can have a pixel for pixel output (for example extracting the embedded images in some formats). 2. The SVG format that will be fairly similar to InkStitch's format.

We have an EmbImage struct to store the raster format.

```
$ embroider test01.csv --render
```

currently creates a blank image. Previously the Hilbert curve test managed to create a correctly rendered version.

Binary download

We need a current 'embroider' command line program download, so people can update without building.

Identify the meaning of these TODO items

- Saving CSV/SVG (rt) + CSV read/write UNKNOWN interpreted as COLOR bug #179
- Lego Mindstorms NXT/EV3 ports and/or commands

Progress Chart

The chart of successful from-to conversions (previously a separate issue) is something that should appear in the README.

Tactile art and braille support

One application I'd like to leave a reminder here for is automating embroidery for blind and partially sighted people.

There are many limitations to making braille (cost, poor support, lack of widespread adoption in the sighted world) and as such there is a strong DIY culture around it.

There are blind internet users who can also run terminal applications using a refreshable braille display, so in theory we could support an application like this for them:

```
$ embroider --braille "Hello, world!" hello.dst
```

which would produce braille that would read "Hello, world!" as an embroidery design.

Another option is tactile fills that use the same fill algorithms but are designed better to facilitate tactile art.

I think the way forward on this is to call up the RNIB business advice line and ask for assistance once we have a working model. That way they can get us in contact with experts to review how legible the output is and usable the software is for the intended audience.

This is less important than getting better machine support but given the high social impact I think it should be a priority.

6.1.6 Development

Contributing

If you're interested in getting involved, here's some guidance for new developers. Currently The Embroidermodder Team is all hobbyists with an interest in making embroidery machines more open and user friendly. If you'd like to support us in some other way you can donate to our Open Collective page (click the Donate button) so we can spend more time working on the project.

All code written for libembroidery should be ANSI C89 compliant if it is C. Using other languages should only be used where necessary to support bindings.

Style

Rather than maintain our own standard for style, please defer to the Python's PEP 7 ([12](#12)) for C style. If it passes the linters for that we consider it well styled for a pull request.

As for other languages we have no house style other than whatever "major" styles exist, for example Java in Google style ([13](#13)) would be acceptable. We'll elect specific standards if it becomes an issue.

Standard

The criteria for a good Pull Request from an outside developer is, from most to least important:

1. No regressions on testing.
2. Add a feature, bug fix or documentation that is already agreed on through GitHub issues or some other way with a core developer.
3. No GUI specific code should be in libembroidery, that's for Embroidermodder. 4. Pedantic/ansi C unless there's a good reason to use another language. 5. Meet the style above (i.e. [PEP 7, Code Lay-out](`https://peps.python.org/pep-0007/#code-lay-out`)). We'll just fix the style if the code's good and it's not a lot of work. 6. 'embroider' should be in POSIX style as a command line program. 7. No dependancies that aren't "standard", i.e. use only the C Standard Library.

1

.6.2. Image Fitting

A currently unsolved problem in development that warrants further research is the scenario where a user wants to feed embroider an image that can then be .

1

.6.3. To Place

A *right-handed coordinate system* is one where up is positive and right is positive. Left-handed is up is positive, left is positive. Screens often use down is positive, right is positive,

including the OpenGL standard so when switching between graphics formats and stitch formats we need to use a vertical flip (‘embPattern_flip‘).

‘0x20‘ is the space symbol, so when padding either 0 or space is preferred and in the case of space use the literal ’ ’.

1

.6.4. To Do

We currently need help with:

1. Thorough descriptions of each embroidery format. 2. Finding resources for each of the branded thread libraries (along with a full citation for documentation). 3. Finding resources for each geometric algorithm used (along with a full citation for documentation). 4. Completing the full ‘–full-test-suite‘ with no segfaults and at least a clear error message (for example ”not implemented yet”). 5. Identifying ”best guesses” for filling in missing information when going from, say ‘.csv‘ to a late ‘.pes‘ version. What should the default be when the data doesn’t clarify? 6. Improving the written documentation. 7. Funding, see the Sponsor button above. We can treat this as ”work” and put far more hours in with broad support in small donations from people who want specific features.

Beyond this the development targets are categories sorted into:

1. [Basic Features](#basic-features) 2. [Code quality and user friendliness](#code-quality-and-user-friendliness) 3. [embroider CLI](#embroider-cli) 4. [Documentation](#documentation) 5. [GUI](#gui) 6. [electronics development](#electronics-development)

1

.6.4.1. Basic features.

