МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

И ПРОГРАММНОЙ ИНЖЕНЕРИИ (КАФЕДРА 43)

КУРСОВОЙ ПРОЕКТ

ЗАЩИЩЕН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Доцент, к.т.н. | / | / | / | А.А. Попов |
| (должность, учёная степень, звание) | (подпись) | (дата защиты) |  | (инициалы, фамилия) |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ |
| ТРАНСЛЯТОР АНГЛИЙСКОГО АЛФАВИТА В КОД МОРЗЕ |
| по курсу: ПРОГРАММИРОВАНИЕ ВСТРОЕННЫХ ПРИЛОЖЕНИЙ |

|  |  |
| --- | --- |
| РАБОТУ ВЫПОЛНИЛИ СТУДЕНТЫ ГРУППЫ: | 4036 |
|  | (номер группы) |
| Н.С. Ячменев | |
| (инициалы, фамилии) | |

|  |  |  |
| --- | --- | --- |
|  | / | / |
| (подпись студента) | | (дата отчета) |

Санкт-Петербург 2023

## Оглавление

[1. Задание на курсовую работу 3](#_Toc153746556)

[2. Техническое задание на прибор 4](#_Toc153746557)

[2.1 Диаграмма вариантов использования 5](#_Toc153746558)

[2.2 Основные требования 6](#_Toc153746559)

[3. Схемы и настройка проекта 7](#_Toc153746560)

[4.Описание реализации 10](#_Toc153746561)

[4.1 Описание состояний программы 10](#_Toc153746562)

[4.2 Описание модулей программы 11](#_Toc153746563)

[4.3 Описание алгоритмов 14](#_Toc153746564)

[5. Тестирование 19](#_Toc153746565)

[5.1. Ввод данных 19](#_Toc153746566)

[5.2 Процесс конвертации в код Морзе 22](#_Toc153746567)

[5.3 Итог тестирования 24](#_Toc153746568)

[6. Экономическая оценка 25](#_Toc153746569)

[7. Заключение 25](#_Toc153746570)

[8. Список используемой литературы 26](#_Toc153746571)

[Приложение 27](#_Toc153746572)

## 1. Задание на курсовую работу

Транслятор английского алфавита в код Морзе.

Используя Keypad 4x4 реализовать ввод букв английского алфавита, знаков препинания и цифр и вывести их на экран LCD 320x240 в строку. В момент нажатия кнопки старт начинать передачу кодом Морзе на светодиод и динамик. Ввод следующего пакета цифр возможен во время передачи. Скорость передачи выбирается из ряда 60/120/180/240 знаков в минуту. Какие кнопки использовать, и какими функциями наделить решить самостоятельно.

## 2. Техническое задание на прибор

В рамках разрабатываемой программы «Транслятор английского алфавита в код Морзе» выделяются следующие исполнители (actors):

* пользователь;
* микроконтроллер (как управляющее устройство).

Основные функции транслятора:

* в режиме бездействия, когда пользователь не взаимодействует с программой, на LCD дисплее демонстрируется интерфейс со всеми основными функциями программы;
* ввод пользователем следующих команд:
* переместить каретку выбора символа влево
* переместить каретку выбора символа вправо
* переместить каретку выбора символа вниз
* переместить каретку выбора символа вверх
* добавить выбранный символ в буфер сообщения
* удалить выбранный символ из буфера сообщения
* начать (перезапустить) трансляцию текущего сообщения в виде кода Морзе
* увеличить скорость передачи
* уменьшить скорость передачи

Для получения команд от пользователя используются следующие кнопки кнопочной панели Keypad 4x4:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 |  |
| 8 | 9 |  |  |
|  |  |  |  |

Таблица 1 – Используемые кнопки кнопочной панели.

Данные кнопки используются для следующих функций:

* кнопка 1 – удалить выбранный символ из буфера сообщения
* кнопка 2 – переместить каретку выбора символа вверх
* кнопка 3 – добавить выбранный символ в буфер сообщения
* кнопка 4 – начать трансляцию текущего сообщения
* кнопка 5 – переместить каретку выбора символа влево
* кнопка 6 – переместить каретку выбора символа вниз
* кнопка 7 – переместить каретку выбора символа вправо
* кнопка 8 – увеличить скорость передачи
* кнопка 9 – уменьшить скорость передачи

Другие кнопки кнопочной панели не используются.

## 2.1 Диаграмма вариантов использования

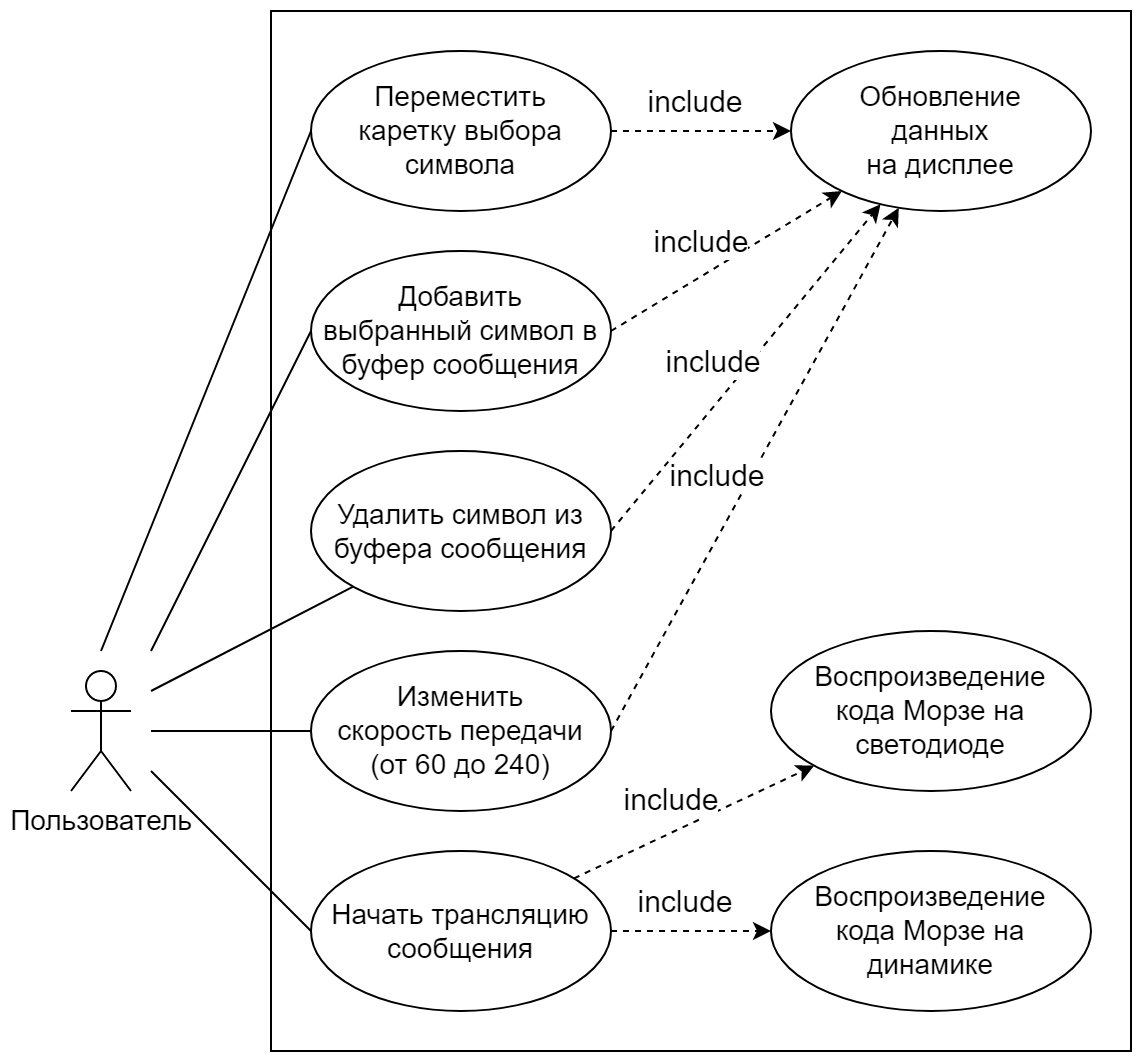


Рисунок 1 – Диаграмма вариантов использования.

## 2.2 Основные требования

Ввод: осуществляется с помощью клавиатуры Keypad 4x4.

Вывод:

* интерфейс программы выводится на экран LCD 320x240;
* код Морзе выводиться при помощи модуля тестирования аналоговых сигналов (динамик) и светодиодов платы STM32F3-Discovery.

Функции:

* выбор символа
* добавление выбранного символа в буфер сообщения
* удаления символа из буфера сообщения
* проигрывание кода Морзе введенной последовательности символов при помощи динамика
* проигрывание кода Морзе введенной последовательности символов при помощи светодиодов
* изменение скорости воспроизведения кода Морзе

Питание: от интерфейса USB или от внешнего источника 3В или 5В.

Размеры и вес: сопоставимы с размером и весом книги.

## 3. Схемы и настройка проекта

При разработке транслятора задействованы следующие модули/платы:

1) Кнопочная панель HC-35 (или подобная 4х4 матрица кнопок), схема в части клавишной панели 4х4:

|  |  |
| --- | --- |
|  |  |

Рисунок 2 – Схема части клавишной панели.

Кнопочная панель подключается к разъёму 8I/Os согласно таблице 1:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Выводы со стороны Open32F3-D разъёма 8I/Os | PC 7 | PC 6 | PC 5 | PC4 | PC 3 | PC 2 | PC1 | PC0 |
| Выводы со стороны кнопочной панели, разъёма KEY A (P1) | L4 | L3 | L2 | L1 | R4 | R3 | R2 | R1 |

Таблица 2 – Контакты подключение кнопочной панели.

Кнопочная панель используется как средство изменения сигнала на заданных линиях, что и приводит к выставлению запроса прерывания. Для снижения влияния явления дребезга и взаимных помех при нажатии кнопок, используем высокий уровень на линиях настроенных как входы, путем подтягивания их к питанию через встроенный в ПВВ pull up резистор. А выходы настроим в режим с открытым стоком и установим на них нули.

2) LCD дисплей:



Рисунок 3 – LCD дисплей.

Управление ЖК-панелью разрешением 320x240 точек осуществляет встроенный однокристальный контроллер/драйвер – ILI9325. Он состоит из 320 каналов драйверов строк и 720 каналов драйверов столбцов, что при глубине каждого канала цвета (RGB) 6 бит 10 составляет 172800 байт памяти отображения графики (GRAM) 240RGBx320 точек.

Для подключения к управляющему внешнему устройству у контроллера ILI9325 имеется 4 вида системных интерфейсов. Это параллельный интерфейс стандарта Intel 8080 (Parallel Bus) с настраиваемой шириной шины данных (8, 9, 16, 18 бит), интерфейс VSYNC, последовательный интерфейс SPI и 6/16/18-разрядный RGB интерфейс.

LCD ILI9325 подключён к микроконтроллеру STM32F303VCT6 в режиме параллельного интерфейса Intel 8080 c 16-разрядной шиной данных, схема подключения показана на рисунке 4:

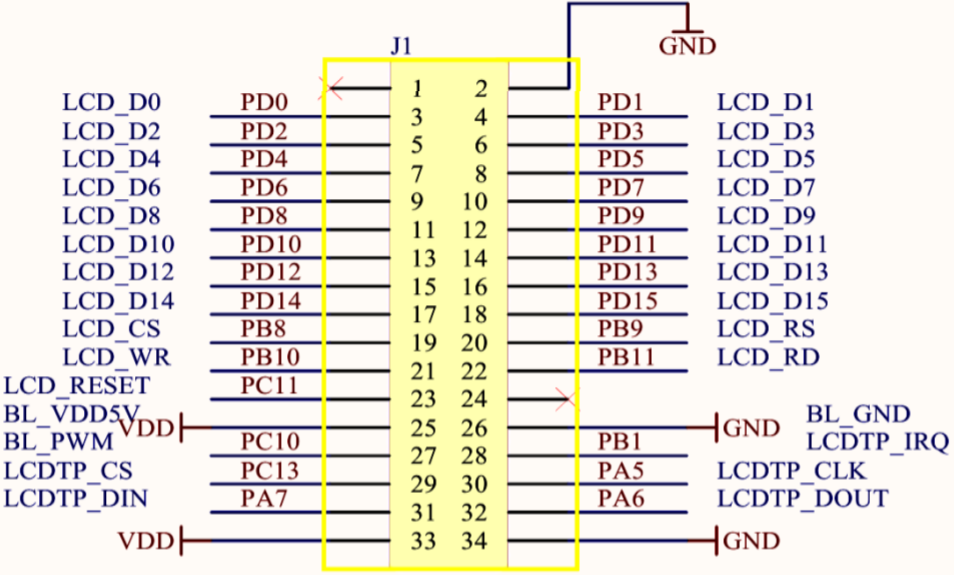


Рисунок 4 – Схема подключения LCD дисплея к плате Open32F3-D

3) Модуль тестирования аналоговых сигналов:

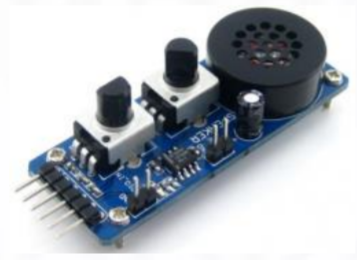


Рисунок 5 – Модуль тестирования аналоговых сигналов.

Модуль тестирования аналоговых сигналов подключается по схеме, показанной в таблице 2:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выводы со стороны Open32F3-D разъёма SPI | PA4 | PF4 | PF2 | GND | VDD |
| Выводы со стороны модуля - разъёма P1, на разъём H1 пять вольт выведены отдельным проводом | DAC | ADC2 | ADC1 | GND | VCC (3.3V) |

Таблица 3 – Контакты подключение модуля аналоговых сигналов.

## 4.Описание реализации

## 4.1 Описание состояний программы

После запуска программа начинает ожидать ввода от пользователя с помощью кнопочной панели. При появлении вода от пользователя программа реагирует на ввод в зависимости от того какая кнопка была нажата. Возможны следующие сценарии действий:

1) Переместить каретку выбора символа, при нажатии на соответствующие кнопки. При этом микроконтроллер отображает измененное положение каретки выбора символа. Далее микроконтроллер ожидает следующего ввода от пользователя.

2) Добавить выбранный символ в буфер сообщения, при нажатии на соответствующую кнопку на кнопочной панели. При этом микроконтроллер отображает измененный буфер сообщения. Далее микроконтроллер ожидает следующего ввода от пользователя.

3) Удалить выбранный символ из буфера сообщения, при нажатии на соответствующую кнопку на кнопочной панели. При этом микроконтроллер отображает измененный буфер сообщения. Далее микроконтроллер ожидает следующего ввода от пользователя.

