Chương 3

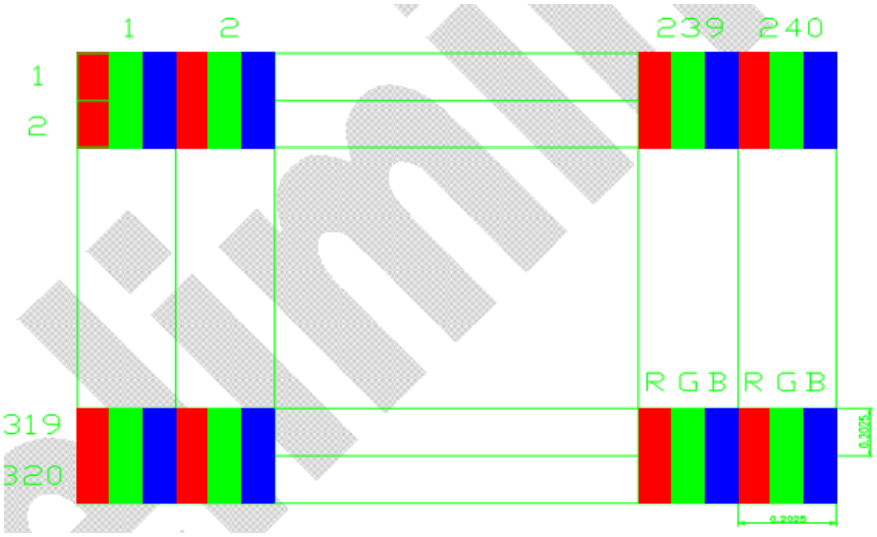
CÁC BÀI THỰC HÀNH ARM CORTEX – M3 STM32F103VET GIAO TIẾP GLCD TFT 3.2

- **GIỚI THIỆU**
- **MẠCH GIAO TIẾP VI ĐIỀU KHIỂN VỚI GLCD**
- **CÁC CHƯƠNG TRÌNH ĐIỀU KHIỂN GLCD**
  CÁC BÀI THỰC HÀNH HIỂN THỊ KÝ TỰ, VẼ HÌNH TRÊN GLCD
  CÁC BÀI THỰC HÀNH HIỂN THỊ HÌNH ẢNH TRÊN GLC
- **CÁC HÀM ĐIỀU KHIỂN GLCD TRONG THƯ VIỆN <TV_GLCD_ILI932X.H>**

I. GIỚI THIỆU

Chương này trình bày các hàm thư viện để sử dụng cho lập trình phần GLCD và touch screen.

Màn hình GLCD có kích thước dài và rộng là 320×240.



Hình 3-1. Kích thước màn hình GLCD.

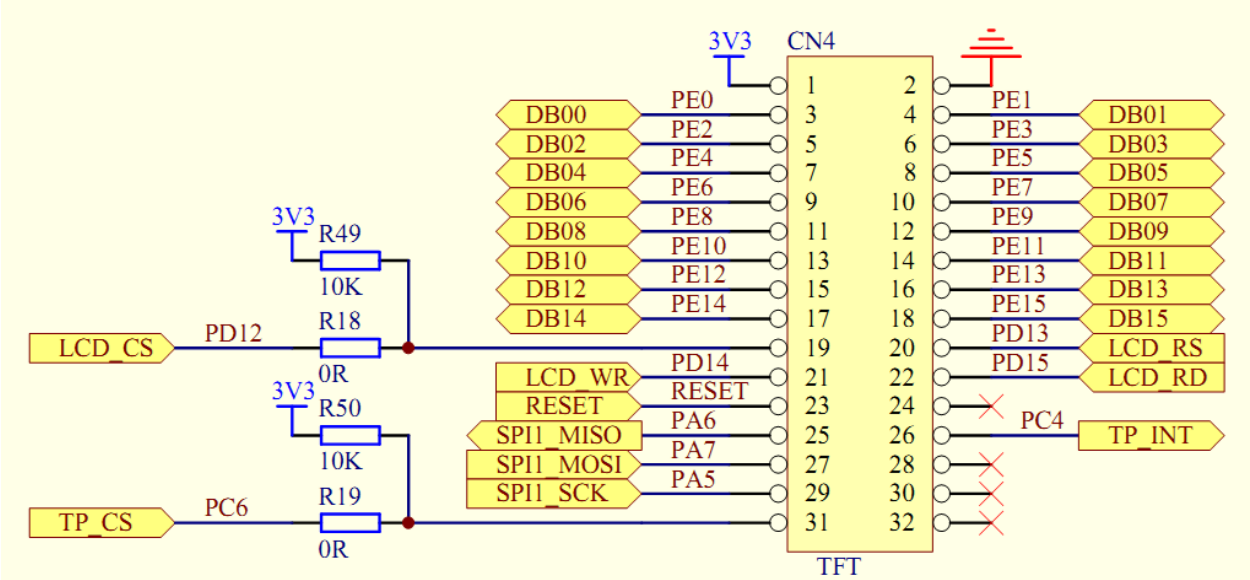
Tọa độ điểm đầu tiên là (1, 1) và tọa độ điểm cuối cùng là (240, 320).

Trong lập trình thường thì là (0,0) cho đến (239, 319).

Tổng quát tọa độ (x, y) thì x tính theo chiều rộng, y tính theo chiều dài, x có giới hạn từ 0 đến 239, y có giới hạn từ 0 đến 319.

II. MẠCH GIAO TIẾP VI ĐIỀU KHIỂN VỚI GLCD

Sơ đồ mạch giao tiếp như hình 3-2



Hình 3-2. Sơ đồ mạch vi điều khiển ARM giao tiếp GLCD.

Chip sử dụng điều khiển màn hình GLCD 3.2 inch có tên là IC SSD1289 có mã thiết bị là 0x8999.

Hình 3-2 là socket giao tiếp với màn hình GLCD TFT, vi điều khiển sử dụng port E có 16 bit để giao tiếp, các bit điều khiển LCD_CS, LCD_RS, LCD_WR, LCD_RD tương ứng với các bit PD12, PD13, PD14, PD15.

Chú ý tín hiệu LCD_CS có điện trở kéo lên và tích cực mức 0.

Các tín hiệu còn lại là giao tiếp màn cảm ứng touch.

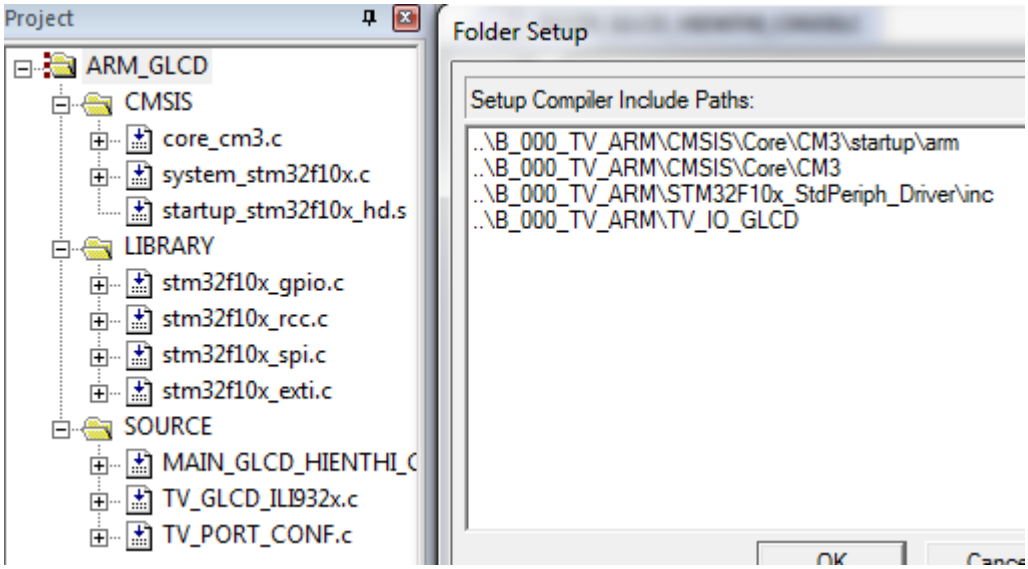
III. CÁC CHƯƠNG TRÌNH ĐIỀU KHIỂN GLCD

Phần này thực hành các bài điều khiển GLCD hiển thị các thông tin text và hình ảnh.

1. CÁC BÀI THỰC HÀNH HIỂN THỊ KÝ TỰ, VẼ HÌNH TRÊN GLCD

Bài mẫu 301. Chương trình điều khiển GLCD 320×240 hiển thị chuỗi trên màn hình. Tạo thư mục "BAI_301_GLCD_HT_CHUOI" để lưu project.

a. Mục đích : làm quen với các hàm điều khiển hiển thị text trên GLCD.
b. Lưu đồ: khởi tạo port, GLCD, hiển thị các thông tin.
c. Hình các file chính và thư viện:



Hình 3-3. Các nhóm lưu thư mục và đường dẫn bài 301.

d. Chương trình:

```
#include "stm32f10x.h"
#include "hardware_conf.h"
#include "TV_PORT_CONF.h"
#include "TV_GLCD_ILI932x.h"
int main(void)
{
        SystemInit();   PORT_CONF();    LCD_INIT();
        LCD_CLEAR(YELLOW);   BACK_COLOR=YELLOW;   POINT_COLOR=RED;

        LCD_SHOW_STRING_X(10,0,"DAI HOC SU PHAM KY THUAT");
        LCD_SHOW_STRING_X(10,20,"THANH PHO HO CHI MINH");
        LCD_SHOW_STRING_X(10,40,"KHOA DIEN - DIEN TU");
```

```
                LCD_SHOW_STRING_X(10,60,"BO MON DIEN TU CONG NGHIEP");
                LCD_SHOW_STRING_X(10,80,"VI DIEU KHIEN ARM STM32");
                LCD_SHOW_STRING_X(10,100,"BAT DAU MOT THACH THUC MOI");
                LCD_SHOW_STRING_X(10,120,"0903 982 443 SDSDLSLDKLKDLSKDLSKDLSKD");
                while(1)
                {
                }
}
```