1. Incorporate `#if 0`ed parts of `libembroidery.c`. 2. Interpret how to write formats that have a read mode from the source code and vice versa. 3. Document the specifics of the file formats here for embroidery machine specific formats. Find websites and other sources that break down the binary formats we currently don’t understand. 4. Find more and better documentation of the structure of the headers for the formats we do understand.

1

.6.4.2. Code quality and user friendliness

1. Document all structs, macros and functions (will contribute directly on the web version). 2. Incorporate experimental code, improve support for language bindings. 3. Make stitch x, y into an EmbVector.

1

.6.4.3. embroider CLI

1. Make -circle flag to add a circle to the current pattern. 2. Make -rect flag to add a rectangle to the current pattern. 3. Make -fill flag to set the current satin fill algorithm for the current geometry. (for example “-fill crosses -circle 11,13,10” fills a circle with center 11mm, 13mm with radius 10mm with crosses). 4. Make -ellipse flag to add to ellipse to the current pattern. 5. Make -bezier flag to add a bezier curve to the current pattern.

1

.6.4.3.1. Embroider pipeline

Adjectives apply to every following noun so

“‘ embroider –satin 0.3,0.6 –thickness 2 –circle 10,20,5 –border 3 –disc 30,40,10 –arc 30,50,10,60 output.pes “‘

Creates:

1. a circle with properties: thickness 2, satin 0.3,0.6 2. a disc with properties: 3. an arc with properties:

in that order then writes them to the output file ‘output.pes‘.

1

.6.4.4. Documentation

1. Create csv data files for thread tables. 2. Convert tex to markdown, make tex an output of ‘build.bash‘. 3. Run ‘sloccount‘ on ‘extern/‘ and ‘.‘ (and) so we know the current scale of the project, aim to get this number low. Report the total as part of the documentation. 4. Try to get as much of the source code that we maintain into C as possible so new developers don’t need to learn multiple languages to have an effect. This bars the embedded parts of the code.

1

.6.4.5. GUI

1. Make MobileViewer also backend to ‘libembroidery‘ with a Java wrapper. 2. Make iMobileViewer also backend to ‘libembroidery‘ with a Swift wrapper. 3. Share some of the MobileViewer and iMobileViewer layout with the main EM2. Perhaps combine those 3 into the Embroidermodder repository so there are 4 repositories total. 4. Convert layout data to JSON format and use cJSON for parsing.

1

.6.4.6. Electronics development

1. Currently experimenting with Fritzing[8], upload netlists to embroiderbot when they can run simulations using the asm in ‘libembroidery‘. 2. Create a common assembly for data that is the same across chipsets ‘libembrodiery_data_internal.s‘. 3. Make the defines part of ‘embroidery.h‘ all systems and the function list ”c code only”. That way we can share some development between assembly and C versions.

6.1.7 1

.7. Formats

1

.7.1. Overview

1

.7.1.1. Read/Write Support Levels

The table of read/write format support levels uses the status levels described here:

— Status — Description — ————— — None (0) — Either the format produces no output, reporting an error. Or it produces a Tajima dst file as an alternative. — — Poor (1) — A file somewhat similar to our examples is produced. We don’t know how well it runs on machines in practice as we don’t have any user reports or personal tests. — — Basic (2) — Simple files in this format run well on machines that use this format. — — Standard (3) — Files with non-standard features work on machines and we have good documentation on the

format. — — Reliable (4) — All known features don't cause crashes. Almost all work as expected. — — Good (5) — All known features of the format work on machines that use this format. Translations from and to this format preserve all features present in both. —

So all formats can, in principle, have good read and good write support, because it's defined in relation to files that we have described the formats for.

1

.7.1.2. Test Support Levels

— Status — Description — ————— — None (0) — No tests have been written to test the specifics of the format. — — Basic (1) — Stitch Lists and/or colors have read/write tests. — — Thorough (2) — All features of that format has at least one test. — — Fuzz (2) — Can test the format for uses of features that we haven't thought of by feeding in nonsense that is designed to push possibly dangerous weaknesses to reveal themselves. — — Complete (3) — Both thorough and fuzz testing is covered. —

So all formats can, in principle, have complete testing support, because it's defined in relation to files that we have described the formats for.

