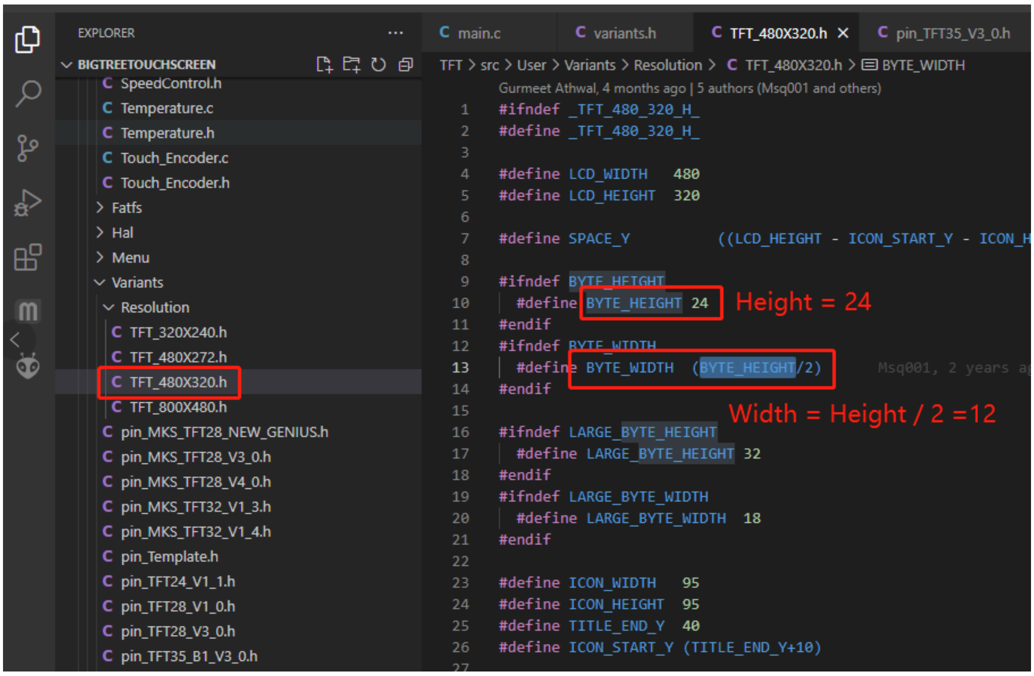
1. 首先请找到你所要显示的字符在 unicode 中的编码范围 https://www.unicode.org/charts/ 例如:

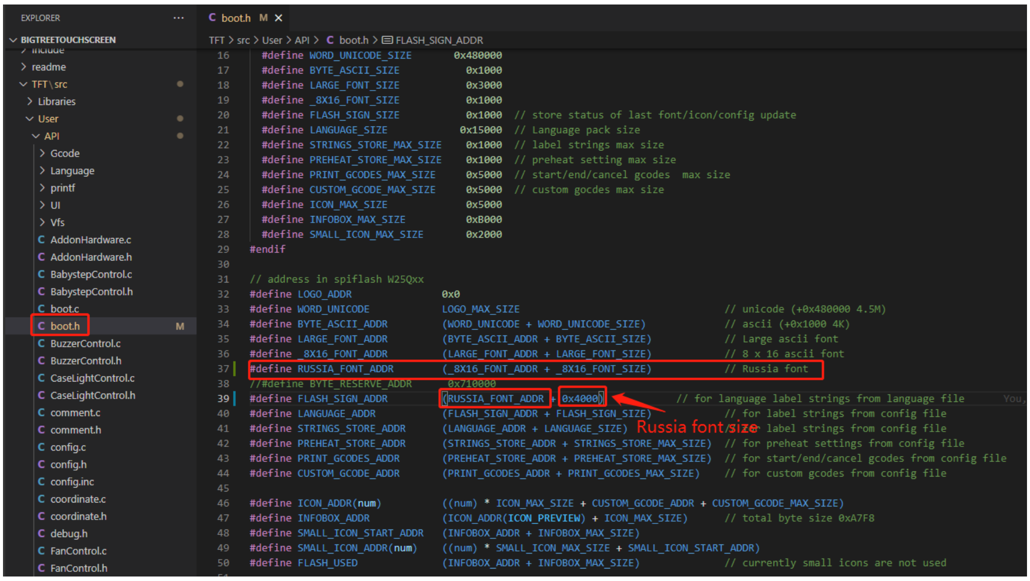
   Basic Latin(ASCII) 是 0x00~0x7F https://www.unicode.org/charts/PDF/U0000.pdf
   Cyrillic(Russia) 是 0x400~0x4FF https://www.unicode.org/charts/PDF/U0400.pdf
   Armenian 是 0x530~0x58F https://www.unicode.org/charts/PDF/U0530.pdf
   Latin extended (Czech/French/etc...)