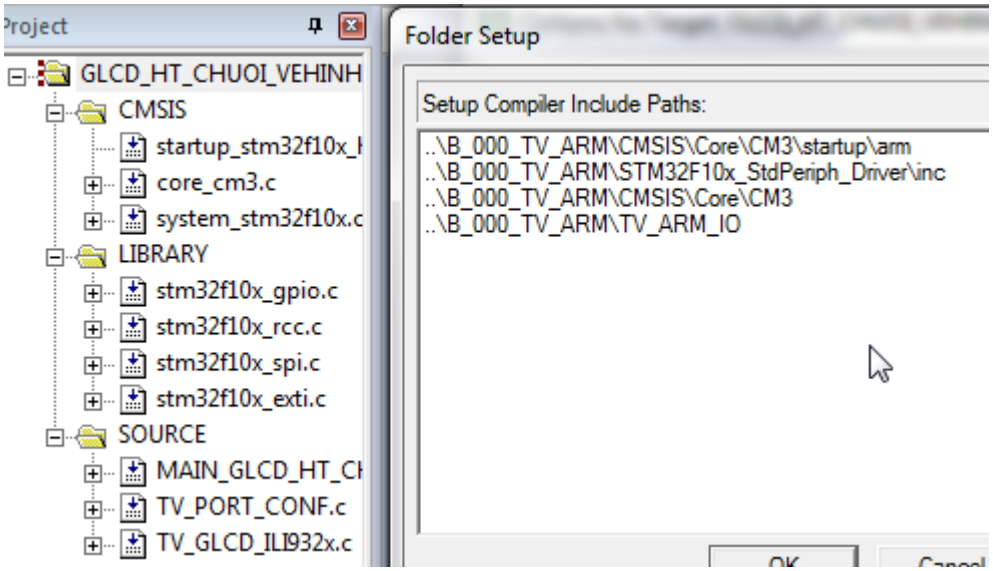
e. Tiến hành biên dịch và nạp.
f. Quan sát kết quả:
g. Giải thích chương trình: chương trình chính tiến hành khởi tạo hệ thống và khởi tạo port giống như các bài trước. Khởi tạo LCD, xóa màn hình bằng màu vàng, khởi tạo màu nền, khởi tạo màu chữ, tiến hành thực hiện các hàm hiển thị chuỗi. Chuỗi bắt đầu tại vị trí 10 theo trục x, trục y thì hàng đầu tiên là 0, hàng tiếp theo cách hàng thứ nhất 20 pixel. Mỗi ký tự dùng 16 pixel.
h. Cho phép thay đổi: bạn có thể thay đổi vị trí nằm trong giới hạn, thay đổi kích thước chuỗi, thay đổi màu nền, thay đổi màu led. Xem tên các màu ở hàm xóa màn hình.
i. Giải thích các chương trình trong thư viện: Phần này giải thích các hàm sử dụng liên quan đến GLCD.
j. Đọc và giải thích nguyên lý các hàm trong thư viện GLCD ở cuối chương mới tiếp tục thực hiện các bài tiếp theo.

TT	LỆNH	GIẢI THÍCH
1	#include "TV_GLCD_ILI932x.h"	Khai báo 4 file thư viện .h và kèm theo là .c của GLCD, trong thư viện này khai báo các định nghĩa phần cứng giao tiếp giữa vi điều khiển ARM và IC điều khiển GLCD, khai báo các thanh ghi hay địa chỉ của IC điều khiển GLCD, khai báo màu, ...
2	*Chương trình chính gọi hàm* LCD_INIT();	*Hàm này viết trong thư viện "TV_GLCD_ILI932x.c"* *Chức năng: cấu hình cho các port của ARM giao tiếp IC điều khiển GLCD, khởi tạo GLCD* ***Xem chi tiết bảng theo sau.***
	Lệnh: BACK_COLOR=YELLOW; POINT_COLOR=RED;	*Gán màu nền là màu vàng* *Gán màu chữ là màu đỏ*
3	*Chương trình chính tiếp tục gọi chương trình con:* LCD_ SHOW_STRING_X(10, 0,"DAI HOC SU PHAM KY THUAT");	*Hàm này viết trong thư viện "TV_GLCD_ILI932x.c"* *Chức năng: hiển thị chuỗi tại tọa độ (X,Y) và tiếp theo là chuỗi.* ***Xem chi tiết bảng theo sau.***
4	*Các hàm còn lại tương tự*	

Bài mẫu 302. Chương trình điều khiển GLCD 320×240 hiển thị chuỗi trên màn hình, vẽ hình tròn, vẽ hình vuông, tô đầy 1 hình vuông và hiển thị chữ trong hình vuông, hiển thị số.

Tạo thư mục "BAI_302_GLCD_VEHINH" để lưu project.

a. Mục đích : làm quen với các hàm điều khiển vẽ hình trên GLCD.
b. Lưu đồ: khởi tạo port, GLCD, hiển thị các thông tin, vẽ hình.
c. Hình các file chính và thư viện:



Hình 3-4. Các nhóm lưu thư mục và đường dẫn bài 302.

d. Chương trình:

```
#include "stm32f10x.h"
#include "hardware_conf.h"
#include "TV_PORT_CONF.h"
#include "TV_GLCD_ILI932x.h"
u8 X,R;    u16 Y;

int main(void)
{
                SystemInit();                PORT_CONF();          LCD_INIT();
                LCD_CLEAR(YELLOW);BACK_COLOR=YELLOW;                POINT_COLOR=RED;
                LCD_SHOW_STRING_X(10,0,"DAI HOC SU PHAM KY THUAT");
                LCD_SHOW_STRING_X(10,20,"THANH PHO HO CHI MINH");
                LCD_SHOW_STRING_X(10,40,"KHOA DIEN - DIEN TU");
                LCD_SHOW_STRING_X(10,60,"BO MON DIEN TU CONG NGHIEP");
                LCD_SHOW_STRING_X(10,80,"VI DIEU KHIEN ARM STM32");
                LCD_SHOW_STRING_X(10,100,"NGUYEN DINH PHU");
                LCD_SHOW_STRING_X(10,120,"0903 982 443");
                LCD_SHOW_NUM(10,140,12345,5,16);

                LCD_DRAW_CIRCLE(150,180,30);
                LCD_DRAW_RECTANGLE(10,180,100,200);

                for(Y=250;Y<280;Y++)
                {
                        for(X=100;X<161;X++)
                        {
                                LCD_DRAW_POINT(X,Y);
                        }
                }
                for(R=0;R<30;R++)
                {
                                LCD_DRAW_CIRCLE(50,250,R);;
```

```
                }
                while(1)
                {                       }
}
```

e. Tiến hành biên dịch và nạp.
f. Quan sát kết quả:
g. Giải thích chương trình: giống bài trước chỉ thêm phần hiển thị số trên màn hình, vẽ hình tròn, vẽ hình vuông, tô hình vuông.
h. Cho phép thay đổi: bạn có thể thay đổi vị trí nằm trong giới hạn, thay đổi kích thước hình, thay đổi màu nền, thay đổi màu led.

Bài tập 303. Hãy viết chương trình điều khiển GLCD hiển thị chuỗi "DAI HOC SU PHAM KY THUAT" trên màn hình ở hàng thứ 0, sau đó dịch xuống hàng kế sau thời gian delay 1s (không cần chính xác).

Tạo thư mục "BAI_303_GLCD_CHUOI_DICH" để lưu project.

Bài tập 304. Giống bài 303 nhưng sau khi dịch xuống hàng cuối cùng thì dịch lên lại rồi làm lại.

Tạo thư mục "BAI_304_GLCD_HT_CHUOI_DICH_LX" để lưu project.

Bài tập 305. Hãy tìm hiểu kích thước các font chữ trong chương trình, các hàm hiển thị font chữ.

Dùng phần mềm tạo font để tạo các con số thập phân giống led 7 đoạn với kích thước 40x40 pixel.

Viết chương trình hiển thị 10 con số từ 0 đến 9 trên màn hình.

Tạo thư mục "BAI_305_GLCD_HT_0_9" để lưu project.

Bài tập 306. Hãy viết chương trình đếm từ 0 đến 9 hiển thị trên GLCD với mã các led đã tạo từ bài 305. Vị trí tuỳ chọn.

Tạo thư mục "BAI_306_GLCD_DEM_9" để lưu project.

Bài tập 307. Hãy viết chương trình đếm từ 00 đến 99 hiển thị trên GLCD với mã các led đã tạo từ bài 306. Vị trí tuỳ chọn.

Tạo thư mục "BAI_307_GLCD_DEM_99" để lưu project.

2. CÁC BÀI THỰC HÀNH HIỂN THỊ HÌNH ẢNH TRÊN GLCD

Phần này thực hành các bài hiển thị hình ảnh trên GLCD. Để hiển thị hình ảnh trên GLCD thì hình ảnh phải có kích thước tương thích với kích thước của màn hình GLCD. Nếu ảnh có kích thước lớn hơn kích thước màn hình GLCD thì ta phải nén hoặc cắt bỏ bớt cho phù hợp.

Giả sử ta có ảnh cần hiển thị phù hợp với kích thước của màn hình 320x240. Từ ảnh đó ta dùng phần mềm chuyển đổi ảnh thành mã nhị phân hay mã hex.

Chương trình sẽ tiến hành gởi mã hex ra màn hình để tái tạo lại hình ảnh.

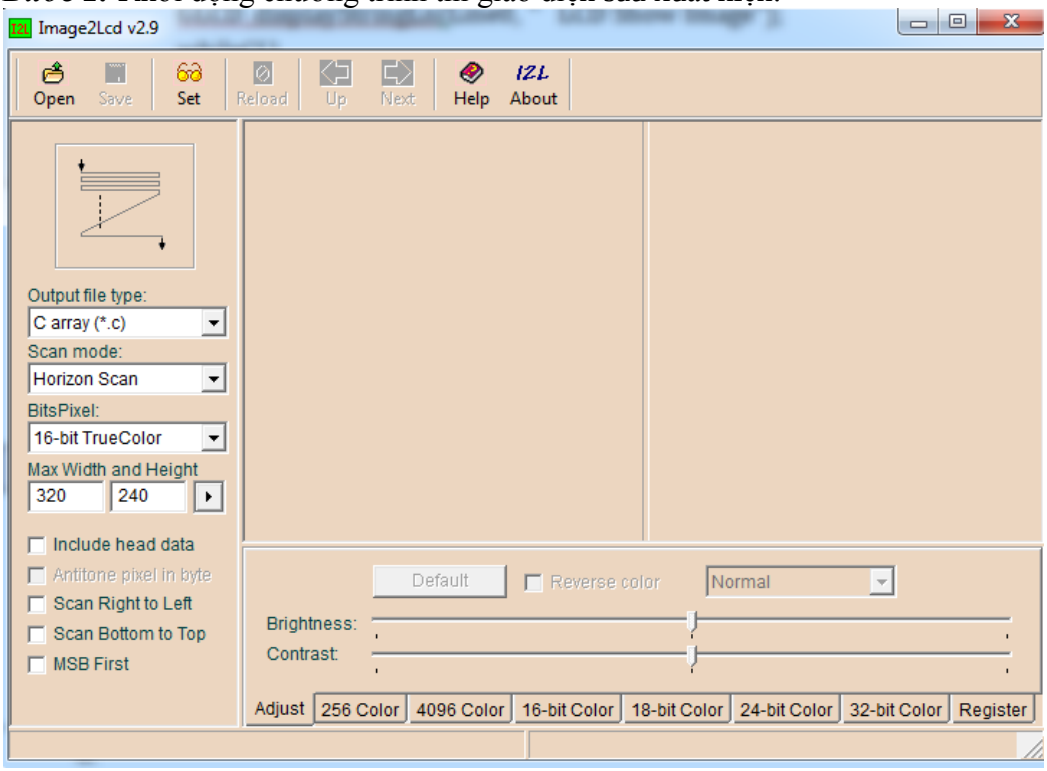
Cách tạo file hex cho hình ảnh

a. Cài phần mềm chuyển file ảnh thành file hex có tên là 'Image2LCD_29.exe', sau khi cài xong thì chép file crack vào thư mục lưu file mới cài là xong.

