DOCUMENT FOR THE CREATION OF A NEW TAMIL FONT

Explanation for the creation of any font:

* Any different kind of font can be easily added to the E paper display
* The font must be under the extension of .C type
* If the required font is not available in .C type and if the font is only available in TrueType font format (.ttf) then an online converter tool can be used to convert the true text font format to a .C format : <https://lvgl.io/tools/font_conv_v5_3>
* After the conversion of the .TTF file to .C file, The next step is to create a header file and add these two files in our project

E-paper display content:

* The E paper display comes with an example to display images and sample font
* This example was what was used to build other fonts
* In the Ee paper display folder there are four files namely e paper font.C , E paper font.H , e-paper-w29.C & e-paper-w29.H
* The E paper font.C is the file that contains the bitmap array of all the characters in a font
* The e paper font.H is the header file for the above mentioned .C file
* The e-paper-w29 .C is the file that as all the printing functions and displaying functions of the E paper display along with the functions for reading the font
* The e-paper-w29.H is the header file for the above mentioned .C file
* Now after creating the required font file and its required header file and after adding them to our project folder, and after making the necessary changes in the code, and in the C make file, these fonts can be used interchangeably at any time

Case Example:

(Creating a new font for Tamil)

Introduction:

* The requirement here was to create a new font for Tamil with the E paper display which was not available online. The way the print function works is by storing the required string, taking the first character and converting it into an ASCII value, then traversing through the entire bitmap array, by the value of the ASCII value multiplied by its height & weight. This is done because each bit or point in the bitmap array is printed as a black pixel on the display.
* In a font file every character must have the same height and the same width. This would seem easy for English because every character has the same with and height. And after printing a particular character at one X, Y coordinate, the cursor must cross the e paper display by the width of the previously printed character and print the next one. Now this again is easy for English as English always follows the forward direction.

Problem:

* The Tamil font is a bit tricky as the converted font from the .TTF file to the .C file has different widths for each character in the font file.
* Also the Tamil language has a certain type of glyphs that must be printed either before a character, or on top of a character, or after a character

Solution:

(Here the solution for creating a Tamil font for the E paper display will be detailed)

* The chosen Tamil .TTF font is suntommy .TTF file
* This was chosen because it follows the bamini encoding format
* The 1330 kurals in our project was also converted into a bamini encoding such that our font file would be the decoder for converting the encoded kurals and printing it in Tamil.
* After opening this link: <https://lvgl.io/tools/font_conv_v5_3> the required details are filled in the boxes after selecting the suntommy.ttf file.
* The required details are the beginning Unicode 32, the final Unicode 126, the maximum height of each character (15, 17 or 19) and a name for the file
* After clicking convert, the required.C file is created. Now this file is for the LVGL. We can create another header file for it by manipulating the example for the default English paper font.C file
* now some changes are to be done in the dot C file by commenting out the last portion of the file which contains the widths of each character and glyph
* After the required Tamil.C and header files are created the next step is to make sure the file is read correctly by the E paper-w29.C file
* since the string to be printed is in an encoded format there is no problem in reading what is meant to be printed
* The only problem that exists is the variable widths of each and every character
* This is done by manually adding an integer array of the widths of the 94 characters in the E-paper-W 29.C file
* This integer array is then used to determine how far the cursor must move after printing a certain character
* Finding out which character must be printed is the same as before: by finding out which ASCII value is available in the data the cursor is shifted by multiplying the ASCII value with its height and width
* Since the height is uniform and same for every character and the width is now available locally the cursor can traverse the required distance and print the character

Extended problems:

* Now after the issue of printing the encoded Tamil in the paper display the remaining issue is printing the glyphs for Tamil
* Now some characters are to be printed before a character, above a character, after a character, or added with the previous character

G-1: ): C:\Users\Anish\AppData\Local\Microsoft\Windows\INetCache\Content.Word\1.png

* For example: ip ich ith id, etc. The way this is printed is by traversing half of the width of the printed character and then printing the dot and then traversing the remaining with of the previous character. Since the E paper display only overwrites the black pixels, they are not erased by the white spaces of the other pixels

G-2:



* The next issue is printing the E and EE
* The source file used to convert from .TTF to .C did not contain the bitmap array character for e and ee. Thus these two characters were manually drawn in paint and then the image was converted to a bitmap array using an online converter tool: [image2cpp (javl.github.io)](https://javl.github.io/image2cpp/)
* The e and ee were added in the main .C file tamil font file by commenting out the bitmap array information for “p“ and “P”. After this is done their widths are also to be edited in the integer array of widths
* After checking for each character if the drawn image works, this issue is fixed

G-3:



* The next issue is printing the UUU glyph.
* This issue is split into two as in the Tamil language the UUU glyph is printed in two ways
* The suuu is available in the bitmap array as # thus the encoder for suuu is R+ and this can be replaced with # in our data set of kurals
* The yuuu, vuuu & puuu are drawn manually in paint and replaced in the font file in place of “(“ and “)” and “5” respectively.
* This was done because our project (thirukkurals) do not use any brackets or numbers while printing the tamil.

G-4:

* The second type of uuu glyph is for ruuu, thuuu, nyuuu, etc. these work automatically for bigger font sizes such as 17 and 19 but do not work very well with smaller font sizes such as 15.
* This for smaller font sizes these glyphs must be manually either left shifted or right shifted by 1 to 2 pixels each for each character

REFERENCES:

1. **Online Font Converter V 5.3**  
   Convert TTF and WOFF fonts to C array

<https://lvgl.io/tools/font_conv_v5_3>

1. tool to change images into byte arrays:

[image2cpp (javl.github.io)](https://javl.github.io/image2cpp/)

1. Bamini to Unicode (& vice versa):

[Bamini to Unicode - UC - Tamil Unicode Converter (tamillexicon.com)](https://www.tamillexicon.com/uc/bamini-unicode)

1. ASCII Table of values (FOR REFERENCE):

[ASCII Table (cmu.edu)](https://www.cs.cmu.edu/~pattis/15-1XX/common/handouts/ascii.html)

1. Unicode Character Ranges (FOR REFERENCE):

[Unicode Character Ranges (jrgraphix.net)](https://jrgraphix.net/research/unicode.php)

(This reference can be used to get Unicode ranges of all languages. For tamil it is OB80 – 0BFF for fonts that use Unicode encoding. Our suntommy font uses Bamini encoding and thus the Unicode range is only 32 – 126)

1. Hexadecimal to Decimal Converter

[0B80 hex to decimal conversion (rapidtables.com)](https://www.rapidtables.com/convert/number/hex-to-decimal.html?x=0B80)