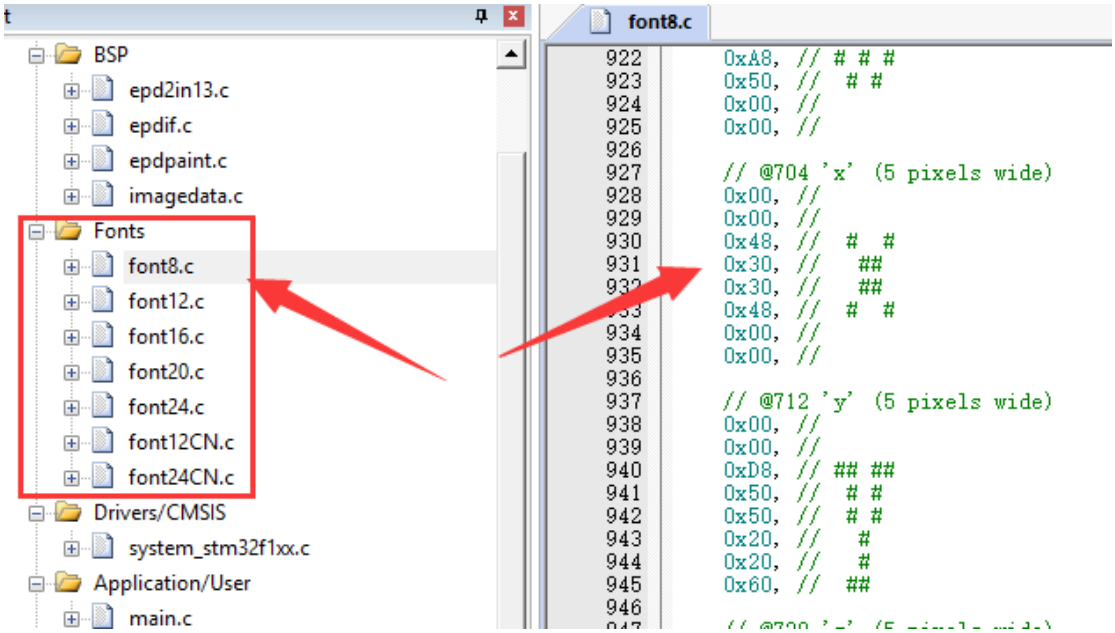
e-Paper 显示中文字符

微雪电子微雪电子墨水屏自推出以来经常有客户问到如何显示中文，今天特意写一个教程展示如果显示中文。字符显示显示实际上也是图片的显示，一个字符就是一张很小的图片，字符串显示，就是有一个个字符的图片拼接在一起显示一个字符。

英文字符显示原理

废话少说，在显示中文之前让我们来了解一下 ASCII 码是如何显示的。下面以 2.13inch e-Paper HAT 的 STM32 程序为例讲解。

要显示字符首先要有字体，示例程序中 Fonts 目录下的文件就是对应不同的字体，打开文件可以到看一堆数据。



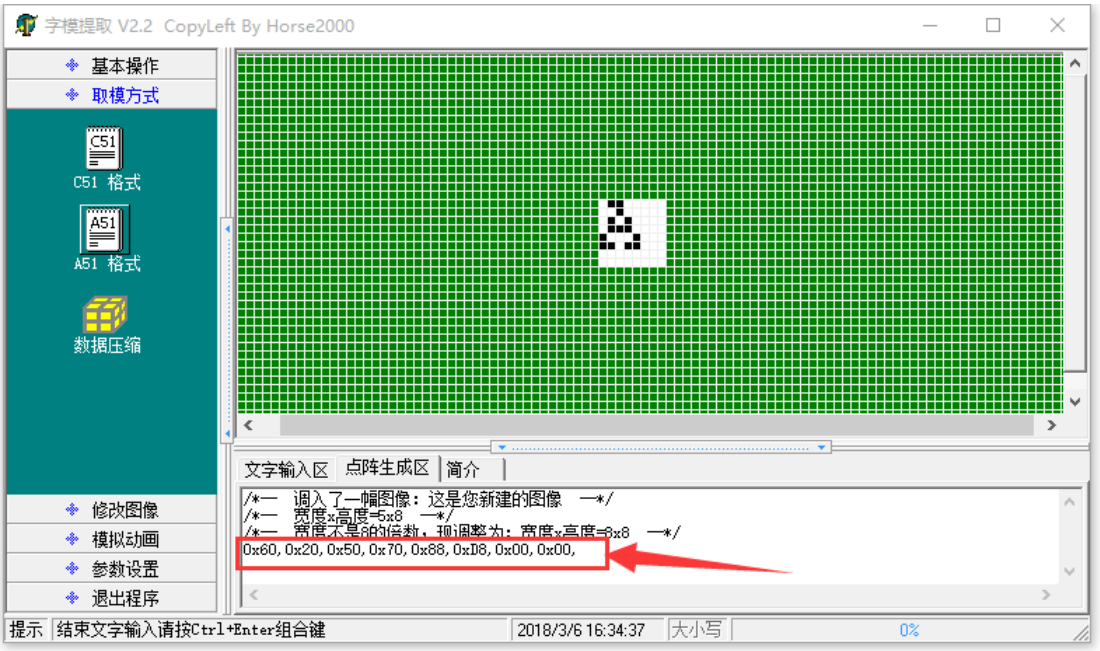
每种字体都有一个结构体分别存储字体的信息。结构体包括数组指针，字体宽度，字体高度。

```
typedef struct _tFont
{
  const uint8_t *table;
  uint16_t Width;
  uint16_t Height;

} sFONT;

extern sFONT Font24;
extern sFONT Font20;
extern sFONT Font16;
extern sFONT Font12;
extern sFONT Font8;
```

```
sFONT Font8 = {
  Font8_Table,
  5, /* Width */
  8, /* Height */
};
```