1

.7.1.3. Documentation Support Levels

— Status — Description — ————— — None (0) — We haven't researched this beyond finding example files. — — Basic (1) — We have a rough sketch of the size and contents of the header if there is one. We know the basic stitch encoding (if there is one), but not necessarily all stitch features. — — Standard (2) — We know some good sources and/or have tested all the features that appear to exist. They mostly work the way we have described. — — Good (3) — All features that were described somewhere have been covered here or we have thoroughly tested our ideas against other softwares and hardwares and they work as expected. — — Complete (4) — There is a known official description and our description covers all the same features. —

Not all formats can have complete documentation because it's based on what information is publically available. So the total score is reported in the table below based on what level we think is available.

1

.7.1.4. Table of Format Support Levels

— *Format* — *Read Support* — *Write Support* — *Specialised Tests* — *Documentation* — *Score* — ——————————— — [Toyota Embroidery Format (.100)](link to toyota-embroidery-format-100) — Basic — Basic — None — None — 2/11 — — [Toyota Embroidery Format (.10o)](link to toyota-embroidery-format-10o) — Basic — Basic — None — None — 2/11 — — [Bernina Embroidery Format (.art)](link to bernina-embroidery-format-art) — None — None — None — None — 0/11 — — [Bitmap Cache Embroidery Format (.bmc)](link to bitmap-cache-embroidery-format-bmc) — Basic — None — None — None — 1/11 — — [Bits and Volts Embroidery Format (.bro)](link to bits-and-volts-embroidery-format-bro) — None — None — None — None — 0/11 — — [Melco Embroidery Format (.cnd)](link to melco-embroidery-format-cnd) — None — None — None — None — 0/11 — — [Embroidery Thread Color Format (.col)](link to embroidery-thread-color-format-col) — Basic — Basic — None — Basic — 0/11 — — [Singer Embroidery Format (.csd)](link to singer-embroidery-format-csd) — None — None — None — None — 0/11 — — [Comma Separated Values (.csv)](link to comma-seperated-values-csv) — None — None — None — None — 0/11 — — [Barudan Embroidery Format (.dat)](link to barudan-embroidery-format-dat) — None —

