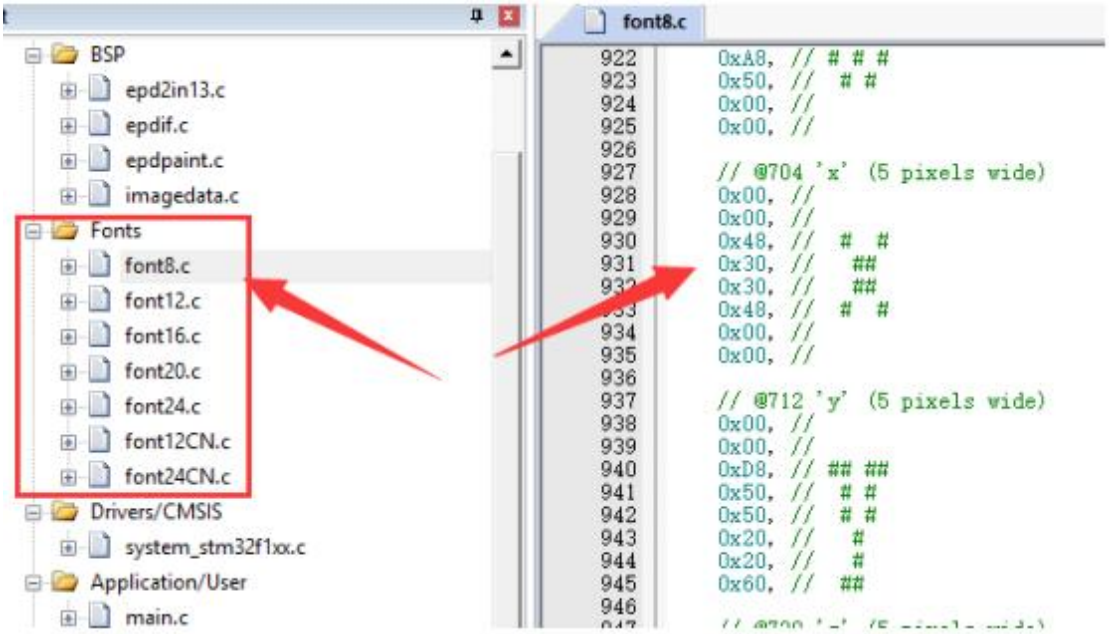
English Character Display Principle

Without further ado, let's learn how ASCII codes are displayed before displaying Chinese character. The following takes the STM32 program of the 2.13inch e-Paper HAT as an example to explain.

To display characters, you must need fonts first. The files of the Fonts directory in the sample program correspond to different fonts. Open the file and you can see a bunch of data.



Each font has a structure that stores information about the font respectively. The structure includes array pointer, font width, font height.

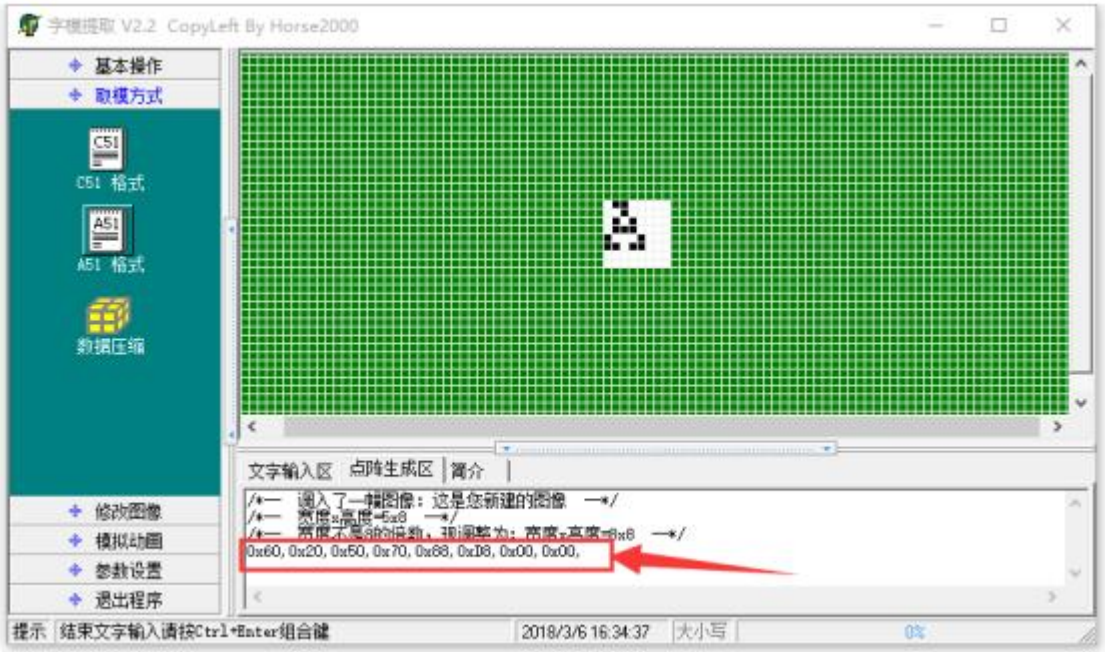
```
typedef struct _tFont
{
  const uint8_t *table;
  uint16_t Width;
  uint16_t Height;

} sFONT;

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

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

The above fonts are copied from the stm32 official sample program. It is an ASCII character. Now let's explain how we make the font. The picture below is the font modulo of the "A" of Font8, we can use the font modulo software to

get the data of the A character.

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



The font data can be got from the font modulo software which extracts the modulo of the font horizontally and vertically, and displays each pixel with an array. For instance, if you want to display the "A" character, you can find the data of the "A" character and then display the font modulo point by point.

