Chương 4

CÁC BÀI THỰC HÀNH ARM CORTEX – M3 STM32F103VET GIAO TIẾP MÀN CẢM ỨNG TOUCH

- **GIỚI THIỆU**
- **KHẢO SÁT VI MẠCH ĐIỀU KHIỂN MÀN CẢM ỨNG TOUCH**
  CẤU TRÚC VI MẠCH ADS 7483
  ỨNG DỤNG VI MẠCH ADS 7483
  MÔ TẢ CHỨC NĂNG
  NGUYÊN LÝ HOẠT ĐỘNG
- **MẠCH GIAO TIẾP VI ĐIỀU KHIỂN VỚI MÀN CẢM ỨNG TOUCH**
  CÁC CHƯƠNG TRÌNH ĐIỀU KHIỂN LED BẰNG MÀN CẢM ỨNG
  CÁC CHƯƠNG TRÌNH ĐIỀU KHIỂN ẢNH DI CHUYỂN BẰNG MÀN CẢM ỨNG
  CÁC CHƯƠNG TRÌNH VẼ ẢNH 2D
- **CÁC HÀM ĐIỀU KHIỂN GLCD TRONG THƯ VIỆN <TV_TOUCH.C>**

I. GIỚI THIỆU

Chương này trình bày sơ đồ giao tiếp vi điều khiển ARM với màn cảm ứng touch và các hàm thư viện để sử dụng cho lập trình ứng dụng touch.

Màn cảm ứng touch có kích thước trùng với kích thước màn hình GLCD và được dán chồng lên màn hình GLCD.

Màn cảm ứng touch dùng để điều khiển giống như các điện thoại đang sử dụng.

Màn hình GLCD là thiết bị đầu ra dùng để hiển thị thông tin, màn cảm ứng là thiết bị đầu vào dùng để điều khiển.

Ở chương 3 ta đã sử dụng màn hình để hiển thị thông tin và màn hình có kích thước là 3,2 inch, khi muốn hiển thị ký tự hay hình ảnh ta phải xác định vị trí toạ độ x và y mà ta muốn hiển thị.

Trục x có chiều rộng 240 pixel và chiều dài là 320 pixel.

Màn cảm ứng touch có kích thước trùng với màn hình thì ta mới sử dụng được.

Tương tự như màn hình GLCD thì màn cảm ứng touch cũng có toạ độ x và y giống như vậy.

Khi ta chạm vào màn cảm ứng thì kết quả trả về là toạ độ của điểm chạm.

Để điều khiển bằng màn cảm ứng thì ta phải lập trình tạo ra một biểu tượng điều khiển ví dụ biểu tượng tắt mở loa hiển thị trên màn hình GLCD ở 1 vùng nào đó do ta quyết định.

Tiếp theo ta kiểm tra xem có nhấn touch hay không, nếu có thì tiếp tục kiểm tra có chạm vào các vị trí nằm trong vùng của biểu tượng điều khiển hay không, nếu đúng thì tiến hành điều khiển chức năng tương ứng.

II. KHẢO SÁT VI MẠCH ĐIỀU KHIỂN MÀN CẢM ỨNG TOUCH

1. *CẤU TRÚC VI MẠCH ADS 7483*

- Giao tiếp màn hình cảm ứng dùng 4 dây.
- Chuyển đổi tỉ lệ với đơn vị met.
- Nguồn cung cấp từ 2,7V đến 5V.
- Tốc độ chuyển đổi có thể lên đến 125kHz.
- Giao tiếp với vi điều khiển dạng nối tiếp SPI.
- Có thể lập trình 2 chế độ phân giải là 8 bit hoặc 12 bit.
- Có 2 ngõ vào tín hiệu tương tự phụ.
- Có chế độ điều khiển tiết kiệm nguồn.

2. *ỨNG DỤNG VI MẠCH ADS 7483*

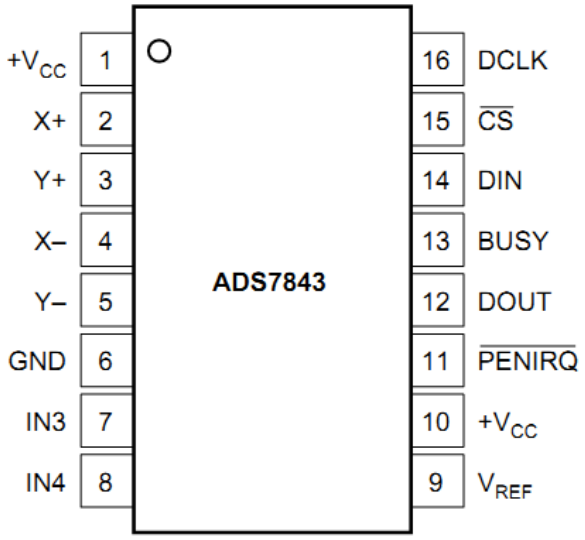
- Dùng trong các thiết bị trợ giúp số cá nhân
- Các thiết bị đo lường
- Các thiết bị đầu cuối
- Giám sát màn hình cảm ứng

3. *MÔ TẢ CHỨC NĂNG*

Vi mạch ADS 7843 là bộ chuyển đổi ADC 12 bit với giao tiếp nối tiếp đồng bộ và chuyển mạch điện trở thấp để điều khiển màn hình cảm ứng chạm. Tiêu tán công suất trung bình là 750µW tại tần số 125kHz và dùng nguồn 2,7V.

Điện áp tham chiếu (V_{REF}) có thể thay đổi từ 1V đến +Vcc dùng để cung cấp dãy điện áp ngõ vào tương ứng từ 0V đến V_{REF}. IC có tích hợp chức năng tắt nguồn để giảm công suất tiêu tán còn 0,5 µW. Vi mạch ADS7843 được chỉ định hoạt động ở nguồn 2,7V. Công suất tiêu tán thấp, hoạt động tần số cao và giao tiếp chuyển mạch trên bo làm cho vi mạch ADS7843 là vi mạch lý tưởng cho các hệ thống hoạt động với nguồn pin.

Sơ đồ chân như hình 4-1:

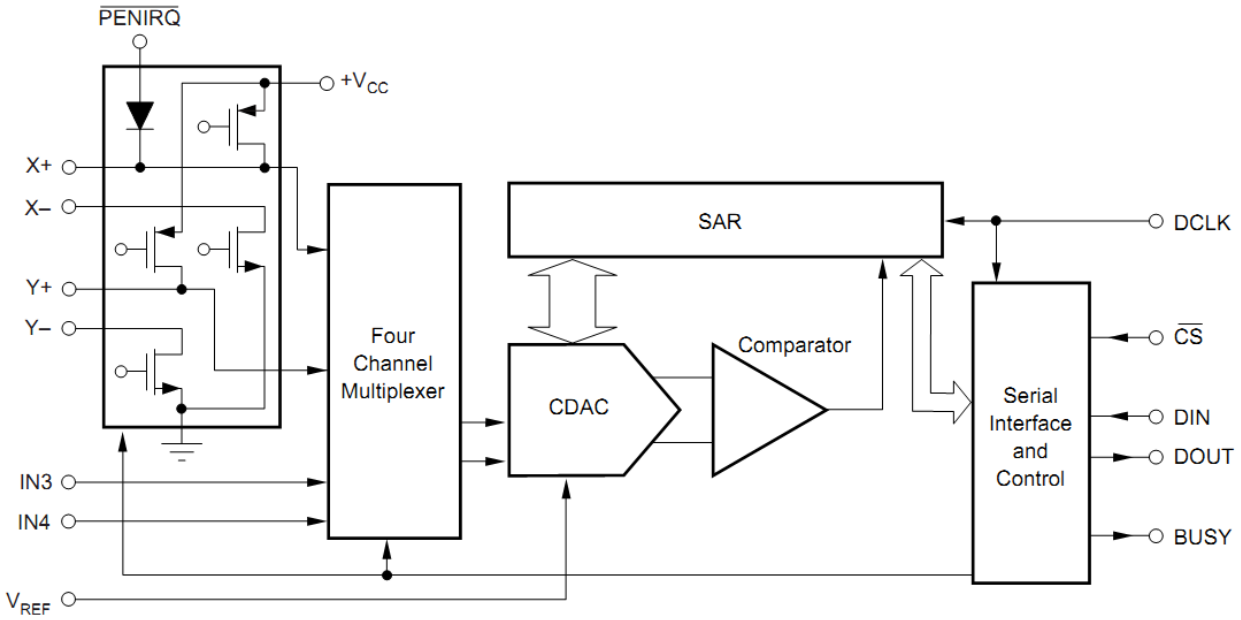


Hình 4-1. Sơ đồ chân IC ADS 7843.

Bảng 4-1. Chức năng các chân:

PIN	NAME	DESCRIPTION
1	$+V_{CC}$	Power Supply, 2.7V to 5V.
2	X+	X+ Position Input. ADC input Channel 1.
3	Y+	Y+ Position Input. ADC input Channel 2.
4	X–	X– Position Input
5	Y–	Y– Position Input
6	GND	Ground
7	IN3	Auxiliary Input 1. ADC input Channel 3.
8	IN4	Auxiliary Input 2. ADC input Channel 4.
9	V_{REF}	Voltage Reference Input
10	$+V_{CC}$	Power Supply, 2.7V to 5V.
11	$\overline{PENIRQ}$	Pen Interrupt. Open anode output (requires 10kΩ to 100kΩ pull-up resistor externally).
12	DOUT	Serial Data Output. Data is shifted on the falling edge of DCLK. This output is high impedance when $\overline{CS}$ is HIGH.
13	BUSY	Busy Output. This output is high impedance when $\overline{CS}$ is HIGH.
14	DIN	Serial Data Input. If $\overline{CS}$ is LOW, data is latched on rising edge of DCLK.
15	$\overline{CS}$	Chip Select Input. Controls conversion timing and enables the serial input/output register.
16	DCLK	External Clock Input. This clock runs the SAR conversion process and synchronizes serial data I/O.