None — None — None — 0/11 — — [Melco Embroidery Format (.dem)](link to melco-embroidery-format-dem) — None — None — None — None — 0/11 — — [Barudan Embroidery Format (.dsb)](link to barudan-embroidery-format-dsb) — None — None — None — None — 0/11 — — [Tajima Embroidery Format (.dst)](link to tajima-embroidery-format-dst) — None — None — None — None — 0/11 — — [ZSK USA Embroidery Format (.dsz)](link to zsk-usa-embroidery-format-dsz) — None — None — None — None — 0/11 — — [Drawing Exchange Format (.dxf)](link to drawing-exchange-format-dxf) — None — None — None — None — 0/11 — — [Embird Embroidery Format (.edr)](link to embird-embroidery-format-edr) — None — None — None — None — 0/11 — — [Elna Embroidery Format (.emd)](link to elna-embroidery-format-emd) — None — None — None — None — 0/11 — — [Melco Embroidery Format (.exp)](link to melco-embroidery-format-exp) — None — None — None — None — 0/11 — — [Eltac Embroidery Format (.exy)](link to eltac-embroidery-format-exy) — None — None — None — None — 0/11 — — [Sierra Expanded Embroidery Format (.eys)](link to sierra-expanded-embroidery-format-eys) — None — None — None — None — 0/11 — — [Fortron Embroidery Format (.fxy)](link to fortran-embroidery-format-fxy) — None — None — None — None — 0/11 — — [Smoothie G-Code Embroidery Format (.gc)](link to smoothie-g-code-embroidery-format-gc) — None — None — None — None — 0/11 — — [Great Notions Embroidery Format (.gnc)](link to great-notions-embroidery-format-gnc) — None — None — None — None — 0/11 — — [Gold Thread Embroidery Format (.gt)](link to gold-thread-embroidery-format-gt) — None — None — None — None — 0/11 — — [Husqvarna Viking Embroidery Format (.hus)](link to husqvarna-viking-embroidery-format-hus) — None — None — None — None — 0/11 — — [Inbro Embroidery Format (.inb)](link to inbro-embroidery-format-inb) — None — None — None — None — 0/11 — — [Embroidery Color Format (.inf)](link to embroidery-color-format-inf) — None — None — None — None — 0/11 — — [Janome Embroidery Format (.jef)](link to janome-embroidery-format-jef) — None — None — None — None — 0/11 — — [Pfaff Embroidery Format (.ksm)](link to pfaff-embroidery-format-ksm) — None — None — None — None — 0/11 — — [Pfaff Embroidery Format (.max)](link to pfaff-embroidery-format-max) — None — None — None — None — 0/11 — — [Mitsubishi Embroidery Format (.mit)](link to mitsubishi-embroidery-format-mit) — None — None — None — None — 0/11 — — [Ameco Embroidery Format (.new)](link to ameco-embroidery-format-new) — None — None — None — None — 0/11 — — [Melco Embroidery Format (.ofm)](link to melco-embroidery-format-ofm) — None — None — None — None — 0/11 — — [Pfaff Embroidery Format (.pcd)](link to pfaff-embroidery-format-pcd) — None — None — None — None — 0/11 — — [Pfaff Embroidery Format (.pcm)](link to pfaff-embroidery-format-pcm) — None — None — None — None — 0/11 — — [Pfaff Embroidery Format (.pcq)](link to pfaff-embroidery-format-pcq) — None — None — None — None — 0/11 — — [Pfaff Embroidery Format (.pcs)](link to pfaff-embroidery-format-pcs) — None — None — None — None — 0/11 — — [Brother Embroidery Format (.pec)](link to brother-embroidery-format-pec) — None — None — None — None — 0/11 — — [Brother Embroidery Format (.pel)](link to brother-embroidery-format-pel) — None — None — None — None — 0/11 — — [Brother Embroidery Format (.pem)](link to brother-embroidery-format-pem) — None — None — None — None — 0/11 — — [Brother Embroidery Format (.pes)](link to brother-embroidery-format-pes) — None — None — None — None — 0/11 — — [Brother Embroidery Format (.phb)](link to brother-embroidery-format-phb) — None — None — None — None — 0/11 — — [Brother Embroidery Format (.phc)](link to brother-embroidery-format-phc) — None — None — None — None — 0/11 — — [AutoCAD Embroidery Format (.plt)](link to brother-embroidery-format-plt) — None — None — None — None — 0/11 — — [RGB Embroidery Format (.rgb)](link to rgb-embroidery-format-rgb) — None — None — None — None — 0/11 — — [Janome Embroidery Format (.sew)](link to janome-embroidery-format-sew) — None — None — None — None — 0/11 — — [Husqvarna Viking Embroidery Format (.shv)](link to husqvarna-viking-embroidery-format-shv) — None

— None — None — None — 0/11 — — [Sunstar Embroidery Format (.sst)](link to sunstar-embroidery-format-sst) — None — None — None — None — 0/11 — — [Data Stitch Embroidery Format (.stx)](link to data-stitch-embroidery-format-stx) — None — None — None — None — 0/11 — — [Scalable Vector Graphics (.svg)](link to scalable-vector-graphics-svg) — None — None — None — None — 0/12 — — [Pfaff Embroidery Format (.t01)](link to pfaff-embroidery-format-t01) — None — None — None — None — 0/11 — — [Pfaff Embroidery Format (.t09)](link to pfaff-embroidery-format-t09) — None — None — None — None — 0/11 — — [Happy Embroidery Format (.tap)](link to happy-embroidery-format-tap) — None — None — None — None — 0/11 — — [ThredWorks Embroidery Format (.thr)](link to thredworks-embroidery-format-thr) — None — None — None — None — 0/11 — — [Text File (.txt)](link to text_file_txt) — None — None — None — None — 0/11 — — [Barudan Embroidery Format (.u00)](link to barudan-embroidery-format-u00) — None — None — None — None — 0/11 — — [Barudan Embroidery Format (.u01)](link to barudan-embroidery-format-u01) — None — None — None — None — 0/11 — — [Pfaff Embroidery Format (.vip)](link to pfaff-embroidery-format-vip) — None — None — None — None — 0/11 — — [Pfaff Embroidery Format (.vp3)](link to pfaff-embroidery-format-vp3) — None — None — None — None — 0/11 — — [Singer Embroidery Format (.xxx)](link to singer-embroidery-format-xxx) — None — None — None — None — 0/11 — — [ZSK USA Embroidery Format (.zsk)](link to zsk_usa_embroidery_format_zsk) — None — — None — None — None — 0/11 — — *Total* — - — - — - — - — 0/671 —

For a total of 0

1

.7.2. Toyota Embroidery Format (.100)

The Toyota 100 format is a stitch-only format that uses an external color file.
The stitch encoding is in 4 byte chunks.