以上字体是在 stm32 官方的示例程序复制过来的。是 ASCII 的字体，下来我们来讲解一下我们怎么制作字体。下面图片是 Font8 字体"A"字符的字模，我们可以用字模软件来得到 A 字符的数据。

```
// @264 'A' (5 pixels wide)
0x60, //  ##
0x20, //   #
0x50, //  # #
0x70, //  ###
0x88, // #   #
0xD8, // ## ##
0x00, //
0x00, //
```



字体数据就是用字模提取软件，将字模按照横向取模或者纵向取模，将逐个像素点用数组表示。要显示“A”字符就找到“A”字符的数据然后将字模逐点显示出来就行了

```c
/**
 *  @brief: this draws a charactor on the frame buffer but not refresh
 */
void Paint_DrawCharAt(Paint* paint, int x, int y, char ascii_char, sFONT* font, int colored) {
    int i, j;
    unsigned int char_offset = (ascii_char - ' ') * font->Height * (font->Width / 8 + (font->Width % 8 ? 1 : 0));
    const unsigned char* ptr = &font->table[char_offset];

    for (j = 0; j < font->Height; j++) {
        for (i = 0; i < font->Width; i++) {
            if (*ptr & (0x80 >> (i % 8))) {
                Paint_DrawPixel(paint, x + i, y + j, colored);
            }
            if (i % 8 == 7) {
                ptr++;
            }
        }
        if (font->Width % 8 != 0) {
            ptr++;
        }
    }
}
```

这里需要注意一点就是红框这里，字体数组是按照 ASCII 的顺序存储的，第一个字符就是空格 " ",而每个字符的数据大小是相同的。所以将 A 的 ASCII 码减去空格键的 ASCII 码，这样就可以找到字符"A"的数据开始位置。

```c
/**
 *  @brief: this displays a string on the frame buffer but not refresh
 */
void Paint_DrawStringAt(Paint* paint, int x, int y, const char* text, sFONT* font, int colored) {
    const char* p_text = text;
    unsigned int counter = 0;
    int refcolumn = x;

    /* Send the string character by character on EPD */
    while (*p_text != 0) {
        /* Display one character on EPD */
        Paint_DrawCharAt(paint, refcolumn, y, *p_text, font, colored);
        /* Decrement the column position by 16 */
        refcolumn += font->Width;
        /* Point on the next character */
        p_text++;
        counter++;
    }
}
```

字符串显示就是将一个个字符显示出来。

字符集

好了，我们已经大概知道怎么显示英文字符了。在显示中文之前还需要了解字符集。什么是字符集呢？字符集就是所有字符的集合，ASCII 码就是一个字符集，ASCII 只有 0~127 个字符。用一个字节表示。只能显示英文，不能显示中文。

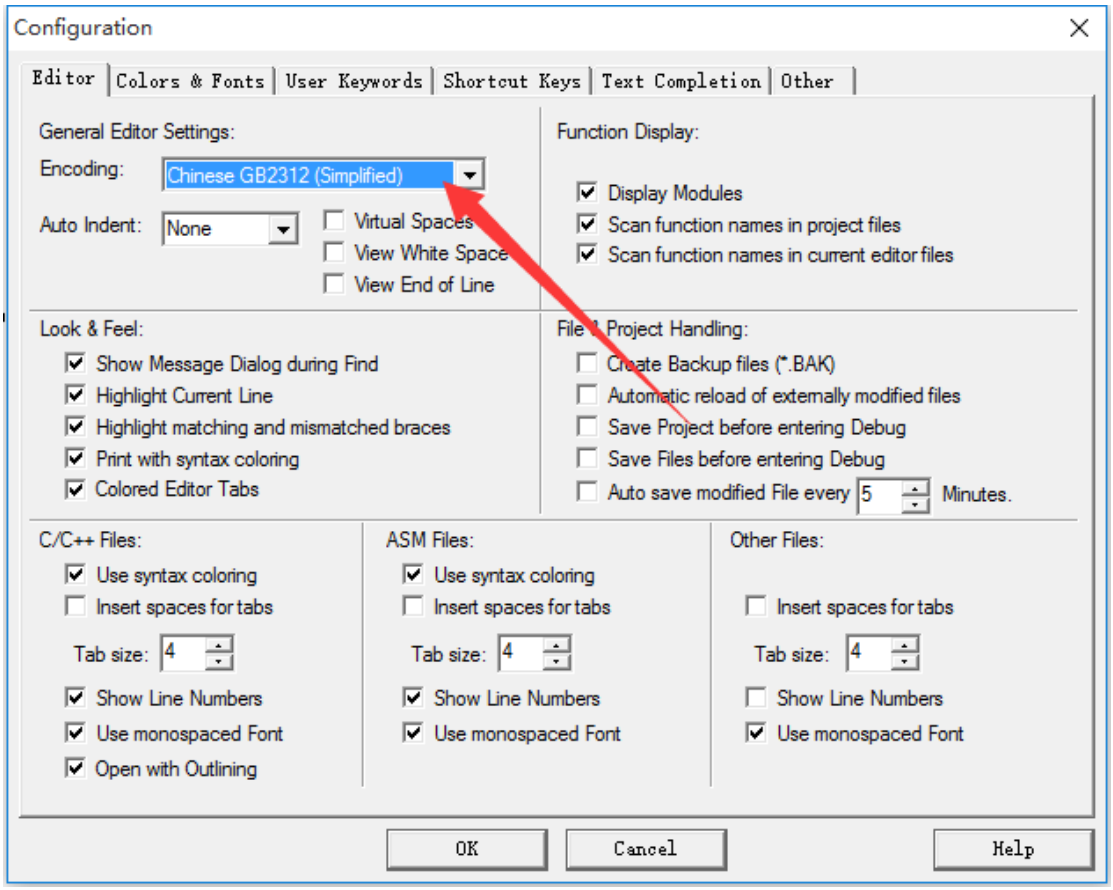
ASCII表

(American Standard Code for Information Interchange 美国标准信息交换代码)