Sơ đồ khối như hình 4-2:

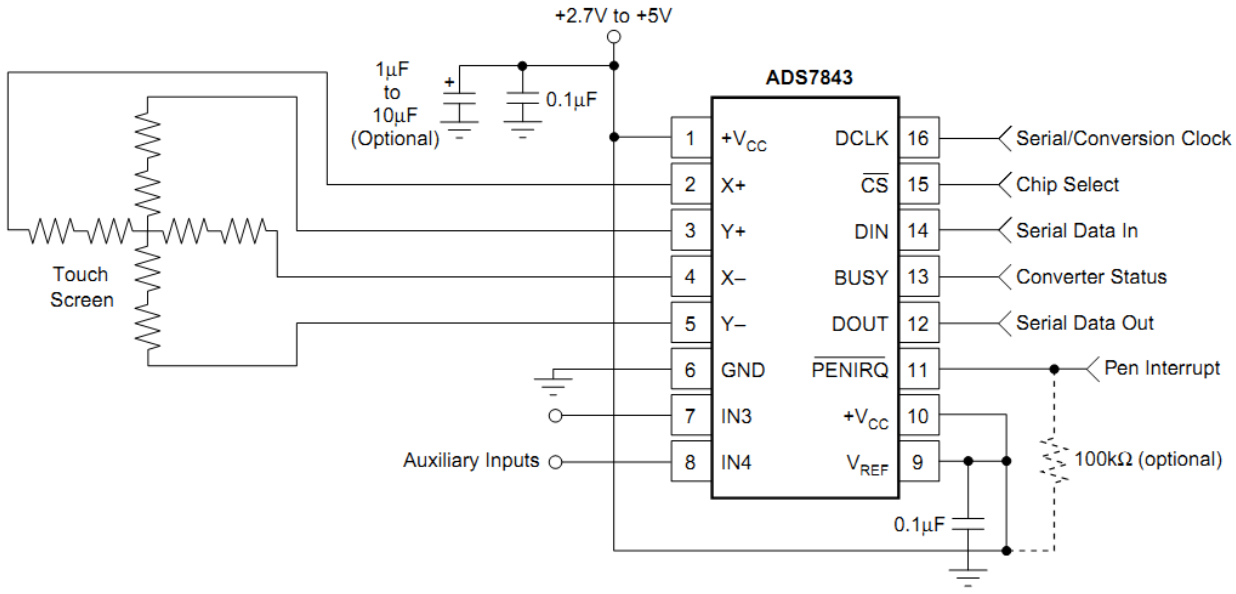


Hình 4-2. Sơ đồ khối IC ADS 7843.

4. NGUYÊN LÝ HOẠT ĐỘNG

Vi mạch ADS 7843 là bộ chuyển đổi ADC dùng phương pháp thanh ghi xấp xỉ gần đúng SAR.

Hoạt động cơ bản của IC 7843 được trình bày ở hình 4-3.



Hình 4-3. Sơ đồ kết nối mạch hoạt động.

Vi mạch yêu cầu sử dụng điện áp tham chiếu ngoài và nguồn xung clock ngoài. IC sử dụng nguồn đơn từ 2,7V đến 5,25V. Điện áp tham chiếu có thể thiết lập bất kỳ từ 1V đến +Vcc. Giá trị của điện áp tham chiếu được thiết lập trực tiếp dãy ngõ vào của bộ chuyển đổi. Dòng điện tham chiếu ngõ vào trung bình tùy thuộc vào tốc độ chuyển đổi.

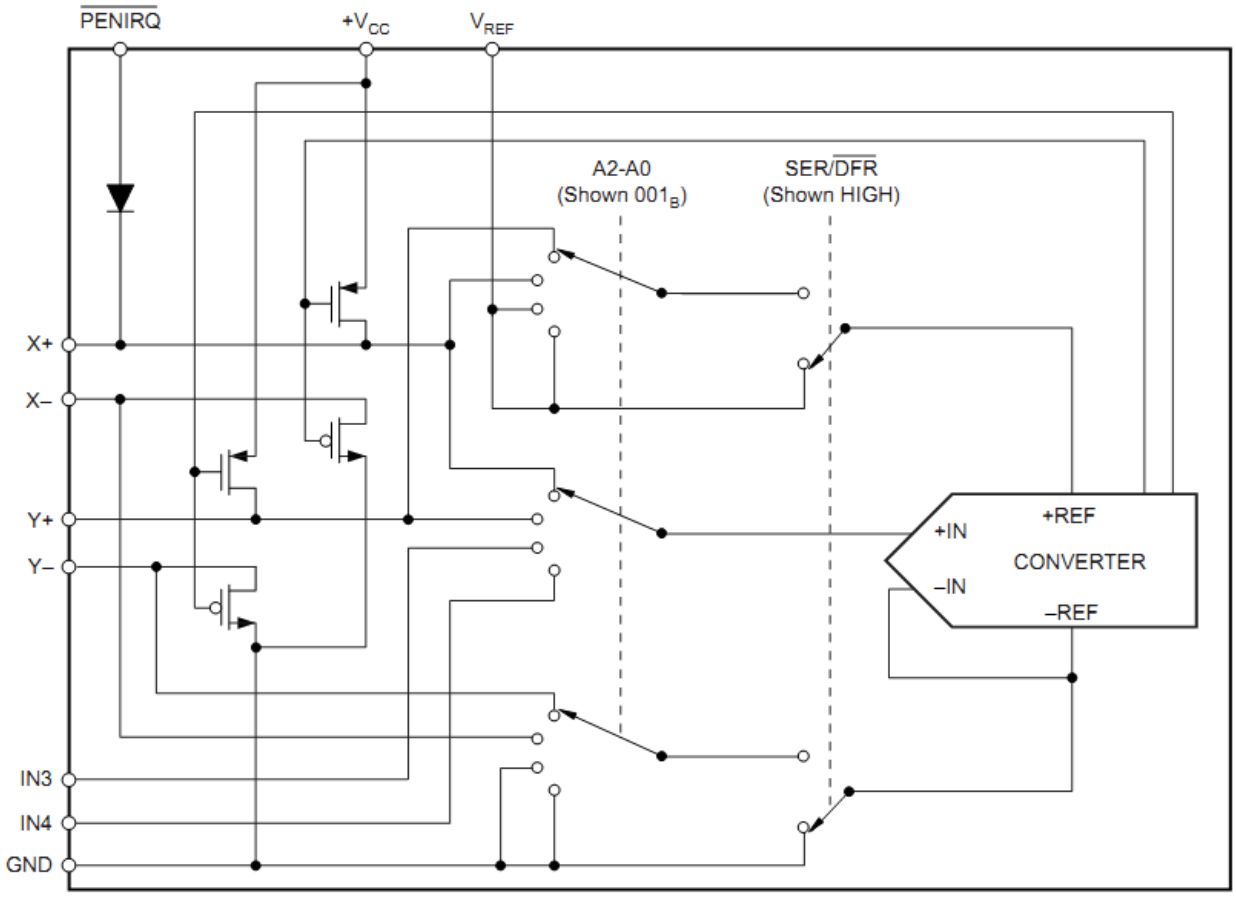
Ngõ vào tương tự đưa đến bộ chuyển đổi thông qua mạch đa hợp 4 kênh. Cấu hình duy nhất của chuyển mạch điện trở thấp cho phép kênh ngõ vào ADC không được chọn để cung cấp năng lượng và chân đi cùng tạo ra mass cho thiết bị bên ngoài.

Bằng cách giữ nguyên ngõ vào khác nhau đó đưa đến bộ chuyển đổi và kiến trúc điện áp tham chiếu khác nhau thì nó có thể phủ định sai số điện trở của switch.

a. Ngõ vào tương tự

Mạch đa hợp có sơ đồ khối như hình 4-x1, ngõ vào sai biệt của ADC và bộ chuyển đổi với điện áp tham chiếu sai biệt. Bảng 1 và bảng 2 trình bày mối quan hệ giữa A2, A1, A0 và bit điều khiển SER/$\overline{DFR}$ và cấu hình của IC ADS7843. Bit điều khiển được cung cấp nối tiếp thông qua chân DIN – xem phần giao tiếp số của tài liệu để có thêm chi tiết.

Khi bộ chuyển đổi vào trạng thái chờ, điện áp khác nhau giữa các ngõ vào +IN và –IN (xem hình 4-x1) được bắt lấy bởi dãy tụ bên trong. Dòng điện vào của các ngõ vào tương tự phụ thuộc vào tốc độ chuyển đổi của vi mạch. Trong khoảng thời gian lấy mẫu, nguồn tín hiệu vào phải được nạp vào tụ lấy mẫu bên trong (thường là 25pF). Sau khi tụ lấy mẫu đã được nạp đầy, thì dòng ngõ vào bằng 0. Tốc độ nạp từ nguồn tương tự đến bộ chuyển đổi là hàm của tốc độ chuyển đổi.



Hình 4-4. Sơ đồ khối đơn giản của khối ngõ vào tương tự.

Bảng 4-2. Cấu hình ngõ vào, chế độ tham chiếu nguồn đơn (SER/$\overline{DFR}$ ở mức 1)

A2	A1	A0	X+	Y+	IN3	IN4	–IN(1)	X SWITCHES	Y SWITCHES	+REF(1)	–REF(1)
0	0	1	+IN				GND	OFF	ON	$+V_{REF}$	GND
1	0	1		+IN			GND	ON	OFF	$+V_{REF}$	GND
0	1	0			+IN		GND	OFF	OFF	$+V_{REF}$	GND
1	1	0				+IN	GND	OFF	OFF	$+V_{REF}$	GND

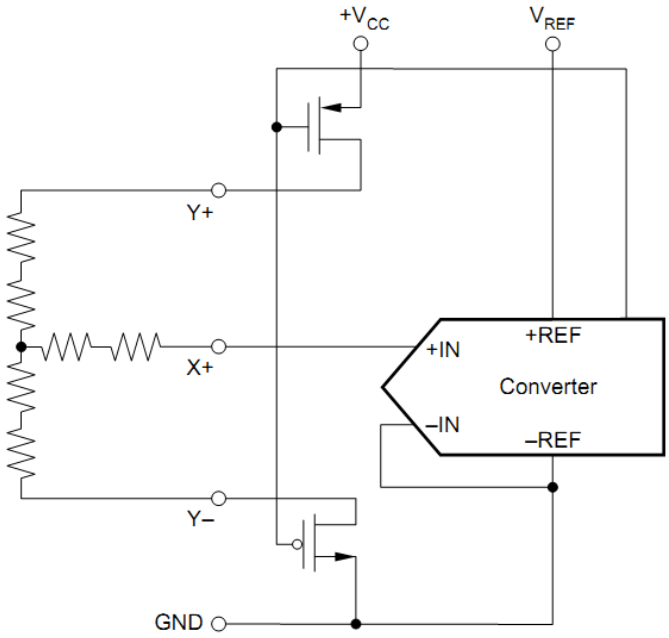
Bảng 4-3. Cấu hình ngõ vào, chế độ tham chiếu vi sai (SER/$\overline{DFR}$ ở mức 0)

A2	A1	A0	X+	Y+	IN3	IN4	–IN(1)	X SWITCHES	Y SWITCHES	+REF(1)	–REF(1)
0	0	1	+IN				–Y	OFF	ON	+Y	–Y
1	0	1		+IN			–X	ON	OFF	+X	–X
0	1	0			+IN		GND	OFF	OFF	$+V_{REF}$	GND
1	1	0				+IN	GND	OFF	OFF	$+V_{REF}$	GND

b. *Ngõ vào tham chiếu*

Điện áp khác biệt giữa +REF và –REF trong hình 4-x1 thiết lập dãy ngõ vào tương tự. IC ADS7843 sẽ hoạt động với nguồn điện áp tham chiếu nằm trong dãy từ 1V đến +Vcc. Có các thành phần tới hạn cần quan tâm đến ngõ vào tham chiếu và độ rộng dãy điện áp của điện áp tham chiếu. Khi điện áp tham chiếu giảm thì trọng số điện áp tương tự của mỗi mã nhị phân ngõ ra cũng giảm. Giá trị điện áp này được xem là độ phân giải và bằng điện áp tham chiếu chia cho 4096. Bất kỳ điện áp lệch và lỗi hệ số khuếch đại vốn có trong ADC sẽ làm tăng độ phân giải cho dù điện áp tham chiếu giảm. Ví dụ với điện áp lệch đã cho của bộ chuyển đổi là 2LSB (giá trị điệp áp tương ứng với bit LSB thay đổi chính là độ phân giải) với điện áp tham chiếu là 2,5V thì nó sẽ tăng lên là 5LSB với điện áp tham chiếu là 1V. Trong trường hợp này điện áp lệch chính xác của vi mạch là 1,22mV. Với điện áp tham chiếu càng thấp thì cần phải quan tâm đến nhiễu mức thấp, độ gợn sóng của nguồn cung cấp, nhiễu của điện áp tham chiếu và nhiễu của tín hiệu vào.