2. 我们以 Russia 为例，你需要先生成一个 Russia 所有字符的点阵字体，字体的高度必须为 8 的整数倍，扫描方式要求先从上到下，再从左到右，高位在前。字体的默认尺寸高*宽：24 * 12(TFT35 / TFT43 / TFT50 / TFT70)、16 * 8(TFT24 / TFT28)，可以在屏幕对应的头文件中找到屏幕实际的分辨率，目前有四种分辨率：320 x 240(TFT24 / TFT28)、480 x 272(TFT43/TFT50)、480 x 320(TFT35)、800 x 480(TFT70)，然后在分辨率头文件中修改字体的尺寸，例如我们用的是 TFT35，找到 480 x 320 的头文件修改字体的尺寸。



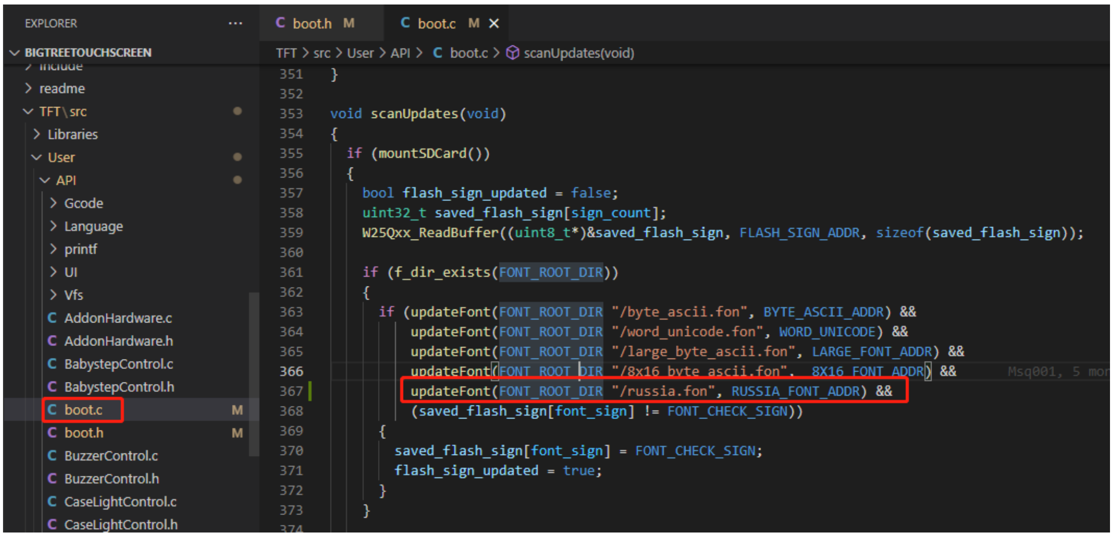
```c
#ifndef _TFT_480_320_H_
#define _TFT_480_320_H_

#define LCD_WIDTH   480
#define LCD_HEIGHT  320

#define SPACE_Y          ((LCD_HEIGHT - ICON_START_Y - ICON_H

#ifndef BYTE_HEIGHT
  #define BYTE_HEIGHT 24
#endif
#ifndef BYTE_WIDTH
  #define BYTE_WIDTH   (BYTE_HEIGHT/2)
#endif

#ifndef LARGE_BYTE_HEIGHT
  #define LARGE_BYTE_HEIGHT 32
#endif
#ifndef LARGE_BYTE_WIDTH
  #define LARGE_BYTE_WIDTH  18
#endif

#define ICON_WIDTH   95
#define ICON_HEIGHT  95
#define TITLE_END_Y  40
#define ICON_START_Y (TITLE_END_Y+10)
```

3. 在 boot.h 中新增点阵字体存放在 SPI Flash 中的起始地址(注意字体文件的总大小，不要跟其他的字体地址有重叠，且 Flash 的总容量为 8Mbyte，结束地址为 0x800000)