高四位 / 低四位		ASCII控制字符												ASCII打印字符											
		0000						0001						0010		0011		0100		0101		0110		0111	
		0						1						2		3		4		5		6		7	
		十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	Ctrl
0000	0	0		^@	NUL	\0	空字符	16	▶	^P	DLE		数据链路转义	32		48	0	64	@	80	P	96	`	112	p	
0001	1	1	☺	^A	SOH		标题开始	17	◀	^Q	DC1		设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q	
0010	2	2	☻	^B	STX		正文开始	18	↕	^R	DC2		设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r	
0011	3	3	♥	^C	ETX		正文结束	19	‼	^S	DC3		设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s	
0100	4	4	♦	^D	EOT		传输结束	20	¶	^T	DC4		设备控制 4	36	$	52	4	68	D	84	T	100	d	116	t	
0101	5	5	♣	^E	ENQ		查询	21	§	^U	NAK		否定应答	37	%	53	5	69	E	85	U	101	e	117	u	
0110	6	6	♠	^F	ACK		肯定应答	22	▬	^V	SYN		同步空闲	38	&	54	6	70	F	86	V	102	f	118	v	
0111	7	7	•	^G	BEL	\a	响铃	23	↨	^W	ETB		传输块结束	39	'	55	7	71	G	87	W	103	g	119	w	
1000	8	8	◘	^H	BS	\b	退格	24	↑	^X	CAN		取消	40	(	56	8	72	H	88	X	104	h	120	x	
1001	9	9	○	^I	HT	\t	横向制表	25	↓	^Y	EM		介质结束	41	)	57	9	73	I	89	Y	105	i	121	y	
1010	A	10	◙	^J	LF	\n	换行	26	→	^Z	SUB		替代	42	*	58	:	74	J	90	Z	106	j	122	z	
1011	B	11	♂	^K	VT	\v	纵向制表	27	←	^[	ESC	\e	溢出	43	+	59	;	75	K	91	[	107	k	123	{	
1100	C	12	♀	^L	FF	\f	换页	28	∟	^\	FS		文件分隔符	44	,	60	<	76	L	92	\	108	l	124	|	
1101	D	13	♪	^M	CR	\r	回车	29	↔	^]	GS		组分隔符	45	-	61	=	77	M	93	]	109	m	125	}	
1110	E	14	♫	^N	SO		移出	30	▲	^^	RS		记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~	
1111	F	15	☼	^O	SI		移入	31	▼	^-	US		单元分隔符	47	/	63	?	79	O	95	_	111	o	127	⌂	^Backspace 代码：DEL

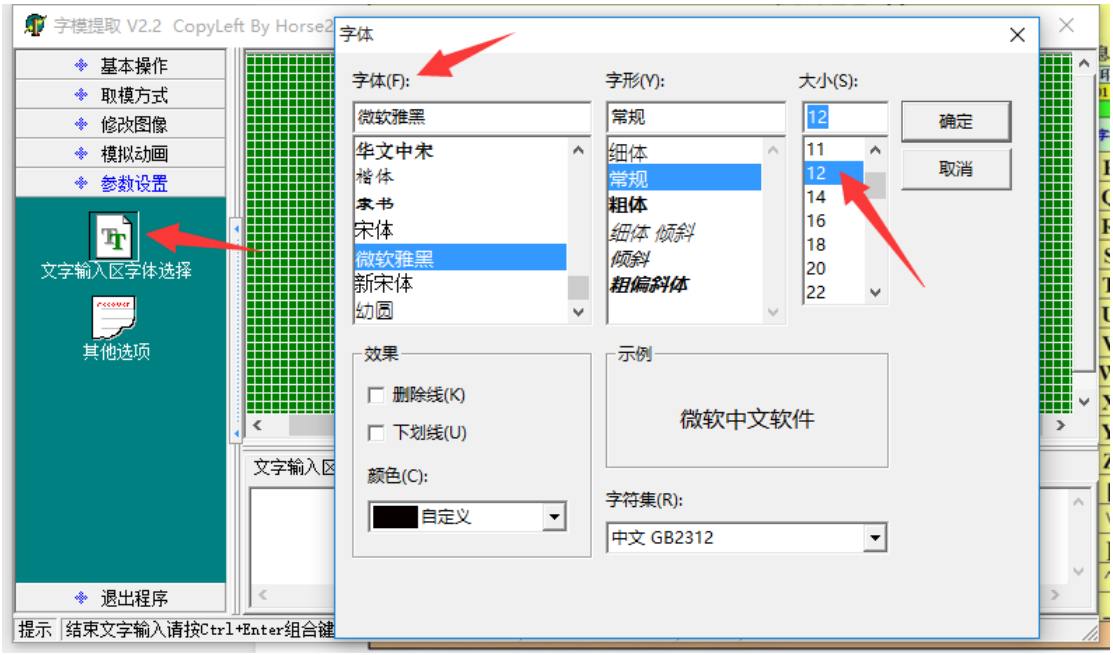
注：表中的ASCII字符可以用“Alt + 小键盘上的数字键”方法输入。 2013/08/08

