# 7. LCD

http://www.openloongson.org/forum.php?mod=viewthread&tid=141&extra=page%3D1

## 7.1 硬件接口

开发板的底板上电路如下图示。



## 7.2 GPIO口操作函数

LCD模块操作实际上是GPIO口的操作。以前在单片机中，可以简单置位、复位，在linux系统中，操作GPIO的方法有很多，可采用中级篇中的《GPIO硬件编程方法》节，但最直接的方法是寄存器操作。

操作寄存器主要是对GPIO的相关寄存器操作，如图所示。在应用编程中，需要使用mmap系统调用。使用mmap映射文件到进程后,就可以直接操作这段虚拟地址进行文件的读写等操作,不必再调用read,write等系统调用。相关操作参考《9. 内核访问外设I/O资源》。



/\*gpio\_fun.c\*/

#include<stdio.h>

#include<stdlib.h>

#include<stdint.h>

#include<string.h>

#include<unistd.h>

#include<sys/mman.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h>

#define GPIO\_CFG0 0xC0

#define GPIO\_EN0 0xD0

#define GPIO\_IN0 0xE0

#define GPIO\_OUT0 0xF0

#define CBUS\_FIRST0 0x1C0

unsigned char \* map\_base;

int fd;

int gpio\_open(void)

{

fd = open("/dev/mem", O\_RDWR|O\_SYNC);

if (fd == -1) {

return (-1);

}

/\* °Ñ 0x1fd01000 ¿ªÊŒ0x1000×ÖœÚ£¬Ó³Éäµœmap\_base \*/

map\_base = mmap(NULL, 0x1000, PROT\_READ|PROT\_WRITE, MAP\_SHARED, fd, 0x1fd01000);

if (map\_base == 0) {

printf("NULL pointer!\n");

}

return 0;

}

void gpio\_close(void)

{

close(fd);

munmap(map\_base, 0x1000);

}

void gpio\_cfg\_init(int gpio\_number, int function)

{

uint16\_t addr\_offset;

uint16\_t bit\_offset;

int index;

uint32\_t data\_temp;

index = gpio\_number / 32;

bit\_offset = gpio\_number % 32;

addr\_offset = GPIO\_CFG0 + index\*4;

data\_temp = \*(volatile unsigned long \*)(map\_base+addr\_offset);

if(function == 0)

{

data\_temp = data\_temp & (~(1<<bit\_offset));//clr

}

else

{

data\_temp = data\_temp | ((1<<bit\_offset));//set

}

\*(volatile unsigned long \*)(map\_base+addr\_offset)=data\_temp;

}

void gpio\_en\_init(int gpio\_number, int function)

{

uint16\_t addr\_offset;

uint16\_t bit\_offset;

int index;

uint32\_t data\_temp;

index = gpio\_number / 32;

bit\_offset = gpio\_number % 32;

addr\_offset = GPIO\_EN0 + index\*4;

data\_temp = \*(volatile unsigned long \*)(map\_base+addr\_offset);

if(function == 0)

{

data\_temp = data\_temp & (~(1<<bit\_offset));//clr

}

else

{

data\_temp = data\_temp | ((1<<bit\_offset));//set

}

\*(volatile unsigned long \*)(map\_base+addr\_offset)=data\_temp;

}

void gpio\_func\_init(int gpio\_number, int function)

{

uint16\_t addr\_offset;

uint16\_t bit\_offset;

int index,i;

uint32\_t data\_temp;

index = gpio\_number / 32;

bit\_offset = gpio\_number % 32;

addr\_offset = CBUS\_FIRST0 + index\*4;

data\_temp = \*(volatile unsigned long \*)(map\_base+addr\_offset);

for(i=1;i<6;i++)

{

data\_temp = \*(volatile unsigned long \*)(map\_base+addr\_offset+(i-1)\*0x10);

if(i == function)//set

{

data\_temp = data\_temp | ((1<<bit\_offset));//set

}

else//clr

{

data\_temp = data\_temp & (~(1<<bit\_offset));//clr

}

\*(volatile unsigned long \*)(map\_base+addr\_offset)=data\_temp;

