Министерство образования и науки
Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра конструирования узлов и деталей радиоэлектронной аппаратуры
(КУДР)

А.А. Бомбизов, Е.И. Тренкаль

CoIDE. РАБОТА С ЭКРАНОМ

Методические указания к выполнению
лабораторной и самостоятельной работы
по дисциплине «Микропроцессорные устройства»

Томск 2017

1 Введение

Визуализация информации и организация взаимодействия с пользователем является важным этапом процесса разработки радиоэлектронных устройств. Современная компонентная база позволяет использовать в новых устройствах жидкокристаллические экраны с сенсорным управлением.

Целью настоящей работы является 1) освоение базовых приёмов вывода информации на жидкокристаллический экран; 2) организация взаимодействия с пользователем через сенсорную панель.

2 Краткая теория

Подключение жидкокристаллического экрана к микроконтроллеру осуществляется с использованием определенного аппаратного интерфейса и передачей данных по протоколу с заданными временными и амплитудными параметрами. В самых простых случаях при использовании микроконтроллеров младших моделей организацию транспортировки изображения приходится осуществлять программным способом через обычные линии портов ввода-вывода. При этом программа микроконтроллера помимо формирования изображения должна отвечать за служебные синхроимпульсы, тактирование и формирование пакетов данных в параллельном или последовательном аппаратном интерфейсе. В таком случае процесс транспорта изображения к дисплею займет большую часть вычислительных ресурсов микроконтроллера.

В рамках настоящей работы используется отладочная плата STM32F429I-DISC1, на которой установлен дисплей ILI9341 со встроенной сенсорной панелью под управлением контроллера STMPE811. Структурная схема подключения изображена на рисунке 1.

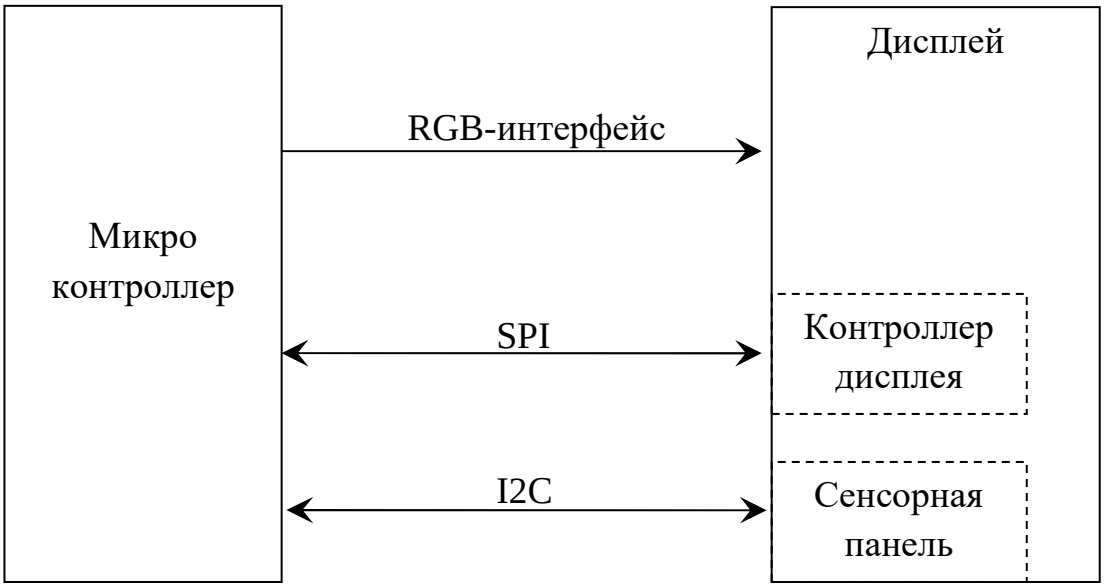


Рисунок 1 – Структурная схема

Передача изображения на дисплей производится по классическому варианту RGB-интерфейса с посылкой кадровых и строковых синхроимпульсов (VSYNC, HSYNC), передачей информации о цвете по параллельной шине с сопутствующими тактовыми импульсами. Интерфейс SPI служит для конфигурации дисплея. К примеру, можно выполнить переключение цветовой палитры с 18-битного (RGB666) режима на 16-битный (RGB565). Взаимодействие с контроллером сенсорной панели осуществляется по интерфейсу I2C.

Используемый микроконтроллер STM32F429 содержит в себе модуль LTDC для обработки и передачи изображения на дисплей. Выводимая графическая информация должна быть предварительно подготовлена в виде кадрового буфера.

