

1. First, find the encoding range of the character you want to display in unicode

<https://www.unicode.org/charts/>

For example

Basic Latin(ASCII) is 0x00~0x7F <https://www.unicode.org/charts/PDF/U0000.pdf>

Cyrillic(Russia) is 0x400~0x4FF <https://www.unicode.org/charts/PDF/U0400.pdf>

Armenian is 0x530~0x58F <https://www.unicode.org/charts/PDF/U0530.pdf>

Latin extended (Czech/French/etc...)

2. Let's take Russia as an example. You need to create a dot matrix font with all characters in Russia. The height of the font must be an integral multiple of 8, and the scanning method of font requires first from top to bottom, then from left to right, and the high position is first. The default size of the font is high*width: 24 * 12(TFT35 / TFT43 / TFT50 / TFT70), 16 * 8(TFT24 / TFT28), You can find the actual resolution of the screen in the header file corresponding to the screen. At present, there are four resolutions: 320 x 240 (TFT24 / TFT28), 480 x 272 (TFT 43 / TFT 50), 480 x 320 (TFT 35), 800 x 480 (TFT70), and then modify the font size in the resolution header file. For example, we use TFT35 to find the 480 x 320 header file to modify the font size.

```
EXPLORER
├── BIGTREETOUCHSCREEN
│   ├── SpeedControl.h
│   ├── Temperature.c
│   ├── Temperature.h
│   ├── Touch_Encoder.c
│   ├── Touch_Encoder.h
│   ├── Fatfs
│   ├── Hal
│   ├── Menu
│   ├── Variants
│   └── Resolution
│       ├── TFT_320X240.h
│       ├── TFT_480X272.h
│       └── TFT_480X320.h
├── pin_MKS_TFT28_NEW_GENIUS.h
├── pin_MKS_TFT28_V3_0.h
├── pin_MKS_TFT28_V4_0.h
├── pin_MKS_TFT32_V1_3.h
├── pin_MKS_TFT32_V1_4.h
├── pin_Template.h
├── pin_TFT24_V1_1.h
├── pin_TFT28_V1_0.h
├── pin_TFT28_V3_0.h
└── pin_TFT35_B1_V3_0.h

main.c  variants.h  TFT_480X320.h X  pin_TFT35_V3_0.h
TFT > src > User > Variants > Resolution > C TFT_480X320.h > BYTE_WIDTH
Gurmeet Athwal, 4 months ago | 5 authors (Msq001 and others)
1  #ifndef _TFT_480_320_H_
2  #define _TFT_480_320_H_
3
4  #define LCD_WIDTH  480
5  #define LCD_HEIGHT 320
6
7  #define SPACE_Y    ((LCD_HEIGHT - ICON_START_Y - ICON_H
8
9  #ifndef BYTE_HEIGHT
10 #define BYTE_HEIGHT 24
11 #endif
12 #ifndef BYTE_WIDTH
13 #define BYTE_WIDTH  (BYTE_HEIGHT/2)
14 #endif
15
16 #ifndef LARGE_BYTE_HEIGHT
17 #define LARGE_BYTE_HEIGHT 32
18 #endif
19 #ifndef LARGE_BYTE_WIDTH
20 #define LARGE_BYTE_WIDTH  18
21 #endif
22
23 #define ICON_WIDTH  95
24 #define ICON_HEIGHT 95
25 #define TITLE_END_Y 40
26 #define ICON_START_Y (TITLE_END_Y+10)
27
```