4) Начать (перезапустить) трансляцию текущего сообщения в виде кода Морзе, при нажатии на соответствующую кнопку на кнопочной панели. При этом микроконтроллер начинает отображать процесс перевода сообщения в код Морзе и соответственно проигрывать код Морзе на динамике и светодиодах. Если в момент нажатия на кнопку микроконтроллер уже проигрывает сообщение, то воспроизведение обрывается и начинается воспроизведение нового сообщения. При этом параллельно этому процессу пользователь может продолжать работать с буфером сообщения, то есть микроконтроллер ожидает следующего ввода от пользователя.

5) Изменить скорость передачи, при нажатии на соответствующие кнопки кнопочной панели. При этом микроконтроллер отображает текущую скорость воспроизведения на экране. Далее микроконтроллер ожидает следующего ввода от пользователя.

Во время своей работы программа переходит между следующими состояниями:

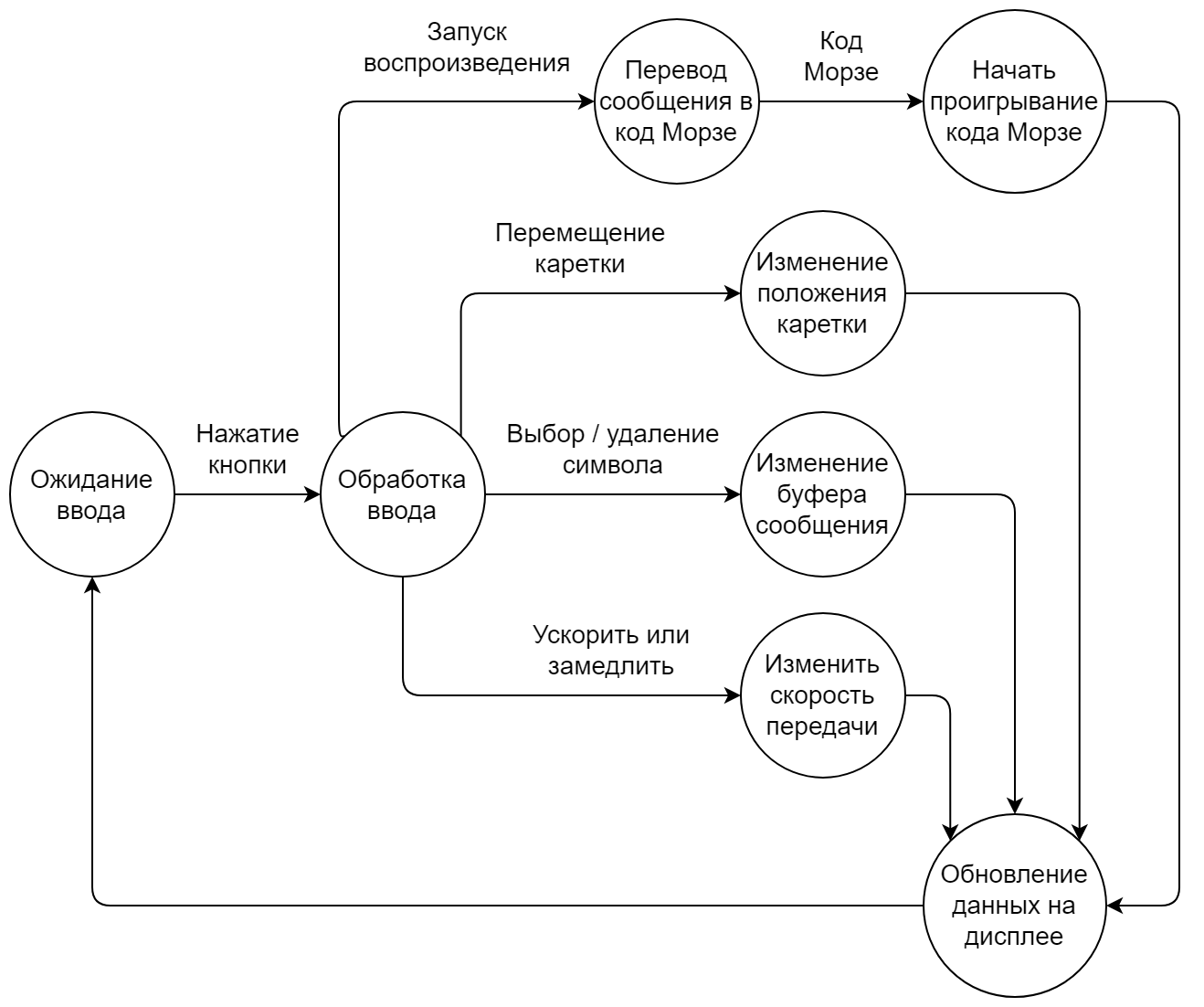


Рисунок 6 – Диаграмма состояний.

## 4.2 Описание модулей программы

Реализованная программа состоит из следующих модулей:

* main – модуль содержит главную функцию приложения запускаемую в начале работы программы. Ее функциями является инициализация начального состояния приложения и запуск главного цикла программы.
* keypad – модуль отвечает за обработку нажатия кнопок и передачу событий ввода пользователя другим модулям.
* timers – модуль отвечает за создание и работу таймеров, отвечающих за периодические события программы.
* displayctrl – модуль отвечает за обработку событий ввода пользователя и отображение текущего состояния работы программы на экране. Помимо этого модуль отвечает за перевод сообщения в код Морзе и воспроизведение его на динамике и светодиодах.
* LCD\_ILI9325 – библиотека предоставляющая функционал для рисования на LCD дисплее.
* asciiLib – обертка над библиотекой LCD\_ILI9325, предоставляющая упрощенный способ вывода на экран ASCII текста.

Модули программы связаны между собой следующим образом:

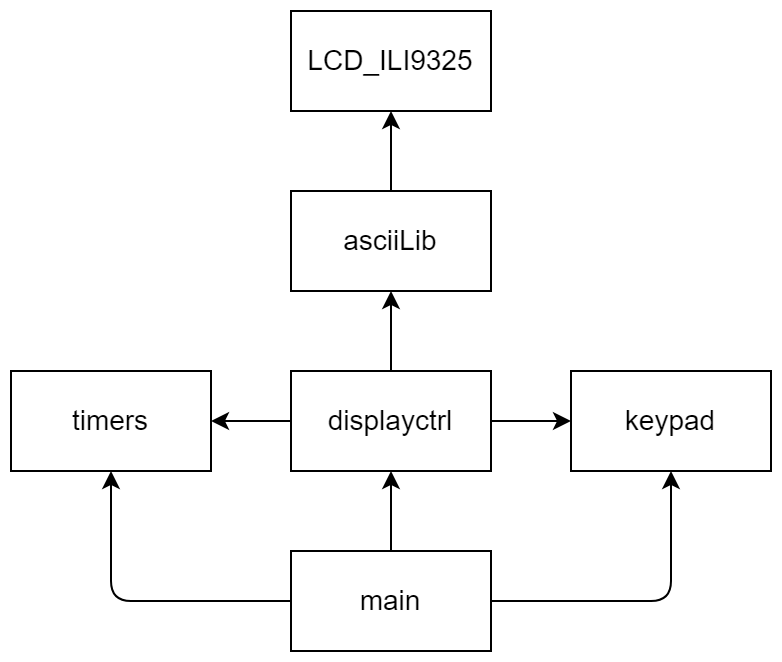


Рисунок 7 – Диаграмма взаимосвязи модулей программы.

В модулях используются следующие предоставляемые функции:

Модуль main:

* int main(void) – главная функция приложения
* void SystemClock\_Config(void) – функция для конфигурации микроконтроллера

Модуль keypad:

* void InitKpd4(void) – функция инициализации модуля
* void ProcessKeyKpd4(void) – функция отвечающая за обработку нажатий кнопок
* uint16\_t getKeycode(void) – функция возвращающая код нажатой копки. В случае если ни одна кнопка не была нажата, функция возвращает невалидное значение

Модуль timers:

* void InitTimers(void) – функция инициализации модуля
* void ProcessTimers(void) – данная функция отвечает за функционирование таймеров
* unsigned int GetTimer(char Timer) – функция для получения текущего значения таймера по его номеру
* void ResetTimer(char Timer) – функция сброса таймера по его номеру

Модуль displayctrl:

* void InitDisplay(void) – функция инициализации модуля
* void ProcessDisplay(void) – функция обработки полученных событий и обновления данных на дисплее

Модуль (библиотека) asciiLib:

* void GetASCIICode(unsigned char\* pBuffer,unsigned char ASCII) – функция для заполнения буфера pBuffer изображением соответствующим переданному коду ASCII символа
* void LCDprintstr(char\* str, int16\_t y,uint16\_t bkcolor, uint16\_t charcolor) – функция для обрисовки переданного текста на экране в указанных координатах и указанного цвета

Модуль (библиотека) LCD\_ILI9325: функции данной библиотеки напрямую не используются. Все вызовы данной библиотеки инкапсулированы в используемой библиотеке asciiLib.

## 4.3 Описание алгоритмов

Программа начинает свою работу с главной функции main. Далее приведена блок-схема данной функции:

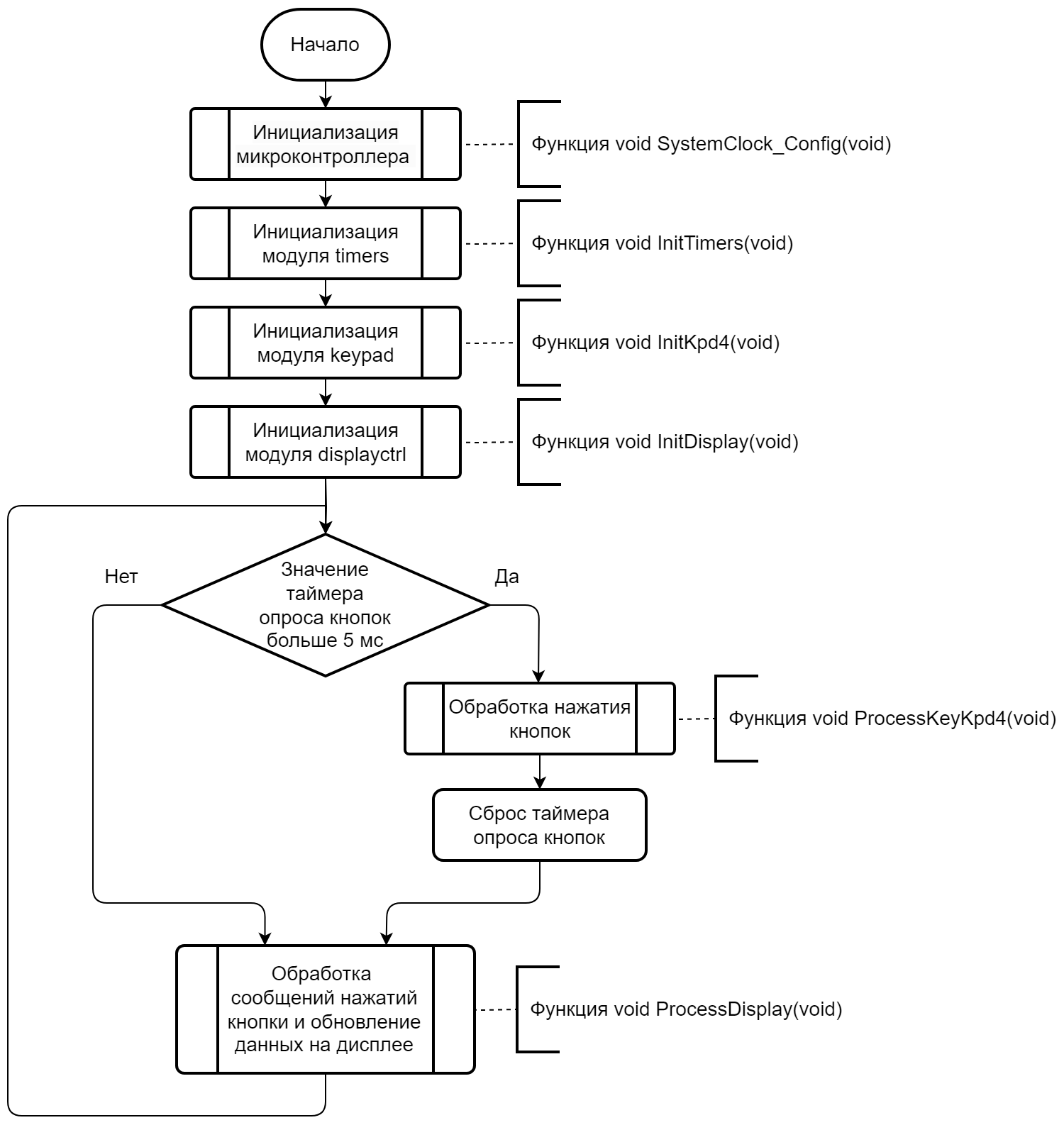


Рисунок 8 – Блок-схема главной функции main.

Основная работа программы происходит в функции модуля displayctrl void ProcessDisplay(void). Блок-схема данной функции приведена далее:

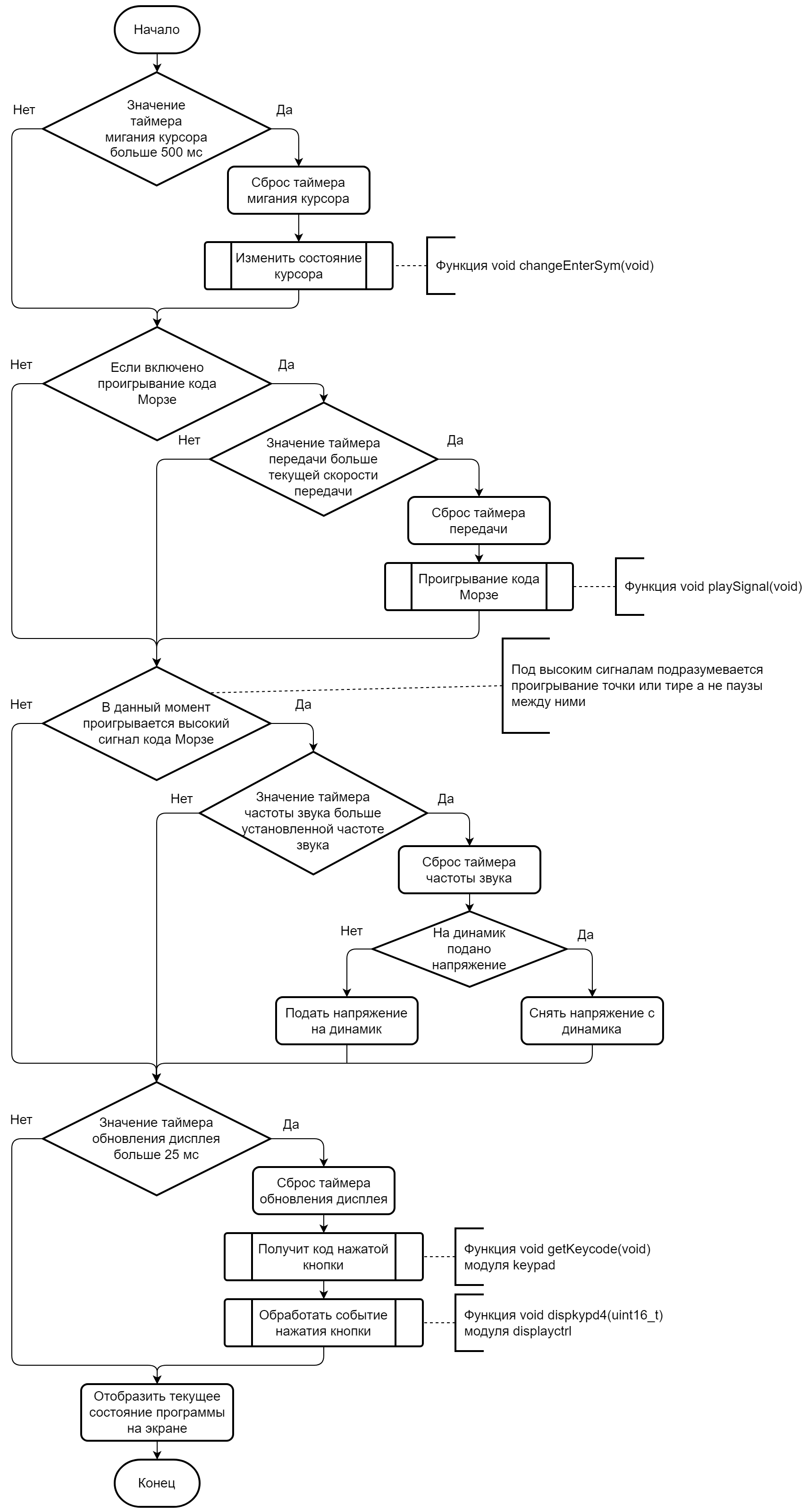


Рисунок 9 – Блок-схема функции ProcessDisplay.