Кадровый буфер (англ. framebuffer) (другие названия: буфер кадра, видеобуфер, фреймбуфер) – реальное или виртуальное электронное устройство или область памяти для кратковременного хранения одного или нескольких кадров в цифровом виде перед его отправкой на устройство видеовывода.

Обычно кадр хранится в виде последовательности цветовых значений каждого пикселя изображения.

LTDC может осуществлять работу с двумя кадровыми буферами, причем первый считается нижним, а второй накладывается сверху. При выводе каждого кадра модуль LTDC считывает оба кадровых буфера (слоя) и производит их смешивание. Степень видимости каждого слоя определяется степенью прозрачности слоя в целом или степенью прозрачности какого-либо ключевого цвета. В текущей работе смешивание будет производиться с общей степенью прозрачности слоя.

Вывод изображения осуществляется прямой записью информации о цвете пикселя в соответствующую ячейку памяти. В данной работе кадровый буфер размещается во внешней памяти и начинается с адреса 0xD0000000. Размер кадра составляет 240*320*2=153600 байт, 2 – количество байт для определения цвета пикселя. Следующий кадровый буфер размещается по адресу 0xD0000000+153600. Определение местоположения ячейки памяти должно выполняться согласно координатной плоскости дисплея, схематично изображенного на рисунке 2. Конечный адрес высчитывается в соответствии с формулой: *адресПозиции=АдресНачалаБуфера + (2*(Y*Ширина+X))*.

Как правило, на практике расчеты требуемых адресов инкапсулированы в специально подготовленных функциях, адаптированных для комфортного программирования графики. Для этих целей в настоящей

работе использована заимствованная библиотека UB (www.mikrocontroller-4u.de).

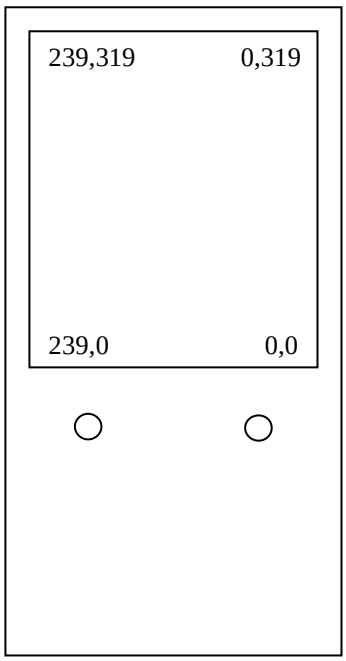


Рисунок 2 – Система координат дисплея

Описание доступных в проекте функций для работы с изображением

Далее описан минимальный набор функций для осуществления вывода изображения на дисплей.

void SystemInit(void);

Данная функция выполняет переключение тактовой частоты с базовой (16 МГц) на повышенную (168 МГц).

ErrorStatus UB_LCD_Init(void);

Выполняет следующие настройки:

1) Включает тактирование портов ввода-вывода;

2) Переключает выводы микроконтроллера на использование альтернативных функций;

3) Активирует SPI для взаимодействия с дисплеем с целью его конфигурации;

4) Конфигурирует дисплей;

5) Конфигурирует контроллер внешней памяти;

6) Запускает тактирование и конфигурирует выводы контроллера LTDC.

void UB_LCD_LayerInit_Fullscreen(void);

Выполняет настройку размера изображения, цветового режима и способа смешивания слоёв.

void UB_LCD_SetLayer_1(void);

Делает первый слой активным для функций рисования.

void UB_LCD_SetLayer_2(void);

Делает второй слой активным для функций рисования.

void UB_LCD_FillLayer(uint16_t color);

Осуществляет закраску слоя определенным цветом. В рамках данной работы кодирование цвета каждого пикселя выполняется в формате RGB565, то есть на один пиксель приходится два байта цветовой информации. Информация о распределении цветовых компонент отображена в таблице 1.