1

.7.3. Toyota Embroidery Format (.10o)

The Toyota 10o format is a stitch-only format that uses an external color file.
The stitch encoding is in 3 byte chunks.

1

.7.4. Bernina Embroidery Format (.art)

We don't know much about this format. TODO: Find a source.

1

.7.5. Bitmap Cache Embroidery Format (.bmc)

We don't know much about this format. TODO: Find a source.

1

.7.6. Bits and Volts Embroidery Format (.bro)

The Bits and Volts bro format is a stitch-only format that uses an external color file.
The header is 256 bytes. There's a series of unknown variables in the header.
The stitch list uses a variable length encoding which is 2 bytes for any stitch

1

.7.7. Melco Embroidery Format (.cnd)

The Melco cnd format is a stitch-only format.

We don't know much about this format. TODO: Find a source.

1

.7.8. Embroidery Thread Color Format (.col)

An external color file format for formats that do not record their own colors.

It is a human-readable format that has a header that is a single line containing only the number of threads in decimal followed by the windows line break

```
'\r\n'
```

.

Then the rest of the file is a comma seperated value list of all threads with 4 values per line: the index of the thread then the red, green and blue channels of the color in that order.

1

.7.8.1. Example

If we had a pattern called "example" with four colors: black, red, magenta and cyan in that order then the file is (with the white space written out):

example.col

```
4\r\n
0,0,0,0\r\n
1,255,0,0\r\n
2,0,255,0\r\n
3,0,0,255\r\n
```

1

.7.9. Singer Embroidery Format (.csd)

Stitch Only Format

1

.7.10. Comma Separated Values (.csv)

Comma Seperated Values files aren't a universal system, here we aim to offer a broad support.

— Control Symbol — Type — — ———————— — '#' — COMMENT — — — '¿' — VARIABLE — To store records of a pattern's width, height etc. This means that data stored in the header of say a '.dst' file is preserved. — — '$' — THREAD — — — '*' — STITCH — — — '*' — JUMP — — — '*' — COLOR — To change a color: used for trim as well — — '*' — END — To end a pattern. — — '*' — UNKNOWN — For any feature that we can't identify. —

Barudan Embroidery Format (.dat)

Stitch Only Format

Melco Embroidery Format (.dem)

Stitch Only Format

Barudan Embroidery Format (.dsb)

- Stitch Only Format.

X Basic Read Support

Basic Write Support

Well Tested Read

Well Tested Write

Tajima Embroidery Format (.dst)

- Stitch Only Format.

X Basic Read Support

X Basic Write Support

Well Tested Read

Well Tested Write

.DST (Tajima) embroidery file read/write routines Format comments are thanks to `tspilman@dalcoathletic.com` who's notes appeared at `http://www.wotsit.org` under Tajima Format.

1

.7.14.1. Header

The header seems to contain information about the design. Seems to be ASCII text delimited by 0x0D (carriage returns). This must be in the file for most new software or hardware to consider it a good file! This is much more important than I originally believed. The header is 125 bytes in length and padded out by 0x20 to 512 bytes total. All entries in the header seem to be 2 ASCII characters followed by a colon, then it's value trailed by a carriage return.

— *C memory* — *Description* — ——————————— — 'char LA[16+1];' — First is the 'LA' entry, which is the design name with no path or extension information. The blank is 16 characters in total, but the name must not be longer that 8 characters and padded out with spaces (0x20). — — 'char ST[7+1];' — Next is the stitch count ST, this is a 7 digit number padded by leading zeros. This is the total stitch count including color changes, jumps, nups, and special records. — — 'char CO[3+1];' — Next, is CO or colors, a 3 digit number padded by leading zeros. This is the number of color change records in the file. — — 'char POSX[5+1];' — Next is +X or the positive X extent in centimeters, a 5 digit non-decimal number padded by leading zeros. — — 'char NEGX[5+1];' — Following is the -X or the negative X extent in millimeters, a 5 digit non-decimal number padded by leading zeros. — — 'char POSY[5+1];' — Again, the +Y extents. — — 'char NEGY[5+1];' — Again, the -Y extents. — — 'char AX[6+1]; char AY[6+1];' — AX and AY should express the relative coordinates of the last point from the start point in 0.1 mm. If the start and last points are the same, the coordinates are (0,0). — — 'char MX[6+1];' 'char MY[6+1];' — MX and MY should express coordinates of the last point of the previous file for a multi-volume design. A multi-volume design means a design consisted of two or more files. This was used for huge designs that can not be stored in a single paper tape roll. It is not used so much (almost never) nowadays. — — 'char PD[9+1];' — PD is also storing some information for multi-volume design. —

Uses 3 byte per stitch encoding with the format as follows:

The 3 byte encoding for the dxf format.

