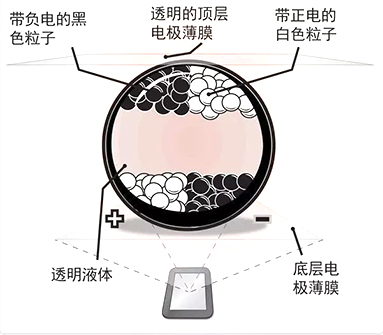
**电子墨水屏**

**一、墨水屏简介**

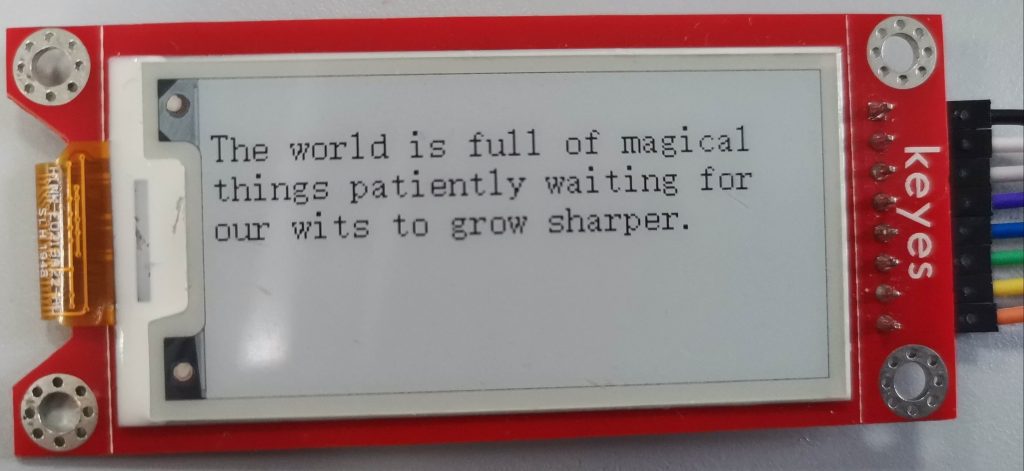
电子墨水屏技术最早可以追溯到 1996 年，它基于美国麻省理工学院媒体实验室（MIT Media Lab）的一项研究，利用电泳技术（EPD）实现显示，这类屏幕的显示效果十分接近传统纸张，因此也被成为“电子纸”。1997 年，麻省理工学院的教授 Joseph Jacobson 创立 E-Ink 公司，开始推动电子纸技术走向商业化，电子墨水技术成为电子纸的主流。

电子墨水与印刷使用的墨水很相似，都是用颜料所制，这也是为什么我们看到电子墨水屏和传统纸张显示效果相似的原因。电子墨水通常会制成薄膜，由大量微囊组成，这些微囊只有人类头发的直径大小。微囊中的黑白小球是带不同电荷的色素颗粒，初始状态下，色素颗粒悬浮在微囊中，当施加一定方向的电场后，相应的色素颗粒被推到顶部，微囊就会显示不同的颜色，而不同颜色的微囊组成了各种文字和图案。



选购墨水屏时建议买微雪的墨水屏，各种资料比较丰富，方便后期开发。

**二、 2.13寸屏幕属性**



本次使用的墨水屏是在淘宝上随便找了一家店买的，大小为2.13寸，分辨率250\*122,采用spi通信。商家只给了树莓派和arduino的示例代码。如果想用stm32f103来驱动的话需要自己进行移植。

当给屏幕对应像素点写1时，屏幕为白色，写0时，为黑色，故在取字模时要选择阳码。取模方式为逐列式，具体配置参见本文第四部分。

想要成功驱动屏幕需要给屏幕的寄存器写入正确的值，屏幕的数据手册给出了各个寄存器的作用和配置方法。

此屏幕的商家没有提供数据手册，我找到了微雪同尺寸墨水屏的数据手册可以作为参考：[下载](http://47.101.64.18:8888/down/mgJJ2qfyDg1n)

以下为常用寄存器：

|  |  |  |  |
| --- | --- | --- | --- |
| 寄存器 | 功能 | 配置 | 参数 |
| 0x22 | 刷新控制 | 全局刷新： 局部刷新 ： | 0xf7 0xff |
| 0x20 | 刷新 |  |  |
| 0x24 | 写入要显示内容，指针自动增加 |  |  |
| 0x44 | 显示起始和结束位置x坐标 |  |  |
| 0x45 | 显示起始和结束位置y坐标 |  |  |
| 0x4e | 显示起始地址x坐标 |  |  |
| 0x4f | 显示起始地址y坐标 |  |  |

**三、主要代码（Stm32）**

屏幕与墨水屏连线：

* SDI ——> A7
* SCLK ——> A5
* CS ——> A4
* D/C ——> B1
* RES ——> B0
* BUSY ——> A6

对端口使能：

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE); //使能A端口时钟

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_4|GPIO\_Pin\_5|GPIO\_Pin\_7;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP; //推挽输出

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;//速度50MHz

GPIO\_Init(GPIOA, &GPIO\_InitStructure); //初始化GPIOD3,6

GPIO\_SetBits(GPIOA,GPIO\_Pin\_5|GPIO\_Pin\_7|GPIO\_Pin\_4);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_6;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPD; //PA0设置成输入，默认下拉