3. Set the dot matrix font in boot.h to store the starting address in SPI Flash (note the total size of the font file, do not overlap with other font addresses, and the total capacity of Flash is 8MByte, the ending address is 0x800000)

```

16 #define WORD_UNICODE_SIZE 0x480000
17 #define BYTE_ASCII_SIZE 0x1000
18 #define LARGE_FONT_SIZE 0x3000
19 #define _8X16_FONT_SIZE 0x1000
20 #define FLASH_SIGN_SIZE 0x1000 // store status of last font/icon/config update
21 #define LANGUAGE_SIZE 0x15000 // Language pack size
22 #define STRINGS_STORE_MAX_SIZE 0x1000 // label strings max size
23 #define PREHEAT_STORE_MAX_SIZE 0x1000 // preheat setting max size
24 #define PRINT_GCODES_MAX_SIZE 0x5000 // start/end/cancel gcodes max size
25 #define CUSTOM_GCODE_MAX_SIZE 0x5000 // custom gcodes max size
26 #define ICON_MAX_SIZE 0x5000
27 #define INFOBOX_MAX_SIZE 0x8000
28 #define SMALL_ICON_MAX_SIZE 0x2000
29 #endif
30
31 // address in spiFlash W25Qxx
32 #define LOGO_ADDR 0x0
33 #define WORD_UNICODE LOGO_MAX_SIZE // unicode (+0x480000 4.5M)
34 #define BYTE_ASCII_ADDR (WORD_UNICODE + WORD_UNICODE_SIZE) // ascii (+0x1000 4K)
35 #define LARGE_FONT_ADDR (BYTE_ASCII_ADDR + BYTE_ASCII_SIZE) // Large ascii font
36 #define _8X16_FONT_ADDR (LARGE_FONT_ADDR + LARGE_FONT_SIZE) // 8 x 16 ascii font
37 #define RUSSIA_FONT_ADDR (_8X16_FONT_ADDR + _8X16_FONT_SIZE) // Russia font
38 // #define BYTE_RESERVE_ADDR 0x710000
39 #define FLASH_SIGN_ADDR (RUSSIA_FONT_ADDR + 0x40000) // for language label strings from language file
40 #define LANGUAGE_ADDR (FLASH_SIGN_ADDR + FLASH_SIGN_SIZE) // for label strings from config file
41 #define STRINGS_STORE_ADDR (LANGUAGE_ADDR + LANGUAGE_SIZE) // for label strings from config file
42 #define PREHEAT_STORE_ADDR (STRINGS_STORE_ADDR + STRINGS_STORE_MAX_SIZE) // for preheat settings from config file
43 #define PRINT_GCODES_ADDR (PREHEAT_STORE_ADDR + PREHEAT_STORE_MAX_SIZE) // for start/end/cancel gcodes from config file
44 #define CUSTOM_GCODE_ADDR (PRINT_GCODES_ADDR + PRINT_GCODES_MAX_SIZE) // for custom gcodes from config file
45
46 #define ICON_ADDR(num) ((num) * ICON_MAX_SIZE + CUSTOM_GCODE_ADDR + CUSTOM_GCODE_MAX_SIZE)
47 #define INFOBOX_ADDR (ICON_ADDR(ICON_PREVIEW) + ICON_MAX_SIZE) // total byte size 0xA7F8
48 #define SMALL_ICON_START_ADDR (INFOBOX_ADDR + INFOBOX_MAX_SIZE)
49 #define SMALL_ICON_ADDR(num) ((num) * SMALL_ICON_MAX_SIZE + SMALL_ICON_START_ADDR)
50 #define FLASH_USED (INFOBOX_ADDR + INFOBOX_MAX_SIZE) // currently small icons are not used

```

4. Add the ability to update fonts to SPI Flash in boot.c

```

351 }
352
353 void scanUpdates(void)
354 {
355     if (mountSDCard())
356     {
357         bool flash_sign_updated = false;
358         uint32_t saved_flash_sign[sign_count];
359         W25Qxx_ReadBuffer((uint8_t*)&saved_flash_sign, FLASH_SIGN_ADDR, sizeof(saved_flash_sign));
360
361         if (f_dir_exists(FONT_ROOT_DIR))
362         {
363             if (updateFont(FONT_ROOT_DIR "/byte_ascii.fon", BYTE_ASCII_ADDR) &&
364                 updateFont(FONT_ROOT_DIR "/word_unicode.fon", WORD_UNICODE) &&
365                 updateFont(FONT_ROOT_DIR "/large_byte_ascii.fon", LARGE_FONT_ADDR) &&
366                 updateFont(FONT_ROOT_DIR "/8x16_byte_ascii.fon", _8X16_FONT_ADDR) &&
367                 updateFont(FONT_ROOT_DIR "/russia.fon", RUSSIA_FONT_ADDR) &&
368                 (saved_flash_sign[font_sign] != FONT_CHECK_SIGN))
369             {
370                 saved_flash_sign[font_sign] = FONT_CHECK_SIGN;
371                 flash_sign_updated = true;
372             }
373         }
374     }