Điện áp đưa vào ngõ vào V_{REF} không được đệm và điều khiển trực tiếp thành phần bộ chuyển đổui tụ ADC (CADC) của IC ADS7843. Thường thì dòng ngõ vào là 13µA tại điện áp tham chiếu là 2,7V và tần số lấy mẫu là 125kHz. Giá trị này thay đổi vài µA tùy thuộc vào kết quả chuyển đổi. Dòng điện tham chiếu giảm cùng với tốc độ chuyển đổi và điện áp tham chiếu.



Hình 4-5. Sơ đồ khối đơn giản của nguồn tham chiếu đơn

*(*SER/ $\overline{DFR}$ ở mức 1, các switch Y được phép, X+ là ngõ vào tương tự*).*

Đại lượng đo vị trí Y hiện tại của thiết bị đang chỉ được tạo ra bằng cách nối ngõ vào X+ với ADC, mở các bộ thúc Y+ và Y- và số hóa điện áp trên X+ (xem hình 4-x2). Để đo giá trị này thì điện trở ở đầu X+ không được làm ảnh hưởng đến bộ chuyển đổi (nó không làm ảnh hưởng đến thời gian thiết lập nhưng điện trở có giá trị nhỏ đủ không làm ảnh hưởng).

Tuy nhiên do điện trở giữa Y+ và Y- khá nhỏ, điện trở của bộ thúc Y tạo ra sự sai lệch nhỏ. Trong trường hợp nằm ngoài khá xa, thì nó không thể đạt ngõ vào với điện áp 0V hoặc ngõ vào đạt giá trị cực đại bất chấp vị trí chạm vào thiết bị trên màn cảm ứng bởi vì một phần điện áp rơi trên các

switch bên trong. Ngoài ra, điện trở switch bên trong thì không giống với rãnh điện trở trên màn cảm ứng chạm tạo ra nguồn sai số mở rộng.

Trường hợp này có thể được làm giảm bớt như ở hình 4-x3.

Bằng cách thiết lập bit SER/$\overline{DFR}$ ở mức thấp, các ngõ vào +REF và –REF được nối đến ngõ vào Y+ và Y-. Điều này làm bộ ADC tuyến tính. Kết quả chuyển đổi luôn là phần trăm của điện trở bên ngoài, bất chấp nó thay đổi tỷ lệ với điện trở của switch bên trong như thế nào.

III. MẠCH GIAO TIẾP VI ĐIỀU KHIỂN VỚI MÀN CẢM ỨNG TOUCH

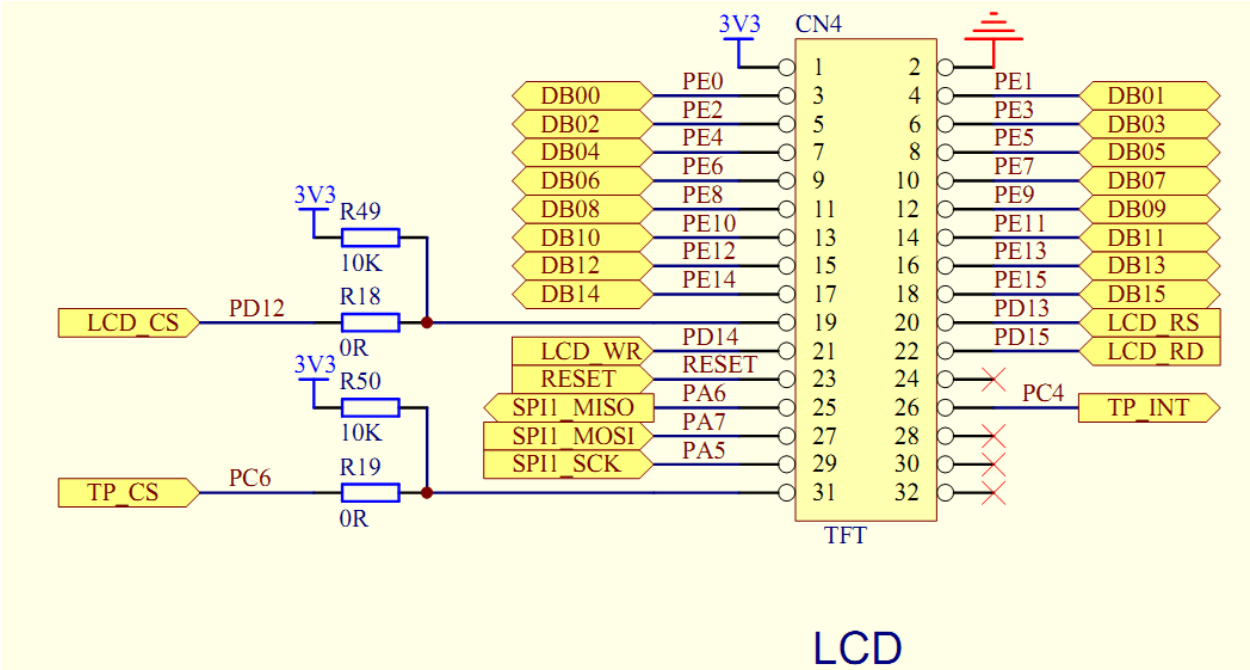
Sơ đồ mạch giao tiếp như hình 4-6, các tín hiệu giao tiếp vi điều khiển ARM với màn cảm ứng touch gồm 5 tín hiệu:

TP_CS dùng port PC6.

TP_INT dùng port PC4.

Giao tiếp SPI : SPI1_MISO dùng port PA6, SPI1_MOSI dùng port PA7, SPI1_SCK dùng port PA5.

Connector CN4 vừa giao tiếp màn hình GLCD và touch.



Hình 4-6. Sơ đồ mạch vi điều khiển ARM giao tiếp GLCD và touch.

Chip sử dụng điều khiển màn hình cảm ứng là ADS7483.

IV. CÁC CHƯƠNG TRÌNH ỨNG DỤNG

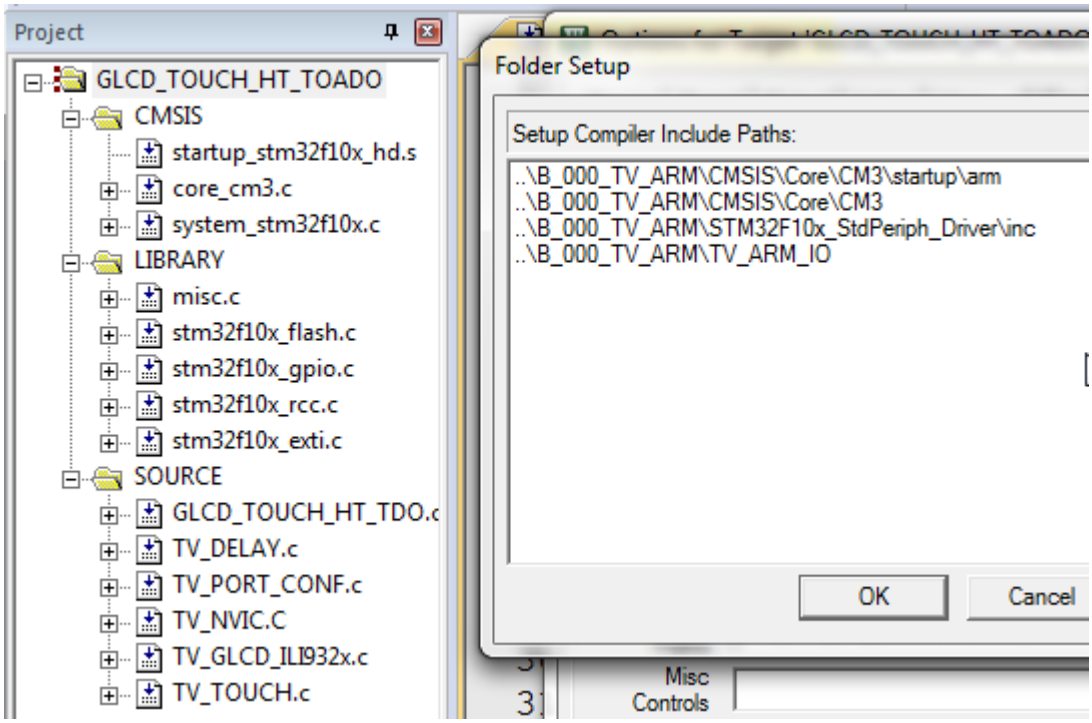
1. CÁC CHƯƠNG TRÌNH ĐIỀU KHIỂN LED BẰNG MÀN CẢM ỨNG

Phần này thực hành các bài điều khiển tắt mở LED đơn bằng màn cảm ứng, hiển thị trên GLCD. Các bài thực hành bao gồm điều khiển led mở led tắt bằng 2 nút nhấn ON và OFF, điều khiển bằng biểu tượng bitmap, điều khiển bằng 1 biểu tượng ON/OFF.

Bài mẫu 401. Chương trình hiển thị toạ độ điểm chạm của màn cảm ứng touch trên màn hình GLCD 320×240.