GPIO\_Init(GPIOA, &GPIO\_InitStructure);//初始化GPIOA.0

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB, ENABLE); //使能B端口时钟

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0|GPIO\_Pin\_1;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP; //推挽输出

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;//速度50MHz

GPIO\_Init(GPIOB, &GPIO\_InitStructure); //初始化GPIOD3,6

GPIO\_SetBits(GPIOB,GPIO\_Pin\_0|GPIO\_Pin\_1);

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

SPI通信：

//SPI写命令，一次写一个字节

void SPI\_Write(unsigned char value)

{

unsigned char i;

SPI\_Delay(1);

for(i=0; i<8; i++)

{

EPD\_W21\_CLK\_0;

SPI\_Delay(1);

if(value & 0x80)

EPD\_W21\_MOSI\_1;

else

EPD\_W21\_MOSI\_0;

value = (value << 1);

SPI\_Delay(1);

delay\_us(1);

EPD\_W21\_CLK\_1;

SPI\_Delay(1);

}

}

//写命令，即想要修改的寄存器。向墨水屏写入命令时需将CS、DC设置为0,写完后再将CS设置为1

void Epaper\_Write\_Command(unsigned char command)

{

SPI\_Delay(1);

EPD\_W21\_CS\_0;

EPD\_W21\_DC\_0; // command write

SPI\_Write(command);

EPD\_W21\_CS\_1;

}

//写数据，即写入寄存器的内容。向墨水屏写入数据时需将CS设置为0,DC设置为1,写完后再将CS设置为1

void Epaper\_Write\_Data(unsigned char command)

{

SPI\_Delay(1);

EPD\_W21\_CS\_0;

EPD\_W21\_DC\_1; // command write

SPI\_Write(command);

EPD\_W21\_CS\_1;

}

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

屏幕初始化：

void EPD\_HW\_Init(void)

{

EPD\_W21\_RST\_0;

delay\_ms(1);

EPD\_W21\_RST\_1; //hard reset

delay\_ms(1);

Epaper\_READBUSY();

Epaper\_Write\_Command(0x12); //SWRESET

Epaper\_READBUSY();

Epaper\_Write\_Command(0x01); //Driver output control

Epaper\_Write\_Data(0xF9);

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Command(0x11); //data entry mode

Epaper\_Write\_Data(0x01);

Epaper\_Write\_Command(0x44); //set Ram-X address start/end position

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Data(0x0F); //0x0F-->(15+1)\*8=128

Epaper\_Write\_Command(0x45); //set Ram-Y address start/end position

Epaper\_Write\_Data(0xF9); //0xF9-->(249+1)=250

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Command(0x3C); //BorderWavefrom

Epaper\_Write\_Data(0x01);

Epaper\_Write\_Command(0x18);

Epaper\_Write\_Data(0x80);

Epaper\_Write\_Command(0x4E); // set RAM x address count to 0;

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Command(0x4F); // set RAM y address count to 0X199;

Epaper\_Write\_Data(0xF9);

Epaper\_Write\_Data(0x00);

Epaper\_READBUSY();

}

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

屏幕刷新：

//全屏刷新

void EPD\_Update(void)

{

Epaper\_Write\_Command(0x22);

Epaper\_Write\_Data(0xF7);

Epaper\_Write\_Command(0x20);

Epaper\_READBUSY();

}

//局部刷新。经过实际测试，该款屏幕不能局部刷新，此函数作用与上一个相同

void EPD\_Part\_Update(void)

{

Epaper\_Write\_Command(0x22);

Epaper\_Write\_Data(0xff);

Epaper\_Write\_Command(0x20);

Epaper\_READBUSY();

}

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

全屏显示：

//屏幕全黑，写0为黑

void EPD\_WhiteScreen\_Black(void)

{

unsigned int i,k;

Epaper\_Write\_Command(0x44); //set Ram-X address start/end position

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Data(0x0F); //0x0F-->(15+1)\*8=128

Epaper\_Write\_Command(0x45); //set Ram-Y address start/end position

Epaper\_Write\_Data(0xF9); //0xF9-->(249+1)=250

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Command(0x4E); // set RAM x address count to 0;

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Command(0x4F); // set RAM y address count to 0X199;

Epaper\_Write\_Data(0xF9);

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Command(0x24);

for(k=0;k<250;k++)

{

for(i=0;i<16;i++)

{

Epaper\_Write\_Data(0x00);

}

}

EPD\_Update();

}

//屏幕全白，写1为白

void EPD\_WhiteScreen\_White(void)

{

unsigned int i,k;

Epaper\_Write\_Command(0x44); //set Ram-X address start/end position

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Data(0x0F); //0x0F-->(15+1)\*8=128

Epaper\_Write\_Command(0x45); //set Ram-Y address start/end position

Epaper\_Write\_Data(0xF9); //0xF9-->(249+1)=250

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Command(0x4E); // set RAM x address count to 0;

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Command(0x4F); // set RAM y address count to 0X199;

Epaper\_Write\_Data(0xF9);

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Command(0x24);

for(k=0;k<250;k++)

{

for(i=0;i<16;i++)

{

Epaper\_Write\_Data(0xff);

}

}

EPD\_Update();

}

//全屏显示图片

void EPD\_WhiteScreen\_ALL(const unsigned char \* datas)

{

unsigned int i;

Epaper\_Write\_Command(0x44); //set Ram-X address start/end position

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Data(0x0F); //0x0F-->(15+1)\*8=128