所以要显示中文就必须使用中文的字符集。中文比较常用的字符集是 GB2312 ,GBK。GB2312 是对 ASCII 的中文扩展。兼容 ASCII。而 GBK 是 GB2312 的扩展，兼容 GB2312,能显示更多的中文。有兴趣的同学可以网上一下这两个字符集的定义，如果要显示中文我们只需要知道。ASCII 码用一个字节表示，中文用两个字节表示。第一个字节小于 127 的字符就是 ASCII 码，占一个字节。第一个字节大于 127 的字符就是中文，由两个字节连在一起表示一个汉字。由于中文需要两个字节，首先要将 keil 设置为 GB2312 编码方式.点击 Edit ->Configuration 打开配置窗口，选择 GB2312 编码。



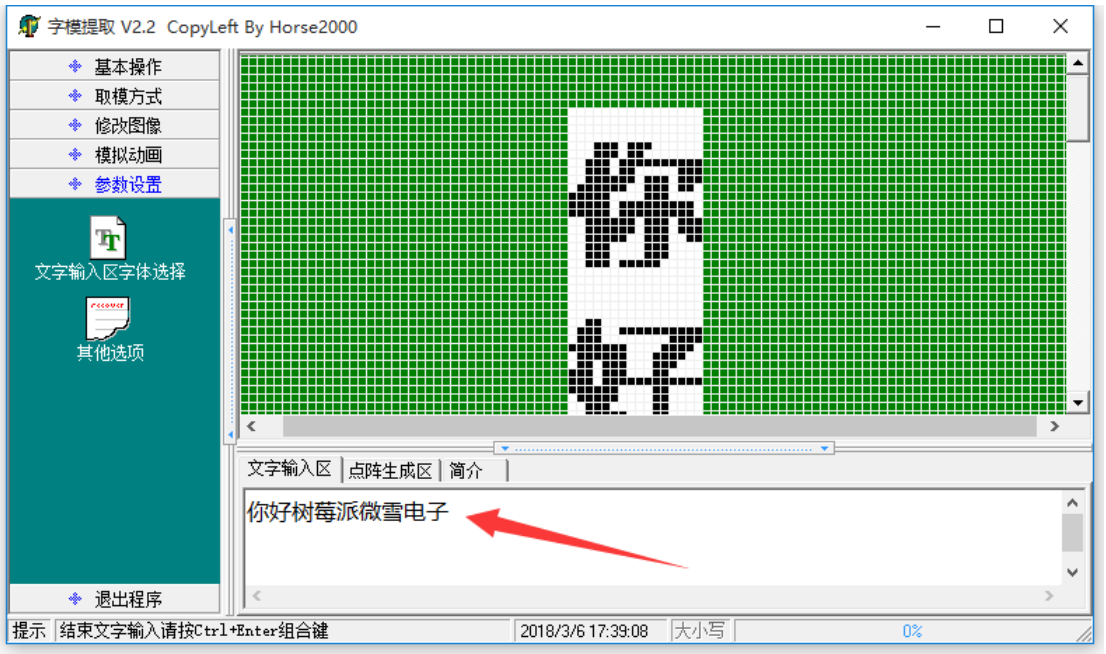
制作中文字符

要显示中文字符，我们首先要制作中文字体。打开字体取模软件，在“参数设置”中点击“字体输入区字体选择”选择对应的字体和大小。

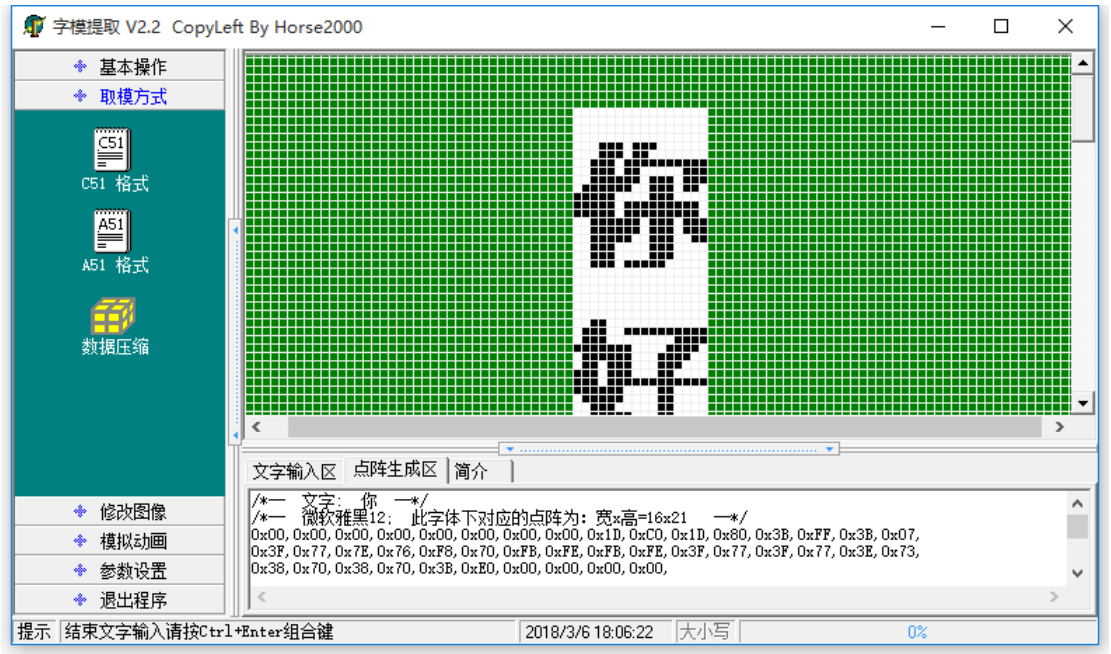


其他选项可以设置取模方式。

在文字输入区输入要取模的汉字，按 Ctrl + Enter 结束输入



在“取模方式”中选择 “C51 格式” 就可以得到字模数据。



要将中文字库添加到程序中，首先要用一个结构体存储字体信息。

```
#include <stdint.h>

typedef struct _tFont
{
  const uint8_t *table;
  uint16_t Width;
  uint16_t Height;

} sFONT;


typedef struct                                          // 汉字字模数据结构
{
  unsigned char index[2];                               // 汉字内码索引
  const char matrix[MAX_HEIGHT_FONT*MAX_WIDTH_FONT/8];  // 点阵码数据
}CH_CN;


typedef struct
{
  const CH_CN *table;
  uint16_t size;
  uint16_t ASCII_Width;
  uint16_t Width;
  uint16_t Height;

}cFONT;

extern sFONT Font24;
extern sFONT Font20;
extern sFONT Font16;
extern sFONT Font12;
extern sFONT Font8;

extern cFONT Font12CN;
extern cFONT Font24CN;
#ifdef __cplusplus
}
#endif
```

和 ASCII 码 sFONT 结构体类似，中文的结构体为 cFONT，包含有字摸数据指针，字符个数,字体宽度，字体高度，还有一个 ASCII 字体宽度。而中文数据是 CH_CN 结构体数组构成，每个字符包含“汉字内码索引”和“点阵码数据”。索引是为了找到这个字符的数据，点阵码数据就是要显示的数据，这个是字模提取软件得到的。

```
const CH_CN Font12CN_Table[] =
{
/*--  文字:  你  --*/
/*--  微软雅黑12;  此字体下对应的点阵为:宽x高=16x21   --*/
{"你",
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1D,0xC0,0x1D,0x80,0x3B,0xFF,0x3B,0x07,
0x3F,0x77,0x7E,0x76,0xF8,0x70,0xFB,0xFE,0xFB,0xFE,0x3F,0x77,0x3F,0x77,0x3E,0x73,
0x38,0x70,0x38,0x70,0x3B,0xE0,0x00,0x00,0x00,0x00},

/*--  文字:  好  --*/
/*--  微软雅黑12;  此字体下对应的点阵为:宽x高=16x21   --*/
{"好",
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x00,0x73,0xFF,0x70,0x0F,0xFE,0x1E,
0x7E,0x3C,0x6E,0x38,0xEE,0x30,0xEF,0xFF,0xFC,0x30,0x7C,0x30,0x38,0x30,0x3E,0x30,
0x7E,0x30,0xE0,0x30,0xC1,0xF0,0x00,0x00,0x00,0x00},

/*--  文字:  树  --*/
/*--  微软雅黑12;  此字体下对应的点阵为:宽x高=16x21   --*/
{"树",
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x0E,0x30,0x0E,0x3F,0xEE,0x30,0xEE,
0xFC,0xFF,0x76,0xCE,0x77,0xFE,0x7B,0xFE,0xFF,0xFE,0xF3,0xDE,0xF3,0xCE,0x37,0xEE,
0x3E,0x6E,0x3C,0x0E,0x30,0x3E,0x00,0x00,0x00,0x00},

/*--  文字:  莓  --*/
/*--  微软雅黑12;  此字体下对应的点阵为:宽x高=16x21   --*/
{"莓",
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x06,0x70,0xFF,0xFF,0x3E,0x70,0x38,0x00,
0x7F,0xFF,0xE0,0x00,0xFF,0xFC,0x3B,0x8C,0x39,0xCC,0xFF,0xFF,0x73,0x9C,0x71,0xDC,
0x7F,0xFF,0x00,0x1C,0x01,0xF8,0x00,0x00,0x00,0x00},
```

下面就是微软雅黑 12 号字体的结构体变量，包含这个字体的信息。其中 Font12N_Table 为数据指针，字符个数为 sizeof(Font12CN_Table)/sizeof(CH_CN)，ASCII 字符宽度为 11，汉字宽度为 16，字体高度为 21.