— *Bit* — *7* — *6* — *5* — *4* — *3* — *2* — *1* — *0* — ——————————————————————— — Byte 0 — y+1 — y-1 — y+9 — y-9 — x-9 — x+9 — x-1 — x+1 — — Byte 1 — y+3 — y-3 — y+27 — y-27 — x-27 — x+27 — x-3 — x+3 — — Byte 2 — jump — color change — y+81 — y-81 — x-81 — x+81 — set — set —

T01 and Tap appear to use Tajima Ternary.

Where the stitch type is determined as:

- Normal Stitch '00000011 0x03'
- Jump Stitch '10000011 0x83'
- Stop/Change Color '11000011 0xC3'
- End Design '11110011 0xF3'

Inclusive or'ed with the last byte.

Note that:

1. The max stitch length is the largest sum of '1+3+9+27+81=121' where the unit length is 0.1mm so 12.1mm. 2. The coordinate system is right handed.

1

.7.15. ZSK USA Embroidery Format (.dsz)

The ZSK USA dsz format is stitch-only.

1

.7.16. Drawing Exchange Format (.dxf)

Graphics format.

1

.7.17. Embird Embroidery Format (.edr)

Stitch Only Format

1

.7.18. Elna Embroidery Format (.emd)

Stitch Only Format.

1

.7.19. Melco Embroidery Format (.exp)

Stitch Only Format.

1

.7.20. Eltac Embroidery Format (.exy)

Stitch Only Format.

1

.7.21. Sierra Expanded Embroidery Format (.eys)

Stitch Only Format.

Smoothie G-Code Embroidery Format (.fxy)?

1

.7.22. Fortron Embroidery Format (.fxy)
Stitch Only Format.

1

.7.23. Great Notions Embroidery Format (.gnc)
Stitch Only Format.

1

.7.24. Gold Thread Embroidery Format (.gt)
Stitch Only Format.

1

.7.25. Husqvarna Viking Embroidery Format (.hus)
Stitch Only Format.

1

.7.26. Inbro Embroidery Format (.inb)
Stitch Only Format.

1

.7.27. Embroidery Color Format (.inf)
Stitch Only Format.

1

.7.28. Janome Embroidery Format (.jef)
Stitch Only Format.

1

.7.29. Pfaff professional Design format (.ksm)
Stitch Only Format.

1

.7.30. Pfaff Embroidery Format (.max)
Stitch Only Format.

1

.7.31. Mitsubishi Embroidery Format (.mit)
Stitch Only Format.

1

.7.32. Ameco Embroidery Format (.new)
Stitch Only Format.

1

.7.33. Melco Embroidery Format (.ofm)
Stitch Only Format.

1

.7.34. Pfaff PCD File Format
Stitch Only Format.
The format uses a signed 3 byte-length number type.
See the description here ([5](link to 5)) for the overview of the format.
For an example of the format see ([11](link to 11)).

1

.7.35. Pfaff Embroidery Format (.pcm)
The Pfaff pcm format is stitch-only.

1

.7.36. Pfaff Embroidery Format (.pcq)
The Pfaff pcq format is stitch-only.

1

.7.37. Pfaff Embroidery Format (.pcs)
The Pfaff pcs format is stitch-only.

1

.7.38. Brother Embroidery Format (.pec)
The Brother pec format is stitch-only.

1

.7.39. Brother Embroidery Format (.pel)
The Brother pel format is stitch-only.

1

.7.40. Brother Embroidery Format (.pem)
The Brother pem format is stitch-only.

1

.7.41. Brother Embroidery Format (.pes)
The Brother pes format is stitch-only.

1

.7.42. Brother Embroidery Format (.phb)
The Brother phb format is stitch-only.

1

.7.43. Brother Embroidery Format (.phc)
The Brother phc format is stitch-only.