В блок-схеме приведенной выше используется вызов функции проигрывания кода Морзе void playSignal(void). Далее приведена блок-схема этой функции:

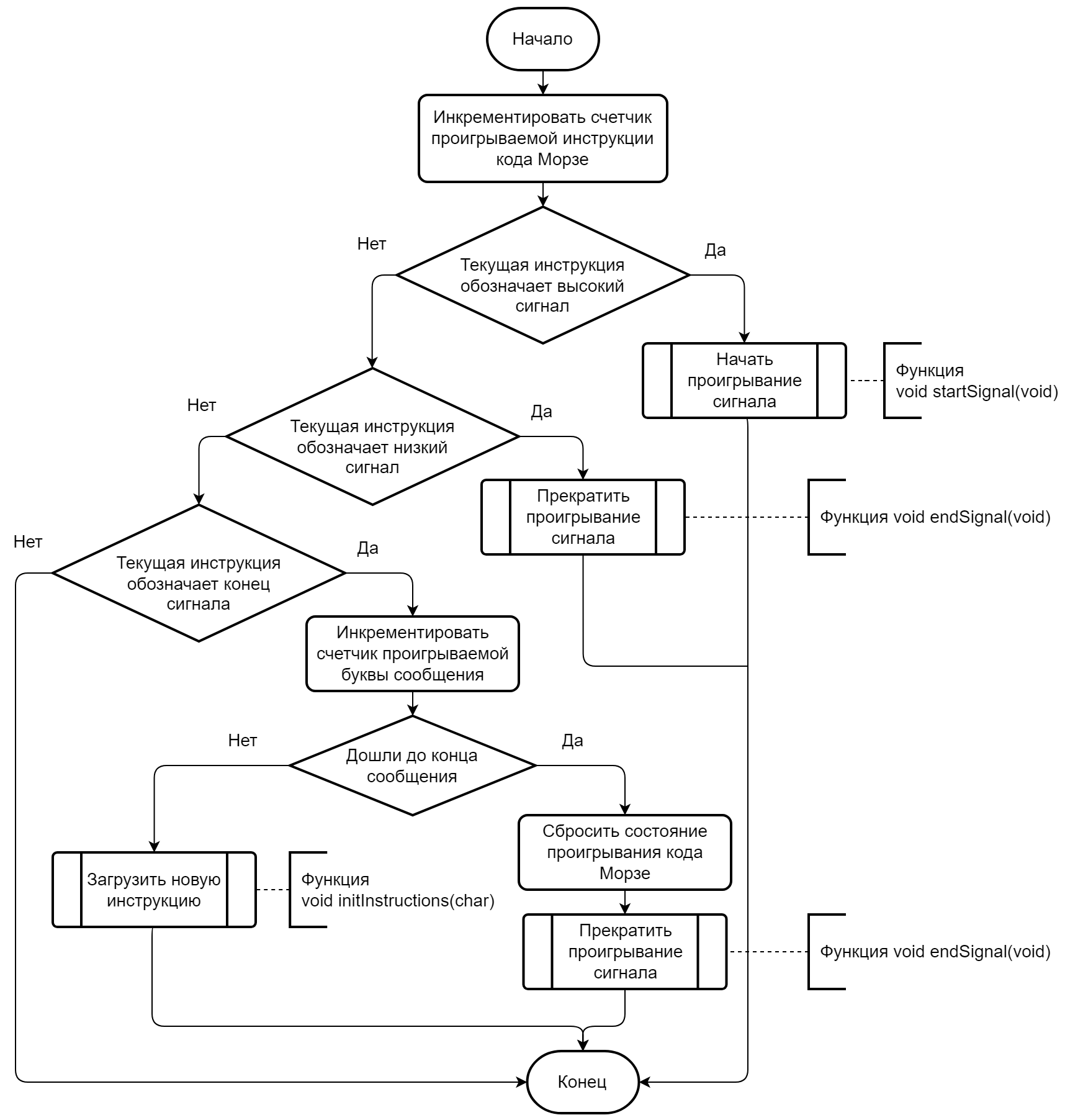


Рисунок 10 – Блок-схема функции playSignal.

Функция начала проигрывания сигнала void startSignal(void) и функция прекращения проигрывания сигнала void endSignal(void) подает и убирает напряжение со светодиодов. При этом флаг проигрывания высокого сигнала устанавливается на соответствующее значение.

Функция загрузки инструкции воспроизведения нового сигнала void initInstructions(char) заполняет буфер инструкции значениями высокого или низкого сигнала кода Морзе в зависимости от переданного символа.

Для создания инструкции используется следующая заранее подготовленная таблица:

const char morseCode[50][6] = {

{'.','-','0','0','0','0'}, //1 a

{'-','.','.','.','0','0'}, //2 b

{'-','.','-','.','0','0'}, //3 c

{'-','.','.','0','0','0'}, //4 d

{'.','0','0','0','0','0'}, //5 e

{'.','.','-','.','0','0'}, //6 f

{'-','-','.','0','0','0'}, //7 g

{'.','.','.','.','0','0'}, //8 h

{'.','.','0','0','0','0'}, //9 i

{'.','-','-','-','0','0'}, //10 j

{'-','.','-','0','0','0'}, //11 k

{'.','-','.','.','0','0'}, //12 l

{'-','-','0','0','0','0'}, //13 m

{'-','.','0','0','0','0'}, //14 n

{'-','-','-','0','0','0'}, //15 o

{'.','-','-','.','0','0'}, //16 p

{'-','-','.','-','0','0'}, //17 q

{'.','-','.','0','0','0'}, //18 r

{'.','.','.','0','0','0'}, //19 s

{'-','0','0','0','0','0'}, //20 t

{'.','.','-','0','0','0'}, //21 u

{'.','.','.','-','0','0'}, //22 v

{'.','-','-','0','0','0'}, //23 w

{'-','.','.','-','0','0'}, //24 x

{'-','.','-','-','0','0'}, //25 y

{'-','-','.','.','0','0'}, //26 z

{'0','0','0','0','0','0'}, //27 space

{'.','-','.','-','.','-'}, //28 .

{'-','-','.','.','-','-'}, //29 ,

{'-','-','-','.','.','.'}, //30 :

{'.','.','-','-','.','.'}, //31 ?

{'.','-','-','-','-','.'}, //32 '

{'-','.','.','.','.','-'}, //33 -

{'-','.','.','-','.','0'}, //34 /

{'-','.','-','-','.','0'}, //35 (

{'-','.','-','-','.','-'}, //36 )

{'.','-','.','.','-','.'}, //37 "

{'-','.','.','.','-','0'}, //38 =

{'.','-','.','-','.','0'}, //39 +

{'.','-','-','.','-','.'}, //40 @

{'.','-','-','-','-','0'}, //41 1

{'.','.','-','-','-','0'}, //42 2

{'.','.','.','-','-','0'}, //43 3

{'.','.','.','.','-','0'}, //44 4

{'.','.','.','.','.','0'}, //45 5

{'-','.','.','.','.','0'}, //46 6

{'-','-','.','.','.','0'}, //47 7

{'-','-','-','.','.','0'}, //48 8

{'-','-','-','-','.','0'}, //49 9

{'-','-','-','-','-','0'}, //50 0

};

При составлении инструкции используются следующие правила:

* Длительность точки – 1 у.е.
* Длительность тире – 3 у.е.
* Пауза между сигналами, образующими одну букву – 1 у.е.
* Пауза между двумя буквами – 3 у.е.
* Пауза между двумя словами – 7 у.е.
* Значение у.е. зависит от скорости передачи.

Так инструкция для слова «au» с кодами Морзе «a»: (точка, тире) и «u»: (точка, точка, тире) будет следующей:

* одна единица (1) – обозначающая точку буквы «a»
* один ноль (0) – обозначающий паузу между сигналами, образующими одну букву
* три единицы (111) – обозначающие тире буквы «a»
* три ноля (000) – обозначающие паузу между двумя буквами
* одна единица (1) – обозначающая точку буквы «u»
* один ноль (0) – обозначающий паузу между сигналами, образующими одну букву
* одна единица (1) – обозначающая точку буквы «u»
* один ноль (0) – обозначающий паузу между сигналами, образующими одну букву
* три единицы (111) – обозначающие тире буквы «u»
* знак конца инструкции (\0) – обозначающий конец инструкции

Таким образом, для слова «au» получается следующая инструкция: «101110001010111\0».

## 5. Тестирование

Тесты для контроля соответствия прибора техническому заданию:

## 5.1. Ввод данных

Проверим возможность выбора символа. Для этого нажимаем на кнопки, отвечающие за перемещение каретки выбора символа. Также проверим возможность добавления символа в буфер сообщения при нажатии соответствующей кнопки. Каретка перемещается, символы добавляются в буфер сообщения:

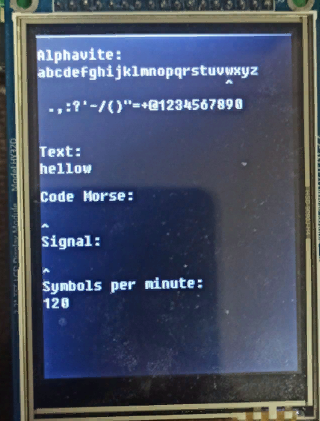


Рисунок 11 – Выбор символа.

Проверим возможность удаления символа при нажатии на соответствующую кнопку. Кнопка удаляется:

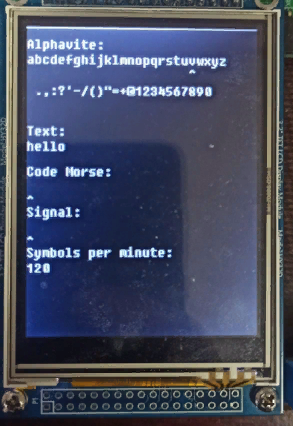


Рисунок 12 – Удаление символа.

Проверим правильность работы программы при попытке выйти кареткой выбора символа за пределы возможных символов. Для этого перемещаем каретку до упора. Каретка останавливается на последнем символе:

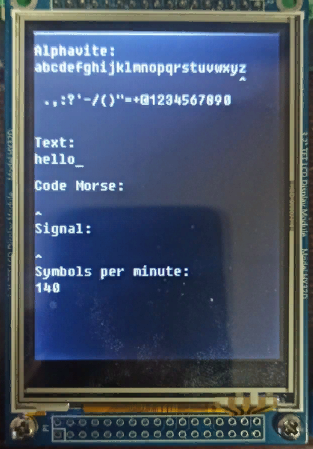


Рисунок 13 – Обработка выхода за границы алфавита.

Проверим случай переполнения буфера сообщения. Для этого будем заполнять буфер до предела. При достижении придела буфер сообщения перестал расти, а событие добавления новой буквы стало игнорироваться:

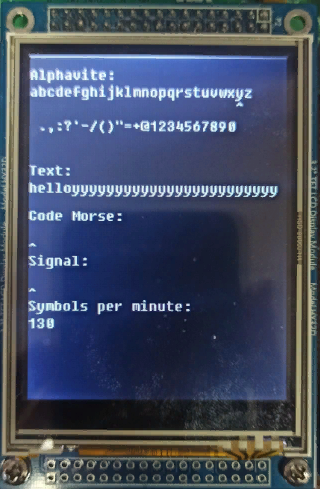


Рисунок 14 – Обработка выхода за границы поля сообщения.

## 5.2 Процесс конвертации в код Морзе

Проверим правильность работы перевода сообщения в код Морзе. Для этого запустим передачу кода Морзе, нажав на соответствующую кнопку. Как видим, начался процесс передачи кода Морзе на светодиод и динамик. При этом на экране начало отображаться текущее состояние перевода сообщения в код Морзе:

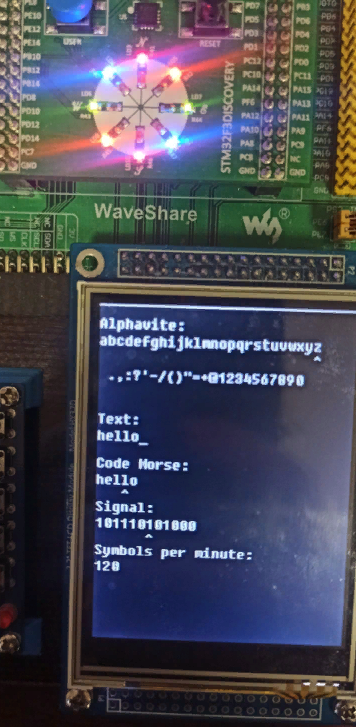


Рисунок 15 – Процесс работы.

Проверим возможность увеличения скорости трансляции при нажатии на соответствующую копку. После нажатия на экране отобразилось новое значение, а скорость воспроизведения увеличилась:

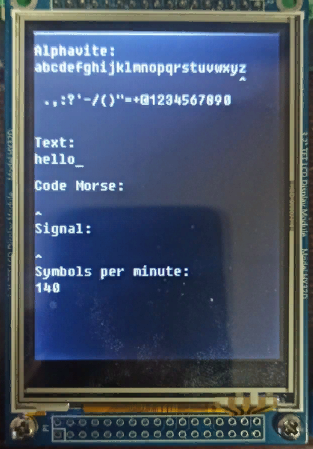


Рисунок 16 – Увеличение скорости передачи.