Tạo thư mục "BAI_401_TOUCH_XY_GLCD" để lưu project.

a. Mục đích: làm quen với các hàm giao tiếp màn cảm ứng touch.
b. Lưu đồ: khởi tạo port giao tiếp với touch, GLCD, ngắt, hiển thị toạ độ điểm chạm.
c. Hình các file chính và thư viện:



Hình 4-7. Các nhóm lưu thư mục và đường dẫn bài 401.

d. Chương trình:

```
#include      "stm32f10x.h"
#include      "hardware_conf.h"
#include      "TV_PORT_CONF.h"
#include      "TV_GLCD_ILI932x.h"
#include      "TV_TOUCH.h"
#include      "TV_DELAY.h"
#include      "TV_NVIC.h"
u8 TT_TOUCH;
int main(void)
{
            SystemInit();     PORT_CONF();
            NVIC_CONFIGURATION_TOUCH();
            LCD_INIT();             LCD_CLEAR(WHITE);
            LCD_SHOW_STRING_X(10,10,"HIEN THI TOA DO DIEM CHAM");
            TOUCH_INIT();
            while(1)
            {
                        TT_TOUCH=TOUCH_PRESS();
                        if(TT_TOUCH==1)
```

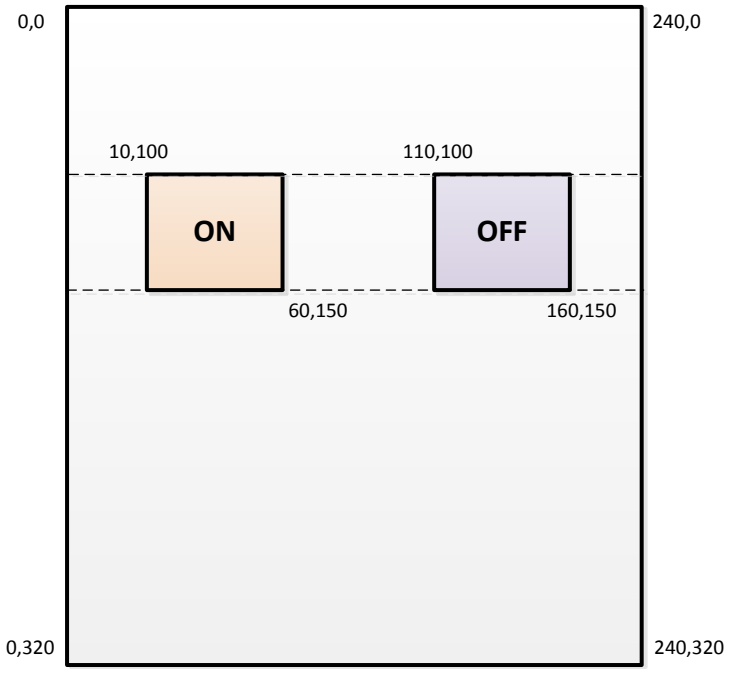
```
                    {
                        LCD_SHOW_NUM(10,40,Pen_Point.X0,3,16);
                        LCD_SHOW_NUM(40,40,Pen_Point.Y0,3,16);
                    }
                }
}
```

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, LCD, touch, khởi tạo ngắt phục vụ cho giao tiếp màn cảm ứng touch. Kiểm tra xem có chạm màn hình hay không, nếu có thì thực hiện chuyển đổi ra toạ độ và gọi hàm hiển thị toạ độ của X và Y.
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.

Bài mẫu 402. Chương trình điều khiển led sáng tắt bằng 2 nút nhấn ON và OFF được tạo trên màn hình GLCD và màn cảm ứng touch. Khi nhấn ON thì led sáng, khi nhấn OFF thì led tắt.

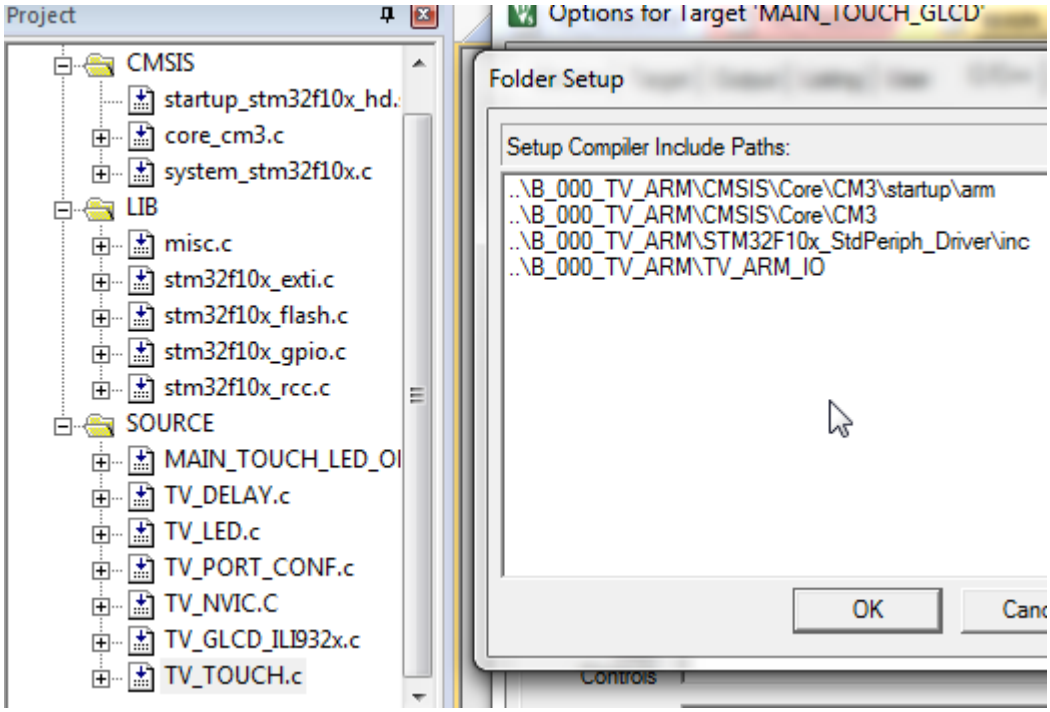
Tạo thư mục "BAI_402_TOUCH_GLCD_LED_2_ONOFF" để lưu project.

a. Mục đích: điều khiển led bằng màn cảm ứng touch.
b. Lưu đồ:
Bước 1: Khởi tạo: dao động, port, ngắt, GLCD, touch. Led.
Bước 2: Tiến hành vẽ 2 ô vuông trên màn hình GLCD tượng trưng cho 2 nút nhấn: ON, OFF. Các thông số cho trên màn hình.
Bước 3: Kiểm tra toạ độ điểm nhấn nếu nằm trong vùng " ON" thì làm led sáng, nếu led đã sáng thì kiểm tra điểm nhấn nếu nằm trong vùng " OFF" thì làm led tắt.
c. Hình toạ độ các nút ON và OFF trên màn hình:



Hình 4-8. Vị trí các nút ON và OFF của bài 402.

d. Hình các file chính và thư viện:



Hình 4-9. Các nhóm lưu thư mục và đường dẫn bài 402.

e. Chương trình:

```
#include "stm32f10x.h"
#include "hardware_conf.h"
#include "TV_PORT_CONF.h"
#include "TV_GLCD_ILI932x.h"
#include "TV_TOUCH.h"
#include "TV_DELAY.h"
#include "TV_LED.h"
#include "TV_NVIC.h"
uint8_t  TT_LED=0,TT_TOUCH;
const unsigned char X_ON1=10,X_ON2=60;
const unsigned int Y_ON1=100,Y_ON2=150;
const unsigned char X_OFF1=110,X_OFF2=160;
const unsigned int Y_OFF1=100,Y_OFF2=150;
/*********************************************************
CHUONG TRINH CHINH
*********************************************************/
int main(void)
{
                SystemInit();       PORT_CONF();
                NVIC_CONFIGURATION_TOUCH();
                LCD_INIT();         TOUCH_INIT();           LED_INIT_2();

                LCD_CLEAR(YELLOW);
                BACK_COLOR=YELLOW;
                POINT_COLOR=RED;

                LCD_SHOW_STRING_X(10,10,"DIEU KHIEN LED ON OFF");
                LCD_DRAW_RECTANGLE(X_ON1,Y_ON1,X_ON2,Y_ON2);
                LCD_DRAW_RECTANGLE(X_OFF1,Y_OFF1,X_OFF2,Y_OFF2);
```

```
            LCD_SHOW_STRING_X(X_ON1+10,Y_ON1+10," ON");
            LCD_SHOW_STRING_X(X_OFF1+10,Y_OFF1+10,"OFF");
            while(1)
            {
                        TT_TOUCH=TOUCH_PRESS();
                        if (TT_TOUCH==1)
                        {
                              LCD_SHOW_NUM(10,40,Pen_Point.X0,5,16);
                              LCD_SHOW_NUM(10,60,Pen_Point.Y0,5,16);

                              if ((Pen_Point.X0>X_ON1)&&(Pen_Point.X0<X_ON2)&&
                                    (Pen_Point.Y0>Y_ON1)&&(Pen_Point.Y0<Y_ON2))
                              {
                                    GPIO_SetBits(GPIOD, GPIO_Pin_8);
                              }
                              else
                              if ((Pen_Point.X0>X_OFF1)&&(Pen_Point.X0<X_OFF2)&&
                                    (Pen_Point.Y0>Y_OFF1)&&(Pen_Point.Y0<Y_OFF2))
                              {
                                    GPIO_ResetBits(GPIOD, GPIO_Pin_8);}
                              }
                        }
}
```

f. Tiến hành biên dịch và nạp.

g. Quan sát kết quả: nhấn ON thì led sáng, nhấn OFF thì led tắt.

h. Giải thích chương trình: gọi hàm kiểm tra có chạm touch không, nếu có thì trả về trạng thái là 1, nếu không thì trả về trạng thái là 0. Khi bằng 1 thì kiểm tra xem 2 toạ độ có nằm trong giới hạn của phím ON không, nếu đúng thì mở đèn, nếu không thì kiểm tra nằm trong phạm vi của phím OFF không nếu đúng thì tắt đèn. Nếu không đúng toạ độ thì không làm gì.

i. 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 403. Chương trình điều khiển led sáng tắt bằng 1 nút nhấn ON_OFF được tạo trên màn hình GLCD và màn cảm ứng touch. Khi led tắt thì nút điều khiển hiển thị chữ ON và nếu nhấn thì làm led sáng và nút điều khiển hiển thị chữ OFF thay cho chữ ON và nếu nhấn thì làm led tắt rồi hiển thị lại chữ ON.

Tạo thư mục "BAI_403_TOUCH_GLCD_LED_1_ONOFF" để lưu project.

Bài mẫu 404. Chương trình điều khiển led sáng tắt bằng 2 bitmap hình ngọn đèn màu xanh và màu đen. Khi nhấn bitmap ngọn đèn màu xanh thì led sáng, khi nhấn bitmap ngọn đèn màu đen thì led tắt.

Tạo thư mục "BAI_404_TOUCH_GLCD_LED_2_BITMAP" để lưu project.

a. Mục đích: biết vẽ và điều khiển bằng biểu tượng trên màn cảm ứng.

b. Lưu đồ:

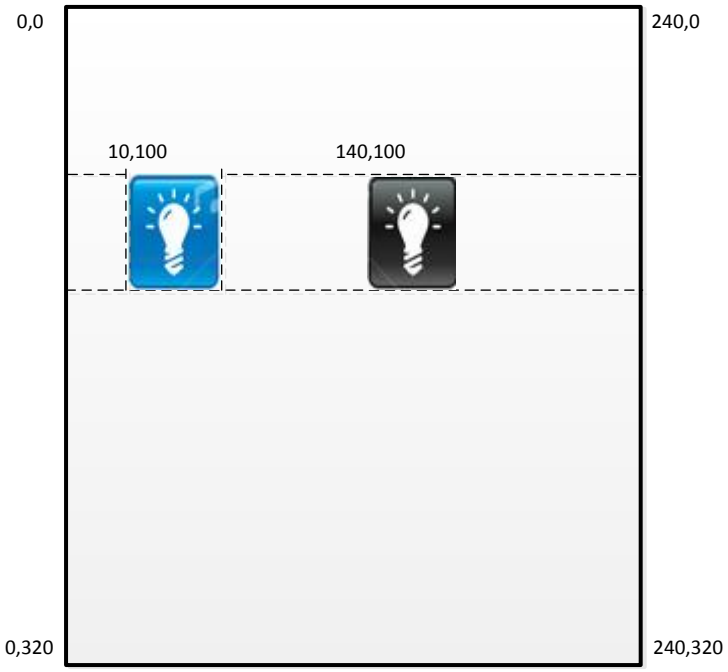
Bước 1: Khởi tạo: dao động, port, ngắt, GLCD, touch. Led.