1

.7.44. AutoCAD Embroidery Format (.plt)
The AutoCAD plt format is stitch-only.

1

.7.45. RGB Embroidery Format (.rgb)
The RGB format is a color-only format to act as an external color file for other formats.

1

.7.46. Janome Embroidery Format (.sew)
The Janome sew format is stitch-only.

1

.7.47. Husqvarna Viking Embroidery Format (.shv)
The Husqvarna Viking shv format is stitch-only.

1

.7.48. Sunstar Embroidery Format (.sst)
The Sunstar sst format is stitch-only.

1

.7.49. Data Stitch Embroidery Format (.stx)
The Data Stitch stx format is stitch-only.

S

calable Vector Graphics (.svg)
The scalable vector graphics (SVG) format is a graphics format maintained by ...

P

faff Embroidery Format (.t01)
The Pfaff t01 format is stitch-only.

P

faff Embroidery Format (.t09)
The Pfaff t09 format is stitch-only.

H

appy Embroidery Format (.tap)
The Happy tap format is stitch-only.

T

hredWorks Embroidery Format (.thr)
The ThreadWorks thr format is stitch-only.

T

ext File (.txt)
The txt format is stitch-only and isn't associated with a specific company.

1

.7.56. Barudan Embroidery Format (.u00)
The Barudan u00 format is stitch-only.

1

.7.57. Barudan Embroidery Format (.u01)
The Barudan u01 format is stitch-only.

1

.7.58. Pfaff Embroidery Format (.vip)
The Pfaff vip format is stitch-only.

Pfaff Embroidery Format (.vp3)

The Pfaff vp3 format is stitch-only.

Singer Embroidery Format (.xxx)

The Singer xxx format is stitch-only.

ZSK USA Embroidery Format (.zsk)

The ZSK USA zsk format is stitch-only.

6.1.8 On Embedded Systems

The library is designed to support embedded environments, so it can be used in CNC applications.

Compatible Boards

We recommend using an Arduino Mega 2560 or another board with equal or greater specs. That being said, we have had success using an Arduino Uno R3 but this will likely require further optimization and other improvements to ensure continued compatibility with the Uno. See below for more information.

1

.8.2. Arduino Considerations

There are two main concerns here: Flash Storage & SRAM.

libembroidery continually outgrows the 32KB of Flash storage on the Arduino Uno and every time this occurs, a decision has to be made as to what capabilities should be included or omitted. While reading files is the main focus on arduino, writing files may also play a bigger role in the future. Long term, it would be most practical to handle the inclusion or omission of any feature via a single configuration header file that the user can modify to suit their needs.

SRAM is in extremely limited supply and it will deplete quickly so any dynamic allocation should occur early during the setup phase of the sketch and sparingly or not at all later in the sketch. To help minimize SRAM consumption on Arduino and ensure libembroidery can be used in any way the sketch creator desires, it is required that any sketch using libembroidery must implement event handlers. See the ino-event source and header files for more information.

There is also an excellent article by Bill Earl on the Adafruit Learning System which covers these topics in more depth: `http://learn.adafruit.com/memories-of-an-arduino?view=all`.

Space

Since a stitch takes 3 bytes of storage and many patterns use more than 10k stitches, we can't assume that the pattern will fit in memory. Therefore we will need to buffer the current pattern on and off storage in small chunks. By the same reasoning, we can't load all of one struct beore looping so we will need functions similar to binaryReadInt16 for each struct.

This means the EmbArray approach won't work since we need to load each element and dynamic memory management is unnecessary because the arrays lie in storage.

TODO: Replace EmbArray functions with embPattern load functions.

Tables

All thread tables and large text blocks are too big to compile directly into the source code. Instead we can package the library with a data packet that is compiled from an assembly program in raw format so the specific padding can be controlled.

In the user section above we will make it clear that this file needs to be loaded on the pattern USB/SD card or the program won't function.

TODO: Start file with a list of offsets to data with a corresponding table to load into with macro constants for each label needed.

Current Pattern Memory Management

It will be simpler to make one file per EmbArray so we keep an EmbFile* and a length, so no malloc call is necessary. So there needs to be a consistent tmpfile naming scheme.

TODO: For each pattern generate a random string of hexadecimal and append it to the filenames like `stitchList_A16F.dat`. Need to check for a file which indicates that this string has been used already.