Epaper\_Write\_Command(0x45); //set Ram-Y address start/end position

Epaper\_Write\_Data(0xF9); //0xF9-->(249+1)=250

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Command(0x4E); // set RAM x address count to 0;

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Command(0x4F); // set RAM y address count to 0X199;

Epaper\_Write\_Data(0xF9);

Epaper\_Write\_Data(0x00);

Epaper\_Write\_Command(0x24);

for(i=0;i<ALLSCREEN\_GRAGHBYTES;i++)

{

Epaper\_Write\_Data(datas[i]);

}

EPD\_Update();

}

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

显示字符和汉字：

//显示8\*16字符

//x,y为要显示的位置。以左上角为起点，y为横轴，x为纵轴

//y取值范围:0~243

//x取值范围:0~111

void ShowChar(unsigned int x,unsigned int y,u8 chr)

{

unsigned int i;

unsigned char c=0;

unsigned int x\_end,y\_start1,y\_start2,y\_end1,y\_end2;

c=chr-' ';//得到字符在数组中的位置

x=x/8; //x轴像素点8个一组，输入寄存器的实际值为组数，故除8

x\_end=x+16/8-1;

y\_start1=0;

y\_start2=y;

if(y >= 256)

{

y\_start1=y\_start2/256;

y\_start2=y\_start2%256;

}

y\_end1=0;

y\_end2=y+8-1;

if(y\_end2>=256)

{

y\_end1=y\_end2/256;

y\_end2=y\_end2%256;

}

Epaper\_Write\_Command(0x44); // 设置字符x轴起始和结束位置

Epaper\_Write\_Data(x); // 起始位置

Epaper\_Write\_Data(x\_end); // 结束位置

Epaper\_Write\_Command(0x45); // 设置字符y轴起始和结束位置,因为y轴共有250个像素，一个字节放不下，所以分成两个字节

Epaper\_Write\_Data(249-y\_start2); // 起始位置高八位，因为屏幕实际起始点为右上角，改为左上角要用249减y

Epaper\_Write\_Data(y\_start1); // 起始位置低八位

Epaper\_Write\_Data(249-y\_end2); // 结束位置高八位

Epaper\_Write\_Data(y\_end1); // 结束位置低八位

Epaper\_Write\_Command(0x4E);

Epaper\_Write\_Data(x);

Epaper\_Write\_Command(0x4F);

Epaper\_Write\_Data(249-y\_start2);

Epaper\_Write\_Data(y\_start1);

Epaper\_Write\_Command(0x24); //写入字符取模后的值

for(i=0;i<16;i++)

{

Epaper\_Write\_Data(F8X16[c\*16+i]);

}

EPD\_Update(); //刷新屏幕

}

//在给屏幕写入要显示的内容后，需要调用EPD\_Updata()刷新后才能在屏幕显示。

//此函数在给屏幕写入8\*16字符后不刷新。

//ShowStrs()通过调用此函数将字符串内容全部写入后再进行刷新

//y:0~243

//x:0~111

void ShowChar1(unsigned int x,unsigned int y,u8 chr)

{

unsigned int i;

unsigned char c=0;

unsigned int x\_end,y\_start1,y\_start2,y\_end1,y\_end2;

c=chr-' ';

x=x/8;

x\_end=x+16/8-1;

y\_start1=0;

y\_start2=y;

if(y >= 256)

{

y\_start1=y\_start2/256;

y\_start2=y\_start2%256;

}

y\_end1=0;

y\_end2=y+8-1;

if(y\_end2>=256)

{

y\_end1=y\_end2/256;

y\_end2=y\_end2%256;

}

Epaper\_Write\_Command(0x44);

Epaper\_Write\_Data(x);

Epaper\_Write\_Data(x\_end);

Epaper\_Write\_Command(0x45);

Epaper\_Write\_Data(249-y\_start2);

Epaper\_Write\_Data(y\_start1);

Epaper\_Write\_Data(249-y\_end2);

Epaper\_Write\_Data(y\_end1);

Epaper\_Write\_Command(0x4E);

Epaper\_Write\_Data(x);

Epaper\_Write\_Command(0x4F);

Epaper\_Write\_Data(249-y\_start2);

Epaper\_Write\_Data(y\_start1);

Epaper\_Write\_Command(0x24);

for(i=0;i<16;i++)

{

Epaper\_Write\_Data(F8X16[c\*16+i]);

}

}

//显示字符串，只能完整显示7行，每行31个字符

void ShowStrs(u8 x,u8 y,u8 \*chr)

{

unsigned char j=0;

while (chr[j]!='\0')

{ ShowChar1(x,y,chr[j]);

y+=8;

if(y>243) //一行满后自动换行

{

y=0;

x+=16;

}

j++;

}

EPD\_Update();

}

//显示汉字，16\*16

void show\_Cinese(unsigned int x,unsigned int y,u8 chr)

{

unsigned int i;

unsigned int x\_end,y\_start1,y\_start2,y\_end1,y\_end2;

x=x/8;

x\_end=x+16/8-1;

y\_start1=0;

y\_start2=y;

if(y >= 256)

{

y\_start1=y\_start2/256;

y\_start2=y\_start2%256;

}

y\_end1=0;

y\_end2=y+16-1;

if(y\_end2>=256)

{

y\_end1=y\_end2/256;

y\_end2=y\_end2%256;

