Лабораторная работа №4
Работа с дисплейным модулем HY32D

Цель работы: ознакомиться характеристиками дисплейного модуля HY32D. Запустить библиотеку и вывести изображение на дисплей.

Оборудование и программное обеспечение: плата UDK32F107V, среда разработки Cube IDE, библиотека SSD1289.h и XPT2046.h, конвертер изображений в исходный код Image2lcd.

Теоретическая информация

Управление контроллером индикатора (SSD1289) осуществляется посредством 16-разрядной параллельной шины. ЖКИ позволяет отображать графические данные с разрешением 320x240 пикселей до 262000 цветов. В состав модуля индикатора входит резистивный Touchscreen с контроллером XPT2046. Учитывая то, что микросхема XPT2046 не поддерживает высокоскоростного обмена по шине SPI, Touch Screen управляется посредством программно организованного SPI-интерфейса, что позволяет освободить аппаратный SPI2 для работы с более быстрыми устройствами, такими, как Dataflash и microSD card.

В конструкцию отладочной платы введена возможность управления подсветкой TFT-матрицы посредством подачи уровня лог.1 на вывод BLCNT разъема модуля ЖКИ. Если нет нужды использовать ЖКИ, то сигналы управления пользователь может использовать для других нужд по своему усмотрению, получив таким образом на плате два разъема расширения вместо одного. С распиновкой разъёма модуля HY32D можно ознакомиться на рис. 1.

№	Сигнал	Функциональное описание
1	+5V	Напряжение питания модуля +5V.
2	GND	Общий провод.
3	D0	Двунаправленная 16-разрядная шина данных / команд контроллера ЖКИ SSD1289.
4	D1	
5	D2	
6	D3	
7	D4	
8	D5	
9	D6	
10	D7	
11	D8	
12	D9	
13	D10	
14	D11	
15	D12	
16	D13	
17	D14	
18	D15	
19	CS	Сигнал выбора SSD1289. Активный уровень – лог.0.
20	RS	Сигнал переключения данные/команда.
21	WR	Сигнал записи по 16-разрядной шине в контроллер ЖКИ SSD1289. Активный уровень – лог.0.
22	RD	Сигнал чтения по 16-разрядной шине из контроллера ЖКИ SSD1289. Активный уровень – лог.0.
23	RESET	Сигнал сброса контроллера SSD1289. Активный уровень – лог.0.
24	TE	Не подключен.
25	BLVDD	Напряжение питания подсветки TFT-матрицы +5V.
26	BLGND	Общий провод подсветки TFT-матрицы.
27	BLCNT	Сигнал включения подсветки TFT-матрицы. Активный уровень – лог.1.
28	TP_IRQ	Сигнал прерывания от Touch Screen. Прерывание возникает по заднему фронту сигнала на этом выводе.
29	TP_CS	Сигнал выбора контроллера XPT2046 на шине SPI. Активный сигнал – лог.0.
30	TP_SCK	Тактовый сигнал шины SPI контроллера Touch Screen XPT2046.
31	TP_SI	Вход данных контроллера XPT2046. Данные «защелкиваются» по переднему фронту тактового сигнала.
32	TP_SO	Выход данных контроллера XPT2046. Данные «защелкиваются» по заднему фронту тактового сигнала.
33	3,3V	Выход напряжения 3,3 с DC/DC-преобразователя
34	GND	Общий провод

Рис. 1. Распиновка контактов модуля

Функции библиотеки дисплея

LCD_Init(); Инициализация ЖКИ

LCD_Write_REG(адрес, данные); Загрузка данных в регистр ЖКИ

LCD_Write_Command(uint8_t Comm);

LCD_Read_REG(uint16_t Adr); Чтение данных из регистра ЖКИ

LCD_SetCursor(uint16_t x,uint16_t y); Установка курсора ЖКИ

LCD_FillScreen(uint16_t Color); Заливка экрана одним цветом

LCD_SetPoint(uint16_t x,uint16_t y,uint16_t Color); Установить точку

LCD_GetPoint(uint16_t x,uint16_t y); читает точку на ЖКИ

LCD_WriteString_8x16(uint16_t x, uint16_t y, char *text, uint16_t charColor, uint16_t bkColor); Вывод на ЖКИ текста шрифтом 8x16 пикселей

LCD_WriteChar_8x16(uint16_t x, uint16_t y, char c, uint16_t t_color, uint16_t b_color); Вывод на ЖКИ символа шрифтом 8x16 пикселей