Bước 2: Tiến hành vẽ 2 ô vuông trên màn hình GLCD tượng trưng cho 2 nút nhấn: ON, OFF.

Bước 3: Kiểm tra nếu nhấn bitmap màu xanh thì làm led sáng, nếu không thì chờ.

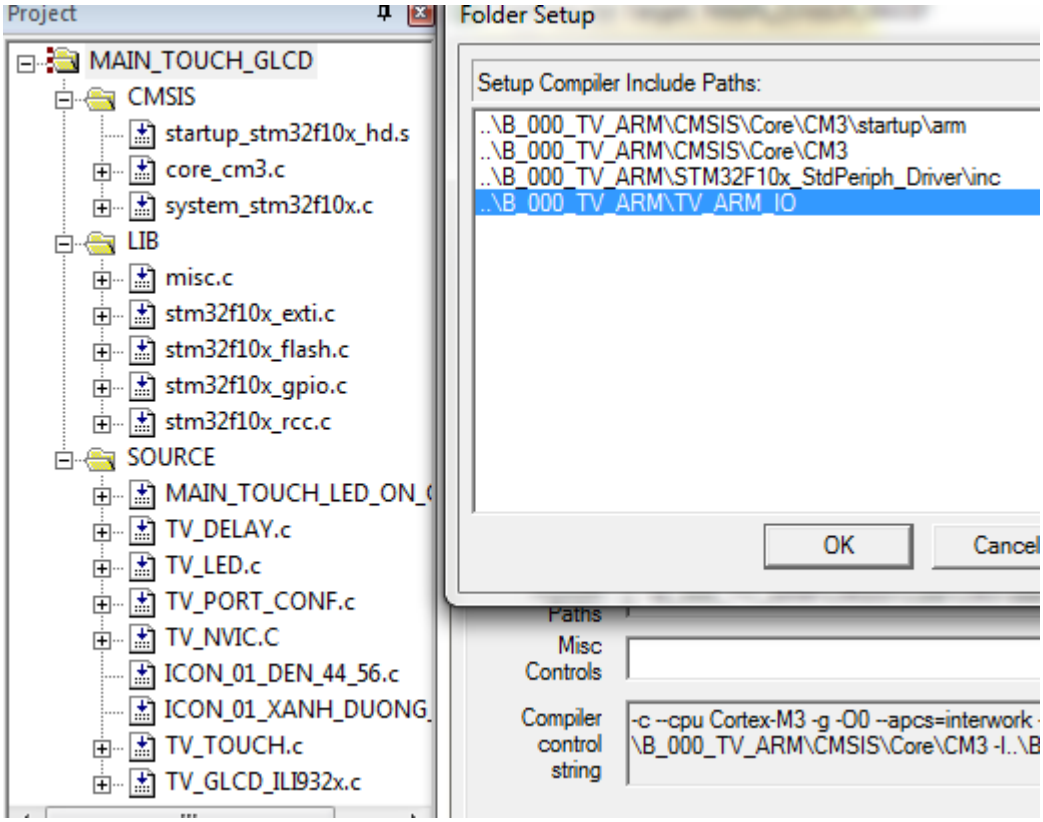
Bước 4: Sau khi nhấn đèn led sáng thì kiểm tra nếu nhấn bitmap màu đen thì làm led tắt, nếu không thì chờ.

c. Hình toạ độ các nút ON và OFF trên màn hình:



Hình 4-10. Vị trí các 2 bitmap.

d. Hình các file chính và thư viện:



Hình 4-11. Các nhóm lưu thư mục và đường dẫn bài 404.

e. Chương trình:

```
#include "stm32f10x.h"
#include "hardware_conf.h"
#include "TV_PORT_CONF.h"
#include "TV_GLCD_ILI932x.h"
#include "TV_TOUCH.h"
#include "TV_DELAY.h"
#include "TV_LED.h"
#include "TV_NVIC.h"

extern u8 gImage_ICON_01_DEN_44_56[];
extern u8 gImage_ICON_01_XANH_DUONG_47_58[];

uint8_t  TT_LED=0,RONG_X=50;
uint16_t TT_TOUCH,DAI_Y=50;

uint8_t   ONX1=10,   ONX2;
uint16_t ONY1=100, ONY2;
uint8_t   OFFX1=140,OFFX2;
uint16_t OFFY1=100,OFFY2;

/*********************************************************
CHUONG TRINH CHINH
*********************************************************/
int main(void)
{
                SystemInit();       PORT_CONF();
```

```
		NVIC_CONFIGURATION_TOUCH();
		LCD_INIT();	TOUCH_INIT();		LED_INIT_2();

		LCD_CLEAR(YELLOW);
		BACK_COLOR=YELLOW;
		POINT_COLOR=RED;

		ONX2=ONX1+RONG_X;
		ONY2=ONY1+DAI_Y;
		OFFX2=OFFX1+RONG_X;
		OFFY2=OFFY1+DAI_Y;

		LCD_SHOW_STRING_X(10,10,"DIEU KHIEN LED ON OFF");
		LCD_WRITE_BITMAP_Y(ONX1,ONY1,47,58, gImage_ICON_01_XANH_DUONG_47_58);
		LCD_WRITE_BITMAP_Y(OFFX1,OFFY1,44,56, gImage_ICON_01_DEN_44_56);

		while(1)
		{
			TT_TOUCH=TOUCH_PRESS();
			if (TT_TOUCH==1)
			{
				LCD_SHOW_NUM(10,40,Pen_Point.X0,5,16);
				LCD_SHOW_NUM(10,60,Pen_Point.Y0,5,16);

				if ((Pen_Point.X0>ONX1)&&(Pen_Point.X0<ONX2)&&
				 (Pen_Point.Y0>ONY1)&&(Pen_Point.Y0<ONY2))
				{
					GPIO_SetBits(GPIOD, GPIO_Pin_8);
				}
				else
				if ((Pen_Point.X0>OFFX1)&&(Pen_Point.X0<OFFX2)&&
				 (Pen_Point.Y0>OFFY1)&&(Pen_Point.Y0<OFFY2))
				{
					GPIO_ResetBits(GPIOD, GPIO_Pin_8);}
				}
			}
}
```

f. Tiến hành biên dịch và nạp.

g. Quan sát kết quả: nhấn lên hình đèn led màu xanh thì led sáng, nhấn lên biểu tượng màu đen thì led tắt.

h. Giải thích chương trình:

i. 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 405. Chương trình điều khiển led sáng tắt bằng 1 bitmap được tạo trên màn hình GLCD và màn cảm ứng touch. Khi led tắt thì nút điều khiển hiển thị chữ bitmap màu xanh và nếu nhấn thì làm led sáng, sau đó nút điều khiển hiển thị bitmap màu đen và nếu nhấn thì làm led tắt rồi hiển thị lại bitmap màu xanh.

Tạo thư mục "BAI_405_TOUCH_GLCD_LED_1_BITMAP" để lưu project.

Bài tập 406. Giống bài 405 nhưng thêm có 2 bitmap để điều khiển 2 led, bitmap màu xanh dương điều khiển led 1, bitmap màu xanh lá điều khiển led 2, bitmap màu đen dung chung để tắt đèn. Phân chia màn hình cho đồng đều.

Tạo thư mục "BAI_406_TOUCH_GLCD_2LED_3BITMAP" để lưu project.

Bài tập 407. Giống bài 406 nhưng thêm 2 bitmap màu đỏ điều khiển led 3 và màu cam để điều khiển led 4. Phân chia màn hình cho đồng đều.

Tạo thư mục "BAI_407_TOUCH_GLCD_4LED_5BITMAP" để lưu project.

2. CÁC CHƯƠNG TRÌNH ĐIỀU KHIỂN ẢNH DI CHUYỂN BẰNG MÀN CẢM ỨNG

Phần này thực hành các bài điều khiển ảnh di chuyển trên màn hình GLCD.

Bài mẫu 411. Chương trình điều khiển 1 bitmap di chuyển trên màn hình: cho hiển thị 1 ảnh bitmap trên màn hình và sau đó chạm ở vị trí nào thì ảnh di chuyển đến vị trí đó.

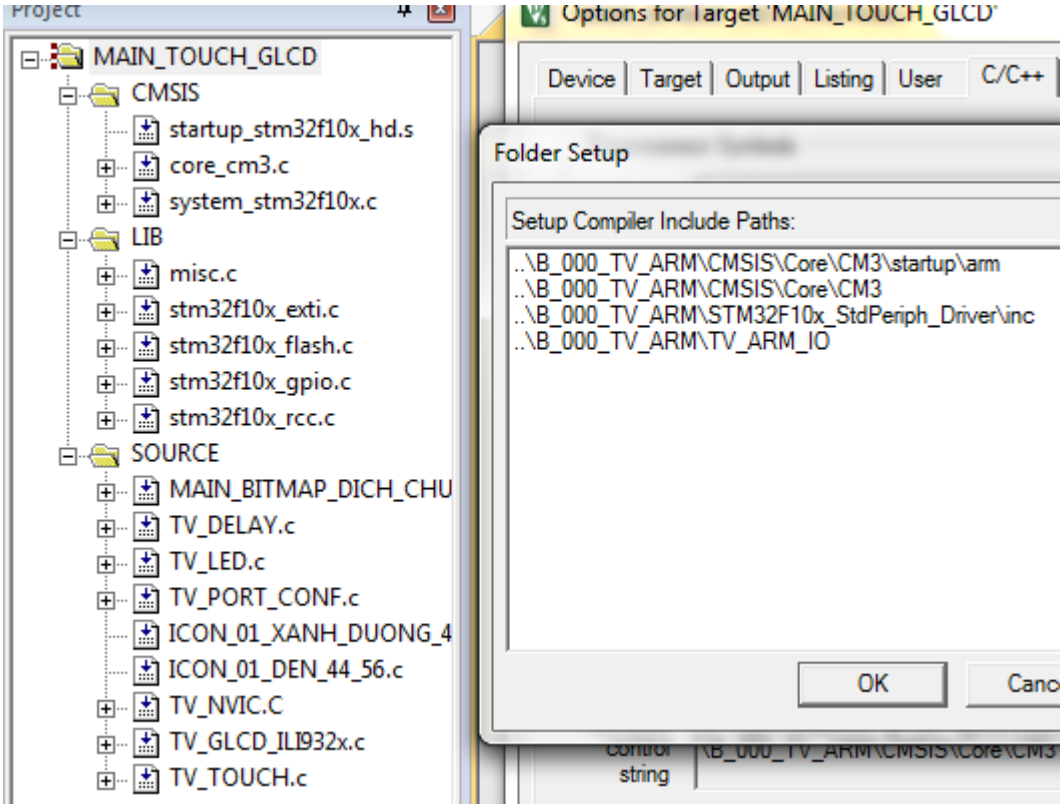
Tạo thư mục "BAI_411_TOUCH_GLCD_BITMAP_DC_1" để lưu project.

a. Mục đích : biết lập trình thay đổi vị trí hiển di chuyển của bitmap.

b. Lưu đồ:

Bước 1: Khởi tạo: dao động, port, ngắt, GLCD, touch. Led.
Bước 2: Tiến hành hiển thị bitmap.
Bước 3: Kiểm tra nếu chạm touch thì xoá hình ảnh trước đó, tính toán vị trí mới và hiển thị lại bitmap.