}

Epaper\_Write\_Command(0x44);

Epaper\_Write\_Data(x);

Epaper\_Write\_Data(x\_end);

Epaper\_Write\_Command(0x45);

Epaper\_Write\_Data(249-y\_start2);

Epaper\_Write\_Data(y\_start1);

Epaper\_Write\_Data(249-y\_end2);

Epaper\_Write\_Data(y\_end1);

Epaper\_Write\_Command(0x4E);

Epaper\_Write\_Data(x);

Epaper\_Write\_Command(0x4F);

Epaper\_Write\_Data(249-y\_start2);

Epaper\_Write\_Data(y\_start1);

Epaper\_Write\_Command(0x24);

for(i=0;i<32;i++)

{

Epaper\_Write\_Data(Hzk[chr][i]);

}

EPD\_Part\_Update();

}

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

完整代码：[下载](http://47.101.64.18:8888/down/Rmfdb80sdiiR" \t "_blank)

**四、取模方法**

在控制墨水屏显示内容的时候是控制每个像素点，写0时像素点为黑色，写1时像素点为白色。想要让这些像素点组成字符则需要将字符转换为对应的01序列，通过此序列来控制像素点的黑白，从而达到显示字符的目的。将字符转换为01序列的过程即为取模，不仅字符可以取模，汉字和图像也可以。

网上有很多取字模的软件，经对比发现PCtoLCD2002最好用。[下载](http://47.101.64.18:8888/down/URgRxQhhBq5U" \t "_blank)

**字符字模制作：**

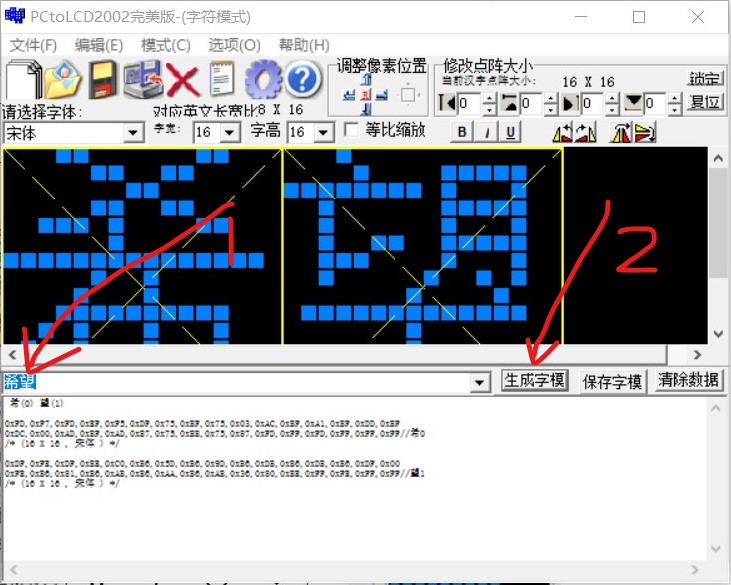
打开软件后选择字符模式：



点击齿轮进行如下配置：



设置好字体和宽高后（一般字符为8\*16，汉字为16\*16），输入要生成的字模，点击生成字模：



将生成的字模放入程序的数组中，供程序调用显示，以下为生成的ASCII、8\*16字符字模数组：

const unsigned char F8X16[]=

{

/\*

(0) !(1) "(2) #(3) $(4) %(5) &(6) '(7) ((8) )(9) \*(10) +(11) ,(12) -(13) .(14) /(15) 0(16) 1(17) 2(18) 3(19)

4(20) 5(21) 6(22) 7(23) 8(24) 9(25) :(26) ;(27) <(28) =(29) >(30) ?(31) @(32) A(33) B(34) C(35) D(36) E(37) F(38) G(39)

H(40) I(41) J(42) K(43) L(44) M(45) N(46) O(47) P(48) Q(49) R(50) S(51) T(52) U(53) V(54) W(55) X(56) Y(57) Z(58) [(59)

\(60) ](61) ^(62) \_(63) `(64) a(65) b(66) c(67) d(68) e(69) f(70) g(71) h(72) i(73) j(74) k(75) l(76) m(77) n(78) o(79)

p(80) q(81) r(82) s(83) t(84) u(85) v(86) w(87) x(88) y(89) z(90) {(91) |(92) }(93) ~(94)

\*/

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,// 0

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xE0,0x33,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,//!1

0xFF,0xFF,0xF7,0xFF,0xCF,0xFF,0xBF,0xFF,0xF7,0xFF,0xCF,0xFF,0xBF,0xFF,0xFF,0xFF,//"2

0xFF,0xFF,0xFD,0xDF,0xFC,0x03,0xE1,0xDF,0xFD,0xDF,0xFC,0x03,0xE1,0xDF,0xFF,0xFF,//#3

0xFF,0xFF,0xF1,0xE7,0xEE,0xFB,0xEE,0xFB,0xC0,0x00,0xEF,0x7B,0xF3,0x87,0xFF,0xFF,//$4

0xF0,0xFF,0xEF,0x73,0xF0,0xCF,0xFE,0x3F,0xF9,0x87,0xE7,0x7B,0xFF,0x87,0xFF,0xFF,//%5

0xFF,0x87,0xF0,0x7B,0xEF,0x3B,0xEE,0xCB,0xF1,0x67,0xFF,0x1B,0xFF,0x7B,0xFF,0xF7,//&6