LCD_SetArea(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2); Выбор зоны для рисования

LCD_WriteChar_5x7(uint16_t x, uint16_t y, char c, uint16_t t_color, uint16_t b_color, uint8_t rot, uint8_t zoom); Вывод на ЖКИ символа шрифтом 5x7 пикселей

LCD_WriteString_5x7(uint16_t x, uint16_t y, char *text, uint16_t charColor, uint16_t b_color, uint8_t rot, uint8_t zoom); Вывод на ЖКИ текста шрифтом 5x7 пикселей

LCD_Draw_Line(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2,uint16_t color); Вывод на ЖКИ линии

LCD_Draw_Circle(uint16_t cx,uint16_t cy,uint16_t r,uint16_t color,uint8_t fill);

Lcd_Circle(uint16_t Xc, uint16_t Yc, uint16_t r, uint16_t color);

LCD_Draw_Rectangle(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2,uint16_t color,uint8_t fill); Вывод на ЖКИ прямоугольника

LCD_Draw_Picture2(uint16_t x0, uint16_t y0, const unsigned char *str); Вывод изображения на ЖКИ

LCD_WriteDataMultiple(uint16_t * pData, int NumItems); Запись блока в буфер дисплея

LCD_ReadDataMultiple(uint16_t * pData, int NumItems); Чтение блока из буфера дисплея

LCD_Write_Data(uint16_t Data); Отправка данных ЖКИ

LCD_Read_Data(); Чтение данных из ЖКИ

На таблице 1 представлены коды и наименования базовых цветов. Каждый цвет в библиотеке объявлен как #define BLACK 0x0000, следовательно вместо кода можно вводить имя этого цвета.

Таблица 1. Соответствие цветов и кода

Соответствие цветов и кода	
BLACK 0x0000	GREEN 0x07E0
WHITE 0xFFFF	BLUE 0x001F
GRAY 0xE79C	RED 0xF800
SKY 0x5D1C	YELLOW 0xFFE0
MAGENTA 0xF81F	CYAN 0x07FF
VIOLET 0x9199	PINK 0xF97F
NAVY 0x000F	ORANGE 0xFCA0
SILVER 0xA510	GOLD 0xA508
BEGH 0xF77B	BROWN 0x8200
DARK_GREEN 0x03E0	DARK_CYAN 0x03EF
MAROON 0x7800	PURPLE 0x780F
DARK_GREY 0x7BEF	LIGHT_GREY 0xC618

Функции библиотеки сенсорного модуля

TouchScreen_Init(); Инициализация Touch Screen

TouchScreen_Read(int *X, int *Y); Получение координат точки

TouchScreen_Calibrate(); Калибровка Touch Screen

TouchScreen_IRQ(void); Сигнал прерывания

uint8_t TouchScreen_MISO(void);

int Read_X(); Чтение X-координаты

int Read_Y(); Чтение Y-координаты

Для того, чтобы опрашивать сенсорый модуль только тогда, когда выполнено нажатие, нужно создать такое условие:

```
int count = 0; //переменная для счётчика срабатывания тачскрина
  for (int i=0; i<3; i++) {
  while(TouchScreen_IRQ() && count <= 200);  //фильтрация тачскрина. Срабатыввет только при отсчёте 200 срабатываний
  TouchScreen_Read(&X, &Y); //считываем координаты нажатий
}
```

Пример программы

Для начала выполнения работы запускаем Cube IDE и создаём новый проект. Основываясь на опыте из предыдущих работ, конфигурируем проект. Следует отметить, что в библиотеке дисплея и сенсорного модуля уже произведена конфигурация портов ввода/вывода, а значит дополнительно их инициализировать не нужно и после конфигурации можно приступать к разработке. В папку inc нужно скопировать ssd1289.h и xpt2046.h, а также набор шрифтов fonts.h. В папку src нужно скопировать ssd1289.c и xpt2046.c.

```
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "ssd1289.h" //библиотека дисплея
#include "xpt2046.h" //библиотека тачскрина
/* USER CODE END Includes */

/* USER CODE BEGIN 2 */
  __enable_irq (); //включаем прерывание
```

```
LCD_Init(); //инициализация дисплея
TouchScreen_Init(); //инициализация тачскрина
TouchScreen_Calibrate(); //запуск калибровки тачскрина
LCD_FillScreen(BLACK); //делаем заливку дисплея чёрным фоном
LCD_WriteString_8x16(110,220, "STM32 Paint",WHITE,BLACK); //пишем текст
  /* USER CODE END 2 */
/* USER CODE BEGIN WHILE */
  while (1)
  {
int X,Y; //координаты точек
int count = 0; //переменная для счётчика срабатывания тачскрина
   for (int i=0; i<3; i++) {
   while(TouchScreen_IRQ() && count <= 200);     //фильтрация тачскрина. Срабатыввет только при отсчёте 200 срабатываний
   TouchScreen_Read(&X, &Y); //считываем координаты нажатий
  LCD_SetPoint(X,Y,WHITE); //в месте касания рисуем белую точку
}
    /* USER CODE END WHILE */
```