b. Copy file ảnh cần hiển thị vào thư mục của chương trình hiển thị hình ảnh cho tiện quản lý.

c. Cách thực hiện tạo file hex:

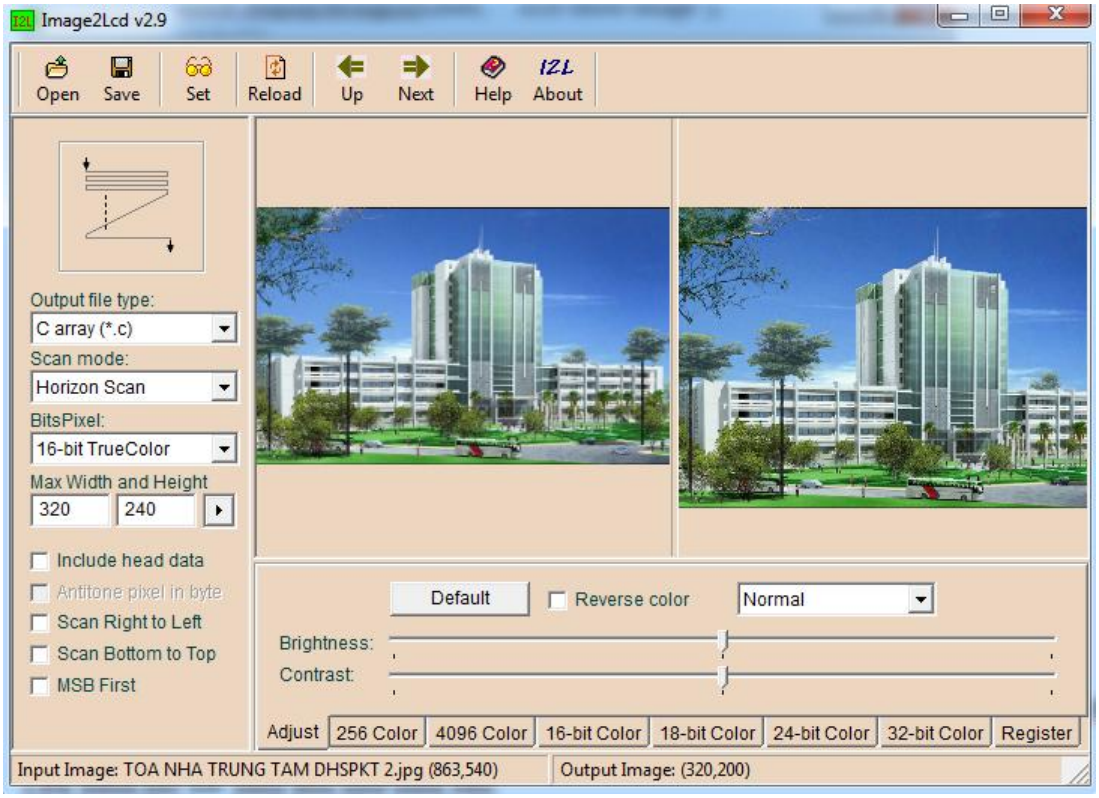
Bước 1: Khởi động chương trình thì giao diện sau xuất hiện:



Hình 3-5. Giao diện phần mềm chuyển đổi ảnh.

Bước 2: Chọn các thông số như trong giao diện: thông số về kích thước cho phù hợp với kích thước lớn nhất của màn hình GLCD.

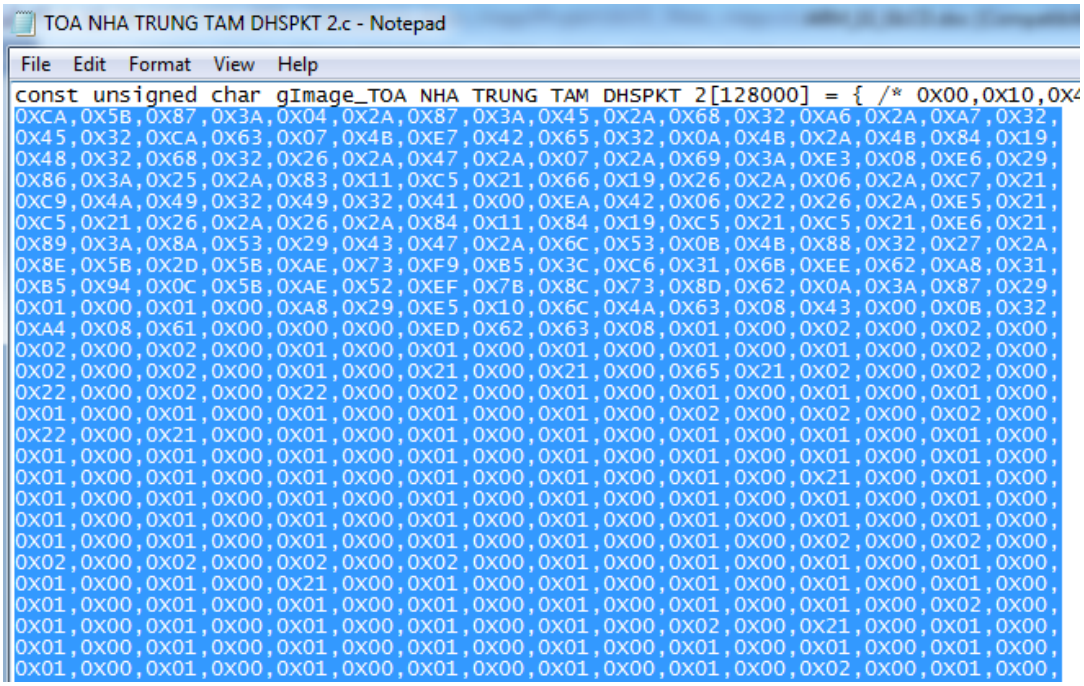
Bước 3: Tiến hành mở file hình ảnh như hình sau:



Hình 3-5. Giao diện sau khi mở ảnh.

Tiến hành bấm "Save" và đặt tên trùng với file của hình.
Sau khi chọn OK thì phần mềm tiến hành chuyển đổi và tạo ra file hex rồi tự động mở file hex bằng notepad.

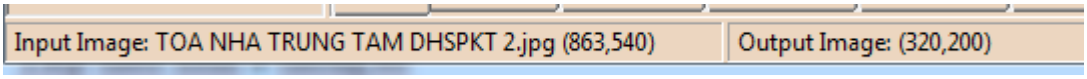
Bước 4: Tiến hành copy dữ liệu số hex từ đầu đến cuối:



Hình 3-6. File hex của ảnh sau khi chuyển đổi.

Sau đó dán vào thay thế cho dữ liệu số hex của hình ảnh cũ.

Bước 4: Tiếp theo xem 2 thông số ở hàng cuối cùng của chương trình chuyển đổi như hình sau:



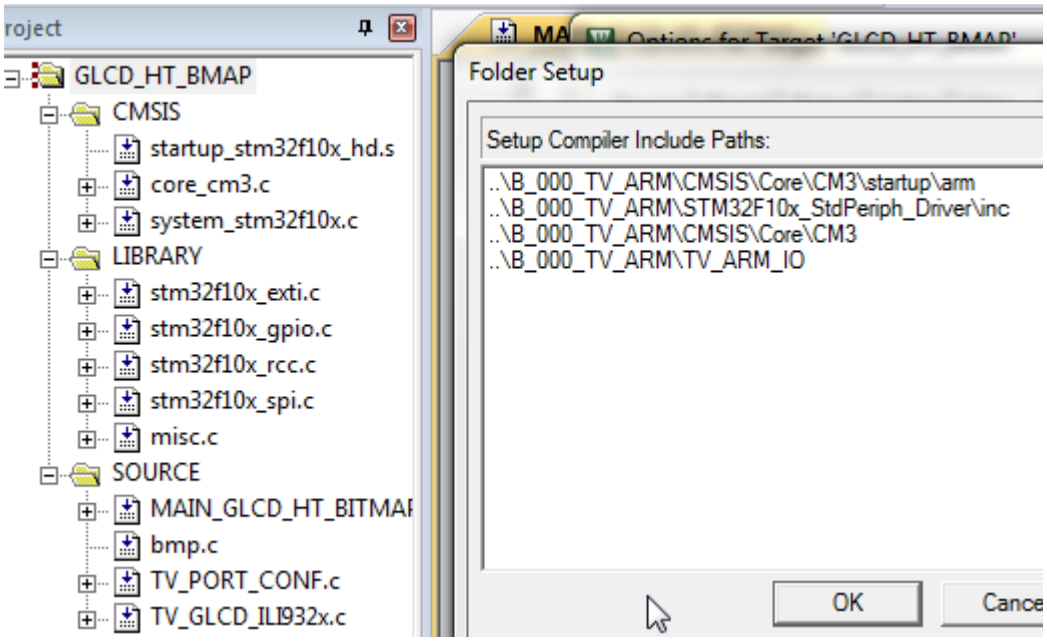
Hình 3-7. Các thông số kích thước của ảnh sau khi chuyển đổi.

Thay đổi 2 thông số đó trong chương trình chính ở hàm hiển thị bit map (0, 0,200,320, Image_Table);
Chú ý thông số chiều dài 320 là y2, chiều rộng 200 là x2, còn x1, y2 thì xuất phát từ 0,0.
Mỗi kích thước ảnh khác nhau thì thông số này cũng khác nhau.

Bài mẫu 321. Chương trình điều khiển GLCD 320×240 hiển thị hình ảnh.

Tạo thư mục "BAI_321_GLCD_BITMAP" để lưu project.

a. Mục đích: biết cách lập trình ARM để hiển thị được 1 hình ảnh trên màn hình GLCD.
b. Lưu đồ: khởi tạo port, GLCD, hiển thị ảnh.
c. Hình các file chính và thư viện:



Hình 3-8. Các nhóm lưu thư mục và đường dẫn bài 321.

d. Chương trình:

```
#include "stm32f10x.h"
#include "hardware_conf.h"
#include "TV_PORT_CONF.h"
#include "TV_GLCD_ILI932x.h"
extern u8 Image_Table[];
int main(void)
{
        SystemInit();    PORT_CONF();    LCD_INIT();
        LCD_CLEAR(WHITE);
        LCD_WRITE_BITMAP_X(0, 0,240,185, Image_Table);
        while(1)
        {
        }
}
```

e. Tiến hành biên dịch và nạp.
f. Quan sát kết quả: là hình ảnh hiển thị trên GLCD.

g. Giải thích chương trình: tiến hành khởi tạo các thanh ghi, khởi tạo GLCD, gọi hàm hiển thị mảng file hex lưu mã hex của hình ảnh ở "Image_table" với kích thước màn hình bắt đầu tại (x1,y1) = (0,0) và kết thúc tại (x2,y2) = (200,320).

h. Cho phép thay đổi: bạn có thể thay đổi vị trí nằm trong giới hạn, thay đổi kích thước hình, thay đổi hình ảnh.