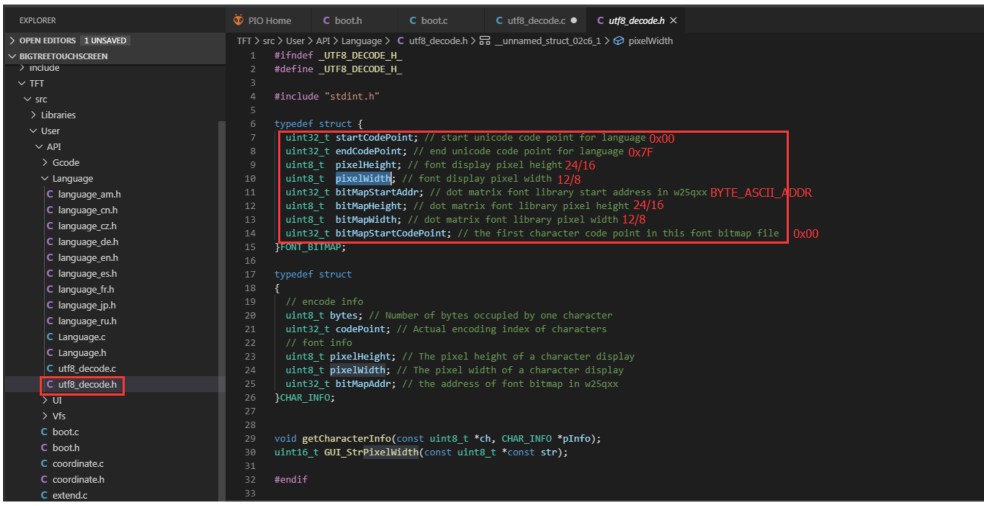
```c
  #define WORD_UNICODE_SIZE       0x480000
  #define BYTE_ASCII_SIZE           0x1000
  #define LARGE_FONT_SIZE           0x3000
  #define _8X16_FONT_SIZE           0x1000
  #define FLASH_SIGN_SIZE           0x1000  // store status of last font/icon/config update
  #define LANGUAGE_SIZE            0x15000  // Language pack size
  #define STRINGS_STORE_MAX_SIZE    0x1000  // label strings max size
  #define PREHEAT_STORE_MAX_SIZE    0x1000  // preheat setting max size
  #define PRINT_GCODES_MAX_SIZE     0x5000  // start/end/cancel gcodes  max size
  #define CUSTOM_GCODE_MAX_SIZE     0x5000  // custom gocdes max size
  #define ICON_MAX_SIZE             0x5000
  #define INFOBOX_MAX_SIZE          0xB000
  #define SMALL_ICON_MAX_SIZE       0x2000
#endif

// address in spiflash W25Qxx
#define LOGO_ADDR               0x0
#define WORD_UNICODE            LOGO_MAX_SIZE                                // unicode (+0x480000 4.5M)
#define BYTE_ASCII_ADDR         (WORD_UNICODE + WORD_UNICODE_SIZE)           // ascii (+0x1000 4K)
#define LARGE_FONT_ADDR         (BYTE_ASCII_ADDR + BYTE_ASCII_SIZE)          // Large ascii font
#define _8X16_FONT_ADDR         (LARGE_FONT_ADDR + LARGE_FONT_SIZE)          // 8 x 16 ascii font
#define RUSSIA_FONT_ADDR        (_8X16_FONT_ADDR + _8X16_FONT_SIZE)          // Russia font
//#define BYTE_RESERVE_ADDR      0x710000
#define FLASH_SIGN_ADDR         (RUSSIA_FONT_ADDR + 0x4000)          // for language label strings from language file
#define LANGUAGE_ADDR           (FLASH_SIGN_ADDR + FLASH_SIGN_SIZE)          // for label strings from config file
#define STRINGS_STORE_ADDR      (LANGUAGE_ADDR + LANGUAGE_SIZE)              // for label strings from config file
#define PREHEAT_STORE_ADDR      (STRINGS_STORE_ADDR + STRINGS_STORE_MAX_SIZE)  // for preheat settings from config file
#define PRINT_GCODES_ADDR       (PREHEAT_STORE_ADDR + PREHEAT_STORE_MAX_SIZE)  // for start/end/cancel gcodes from config file
#define CUSTOM_GCODE_ADDR       (PRINT_GCODES_ADDR + PRINT_GCODES_MAX_SIZE)    // for custom gcodes from config file

#define ICON_ADDR(num)          ((num) * ICON_MAX_SIZE + CUSTOM_GCODE_ADDR + CUSTOM_GCODE_MAX_SIZE)
#define INFOBOX_ADDR            (ICON_ADDR(ICON_PREVIEW) + ICON_MAX_SIZE)      // total byte size 0xA7F8
#define SMALL_ICON_START_ADDR   (INFOBOX_ADDR + INFOBOX_MAX_SIZE)
#define SMALL_ICON_ADDR(num)    ((num) * SMALL_ICON_MAX_SIZE + SMALL_ICON_START_ADDR)
#define FLASH_USED              (INFOBOX_ADDR + INFOBOX_MAX_SIZE)            // currently small icons are not used
```