В итоге получим такой результат (рис. 2).

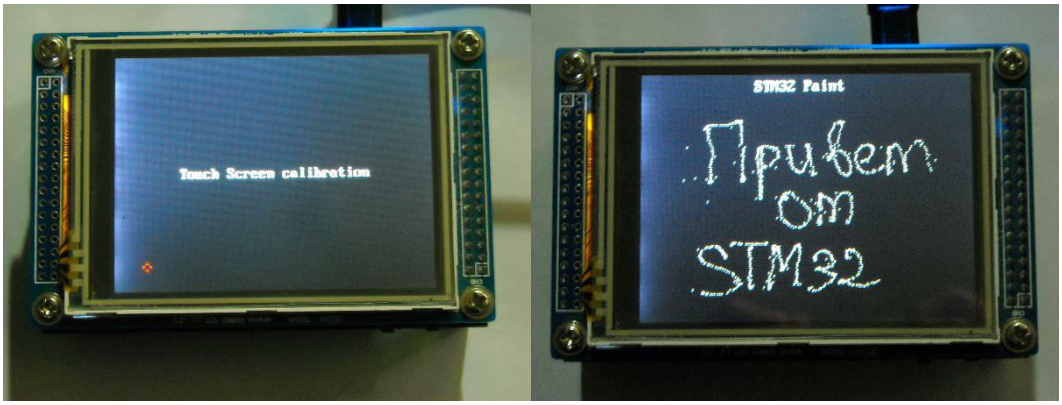

Рис. 2. Результат работы программы

Вывод изображения на дисплей

Теперь попробуем вывести изображение на дисплей. Библиотека дисплея умеет читать только bitmap-изображения в формате исходного кода. Для конвертирования изображений есть много различный программ, но в нашем случае это будет программа «Image2Lcd», так как она наиболее удобная и функциональная. Для запуска на Linux-системах потребуется Wine.

Запустив программу (рис. 3) и выбираем файл изображения. Стоит отметить, что изображение следует заранее развернуть на 90˚ в любом графическом редакторе. Размер изображения следует задать 160x128, так как если задать 320x240, оно будет занимать слишком много памяти в контроллере и его ресурсов может не хватить.

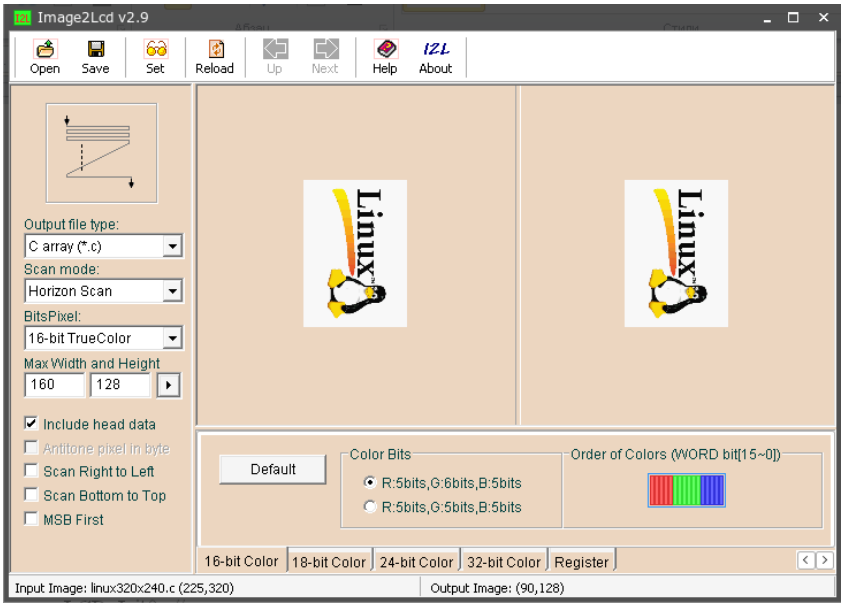

Рис. 3. Интерфейс программы

Для корректной работы требуется установить такие параметры:

BitsPixel: 16-bit TrueColor 160x128;

Scan Mode: Horizon Scan;

Color Bits: R:5bits, G:6bits, B:5bits.