Bài mẫu 322. Chương trình điều khiển GLCD 320×240 hiển thị 2 hình ảnh.

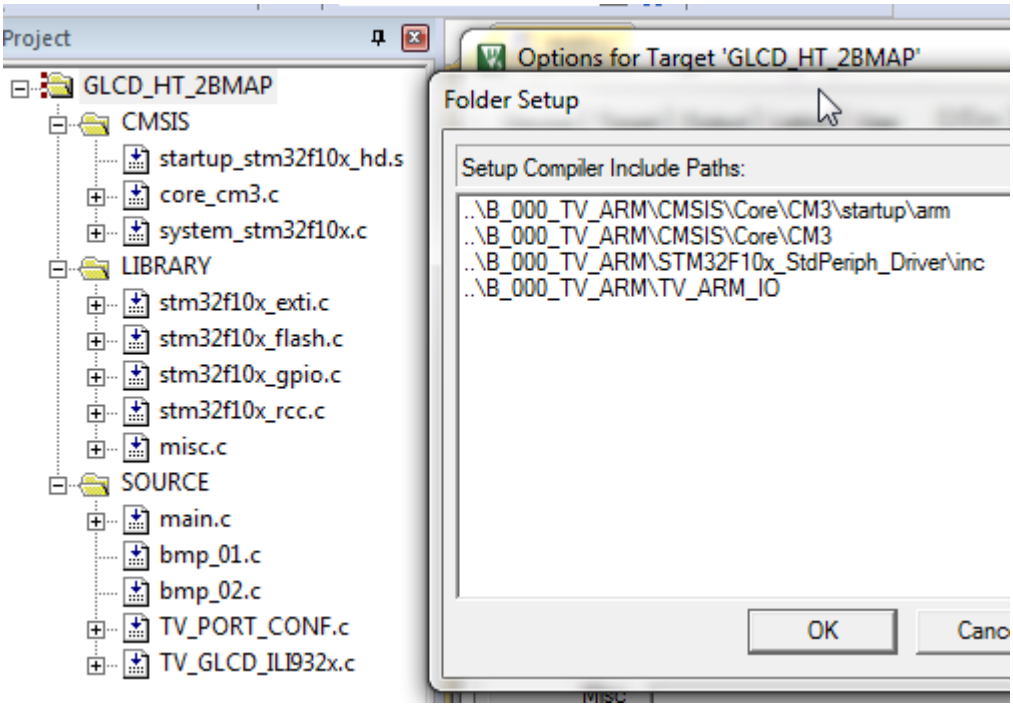
Chú ý 2 ảnh phải có kích thước sao cho đủ hiển thị trên màn hình.

Tạo thư mục "BAI_322_GLCD_BITMAP" để lưu project.

a. Mục đích: biết cách lập trình ARM để hiển thị được 2 hình ảnh trên màn hình GLCD.

b. Lưu đồ: khởi tạo port, GLCD, hiển thị lần lượt ảnh 1 và ảnh 2.

c. Hình các file chính và thư viện:



Hình 3-9. Các nhóm lưu thư mục và đường dẫn bài 322.

d. Chương trình:

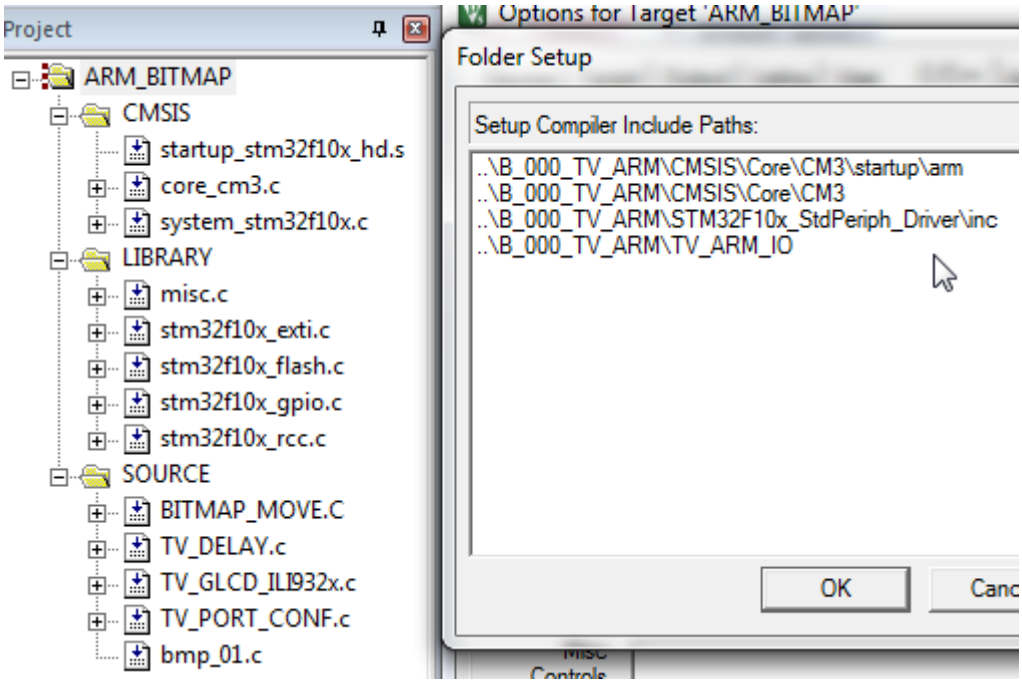
```
#include "stm32f10x.h"
#include "hardware_conf.h"
#include "TV_PORT_CONF.h"
#include "TV_GLCD_ILI932x.h"
extern u8 Image_Table_01[];
extern u8 Image_Table_02[];
int main(void)
{
        SystemInit();          PORT_CONF();          LCD_INIT();
        LCD_CLEAR(YELLOW);
        LCD_WRITE_BITMAP_X(0, 0,48,50, Image_Table_01);
        LCD_WRITE_BITMAP_X(50, 0,48,50, Image_Table_02);
        while(1)
        {          }
}
```

e. Tiến hành biên dịch và nạp.
f. Quan sát kết quả: là hình ảnh hiển thị trên GLCD.
g. Giải thích chương trình: tiến hành khởi tạo các thanh ghi, khởi tạo GLCD, khởi tạo ngắt, gọi hàm hiển thị mảng file hex lưu mã hex của hình ảnh thứ 1 và thứ 2.
Hai ảnh trong ví dụ này là 2 logo kích thước khoảng 50x50.
h. Cho phép thay đổi: bạn có thể thay đổi vị trí nằm trong giới hạn, thay đổi kích thước hình, thay đổi hình ảnh.

Bài mẫu 323. Chương trình điều khiển GLCD 320×240 hiển thị 1 hình ảnh sau đó dịch chuyển với khoảng cách 1 điểm ảnh.

Tạo thư mục "BAI_323_GLCD_BITMAP_DICH" để lưu project.

a. Mục đích: biết lập trình xử lý ảnh di chuyển.
b. Lưu đồ: khởi tạo port, GLCD, hiển thị ảnh, delay, xoá ảnh đã hiển thị, thay đổi sang toạ độ mới và hiển thị lại ảnh, kiểm tra giới hạn kích thước màn hình.
c. Hình các file chính và thư viện:



Hình 3-10. Các nhóm lưu thư mục và đường dẫn bài 323.

d. Chương trình:

```
#include "stm32f10x.h"
#include "hardware_conf.h"
#include "TV_PORT_CONF.h"
#include "TV_GLCD_ILI932x.h"
#include "TV_DELAY.h"
extern u8 Image_Table_01[];

u8 X=0, I, XC=0, XBM, STEP; u16 Y=0, YC=0, YCGH, YBM;
int main(void)
{
        SystemInit();    PORT_CONF();    LCD_INIT();

        LCD_CLEAR(YELLOW);
```

```
                POINT_COLOR=YELLOW;
                XBM=48;YBM=50;STEP = 48;
                while(1)
                {
                        for (I=0;I<5;I++)
                        {
                                Y= I*STEP;
                                LCD_WRITE_BITMAP_X(X,Y,XBM,YBM, Image_Table_01);
                                DELAY_X(5000000);

                                for(YC=Y;YC<Y+STEP;YC++)
                                {
                                        for(XC=X;XC<X+XBM;XC++)
                                        {
                                                LCD_DRAW_POINT(XC,YC);
                                        }
                                }
                        }
                        for(YC=Y;YC<Y+YBM;YC++)
                                {
                                        for(XC=X;XC<X+XBM;XC++)
                                        {
                                                LCD_DRAW_POINT(XC,YC);
                                        }
                                }
                }
}
```

e. Tiến hành biên dịch và nạp.
f. Quan sát kết quả:
g. Giải thích chương trình:
h. Cho phép thay đổi:

Bài tập 324. Chương trình điều khiển GLCD 320×240 hiển thị 1 hình ảnh sau đó dịch chuyển với khoảng cách 5 pixel.

Tạo thư mục "BAI_324_GLCD_BITMAP_DICH_5PIXEL" để lưu project.

Bài tập 325. Hãy viết chương trình điều khiển GLCD 320×240 hiển thị hình ảnh 1 sau đó 5 giây thì hiển thị hình ảnh thứ 2 rồi sau 5 giây thì hiển thị lại hình ảnh thứ 1 và lặp lại.

Tạo thư mục "BAI_325_GLCD_2BITMAP" để lưu project.

Bài tập 326. Hãy viết chương trình điều khiển GLCD 320×240 hiển thị hình ảnh 1 sau đó 5 giây thì hình ảnh thứ 1 dịch chuyển sang trái theo khoảng cách 1 pixel và hình ảnh thứ 2 bắt đầu xuất hiện nối tiếp theo.

Tạo thư mục "BAI_326_GLCD_2BITMAP_1PIXEL" để lưu project.

Hãy dùng ảnh có kích nhỏ 100x120.

Bài tập 327. Hãy lập trình hiển thị hình ảnh điều khiển máy nghe nhạc như hình theo sau:

Tạo thư mục "BAI_327_GLCD_BITMAP_ICON" để lưu project.

Bài tập 328. Hãy dùng phần mềm tạo font chữ và tiến hành tạo mã nhị phân cho biểu tượng điều khiển play (biểu tượng thứ 3 của hàng 2) của ảnh theo sau:

Cách thực hiện:

Dùng phần mềm tạo font chọn kích thước 40x40 rồi tiến hành tạo mã.