}

}

void gpio\_set(int gpio\_number)

{

uint16\_t addr\_offset;

uint16\_t bit\_offset;

int index;

uint32\_t data\_temp;

index = gpio\_number / 32;

bit\_offset = gpio\_number % 32;

addr\_offset = GPIO\_OUT0 + index\*4;

data\_temp = \*(volatile unsigned long \*)(map\_base+addr\_offset);

{

data\_temp = data\_temp | ((1<<bit\_offset));//set

}

\*(volatile unsigned long \*)(map\_base+addr\_offset)=data\_temp;

}

uint32\_t gpio\_get(int gpio\_number)

{

uint16\_t addr\_offset;

uint16\_t bit\_offset;

int index;

uint32\_t data\_temp;

index = gpio\_number / 32;

bit\_offset = gpio\_number % 32;

addr\_offset = GPIO\_IN0 + index\*4;

data\_temp = \*(volatile unsigned long \*)(map\_base+addr\_offset);

return (data\_temp >> bit\_offset)&0x01;

}

void gpio\_clr(int gpio\_number)

{

uint16\_t addr\_offset;

uint16\_t bit\_offset;

int index;

uint32\_t data\_temp;

index = gpio\_number / 32;

bit\_offset = gpio\_number % 32;

addr\_offset = GPIO\_OUT0 + index\*4;

data\_temp = \*(volatile unsigned long \*)(map\_base+addr\_offset);

{

data\_temp = data\_temp & (~(1<<bit\_offset));//clr

}

\*(volatile unsigned long \*)(map\_base+addr\_offset)=data\_temp;

}

uint32\_t data\_get(int gpio\_start\_number, int len)

{

uint16\_t addr\_offset;

uint16\_t bit\_offset;

int index,i;

uint32\_t data\_temp;

index = gpio\_start\_number / 32;

bit\_offset = gpio\_start\_number % 32;

addr\_offset = GPIO\_IN0 + index\*4;

if((bit\_offset+len)>32)

{

len = 32 - bit\_offset;

}

data\_temp = \*(volatile unsigned long \*)(map\_base+addr\_offset);

for( i=bit\_offset+len;i<32;i++)

{

data\_temp = data\_temp & (~(1<<i));//clr

}

return data\_temp>>bit\_offset;

}

void data\_write(int gpio\_start\_number, int len ,uint32\_t data)

{

uint16\_t addr\_offset;

uint16\_t bit\_offset;

int index,i;

uint32\_t data\_temp;

index = gpio\_start\_number / 32;

bit\_offset = gpio\_start\_number % 32;

addr\_offset = GPIO\_OUT0 + index\*4;

if((bit\_offset+len)>32)

{

len = 32 - bit\_offset;

}

data\_temp = \*(volatile unsigned long \*)(map\_base+addr\_offset);

{

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

{

if(0x00 == (0x01&(data>>i))){

data\_temp = data\_temp & (~(1<<(bit\_offset+i)));//clr

}

else{

data\_temp = data\_temp | (1<<(bit\_offset+i)) ;//set

}

}

}

\*(volatile unsigned long \*)(map\_base+addr\_offset)=data\_temp;

}

void gpio\_cfg\_data(int gpio\_start\_number, int len ,uint32\_t data)

{

uint16\_t addr\_offset;

uint16\_t bit\_offset;

int index,i;

uint32\_t data\_temp;

index = gpio\_start\_number / 32;

bit\_offset = gpio\_start\_number % 32;

addr\_offset = GPIO\_CFG0 + index\*4;

if((bit\_offset+len)>32)

{

len = 32 - bit\_offset;

}

data\_temp = \*(volatile unsigned long \*)(map\_base+addr\_offset);

{

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

{

if(0x00 == data){

data\_temp = data\_temp & (~(1<<(bit\_offset+i)));//clr

}

else if(0x01 == data){

data\_temp = data\_temp | (1<<(bit\_offset+i)) ;//set

}

}

}

\*(volatile unsigned long \*)(map\_base+addr\_offset)=data\_temp;