c. Hình các file chính và thư viện:



Hình 4-12. Các nhóm lưu thư mục và đường dẫn bài 404.

d. Chương trình:

```
#include "stm32f10x.h"
#include "hardware_conf.h"
#include "TV_PORT_CONF.h"
#include "TV_GLCD_ILI932x.h"
#include "TV_TOUCH.h"
#include "TV_DELAY.h"
#include "TV_LED.h"
#include "TV_NVIC.h"

extern u8 gImage_ICON_01_XANH_DUONG_47_58[];
uint8_t   VTX1=0,VTX2,TT_TOUCH,RONG_X=47;
uint16_t VTY1=0,VTY2,DAI_Y=58;
/*********************************************************
HAM XOA HINH BITMAP TRUOC KHI DI CHUYEN
*********************************************************/
void XOA_BITMAP()
{
        uint8_t XC;
        uint16_t YC;
        for(YC=0;YC<VTY1+DAI_Y;YC++)
                                {
                                        for(XC=VTX1;XC<VTX1+RONG_X;XC++)
                                        {
                                                LCD_DRAW_POINT(XC,YC);
                                        }
                                }
}
```

```
/*********************************************************
CHUONG TRINH CHINH
*********************************************************/
int main(void)
{
        SystemInit();   PORT_CONF();
        NVIC_CONFIGURATION_TOUCH();
        LCD_INIT();     TOUCH_INIT();       LED_INIT_2();

        LCD_CLEAR(YELLOW);
        BACK_COLOR=YELLOW;
        POINT_COLOR=YELLOW;

        VTX2=VTX1+RONG_X;
        VTY2=VTY1+DAI_Y;

        LCD_WRITE_BITMAP_Y(VTX1,VTY1,RONG_X,DAI_Y,
gImage_ICON_01_XANH_DUONG_47_58);
        while(1)
        {
                TT_TOUCH=TOUCH_PRESS();
                if (TT_TOUCH==1)
                {
                        if (( Pen_Point.X0<200) &&(Pen_Point.Y0<280))

                        {
                                XOA_BITMAP();
                                VTX1=Pen_Point.X0;
                                VTY1=Pen_Point.Y0;
   LCD_WRITE_BITMAP_Y(VTX1,VTY1,RONG_X,DAI_Y, gImage_ICON_01_XANH_DUONG_47_58);

                        }
                }
        }
}
```

e. Tiến hành biên dịch và nạp.
f. Quan sát kết quả: sau khi hiển thị bitmap ta chạm vị trí nào trên màn hình thì sau đó ảnh sẽ di chuyển đến vị trí đó.
g. Giải thích chương trình:
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 mẫu 412. Chương trình điều khiển 1 bitmap di chuyển trên màn hình: cho hiển thị 1 ảnh bitmap trên màn hình và sau đó chạm ở vị trí nào thì ảnh di chuyển đến vị trí đó và kéo điểm chạm di chuyển thì ảnh di chuyển theo.

Tạo thư mục "BAI_412_TOUCH_GLCD_BITMAP_DC_2" để lưu project.

a. Mục đích: biết lập trình kéo bitmap di chuyển.

b. Lưu đồ:

Bước 1: Khởi tạo: dao động, port, ngắt, GLCD, touch. Led.

Bước 2: Tiến hành hiển thị bitmap.

Bước 3: Kiểm tra nếu chạm touch thì xoá hình ảnh trước đó, tính toán vị trí mới và hiển thị lại bitmap.

c. Chương trình:

```
#include "stm32f10x.h"
#include "hardware_conf.h"
#include "TV_PORT_CONF.h"
#include "TV_GLCD_ILI932x.h"
#include "TV_TOUCH.h"
#include "TV_DELAY.h"
#include "TV_LED.h"

extern u8 gImage_ICON_01_DEN_44_56[];
extern u8 gImage_ICON_01_XANH_DUONG_47_58[];

uint8_t  TT_LED=0,RONG_X;
uint16_t TT_TOUCH,DAI_Y;

uint8_t VTX0_1=0,VTX0_2,VTX0_SS=0;
uint16_t VTY0_1=0,VTY0_2,VTY0_SS=0;
uint8_t VTX1_1=0,VTX1_2;
uint16_t VTY1_1=0,VTY1_2;

/*********************************************************
HAM XOA HINH BITMAP TRUOC KHI DI CHUYEN
*********************************************************/
void XOA_BITMAP()
{
        uint8_t XC;
        uint16_t YC;
        for(XC=VTX0_1;XC<VTX0_1+RONG_X;XC++)
        {
                for(YC=VTY0_1;YC<VTY0_1+DAI_Y;YC++)
                {
                        LCD_DRAW_POINT(XC,YC);
                }
        }
}
/*********************************************************
HAM KIEM TRA CO CHAM TOUCH THI DICH BITMAP
*********************************************************/
void TOUCH_PRESS_DICHUYEN()
{
   if(Pen_Point.Key_Sta==Key_Down)
   {
        do{
                TOUCH_ADS7483_CONVERT_POSITION();
```

```
                Pen_Point.Key_Sta=Key_Up;
                DELAY_MS(1);
                if (( Pen_Point.X0!=VTX0_SS) &&(Pen_Point.Y0!=VTY0_SS))

                {
                        XOA_BITMAP();
                        VTX0_SS=Pen_Point.X0;
                        VTY0_SS=Pen_Point.Y0;

                        VTX0_1=Pen_Point.X0;
                        VTY0_1=Pen_Point.Y0;

                        if (VTX0_1>=RONG_X/2) VTX0_1=VTX0_1-RONG_X/2;

                        if (VTY0_1>=DAI_Y/2) VTY0_1=VTY0_1- DAI_Y/2;
                        if (VTX0_1>LCD_W-RONG_X) VTX0_1=LCD_W-RONG_X;

                        if (VTY0_1>LCD_H- DAI_Y) VTY0_1=LCD_H- DAI_Y;
                        VTX0_2=VTX0_1+RONG_X;
                        VTY0_2=VTY0_1+DAI_Y;

   LCD_WRITE_BITMAP_Y(VTX0_1,VTY0_1,RONG_X,DAI_Y, gImage_ICON_01_XANH_DUONG_47_58);
                }
        }while(PEN==0);
   }
}
/*********************************************************
CHUONG TRINH CHINH
*********************************************************/
int main(void)
{
                SystemInit();       PORT_CONF();     LED_INIT_2();
                NVIC_CONFIGURATION();
                LCD_INIT();         TOUCH_INIT();
                LCD_CLEAR(WHITE);
                TOUCH_ADS7483_ADJUST();

                LCD_CLEAR(YELLOW);
                BACK_COLOR=YELLOW;              POINT_COLOR=RED;     //POINT_COLOR=YELLOW;

                RONG_X=47; DAI_Y=58;
                VTX0_2=VTX0_1+RONG_X;           VTY0_2=VTY0_1+DAI_Y;

                LCD_WRITE_BITMAP_Y(VTX0_1,VTY0_1,RONG_X,DAI_Y,
gImage_ICON_01_XANH_DUONG_47_58);
                while(1)
                {
                        TOUCH_PRESS_DICHUYEN();
                }
}
```

d. Tiến hành biên dịch và nạp.
e. Quan sát kết quả: sau khi hiển thị bitmap ta chạm vị trí nào trên màn hình thì sau đó ảnh sẽ di chuyển đến vị trí đó.
f. Giải thích chương trình:
g. 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 mẫu 413. Giống bài 412 nhưng chạm vào ảnh rồi mới kéo ảnh di chuyển.

Tạo thư mục "BAI_413_TOUCH_GLCD_BITMAP_DC_3" để lưu project.

a. Mục đích: biết lập trình chạm và kéo ảnh di chuyển.
b. Lưu đồ:
Bước 1: Khởi tạo: dao động, port, ngắt, GLCD, touch. Led.
Bước 2: Tiến hành hiển thị bitmap.
Bước 3: Kiểm tra nếu chạm touch vào ảnh hiện tại thì mới cho phép kéo ảnh, trước khi kéo thì xoá hình ảnh trước đó, tính toán vị trí mới và hiển thị lại bitmap.
c. Chương trình:

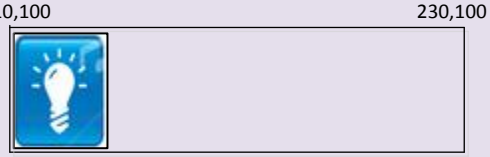
```
#include "stm32f10x.h"
#include "hardware_conf.h"
#include "TV_PORT_CONF.h"
#include "TV_GLCD_ILI932x.h"
#include "TV_TOUCH.h"
#include "TV_DELAY.h"
#include "TV_LED.h"
#include "TV_NVIC.h"

extern u8 gImage_ICON_01_DEN_44_56[];
extern u8 gImage_ICON_01_XANH_DUONG_47_58[];

uint8_t  TT_LED=0,RONG_X;
uint16_t TT_TOUCH,DAI_Y;

uint8_t VTX0_1=0,VTX0_2,VTX0_SS=0;
uint16_t VTY0_1=0,VTY0_2,VTY0_SS=0;
uint8_t VTX1_1=0,VTX1_2;
uint16_t VTY1_1=0,VTY1_2;
/*********************************************************
HAM XOA HINH BITMAP TRUOC KHI DI CHUYEN
*********************************************************/
void XOA_BITMAP()
{
        uint8_t XC;
        uint16_t YC;
        for(YC=0;YC<VTY0_1+DAI_Y;YC++)
        {
                for(XC=VTX0_1;XC<VTX0_1+RONG_X;XC++)
                {
                        LCD_DRAW_POINT(XC,YC);
                }
```

```
        }
}
/*********************************************************
HAM KIEM TRA CO CHAM TOUCH
*********************************************************/
uint8_t TOUCH_PRESS_DICHUYEN()
{
        if(Pen_Point.Key_Sta==Key_Down)
        {
                do{
                        TOUCH_ADS7483_CONVERT_POSITION();
                        Pen_Point.Key_Sta=Key_Up;
                        GPIO_SetBits(GPIOD, GPIO_Pin_9);
                        POINT_COLOR=YELLOW;
                        if ((Pen_Point.X0>VTX0_1)&&(Pen_Point.X0<VTX0_2)&&
                                        (Pen_Point.Y0>VTY0_1)&&(Pen_Point.Y0<VTY0_2))
                        {
                                if (( Pen_Point.X0!=VTX0_SS) &&(Pen_Point.Y0!=VTY0_SS)
                                        &&( Pen_Point.X0<220) &&(Pen_Point.Y0<300))
                                {
                                        XOA_BITMAP();
                                        VTX0_SS=Pen_Point.X0;
                                        VTY0_SS=Pen_Point.Y0;

                                        VTX0_1=Pen_Point.X0;
                                        VTY0_1=Pen_Point.Y0;

                                        if (VTX0_1>RONG_X/2) VTX0_1=VTX0_1-RONG_X/2;
                                        else VTX0_1=0;
                                        if (VTY0_1>DAI_Y/2) VTY0_1=VTY0_1- DAI_Y/2;
                                        else VTY0_1=0;
                                        VTX0_2=VTX0_1+RONG_X;
                                        VTY0_2=VTY0_1+DAI_Y;

                                        POINT_COLOR=RED;
                                        LCD_SHOW_NUM(10,280,VTX0_1,5,16);
                                        LCD_SHOW_NUM(10,300,VTY0_1,5,16);
                                        POINT_COLOR=YELLOW;

   LCD_WRITE_BITMAP_Y(VTX0_1,VTY0_1,RONG_X,DAI_Y, gImage_ICON_01_XANH_DUONG_47_58);

                                }
                        }
                }while(PEN==0);
                return(1);
        }
        else
        {
                GPIO_ResetBits(GPIOD, GPIO_Pin_9);
                return(0);
```

```
        }
}
/*********************************************************
CHUONG TRINH CHINH
*********************************************************/
int main(void)
{
        SystemInit();   PORT_CONF();   LED_INIT_2();
        NVIC_CONFIGURATION_TOUCH();
        LCD_INIT();     TOUCH_INIT();
        LCD_CLEAR(WHITE);
//      TOUCH_ADS7483_ADJUST();

        LCD_CLEAR(YELLOW);
        BACK_COLOR=YELLOW;
        POINT_COLOR=RED;

        RONG_X=47; DAI_Y=58;
        VTX0_2=VTX0_1+RONG_X;
        VTY0_2=VTY0_1+DAI_Y;

        LCD_WRITE_BITMAP_Y(VTX0_1,VTY0_1,RONG_X,DAI_Y,
gImage_ICON_01_XANH_DUONG_47_58);
        while(1)
        {
                TT_TOUCH=TOUCH_PRESS_DICHUYEN();
        }
}
```