Аналогичным способом проверим уменьшения скорости передачи сообщения:

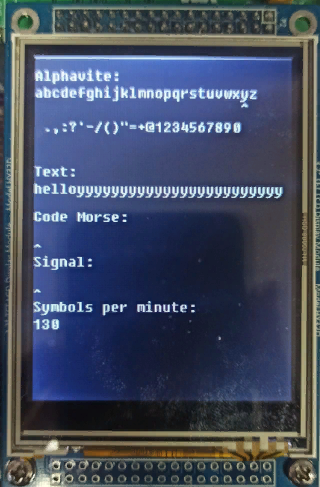


Рисунок 17 – Уменьшение скорости передачи.

Проверим возможность ввода нового сообщения во время проигрывания старого сообщения в виде кода Морзе. Для этого начнем воспроизведение и попробуем изменить сообщение. Как видим, есть возможность вводить новое сообщение в момент проигрывания старого:

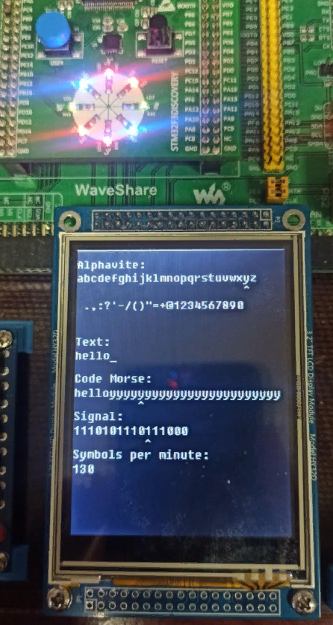


Рисунок 18 – Ввод нового сообщения во время проигрывания кода Морзе.

## 5.3 Итог тестирования

В ходе тестирования программы не было выявлено критических ошибок, влияющих на корректную работу приложения.

Однако, при более комплексном как позитивном, так и негативном тестировании, есть вероятность обнаружить незначительную ошибку / недочет, который в определенной ситуации может вызвать неверный результат или поведения от программы.

## 6. Экономическая оценка

|  |  |
| --- | --- |
| Компонент | Примерная цена |
| Отладочная плата STM32F3 Discovery | 2550 руб. |
| LCD экран ILI9325 | 200 руб. |
| Кнопочная панель 4x4 | 50 руб. |
| Итого: | 2800 руб. |

При массовом производстве, цена на компоненты будет ниже из-за оптовых покупок, что является одним из путей снижения цены.

## 7. Заключение

В результате выполнения данного проекта было осуществлено проектирование приложения на базе микроконтроллера STM32F303VCT6. Были получены основные навыки в области архитектуры и программного обеспечения встроенных систем, а также знания о структуре, функциях и основах программирования микроконтроллеров, позволяющих решать вопросы анализа функционирования программного обеспечения встраиваемых систем.

## 8. Список используемой литературы

1. Лабораторный практикум по изучению ARM микроконтроллеров серии STM32: учебное пособие / А.А. Ключарев, К.А. Кочин; С.-Петербургский государственный ун-т. Аэрокосмического приборостроения - СПб. : Изд-во СПбГУАП, 2022.
2. The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors. Third Edition. Joseph Yiu. ARM Ltd., Cambridge, UK. [электронный ресурс] // URL: <https://www.pdfdrive.com/the-definitive-guide-to-arm-cortex-m3-andcortex-m4processors-e187111520.html>
3. Джозеф Ю. Ядро Cortex-M3 компании ARM. Полное руководство. 2012. ISBN: 978-5-94120-243-0. [электронный ресурс] // URL: <https://bok.xyz/book/2373589/b5c3ad>
4. RM0008. Reference manual STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced Arm®-based 32-bit MCUs [электронный ресурс] // URL: [https://www.st.com/resource/en/reference\_manual/cd00171190- stm32f101xxstm32f102xx-stm32f103xx-stm32f105xx-and-stm32f107xxadvanced-armbased32bit-mcus-stmicroelectronics.pdf](https://www.st.com/resource/en/reference_manual/cd00171190-%20stm32f101xxstm32f102xx-stm32f103xx-stm32f105xx-and-stm32f107xxadvanced-armbased32bit-mcus-stmicroelectronics.pdf)
5. Datasheet STM32F103x8 STM32F103xB Medium-density performance line ARM®- based 32-bit MCU with 64 or 128 KB Flash, USB, CAN, 7 timers, 2 ADCs, 9 com. Interfaces. [электронный ресурс] // URL: <https://www.st.com/resource/en/datasheet/stm32f103c8.pdf>
6. Мартин М. Инсайдерское руководство по STM32 [электронный ресурс] // URL: <https://istarik.ru/file/STM32.pdf>
7. LVGL Documentation v7.11.0. [электронный ресурс] // URL: https://docs.lvgl.io/7.11/

## Приложение

Исходный код программы:

Файл main.c:

#include "main.h"

#include "timers.h"

#include "keypad.h"

#include "displayctrl.h"

void SystemClock\_Config(void);

int main(void)

{

SystemClock\_Config();

//InitMessages(); //инициализация механизма обработки сообщений

InitTimers(); //инициализация таймеров

InitKpd4(); //инициализация автомата Kpd4

InitDisplay(); //инициализация автомата Kpd4

LL\_SYSTICK\_EnableIT();

while (1)

{

if(GetTimer(KPD4\_TIMER)>=5) //каждые 5 мс

{

ProcessKeyKpd4(); //итерация автомата Kpd4

ResetTimer(KPD4\_TIMER);

}

ProcessDisplay();

}

}

void SystemClock\_Config(void)

{ // PLL (HSE) = SYSCLK = HCLK(Hz) = 72000000

LL\_FLASH\_SetLatency(LL\_FLASH\_LATENCY\_2); /\* Set FLASH latency \*/

LL\_RCC\_HSE\_Enable(); /\* Enable HSE and wait for activation\*/

while(LL\_RCC\_HSE\_IsReady() != 1) {}

LL\_RCC\_PLL\_ConfigDomain\_SYS(LL\_RCC\_PLLSOURCE\_HSE\_DIV\_1, LL\_RCC\_PLL\_MUL\_9);

LL\_RCC\_PLL\_Enable();/\* Main PLL configuration and activation \*/

while(LL\_RCC\_PLL\_IsReady() != 1) {}

LL\_RCC\_SetAHBPrescaler(LL\_RCC\_SYSCLK\_DIV\_1); //AHB Prescaler=1

LL\_RCC\_SetSysClkSource(LL\_RCC\_SYS\_CLKSOURCE\_PLL);

while(LL\_RCC\_GetSysClkSource() != LL\_RCC\_SYS\_CLKSOURCE\_STATUS\_PLL){}

LL\_RCC\_SetAPB1Prescaler(LL\_RCC\_APB1\_DIV\_2); //APB1 Prescaler=2

LL\_RCC\_SetAPB2Prescaler(LL\_RCC\_APB2\_DIV\_1); //APB2 Prescaler=1

LL\_Init1msTick(72000000);

LL\_SetSystemCoreClock(72000000);

}

Файл keypad.h:

/\*keypad.h\*/

#include "stdint.h"

#ifndef KEYPAD\_H

#define KEYPAD\_H

void InitKpd4(void);

void ProcessKeyKpd4(void);

uint16\_t getKeycode();

#endif

Файл keypad.c:

/\*keypad.c\*/

#include "timers.h"

#include "keypad.h"

#include <stdio.h>

#include "main.h"

#define DEBOUNCE 10 // задержка на дребезг 10 мс

#define FIRST\_DELAY 500 // задержка первого удержания кнопки 500 мс

#define AUTO\_REPEAT 300 // задержка повторного удержания 300 мс

/\*состояния автомата\*/

#define FSMST\_IDLE 0 // ожидает нажатия на кнопку

#define FSMST\_DEBOUNCE 1 // ожидание завершения дребезга при замыкании контактов

#define FSMST\_KEYDOWN 2 // фиксируется нажатие и формируется сообщение MSG\_KEY\_PRESSED

// если кнопка не нажата, возврат в состояние FSMST\_IDLE

#define FSMST\_KEYDOWNHOLD 3 // формируется сообщение MSG\_KEY\_PRESSED при удержании кнопки в течении времени FIRST\_DELAY

// если кнопка отпущена, возврат в состояние FSMST\_IDLE

#define FSMST\_HOLDAUTO 4 // формируется сообщение MSG\_KEY\_PRESSED при удержании кнопки в течении времени AUTO\_REPEAT

// если кнопка отпущена, возврат в состояние FSMST\_IDLE\*/

uint16\_t scankeypad(void);

static char fsmstate; // переменная состояния автомата

static uint16\_t keycode; // переменная состояния кнопок

static uint16\_t keycodeOut = 0;

uint16\_t getKeycode() {

uint16\_t temp = keycodeOut;

keycodeOut = 0;

return temp;

}

void ProcessKeyKpd4(void)

{

uint16\_t key\_code\_tmp=0;

key\_code\_tmp = scankeypad();

switch (fsmstate)

{

case FSMST\_IDLE: //ни одна кнопка не нажата

if (key\_code\_tmp > 0)

{

keycode = key\_code\_tmp;

ResetTimer(KEYB\_TIMER);

fsmstate = FSMST\_DEBOUNCE;

}

break;

case FSMST\_DEBOUNCE: //кнопка нажата ждём окончания переходного процесса

if (GetTimer(KEYB\_TIMER) > DEBOUNCE)

fsmstate = FSMST\_KEYDOWN;

break;

case FSMST\_KEYDOWN: //если кнопка нажата, посылаем сообщение

if ( key\_code\_tmp == keycode )

{

ResetTimer(KEYB\_TIMER);

//SendMessage(MSG\_KEY\_PRESSED, (void\*)&keycode);

//SendMessage(MSG\_DISPUPDTKEY, (void\*)&keycode);

keycodeOut = keycode;

fsmstate = FSMST\_KEYDOWNHOLD;

}

else

fsmstate = FSMST\_IDLE;

break;

case FSMST\_KEYDOWNHOLD: //если кнопка удерживается, посылаем сообщение

if ( key\_code\_tmp == keycode )

{

if (GetTimer(KEYB\_TIMER) >= FIRST\_DELAY)

{

ResetTimer(KEYB\_TIMER);

//SendMessage(MSG\_KEY\_PRESSED, (void\*)&keycode);

//SendMessage(MSG\_DISPUPDTKEY, (void\*)&keycode);

keycodeOut = keycode;

fsmstate = FSMST\_HOLDAUTO;

}

}

else

fsmstate = FSMST\_IDLE;

break;

case FSMST\_HOLDAUTO: //если кнопка удерживается, посылаем сообщение чаще

if (key\_code\_tmp == keycode)

{

if (GetTimer(KEYB\_TIMER) >= AUTO\_REPEAT)

{

ResetTimer(KEYB\_TIMER);

//SendMessage(MSG\_KEY\_PRESSED, (void\*)&keycode);

//SendMessage(MSG\_DISPUPDTKEY, (void\*)&keycode);

keycodeOut = keycode;

}

}

else

fsmstate = FSMST\_IDLE;

break;

}

}

uint16\_t scankeypad(void)

{

uint16\_t key\_code=0;

uint8\_t tmpbuf=0;

int i;

for(i=0; i<4; i++)

{

LL\_GPIO\_ResetOutputPin(GPIOC,1<<(4+i));

\_\_NOP(); \_\_NOP(); \_\_NOP(); \_\_NOP();

tmpbuf= (uint8\_t)LL\_GPIO\_ReadInputPort(GPIOC);

tmpbuf= ~tmpbuf & 0x0F;

key\_code |=tmpbuf<<(4\*i);

LL\_GPIO\_SetOutputPin(GPIOC,1<<(4+i));

}

return key\_code;

}

void InitKpd4(void)

{

fsmstate=FSMST\_IDLE;

LL\_AHB1\_GRP1\_EnableClock(LL\_AHB1\_GRP1\_PERIPH\_GPIOC);//GPIOC

//PC0,PC1,PC2,PC3 In

LL\_GPIO\_InitTypeDef gpio\_initstruct;

gpio\_initstruct.Pin = LL\_GPIO\_PIN\_0|LL\_GPIO\_PIN\_1|LL\_GPIO\_PIN\_2|LL\_GPIO\_PIN\_3;

gpio\_initstruct.Mode = LL\_GPIO\_MODE\_INPUT;

gpio\_initstruct.Pull = LL\_GPIO\_PULL\_UP;

if (LL\_GPIO\_Init(GPIOC, &gpio\_initstruct) != SUCCESS)

while (1){} /\* Initialization Error \*/

//PC4,PC5,PC6,PC7 Out

gpio\_initstruct.Pin = LL\_GPIO\_PIN\_4|LL\_GPIO\_PIN\_5|LL\_GPIO\_PIN\_6|LL\_GPIO\_PIN\_7;

gpio\_initstruct.Mode = LL\_GPIO\_MODE\_OUTPUT;

gpio\_initstruct.OutputType = LL\_GPIO\_OUTPUT\_PUSHPULL;

gpio\_initstruct.Pull = LL\_GPIO\_PULL\_UP;

if (LL\_GPIO\_Init(GPIOC, &gpio\_initstruct) != SUCCESS)

while (1){} /\* Initialization Error \*/

LL\_GPIO\_SetOutputPin (GPIOC, LL\_GPIO\_PIN\_4|LL\_GPIO\_PIN\_5|LL\_GPIO\_PIN\_6|LL\_GPIO\_PIN\_7);

}

Файл timers.h:

/\*timers.h\*/

#ifndef TIMERS\_h

#define TIMERS\_h

#define MAX\_TIMERS 6 //максимальное количество таймеров

//в этом разделе объявляются константы, служащие идентификаторами таймеров.