}

void gpio\_en\_data(int gpio\_start\_number, int len ,uint32\_t data)

{

uint16\_t addr\_offset;

uint16\_t bit\_offset;

int index,i;

uint32\_t data\_temp;

index = gpio\_start\_number / 32;

bit\_offset = gpio\_start\_number % 32;

addr\_offset = GPIO\_EN0 + index\*4;

if((bit\_offset+len)>32)

{

len = 32 - bit\_offset;

}

data\_temp = \*(volatile unsigned long \*)(map\_base+addr\_offset);

{

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

{

if(0x00 == data){

data\_temp = data\_temp & (~(1<<(bit\_offset+i)));//clr

}

else if(0x01 == data){

data\_temp = data\_temp | (1<<(bit\_offset+i)) ;//set

}

}

}

\*(volatile unsigned long \*)(map\_base+addr\_offset)=data\_temp;

}

/\* Usage:

\* ./gpio\_fun cfg [gpio\_num] [0] // 用法

\* ./gpio\_fun en [gpio\_num] [0]

\* ./gpio\_fun out [gpio\_num] [0]

\* ./gpio\_fun in [gpio\_num] [0]

\* ./gpio\_fun dataout [gpio\_num] [len] data//len bit

\* ./gpio\_fun datain [gpio\_num] [len] data// len bit

\* ./gpio\_fun cfgdata [gpio\_num] [len] 0/1// 8 bit

\* ./gpio\_fun endata [gpio\_num] [len] 0/1// 8 bit

\* ./gpio\_fun function [gpio\_num] functionnum

\*/

int main(int argc, char \*argv[])

{

unsigned int buf[3];

if(argc<3) {

printf("gpio\_fun cfg/en/out/in 50 0\r\n");

printf("gpio\_fun dataout 58 6 85\r\n");

printf("gpio\_fun datain 58 6\r\n");

printf("gpio\_fun cfgdata 58 6 0/1 \r\n");

printf("gpio\_fun endata 58 6 0/1 \r\n");

printf("gpio\_fun function 4 85 ;set gpio85 to function 4 \r\n");

return (-2);

}

if(gpio\_open()<0)

return (-1);

/\* gpio\_num \*/

buf[0] = strtoul(argv[2], NULL, 0);

/\* function \*/

if(argc>=4)

{

buf[1] = strtoul(argv[3], NULL, 0);

}

/\* data \*/

if(argc==5)

{

buf[2] = strtoul(argv[4], NULL, 0);

}

if (strcmp(argv[1], "cfg") == 0)

{

gpio\_cfg\_init(buf[0], buf[1]);

}

else if (strcmp(argv[1], "en") == 0)

{

gpio\_en\_init(buf[0], buf[1]);

}

else if (strcmp(argv[1], "out") == 0)

{

if(1 == buf[1]){

gpio\_set(buf[0]);

}

else if(0 == buf[1]){

gpio\_clr(buf[0]);

}

}

else if (strcmp(argv[1], "in") == 0)

{

printf("gpio%d = %d\r\n",buf[0], gpio\_get(buf[0]) );

}

else if (strcmp(argv[1], "dataout") == 0)

{

if(argc>=5)

{

data\_write(buf[0],buf[1],buf[2]);

}

else

{

printf("gpio\_fun argc<5 !\r\n" );

}

}

else if (strcmp(argv[1], "datain") == 0)

{

printf("gpio%d read %d bit= 0x%02x\r\n",buf[0], buf[1], data\_get(buf[0],buf[1]));

}

else if (strcmp(argv[1], "cfgdata") == 0)

{

if(argc>=5)

{

gpio\_cfg\_data(buf[0],buf[1],buf[2]);

}

else

{

printf("gpio\_fun argc<5 !\r\n" );

}

}

else if (strcmp(argv[1], "endata") == 0)

{

if(argc>=5)

{

gpio\_en\_data(buf[0],buf[1],buf[2]);

}

else

{

printf("gpio\_fun argc<5 !\r\n" );

}

}

else if (strcmp(argv[1], "function") == 0)

{

if(argc>=4)