```

5. In the static FONT_BITMAP font[] array of the utf8_decode.c file, add the character encoding to be parsed. The information to be added is as follows

```

EXPLORER
PIO Home C boot.h C boot.c C utf8_decode.c C utf8_decode.h x
OPEN EDITORS 1 UNSAVED
BIGTREETOUCHSCREEN
include
TFT
src
Libraries
User
API
Gode
Language
C language_am.h
C language_cn.h
C language_cz.h
C language_de.h
C language_en.h
C language_es.h
C language_fr.h
C language_jp.h
C language_ru.h
C language_c
C language_h
C utf8_decode.c
C utf8_decode.h
UI
Vfs
boot.c
boot.h
coordinate.c
coordinate.h
extend.c

TFT > src > User > API > Language > C utf8_decode.h > _unnamed_struct_02c6_1 > pixelWidth
1 #ifndef _UTF8_DECODE_H_
2 #define _UTF8_DECODE_H_
3
4 #include "stdint.h"
5
6 typedef struct {
7     uint32_t startCodePoint; // start unicode code point for language 0x00
8     uint32_t endCodePoint; // end unicode code point for language 0x7F
9     uint8_t pixelHeight; // font display pixel height 24/16
10    uint8_t pixelWidth; // font display pixel width 12/8
11    uint32_t bitMapStartAddr; // dot matrix font library start address in w25qxx BYTE_ASCII_ADDR
12    uint8_t bitMapHeight; // dot matrix font library pixel height 24/16
13    uint8_t bitMapWidth; // dot matrix font library pixel width 12/8
14    uint32_t bitMapStartCodePoint; // the first character code point in this font bitmap file 0x00
15 }FONT_BITMAP;
16
17 typedef struct
18 {
19     // encode info
20     uint8_t bytes; // Number of bytes occupied by one character
21     uint32_t codePoint; // Actual encoding index of characters
22     // font info
23     uint8_t pixelHeight; // The pixel height of a character display
24     uint8_t pixelWidth; // The pixel width of a character display
25     uint32_t bitMapAddr; // the address of font bitmap in w25qxx
26 }CHAR_INFO;
27
28
29 void getCharacterInfo(const uint8_t *ch, CHAR_INFO *pinfo);
30 uint16_t GUI_strPixelWidth(const uint8_t *const str);
31
32 #endif
33

```

```

EXPLORER
PIO Home C boot.h C boot.c C utf8_decode.c C utf8_decode.h
OPEN EDITORS 1 UNSAVED
BIGTREETOUCHSCREEN
Copy to SD Card root directory to update - U...
include
TFT
src
Libraries
User
API
Gode
Language
C language_am.h
C language_cn.h
C language_cz.h
C language_de.h
C language_en.h
C language_es.h
C language_fr.h
C language_jp.h
C language_ru.h
C Language.c
C Language.h
C utf8_decode.c
C utf8_decode.h
UI
Vfs
boot.c
boot.h
coordinate.c
coordinate.h

TFT > src > User > API > Language > C utf8_decode.c > font
1 #include "utf8_decode.h"
2 #include "includes.h"
3
4
5 static FONT_BITMAP font[] = {
6     // Visible ASCII code, from ' ' to '~'
7     // start unicode code point for language
8     0x00, // 0x00 means the first control character 'NULL'
9     // end unicode code point for language
10    0x7F, // 0x7F means the last control character 'DEL'
11    // font display pixel height
12    24,
13    // font display pixel width
14    12,
15    // dot matrix font library start address in w25qxx
16    BYTE_ASCII_ADDR,
17    // dot matrix font library pixel height
18    24,
19    // dot matrix font library pixel width
20    12,
21    // the first character code point in this font bitmap file
22    0x00, // the first character in BYTE_ASCII_ADDR is 0x00('NULL')
23 },
24 { // Czech(Latin 1 Supplement, Extended-A&B)
25     0x80,
26     0x24F,
27     BYTE_HEIGHT,
28     BYTE_WIDTH,
29     WORD_UNICODE,
30     BYTE_HEIGHT,
31     BYTE_WIDTH * 2, // default "word_unicode.fon" dot matrix library font size is 24*24 / 16*16
32     0x0, // the first character in WORD_UNICODE is 0x0000
33 },

```

6. Compile, generate and update new firmware, change the name of the font file to the name "russia.fon" set in the firmware boot.c, put it in the "TFT35 (TFT28, TFT24)/font" folder of the SD card, and then put the SD Insert the card into the card slot of the touch screen, reset the font file, then switch to the language you want in the settings to use your customized font.