d. Tiến hành biên dịch và nạp.
e. Quan sát kết quả: sau khi hiển thị bitmap ta chạm vị trí nào trên màn hình thì sau đó ảnh sẽ di chuyển đến vị trí đó.
f. Giải thích chương trình:
g. 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 414. Hãy lập trình vẽ 1 hình chữ nhật có chiều rộng bằng chiều cao lớn hơn của bitmap icon bóng đèn màu xanh có các thông số như hình theo sau, và có thể kéo ảnh di chuyển trong hình chữ nhật đã vẽ.



Tạo thư mục "BAI_414_BITMAP_DC_HCN" để lưu project.

Bài tập 415. Thêm vào bài 414 một bitmap để điều khiển cho phép/cấm dịch chuyển. Có 1 led để báo hiệu cho phép/cấm: led sáng thì cho phép, led tắt thì cấm.

Tạo thư mục "BAI_415_BITMAP_DC_HCN_ENA" để lưu project.

Bài tập 416. Thêm vào bài 415 là bitmap màu xanh trong hình chữ nhật có thể điều khiển 1 đèn led tắt mở. Khi không cho phép dịch chuyển thì mới có chức năng điều khiển đèn led, khi cho phép dịch chuyển thì không được điều khiển led.

Tạo thư mục "BAI_416_BITMAP_DC_HCN_ENA_DKLED" để lưu project.

3. CÁC CHƯƠNG TRÌNH VẼ ẢNH 2D

Phần này thực hành các bài thực hành vẽ ảnh 2D trên màn hình GLCD.

Bài mẫu 421. Chương trình vẽ hình vuông 2D gồm 2 hình nổi và chìm.

Tạo thư mục "BAI_421_2HV2D_NOI_CHIM" để lưu project.

a. Mục đích : biết lập trình vẽ hình nổi và hình chìm 2D.
b. Lưu đồ:
c. Chương trình:

```
#include "stm32f10x.h"
#include "hardware_conf.h"
#include "TV_PORT_CONF.h"
#include "TV_GLCD_ILI932x.h"
u8 X1=10,RONG=50,i,X2,X;
u16 Y1=10,DAI=50,Y2,Y;

void TO_MAU_HINH_VUONG()
{
		for(Y=Y1;Y<Y2;Y++)
		{
			for(X=X1;X<X2;X++)
			{
				LCD_DRAW_POINT(X,Y);
			}
		}
	}
void VE_HINH_VUONG_NOI(u8 X1, u8 X2,u16 Y1,u16 Y2)
{
		POINT_COLOR=WHITE;
		LCD_DRAW_LINE(X1,Y1,X2,Y1);
		LCD_DRAW_LINE(X1,Y1+1,X2,Y1+1);
		LCD_DRAW_LINE(X1,Y1,X1,Y2);
		LCD_DRAW_LINE(X1+1,Y1,X1+1,Y2);

		POINT_COLOR=BLACK;
		LCD_DRAW_LINE(X1,Y2,X2,Y2);
		LCD_DRAW_LINE(X1,Y2+1,X2,Y2+1);
		LCD_DRAW_LINE(X2,Y1,X2,Y2);
		LCD_DRAW_LINE(X2+1,Y1,X2+1,Y2);
```

```
}

void VE_HINH_VUONG_CHIM(u8 X1, u8 X2,u16 Y1,u16 Y2)
{
		POINT_COLOR=BLACK;
		LCD_DRAW_LINE(X1,Y1,X2,Y1);
		LCD_DRAW_LINE(X1,Y1+1,X2+1,Y1+1);
		LCD_DRAW_LINE(X1,Y1,X1,Y2);
		LCD_DRAW_LINE(X1+1,Y1,X1+1,Y2);

		POINT_COLOR=WHITE;
		LCD_DRAW_LINE(X1,Y2,X2,Y2);
		LCD_DRAW_LINE(X1,Y2+1,X2,Y2+1);
		LCD_DRAW_LINE(X2,Y1,X2,Y2);
		LCD_DRAW_LINE(X2+1,Y1,X2+1,Y2);
}
int main(void)
{
		SystemInit();		PORT_CONF();		LCD_INIT();

		LCD_CLEAR(LGRAY);
		BACK_COLOR=LGRAY;
		POINT_COLOR=LGRAY;

		Y2=Y1+DAI;
		X2=X1+RONG;
		POINT_COLOR=GREEN;
		TO_MAU_HINH_VUONG();
		VE_HINH_VUONG_NOI(X1, X2,Y1,Y2);

		X1=100;
		Y2=Y1+DAI;
		X2=X1+RONG;
		P POINT_COLOR=GREEN;
		TO_MAU_HINH_VUONG();
		VE_HINH_VUONG_CHIM(X1, X2,Y1,Y2);
		while(1)
		{}
}
```

d. Tiến hành biên dịch và nạp.
e. Quan sát kết quả: sau khi hiển thị bạn sẽ nhìn thấy 1 hình vuông 2D nổi rõ và 1 hình vuông chìm.
f. Giải thích chương trình:
g. Cho phép thay đổi: bạn có thể thay đổi vị trí nằm trong giới hạn, thay đổi kích thước hình, thay đổi màu nền.

Bài tập 422. Từ bài 421 bạn hãy lập trình thêm chức năng sau: hãy vẽ thêm hình tam giác trong hình nổi và hình vuông nhỏ trong hình chìm. Giống icon thứ 3 ở hàng 1 (gọi là icon stop) và icon thứ 3 hàng 2 (icon start).

Tạo thư mục "BAI_422_2HV2D_NOI_CHIM_START_STOP" để lưu project.

Bài tập 423. Từ bài 422 bạn hãy lập trình thêm chức năng sau:

Khi bắt đầu thì icon start sẽ nổi, icon stop sẽ chìm đèn led tắt.

Khi nhấn icon start thì led sáng và icon start sẽ thành chìm và icon stop sẽ nổi lên.

Khi nhấn icon stop thì led tắt và icon stop sẽ chìm, icon start sẽ nổi lên.

Tạo thư mục "BAI_423_2HV2D_NOI_CHIM_DKLED" để lưu project.

Bài tập 424. Tạo nút 2 icon UP và DW để thay đổi giá trị trên màn hình GLCD, giá trị nằm trong giới hạn từ 00 đến 99.

Tạo thư mục "BAI_424_UP_DW_99" để lưu project.

Bài tập 425. Hiển thị tất cả 16 màu trong thự viện GLCD trên GLCD theo chiều ngang. Tính toán khoảng cách chiều ngang và chia đều cho các màu. Khi chạm vào màn hình ngoài khu vực hiển thị màu thì điểm chạm đó sẽ hiển thị màu đã chọn, mặc nhiên chọn màu xanh.

Có 1 nút xoá màn hình đã vẽ.

Tạo thư mục "BAI_425_TOUCH_GLCD_VE" để lưu project.

Gợi ý: Kiểm tra nếu chạm touch chạm khu vực vẽ thì cho hiển thị điểm chạm với màu đã chọn, nếu chạm vào khu vực màu thì so sánh toạ độ và cập nhật màu mới

Bài tập 425. Tạo máy tính thực hiện các phép toán cơ bản: cộng, trừ, nhân, chia giống như trên điện thoại hoặc máy tính.

Tạo thư mục "BAI_426_CACULATOR" để lưu project.

Các bước tiến hành:

Tạo bảng chứa các con số, các ký hiệu phép toán bằng cách lập trình vẽ hoặc dùng 1 bitmap giống như vậy.

Tính toán các giới hạn để điều khiển tuoch.

Tạo 1 vùng trắng trên màn hình phía trên bàn phím để hiển thị các con số ta nhấn.

Sau khi thực hiện được các yêu cầu trên thì ta mới tiến hành tính toán cho các phép toán.

Bài tập 426. Hãy lập trình có chức năng vẽ hình vuông trên màn hình GLCD, có chọn màu. Yêu cầu: để vẽ hình vuông thì ta chạm vào màn hình điểm thứ nhất sau đó kéo cho đến điểm thứ 2.

Có chế độ tô màu cho hình vuông hoặc không.

Tạo thư mục "BAI_426_VE_HINH_VUONG" để lưu project.

Bài tập 427. Giống bài 426 nhưng vẽ hình tròn.

Tạo thư mục "BAI_427_VE_HINH_TRON" để lưu project.

Bài tập 428. Vẽ đường thẳng trên màn hình GLCD, có chọn màu.

Tạo thư mục "BAI_428_VE_DUONG_THANG" để lưu project.

Bài tập 429. Hãy lập trình hiển thị 1 bitmap trên màn hình, khi ta dùng tay kéo hình đang hiển thị sang trái thì hình thứ 2 xuất hiện, kéo tiếp thì hình thứ 3 xuất hiện, kéo tiếp thì hình thứ nhất xuất hiện trở lại.

Tạo thư mục "BAI_429_KEO_HINH_TRAI" để lưu project.

Bài tập 430. Hãy bổ xung theo chiều kéo sang phải cho bài 429.

Tạo thư mục "BAI_430_KEO_HINH_2CHIEU" để lưu project.