/\* таймеры keypad \*/

#define KEYB\_TIMER 0 // таймер удержания кнопок

#define KPD4\_TIMER 1 // период опроса состояния кнопок

/\* таймер дисплея \*/

#define DISPUPDT\_TIMER 2 // таймер обновления экрана

#define PLAYTIMER 3

#define SOUNDTIMER 4

#define ENTERTIMER 5

//функции работы с таймерами

void ProcessTimers(void);

void InitTimers(void);

unsigned int GetTimer(char Timer);

void ResetTimer(char Timer);

#endif

Файл timers.c:

#include "timers.h"

static unsigned int v\_timers[MAX\_TIMERS]; //переменные «виртуальных» таймеров

void ProcessTimers(void) //обработчик прерывания таймера/счетчика

{ /\*увеличиваем значение всех переменных-таймеров на 1\*/

int i;

for(i = 0; i < MAX\_TIMERS; i++)

v\_timers[i]++;

}

void InitTimers(void)

{

int i;

for(i = 0; i < MAX\_TIMERS; i++)

v\_timers[i] = 0;

}

unsigned int GetTimer(char Timer)

{

return v\_timers[Timer];

}

void ResetTimer(char Timer)

{

v\_timers[Timer] = 0;

}

void SysTick\_Handler(void)

{

ProcessTimers();

}

Файл displayctrl.h:

/\* displayctrl.h \*/

#ifndef DISPLAYCTRL\_H

#define DISPLAYCTRL\_H

void InitDisplay(void);

void ProcessDisplay(void);

#endif

Файл displayctrl.c:

/\* displayctrl.c \*/

#include "timers.h"

#include "main.h"

#include "AsciiLib.h"

#include "LCD\_ILI9325.h"

#include "displayctrl.h"

#include "keypad.h"

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#define SOUND\_PITCH 4

void dispkypd4(uint16\_t kypd4status);

////////////////////////////////////////////////////////////////////////////////

/\* reverse: переворачиваем строку s на месте \*/

void reverse(char s[])

{

int i, j;

char c;

for (i = 0, j = strlen(s)-1; i<j; i++, j--) {

c = s[i];

s[i] = s[j];

s[j] = c;

}

}

/\* itoa: конвертируем n в символы в s \*/

void itoa(int n, char s[])

{

int i, sign;

if ((sign = n) < 0) /\* записываем знак \*/

n = -n; /\* делаем n положительным числом \*/

i = 0;

do { /\* генерируем цифры в обратном порядке \*/

s[i++] = n % 10 + '0'; /\* берем следующую цифру \*/

} while ((n /= 10) > 0); /\* удаляем \*/

if (sign < 0)

s[i++] = '-';

s[i] = '\0';

reverse(s);

}

////////////////////////////////////////////////////////////////////////////////

const char morseCode[50][6] = {

{'.','-','0','0','0','0'}, //1 a

{'-','.','.','.','0','0'}, //2 b

{'-','.','-','.','0','0'}, //3 c

{'-','.','.','0','0','0'}, //4 d

{'.','0','0','0','0','0'}, //5 e

{'.','.','-','.','0','0'}, //6 f

{'-','-','.','0','0','0'}, //7 g

{'.','.','.','.','0','0'}, //8 h

{'.','.','0','0','0','0'}, //9 i

{'.','-','-','-','0','0'}, //10 j

{'-','.','-','0','0','0'}, //11 k

{'.','-','.','.','0','0'}, //12 l

{'-','-','0','0','0','0'}, //13 m

{'-','.','0','0','0','0'}, //14 n

{'-','-','-','0','0','0'}, //15 o

{'.','-','-','.','0','0'}, //16 p

{'-','-','.','-','0','0'}, //17 q

{'.','-','.','0','0','0'}, //18 r

{'.','.','.','0','0','0'}, //19 s

{'-','0','0','0','0','0'}, //20 t

{'.','.','-','0','0','0'}, //21 u

{'.','.','.','-','0','0'}, //22 v

{'.','-','-','0','0','0'}, //23 w

{'-','.','.','-','0','0'}, //24 x

{'-','.','-','-','0','0'}, //25 y

{'-','-','.','.','0','0'}, //26 z

{'0','0','0','0','0','0'}, //27 space

{'.','-','.','-','.','-'}, //28 .

{'-','-','.','.','-','-'}, //29 ,

{'-','-','-','.','.','.'}, //30 :

{'.','.','-','-','.','.'}, //31 ?

{'.','-','-','-','-','.'}, //32 '

{'-','.','.','.','.','-'}, //33 -

{'-','.','.','-','.','0'}, //34 /

{'-','.','-','-','.','0'}, //35 (

{'-','.','-','-','.','-'}, //36 )

{'.','-','.','.','-','.'}, //37 "

{'-','.','.','.','-','0'}, //38 =

{'.','-','.','-','.','0'}, //39 +

{'.','-','-','.','-','.'}, //40 @

{'.','-','-','-','-','0'}, //41 1

{'.','.','-','-','-','0'}, //42 2

{'.','.','.','-','-','0'}, //43 3

{'.','.','.','.','-','0'}, //44 4

{'.','.','.','.','.','0'}, //45 5

{'-','.','.','.','.','0'}, //46 6

{'-','-','.','.','.','0'}, //47 7

{'-','-','-','.','.','0'}, //48 8

{'-','-','-','-','.','0'}, //49 9

{'-','-','-','-','-','0'}, //50 0

};

char alphavite1[27] = "abcdefghijklmnopqrstuvwxyz";

char alphavite2[25] = " .,:?'-/()\"=+@1234567890";

int selectedX, selectedY, currentX;

int morseX, morseSize, instructionsIndex;

char isPlayedMorse;

char isHideSignal, isHideSignalSound;

char selectedStr[31];

char currentText[31];

char morseText[31];

char instructions[23];

int speed;

int speedInterval;

////////////////////////////////////////////////////////////////////////////////

int toSpeedInterval(int speed) {

float symToSec = (float)speed / 60.0f;

int si = 500.0f / symToSec;

return si;

}

void initSelectedStr(int x, int y) {

for (int i=0; i<30; ++i) {

selectedStr[i] = ' ';

}

if (y == -1 || y == selectedY) {

if (x >= 0) {

selectedStr[x] = '^';

}

}

}

void initData() {

RCC->AHBENR |= RCC\_AHBENR\_GPIOAEN | RCC\_AHBENR\_GPIOEEN | RCC\_AHBENR\_GPIOFEN;

GPIOE->MODER |= 0x55550000;

char roll=0x1F;

SET\_BIT(RCC->AHBENR,RCC\_AHBENR\_GPIOFEN);

//устанавливаем работу линий PB11 на вывод

SET\_BIT(GPIOA->MODER, GPIO\_MODER\_MODER4\_0);

SET\_BIT(GPIOF->MODER, GPIO\_MODER\_MODER2\_0);

// подтягиваем PB11 к питанию Pull-up

//SET\_BIT(GPIOA->PUPDR, GPIO\_PUPDR\_PUPDR4\_0);

//SET\_BIT(GPIOF->PUPDR, GPIO\_PUPDR\_PUPDR2\_0);

selectedX = 0;

selectedY = 0;

currentX = 0;

morseX = 0;

isPlayedMorse = 0;

isHideSignal = 0;

isHideSignalSound = 0;

for (int i=0; i<30; ++i) {

selectedStr[i] = ' ';

currentText[i] = ' ';

morseText[i] = '\0';

}

selectedStr[30] = '\0';

currentText[30] = '\0';

morseText[30] = '\0';

instructions[22] = '\0';

speed = 120;

speedInterval = toSpeedInterval(speed);

}

void selectLeft() {

--selectedX;

if (selectedX < 0) selectedX = 0;

}

void selectRight() {

++selectedX;

if (selectedY == 0) {

if (selectedX > 25) selectedX = 25;

}

else {

if (selectedX > 23) selectedX = 23;

}

}

void selectUp() {

--selectedY;

if (selectedY < 0) selectedY = 0;

}

void selectDown() {

++selectedY;

if (selectedY > 1) selectedY = 1;

if (selectedX > 23) selectedX = 23;

}

void addChar() {

if (currentX == 29) return;

if (selectedY == 0) {

currentText[currentX] = alphavite1[selectedX];

}

else {

currentText[currentX] = alphavite2[selectedX];

}

++currentX;

}

void deleteChar() {

--currentX;

if (currentX < 0) {

currentX = 0;

return;

}

if (currentText[currentX+1] == '\_') {

currentText[currentX+1] = ' ';

currentText[currentX] = '\_';

}

else {

currentText[currentX] = ' ';

}

}

void changeEnterSym() {

if (currentText[currentX] == '\_') {

currentText[currentX] = ' ';

}

else {

currentText[currentX] = '\_';

}

}

void initInstructions(char sym) {

int si = -1;

for (int i=0; i<26; ++i) {

if (sym == alphavite1[i]) {

si = i;

break;

}

}

if (si == -1) {

for (int i=0; i<24; ++i) {

if (sym == alphavite2[i]) {

si = i + 26;

break;

}

}

}

for (int i=0; i<22; ++i) {

instructions[i] = '\0';

}

int ii = 0;

if (sym == ' ') {

instructions[ii] = '0';

instructions[ii+1] = '0';

instructions[ii+2] = '0';

instructions[ii+3] = '0';

}

else {

for (int i=0; i<6; ++i) {

if (morseCode[si][i] == '.') {

instructions[ii] = '1';

++ii;

instructions[ii] = '0';

++ii;

}

else if (morseCode[si][i] == '-') {

instructions[ii] = '1';

++ii;

instructions[ii] = '1';

++ii;

instructions[ii] = '1';

++ii;

instructions[ii] = '0';

++ii;

}

else if (morseCode[si][i] == '0') {

instructions[ii] = '0';

++ii;

instructions[ii] = '0';

++ii;

break;

}

if (i == 5) {

instructions[ii] = '0';

++ii;

instructions[ii] = '0';

++ii;

}

}

}

}

void playSignal();

void startSignal() {

//showRect = true; //

char roll=0xFF;

GPIOE->BSRR = roll<<8;

//isHideSignal = 1;

//isHideSignalSound = 0;

//ResetTimer(SOUNDTIMER);

isHideSignal = 1;

isHideSignalSound = 1;

GPIOA->BSRR= GPIO\_ODR\_4;

ResetTimer(SOUNDTIMER);

}

void endSignal() {

//showRect = false; //

char roll=0xFF;

GPIOE->BRR = (roll)<<8;

//isHideSignal = 0;

//isHideSignalSound = 0;

isHideSignal = 0;

isHideSignalSound = 0;

GPIOA->BRR = GPIO\_ODR\_4;

}

void startMorse() {

morseSize = currentX;

int i = 0;

for (; i<morseSize; ++i) {

morseText[i] = currentText[i];

}

for (; i<31; ++i) {

morseText[i] = '\0';

}

morseX = 0;

instructionsIndex = 0;

if (morseSize == 0) {

for (int i=0; i<22; ++i) {

instructions[i] = '\0';

}

isPlayedMorse = 0;

endSignal();

return;

}

initInstructions(morseText[morseX]);

isPlayedMorse = 1;

ResetTimer(PLAYTIMER);

instructionsIndex = -1;

playSignal();

}

void playSignal() {

++instructionsIndex;

if (instructions[instructionsIndex] == '1') {

startSignal();

}

else if (instructions[instructionsIndex] == '0') {

endSignal();

}

else if (instructions[instructionsIndex] == '\0') {

++morseX;

if (morseX >= morseSize) {

morseX = 0;

isPlayedMorse = 0;

endSignal();

instructionsIndex = 0;

for (int i=0; i<30; ++i) {

morseText[i] = '\0';

}

for (int i=0; i<22; ++i) {

instructions[i] = '\0';

}

}

else {

instructionsIndex = -1;

initInstructions(morseText[morseX]);

playSignal();

}

}

}

void speedUp() {

speed = speed + 10;

if (speed > 240) {

speed = 240;

}

speedInterval = toSpeedInterval(speed);

}

void speedDown() {

speed = speed - 10;

if (speed < 60) {

speed = 60;

}

speedInterval = toSpeedInterval(speed);

}

////////////////////////////////////////////////////////////////////////////////

void InitDisplay(void)

{

initData();

init\_lcd\_ili9325();

}

void ProcessDisplay(void)

{

if (GetTimer(ENTERTIMER) > 500) {

ResetTimer(ENTERTIMER);

changeEnterSym();

}

if (isPlayedMorse == 1 && GetTimer(PLAYTIMER) > speedInterval) {

ResetTimer(PLAYTIMER);

playSignal();

}

if (isHideSignal == 1) {

if (GetTimer(SOUNDTIMER) > SOUND\_PITCH) { // 4

ResetTimer(SOUNDTIMER);

if (isHideSignalSound == 1) {

isHideSignalSound = 0;

GPIOA->BRR = GPIO\_ODR\_4;

}

else {

isHideSignalSound = 1;

GPIOA->BSRR= GPIO\_ODR\_4;

}

}

}

/\* каждые 25 мс обновляем поля экрана: 40 кадр/сек \*/

if(GetTimer(DISPUPDT\_TIMER)>=25)

{

ResetTimer(DISPUPDT\_TIMER);

uint16\_t key = getKeycode();

if (key != 0) {

dispkypd4(key);

}

int y = 10;

LCDprintstr("Alphavite:", y, LCD\_COLOR\_BLACK, LCD\_COLOR\_WHITE);

y = y + 16;

LCDprintstr(alphavite1, y, LCD\_COLOR\_BLACK, LCD\_COLOR\_WHITE);

initSelectedStr(selectedX, 0);

LCDprintstr(selectedStr, y+16, LCD\_COLOR\_BLACK, LCD\_COLOR\_WHITE);

y = y + 16 + 16;

LCDprintstr(alphavite2, y, LCD\_COLOR\_BLACK, LCD\_COLOR\_WHITE);

initSelectedStr(selectedX, 1);

LCDprintstr(selectedStr, y+16, LCD\_COLOR\_BLACK, LCD\_COLOR\_WHITE);

y = y + 16 + 16;

y = y + 10;

LCDprintstr("Text:", y, LCD\_COLOR\_BLACK, LCD\_COLOR\_WHITE);

LCDprintstr(currentText, y+16, LCD\_COLOR\_BLACK, LCD\_COLOR\_WHITE);

y = y + 16 + 16;

y = y + 10;

LCDprintstr("Code Morse:", y, LCD\_COLOR\_BLACK, LCD\_COLOR\_WHITE);

LCDprintstr(morseText, y+16, LCD\_COLOR\_BLACK, LCD\_COLOR\_WHITE);

initSelectedStr(morseX, -1);

LCDprintstr(selectedStr, y+16+16, LCD\_COLOR\_BLACK, LCD\_COLOR\_WHITE);

y = y + 16 + 16;

y = y + 10;

LCDprintstr("Signal:", y, LCD\_COLOR\_BLACK, LCD\_COLOR\_WHITE);

LCDprintstr(instructions, y+16, LCD\_COLOR\_BLACK, LCD\_COLOR\_WHITE);

initSelectedStr(instructionsIndex, -1);

LCDprintstr(selectedStr, y+16+16, LCD\_COLOR\_BLACK, LCD\_COLOR\_WHITE);

y = y + 16 + 16;

char number [5];

itoa(speed, number);

y = y + 10;

LCDprintstr("Symbols per minute:", y, LCD\_COLOR\_BLACK, LCD\_COLOR\_WHITE);

LCDprintstr(number, y+16, LCD\_COLOR\_BLACK, LCD\_COLOR\_WHITE);

}

}

void dispkypd4(uint16\_t kypd4status)

{

uint32\_t i;

char textbuf[17]="";

int count=0;

for(i=0;i<16;i++)

{

if( kypd4status & (1<<i))

{

if(i==4) {

selectLeft();

}

if(i==6) {

selectRight();

}

if(i==1) {

selectUp();

}

if(i==5) {

selectDown();

}

if(i==2) {

addChar();

}

if(i==0) {

deleteChar();

}

if(i==3) {

startMorse();

}

if(i==8) {

speedDown();

}

if(i==9) {

speedUp();

}

textbuf[count+0] = 'K';

textbuf[count+1] = 0x30+((i+1)/10);

textbuf[count+2] = 0x30+((i+1)%10);

textbuf[count+3] = ' ';

count += 4;

}

}

textbuf[count] = 0;

if( count > 16 )

strcpy(textbuf, "Error");

}

Файл asciiLib.h:

#ifndef \_\_AsciiLib\_H

#define \_\_AsciiLib\_H

#include <stdint.h>

//шрифты LCD

void GetASCIICode(unsigned char\* pBuffer,unsigned char ASCII);

void LCDprintstr(char\* str, int16\_t y,uint16\_t bkcolor, uint16\_t charcolor);

#endif

Файл asciiLib.c:

#include "AsciiLib.h"

#include "LCD\_ILI9325.h"

#include <string.h>

static unsigned char const AsciiLib[95][16] = {

{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/\*" ",0\*/

{0x00,0x00,0x00,0x18,0x3C,0x3C,0x3C,0x18,0x18,0x00,0x18,0x18,0x00,0x00,0x00,0x00},/\*"!",1\*/

{0x00,0x00,0x00,0x66,0x66,0x66,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/\*""",2\*/

{0x00,0x00,0x00,0x36,0x36,0x7F,0x36,0x36,0x36,0x7F,0x36,0x36,0x00,0x00,0x00,0x00},/\*"#",3\*/

{0x00,0x18,0x18,0x3C,0x66,0x60,0x30,0x18,0x0C,0x06,0x66,0x3C,0x18,0x18,0x00,0x00},/\*"$",4\*/

{0x00,0x00,0x70,0xD8,0xDA,0x76,0x0C,0x18,0x30,0x6E,0x5B,0x1B,0x0E,0x00,0x00,0x00},/\*"%",5\*/

{0x00,0x00,0x00,0x38,0x6C,0x6C,0x38,0x60,0x6F,0x66,0x66,0x3B,0x00,0x00,0x00,0x00},/\*"&",6\*/

{0x00,0x00,0x00,0x18,0x18,0x18,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/\*"'",7\*/

{0x00,0x00,0x00,0x0C,0x18,0x18,0x30,0x30,0x30,0x30,0x30,0x18,0x18,0x0C,0x00,0x00},/\*"(",8\*/

{0x00,0x00,0x00,0x30,0x18,0x18,0x0C,0x0C,0x0C,0x0C,0x0C,0x18,0x18,0x30,0x00,0x00},/\*")",9\*/

{0x00,0x00,0x00,0x00,0x00,0x36,0x1C,0x7F,0x1C,0x36,0x00,0x00,0x00,0x00,0x00,0x00},/\*"\*",10\*/

{0x00,0x00,0x00,0x00,0x00,0x18,0x18,0x7E,0x18,0x18,0x00,0x00,0x00,0x00,0x00,0x00},/\*"+",11\*/

{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1C,0x1C,0x0C,0x18,0x00,0x00},/\*",",12\*/

{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7E,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/\*"-",13\*/

{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1C,0x1C,0x00,0x00,0x00,0x00},/\*".",14\*/

{0x00,0x00,0x00,0x06,0x06,0x0C,0x0C,0x18,0x18,0x30,0x30,0x60,0x60,0x00,0x00,0x00},/\*"/",15\*/

{0x00,0x00,0x00,0x1E,0x33,0x37,0x37,0x33,0x3B,0x3B,0x33,0x1E,0x00,0x00,0x00,0x00},/\*"0",16\*/

{0x00,0x00,0x00,0x0C,0x1C,0x7C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x00,0x00,0x00,0x00},/\*"1",17\*/

{0x00,0x00,0x00,0x3C,0x66,0x66,0x06,0x0C,0x18,0x30,0x60,0x7E,0x00,0x00,0x00,0x00},/\*"2",18\*/

{0x00,0x00,0x00,0x3C,0x66,0x66,0x06,0x1C,0x06,0x66,0x66,0x3C,0x00,0x00,0x00,0x00},/\*"3",19\*/

{0x00,0x00,0x00,0x30,0x30,0x36,0x36,0x36,0x66,0x7F,0x06,0x06,0x00,0x00,0x00,0x00},/\*"4",20\*/

{0x00,0x00,0x00,0x7E,0x60,0x60,0x60,0x7C,0x06,0x06,0x0C,0x78,0x00,0x00,0x00,0x00},/\*"5",21\*/

{0x00,0x00,0x00,0x1C,0x18,0x30,0x7C,0x66,0x66,0x66,0x66,0x3C,0x00,0x00,0x00,0x00},/\*"6",22\*/

{0x00,0x00,0x00,0x7E,0x06,0x0C,0x0C,0x18,0x18,0x30,0x30,0x30,0x00,0x00,0x00,0x00},/\*"7",23\*/

{0x00,0x00,0x00,0x3C,0x66,0x66,0x76,0x3C,0x6E,0x66,0x66,0x3C,0x00,0x00,0x00,0x00},/\*"8",24\*/

{0x00,0x00,0x00,0x3C,0x66,0x66,0x66,0x66,0x3E,0x0C,0x18,0x38,0x00,0x00,0x00,0x00},/\*"9",25\*/

{0x00,0x00,0x00,0x00,0x00,0x1C,0x1C,0x00,0x00,0x00,0x1C,0x1C,0x00,0x00,0x00,0x00},/\*":",26\*/

{0x00,0x00,0x00,0x00,0x00,0x1C,0x1C,0x00,0x00,0x00,0x1C,0x1C,0x0C,0x18,0x00,0x00},/\*";",27\*/

{0x00,0x00,0x00,0x06,0x0C,0x18,0x30,0x60,0x30,0x18,0x0C,0x06,0x00,0x00,0x00,0x00},/\*"<",28\*/

{0x00,0x00,0x00,0x00,0x00,0x00,0x7E,0x00,0x7E,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/\*"=",29\*/

{0x00,0x00,0x00,0x60,0x30,0x18,0x0C,0x06,0x0C,0x18,0x30,0x60,0x00,0x00,0x00,0x00},/\*">",30\*/

{0x00,0x00,0x00,0x3C,0x66,0x66,0x0C,0x18,0x18,0x00,0x18,0x18,0x00,0x00,0x00,0x00},/\*"?",31\*/

{0x00,0x00,0x00,0x7E,0xC3,0xC3,0xCF,0xDB,0xDB,0xCF,0xC0,0x7F,0x00,0x00,0x00,0x00},/\*"@",32\*/

{0x00,0x00,0x00,0x18,0x3C,0x66,0x66,0x66,0x7E,0x66,0x66,0x66,0x00,0x00,0x00,0x00},/\*"A",33\*/

{0x00,0x00,0x00,0x7C,0x66,0x66,0x66,0x7C,0x66,0x66,0x66,0x7C,0x00,0x00,0x00,0x00},/\*"B",34\*/

{0x00,0x00,0x00,0x3C,0x66,0x66,0x60,0x60,0x60,0x66,0x66,0x3C,0x00,0x00,0x00,0x00},/\*"C",35\*/

{0x00,0x00,0x00,0x78,0x6C,0x66,0x66,0x66,0x66,0x66,0x6C,0x78,0x00,0x00,0x00,0x00},/\*"D",36\*/

{0x00,0x00,0x00,0x7E,0x60,0x60,0x60,0x7C,0x60,0x60,0x60,0x7E,0x00,0x00,0x00,0x00},/\*"E",37\*/

{0x00,0x00,0x00,0x7E,0x60,0x60,0x60,0x7C,0x60,0x60,0x60,0x60,0x00,0x00,0x00,0x00},/\*"F",38\*/

{0x00,0x00,0x00,0x3C,0x66,0x66,0x60,0x60,0x6E,0x66,0x66,0x3E,0x00,0x00,0x00,0x00},/\*"G",39\*/

{0x00,0x00,0x00,0x66,0x66,0x66,0x66,0x7E,0x66,0x66,0x66,0x66,0x00,0x00,0x00,0x00},/\*"H",40\*/

{0x00,0x00,0x00,0x3C,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x3C,0x00,0x00,0x00,0x00},/\*"I",41\*/

{0x00,0x00,0x00,0x06,0x06,0x06,0x06,0x06,0x06,0x66,0x66,0x3C,0x00,0x00,0x00,0x00},/\*"J",42\*/

{0x00,0x00,0x00,0x66,0x66,0x6C,0x6C,0x78,0x6C,0x6C,0x66,0x66,0x00,0x00,0x00,0x00},/\*"K",43\*/

{0x00,0x00,0x00,0x60,0x60,0x60,0x60,0x60,0x60,0x60,0x60,0x7E,0x00,0x00,0x00,0x00},/\*"L",44\*/

{0x00,0x00,0x00,0x63,0x63,0x77,0x6B,0x6B,0x6B,0x63,0x63,0x63,0x00,0x00,0x00,0x00},/\*"M",45\*/

{0x00,0x00,0x00,0x63,0x63,0x73,0x7B,0x6F,0x67,0x63,0x63,0x63,0x00,0x00,0x00,0x00},/\*"N",46\*/

{0x00,0x00,0x00,0x3C,0x66,0x66,0x66,0x66,0x66,0x66,0x66,0x3C,0x00,0x00,0x00,0x00},/\*"O",47\*/

{0x00,0x00,0x00,0x7C,0x66,0x66,0x66,0x7C,0x60,0x60,0x60,0x60,0x00,0x00,0x00,0x00},/\*"P",48\*/

{0x00,0x00,0x00,0x3C,0x66,0x66,0x66,0x66,0x66,0x66,0x66,0x3C,0x0C,0x06,0x00,0x00},/\*"Q",49\*/

{0x00,0x00,0x00,0x7C,0x66,0x66,0x66,0x7C,0x6C,0x66,0x66,0x66,0x00,0x00,0x00,0x00},/\*"R",50\*/

{0x00,0x00,0x00,0x3C,0x66,0x60,0x30,0x18,0x0C,0x06,0x66,0x3C,0x00,0x00,0x00,0x00},/\*"S",51\*/

{0x00,0x00,0x00,0x7E,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x00,0x00,0x00,0x00},/\*"T",52\*/

{0x00,0x00,0x00,0x66,0x66,0x66,0x66,0x66,0x66,0x66,0x66,0x3C,0x00,0x00,0x00,0x00},/\*"U",53\*/

{0x00,0x00,0x00,0x66,0x66,0x66,0x66,0x66,0x66,0x66,0x3C,0x18,0x00,0x00,0x00,0x00},/\*"V",54\*/

{0x00,0x00,0x00,0x63,0x63,0x63,0x6B,0x6B,0x6B,0x36,0x36,0x36,0x00,0x00,0x00,0x00},/\*"W",55\*/

{0x00,0x00,0x00,0x66,0x66,0x34,0x18,0x18,0x2C,0x66,0x66,0x66,0x00,0x00,0x00,0x00},/\*"X",56\*/

{0x00,0x00,0x00,0x66,0x66,0x66,0x66,0x3C,0x18,0x18,0x18,0x18,0x00,0x00,0x00,0x00},/\*"Y",57\*/

{0x00,0x00,0x00,0x7E,0x06,0x06,0x0C,0x18,0x30,0x60,0x60,0x7E,0x00,0x00,0x00,0x00},/\*"Z",58\*/

{0x00,0x00,0x00,0x3C,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x3C,0x00},/\*"[",59\*/

{0x00,0x00,0x00,0x60,0x60,0x30,0x30,0x18,0x18,0x0C,0x0C,0x06,0x06,0x00,0x00,0x00},/\*"\",60\*/

{0x00,0x00,0x00,0x3C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x3C,0x00},/\*"]",61\*/

{0x00,0x18,0x3C,0x66,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/\*"^",62\*/

{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0x00},/\*"\_",63\*/

{0x00,0x00,0x00,0x18,0x18,0x18,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/\*"'",64\*/

{0x00,0x00,0x00,0x00,0x00,0x3C,0x06,0x06,0x3E,0x66,0x66,0x3E,0x00,0x00,0x00,0x00},/\*"a",65\*/

{0x00,0x00,0x00,0x60,0x60,0x7C,0x66,0x66,0x66,0x66,0x66,0x7C,0x00,0x00,0x00,0x00},/\*"b",66\*/

{0x00,0x00,0x00,0x00,0x00,0x3C,0x66,0x60,0x60,0x60,0x66,0x3C,0x00,0x00,0x00,0x00},/\*"c",67\*/

{0x00,0x00,0x00,0x06,0x06,0x3E,0x66,0x66,0x66,0x66,0x66,0x3E,0x00,0x00,0x00,0x00},/\*"d",68\*/

{0x00,0x00,0x00,0x00,0x00,0x3C,0x66,0x66,0x7E,0x60,0x60,0x3C,0x00,0x00,0x00,0x00},/\*"e",69\*/

{0x00,0x00,0x00,0x1E,0x30,0x30,0x30,0x7E,0x30,0x30,0x30,0x30,0x00,0x00,0x00,0x00},/\*"f",70\*/

{0x00,0x00,0x00,0x00,0x00,0x3E,0x66,0x66,0x66,0x66,0x66,0x3E,0x06,0x06,0x7C,0x00},/\*"g",71\*/

{0x00,0x00,0x00,0x60,0x60,0x7C,0x66,0x66,0x66,0x66,0x66,0x66,0x00,0x00,0x00,0x00},/\*"h",72\*/

{0x00,0x00,0x18,0x18,0x00,0x78,0x18,0x18,0x18,0x18,0x18,0x7E,0x00,0x00,0x00,0x00},/\*"i",73\*/

{0x00,0x00,0x0C,0x0C,0x00,0x3C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x78,0x00},/\*"j",74\*/

{0x00,0x00,0x00,0x60,0x60,0x66,0x66,0x6C,0x78,0x6C,0x66,0x66,0x00,0x00,0x00,0x00},/\*"k",75\*/

{0x00,0x00,0x00,0x78,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x7E,0x00,0x00,0x00,0x00},/\*"l",76\*/

{0x00,0x00,0x00,0x00,0x00,0x7E,0x6B,0x6B,0x6B,0x6B,0x6B,0x63,0x00,0x00,0x00,0x00},/\*"m",77\*/

{0x00,0x00,0x00,0x00,0x00,0x7C,0x66,0x66,0x66,0x66,0x66,0x66,0x00,0x00,0x00,0x00},/\*"n",78\*/

{0x00,0x00,0x00,0x00,0x00,0x3C,0x66,0x66,0x66,0x66,0x66,0x3C,0x00,0x00,0x00,0x00},/\*"o",79\*/

{0x00,0x00,0x00,0x00,0x00,0x7C,0x66,0x66,0x66,0x66,0x66,0x7C,0x60,0x60,0x60,0x00},/\*"p",80\*/

{0x00,0x00,0x00,0x00,0x00,0x3E,0x66,0x66,0x66,0x66,0x66,0x3E,0x06,0x06,0x06,0x00},/\*"q",81\*/

{0x00,0x00,0x00,0x00,0x00,0x66,0x6E,0x70,0x60,0x60,0x60,0x60,0x00,0x00,0x00,0x00},/\*"r",82\*/

{0x00,0x00,0x00,0x00,0x00,0x3E,0x60,0x60,0x3C,0x06,0x06,0x7C,0x00,0x00,0x00,0x00},/\*"s",83\*/