Lưu thành file rồi tiến hành viết chương trình hiển thị biểu tượng vừa tạo.

Tạo thư mục "BAI_328_GLCD_ICON_PLAY" để lưu project.

Bài tập 329. Thực hiện hiển thị cho các biểu tượng còn lại và lập trình hiển thị tất cả 10 biểu tượng:

Vị trí: mỗi icon điều khiển có kích thước 40x40, 5 biểu tượng sẽ có kích thước là 200, 5 biểu tượng cách nhau 10 pixel nên hiển thị theo chiều rộng vừa đủ 240 pixel.

Tạo thư mục "BAI_329_GLCD_ICON_ALL" để lưu project.

Sau khi thực hiện xong bạn hãy so sánh dung lượng bài này với bài hiển thị bằnb bitmap

Bài tập 330. Hãy phân tích các icon điều khiển play và dùng các hàm vẽ hình chữ nhật, hình vuông, để vẽ các biểu tượng play.

Cho kích thước 40x40 pixel.

Gợi ý: vẽ hình vuông trước với màu đã chọn, tiến hành vẽ hình tam giác.

Tạo thư mục "BAI_330_GLCD_DRAW_ICON_PLAY" để lưu project.

IV. CÁC HÀM ĐIỀU KHIỂN GLCD TRONG THƯ VIỆN <TV_GLCD_ILI932X.H>

Phần mềm có hỗ trợ file thư viện điều khiển màn hình GLCD

1. *ĐỊNH NGHĨA CÁC THÔNG SỐ*

```
#ifndef  __ILI932X_CONF_H
#define  __ILI932X_CONF_H

extern uint16_t  POINT_COLOR;
extern uint16_t  BACK_COLOR;
//extern const FNT_GB16 GBHZ_16[];
//extern const unsigned char ASCII[][16];

#define LCD_CS        PDout(12)
#define LCD_RS        PDout(13)
#define LCD_WR        PDout(14)
#define LCD_RD        PDout(15)

#define LCD_W         240
#define LCD_H         320
/*****************************************************
DINH NGHIA CAC THANH GHI
*****************************************************/
#define R0          0x00
#define R1          0x01
#define R2          0x02
#define R3          0x03
#define R4          0x04
```

```
#define R5      0x05
#define R6      0x06
#define R7      0x07
#define R8      0x08
#define R9      0x09
#define R10     0x0A
#define R12     0x0C
#define R13     0x0D
#define R14     0x0E
#define R15     0x0F
#define R16     0x10
#define R17     0x11
#define R18     0x12
#define R19     0x13
#define R20     0x14
#define R21     0x15
#define R22     0x16
#define R23     0x17
#define R24     0x18
#define R25     0x19
#define R26     0x1A
#define R27     0x1B
#define R28     0x1C
#define R29     0x1D
#define R30     0x1E
#define R31     0x1F
#define R32     0x20
#define R33     0x21
#define R34     0x22
#define R36     0x24
#define R37     0x25
#define R40     0x28
#define R41     0x29
#define R43     0x2B
#define R45     0x2D
#define R48     0x30
#define R49     0x31
#define R50     0x32
#define R51     0x33
#define R52     0x34
#define R53     0x35
#define R54     0x36
#define R55     0x37
#define R56     0x38
#define R57     0x39
#define R59     0x3B
#define R60     0x3C
```

```
#define R61     0x3D
#define R62     0x3E
#define R63     0x3F
#define R64     0x40
#define R65     0x41
#define R66     0x42
#define R67     0x43
#define R68     0x44
#define R69     0x45
#define R70     0x46
#define R71     0x47
#define R72     0x48
#define R73     0x49
#define R74     0x4A
#define R75     0x4B
#define R76     0x4C
#define R77     0x4D
#define R78     0x4E
#define R79     0x4F
#define R80     0x50
#define R81     0x51
#define R82     0x52
#define R83     0x53
#define R96     0x60
#define R97     0x61
#define R106    0x6A
#define R118    0x76
#define R128    0x80
#define R129    0x81
#define R130    0x82
#define R131    0x83
#define R132    0x84
#define R133    0x85
#define R134    0x86
#define R135    0x87
#define R136    0x88
#define R137    0x89
#define R139    0x8B
#define R140    0x8C
#define R141    0x8D
#define R143    0x8F
#define R144    0x90
#define R145    0x91
#define R146    0x92
#define R147    0x93
#define R148    0x94
```

```
#define R149        0x95
#define R150        0x96
#define R151        0x97
#define R152        0x98
#define R153        0x99
#define R154        0x9A
#define R157        0x9D
#define R192        0xC0
#define R193        0xC1
#define R229        0xE5

/*************************************************
```

2. ĐỊNH NGHĨA CÁC MÀU

```
*************************************************/
#define  BLACK        0x0000
#define  NAVY         0x000F
#define  DGREEN       0x03E0
#define  DCYAN        0x03EF
#define  MAROON       0x7800
#define  PURPLE       0x780F
#define  OLIVE        0x7BE0
#define  LGRAY        0xC618
#define  DGRAY        0x7BEF
#define  BLUE         0x001F
#define  GREEN        0x07E0
#define  CYAN         0x07FF
#define  RED          0xF800
#define  MAGENTA      0xF81F
#define  YELLOW       0xFFE0
#define  WHITE        0xFFFF
#define   IDMCOLOR(color) (((color & 0x001F) << 11) | ((color & 0xF800) >> 11) | (color & 0x07E0))
//#define  BACK_COLOR   BLACK
//#define  POINT_COLOR  WHITE

#define Line0          0
#define Line1          24
#define Line2          48
#define Line3          72
#define Line4          96
#define Line5          120
#define Line6          144
#define Line7          168
#define Line8          192
#define Line9          216
```

```
void LCD_WRITE_STRING(uint16_t x0, uint16_t y0,uint8_t *pcStr, uint16_t color);
void LCD_WRITE_RAM_PREPARE(void);
void LCD_DRAW_POINT(uint16_t x,uint16_t y);
void LCD_SET_CURSOR(uint8_t Xpos,uint16_t Ypos);
void LCD_WRITE_REG(uint16_t LCD_Reg,uint16_t LCD_Dat);
uint16_t LCD_READ_DATA(void);
void LCD_WRITE_CMD(uint16_t LCD_Reg);
void LCD_WRITE_DATA(uint16_t LCD_Dat);
uint16_t LCD_READ_REG(uint16_t LCD_Reg);
void LCD_INIT(void);
void LCD_CONFIGURATION(void);
void LCD_Delay(uint32_t nCount);
uint32_t NUM_POWER(uint8_t M,uint8_t N);
void LCD_SET_DISPLAY_WINDOW(uint8_t X, uint16_t Y, uint8_t X_END, uint16_t Y_END);
void LCD_WRITE_BITMAP_X(u8 Xpos, u16 Ypos, u8 Height, u16 Width, u8 *bitmap);
void LCD_SHOW_STRING_X(uint8_t X,uint16_t Y,__I uint8_t *p);
void LCD_SHOW_NUM(uint8_t x,uint16_t y,u32 num,uint8_t len,uint8_t size);
void LCD_SHOW_CHAR_X(uint8_t X,uint16_t Y,uint8_t chars,uint8_t size,uint8_t mode);
void LCD_WriteBMP(uint8_t Xpos, uint16_t Ypos, uint8_t Height, uint16_t Width, uint8_t *bitmap);
void LCD_CLEAR(uint16_t COLOR);
void LCD_WRITE_BITMAP(uint8_t Xpos, uint16_t Ypos, uint8_t Height, uint16_t Width, uint8_t *bitmap);
void LCD_DISPLAY_STRING_LINE(unsigned int ln, unsigned char *s) ;
void LCD_DISPLAY_STRING_LINE_X(unsigned int ln, unsigned char *s) ;
void LCD_SHOW_NUM_FULL(uint8_t X,uint16_t Y,u32 NUMBER,uint8_t LEN,uint8_t SIZE);
void GLCD_bitmap_p (unsigned int x, unsigned int y, unsigned int w, unsigned int h, unsigned char *bitmap) ;
void LCD_WRITE_BITMAP_Y(u8 Xpos, u16 Ypos, u8 Height, u16 Width, u8 *bitmap);
void LCD_SHOW_CHAR_X3(uint8_t X,uint16_t Y,uint8_t chars,uint8_t size,uint8_t mode);
uint16_t WriteOneASCII(uint8_t *pucMsk,
                       uint16_t x0,
                       uint16_t y0,
                       uint16_t color);
uint16_t WriteOneHzChar(uint8_t *pucMsk,
                        uint16_t x0,
                        uint16_t y0,
                        uint16_t color);
void WriteString(uint16_t x0, uint16_t y0,uint8_t *pcStr, uint16_t color);
void LCD_DRAW_LINE(uint16_t X1, uint16_t Y1, uint16_t X2, uint16_t Y2);

void LCD_DRAW_CIRCLE(uint8_t X0,uint16_t Y0,uint8_t R);
void LCD_DRAW_RECTANGLE(u8 X1, u16 Y1, u8 X2, u16 Y2);
void LCD_CLEAR_STRING_X(uint8_t X,uint16_t Y,__I uint8_t *p);
```

```
//uint16_t findHzIndex(uint8_t *hz) ;

#endif
/****************************** End File *******************************/
```

V. CÁC HÀM ĐIỀU KHIỂN GLCD TRONG THƯ VIỆN <TV_GLCD_ILI932X.C>

Phần mềm có hỗ trợ file thư viện điều khiển màn hình GLCD

1. *HÀM GHI LỆNH ĐIỀU KHIỂN VÀO GLCD*

- Tên hàm: LCD_WRITE_CMD (uint16_t LCD_REG)
- Thông số: có 1 thông số là mã lệnh điều khiển chức năng nào đó cho GLCD.
- Chức năng: thực hiện công việc chọn chip, thanh ghi lệnh, gởi lệnh, tạo xung ghi.
- Hàm chi tiết như sau :

```
void LCD_WRITE_CMD (uint16_t LCD_REG)
{
 LCD_CS = 0;            LCD_RS = 0;
 GPIOE->ODR = LCD_REG;
 LCD_WR = 0;            LCD_WR = 1;
 LCD_CS = 1;
}
```

2. *HÀM GHI DỮ LIỆU VÀO GLCD*

- Tên hàm: LCD_WRITE_DATA (uint16_t LCD_DATA)
- Thông số: có 1 thông số là dữ liệu gởi ra cho GLCD.
- Chức năng: thực hiện công việc chọn chip, thanh ghi dữ liệu, gởi dữ liệu, tạo xung ghi.
- Hàm chi tiết như sau :

```
void LCD_WRITE_DATA (uint16_t LCD_DATA)
{
        LCD_CS = 0;            LCD_RS = 1;
        GPIOE->ODR = LCD_DATA;
        LCD_WR = 0;            LCD_WR = 1;
        LCD_CS = 1;
}
```