{

gpio\_func\_init(buf[0],buf[1]);

}

else

{

printf("gpio\_fun argc<4 !\r\n" );

}

}

gpio\_close();

return (0);

}

代码中，main函数测试用，测试成功后，将main改成其它名字。

## 7.3 LCD操作

这里使用LCD是来自于

<https://item.taobao.com/item.htm?spm=a1z10.5-c.w4002-1891666121.32.hqEYu4&id=25142720480>

使用控制器NT35310 ，屏幕大小为480\*320。

直接在驱动例程上进行修改，将GPIO口的操作换成7.3节的GPIO口操作的函数，还延时函数的替换。代码太长，这里不再列出。这里只说明初始化过程。

首先从0x00处读取ID，如果读不到，再从0xD3处读判断是否9341, 如果不是再从0xBF读判断是否6804，如果不是再从0xD4判断是否35310，如果不是再从0XDA00读判断是否NT35510，如果不是再从0XA1读判断是否SSD1963。

lcddev.id = LCD\_ReadReg(0x0000);

if(lcddev.id<0XFF||lcddev.id==0XFFFF||lcddev.id==0X9300)//读到ID不正确,新增lcddev.id==0X9300判断，因为9341在未被复位的情况下会被读成9300

{

//尝试9341 ID的读取

LCD\_WR\_REG(0XD3);

LCD\_RD\_DATA(); //dummy read

LCD\_RD\_DATA(); //读到0X00

lcddev.id=LCD\_RD\_DATA(); //读取93

lcddev.id<<=8;

lcddev.id|=LCD\_RD\_DATA(); //读取41

if(lcddev.id!=0X9341) //非9341,尝试是不是6804

{

LCD\_WR\_REG(0XBF);

LCD\_RD\_DATA(); //dummy read

LCD\_RD\_DATA(); //读回0X01

LCD\_RD\_DATA(); //读回0XD0

lcddev.id=LCD\_RD\_DATA();//这里读回0X68

lcddev.id<<=8;

lcddev.id|=LCD\_RD\_DATA();//这里读回0X04

if(lcddev.id!=0X6804) //也不是6804,尝试看看是不是NT35310

{

LCD\_WR\_REG(0XD4);

LCD\_RD\_DATA(); //dummy read

LCD\_RD\_DATA(); //读回0X01

lcddev.id=LCD\_RD\_DATA(); //读回0X53

lcddev.id<<=8;

lcddev.id|=LCD\_RD\_DATA(); //这里读回0X10

if(lcddev.id!=0X5310) //也不是NT35310,尝试看看是不是NT35510

{

LCD\_WR\_REG(0XDA00);

LCD\_RD\_DATA(); //读回0X00

LCD\_WR\_REG(0XDB00);

lcddev.id=LCD\_RD\_DATA();//读回0X80

lcddev.id<<=8;

LCD\_WR\_REG(0XDC00);

lcddev.id|=LCD\_RD\_DATA();//读回0X00

if(lcddev.id==0x8000)lcddev.id=0x5510;//NT35510读回的ID是8000H,为方便区分,我们强制设置为5510

if(lcddev.id!=0X5510) //也不是NT5510,尝试看看是不是SSD1963

{

LCD\_WR\_REG(0XA1);

lcddev.id=LCD\_RD\_DATA();

lcddev.id=LCD\_RD\_DATA(); //读回0X57

lcddev.id<<=8;

lcddev.id|=LCD\_RD\_DATA(); //读回0X61

if(lcddev.id==0X5761)lcddev.id=0X1963;//SSD1963读回的ID是5761H,为方便区分,我们强制设置为1963

}

}

}

}

}

printf(" LCD ID:%x\r\n",lcddev.id); //打印LCD ID

最终运行结果为：LCD控制器为NT35310。

[root@Loongson:/app]#./testlcd

Init LCD

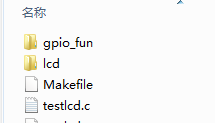
LCD ID:5310

DRIVER GET LCD ID:0x5310

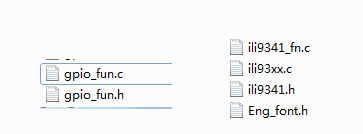
ili9341.c-Init\_ili9341():ILI9341 init Successful!