```
/**
 * @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++;
        }
    }
}
```

One thing to note here is the red box, the font array is stored in ASCII order, the first character is a space " ", and the data size of each character is the same. So subtract the ASCII code of the space bar from the ASCII code of A to find the starting position of the data for the character "A".

```
/**
 *  @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++;
    }
}
```

String display is to display each character.

Character Set

Well, we already know about how to display English characters. You also need to understand the character set before displaying Chinese. What is a character set? A character set is a collection of all characters, ASCII code is a character set, and ASCII has only 0~127 characters representing with one byte. Only English can be displayed, not Chinese.

ASCII表

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

高四位 / 低四位	0000	0001	0010	0011	0100	0101	0110	0111
	ASCII控制字符		ASCII打印字符					
	0	1	2	3	4	5	6	7
0000	0	16 ►	32	48 0	64 @	80 P	96 `	112 p
0001	1 ☺	17 ◄	33 !	49 1	65 A	81 Q	97 a	113 q
0010	2 ☻	18 ↕	34 "	50 2	66 B	82 R	98 b	114 r
0011	3 ♥	19 ‼	35 #	51 3	67 C	83 S	99 c	115 s
0100	4 ♦	20 ¶	36 $	52 4	68 D	84 T	100 d	116 t
0101	5 ♣	21 §	37 %	53 5	69 E	85 U	101 e	117 u
0110	6 ♠	22 ▬	38 &	54 6	70 F	86 V	102 f	118 v
0111	7 •	23 ↨	39 '	55 7	71 G	87 W	103 g	119 w
1000	8 ◘	24 ↑	40 (	56 8	72 H	88 X	104 h	120 x
1001	9 ○	25 ↓	41)	57 9	73 I	89 Y	105 i	121 y
1010	A 10 ◙	26 →	42 *	58 :	74 J	90 Z	106 j	122 z
1011	B 11 ♂	27 ←	43 +	59 ;	75 K	91 [	107 k	123 {
1100	C 12 ♀	28 ∟	44 ,	60 <	76 L	92 \	108 l	124 |
1101	D 13 ♪	29 ↔	45 -	61 =	77 M	93]	109 m	125 }
1110	E 14 ♫	30 ▲	46 .	62 >	78 N	94 ^	110 n	126 ~
1111	F 15 ☼	31 ▼	47 /	63 ?	79 O	95 _	111 o	127 ⌂

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

So to display Chinese, you must use the Chinese character set. The more

commonly used character sets in Chinese are GB2312 and GBK.

GB2312 is an extension of ASCII for Chinese characters and is compatible with ASCII. GBK is an extension of GB2312, compatible with GB2312, and can display more Chinese. If you are interested, you can find the definitions of these two character sets online.

If we want to display Chinese, we only need to learn that the ASCII code is represented by one byte while Chinese is represented by two bytes. The character whose first byte is less than 127 is the ASCII code that is only one byte. The character whose first byte is greater than 127 is Chinese, and two bytes are connected together to represent a Chinese character. As Chinese character requires two bytes, you must set Keil to GB2312 encoding mode first. Click Edit -> Configuration to open the configuration window and select Chinese GB2312 (Simplified).