3. *HÀM GHI LỆNH VÀ GHI DỮ LIỆU VÀO GLCD*

- Tên hàm: LCD_WRITE_REG (uint16_t LCD_REG, uint16_t LCD_DATA)
- Thông số: có 2 thông số lần lượt là mã lệnh điều khiển và là dữ liệu gởi ra cho GLCD.
- Chức năng: thực hiện công việc gởi mã điều khiển sau đó gởi dữ liệu ra GLCD. Hàm này tổng hợp của 2 hàm.
- Hàm chi tiết như sau:

```
void LCD_WRITE_REG (uint16_t LCD_REG,uint16_t LCD_DATA)
{
```

```
        LCD_WRITE_CMD(LCD_REG);
        LCD_WRITE_DATA(LCD_DATA);
}
```

4. HÀM ĐỌC GIÁ TRỊ CỦA THANH GHI TRONG GLCD

- Tên hàm: uint16_t LCD_READ_REG (uint16_t LCD_REG)
- Thông số: có 1 thông số là tên hoặc địa chỉ thanh ghi cần đọc từ GLCD.
- Chức năng: thực hiện công việc đọc giá trị thanh ghi hay ô nhớ trong GLCD và trả về giá trị gán cho hàm gọi.
- Hàm chi tiết như sau:

```
uint16_t LCD_READ_REG (uint16_t LCD_REG)
{
        uint16_t temp;
        LCD_WRITE_CMD(LCD_REG);
        GPIOE->CRH =  0x44444444;          GPIOE->CRL =  0x44444444;
        LCD_CS = 0;          LCD_RS = 1;          LCD_RD = 0;
        temp = GPIOE->IDR&0xFFFF;
        LCD_RD = 1;          LCD_CS = 1;
        GPIOE->CRH = 0x33333333;          GPIOE->CRL = 0x33333333;
        return temp;
}
```

5. HÀM ĐỌC DỮ LIỆU TỪNG VÙNG NHỚ HIỂN THỊ TRONG GLCD

- Tên hàm: uint16_t LCD_READ_DATA()
- Thông số: không có thông số nên cần phải khởi tạo con trỏ trước khi đọc.
- Chức năng: thực hiện công việc đọc giá của ô nhớ trong GLCD và trả về giá trị gán cho hàm gọi.
- Hàm chi tiết như sau:

```
uint16_t LCD_READ_DATA()
{
        uint16_t temp;
        GPIOE->CRH =  0x44444444;          GPIOE->CRL =  0x44444444;
        LCD_CS = 0;          LCD_RS = 1;          LCD_RD = 0;
        temp = GPIOE->IDR;
        LCD_RD = 1;          LCD_CS = 1;
        GPIOE->CRH =         0x33333333;          GPIOE->CRL =  0x33333333;
        return temp;
}
```

6. HÀM KHỞI TẠO CÁC PORT CỦA VI ĐIỀU KHIỂN GIAO TIẾP VỚI GLCD

- Tên hàm: LCD_Configuration()
- Thông số: không có thông số.
- Chức năng: thực hiện công việc định cấu hình cho các port IO của vi điều khiển giao tiếp với GLCD bao gồm cả port E 16 bit và 4 tín hiệu điều khiển của port D. Hàm này thường là cố định chỉ gọi 1 lần khi có liên quan đến GLCD.

- Hàm chi tiết như sau:

```
void LCD_Configuration()
{
	GPIO_InitTypeDef GPIO_LCD;
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD| RCC_APB2Periph_GPIOE,ENABLE);

	GPIO_LCD.GPIO_Pin = GPIO_Pin_12 | GPIO_Pin_13 | GPIO_Pin_14 | GPIO_Pin_15;
	GPIO_LCD.GPIO_Mode = GPIO_Mode_Out_PP;
	GPIO_LCD.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_Init(GPIOD, &GPIO_LCD);

	GPIO_LCD.GPIO_Pin = GPIO_Pin_All;
	GPIO_LCD.GPIO_Mode = GPIO_Mode_Out_PP;
	GPIO_LCD.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_Init(GPIOE, &GPIO_LCD);
}
```

7. HÀM KHỞI TẠO GLCD

- Tên hàm: LCD_INIT(void)
- Thông số: không có thông số.
- Chức năng: thực hiện công việc khởi tạo port và GLCD.
- Hàm chi tiết như sau:

```
void LCD_INIT(void)
{
	LCD_CONFIGURATION();
	LCD_WRITE_REG(0x0000,0x0001);
	LCD_Delay(5); // LCD_Delay 50 ms

	DeviceCode = LCD_READ_REG(0x0000);
	if(DeviceCode==0x8999)		//SSD1289
	{
		//************* Start Initial Sequence **********//
		LCD_WRITE_REG(0x00, 0x0001); // Start internal OSC.
		LCD_WRITE_REG(0x01,0x3B3F);
//Driver output control, RL=0;REV=1;GD=1;BGR=0;SM=0;TB=1
		LCD_WRITE_REG(0x02, 0x0600); // set 1 line inversion
		//************* Power control setup ************/
		LCD_WRITE_REG(0x0C, 0x0007); // Adjust VCIX2 output voltage
		LCD_WRITE_REG(0x0D, 0x0006); // Set amplitude magnification of VLCD63
		LCD_WRITE_REG(0x0E, 0x3200); // Set alternating amplitude of VCOM
		LCD_WRITE_REG(0x1E, 0x00BB); // Set VcomH voltage
		LCD_WRITE_REG(0x03, 0x6A64); // Step-up factor/cycle setting
		//************ RAM position control **********/
		LCD_WRITE_REG(0x0F, 0x0000); // Gate scan position start at G0.
		LCD_WRITE_REG(0x44, 0xEF00); // Horizontal RAM address position
		LCD_WRITE_REG(0x45, 0x0000); // Vertical RAM address start position
		LCD_WRITE_REG(0x46, 0x013F); // Vertical RAM address end position
		// ---------- Adjust the Gamma Curve ----------//
```

```
                LCD_WRITE_REG(0x30, 0x0000);
                LCD_WRITE_REG(0x31, 0x0706);
                LCD_WRITE_REG(0x32, 0x0206);
                LCD_WRITE_REG(0x33, 0x0300);
                LCD_WRITE_REG(0x34, 0x0002);
                LCD_WRITE_REG(0x35, 0x0000);
                LCD_WRITE_REG(0x36, 0x0707);
                LCD_WRITE_REG(0x37, 0x0200);
                LCD_WRITE_REG(0x3A, 0x0908);
                LCD_WRITE_REG(0x3B, 0x0F0D);
                //************* Special command **************/
                LCD_WRITE_REG(0x28, 0x0006); // Enable test command
                LCD_WRITE_REG(0x2F, 0x12EB); // RAM speed tuning
                LCD_WRITE_REG(0x26, 0x7000); // Internal Bandgap strength
                LCD_WRITE_REG(0x20, 0xB0E3); // Internal Vcom strength
                LCD_WRITE_REG(0x27, 0x0044); // Internal Vcomh/VcomL timing
                LCD_WRITE_REG(0x2E, 0x7E45); // VCOM charge sharing time
                //************* Turn On display ******************/
                LCD_WRITE_REG(0x10, 0x0000); // Sleep mode off.
                LCD_Delay(8); // Wait 30mS
                LCD_WRITE_REG(0x11, 0x6870);
// Entry mode setup. 262K type B, take care on the data bus with 16it only
                LCD_WRITE_REG(0x07, 0x0033); // Display ON        */
        }
}
```

Trong hàm khởi tạo GLCD này có gọi hàm khởi tạo port của vi điều khiển giao tiếp với GLCD đã trình bày chi tiết ở hàm số 6.

Lệnh tiếp theo là cho phép bộ dao động của IC điều khiển GLCD hoạt động, delay để chờ bộ dao động ổn định.

Tiến hành đọc mã của IC điều khiển màn hình và lưu vào biến DeviceCode.

Tiến hành kiểm tra nếu đúng là mã của thiết bị thì thực hiện lệnh khởi tạo tương ứng với thiết bị điều khiển màn hình GLCD vì có rất nhiều chip điều khiển màn hình có mã khác nhau và lệnh khởi tạo cũng khác nhau, trong kit này sử dụng chip có mã là 0x8999 thì tiến hành các lệnh khởi tạo tương ứng, bỏ đi các lệnh khởi tạo của các chip khác.

Các lệnh khởi tạo tiếp theo được cho trong datasheet của IC SSD1289.

8. HÀM THIẾT LẬP VỊ TRÍ CON TRỎ TRÊN MÀN HÌNH GLCD

- Tên hàm: LCD_SET_CURSOR (u8 Xpos, u16 Ypos)
- Thông số: thông số thứ nhất là tọa độ x, giá trị 8 bit, thông số thứ hai là tọa độ y, giá trị 16 bit.
- Chức năng: Kiểm tra mã của thiết bị GLCD rồi thực hiện lệnh khởi tạo con trỏ đúng với vị trị tọa độ (x,y).
- Hàm chi tiết như sau:

```
void LCD_SET_CURSOR (u8 Xpos, u16 Ypos)
{
        if(DeviceCode==0x8999||DeviceCode==0x9919)
        {
                LCD_WRITE_REG(0x004E, Xpos);
                LCD_WRITE_REG(0X004F, Ypos);
```

```
        }
}
```

9. HÀM VẼ 1 ĐIỂM SÁNG TRÊN MÀN HÌNH GLCD

- Tên hàm: LCD_DRAW_POINT (uint16_t X, uint16_t Y)
- Thông số: thông số thứ nhất là tọa độ x, giá trị 8 bit, thông số thứ hai là tọa độ y, giá trị 16 bit.
- Chức năng: thực hiện 3 công việc:
  - Gọi hàm thiết lập con trỏ tại tọa độ (X,Y).
  - Gọi hàm ghi lệnh vào GLCD.
  - Gọi hàm thiết lập màu cho con trỏ.
- Hàm chi tiết như sau :

```
void LCD_DRAW_POINT (uint16_t X, uint16_t Y)
{
                LCD_SET_CURSOR(X,Y);
                LCD_WRITE_CMD(R34);
                LCD_WRITE_DATA(POINT_COLOR);
}
```

10. HÀM THIẾT LẬP CỬA SỔ HIỂN THỊ TRÊN MÀN HÌNH GLCD

- Tên hàm: LCD_SET_DISPLAY_WINDOW(uint8_t X, uint16_t Y, uint8_t X_END, uint16_t Y_END)
- Thông số: cặp thông số thứ nhất là tọa độ thứ nhất (x,y) và cặp thông số thứ 2 là giá trị lệch so với cặp tọa độ thứ nhất hay tọa độ thứ 2 bằng (x+x_end, y+y_end).
- Chức năng: thực hiện thiết lập cửa sổ.
- Hàm chi tiết như sau:

```
void LCD_SET_DISPLAY_WINDOW(uint8_t X, uint16_t Y, uint8_t X_END, uint16_t Y_END)
{
        if(DeviceCode==0x8999)
        {
                LCD_WRITE_REG (0x44, X|((X_END+X)<<8));
                LCD_WRITE_REG (0x45, Y);
                LCD_WRITE_REG (0x46, Y+Y_END);
        }
}
```

11. HÀM HIỂN THỊ CHUỖI TRÊN MÀN HÌNH GLCD THEO TRỤC X

- Tên hàm: LCD_ShowString_X(uint8_t x,uint16_t y,__I uint8_t *p)
- Thông số: các thông số là tọa độ bắt đầu hiển thị chuỗi (x,y), thông số tiếp theo là chuỗi cần hiển thị.
- Chức năng:
  Kiểm tra chuỗi hiển thị nếu không phải là ký tự kết thúc chuỗi thì tiến hành hiển thị.
  Trước khi hiển thị thì kiểm tra tọa độ bắt đầu theo trục x nếu vượt quá giới hạn thì chuyển sang hàng kế và x bắt đầu lại từ 0.
  Kiểm tra giá trị của y nếu vượt quá giới hạn thì về tọa độ (0,0), tiến hành xóa màn hình bằng màu nền rồi cho hiển thị bắt đầu từ 0,0.
  Gọi hàm hiển thị từng ký tự tại tọa độ (x,y) với kích thước 16.