4. 在 boot.c 中添加将字体更新到 SPI Flash 中的功能

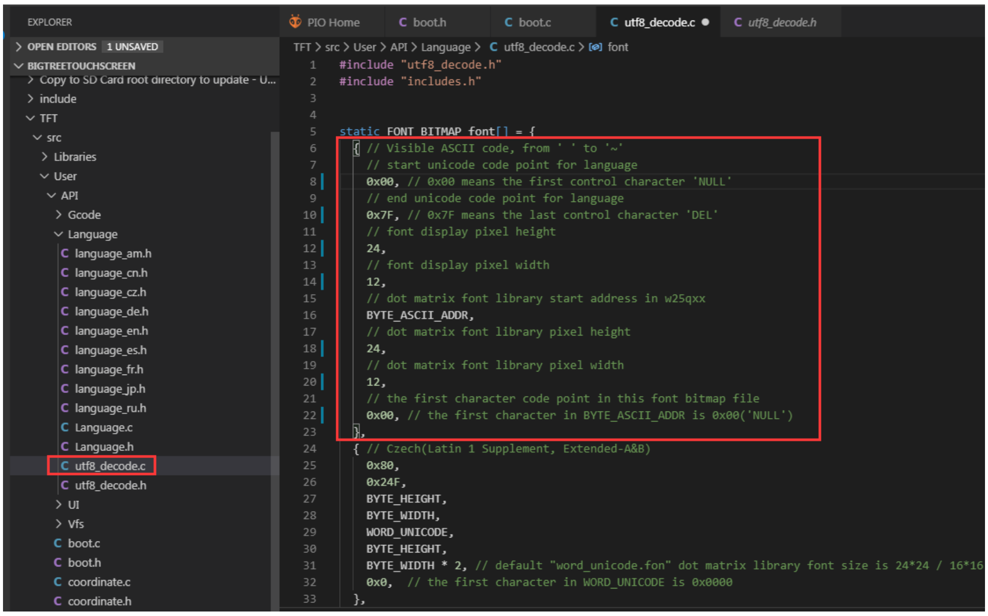


```c
}

void scanUpdates(void)
{
  if (mountSDCard())
  {
    bool flash_sign_updated = false;
    uint32_t saved_flash_sign[sign_count];
    W25Qxx_ReadBuffer((uint8_t*)&saved_flash_sign, FLASH_SIGN_ADDR, sizeof(saved_flash_sign));

    if (f_dir_exists(FONT_ROOT_DIR))
    {
      if (updateFont(FONT_ROOT_DIR "/byte_ascii.fon", BYTE_ASCII_ADDR) &&
          updateFont(FONT_ROOT_DIR "/word_unicode.fon", WORD_UNICODE) &&
          updateFont(FONT_ROOT_DIR "/large_byte_ascii.fon", LARGE_FONT_ADDR) &&
          updateFont(FONT_ROOT_DIR "/8x16_byte_ascii.fon", _8X16_FONT_ADDR) &&
          updateFont(FONT_ROOT_DIR "/russia.fon", RUSSIA_FONT_ADDR) &&
          (saved_flash_sign[font_sign] != FONT_CHECK_SIGN))
      {
        saved_flash_sign[font_sign] = FONT_CHECK_SIGN;
        flash_sign_updated = true;
      }
    }
```