0xFF,0xFF,0xB7,0xFF,0x8F,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,//'7

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF8,0x1F,0xE7,0xE7,0xDF,0xFB,0xBF,0xFD,0xFF,0xFF,//(8

0xFF,0xFF,0xBF,0xFD,0xDF,0xFB,0xE7,0xE7,0xF8,0x1F,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,//)9

0xFD,0xBF,0xFD,0xBF,0xFE,0x7F,0xF0,0x0F,0xFE,0x7F,0xFD,0xBF,0xFD,0xBF,0xFF,0xFF,//\*10

0xFF,0xFF,0xFF,0x7F,0xFF,0x7F,0xFF,0x7F,0xF8,0x0F,0xFF,0x7F,0xFF,0x7F,0xFF,0x7F,//+11

0xFF,0xFF,0xFF,0xF6,0xFF,0xF1,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,//,12

0xFF,0xFF,0xFF,0x7F,0xFF,0x7F,0xFF,0x7F,0xFF,0x7F,0xFF,0x7F,0xFF,0x7F,0xFF,0xFF,//-13

0xFF,0xFF,0xFF,0xF3,0xFF,0xF3,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,//.14

0xFF,0xFF,0xFF,0xF9,0xFF,0xE7,0xFF,0x1F,0xFC,0xFF,0xE3,0xFF,0xDF,0xFF,0xFF,0xFF,///15

0xFF,0xFF,0xF8,0x0F,0xF7,0xF7,0xEF,0xFB,0xEF,0xFB,0xF7,0xF7,0xF8,0x0F,0xFF,0xFF,//016

0xFF,0xFF,0xFF,0xFF,0xF7,0xFB,0xF7,0xFB,0xE0,0x03,0xFF,0xFB,0xFF,0xFB,0xFF,0xFF,//117

0xFF,0xFF,0xF1,0xF3,0xEF,0xEB,0xEF,0xDB,0xEF,0xBB,0xEF,0x7B,0xF0,0xF3,0xFF,0xFF,//218

0xFF,0xFF,0xF3,0xE7,0xEF,0xFB,0xEF,0x7B,0xEF,0x7B,0xEE,0xBB,0xF1,0xC7,0xFF,0xFF,//319

0xFF,0xFF,0xFF,0x9F,0xFE,0x5F,0xFD,0xDB,0xF3,0xDB,0xE0,0x03,0xFF,0xDB,0xFF,0xDB,//420

0xFF,0xFF,0xE0,0x67,0xEE,0xFB,0xEE,0xFB,0xEE,0xFB,0xEF,0x77,0xEF,0x8F,0xFF,0xFF,//521

0xFF,0xFF,0xF8,0x0F,0xF7,0x77,0xEE,0xFB,0xEE,0xFB,0xF6,0xFB,0xFF,0x07,0xFF,0xFF,//622

0xFF,0xFF,0xE7,0xFF,0xEF,0xFF,0xEF,0x83,0xEE,0x7F,0xE9,0xFF,0xE7,0xFF,0xFF,0xFF,//723

0xFF,0xFF,0xF1,0xC7,0xEE,0xBB,0xEF,0x7B,0xEF,0x7B,0xEE,0xBB,0xF1,0xC7,0xFF,0xFF,//824

0xFF,0xFF,0xF0,0x7F,0xEF,0xB7,0xEF,0xBB,0xEF,0xBB,0xF7,0x77,0xF8,0x0F,0xFF,0xFF,//925

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFC,0xF3,0xFC,0xF3,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,//:26

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFE,0xF8,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,//;27

0xFF,0xFF,0xFF,0x7F,0xFE,0xBF,0xFD,0xDF,0xFB,0xEF,0xF7,0xF7,0xEF,0xFB,0xFF,0xFF,//<28

0xFF,0xFF,0xFD,0xBF,0xFD,0xBF,0xFD,0xBF,0xFD,0xBF,0xFD,0xBF,0xFD,0xBF,0xFF,0xFF,//=29

0xFF,0xFF,0xEF,0xFB,0xF7,0xF7,0xFB,0xEF,0xFD,0xDF,0xFE,0xBF,0xFF,0x7F,0xFF,0xFF,//>30

0xFF,0xFF,0xF1,0xFF,0xED,0xFF,0xEF,0xF3,0xEF,0x13,0xEE,0xFF,0xF1,0xFF,0xFF,0xFF,//?31

0xFC,0x1F,0xF3,0xE7,0xEC,0x1B,0xEB,0xEB,0xE8,0x0B,0xF7,0xEB,0xF8,0x17,0xFF,0xFF,//@32

0xFF,0xFB,0xFF,0xC3,0xFC,0x3B,0xE3,0xBF,0xF8,0xBF,0xFF,0x1B,0xFF,0xE3,0xFF,0xFB,//A33

0xEF,0xFB,0xE0,0x03,0xEE,0xFB,0xEE,0xFB,0xEE,0xFB,0xF1,0x77,0xFF,0x8F,0xFF,0xFF,//B34

0xFC,0x1F,0xF3,0xE7,0xEF,0xFB,0xEF,0xFB,0xEF,0xFB,0xEF,0xF7,0xE3,0xEF,0xFF,0xFF,//C35

0xEF,0xFB,0xE0,0x03,0xEF,0xFB,0xEF,0xFB,0xEF,0xFB,0xF7,0xF7,0xF8,0x0F,0xFF,0xFF,//D36