Sau mỗi ký tự thì tăng x lên 8 để hiển thị ký tự tiếp theo, tăng con trỏ chuỗi lên 1 để trỏ đến ký tự kế.

Làm cho đến khi gặp ký tự kết thúc chuỗi thì ngừng.

- Hàm chi tiết như sau:

```
#define MAX_CHAR_POSX 232
#define MAX_CHAR_POSY 304
void LCD_SHOW_STRING_X (uint8_t X, uint16_t Y,__I uint8_t *p)
{
  while(*p!='\0')
  {
    if(X>MAX_CHAR_POSX){X=0;X+=16;}
    if(Y>MAX_CHAR_POSY){Y=X=0;LCD_Clear(BACK_COLOR);}
    LCD_SHOW_CHAR_X (X,Y,*p,16,0);
    X+=8;
    p++;
  }
}
```

12. HÀM XÓA CHUỖI TRÊN MÀN HÌNH GLCD THEO TRỤC X

- Tên hàm: LCD_CLEAR_STRING_X (uint8_t X, uint16_t Y,__I uint8_t *p)
- Thông số: các thông số là tọa độ bắt đầu hiển thị chuỗi (X,Y), thông số tiếp theo là chuỗi rỗng gồm các ký tự trắng để hiển thị xóa chuỗi đã hiển thị trước đó.
- Chức năng:
  Giống như hàm hiển thị chuỗi.
- Hàm chi tiết như sau:

```
void LCD_CLEAR_STRING_X (uint8_t X, uint16_t Y,__I uint8_t *p)
{
  while(*p!='\0')
  {
    if(X>MAX_CHAR_POSX){X=0;X+=16;}
    if(Y>MAX_CHAR_POSY){Y=X=0;LCD_Clear(BACK_COLOR);}
    LCD_SHOW_CHAR_X(X, Y,*p,16,0);
    X+=8;
    p++;
  }
}
```

13. HÀM HIỂN THỊ TỪNG KÝ TỰ TRÊN MÀN HÌNH GLCD THEO TRỤC X

- Tên hàm: LCD_SHOW_CHAR_X (uint8_t X,uint16_t Y,uint8_t chars,uint8_t size,uint8_t mode)
- Thông số: các thông số là tọa độ bắt đầu hiển thị chuỗi (X,Y), thông số tiếp theo là ký tự, thông số thứ 4 là kích thước là 12 hoặc 16 và thông số cuối cùng là mode hoặc 0 hoặc 1 nếu bằng 0 thì hiển thị chữ và màu nền, bằng 1 thì chỉ hiển thị chữ, màu nền giữ nguyên.
- Chức năng: hiển thị 1 ký tự bằng cách thực hiện các bước sau:
  Kiểm tra tọa độ nếu vượt quá giới hạn thì thoát, nếu không vượt thì tiến hành hiển thị, các bước thực hiện như sau :

Thiết lập cửa sổ hiển thị trong phạm vi 1 ký tự có tọa độ bắt đầu là (x,y) và giá trị lệch theo trục x bằng kích thước ký tự chia cho 2 trừ 1, theo trục y bằng kích thước ký tự trừ 1.

- Ví dụ 1: tọa độ (X,Y) = (0,0), kích thước ký tự là 12 thì tọa độ thứ 2 là (X+12/2-1, Y+12-1) = (X+5,Y+11) nếu thay (X,Y) = (0,0) thì kích thước cho ký tự là 6,12 vì tính từ 0.
- Ví dụ 2: tọa độ (X,Y) = (0,0), kích thước ký tự là 16 thì tọa độ thứ 2 là (X+16/2-1, Y+16-1) = (X+7,Y+15) nếu thay (X,Y) = (0,0) thì kích thước cho ký tự là 8,16 vì tính từ 0.

Tiến hành gọi hàm thiết lập con trỏ hiển thị, gọi hàm chuẩn bị ghi dữ liệu vào RAM của GLCD. Kiểm tra nếu biến mode phủ định bằng 1 thì tiến hành thực hiện vòng lặp for chạy từ 0 đến kích thước ký tự trừ 1 tiến hành các lệnh trong vòng lặp:

- Nếu kích thước kí tự bằng 12 thì tiến hành lấy mã trong mảng mã có tên là ACII_1206.
- Nếu kích thước kí tự bằng 16 thì tiến hành lấy mã trong mảng mã có tên là ACII_1608.
- Vị trí lấy ký tự bằng mã ký tự trong bảng mã ASCII chuẩn trừ đi cho 0x20. Tại sao trừ cho 0x20 thì các bạn hãy tự tìm hiểu.

Chú ý: các thông số kích thước đã nêu ở 2 ví dụ trên.

Tiếp theo là vòng lặp for sẽ tiến hành kiểm tra từng bit của mã lưu trong biến temp: nếu bằng 1 thì ghi dữ liệu màu đã chọn cho điểm sáng đó, nếu bằng không thì ghi dữ liệu màu nền đã chọn. Sau khi ghi xong byte dữ liệu lưu trong temp thì vòng lặp for bên ngoài sẽ thực hiện tiếp những byte còn lại của ký tự.

- Hàm chi tiết như sau:

```
void LCD_SHOW_CHAR_X(uint8_t X,uint16_t Y,uint8_t chars,uint8_t size,uint8_t mode)
{
	uint8_t temp;
	uint8_t pos,t;
  if(X>MAX_CHAR_POSX||Y>MAX_CHAR_POSY) return;

		LCD_SET_DISPLAY_WINDOW(X,Y,(size/2-1),size-1);
		LCD_SET_CURSOR(X,Y);
		LCD_WRITE_RAM_PREPARE();
		if(!mode)
		{
			for(pos=0;pos<size;pos++)
			{
				if(size==12)	temp=ASCII_1206[chars-0x20][pos];
				else		temp=ASCII_1608[chars-0x20][pos];
				for(t=0;t<size/2;t++)
				{
					if((temp<<t)&0x80)	LCD_WRITE_DATA(POINT_COLOR);
					else			LCD_WRITE_DATA(BACK_COLOR);
				}
			}
		}
		else
		{
			for(pos=0;pos<size;pos++)
			{
				if(size==12)temp=ASCII_1206[chars-0x20][pos];
```

```
                                else temp=ASCII_1608[chars-0x20][pos];
                                for(t=0;t<size/2;t++)
                                {
                                        if((temp<<t)&0x80)LCD_DRAW_POINT(X+t,Y+pos);
                                }
                        }
                }
//      LCD_WRITE_REG(0x43, 0x0000);
                LCD_WRITE_REG(0x44, 0xEF00);
                LCD_WRITE_REG(0x45, 0x0000);
                LCD_WRITE_REG(0x46, 0x013F);
}
```

Bảng mã của các ký tự lưu trong file "TV_FONT.h"

{0x00,0x00,0x00,0x10,0x10,0x18,0x28,0x28,0x24,0x3C,0x44,0x42,0x42,0xE7,0x00,0x00,// "A"

Bảng 3-1: Mã các ký tự A

BYTE	7	6	5	4	3	2	1	0
00								
00								
00								
10	0	0	0	1	0	0	0	0
10	0	0	0	1	0	0	0	0
18	0	0	0	1	1	0	0	0
28	0	0	1	0	1	0	0	0
28	0	0	1	0	1	0	0	0
24	0	0	1	0	0	1	0	0
3C	0	0	1	1	1	1	0	0
44	0	1	0	0	0	1	0	0
42	0	1	0	0	0	0	1	0
42	0	1	0	0	0	0	1	0
E7	1	1	1	0	0	1	1	1
00	0	0	0	0	0	0	0	0
00	0	0	0	0	0	0	0	0

```
/* 'A' */
  0x0000, 0x0380, 0x0380, 0x06C0, 0x06C0, 0x06C0, 0x0C60, 0x0C60,
  0x1830, 0x1830, 0x1830, 0x3FF8, 0x3FF8, 0x701C, 0x600C, 0x600C,
  0xC006, 0xC006, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
  /* 'B' */
```

TT	BYTE	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	BYTE
0	00	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00
1	03	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	80
2	03	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	80
3	06	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	C0
4	06	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	C0
5	06	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	C0
6	0C	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	60
7	0C	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	60
8	18	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	30
9	18	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	30
10	18	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	30
11	3F	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	F8

12	3F	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	F8
13	70	0	1	1	1	0	0	0	0	0	0	0	1	1	1	0	0	1C
14	60	0	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0C
15	60	0	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0C
16																		
17																		
18																		
19																		
20																		
21																		
22																		
23																		

Ký tự số 0 có mã ASCII là 0x30 (48 thập phân), trong bảng font chữ thì có số thứ tự (16) nên trừ đi 32 tương đương (0x20)

14. HÀM XÓA TOÀN BỘ MÀN HÌNH GLCD BẰNG 1 TRONG 12 MÀU

- Tên hàm: LCD_CLEAR(uint16_t COLOR)
- Thông số: có 1 thông số là màu của màn hình GLCD.
- Chức năng: thực hiện 3 công việc:
  - Gọi hàm thiết lập con trỏ tại tọa độ (0,0).
  - Gọi hàm ghi lệnh vào GLCD.
  - Cho vòng lặp for thực hiện ghi dữ liệu color vào vùng nhớ dữ liệu của GLCD. Dùng hàm ghi dữ liệu LCD_WRITE_DATA(COLOR); số điểm ảnh bằng 240×320 = 76800.
- Hàm chi tiết như sau :

```
void LCD_CLEAR (uint16_t COLOR)
{
    uint32_t index=0;
    LCD_SET_CURSOR(0x00,0x0000);
    LCD_WRITE_RAM_PREPARE();
    for(index=0;index<76800;index++)
    {
        LCD_WRITE_DATA(COLOR);
    }
}
```