## 7.4 编写Makefile

本应用程序中使用的文件结构如下图示。



其中gpio\_fun中为GPIO口的操作函数。Lcd中为LCD的驱动。



下面编写通用的Makefile，指明了头文件的位置，目标文件，并指定输出文件testlcd。

# 指令编译器和选项

CC=mipsel-linux-gcc

CFLAGS=-Wall -std=gnu99 -Wstrict-prototypes

# 目标文件

TARGET=testlcd

SRCS = testlcd.c \

./gpio\_fun/gpio\_fun.c \

./lcd/ili9341\_fn.c\

./lcd/ili93xx.c

INC = -I./gpio\_fun -I./lcd

OBJS = $(SRCS:.c=.o)

$(TARGET):$(OBJS)

@echo TARGET:$@

@echo OBJECTS:$^

$(CC) -o $@ $^

clean:

rm -rf $(TARGET) $(OBJS)

%.o:%.c

$(CC) $(CFLAGS) $(INC) -o $@ -c $<

## 7.5 代码及运行结果

### 1)显示字符

LCD\_Clear(BLUE); //设置背景色

LCD\_ShowString(30,40,400,24,24,"OPEN LOONGSON Smart Loong V2.0^\_^");

LCD\_ShowString(30,70,200,16,16,"TFTLCD TEST ");

LCD\_ShowString(30,90,200,16,16,"sundm75");

LCD\_ShowString(30,110,200,16,16,lcd\_id); //显示LCD ID

LCD\_ShowString(30,130,200,12,12,"2017/5/24");

### 2）显示图片

用Image2Lcd转换图片。

（1）设置输出数据格式 C语言数组

（2）设置输出灰度 16位真彩色

（3）设置最大宽度高度不得大于LCD的宽、高（320、480）。

（4）最后数据头文件picture.h。



代码如下：

/\* 画 picture \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名 称：void lcd\_draw\_picture(u16 StartX,u16 StartY,u16 EndX,u16 EndY,u16 \*picture)

\* 功 能：在指定座标范围显示一副图片

\* 入口参数：StartX 行起始座标

\* StartY 列起始座标

\* EndX 行结束座标

\* EndY 列结束座标

picture 图片头指针

\* 出口参数：无

\* 说 明：图片取模格式为水平扫描，16位颜色模式 (RGB565)

\* 调用方法：lcd\_draw\_picture(0,0,100,100,(u16\*)demo);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void lcd\_draw\_picture(u16 start\_x, u16 start\_y, u16 end\_x, u16 end\_y, u16 \*picture)

{

u32 length;

u32 i;

u16 x, y;

x = start\_x;

y = start\_y;

length = (end\_x - start\_x + 1) \* (end\_y - start\_y + 1) ;

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

{

LCD\_Fast\_DrawPoint(x, y, \*picture++);

y++;

if(y > end\_y)

{

x++;

y = start\_y;

}

}

}

/\*-------------测试LCD函数，在当前LCD屏幕中心画出PictureAddr图片-------------\*/

void DrawPicture\_Center(u16 \*PictureAddr)

{

u16 PictureWidth;

u16 PictureHeight;

u16 \* picuturepoint;

picuturepoint=PictureAddr;

PictureHeight = picuturepoint[1];

PictureWidth = picuturepoint[2];