0xEF,0xFB,0xE0,0x03,0xEE,0xFB,0xEE,0xFB,0xE8,0x3B,0xEF,0xFB,0xF7,0xE7,0xFF,0xFF,//E37

0xEF,0xFB,0xE0,0x03,0xEE,0xFB,0xEE,0xFF,0xE8,0x3F,0xEF,0xFF,0xF7,0xFF,0xFF,0xFF,//F38

0xFC,0x1F,0xF3,0xE7,0xEF,0xFB,0xEF,0xFB,0xEF,0xBB,0xE3,0x87,0xFF,0xBF,0xFF,0xFF,//G39

0xEF,0xFB,0xE0,0x03,0xEF,0x7B,0xFF,0x7F,0xFF,0x7F,0xEF,0x7B,0xE0,0x03,0xEF,0xFB,//H40

0xFF,0xFF,0xEF,0xFB,0xEF,0xFB,0xE0,0x03,0xEF,0xFB,0xEF,0xFB,0xFF,0xFF,0xFF,0xFF,//I41

0xFF,0xFC,0xFF,0xFE,0xEF,0xFE,0xEF,0xFE,0xE0,0x01,0xEF,0xFF,0xEF,0xFF,0xFF,0xFF,//J42

0xEF,0xFB,0xE0,0x03,0xEE,0xFB,0xFC,0x7F,0xEB,0x9B,0xE7,0xE3,0xEF,0xFB,0xFF,0xFF,//K43

0xEF,0xFB,0xE0,0x03,0xEF,0xFB,0xFF,0xFB,0xFF,0xFB,0xFF,0xFB,0xFF,0xF3,0xFF,0xFF,//L44

0xEF,0xFB,0xE0,0x03,0xE0,0x7F,0xFF,0x83,0xE0,0x7F,0xE0,0x03,0xEF,0xFB,0xFF,0xFF,//M45

0xEF,0xFB,0xE0,0x03,0xF3,0xFB,0xFC,0xFF,0xFF,0x1F,0xEF,0xE7,0xE0,0x03,0xEF,0xFF,//N46

0xF8,0x0F,0xF7,0xF7,0xEF,0xFB,0xEF,0xFB,0xEF,0xFB,0xF7,0xF7,0xF8,0x0F,0xFF,0xFF,//O47

0xEF,0xFB,0xE0,0x03,0xEF,0x7B,0xEF,0x7F,0xEF,0x7F,0xEF,0x7F,0xF0,0xFF,0xFF,0xFF,//P48

0xF8,0x0F,0xF7,0xF7,0xEF,0xEB,0xEF,0xEB,0xEF,0xF3,0xF7,0xF5,0xF8,0x0D,0xFF,0xFF,//Q49

0xEF,0xFB,0xE0,0x03,0xEE,0xFB,0xEE,0xFF,0xEE,0x3F,0xEE,0xCF,0xF1,0xF3,0xFF,0xFB,//R50

0xFF,0xFF,0xF1,0xE3,0xEE,0xFB,0xEF,0x7B,0xEF,0x7B,0xEF,0xBB,0xE3,0xC7,0xFF,0xFF,//S51

0xE7,0xFF,0xEF,0xFF,0xEF,0xFB,0xE0,0x03,0xEF,0xFB,0xEF,0xFF,0xE7,0xFF,0xFF,0xFF,//T52

0xEF,0xFF,0xE0,0x07,0xEF,0xFB,0xFF,0xFB,0xFF,0xFB,0xEF,0xFB,0xE0,0x07,0xEF,0xFF,//U53

0xEF,0xFF,0xE1,0xFF,0xEE,0x1F,0xFF,0xE3,0xFF,0x8F,0xEC,0x7F,0xE3,0xFF,0xEF,0xFF,//V54

0xEF,0xFF,0xE0,0x3F,0xFF,0x83,0xE0,0x7F,0xFF,0x83,0xE0,0x3F,0xEF,0xFF,0xFF,0xFF,//W55

0xEF,0xFB,0xE7,0xF3,0xE9,0xCB,0xFE,0x3F,0xFE,0x3F,0xE9,0xCB,0xE7,0xF3,0xEF,0xFB,//X56

0xEF,0xFF,0xE3,0xFF,0xEC,0xFB,0xFF,0x03,0xEC,0xFB,0xE3,0xFF,0xEF,0xFF,0xFF,0xFF,//Y57

0xF7,0xFB,0xEF,0xE3,0xEF,0x9B,0xEF,0x7B,0xEC,0xFB,0xE3,0xFB,0xEF,0xE7,0xFF,0xFF,//Z58

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x80,0x01,0xBF,0xFD,0xBF,0xFD,0xBF,0xFD,0xFF,0xFF,//[59

0xFF,0xFF,0xDF,0xFF,0xE3,0xFF,0xFC,0x7F,0xFF,0x9F,0xFF,0xE3,0xFF,0xFC,0xFF,0xFF,//\60

0xFF,0xFF,0xBF,0xFD,0xBF,0xFD,0xBF,0xFD,0x80,0x01,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,//]61

0xFF,0xFF,0xFF,0xFF,0xDF,0xFF,0xBF,0xFF,0xBF,0xFF,0xDF,0xFF,0xFF,0xFF,0xFF,0xFF,//^62