由于微软雅黑字体中，ASCII 字符的宽度会比汉字小一点，所以为了显示好看，设置两种宽度。

```
cFONT Font12CN = {
  Font12CN_Table,
  sizeof(Font12CN_Table)/sizeof(CH_CN),  /*size of table*/
  11, /* ASCII Width */
  16, /* Width */
  21, /* Height */
};
```

显示中文字符

字体已经制作好了，那么接下来显示了。在程序中添加一个中文字符函数函数。

```
/**
 *  @brief: this displays a string on the frame buffer but not refresh
 */
void Paint_DrawCNStringAt(Paint* paint, int x, int y, const char* text, cFONT* font, int colored) {
    const char* p_text = text;
    int refcolumn = x;
    int i, j,Num;

    /* Send the string character by character on EPD */
    while (*p_text != 0) {
        if(*p_text <= 0x7F){              //ASCII
            for(Num = 0;Num < font->size ;Num++){
                if(*p_text== font->table[Num].index[0]){
                    const char* ptr = &font->table[Num].matrix[0];

                    for (j = 0; j < font->Height; j++) {
                        for (i = 0; i < font->Width; i++) {
                            if (*ptr & (0x80 >> (i % 8))) {
                                Paint_DrawPixel(paint, refcolumn + i, y + j, colored);
                            }
                            if (i % 8 == 7) {
                                ptr++;
                            }
                        }
                        if (font->Width % 8 != 0) {
                            ptr++;
                        }
                    }
                    break;
                }
            }
            /* Point on the next character */
            p_text += 1;
            /* Decrement the column position by 16 */
            refcolumn += font->ASCII_Width;
        }else{                                          //中文
            for(Num = 0;Num < font->size ;Num++){
                if((*p_text== font->table[Num].index[0]) && (*(p_text+1) == font->table[Num].index[1])){
                    const char* ptr = &font->table[Num].matrix[0];

                    for (j = 0; j < font->Height; j++) {
                        for (i = 0; i < font->Width; i++) {
                            if (*ptr & (0x80 >> (i % 8))) {
                                Paint_DrawPixel(paint, refcolumn + i, y + j, colored);
                            }
                            if (i % 8 == 7) {
                                ptr++;
                            }
                        }
                        if (font->Width % 8 != 0) {
                            ptr++;
                        }
                    }
                    break;
                }
            }
            /* Point on the next character */
            p_text += 2;
            /* Decrement the column position by 16 */
            refcolumn += font->Width;
        }
    }
}
```