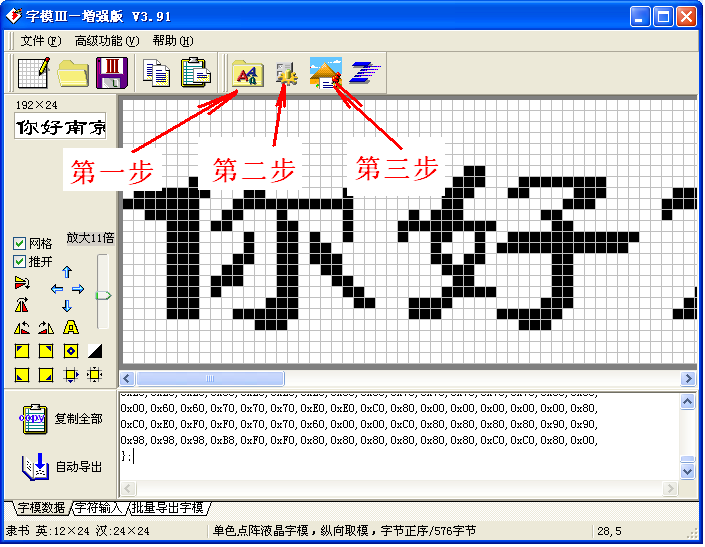
lcd\_draw\_picture((480-PictureWidth+1)/2, (320-PictureHeight+1)/2, ((480+PictureWidth+1)/2)-1,

((320+PictureHeight+1)/2)-1, (u16 \*)(PictureAddr + 3));