Special Notes

Due to historical reasons and to remain compatible with the Arduino 1.0 IDE, this folder must be called "utility". Refer to the arduino build process for more info: `https://arduino.github.io/arduino-cli/0.19/sketch-build-process/`.

libembroidery relies on the Arduino SD library for reading files. See the ino-file source and header files for more information.

6.1.9 1

.9. The Assembly Split

One problem to the problem of supporting both systems with abundant memory (such as a 2010s or later desktop) and with scarce memory (such as embedded systems) is that they don't share the same assembly language. To deal with this: there will be two equivalent software which are hand engineered to be similar but one will be in C and the other in the assembly dialects we support.

All assembly will be intended for embedded systems only, since a slightly smaller set of features will be supported. However, we will write a 'x86' version since that can be tested.

That way the work that has been done to simplify the C code can be applied to the assembly versions.

6.1.10 Build

To build the documentation run 'make'. This should run no problem on a normal Unix-like environment assuming pandoc is available.

- Pandoc creates the content of the page by converting the markdown to html.
- Pandoc also creates the printer-friendly documentation from the same markdown.
- Markdown acts as a go-between because it is easy to alter directly in the GH editor.

This way: 1. We write one set of documents for all projects. 2. The website can be simple and static, supporting machines that don't run javascript. 3. We control the styling of each version independently of our editing (Markdown) version 4. The printer-friendly documentation can have nicely rendered fonts and well placed figures.

6.1.11 1

.11. Features

1

.11.1. Bindings

Bindings for libembroidery are maintained for the languages we use internally in the project, for other languages we consider that the responsibility of other teams using the library.

So libembroidery is going to be supported on:

- C (by default)
- C++ (also by default)
- Java (for the Android application MobileViewer)
- Swift (for the iOS application iMobileViewer)

For C# we recommend directly calling the function directly using the DllImport feature:

```
[DllImport("libembroidery.so", EntryPoint="readCsv")]
```

see this StackOverflow discussion [for help](`https://stackoverflow.com/questions/11425202/is-it-possible-to-call-a-c-function-from-c-net`).

For Python you can do the same using [ctypes](`https://www.geeksforgeeks.org/how-to-call-a-c-fun`

6.1.12 1

.12. Threads
    * [DXF Color Table](link to dxf color table) * [HUS Color Table](link to hus color table) * [JEF Color Table](link to jef color table) * [PCM Color Table](link to pcm color table) * [PEC Color Table](link to pec color table)

DXF color table

HUS color table

JEF color table

PCM color table

PEC color table

6.1.13 Other Supported Thread Brands

The thread lists that aren't preprogrammed into formats but are indexed in the data file for the purpose of conversion or fitting to images/graphics.

- Arc Polyester
- Arc Rayon
- Coats and Clark Rayon
- Exquisite Polyester
- Fufu Polyester
- Fufu Rayon
- Hemingworth Polyester
- Isacord Polyester
- Isafil Rayon
- Marathon Polyester
- Marathon Rayon
- Madeira Polyester
- Madeira Rayon
- Metro Polyester
- Pantone
- Robison Anton Polyester
- Robison Anton Rayon
- Sigma Polyester
- Sulky Rayon
- ThreadArt Rayon

- ThreadArt Polyester
- ThreaDelight Polyester
- Z102 Isacord Polyester

6.1.14 Bibliography

1. Rudolf _Technical Info_ [http://www.achatina.de/sewing/main/TECHNICL.HTM
(Accessed 25 November 2021)
2. fineEmbStudio2021
3. Edutech format description (eduTechWikiDST)
4. KDE Liberty Description (libertyTajima)
5. The Sewing Witch _PCD2FMT_ [http://www.sewingwitch.com/sewing/bin/pcd2fm
6. [http://steve-jansen.github.io/guides/windows-batch-scripting/part-7-fun
7. [https://stackoverflow.com/questions/16286134/imagemagick-how-can-i-work
8. Fritzing [https://github.com/fritzing/fritzing-app](https://github.com/f
9. Sahoo, P., Wilkins, C., and Yeager, J., \Threshold selection using Renyi
10. [http://www.fmwconcepts.com/imagemagick/sahoothresh/index.php](http://
11. FINDME
12. G. van Rossum and B. Warsaw "Python PEP 7" [https://peps.python.org/pe
13. Google et al. "Google Java Style Guide" [https://google.github.io/styl

Chapter 7

Conclusions

.1 GNU Free Documentation License

Appendix A

Color Charts