下面来分析一下这个函数，这个函数主要实现的功能就是在(x,y) 坐标显示 text 字符串，用 cFONT 字体，显示颜色为 colored。

程序首先用 p_text 指针指向要显示的字符串。只要 p_text 指针不为 0 就一直循环显示字符。

首先读取第一个字符，判断是否大于 127（0x7F），如果小于则为 ASCII 码，占一个字节，否则为汉字，占两个字节。

ASCII 码，则用第一个字节和字体数据中的所有 “汉字内码索引"逐个比较，直到找到

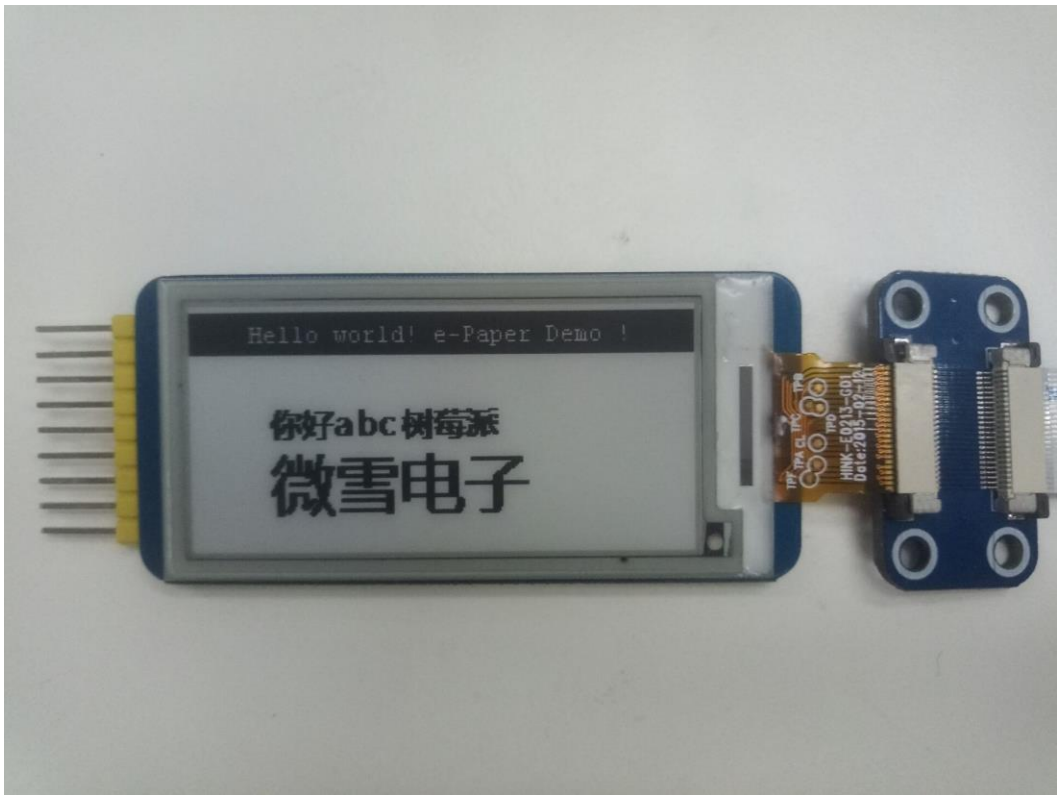
对应的字符位置。然后讲 ptr 指针指向这个字符的点阵码数据，最后将数据一个一个像素的显示出来。

如果是汉字，则用第一个字节和后面一个字节一起 找到对应汉字字符的位置。显示完成后，p_text 指向下一个字符，refcolumn 指向下一个字符要显示的 x 轴位置。