Таблица 1

Название переменной	Значение цвета	Маска
color_Value	xxxxxxxxxxxxxxxx	*0xFFFF*
red_Value	1111100000000000	*0xF800*
green_Value	0000011111100000	*0x07E0*
blue_Value	0000000000011111	*0x001F*

Программно формирование цветовой двухбайтовой ячейки можно выполнить следующим образом:

color_Value = (red_Value<<11) | (green_Value<<5) | blue_Value;

Обратное преобразование

red_Value = (0xF800 & color_Value) >> 11;

green_Value = (0x07E0 & color_Value) >> 5;

blue_Value = 0x001F & color_Value;

Для удобства в библиотеке сформирован макрос для формирования цветовой ячейки

#define RGB565(R,G,B) (R<<11) | (G<<5) | B;

Пример использования в программе макроса для заливки экрана красным цветом:

UB_LCD_FillLayer(RGB565(31,0,0));

void UB_LCD_SetTransparency(uint8_t wert);

Установка степени прозрачности текущего слоя. Значение параметра *wert* может быть установлено в пределах от 0 до 255. Значение 0 соответствует абсолютно прозрачному (невидимому) состоянию выбранного слоя, 255 – полностью непрозрачное состояние.

void UB_LCD_SetCursor2Draw(uint16_t xpos, uint16_t ypos);

Выполняет расчет адреса ячейки памяти, соответствующей координате *xpos и ypos*.

void UB_LCD_DrawPixel(uint16_t color);

Помещает цвет из переменной *color* в ячейку памяти фрейм буфера. Позиция ячейки памяти определяется функцией *SetCursor2Draw*.

void UB_LCD_SetMode(LCD_MODE_t mode);

Данная функция переключает ориентацию изображения на экране из горизонтального режима (ландшафт) в вертикальный (портрет) и наоборот. Если выразиться проще, то она меняет оси X и Y местами. Переменная *mode* может принимать значения *LANDSCAPE* и *PORTRAIT*.

void UB_LCD_Rotate_0(void);

void UB_LCD_Rotate_180(void);

Эта пара функций отвечает за поворот изображения на экране: *Rotate_180* – поворот на 180 градусов; *Rotate_0* – возврат в исходное положение.

void UB_LCD_Copy_Layer1_to_Layer2(void);

void UB_LCD_Copy_Layer2_to_Layer1(void);

Данные функции предназначены для копирования изображения из первого слоя во второй и обратно.

void DrawLine(int x1,int y1,int x2,int y2,int color);

Выполняет рисование отрезка с координатами (*x1*,*y1*,*x2*,*y2*) и цветом *color*.

Описание доступных в проекте функций для работы с сенсорным экраном

Для работы с сенсорным экраном достаточно использовать только две функции и одну глобальную переменную.

UB_Touch_Init();

Выполняет инициализацию контроллера сенсорного экрана.

UB_Touch_Read();

Выполняет чтение состояния (есть ли в данный момент прикосновение) сенсора и координаты прикосновения. В случае поступления новых данных выполняется обновление переменной *Touch_Data*.

```
typedef struct
{
 Touch_Status_t status;
  uint16_t xp;
  uint16_t yp;
}Touch_Data_t;
```

Данная структура описывает текущее состояние сенсора:

Поле *status* может принимать два значения: *TOUCH_PRESSED*, если в текущий момент контроллер зафиксировал нажатие, и *TOUCH_RELEASED* – никаких действий с сенсором не предпринимается.

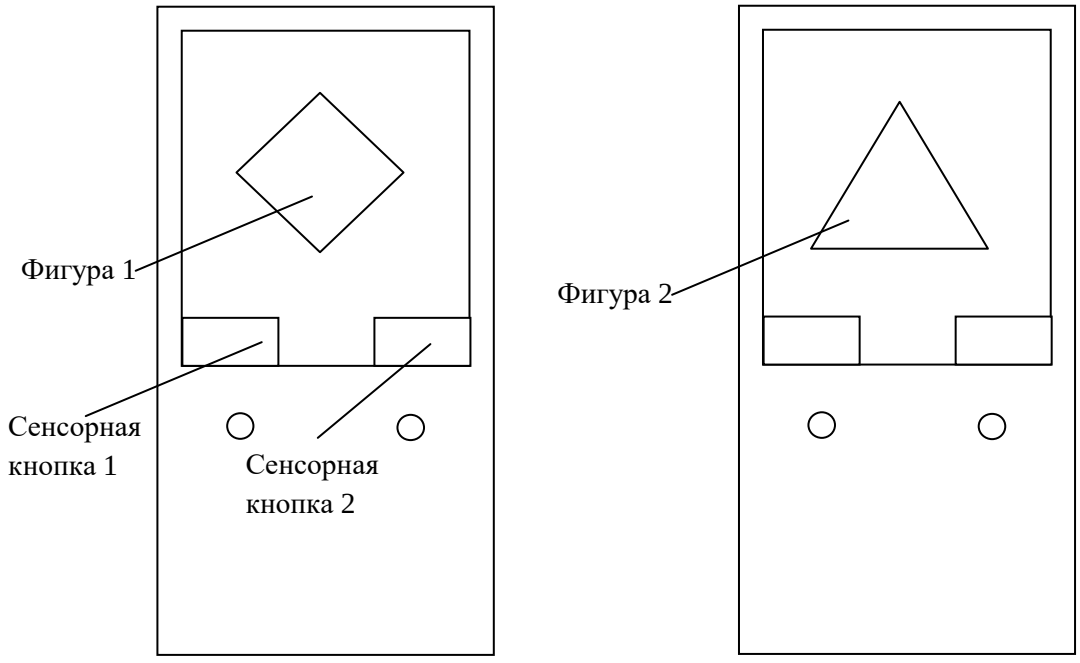
Таким образом, чтобы работать с сенсором, достаточно единожды в начале программы выполнить инициализацию контроллера сенсорного