0xFF,0xFE,0xFF,0xFE,0xFF,0xFE,0xFF,0xFE,0xFF,0xFE,0xFF,0xFE,0xFF,0xFE,0xFF,0xFE,//\_63

0xFF,0xFF,0xBF,0xFF,0xBF,0xFF,0xDF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,//`64

0xFF,0xFF,0xFF,0x67,0xFE,0xDB,0xFE,0xDB,0xFE,0xB7,0xFF,0x03,0xFF,0xFB,0xFF,0xFF,//a65

0xF7,0xFF,0xF0,0x03,0xFF,0x77,0xFE,0xFB,0xFE,0xFB,0xFF,0x77,0xFF,0x8F,0xFF,0xFF,//b66

0xFF,0xFF,0xFF,0x8F,0xFF,0x77,0xFE,0xFB,0xFE,0xFB,0xFE,0xFB,0xFF,0x77,0xFF,0xFF,//c67

0xFF,0xFF,0xFF,0x07,0xFE,0xFB,0xFE,0xFB,0xFE,0xFB,0xF6,0xF7,0xF0,0x03,0xFF,0xFB,//d68

0xFF,0xFF,0xFF,0x07,0xFE,0xDB,0xFE,0xDB,0xFE,0xDB,0xFE,0xDB,0xFF,0x17,0xFF,0xFF,//e69

0xFF,0xFF,0xFE,0xFB,0xFE,0xFB,0xF8,0x03,0xF6,0xFB,0xF6,0xFB,0xFB,0xFF,0xFF,0xFF,//f70

0xFF,0xFF,0xFF,0x29,0xFE,0xD6,0xFE,0xD6,0xFE,0xD6,0xFE,0x36,0xFE,0xF9,0xFF,0xFF,//g71

0xF7,0xFB,0xF0,0x03,0xFF,0x7B,0xFE,0xFF,0xFE,0xFF,0xFE,0xFB,0xFF,0x03,0xFF,0xFB,//h72

0xFF,0xFF,0xFE,0xFB,0xE6,0xFB,0xE6,0x03,0xFF,0xFB,0xFF,0xFB,0xFF,0xFF,0xFF,0xFF,//i73

0xFF,0xFF,0xFF,0xFC,0xFF,0xFE,0xFE,0xFE,0xE6,0xFE,0xE6,0x01,0xFF,0xFF,0xFF,0xFF,//j74

0xF7,0xFB,0xF0,0x03,0xFF,0xDB,0xFF,0x9F,0xFE,0x6B,0xFE,0xF3,0xFE,0xFB,0xFF,0xFF,//k75

0xFF,0xFF,0xF7,0xFB,0xF7,0xFB,0xE0,0x03,0xFF,0xFB,0xFF,0xFB,0xFF,0xFF,0xFF,0xFF,//l76

0xFE,0xFB,0xFE,0x03,0xFE,0xFB,0xFE,0xFF,0xFE,0x03,0xFE,0xFB,0xFE,0xFF,0xFF,0x03,//m77

0xFE,0xFB,0xFE,0x03,0xFF,0x7B,0xFE,0xFF,0xFE,0xFF,0xFE,0xFB,0xFF,0x03,0xFF,0xFB,//n78

0xFF,0xFF,0xFF,0x07,0xFE,0xFB,0xFE,0xFB,0xFE,0xFB,0xFE,0xFB,0xFF,0x07,0xFF,0xFF,//o79

0xFE,0xFE,0xFE,0x00,0xFF,0x76,0xFE,0xFB,0xFE,0xFB,0xFF,0x77,0xFF,0x8F,0xFF,0xFF,//p80

0xFF,0xFF,0xFF,0x8F,0xFF,0x77,0xFE,0xFB,0xFE,0xFB,0xFF,0x76,0xFE,0x00,0xFF,0xFE,//q81

0xFE,0xFB,0xFE,0xFB,0xFE,0x03,0xFF,0x7B,0xFE,0xFB,0xFE,0xFF,0xFE,0x7F,0xFF,0xFF,//r82

0xFF,0xFF,0xFF,0x33,0xFE,0xDB,0xFE,0xDB,0xFE,0xDB,0xFE,0xDB,0xFE,0x67,0xFF,0xFF,//s83

0xFF,0xFF,0xFE,0xFF,0xFE,0xFF,0xF8,0x07,0xFE,0xFB,0xFE,0xFB,0xFF,0xF7,0xFF,0xFF,//t84

0xFE,0xFF,0xFE,0x07,0xFF,0xFB,0xFF,0xFB,0xFF,0xFB,0xFE,0xF7,0xFE,0x03,0xFF,0xFB,//u85

0xFE,0xFF,0xFE,0x3F,0xFE,0xCF,0xFF,0xF3,0xFE,0xCF,0xFE,0x3F,0xFE,0xFF,0xFF,0xFF,//v86

0xFE,0x7F,0xFE,0x8F,0xFF,0xF3,0xFE,0xCF,0xFE,0x1F,0xFF,0xE3,0xFE,0x9F,0xFE,0x7F,//w87

0xFF,0xFF,0xFE,0xFB,0xFE,0x73,0xFE,0x8F,0xFF,0x8B,0xFE,0x73,0xFE,0xFB,0xFF,0xFF,//x88

0xFE,0xFF,0xFE,0x7E,0xFE,0x9E,0xFF,0xE1,0xFF,0xE7,0xFE,0x9F,0xFE,0x7F,0xFE,0xFF,//y89