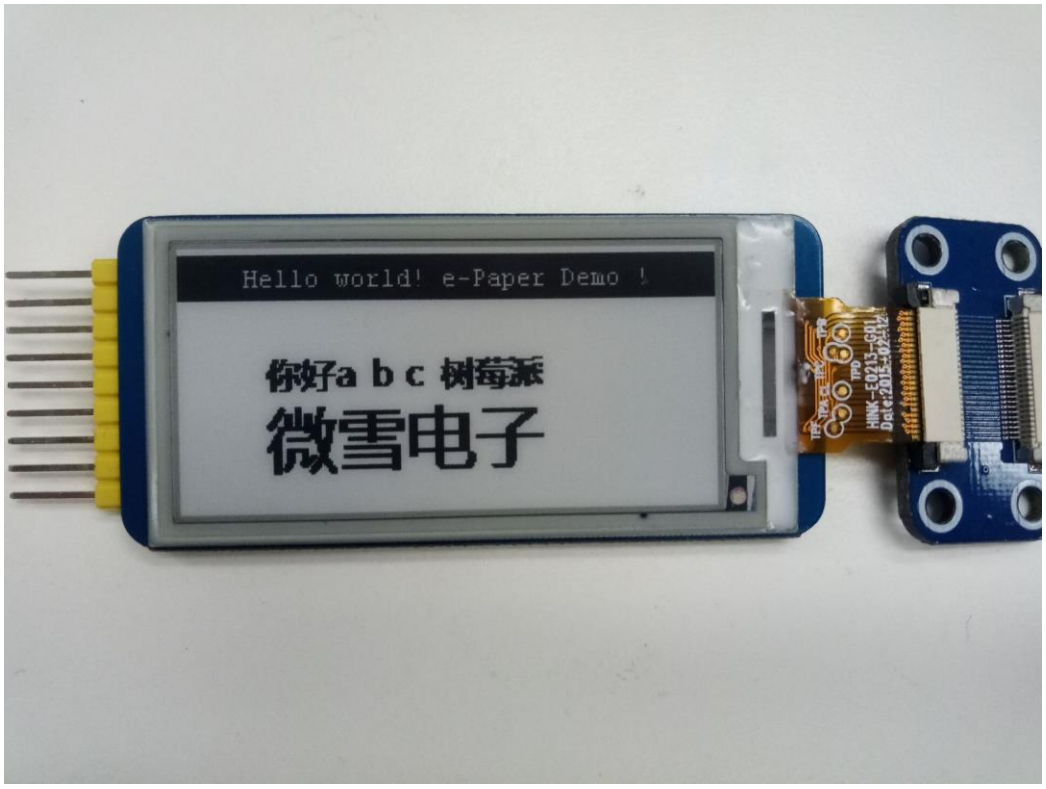
下面为 main 函数中 e-Paper 的显示程序。

```
/* USER CODE BEGIN 2 */
EPD epd;
if (EPD_Init(&epd, lut_full_update) != 0) {
  printf("e-Paper init failed\n");
  return -1;
}

Paint paint;
Paint_Init(&paint, frame_buffer, epd.width, epd.height);
Paint_Clear(&paint, UNCOLORED);
Paint_SetRotate(&paint, ROTATE_90);

/* For simplicity, the arguments are explicit numerical coordinates */
/* Write strings to the buffer */
Paint_DrawFilledRectangle(&paint, 0, 10, 250, 30, COLORED);
Paint_DrawStringAt(&paint, 30, 16, "Hello world! e-Paper Demo !", &Font12, UNCOLORED);

Paint_DrawCNStringAt(&paint, 40, 54,"你好abc树莓派", &Font12CN, COLORED);
Paint_DrawCNStringAt(&paint, 40, 74,"微雪电子", &Font24CN, COLORED);


/* Display the frame_buffer */
EPD_SetFrameMemory(&epd, frame_buffer, 0, 0, Paint_GetWidth(&paint), Paint_GetHeight(&paint));
EPD_DisplayFrame(&epd);
EPD_DelayMs(&epd, 2000);
```

实际显示效果为下图。



如果 ASCII 码的宽度和汉字宽度一样就是显示下面这样，所以为了显示效果更好看一点，程序中设置 ASCII 码和汉字宽度不一样。



最后附赠上整个工程，开发板用 XNUCLEO_F103RB，2.13inch e-Paper HAT 墨水屏。

http://www.waveshare.net/w/upload/7/7f/Stm32.7z

程序中只是添加少量汉字，根据实际情况添加需要用到的汉字，如果需要 GB2312 字符集所以的字符则需要很大的空间，可以考虑点击 SD 卡或者 Flash 芯片存储，或者使用字库芯片，这里不再详细介绍。

树莓派显示中文字符

前面已经介绍如何在 stm32 上显示中文字符，下面我们来介绍一下用树莓派控制 e-Paper 时如果中文。

用树莓派显示中文时推荐用 python 程序，使用 python-image 库这样显示中文会非常简单。

首先我们来看一下 python 程序是如何显示英文字符串的。

```
# For simplicity, the arguments are explicit numerical coordinates
image = Image.new('1', (epd2in13.EPD_WIDTH, epd2in13.EPD_HEIGHT), 255)  # 255: clear the frame
draw = ImageDraw.Draw(image)
font = ImageFont.truetype('/usr/share/fonts/truetype/freefont/FreeMonoBold.ttf', 12)
draw.rectangle((0, 10, 128, 30), fill = 0)
draw.text((30, 14), 'Hello world!', font = font, fill = 255)
draw.text((30, 36), 'e-Paper Demo', font = font, fill = 0)
draw.line((16, 60, 56, 60), fill = 0)
draw.line((56, 60, 56, 110), fill = 0)
draw.line((16, 110, 56, 110), fill = 0)
draw.line((16, 110, 16, 60), fill = 0)
draw.line((16, 60, 56, 110), fill = 0)
draw.line((56, 60, 16, 110), fill = 0)
draw.arc((70, 60, 130, 120), 0, 360, fill = 0)
draw.rectangle((16, 130, 56, 180), fill = 0)
draw.chord((70, 130, 130, 190), 0, 360, fill = 0)

epd.clear_frame_memory(0xFF)
epd.set_frame_memory(image, 0, 0)
epd.display_frame()

epd.delay_ms(2000)
```

从程序中，可以发现 python 程序不需要制作字库，而是使用系统自带的字库。所以要显示中文就非常简单了，只需要使用系统中的中文字库就行了。

树莓派 raspbian 系统是没有自带中文字体的。需要安装中文字库

```
sudo apt-get update
sudo apt-get install ttf-wqy-zenhei ttf-wqy-microhei
```

安装完成后，在下面目录中发现有两个中文字库

```
pi@raspberrypi:/usr/share/fonts/truetype/wqy $ ls
wqy-microhei.ttc  wqy-zenhei.ttc
```

