Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Школа Инженерная школа неразрушающего контроля и безопасности

Направление подготовки (специальность) Электроника и наноэлектроника

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту/работе

по дисциплине Микроконтроллерные устройства

(Название дисциплины)

на тему «Разработка генератора сигналов»

Выполнил студент гр. 1А02 _____ Кушков Е. О.

(Номер группы) (Подпись) (Ф.И.О.)

Дата сдачи пояснительной записки преподавателю ____ _____2023 г.

Руководитель старший преподаватель ОЭИ ИШНКБ Мусоров И.С.

(Ученая степень, ученое звание, должность) (Ф.И.О.)

_____ _____

(Оценка руководителя) (Подпись)

_____ _____ 2023 г.

(Дата проверки)

Курсовой проект/работу студент Кушков Е. О. выполнил и защитил

(Ф.И.О.)

с оценкой _____.

Члены комиссии: _____

_____

_____

_____ _____ 2023 г.

(дата защиты)

Томск 2023 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Отделение электронной инженерии
И.о. заведующего кафедрой - руководителя отделения на правах кафедры _____
_____ П.Ф. Баранов
(подпись, дата)

ЗАДАНИЕ
на выполнение курсового проекта
по дисциплине «Основы микропроцессорной техники»

Студенту гр. 1А02

1 Тема курсового проекта: Разработка генератора сигналов
2 Срок сдачи студентом готовой работы 30.05.2023
3 Исходные данные к работе
Разработать генератор электрических сигналов разной формы.
Напряжение питания: 5 В.
Дисплей: MT12864J.
ЦАП: DAC7512N/3K
Управление: перемещение по пунктам меню с помощью кнопок матричной клавиатуры.
На дисплее должно быть создано меню, в котором возможно выбрать следующие пункты: а) параметры сигнала, б) состояние выхода. В пункте меню – параметры сигнала должна быть возможность выбрать форму сигнала: прямоугольный, треугольный, синусоидальный. Для каждого выбранного сигнала должны отображаться доступные для настройки параметры.
4 Содержание пояснительной записки
Введение
Выбор и обоснование структурной схемы
Описание и обоснование выбранных компонентов
Разработка принципиальной схемы
Алгоритм программы
Код программы
Экспериментальные результаты
Заключение
Принципиальная схема устройства (в соответствии с ЕСКД)
Перечень элементов (в соответствии с ЕСКД)
5 Дата выдачи задания на выполнение курсового проекта _____

Руководитель
_____ И.С. Мусоров

Задание принял к исполнению
_____ Е.О. Кушков

Оглавление

Введение 4
1. Выбор и обоснование структурной схемы 5
2. Описание и обоснование выбранных компонентов 6
3. Разработка принципиальной схемы 8
4. Алгоритм и код программы 9
5. Экспериментальные результаты 10
Заключение 13
Список использованной литературы 14
Приложение А. Принципиальная схема 15
Приложение Б. Перечень элементов 16
Приложение В. Код программы 17

Введение

Целью курсовой работы является разработка генератора электрических сигналов разной формы. Устройство основано на микроконтроллере STM8S207RB [1][2].

Исходя из поставленной цели, можно выделить следующие задачи:

1. Изучить принцип работы ЦАП.
2. Изучить передачу данных по протоколу SPI.
3. Разработать структурную схему устройства.
4. Разработать меню управления генератором сигналов с вводом данных с помощью матричной клавиатуры.
5. Осуществить вывод сигнала заданной формы с помощью 12-и разрядного ЦАП.
6. Объединить управление ЦАПом с меню, получив непосредственно готовую программу для генератора сигналов разной формы, с вводом данных через матричную клавиатуру.
7. Собрать схему генератора сигналов.
8. Протестировать корректность работы устройства.
9. Создать принципиальную схему устройства.
10. Создать перечень элементов генератора сигналов.

Результатом работы является готовое устройство, способной генерировать электрический сигнал трех форм: синус, треугольник и меандр. У каждой формы сигнала есть настройка параметров. Навигация по меню и настройка осуществляется кнопками матричной клавиатуры.

1. Выбор и обоснование структурной схемы

Структурная схема генератора сигналов содержит микроконтроллер, ЦАП, матричную клавиатуру и дисплей. На рисунке 1 представлена структурная схема генератора сигналов.