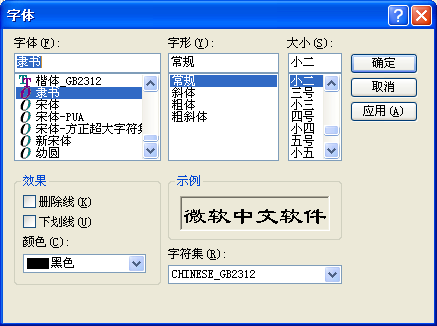
}

### 3）显示汉字

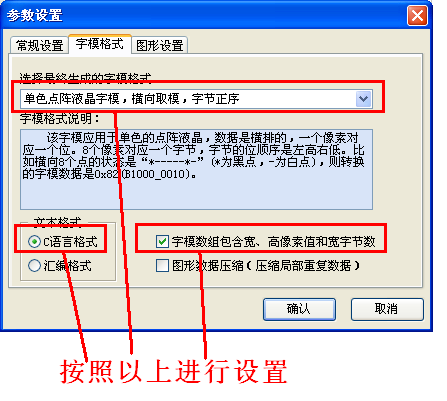
用 字模III增强版 ,生成字模的汉字库。



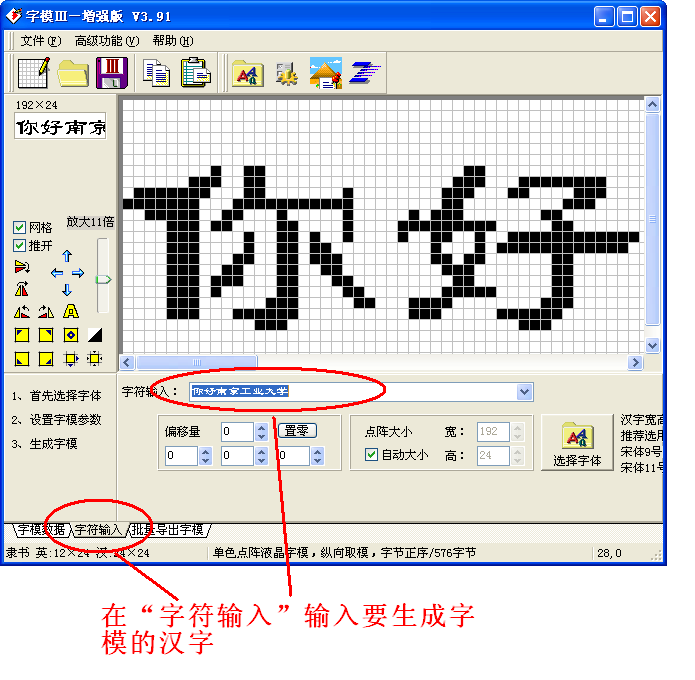
第一步



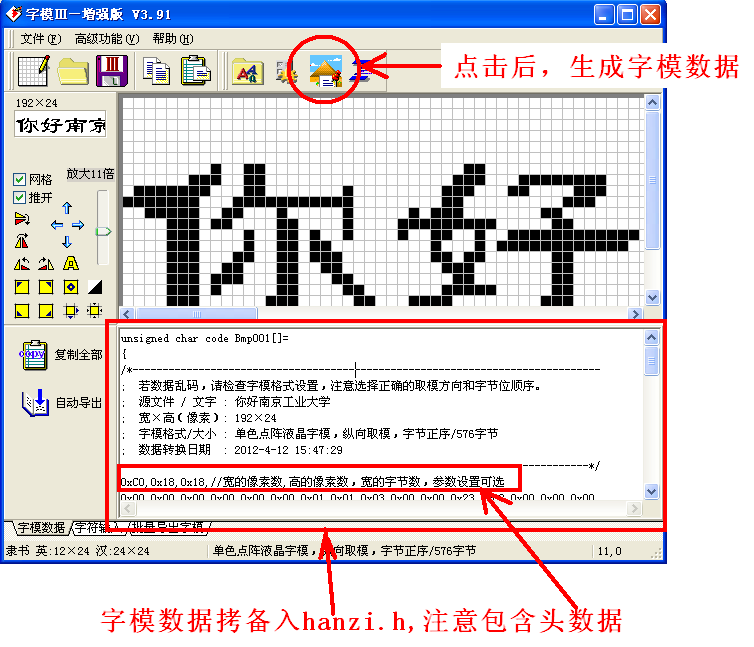
第二步



汉字显示软件操作：



第三步，生成字模数据



代码如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名 称：void lcd\_draw\_hanzi(u16 x,u16 y,u8 \*hanziaddr,u16 charColor,u16 bkColor)

\* 功 能：在指定座标显示一个由字模软件生成的汉字字符

\* 入口参数：x 行座标

\* y 列座标

hanziaddr 汉字头指针

\* charColor 字符的颜色

\* bkColor 字符背景颜色

\* 出口参数：无

\* 说 明：显示范围限定为汉字头文件中，用横向取模的单色点阵液晶字模，横向取模，字节正序

\* 调用方法：lcd\_draw\_hanzi(10,10,(u8\*)Str\_table1,White, HYALINE);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void lcd\_draw\_hanzi(u16 x,u16 y,u8 \*hanziaddr,u16 charColor,u16 bkColor) //

{

u16 i=0;

u16 j=0;

u16 m=0;

u8 width,high, wbyte;

u8 \*hanzipoint = hanziaddr;

u8 tmp\_char = 0;

width = \*hanzipoint++;

high = \*hanzipoint++;

wbyte = \*hanzipoint++;

if(HYALINE == bkColor)

{

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

{

for (j=0;j<wbyte;j++)

{

tmp\_char=\*hanzipoint++;

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

{

if ( (tmp\_char >>7-m) & 0x01 == 0x01)

{

POINT\_COLOR = charColor;// 字符颜色

LCD\_DrawPoint(x+j\*8+m,y+i);

}

else

{

// do nothing // 透明背景

}

}

}

}

}

else

{

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

{

for (j=0;j<wbyte;j++)

{

tmp\_char=\*hanzipoint++;

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

{

if ( (tmp\_char >>7-m) & 0x01 == 0x01)

{

POINT\_COLOR = charColor;// 字符颜色

LCD\_DrawPoint(x+j\*8+m,y+i);

}

else

{

POINT\_COLOR = bkColor;// 背景颜色

LCD\_DrawPoint(x+i,y+i-8);

}

}

}

}

}

}

### 4）画线、圆圈

LCD\_DrawLine(20,20, 20,300); //画线

LCD\_DrawLine(410,30, 410,290); //画线

LCD\_DrawRectangle(3,15,450,310); //画矩形

LCD\_ShowString(30,40,400,24,24,"Picture 214 X 300");//写字符

主程序testlcd.c：

int main(int argc,char \* argv[])

{

u8 x=0;

u8 lcd\_id[64]; //存放LCD ID字符串

gpio\_open();

Init\_ili9341();

POINT\_COLOR=RED;

sprintf((char\*)lcd\_id,"LCD ID:%04X",lcddev.id);//将LCD ID打印到lcd\_id数组。

while(1)

{

switch(x)

{

case 0:LCD\_Clear(WHITE);