下面是显示中文的程序。首先添加中文字体，然后用 draw.text 函数就行了，这里需要注意的是，包含中文的字符串需要在字符串前面添加"u"，表示使用 unicode 编码。

```
# For simplicity, the arguments are explicit numerical coordinates
image = Image.new('1', (epd2in13.EPD_WIDTH, epd2in13.EPD_HEIGHT), 255)  # 255: clear the frame
#image = image.rotate(90)
draw = ImageDraw.Draw(image)
font = ImageFont.truetype('/usr/share/fonts/truetype/freefont/FreeMonoBold.ttf', 12)
draw.rectangle((0, 10, 128, 30), fill = 0)
draw.text((20, 14), 'Hello world!', font = font, fill = 255)
draw.text((20, 36), 'e-Paper Demo', font = font, fill = 0)
font = ImageFont.truetype('/usr/share/fonts/truetype/wyq/wqy-microhei.ttc', 18)
draw.text((10, 56), u'你好，树莓派', font = font, fill = 0)
font = ImageFont.truetype('/usr/share/fonts/truetype/wyq/wqy-zenhei.ttc', 24)
draw.text((20, 80), u'微雪电子', font = font, fill = 0)
```

另外还需要在文件开头添加 coding: utf-8，否先显示不了中文，另外文件需要用 utf-8 编码。

```
#!/usr/bin/python
# -*- coding:utf-8 -*-
```

如果还是显示不正常，可以用 Notepad++ 软件打开文件，选择以 UTF-8 格式编码。

最后放上源程序和显示效果图

http://www.waveshare.net/w/upload/b/b8/Python.7z

Arduino 显示中文字符

arduino 显示中文字符和 stm32 大概类似，但是有些地方不一样。

第一个就是 arduino IDE 不支持设置文件编码，暂不清楚 Arduino IDE 的编码方式，但是经过测试可知，ASCII 码用一个字节表示，中文用三个字节表示。第一个字节小于 127 的字符就是 ASCII 码，占一个字节。第一个字节大于 127 的字符就是中文，由三个字节连在一起表示一个汉字。因此汉字码索引需要用三个字节存储。

```
typedef struct                                          // 汉字字模数据结构
{
  char index[3];                                        // 汉字内码索引
  const char matrix[MAX_HEIGHT_FONT*MAX_WIDTH_FONT/8];  // 点阵码数据
}CH_CN;
```

第二个就是 arduino 存储读取 flash 数据和 stm32 不一样。存储数据需要添加<avr/pgmspace.h>头文件，并用 PROGMEN 关键字定义，表示数据存储在 flash 中。

```
#include "fonts.h"
#include <avr/pgmspace.h>

//
//  Font data for Courier New 12pt
//

const CH_CN Font12CN_Table[] PROGMEM=
{
/*—  文字:  你  —*/
/*—  微软雅黑12:  此字体下对应的点阵为:宽x高=16x21  —*/
{"你",
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1D,0xC0,0x1D,0x80,0x3B,0xFF,0x3B,0x07,
0x3F,0x77,0x7E,0x76,0xF8,0x70,0xFB,0xFE,0xFB,0xFE,0x3F,0x77,0x3F,0x77,0x3E,0x73,
0x38,0x70,0x38,0x70,0x3B,0xE0,0x00,0x00,0x00,0x00},
```

读取数据的时候用 pgm_read_byte()函数读取。

中文字符显示函数

```
/**
    @brief: this displays a string on the frame buffer but not refresh
*/
void Paint::DrawCNStringAt(int x, int y, const char* text, cFONT* font, int colored) {
  const unsigned char* p_text = text;

  int refcolumn = x;
  int i, j, Num;
  /* Send the string character by character on EPD */
  while (*p_text != 0) {
    if (*p_text < 0x7F) {                                   //ASCII
        for (Num = 0; Num < font->size ; Num++) {
            if (*p_text == pgm_read_byte(&font->table[Num].index[0])) {
                const char* ptr = &font->table[Num].matrix[0];

                for (j = 0; j < font->Height; j++) {
                    for (i = 0; i < font->Width; i++) {
                        if (pgm_read_byte(ptr) & (0x80 >> (i % 8))) {
                            DrawPixel(refcolumn + i, y + j, colored);
                        }
                        if (i % 8 == 7) {
                            ptr++;
                        }
                    }
                    if (font->Width % 8 != 0) {
                      ptr++;
                    }
                }
                break;
            }
        }
        /* Point on the next character */
        p_text += 1;
        /* Decrement the column position by 16 */
        refcolumn += font->ASCII_Width;
    } else {                                    //中文
        for (Num = 0; Num < font->size ; Num++) {
            if ((*p_text == pgm_read_byte(&font->table[Num].index[0])) && (*(p_text + 1) == \
                pgm_read_byte(&font->table[Num].index[1])) && (*(p_text + 2) == pgm_read_byte(&font->table[Num].index[2]))) {

                const char* ptr = &font->table[Num].matrix[0];

                for (j = 0; j < font->Height; j++) {
                    for (i = 0; i < font->Width; i++) {
                        if (pgm_read_byte(ptr) & (0x80 >> (i % 8))) {
                            DrawPixel(refcolumn + i, y + j, colored);
                        }
                        if (i % 8 == 7) {
                            ptr++;
                        }
                    }
                    if (font->Width % 8 != 0) {
                        ptr++;
                    }
                }
                break;
            }
        }
        /* Point on the next character */
        p_text += 3;
        /* Decrement the column position by 16 */
        refcolumn += font->Width;
    }
  }
}
```

需要注意的是，中文字符是三个字节表示。用 pgm_read_byte()读取字体数据。

最后放上源程序。

Epd2in13-demo.7z