экрана, после чего в рабочем бесконечном цикле или по прерыванию таймера выполнять опрос состояния.

```
while(1)
{
        UB_Touch_Read(); //опрос состояния
        if(Touch_Data.status==TOUCH_PRESSED) //если есть прикосновение
        {
                //Если прикосновение произошло в области с координатами
                //(120,100,80,50)
                if((120>Touch_Data.xp)&& (Touch_Data.xp>80)&&
                        (100>Touch_Data.yp)&& (Touch_Data.yp>50))
                {
                        //то выполняем обработку этого прикосновения
                }
        }
}
```

3 Порядок выполнения работы

В ходе данной работы будут освоены следующие разделы: 1) вывод изображения на разные слои; 2) обработка прикосновения к сенсорному экрану. В результате должна быть разработана программа, реализующая две сенсорные кнопки на экране, изменяющая выводимое изображение в зависимости от нажатой сенсорной кнопки (рисунок 3).



а) б)

а) изображение ромба на экране при нажатии на кнопку 1;
б) изображение треугольника на экране при нажатии на кнопку 2.

Рисунок 3 – Эскиз будущего интерфейса

Подготовка дисплея и микроконтроллера к работе.

3.1 Изучите предложенный в п. 2 теоретический материал.

3.2 Откройте прилагаемую к работе заготовку проекта.

3.3 В функции *main()* сконфигурируйте повышенную тактовую частоту.

3.4 Сконфигурируйте дисплей при помощи функций *UB_LCD_Init()* и *UB_LCD_LayerInit_Fullscreen()*.

3.5 Сконфигурируйте сенсорный экран.

3.6 Выполните переключение на нижний слой.

3.7 Сделайте слой не прозрачным и установите цвет фона по вашему усмотрению.

3.8 Выполните переключение на верхний слой.

3.9 Установите черный цвет фона и среднюю степень прозрачности.

3.10 Скомпилируйте программу.

3.11 Запрограммируйте микроконтроллер и проверьте работоспособность программы.

Создание кнопок

3.12 На нижнем слое нарисуйте линиями контуры кнопок как показано на рисунке 3.

3.13 Добавьте обработку нажатия кнопок как описано в теоретической части.

3.14 В обработчике первой кнопки выполните на верхнем слое очистку фона и рисование ромба.

3.15 В обработчике второй кнопки выполните на верхнем слое очистку фона и рисование треугольника.

3.16 Скомпилируйте программу.

3.17 Запрограммируйте микроконтроллер и проверьте работоспособность программы.

3.18 Оформите отчет, содержащий титульный лист, введение, ход выполнения работы, ответы на контрольные вопросы и выводы.

3.19 Защитите отчет у преподавателя.

4 Контрольные вопросы

4.1 Каким образом формируется кадровый буфер?

4.2 Что вызывает изменение цвета пикселя на экране?

4.3 Каким образом изменяется прозрачность слоя?

4.4 Каким образом преобразовать цвет в формат RGB565?

4.5 Как на практике выполняется работа с сенсорным экраном?

Список литературы

1. RM0090. Reference manual. STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced ARM®-based 32-bit MCUs.– URL: www.st.com/resource/en/reference_manual/DM00031020.pdf (дата обращения: 10.01.2017).

2. PM0214. Programming manual. STM32F3, STM32F4 and STM32L4 Series. Cortex®-M4 programming manual.– URL: www.st.com/resource/en/programming_manual/DM00046982.pdf (дата обращения: 10.01.2017).

3. MB1075. STM32F429I-DISCO schematics.– URL: http://www.st.com/resource/en/schematic_pack/stm32f429i-disco_sch.zip.

4. GNU ARM Assembler Quick Reference.– URL://http://www.ic.unicamp.br/~celio/mc404-2014/docs/gnu-arm-directives.pdf.