5. 在 utf8_decode.c 文件的 static FONT_BITMAP font[] 数组中，添加待解析的字符编码，需要添加的信息如下

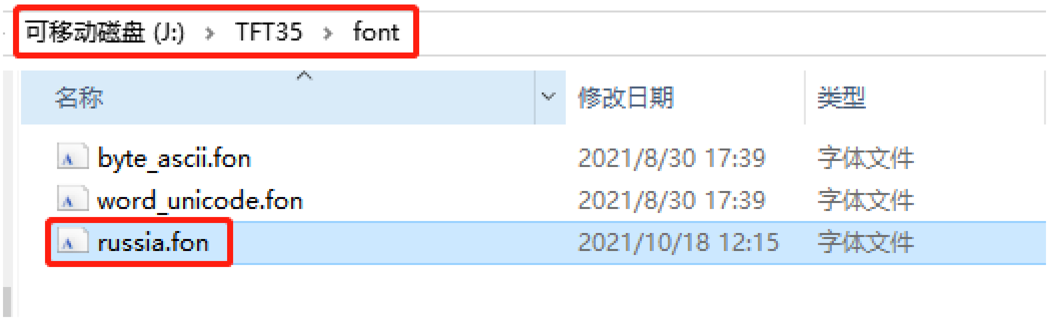


```c
#ifndef _UTF8_DECODE_H_
#define _UTF8_DECODE_H_

#include "stdint.h"

typedef struct {
  uint32_t startCodePoint; // start unicode code point for language 0x00
  uint32_t endCodePoint; // end unicode code point for language 0x7F
  uint8_t  pixelHeight; // font display pixel height 24/16
  uint8_t  pixelWidth; // font display pixel width 12/8
  uint32_t bitMapStartAddr; // dot matrix font library start address in w25qxx BYTE_ASCII_ADDR
  uint8_t  bitMapHeight; // dot matrix font library pixel height 24/16
  uint8_t  bitMapWidth; // dot matrix font library pixel width 12/8
  uint32_t bitMapStartCodePoint; // the first character code point in this font bitmap file  0x00
}FONT_BITMAP;

typedef struct
{
  // encode info
  uint8_t bytes; // Number of bytes occupied by one character
  uint32_t codePoint; // Actual encoding index of characters
  // font info
  uint8_t pixelHeight; // The pixel height of a character display
  uint8_t pixelWidth; // The pixel width of a character display
  uint32_t bitMapAddr; // the address of font bitmap in w25qxx
}CHAR_INFO;


void getCharacterInfo(const uint8_t *ch, CHAR_INFO *pInfo);
uint16_t GUI_StrPixelWidth(const uint8_t *const str);

#endif
```



```c
#include "utf8_decode.h"
#include "includes.h"


static FONT_BITMAP font[] = {
  { // Visible ASCII code, from ' ' to '~'
    // start unicode code point for language
    0x00, // 0x00 means the first control character 'NULL'
    // end unicode code point for language
    0x7F, // 0x7F means the last control character 'DEL'
    // font display pixel height
    24,
    // font display pixel width
    12,
    // dot matrix font library start address in w25qxx
    BYTE_ASCII_ADDR,
    // dot matrix font library pixel height
    24,
    // dot matrix font library pixel width
    12,
    // the first character code point in this font bitmap file
    0x00, // the first character in BYTE_ASCII_ADDR is 0x00('NULL')
  },
  { // Czech(Latin 1 Supplement, Extended-A&B)
    0x80,
    0x24F,
    BYTE_HEIGHT,
    BYTE_WIDTH,
    WORD_UNICODE,
    BYTE_HEIGHT,
    BYTE_WIDTH * 2, // default "word_unicode.fon" dot matrix library font size is 24*24 / 16*16
    0x0,  // the first character in WORD_UNICODE is 0x0000
  },
```

6. 编译生成并更新新的固件，将字体文件的名称修改为固件 boot.c 中设置的名称 “russia.fon”, 放入 SD 卡的 ”TFT35(TFT28、TFT24)/font”文件夹中，将 SD 卡插入触摸屏的卡槽，复位更新字体文件, 更新完成后在设置中切换到你所使用的语言即可享用你自定义的字体。



可移动磁盘 (J:) › TFT35 › font

名称	修改日期	类型
byte_ascii.fon	2021/8/30 17:39	字体文件
word_unicode.fon	2021/8/30 17:39	字体文件
russia.fon	2021/10/18 12:15	字体文件