LCD\_Display\_Dir(1); //横屏

POINT\_COLOR = BLACK;

BACK\_COLOR = WHITE;

LCD\_ShowString(30,40,400,24,24,"OPEN LOONGSON Smart Loong V2.0^\_^");

LCD\_ShowString(30,70,200,16,16,"TFTLCD TEST ");

LCD\_ShowString(30,90,200,16,16,"sundm75");

LCD\_ShowString(30,110,200,16,16,lcd\_id); //显示LCD ID

LCD\_ShowString(30,130,200,12,12,"2017/5/24");

break;

case 1://LCD\_Clear(BLACK);

POINT\_COLOR = WHITE;

BACK\_COLOR = BLACK;

LCD\_ShowString(30,40,400,24,24,"OPEN LOONGSON Smart Loong V2.0^\_^");

LCD\_ShowString(30,70,200,16,16,"TFTLCD TEST ");

LCD\_ShowString(30,90,200,16,16,"sundm75");

LCD\_ShowString(30,110,200,16,16,lcd\_id); //显示LCD ID

LCD\_ShowString(30,130,200,12,12,"2017/5/24");

break;

case 2://LCD\_Clear(BLUE);

POINT\_COLOR = RED;

BACK\_COLOR = BLUE;

LCD\_ShowString(30,40,400,24,24,"OPEN LOONGSON Smart Loong V2.0^\_^");

LCD\_ShowString(30,70,200,16,16,"TFTLCD TEST ");

LCD\_ShowString(30,90,200,16,16,"sundm75");

LCD\_ShowString(30,110,200,16,16,lcd\_id); //显示LCD ID

LCD\_ShowString(30,130,200,12,12,"2017/5/24");

break;

case 3://LCD\_Clear(RED);

POINT\_COLOR = BLUE;

BACK\_COLOR = RED;

LCD\_ShowString(30,40,400,24,24,"OPEN LOONGSON Smart Loong V2.0^\_^");

LCD\_ShowString(30,70,200,16,16,"TFTLCD TEST ");

LCD\_ShowString(30,90,200,16,16,"sundm75");

LCD\_ShowString(30,110,200,16,16,lcd\_id); //显示LCD ID

LCD\_ShowString(30,130,200,12,12,"2017/5/24");

break;

case 4:

LCD\_Display\_Dir(0); //竖屏

LCD\_Clear(BLUE);

lcd\_draw\_hanzi(5,420,(u8\*)Hanzi, WHITE, HYALINE);

LCD\_Display\_Dir(1); //横屏

DrawPicture\_Center(gImage\_picture);

msleep(500);

break;

case 5:

LCD\_Display\_Dir(1); //横屏

POINT\_COLOR = RED;

BACK\_COLOR = BLUE;

LCD\_DrawLine(20,20, 20,300);

POINT\_COLOR = GREEN;

LCD\_DrawLine(405,30, 405,290);

LCD\_DrawRectangle(3,15,450,310);

LCD\_Display\_Dir(0); //竖屏

LCD\_ShowString(30,45,400,29,24,"Smart Loong V2.0");

LCD\_ShowString(100,20,400,4,24,"Loongson");

POINT\_COLOR = RED;

LCD\_Draw\_Circle(30,15,10);

LCD\_Draw\_Circle(50,15,10);

LCD\_Draw\_Circle(70,15,10);

LCD\_Draw\_Circle(90,15,10);

LCD\_Draw\_Circle(110,15,10);

LCD\_Draw\_Circle(130,15,10);

LCD\_Draw\_Circle(150,15,10);

LCD\_Draw\_Circle(180,15,10);

LCD\_Draw\_Circle(200,15,10);

LCD\_Draw\_Circle(220,15,10);

LCD\_Draw\_Circle(240,15,10);

LCD\_Draw\_Circle(260,15,10);

LCD\_Draw\_Circle(280,15,10);

LCD\_Draw\_Circle(300,15,10);

msleep(800);

break;

}

x++;

if(x==6)x=0;

msleep(300);

}

gpio\_close();

return 0;

}

程序运行结果：