{0x00,0x00,0x00,0x30,0x30,0x7E,0x30,0x30,0x30,0x30,0x30,0x1E,0x00,0x00,0x00,0x00},/\*"t",84\*/

{0x00,0x00,0x00,0x00,0x00,0x66,0x66,0x66,0x66,0x66,0x66,0x3E,0x00,0x00,0x00,0x00},/\*"u",85\*/

{0x00,0x00,0x00,0x00,0x00,0x66,0x66,0x66,0x66,0x66,0x3C,0x18,0x00,0x00,0x00,0x00},/\*"v",86\*/

{0x00,0x00,0x00,0x00,0x00,0x63,0x6B,0x6B,0x6B,0x6B,0x36,0x36,0x00,0x00,0x00,0x00},/\*"w",87\*/

{0x00,0x00,0x00,0x00,0x00,0x66,0x66,0x3C,0x18,0x3C,0x66,0x66,0x00,0x00,0x00,0x00},/\*"x",88\*/

{0x00,0x00,0x00,0x00,0x00,0x66,0x66,0x66,0x66,0x66,0x66,0x3C,0x0C,0x18,0xF0,0x00},/\*"y",89\*/

{0x00,0x00,0x00,0x00,0x00,0x7E,0x06,0x0C,0x18,0x30,0x60,0x7E,0x00,0x00,0x00,0x00},/\*"z",90\*/

{0x00,0x00,0x00,0x0C,0x18,0x18,0x18,0x30,0x60,0x30,0x18,0x18,0x18,0x0C,0x00,0x00},/\*"{",91\*/

{0x00,0x00,0x00,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x00},/\*"|",92\*/

{0x00,0x00,0x00,0x30,0x18,0x18,0x18,0x0C,0x06,0x0C,0x18,0x18,0x18,0x30,0x00,0x00},/\*"}",93\*/

{0x00,0x00,0x00,0x71,0xDB,0x8E,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/\*"~",94\*/

};

//копирует по коду ASCII матрицу символа в буффер

void GetASCIICode(unsigned char\* pBuffer,unsigned char ASCII)

{

memcpy(pBuffer,AsciiLib[(ASCII - 32)] ,16);

}

//печать в буффер символа

void putbuf\_char(uint16\_t\* bufchar, uint8\_t ASCI,uint16\_t bkcolor, uint16\_t charcolor)//16x8=128

{

uint16\_t i, j;

uint8\_t buffer[16], tmp\_char;

GetASCIICode(buffer,ASCI); /\* get font data \*/

for( i=0; i<16; i++ )

{

tmp\_char = buffer[i];

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

{

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

bufchar[i\*8+j]=charcolor;

else

bufchar[i\*8+j]=bkcolor;

}

}

}

//печать строки на экран

void LCDprintstr(char\* str, int16\_t y,uint16\_t bkcolor, uint16\_t charcolor)//240/8=30 символов в строке

{

int len = strlen(str);

if(len > 30)

len=30;

const uint16\_t sizestr=30;

uint16\_t bstr[16\*8\*sizestr];//3840 точек \* 2 байта = 7 680 байт = 7,5 Кб

int i,nchr,irow,icol;

for(i=0;i<16\*8\*sizestr;i++)

bstr[i]=bkcolor;

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

{

if( (str[nchr] >= 0x20 ) && (str[nchr] <=0x7E) )

{

uint16\_t bufchar[16\*8];

putbuf\_char(bufchar, str[nchr],bkcolor,charcolor);

for(irow=0;irow<16;irow++)

{

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

{

bstr[(nchr\*8)+irow\*(sizestr\*8)+icol]=bufchar[irow\*8+icol];

}

}

}

}

lcd\_fill\_region(0, y, 8\*sizestr, y+16, bstr);

}

Файл LCD\_ILI9325.h:

#ifndef \_\_LCD\_ILI9325\_H

#define \_\_LCD\_ILI9325\_H

#include <stdint.h>

#define LCD\_WIDTH 240

#define LCD\_HEIGTH 320

void init\_lcd\_ili9325(void);

void lcd\_fill\_region(int16\_t x1, int16\_t y1, int16\_t x2, int16\_t y2, uint16\_t \*colors\_data);

void lcd\_fill\_color(int16\_t x1, int16\_t y1, int16\_t x2, int16\_t y2, uint16\_t color);

void lcd\_set\_pixel(int16\_t x, int16\_t y, uint16\_t color);

#define LCD\_RGB565\_CONVERT\_M(red, green, blue) (((red >> 3) << 11) | ((green >> 2) << 5) | (blue >> 3))

// Some color constants

#define LCD\_COLOR\_BLACK LCD\_RGB565\_CONVERT\_M(0x00, 0x00, 0x00)

#define LCD\_COLOR\_MAROON LCD\_RGB565\_CONVERT\_M(0x80, 0x00, 0x00)

#define LCD\_COLOR\_GREEN LCD\_RGB565\_CONVERT\_M(0x00, 0x80, 0x00)

#define LCD\_COLOR\_OLIVE LCD\_RGB565\_CONVERT\_M(0x80, 0x80, 0x00)

#define LCD\_COLOR\_NAVY LCD\_RGB565\_CONVERT\_M(0x00, 0x00, 0x80)

#define LCD\_COLOR\_PURPLE LCD\_RGB565\_CONVERT\_M(0x80, 0x00, 0x80)

#define LCD\_COLOR\_TEAL LCD\_RGB565\_CONVERT\_M(0x00, 0x80, 0x80)

#define LCD\_COLOR\_SILVER LCD\_RGB565\_CONVERT\_M(0xC0, 0xC0, 0xC0)

#define LCD\_COLOR\_GRAY LCD\_RGB565\_CONVERT\_M(0x80, 0x80, 0x80)

#define LCD\_COLOR\_RED LCD\_RGB565\_CONVERT\_M(0xFF, 0x00, 0x00)

#define LCD\_COLOR\_LIME LCD\_RGB565\_CONVERT\_M(0xFF, 0xFF, 0x00)

#define LCD\_COLOR\_BLUE LCD\_RGB565\_CONVERT\_M(0x00, 0x00, 0xFF)

#define LCD\_COLOR\_FUCHSIA LCD\_RGB565\_CONVERT\_M(0xFF, 0x00, 0xFF)

#define LCD\_COLOR\_AQUA LCD\_RGB565\_CONVERT\_M(0x00, 0xFF, 0xFF)

#define LCD\_COLOR\_WHITE LCD\_RGB565\_CONVERT\_M(0xFF, 0xFF, 0xFF)

typedef struct {

char cnamecolor[16];

uint16\_t uwcolor;

} Tcolor;

extern const Tcolor aEGAcolr[16];

#endif

Файл LCD\_ILI9325.c:

/\*LCD\_ILI9325.c\*/

#include "LCD\_ILI9325.h"

#include "main.h"

const Tcolor aEGAcolr[16]= {

{"BLACK", LCD\_RGB565\_CONVERT\_M(0x00, 0x00, 0x00)},// 0 - чёрный

{"BLUE", LCD\_RGB565\_CONVERT\_M(0x00, 0x00, 0x80)},// 1 - синий

{"GREEN", LCD\_RGB565\_CONVERT\_M(0x00, 0x80, 0x00)},// 2 - зелёный

{"CYAN", LCD\_RGB565\_CONVERT\_M(0x00, 0x80, 0x80)},// 3 - синезелёный TEAL

{"RED", LCD\_RGB565\_CONVERT\_M(0x80, 0x00, 0x00)},// 4 - красный

{"MAGENTA", LCD\_RGB565\_CONVERT\_M(0x80, 0x00, 0x80)},// 5 - пурпурный PURPLE

{"BROWN", LCD\_RGB565\_CONVERT\_M(0x80, 0x40, 0x00)},// 6 - коричневый MAROON

{"LIGHT GRAY", LCD\_RGB565\_CONVERT\_M(0x80, 0x80, 0x80)},// 7 - светло-серый SILVER

{"DARK GRAY", LCD\_RGB565\_CONVERT\_M(0x40, 0x40, 0x40)},// 8 - тёмно-серый

{"BRIGHT BLUE", LCD\_RGB565\_CONVERT\_M(0x40, 0x40, 0xFF)},// 9 - голубой

{"BRIGHT GREEN", LCD\_RGB565\_CONVERT\_M(0x40, 0xFF, 0x40)},// 10 - ярко-зелёный

{"BRIGHT CYAN", LCD\_RGB565\_CONVERT\_M(0x40, 0xFF, 0xFF)},// 11 - яркий сине-зелёный

{"BRIGHT RED", LCD\_RGB565\_CONVERT\_M(0xFF, 0x40, 0x40)},// 12 - ярко-красный

{"BRIGHT MAGENTA", LCD\_RGB565\_CONVERT\_M(0xFF, 0x40, 0xFF)},// 13 - ярко-пурпурный

{"BRIGHT YELLOW", LCD\_RGB565\_CONVERT\_M(0xFF, 0xFF, 0x40)},// 14 - жёлтый

{"BRIGHT WHITE", LCD\_RGB565\_CONVERT\_M(0xFF, 0xFF, 0xFF)},// 15 - ярко-белый

};

static void gpio\_lcd\_init(void);

static void board\_lcd\_reset(void);

static void bus\_lcd\_write\_reg(uint16\_t address, uint16\_t value);

static void bus\_lcd\_read\_reg(uint16\_t address, uint16\_t \*value);

static void lcd\_bus\_test(void);

/\*LCD pins

1 2 GND

LCD\_D0 PD0 3 4 PD1 LCD\_D1

LCD\_D2 PD2 5 6 PD3 LCD\_D3

LCD\_D4 PD4 7 8 PD5 LCD\_D5

LCD\_D6 PD6 9 10 PD7 LCD\_D7

LCD\_D8 PD8 11 12 PD9 LCD\_D9

LCD\_D10 PD10 13 14 PD11 LCD\_D11

LCD\_D12 PD12 15 16 PD13 LCD\_D13

LCD\_D14 PD14 17 18 PD15 LCD\_D15

LCD\_CS PB8 19 20 PB9 LCD\_RS

LCD\_WR PB10 21 22 PB11 LCD\_RD

LCD\_RESET PC11 23 24

BL\_VDD5V VDD 25 26 GND BL\_GND

BL\_PWM PC10 27 28 PB1 LCDTP\_IRQ

LCDTP\_CS PC13 29 30 PA5 LCDTP\_CLK

LCDTP\_DIN PA7 31 32 PA6 LCDTP\_DOUT

VDD 33 34 GND

\*/

#define LCD\_RESET\_Pin LL\_GPIO\_PIN\_11

#define LCD\_RESET\_GPIO\_Port GPIOC

#define LCD\_BL\_PWM\_Pin LL\_GPIO\_PIN\_10

#define LCD\_BL\_PWM\_GPIO\_Port GPIOC

#define LCD\_CONTROLBUS\_PORT GPIOB

#define LCD\_DATABUS\_PORT GPIOD

// capture bus LCD: reset #CS

#define LCD\_CAPTURE\_BUS LCD\_CONTROLBUS\_PORT->BRR = LL\_GPIO\_PIN\_8

// release bus LCD: set #CS

#define LCD\_RELEASE\_BUS LCD\_CONTROLBUS\_PORT->BSRR = LL\_GPIO\_PIN\_8

// switch to index mode

#define LCD\_CLEAR\_RS LCD\_CONTROLBUS\_PORT->BRR = LL\_GPIO\_PIN\_9

// switch to data mode

#define LCD\_SET\_RS LCD\_CONTROLBUS\_PORT->BSRR = LL\_GPIO\_PIN\_9

// start write signal

#define LCD\_CLEAR\_WR LCD\_CONTROLBUS\_PORT->BRR = LL\_GPIO\_PIN\_10

// stop write signal

#define LCD\_SET\_WR LCD\_CONTROLBUS\_PORT->BSRR = LL\_GPIO\_PIN\_10

// start read signal

#define LCD\_CLEAR\_RD LCD\_CONTROLBUS\_PORT->BRR = LL\_GPIO\_PIN\_11

// stop read signal

#define LCD\_SET\_RD LCD\_CONTROLBUS\_PORT->BSRR = LL\_GPIO\_PIN\_11

// helper register addresses

#define LCD\_REGISTER\_GRAM\_X\_ADDR 0x0020

#define LCD\_REGISTER\_GRAM\_Y\_ADDR 0x0021

#define LCD\_REGISTER\_GRAM\_WRITE\_DATA 0x0022

#define LCD\_REGISTER\_GRAM\_X\_ADDR\_START 0x0050

#define LCD\_REGISTER\_GRAM\_X\_ADDR\_END 0x0051

#define LCD\_REGISTER\_GRAM\_Y\_ADDR\_START 0x0052

#define LCD\_REGISTER\_GRAM\_Y\_ADDR\_END 0x0053

static void board\_lcd\_reset()

{

// reset LCD

LL\_GPIO\_ResetOutputPin(LCD\_RESET\_GPIO\_Port, LCD\_RESET\_Pin);

LL\_mDelay(50);//50ms

LL\_GPIO\_SetOutputPin(LCD\_RESET\_GPIO\_Port, LCD\_RESET\_Pin);

}

static void bus\_lcd\_write\_reg(uint16\_t address, uint16\_t value)

{

// write register address

LCD\_CLEAR\_RS; //устанавливаем низкий уровень на линии LCD\_RS, т.е. на шине данных будет выставлен адресс управляющего регистра

LCD\_DATABUS\_PORT->ODR = address; //устанавливаем данные на шину

LCD\_CLEAR\_WR; //устанавливаем низкий уровень на линии LCD\_WR, т.е. данные на шине записываются в контроллёр, LCD\_RD - высокий уровень

\_\_NOP();

LCD\_SET\_WR; //устанавливаем высокий уровень на линии LCD\_WR, по изменению уровня контроллер считал данные

LCD\_SET\_RS; //устанавливаем высокий уровень на линии LCD\_RS, т.е. на шине данных будет выставлены данные

// write register data

LCD\_DATABUS\_PORT->ODR = value;

LCD\_CLEAR\_WR;//устанавливаем низкий уровень на линии LCD\_WR, т.е. данные на шине записываются в контроллёр, LCD\_RD - высокий уровень

\_\_NOP();

LCD\_SET\_WR;

}

static void board\_lcd\_write\_words(uint16\_t address, uint16\_t \*data, size\_t size)

{

uint16\_t\* data\_end;

// write register address

LCD\_CLEAR\_RS;

LCD\_DATABUS\_PORT->ODR = address;

LCD\_CLEAR\_WR;

\_\_NOP();

LCD\_SET\_WR;

LCD\_SET\_RS;

// write register data

data\_end = data + size;

while (data != data\_end) {

LCD\_DATABUS\_PORT->ODR = \*data;

LCD\_CLEAR\_WR;

\_\_NOP();

LCD\_SET\_WR;

data++;

}

}

static void bus\_lcd\_read\_reg(uint16\_t address, uint16\_t \*value)

{

uint32\_t data\_port\_mode;

volatile uint32\_t i;

// write register address

LCD\_CLEAR\_RS;

LCD\_DATABUS\_PORT->ODR = address;