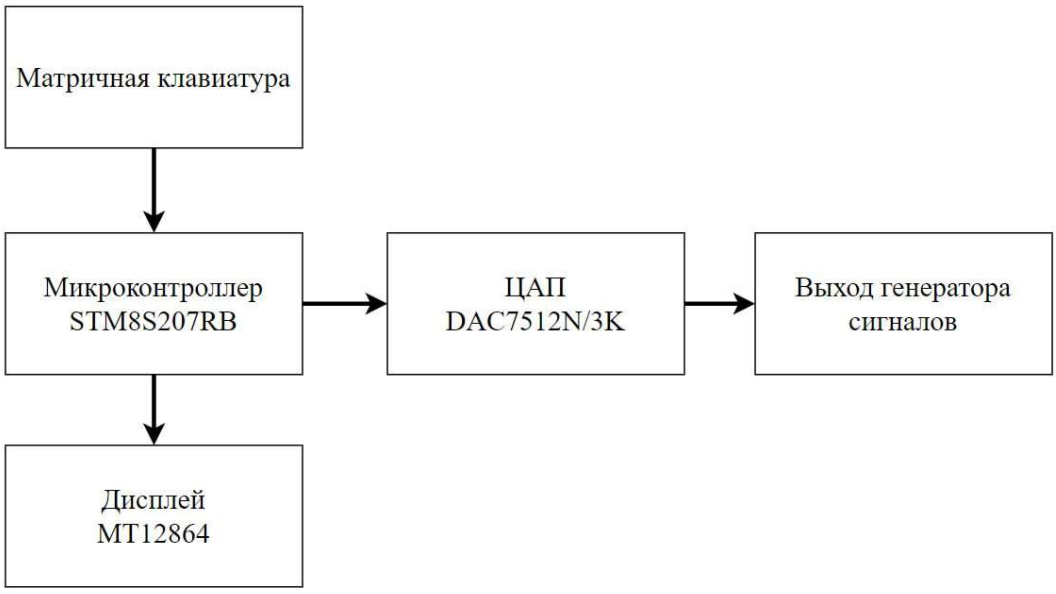

Рисунок 1 – Структурная схема генератора сигналов

Микроконтроллер STM8S207RB является управляющим элементом, задача которого заключается в генерации цифрового сигнала, отправке данных сигнала по SPI ЦАП, отрисовке меню на дисплее и опросе кнопок матричной клавиатуры.

Дисплей MT12864J [3] служит для отрисовки меню, выводе данных о текущих параметрах сигнала и состоянии генератора.

Взаимодействие пользователя с устройством осуществляется путем навигации по меню, установке необходимых значений параметров сигнала с помощью матричной клавиатуры.

Основой генератора сигналов служит 12-и разрядный ЦАП DAC7512N/3K [4], способный устанавливать 4096 уровней напряжения на выходе.

2. Описание и обоснование выбранных компонентов

Разработка генератора электрических сигналов разной формы велась на отладочной плате с контроллером STM8S207RB. Отладочная плата содержит контроллер с необходимой обвязкой, программатор ST-LINK, коннекторы для подключения дисплея MT12864J, матричную клавиатуру, 12-и битный ЦАП. На рисунке 2 представлена отладочная плата с микроконтроллером STM8S207RB.