0xFF,0xFF,0xFE,0x7B,0xFE,0xF3,0xFE,0xCB,0xFE,0xBB,0xFE,0x7B,0xFE,0xF3,0xFF,0xFF,//z90

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x7F,0xC0,0x83,0xBF,0xFD,0xBF,0xFD,//{91

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x00,0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,//|92

0xBF,0xFD,0xBF,0xFD,0xC0,0x83,0xFF,0x7F,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,//}93

0xFF,0xFF,0xBF,0xFF,0x7F,0xFF,0xBF,0xFF,0xBF,0xFF,0xDF,0xFF,0xBF,0xFF,0xFF,0xFF,//~94

};

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**图片字模制作：**

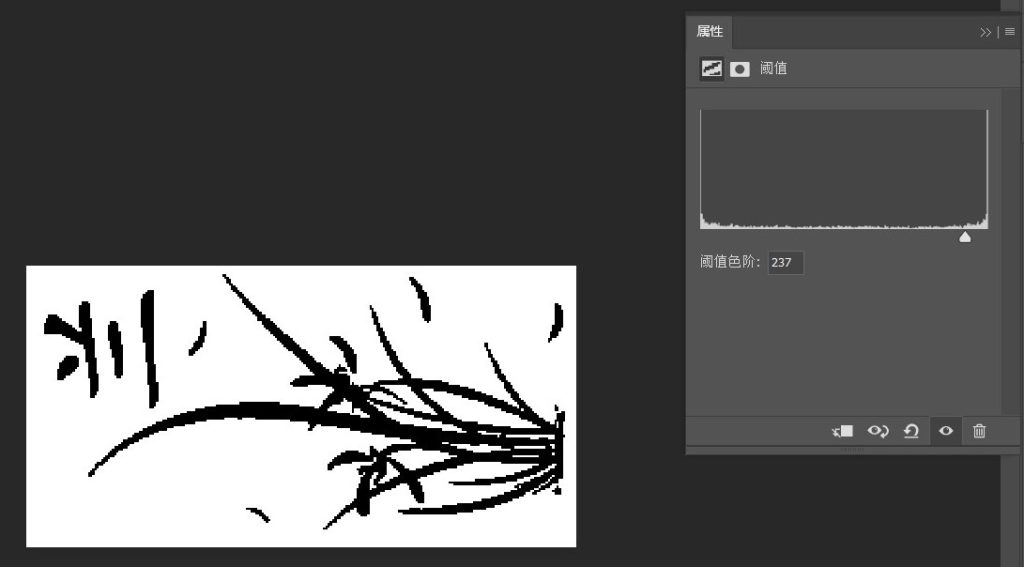
图片可以全屏显示或在固定位置显示，本例只考虑制作全屏显示字模，即尺寸为250\*122。其他尺寸的图片字模制作和代码编写大同小异，这里不再赘述。

首先在ps新建文件并绘制图案：尺寸250\*128像素（由于墨水屏x轴每次输入八位，故扩充为128像素，实际显示时最下面的六排像素点不会显示出来）。

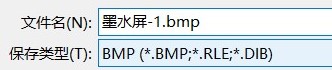


创建调整图层：阙值。调整阙值色阶至合适位置：

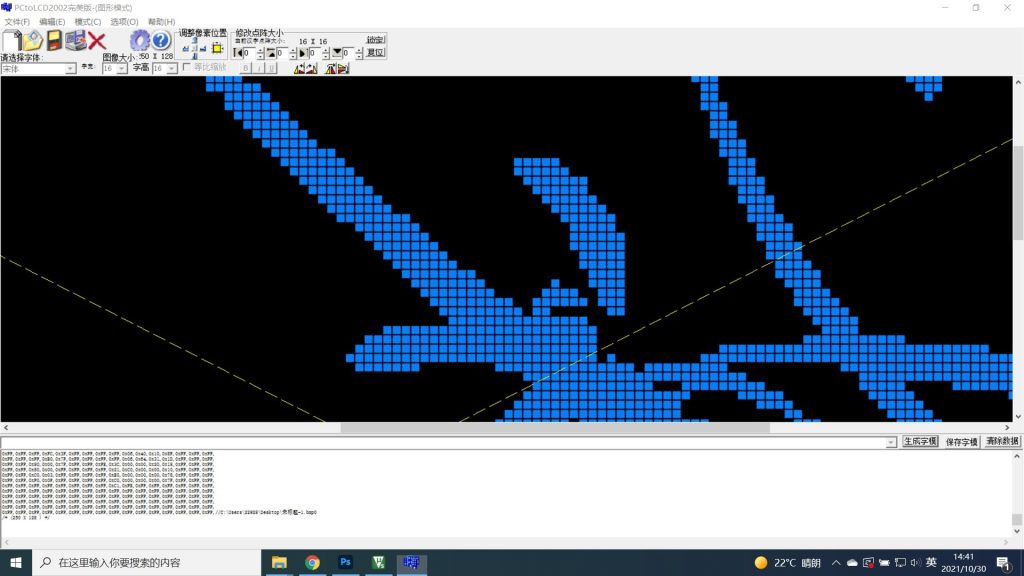




将图片保存为bmp格式：



调整取取模软件为图形模式，打开刚刚保存的图片，点击生成字模：



将生成的字模复制到程序中即可：