В случае необходимости можно скорректировать уровень яркости и контрастности во вкладке Adjust. По окончанию конфигурации файл нужно сохранить в формате bitmap.**h** и скопировать в папку inc проекта, после чего в

самую верхнюю строку добавить разрешение изображения, как на примере показано ниже:

const unsigned char gImage_linux_image[40968] = { **128,0, 160,0,**
0X00,0X0C,0X80,0X00,0XA0,0X00,0X00,0X1B,
0XFF,0X0F,0XFF,0X0F,0XFF,0X0F,0XFF,0X0F,0XFF,0X0F,0XFF,0X0F,0XFF,0X0F,0XFF,0X0F,
0XFF,0X0F,0XFF,0X0F,0XFF,0X0F,0XFF,0X0F,0XFF,0X0F,0XFF,0X0F,0XFF,0X0F,0XFF,0X0F,0XFF,0X0F……..};

После того, как все необходимые операции с изображением выполнены, можно приступать к написанию кода.

```
/* USER CODE BEGIN Includes */
#include "ssd1289.h" //библиотека дисплея
#include "bitmap.h" //массив с изображением
/* USER CODE END Includes */

/* USER CODE BEGIN 2 */
LCD_Init(); //инициализация дисплея
LCD_FillScreen(0X0FFF);
LCD_Draw_Picture2(64,80, gImage_linux_image); //отображение массива с излбражением на дисплее
/* USER CODE END 2 */
```

В результате написания этой программы, можно наблюдать изображение, в данном случае это логотип Linux (рис. 4).

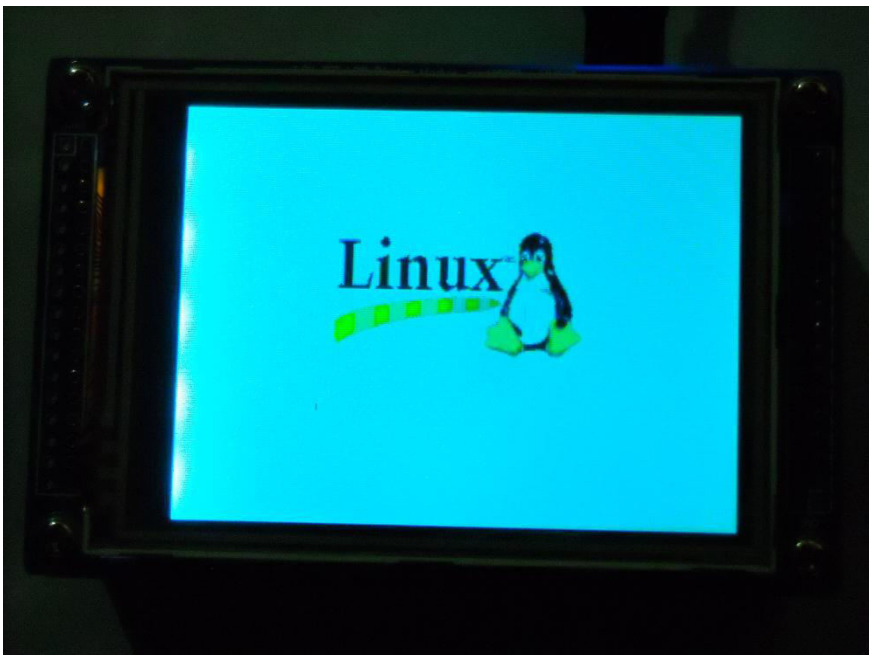

Рис. 4. Вывод изображения на дисплей

Ход работы

1. На основе кода примера приложения создать свой проект в среде разработки и проверить его работоспособность.
2. Ознакомиться с работой используемых функций, просмотрев исходный код, а также комментарии в исходных файлах библиотеки и в режиме отладки.
3. Запрограммировать плату и продемонстрировать работу программы.