LCD\_CLEAR\_WR;

\_\_NOP();

LCD\_SET\_WR;

LCD\_SET\_RS;

// read register

// reconfigure data port to read mode

data\_port\_mode = LCD\_DATABUS\_PORT->MODER;

LCD\_DATABUS\_PORT->MODER = LL\_GPIO\_MODE\_INPUT;

// read data

LCD\_CLEAR\_RD;

for(i=0;i<3;i++) // ждём 0,2 мксек

\_\_NOP();

\*value = LCD\_DATABUS\_PORT->IDR;

LCD\_SET\_RD;

// restore gpio mode

LCD\_DATABUS\_PORT->MODER = data\_port\_mode;

}

static void lcd\_bus\_test(void)

{

uint16\_t i;

uint16\_t act\_val = 0;

for (i = 0xEE; i < 0xF1; i++) {

bus\_lcd\_write\_reg(0x0020, i);

bus\_lcd\_read\_reg(0x0020, &act\_val);

if (act\_val != i) {

/\* Initialization Error \*/

while (1){}

}

}

bus\_lcd\_write\_reg(0x0020, 0x00);

}

static void gpio\_lcd\_init(void)

{

LL\_GPIO\_InitTypeDef gpio\_initstruct = {0};

/\* GPIO Ports Clock Enable \*/

LL\_AHB1\_GRP1\_EnableClock(LL\_AHB1\_GRP1\_PERIPH\_GPIOD);

LL\_AHB1\_GRP1\_EnableClock(LL\_AHB1\_GRP1\_PERIPH\_GPIOB);

LL\_AHB1\_GRP1\_EnableClock(LL\_AHB1\_GRP1\_PERIPH\_GPIOC);

/\*Configure GPIO pin lcd bus control

PB.10(WR), PB.11(RD) , PB.08(CS), PB.09(RS)\*/

LL\_GPIO\_SetOutputPin(LCD\_CONTROLBUS\_PORT, LL\_GPIO\_PIN\_8 | LL\_GPIO\_PIN\_9

|LL\_GPIO\_PIN\_10 | LL\_GPIO\_PIN\_11);

gpio\_initstruct.Pin = LL\_GPIO\_PIN\_8 | LL\_GPIO\_PIN\_9

|LL\_GPIO\_PIN\_10 | LL\_GPIO\_PIN\_11;

gpio\_initstruct.Mode = LL\_GPIO\_MODE\_OUTPUT;

gpio\_initstruct.OutputType = LL\_GPIO\_OUTPUT\_PUSHPULL;

gpio\_initstruct.Pull = LL\_GPIO\_PULL\_NO;

gpio\_initstruct.Speed = LL\_GPIO\_SPEED\_FREQ\_HIGH;

if (LL\_GPIO\_Init(LCD\_CONTROLBUS\_PORT, &gpio\_initstruct) != SUCCESS)

{

/\* Initialization Error \*/

while (1){}

}

/\*Configure GPIO pin data bus

PD.00(D0), PD.01(D1), PD.02(D2), PD.03(D3),

PD.04(D4), PD.05(D5), PD.06(D6), PD.07(D7),

PD.08(D8), PD.09(D9), PD.10(D10), PD.11(D11),

PD.12(D12), PD.13(D13), PD.14(D14), PD.15(D15)\*/

LL\_GPIO\_SetOutputPin(LCD\_DATABUS\_PORT, LL\_GPIO\_PIN\_ALL);

gpio\_initstruct.Pin = LL\_GPIO\_PIN\_ALL;

gpio\_initstruct.Mode = LL\_GPIO\_MODE\_OUTPUT;

gpio\_initstruct.OutputType = LL\_GPIO\_OUTPUT\_PUSHPULL;

gpio\_initstruct.Pull = LL\_GPIO\_PULL\_NO;

gpio\_initstruct.Speed = LL\_GPIO\_SPEED\_FREQ\_HIGH;

if (LL\_GPIO\_Init(LCD\_DATABUS\_PORT, &gpio\_initstruct) != SUCCESS)

{

/\* Initialization Error \*/

while (1){}

}

/\*Configure GPIO pin reset, backlight control PWM: PC.11(RST) PC.10(BL\_PWM) \*/

gpio\_initstruct.Pin = LL\_GPIO\_PIN\_11 | LL\_GPIO\_PIN\_10;

gpio\_initstruct.Mode = LL\_GPIO\_MODE\_OUTPUT;

gpio\_initstruct.OutputType = LL\_GPIO\_OUTPUT\_OPENDRAIN;

gpio\_initstruct.Pull = LL\_GPIO\_PULL\_UP;

gpio\_initstruct.Speed = LL\_GPIO\_SPEED\_FREQ\_LOW;

if (LL\_GPIO\_Init(GPIOC, &gpio\_initstruct) != SUCCESS)

{

/\* Initialization Error \*/

while (1){}

}

LL\_GPIO\_SetOutputPin(LCD\_BL\_PWM\_GPIO\_Port,LCD\_BL\_PWM\_Pin);//Максимальная яркость

}

void init\_lcd\_ili9325(void)

{

uint16\_t device\_code = 0;

gpio\_lcd\_init();

// reset LCD

board\_lcd\_reset();

// read device code

LCD\_CAPTURE\_BUS;

bus\_lcd\_read\_reg(0x0000, &device\_code);

if (device\_code == 0x9325 || device\_code == 0x9328) {

// test write operation

lcd\_bus\_test();

bus\_lcd\_write\_reg(0xE5, 0x78F0); /\* set SRAM internal timing \*/

bus\_lcd\_write\_reg(0x01, 0x0100); /\* set Driver Output Control \*/

bus\_lcd\_write\_reg(0x02, 0x0700); /\* set 1 line inversion \*/

bus\_lcd\_write\_reg(0x03, 0x1030); /\* set GRAM write direction and BGR=1 \*/

bus\_lcd\_write\_reg(0x04, 0x0000); /\* Resize register \*/

bus\_lcd\_write\_reg(0x08, 0x0207); /\* set the back porch and front porch \*/

bus\_lcd\_write\_reg(0x09, 0x0000); /\* set non-display area refresh cycle ISC[3:0] \*/

bus\_lcd\_write\_reg(0x0A, 0x0000); /\* FMARK function \*/

bus\_lcd\_write\_reg(0x0C, 0x0000); /\* RGB interface setting \*/

bus\_lcd\_write\_reg(0x0D, 0x0000); /\* Frame marker Position \*/

bus\_lcd\_write\_reg(0x0F, 0x0000); /\* RGB interface polarity \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*Power On sequence \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bus\_lcd\_write\_reg(0x10, 0x0000); /\* SAP, BT[3:0], AP, DSTB, SLP, STB \*/

bus\_lcd\_write\_reg(0x11, 0x0007); /\* DC1[2:0], DC0[2:0], VC[2:0] \*/

bus\_lcd\_write\_reg(0x12, 0x0000); /\* VREG1OUT voltage \*/

bus\_lcd\_write\_reg(0x13, 0x0000); /\* VDV[4:0] for VCOM amplitude \*/

bus\_lcd\_write\_reg(0x07, 0x0001);

LL\_mDelay(50);/\* Delay50ms \*/

/\* Dis-charge capacitor power voltage \*/

bus\_lcd\_write\_reg(0x10, 0x1090); /\* SAP, BT[3:0], AP, DSTB, SLP, STB \*/

bus\_lcd\_write\_reg(0x11, 0x0227); /\* Set DC1[2:0], DC0[2:0], VC[2:0] \*/

LL\_mDelay(50); /\* Delay 50ms \*/

bus\_lcd\_write\_reg(0x12, 0x001F);

LL\_mDelay(50); /\* Delay 50ms \*/

bus\_lcd\_write\_reg(0x13, 0x1500); /\* VDV[4:0] for VCOM amplitude \*/

bus\_lcd\_write\_reg(0x29, 0x0027); /\* 04 VCM[5:0] for VCOMH \*/

bus\_lcd\_write\_reg(0x2B, 0x000D); /\* Set Frame Rate \*/

LL\_mDelay(50); /\* Delay 50ms \*/

bus\_lcd\_write\_reg(0x20, 0x0000); /\* GRAM horizontal Address \*/

bus\_lcd\_write\_reg(0x21, 0x0000); /\* GRAM Vertical Address \*/

/\* ----------- Adjust the Gamma Curve ---------- \*/

bus\_lcd\_write\_reg(0x30, 0x0000);

bus\_lcd\_write\_reg(0x31, 0x0707);

bus\_lcd\_write\_reg(0x32, 0x0307);

bus\_lcd\_write\_reg(0x35, 0x0200);

bus\_lcd\_write\_reg(0x36, 0x0008);

bus\_lcd\_write\_reg(0x37, 0x0004);

bus\_lcd\_write\_reg(0x38, 0x0000);

bus\_lcd\_write\_reg(0x39, 0x0707);

bus\_lcd\_write\_reg(0x3C, 0x0002);

bus\_lcd\_write\_reg(0x3D, 0x1D04);

/\* ------------------ Set GRAM area --------------- \*/

bus\_lcd\_write\_reg(0x50, 0x0000); /\* Horizontal GRAM Start Address \*/

bus\_lcd\_write\_reg(0x51, 0x00EF); /\* Horizontal GRAM End Address \*/

bus\_lcd\_write\_reg(0x52, 0x0000); /\* Vertical GRAM Start Address \*/

bus\_lcd\_write\_reg(0x53, 0x013F); /\* Vertical GRAM Start Address \*/

bus\_lcd\_write\_reg(0x60, 0xA700); /\* Gate Scan Line \*/

bus\_lcd\_write\_reg(0x61, 0x0001); /\* NDL,VLE, REV \*/

bus\_lcd\_write\_reg(0x6A, 0x0000); /\* set scrolling line \*/

/\* -------------- Partial Display Control --------- \*/

bus\_lcd\_write\_reg(0x80, 0x0000);

bus\_lcd\_write\_reg(0x81, 0x0000);

bus\_lcd\_write\_reg(0x82, 0x0000);

bus\_lcd\_write\_reg(0x83, 0x0000);

bus\_lcd\_write\_reg(0x84, 0x0000);

bus\_lcd\_write\_reg(0x85, 0x0000);

/\* -------------- Panel Control ------------------- \*/

bus\_lcd\_write\_reg(0x90, 0x0010);

bus\_lcd\_write\_reg(0x92, 0x0600);

bus\_lcd\_write\_reg(0x07, 0x0133); /\* 262K color and display ON \*/

}

LCD\_RELEASE\_BUS;

lcd\_fill\_color(0, 0, LCD\_WIDTH,LCD\_HEIGTH, LCD\_COLOR\_BLACK);

}

void lcd\_fill\_region(int16\_t x1, int16\_t y1, int16\_t x2, int16\_t y2, uint16\_t \*colors\_data)

{

size\_t total\_pixes;

// set window

if (x1 > x2) {

int16\_t x\_tmp = x1;

x1 = x2;

x2 = x\_tmp;

}

if (y1 > y2) {

int16\_t y\_tmp = y1;

y1 = y2;

y2 = y\_tmp;

}

LCD\_CAPTURE\_BUS;

// set gram coordinates

bus\_lcd\_write\_reg(LCD\_REGISTER\_GRAM\_X\_ADDR, x1);

bus\_lcd\_write\_reg(LCD\_REGISTER\_GRAM\_Y\_ADDR, y1);

bus\_lcd\_write\_reg(LCD\_REGISTER\_GRAM\_X\_ADDR\_START, x1);

bus\_lcd\_write\_reg(LCD\_REGISTER\_GRAM\_X\_ADDR\_END, x2 - 1);

bus\_lcd\_write\_reg(LCD\_REGISTER\_GRAM\_Y\_ADDR\_START, y1);

bus\_lcd\_write\_reg(LCD\_REGISTER\_GRAM\_Y\_ADDR\_END, y2 - 1);

// write pixels

total\_pixes = (x2 - x1) \* (y2 - y1);

board\_lcd\_write\_words(LCD\_REGISTER\_GRAM\_WRITE\_DATA, colors\_data, total\_pixes);

LCD\_RELEASE\_BUS;

}

void lcd\_fill\_color(int16\_t x1, int16\_t y1, int16\_t x2, int16\_t y2, uint16\_t color)

{

size\_t i;

size\_t total\_pixes;

const size\_t max\_buff\_size = 128;

size\_t buff\_size;

uint16\_t buff[max\_buff\_size];

// set window

if (x1 > x2) {

int16\_t x\_tmp = x1;

x1 = x2;

x2 = x\_tmp;

}

if (y1 > y2) {

int16\_t y\_tmp = y1;

y1 = y2;

y2 = y\_tmp;

}

// set gram coordinates

LCD\_CAPTURE\_BUS;

bus\_lcd\_write\_reg(LCD\_REGISTER\_GRAM\_X\_ADDR, x1);

bus\_lcd\_write\_reg(LCD\_REGISTER\_GRAM\_Y\_ADDR, y1);

bus\_lcd\_write\_reg(LCD\_REGISTER\_GRAM\_X\_ADDR\_START, x1);

bus\_lcd\_write\_reg(LCD\_REGISTER\_GRAM\_X\_ADDR\_END, x2 - 1);

bus\_lcd\_write\_reg(LCD\_REGISTER\_GRAM\_Y\_ADDR\_START, y1);

bus\_lcd\_write\_reg(LCD\_REGISTER\_GRAM\_Y\_ADDR\_END, y2 - 1);

// prepare buffer

total\_pixes = (x2 - x1) \* (y2 - y1);

buff\_size = total\_pixes < max\_buff\_size ? total\_pixes : max\_buff\_size;

for (i = 0; i < buff\_size; i++) {

buff[i] = color;

}

i = total\_pixes;

while (i > 0) {

size\_t pixels\_to\_update = i >= buff\_size ? buff\_size : i;

// write pixels

board\_lcd\_write\_words(LCD\_REGISTER\_GRAM\_WRITE\_DATA, buff, pixels\_to\_update);

i -= pixels\_to\_update;

}

LCD\_RELEASE\_BUS;

}

void lcd\_set\_pixel(int16\_t x, int16\_t y, uint16\_t color)

{

LCD\_CAPTURE\_BUS;

// set gram coordinates

bus\_lcd\_write\_reg(LCD\_REGISTER\_GRAM\_X\_ADDR, x);

bus\_lcd\_write\_reg(LCD\_REGISTER\_GRAM\_Y\_ADDR, y);

bus\_lcd\_write\_reg(LCD\_REGISTER\_GRAM\_X\_ADDR\_START, x);

bus\_lcd\_write\_reg(LCD\_REGISTER\_GRAM\_X\_ADDR\_END, x);

bus\_lcd\_write\_reg(LCD\_REGISTER\_GRAM\_Y\_ADDR\_START, y);

bus\_lcd\_write\_reg(LCD\_REGISTER\_GRAM\_Y\_ADDR\_END, y);

// write pixel

bus\_lcd\_write\_reg(LCD\_REGISTER\_GRAM\_WRITE\_DATA, color);

LCD\_RELEASE\_BUS;

}