15. HÀM TÍNH LŨY THỪA

- Tên hàm: NUM_POWER(uint8_t M,uint8_t N)
- Thông số: có 2 thông số để tính là m và n.
- Chức năng: Tính result = $m^{(n-1)}$
- Hàm chi tiết như sau :

```
uint32_t NUM_POWER (uint8_t M,uint8_t N)
{
    u32 RESULT=1;
    while(N--)RESULT*=M;
    return RESULT;
}
```

- Giải thích:
  Khai báo biến lưu kết quả là "RESULT" và gán giá trị ban đầu bằng 1. Thực hiện lệnh while giảm giá trị N đi 1 và tính kết quả "RESULT" bằng "RESULT" nhân với giá trị M, làm cho đến khi N bằng 0 thì trả kết quả. Ví dụ: cho M = 10 và N = 4 thì RESULT = 1*10*10*10 = 1000

16. HÀM HIỂN THỊ SỐ TRÊN MÀN HÌNH GLCD

- Tên hàm: LCD_SHOW_NUM (uint8_t x,uint16_t y,u32 num,uint8_t len,uint8_t size)
- Thông số: có 5 thông số: tọa độ bắt đầu hiển thị (x,y), con số cần hiển thị tối đa là 32 bit, chiều dài của con số, kích thước ký tự hiển thị..
- Chức năng: từ con số cần hiển thị sẽ được chia tách thành từng con số bắt đầu với số có trọng số lớn nhất, sau đó tiến hành kiểm tra nếu số không có nghĩa thì hiển thị ký tự trắng, nếu có nghĩ thì hiển thị, làm đến con số cuối cùng.
- Hàm chi tiết như sau :

```
void LCD_SHOW_NUM (uint8_t X,uint16_t Y, u32 NUMBER, uint8_t LEN, uint8_t SIZE)
{
    uint8_t t, temp, enshow=0;
    for(t=0;t<LEN;t++)
    {
        temp=(NUMBER/NUM_POWER(10,LEN-t-1))%10;
        if(enshow==0&&t<(LEN-1))
        {
            if(temp==0)
            {
                LCD_SHOW_CHAR_X(X+(SIZE/2)*t,Y,' ',SIZE,0);
                continue;
            }
            else
                enshow=1;
        }
        LCD_SHOW_CHAR_X(X+(SIZE/2)*t,Y,temp+'0',SIZE,0);
    }
}
```

- Giải thích:

  Vòng lặp for sẽ tiến hành hiển thị số ký tự của con số là "num" phụ thuộc vào biến chiều dài nhận được là "len".

  Lệnh "temp=(NUMBER/NUM_POWER(10,LEN-t-1))%10;" có chức năng gọi hàm tính M lũy thừa N trừ 1 với M bằng 10, còn n phụ thuộc vào chiều dài con số cần hiển thị.

  Ví dụ cho con số cần hiển thị là 12345 gồm 5 số.

  Khi đó hàm "NUM_POWER (10,5-0-1=4)", kết quả trả về là 10000, hàm sẽ lấy giá trị 12345 chia nguyên cho 10000 sẽ được là 1, chia tiếp cho 10 lấy số dư nên được kết qua sau cùng bằng 1 chính là hàng chục ngàn.

  Khi biến t tăng lên 1 thì hàm "NUM_POWER (10,5-0-1=4)", kết quả trả về là 10000, hàm sẽ lấy giá trị 12345 chia nguyên cho 10000 sẽ được là 1, chia tiếp cho 10 lấy số dư nên được kết qua sau cùng bằng 1 chính là hàng chục ngàn.

17. HÀM VẼ ĐƯỜNG THẲNG

- Tên hàm: LCD_DRAW_LINE (uint16_t X1, uint16_t Y1, uint16_t X2, uint16_t Y2)
- Thông số: có 2 cặp thông số thứ nhất là tọa độ (X1,Y1) và (X2,Y2).
- Chức năng: vẽ đường thẳng từ toạ độ (X1,Y1) đến toạ độ (X2,Y2).
- Hàm chi tiết như sau :

```
void LCD_DRAW_LINE (uint16_t X1, uint16_t Y1, uint16_t X2, uint16_t Y2)
```

```
{
	uint16_t t;
	int xerr=0,	yerr=0,delta_x,delta_y,	distance;
	int incx, incy, uRow, uCol;

	delta_x = X2-X1;
	delta_y = Y2-Y1;
	uRow = X1;
	uCol = Y1;
	if(delta_x>0)	incx=1;
	else if(delta_x==0)	incx=0;
	else {incx=-1;delta_x=-delta_x;}
	if(delta_y>0)	incy=1;
	else if(delta_y==0)	incy=0;
	else{incy=-1;	delta_y=	-delta_y;}
	if( delta_x>delta_y)	distance = delta_x;
	else distance = delta_y;
	for(t=0;t<=distance+1;t++ )
	{
		LCD_DRAW_POINT(uRow,uCol);
		xerr+=delta_x ;
		yerr+=delta_y ;
		if(xerr>distance)
		{
			xerr-=distance;
			uRow+=incx;
		}
		if(yerr>distance)
		{
			yerr-=distance;
			uCol+=incy;
		}
	}
}
```

18. HÀM VẼ TRỤC TỌA ĐỘ VÀ VÒNG TRÒN

- Tên hàm: Drow_Touch_Point (uint8_t x,uint16_t y)
- Thông số: thông số thứ nhất là tọa độ x, giá trị 8 bit, thông số thứ hai là tọa độ y, giá trị 16 bit.
- Chức năng: thực hiện 3 công việc:
  - Gọi hàm vẽ 2 đường thẳng vuông góc với nhau có khoảng cách 26 điểm với tâm là tọa độ (x,y).
  - Gọi 4 hàm để vẽ 4 điểm gần tâm tọa độ (x,y) nằm ở giữa góc vuông.
  - Gọi hàm vẽ vòng tròn tâm nằm ở tọa độ (x,y) có bán kính bằng 6.
- Hàm chi tiết như sau :

```
void Drow_Touch_Point(uint8_t x,uint16_t y)
{
```

```
        LCD_DrawLine(x-12,y,x+13,y);
        LCD_DrawLine(x,y-12,x,y+13);
        LCD_DrawPoint(x+1,y+1);
        LCD_DrawPoint(x-1,y+1);
        LCD_DrawPoint(x+1,y-1);
        LCD_DrawPoint(x-1,y-1);
        Draw_Circle(x,y,6);
}
```

19. HÀM VẼ HÌNH VUÔNG

- Tên hàm: LCD_DRAW_RECTANGLE (u8 X1, u16 Y1, u8 X2, u16 Y2)
- Thông số: các thông số là tọa độ (X1,Y1) và (X2,Y2).
- Chức năng: thực hiện vẽ hình vuông bằng cách vẽ 4 đường thẳng.
- Hàm chi tiết như sau :

```
void LCD_DRAW_RECTANGLE (u8 X1, u16 Y1, u8 X2, u16 Y2)
{
        LCD_DRAW_LINE(X1,Y1,X2,Y1);
        LCD_DRAW_LINE(X1,Y1,X1,Y2);
        LCD_DRAW_LINE(X1,Y2,X2,Y2);
        LCD_DRAW_LINE(X2,Y1,X2,Y2);
}
```

20. HÀM VẼ HÌNH TRÒN

- Tên hàm: LCD_DRAW_CIRCLE (uint8_t X0,uint16_t Y0,uint8_t R)
- Thông số: các thông số là tọa độ (X0,Y0) và bán kính R.
- Chức năng: thực hiện vẽ hình tròn.
- Hàm chi tiết như sau:

```
void LCD_DRAW_CIRCLE (uint8_t X0,uint16_t Y0,uint8_t R)
{
        int a,b;
        int di;
        a=0;b=R;
        di=3-(R<<1);
        while(a<=b)
        {
                LCD_DRAW_POINT(X0-b,Y0-a);          //3
                LCD_DRAW_POINT(X0+b,Y0-a);          //0
                LCD_DRAW_POINT(X0-a,Y0+b);          //1
                LCD_DRAW_POINT(X0-b,Y0-a);          //7
                LCD_DRAW_POINT(X0-a,Y0-b);          //2
                LCD_DRAW_POINT(X0+b,Y0+a);          //4
                LCD_DRAW_POINT(X0+a,Y0-b);          //5
                LCD_DRAW_POINT(X0+a,Y0+b);          //6
                LCD_DRAW_POINT(X0-b,Y0+a);
                a++;
```

```
            if(di<0)di +=4*a+6;
            else
            {
                di+=10+4*(a-b);
                b--;
            }
            LCD_DRAW_POINT(X0+a,Y0+b);
        }
}
```

21. HÀM HIỂN THỊ BIT MAP

- Tên hàm: LCD_DRAW_RECTANGLE (u8 X1, u16 Y1, u8 X2, u16 Y2)
- Thông số: các thông số là tọa độ (X1,Y1) và (X2,Y2).
- Chức năng: thực hiện vẽ hình vuông bằng cách vẽ 4 đường thẳng.
- Hàm chi tiết như sau : volume

```
void LCD_DRAW_RECTANGLE (u8 X1, u16 Y1, u8 X2, u16 Y2)
{
    LCD_DRAW_LINE(X1,Y1,X2,Y1);
    LCD_DRAW_LINE(X1,Y1,X1,Y2);
    LCD_DRAW_LINE(X1,Y2,X2,Y2);
    LCD_DRAW_LINE(X2,Y1,X2,Y2);
}
```

22. HÀM HIỂN THỊ SỐ TRÊN MÀN HÌNH GLCD

- Tên hàm: LCD_ShowNum(u8 x,u8 y,u32 num,u8 len,u8 size)
- Thông số: các thông số là tọa độ (x1,y1) và (x2,y2).
- Chức năng: hiển thị số có chiều dài 32 bit, có kích thước trên màn hình GLCD tại tọa độ x,y.
- Hàm chi tiết như sau :

```
void LCD_ShowNum(u8 x,u8 y,u32 num,u8 len,u8 size)
{
    u8 t,temp;
    u8 enshow=0;
    for(t=0;t<len;t++)
    {
        temp=(num/mypow(10,len-t-1))%10;
        if(enshow==0&&t<(len-1))
        {
            if(temp==0)
            {
                LCD_ShowChar(x+(size/2)*t,y,' ',size,0);
                continue;
            }else enshow=1;
        }
        LCD_ShowChar(x+(size/2)*t,y,temp+'0',size,0);
```

```
        }
}
```