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

Phần mềm có hỗ trợ file thư viện điều khiển màn cảm ứng touch screen trong file <touch.h>:

1. ĐỊNH NGHĨA CÁC THÔNG SỐ VÀ CÁC HÀM

```
#ifndef __TOUCH_H
#define __TOUCH_H

#define Key_Down 0x01
#define Key_Up   0x00

typedef struct
{
        u16 X0;         u16 Y0;
        u16 X;          u16 Y;
        u8  Key_Sta;
        float xfac;     float yfac;
        short xoff;     short yoff;
}Pen_Holder;
extern Pen_Holder Pen_Point;
```

- Định nghĩa các tín hiệu giao tiếp ARM với màn cảm ứng theo chuẩn SPI

```
#define PEN     PCin(4)             //PC4 INT
#define DOUT    PAin(6)             //PA6  MISO
#define TDIN    PAout(7)            //PA7  MOSI
#define TCLK    PAout(5)            //PA5  SCLK
#define TCS     PCout(6)            //PC6  CS

/* ADS7843/7846/UH7843/7846/XPT2046/TSC2046 */
#define  CMD_RDY    0X90
#define  CMD_RDX    0XD0

void Touch_Init(void);
void Touch_Adjust(void);
void Convert_Pos(void);
void Pen_Int_Set(uint8_t en);
void Touch_Configuration(void);
void ADS_Write_Byte(uint8_t num);
```

```
uint16_t ADS_Read_AD(uint8_t CMD);
uint16_t ADS_Read_XY(uint8_t xy);
uint8_t Read_TP_Once(void);
uint8_t Read_ADS2(uint16_t *x,uint16_t *y);
uint8_t Read_ADS(uint16_t *x,uint16_t *y);
void Draw_Big_Point(uint8_t x,uint16_t y);
void Drow_Touch_Point(uint8_t x,uint16_t y);
#endif
```

2. HÀM GHI DỮ LIỆU RA IC ADS7843

- Tên hàm: ADS_Write_Byte(uint8_t num)
- Thông số: num là số cần ghi.
- Chức năng: dịch dữ liệu 8 bit của biến num ra chân TDIN. Mỗi bit được dịch theo xung TCLK.
- Hàm chi tiết như sau :

```
void ADS_Write_Byte(uint8_t num)
{
    uint8_t count=0;
    for(count=0;count<8;count++)
    {
        if(num&0x80)    TDIN=1;
        else            TDIN=0;
        num<<=1;
        TCLK=0;         TCLK=1;
    }
}
```

3. HÀM GHI LỆNH RỒI ĐỌC DỮ LIỆU TỪ IC ADS7843

- Tên hàm: uint16_t ADS_Read_AD(uint8_t CMD)
- Thông số: CMD là mã lệnh cần ghi.
- Chức năng: chọn thiết bị giao tiếp SPI rồi ghi gọi hàm ghi dữ liệu ra IC ADS7843, delay, đọc dữ liệu nối tiếp 16 bit về, không chọn chip, trả về kết quả là dữ liệu 12 bt.
- Hàm chi tiết như sau :

```
uint16_t ADS_Read_AD(uint8_t CMD)
{
    uint8_t i,count=0;
    uint16_t Num=0;

    TCLK=0;  TCS=0;
    ADS_Write_Byte(CMD);
    for(i=100;i>0;i--);

    TCLK=1;         TCLK=0;
    for(count=0;count<16;count++)
    {
        Num<<=1;
        TCLK=0;             TCLK=1;
        if(DOUT)    Num++;
```

```
        }
        Num>>=4;
        TCS=1;
        return(Num);
}
```

4. HÀM ĐỌC GIÁ TRỊ XY IC ADS7843

- Tên hàm: uint16_t ADS_Read_XY(uint8_t xy)
- Thông số: xy là mã lệnh cần ghi.
- Chức năng: tiến hành đọc tại giá trị xy 15 lần lưu từng giá trị vào mãng buf, sắp xếp 15 giá trị theo thứ tự từ nhỏ đến lớn, cộng 5 giá trị nhỏ nhất đầu tiên, chia trung bình cho 5, trả về cho hàm gọi.
- Hàm chi tiết như sau :

```
#define READ_TIMES 15
#define LOST_VAL 5
uint16_t ADS_Read_XY(uint8_t xy)
{
        uint16_t i, j;
        uint16_t buf[READ_TIMES];
        uint16_t sum=0;
        uint16_t temp;
        for(i=0;i<READ_TIMES;i++)
        {
                buf[i]=ADS_Read_AD(xy);
        }
        for(i=0;i<READ_TIMES-1; i++)
        {
                for(j=i+1;j<READ_TIMES;j++)
                {
                        if(buf[i]>buf[j])
                        {
                                temp=buf[i];
                                buf[i]=buf[j];
                                buf[j]=temp;
                        }
                }
        }
        sum=0;
        for(i=LOST_VAL;i<READ_TIMES-LOST_VAL;i++)sum+=buf[i];
        temp=sum/(READ_TIMES-2*LOST_VAL);
        return temp;
}
```

5. HÀM ĐỌC GIÁ TRỊ X VÀ GIÁ TRỊ Y

- Tên hàm: uint8_t Read_ADS (uint16_t *x,uint16_t *y)
- Thông số: nhận 2 giá trị con trỏ x và y.

- Chức năng: tiến hành đọc tại giá trị x, gán cho biến xtemp, đọc tại giá trị y, gán cho biến ytemp, kiểm tra giới hạn nếu 1 trong 2 nhỏ hơn 100 thì trả về giá trị 0, ngược lại trả về giá trị 1, lưu 2 giá trị xtemp và ytemp vào địa chỉ tương ứng của 2 con trỏ.
- Hàm chi tiết như sau :

```
uint8_t Read_ADS(uint16_t *x,uint16_t *y)
{
	uint16_t xtemp,ytemp;
	xtemp=ADS_Read_XY(CMD_RDX);
	ytemp=ADS_Read_XY(CMD_RDY);

	if(xtemp<100||ytemp<100)return 0;
	*x=xtemp;
	*y=ytemp;
	return 1;
}
```

6. HÀM KHỞI TẠO CẤU HÌNH CÁC PORT GIAO TIẾP

- Tên hàm: Touch_Configuration()
- Thông số: không có
- Chức năng: khởi tạo các port điều khiển giao tiếp IC ADS7843.
- Hàm chi tiết như sau :

```
void Touch_Configuration()
{
	GPIO_InitTypeDef GPIO_InitStructure;
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOC,
ENABLE );

	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5 | GPIO_Pin_7;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_Init(GPIOA,&GPIO_InitStructure);

	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
	GPIO_Init(GPIOA,&GPIO_InitStructure);

	//Configure PC6 pin: TP_CS pin
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_Init(GPIOC,&GPIO_InitStructure);

	//Configure PC4 pin: TP_INT pin
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
	GPIO_Init(GPIOC,&GPIO_InitStructure);
```

```
}
```

7. HÀM KHỞI TẠO IC ADS7843 ĐIỀU KHIỂN TOUCH SCREEN

- Tên hàm: Touch_Init()
- Thông số: không có
- Chức năng: khởi tạo các port điều khiển IC ADS7843 và các ngắt. Khởi gán các hệ số để tính toạ độ điểm chạm, các biến đã khai báo ở đầu chương trình.
- Hàm chi tiết như sau :

```
void Touch_Init()
{
        Touch_Configuration();
        Read_ADS(&Pen_Point.X,&Pen_Point.Y);
        /* Connect PEN EXTI Line to Key Button GPIO Pin */
        GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource4);
        /* Configure PEN EXTI Line to generate an interrupt on falling edge */
        EXTI_InitStructure.EXTI_Line = EXTI_Line4;
        EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
        EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
        EXTI_InitStructure.EXTI_LineCmd = ENABLE;
        EXTI_Init(&EXTI_InitStructure);
        /* Generate software interrupt: simulate a falling edge applied on PEN EXTI line */
        EXTI_GenerateSWInterrupt(EXTI_Line4);

        Pen_Point.xfac=0.13297872245311738;
        Pen_Point.xoff= -15;
        Pen_Point.yfac= -0.184453;
        Pen_Point.yoff=348;
}
```

8. HÀM ĐỌC TỌA ĐỘ ĐIỂM CHẠM

- Tên hàm: uint8_t Read_ADS2(uint16_t *x,uint16_t *y)
- Thông số: có 2 thông số là địa chỉ con trỏ lưu tọa độ điểm chạm
- Chức năng: đọc 2 tọa độ có kiểm tra giới hạn, thoát nếu bằng 0. Nếu hợp lệ thì kiểm tra giới hạn của điểm nhấn x1, y1 và x2, y2 cùng với thông số lệch, nếu đúng thì tính trung bình toạ độ x và y rồi lưu vào và thoát.
- Hàm chi tiết như sau :

```
#define ERR_RANGE 50
uint8_t Read_ADS2(uint16_t *x,uint16_t *y)
{
        uint16_t x1,y1;
        uint16_t x2,y2;
        uint8_t flag;
   flag=Read_ADS(&x1,&y1);
   if(flag==0)  return(0);
```

```
   flag=Read_ADS(&x2,&y2);
   if(flag==0)  return(0);

   if(((x2<=x1&&x1<x2+ERR_RANGE)||(x1<=x2&&x2<x1+ERR_RANGE))
   &&((y2<=y1&&y1<y2+ERR_RANGE)||(y1<=y2&&y2<y1+ERR_RANGE)))
   {
      *x=(x1+x2)/2;
      *y=(y1+y2)/2;
      return 1;
   }else return 0;
}
```

9. HÀM CHUYỂN ĐỔI GIÁ TRỊ ĐỌC VỀ THÀNH TỌA ĐỘ CỦA ĐIỂM CHẠM

- Tên hàm: Convert_Pos (void)
- Thông số: không có
- Chức năng: gọi hàm đọc toạ độ điểm chạm nếu có thì tính toán đưa về giá trị toạ độ.
- Hàm chi tiết như sau :

```
void Convert_Pos(void)
{
        if(Read_ADS2(&Pen_Point.X,&Pen_Point.Y))
        {
                Pen_Point.X0=Pen_Point.xfac*Pen_Point.X+Pen_Point.xoff;
                Pen_Point.Y0=Pen_Point.yfac*Pen_Point.Y+Pen_Point.yoff;
        }
}
```

10. HÀM KHỞI TẠO NGẮT LIÊN QUAN ĐẾN TOUCH

- Tên hàm: Pen_Int_Set (uint8_t en)
- Thông số: không có
- Chức năng: nếu cho phép thì làm bit cho phép ngắt bằng 1, bit thứ 4.
- Hàm chi tiết như sau :

```
void Pen_Int_Set (uint8_t en)
{
        if(en)   EXTI->IMR|=1<<4;
        else     EXTI->IMR&=~(1<<4);
}
```

11. HÀM KIỂM TRA TOUCH CÓ CHỐNG DỘI

- Tên hàm: uint8_t TOUCH_PRESS(void)
- Thông số: không có
- Chức năng: kiểm tra touch nếu có chạm thì chống dội và trả về kết quả là 1
- Hàm chi tiết như sau :

```
/************************************************************
HAM KIEM TRA CO CHAM TOUCH CO CHONG DOI
************************************************************/
uint8_t TOUCH_PRESS(void)
{
u8 di;
		if(Pen_Point.Key_Sta==Key_Down)
		{
			for(di=0;di<3;di++)
			{
				do
				{
					TOUCH_ADS7483_CONVERT_POSITION();
					Pen_Point.Key_Sta=Key_Up;
					DELAY_MS(2);
				}
				while(PEN==0);
			}
				return(1);
		}
		else return(0);
}
```