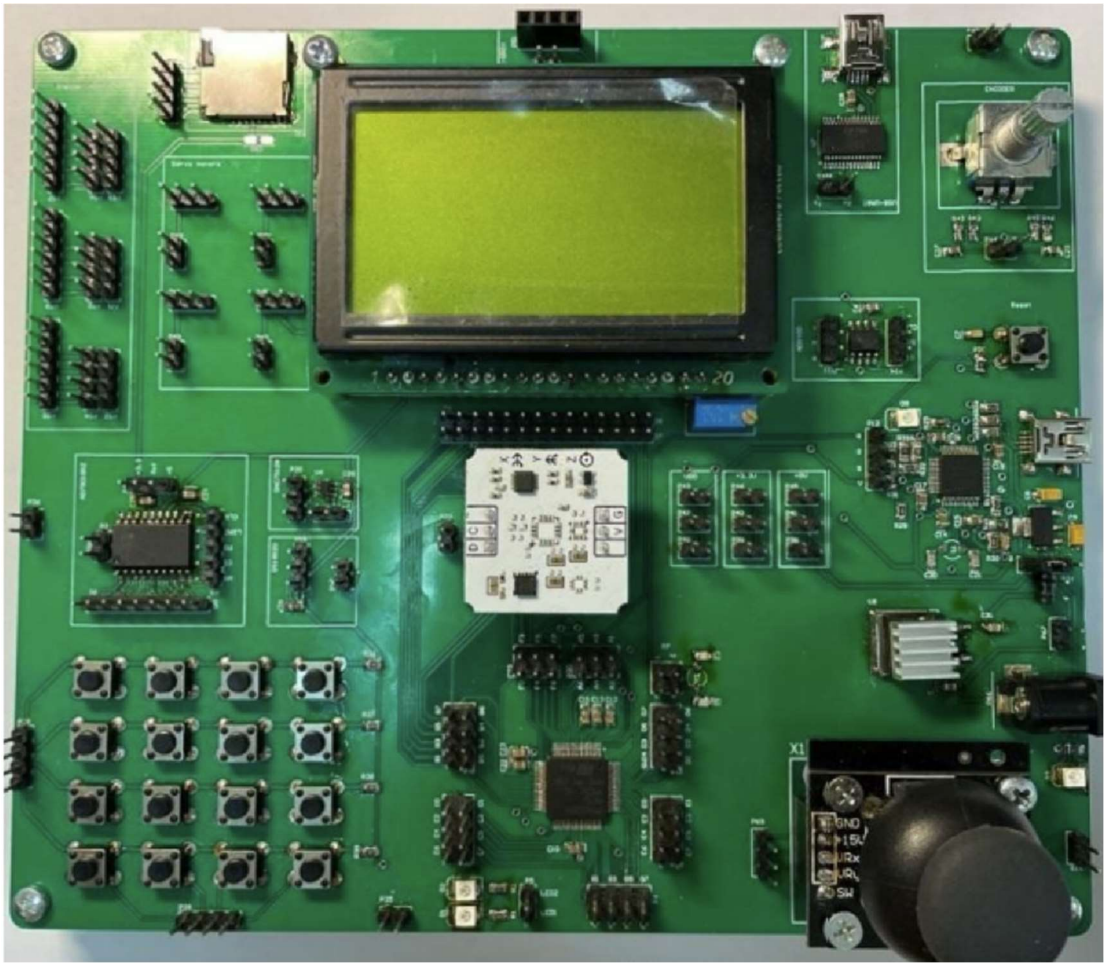

Рисунок 2 – Отладочная плата с микроконтроллером STM8S207RB

Микроконтроллер STM8S207RB от компании STMicroelectronics – это 8-битный микроконтроллер, базово работающий на частоте 2 МГц, однако есть возможность разогнать частоту вплоть до 16 МГц. Он имеет встроенную память FLASH в размере 128 кбайт, 6 кбайт оперативной памяти (RAM) и 2 кбайт данных EEPROM. В микроконтроллере также есть множество

периферийных устройств, таких как аппаратный умножитель, таймеры общего назначения, программируемый интерфейс последовательной связи (UART), интерфейсы SPI и I2C, ADC и т.д.

Матричная клавиатура представляет собой четыре ряда по четыре кнопки, одним контактом кнопки подтянуты через резистор к питанию, на другой контакт каждой кнопки подключен к выводам микроконтроллера. При нажатии кнопки на выводах микроконтроллера меняется уровень напряжения. Таким образом можно понять какая кнопка была нажата.

Дисплей MT12864J является графическим дисплеем с разрешением 128x64 пикселя. Он имеет два кристалла по 64x64 пикселя, что позволяет более удобно отрисовывать меню.

DAC7512N/3K является 12-и битным ЦАП с последовательным интерфейсом данных SPI. Ввиду большого количества разрядов, данный ЦАП позволяет очень точно выставить уровень напряжения на выходе (опорное напряжение 5 В разбивается на 4096 уровней, что составляет шаг в 1,22 мВ), однако теряя при этом в скорости.

3. Разработка принципиальной схемы

Принципиальная схема приведена в приложении А. Матричная клавиатура подключалась к выводам порта PE. Выходы, отвечающие за горизонтальные ряды кнопок подключены к выводам PE0 – PE3. Выходы матричной клавиатуры, отвечающие за вертикальное определение кнопок, подключены к выводам PE4 – PE7. Изначально при нажатии кнопки начинается обработка внешнего прерывания, определяется горизонтальное положение нажатой кнопки, затем проходом по вертикалям определяется вертикальное положение нажатой кнопки. Таким образом, при нажатии кнопки всегда известно ее местоположение.

Принципиальная схема включает необходимую обвязку микроконтроллера в виде конденсаторов по питанию, кнопки перезагрузки и подключения выводов программатора, для загрузки прошивки на плату.

Дисплей подключается к выводам портов PB и PF. Изменяя положение потенциометра R6 изменяется контрастность дисплея.

ЦАП подключается к выводам, имеющим альтернативную функцию SPI. Так, вывод SCLK ЦАП подключается к PC5, а вывод данных к MOSI PC6. Вывод CS подключается к любому выводу порта PA.

Перечень элементов приведен в приложении Б.

4. Алгоритм и код программы

На рисунке 3 представлен основной алгоритм программы. При нажатии кнопок начинается обработка внешнего прерывания. Если была нажата кнопка, изменяющая состояние генератора в СТАРТ, то начинается вывод на ЦАП сигнала с заранее выбранными параметрами. Если состояние генератора СТОП и нажата кнопка изменения положения курсора, то смотрится состояние курсора (выбор параметров или навигация) и в зависимости от этого перемещается курсор, или изменяется выбранный параметр. Ввод числовых показателей осуществляется при состоянии генератора СТОП, положении курсора на нужной строчке меню, и состоянии курсора выбор параметров. При нажатии на первые шесть кнопок изменяется соответствующий разряд. Седьмая кнопка обнуляет параметр.

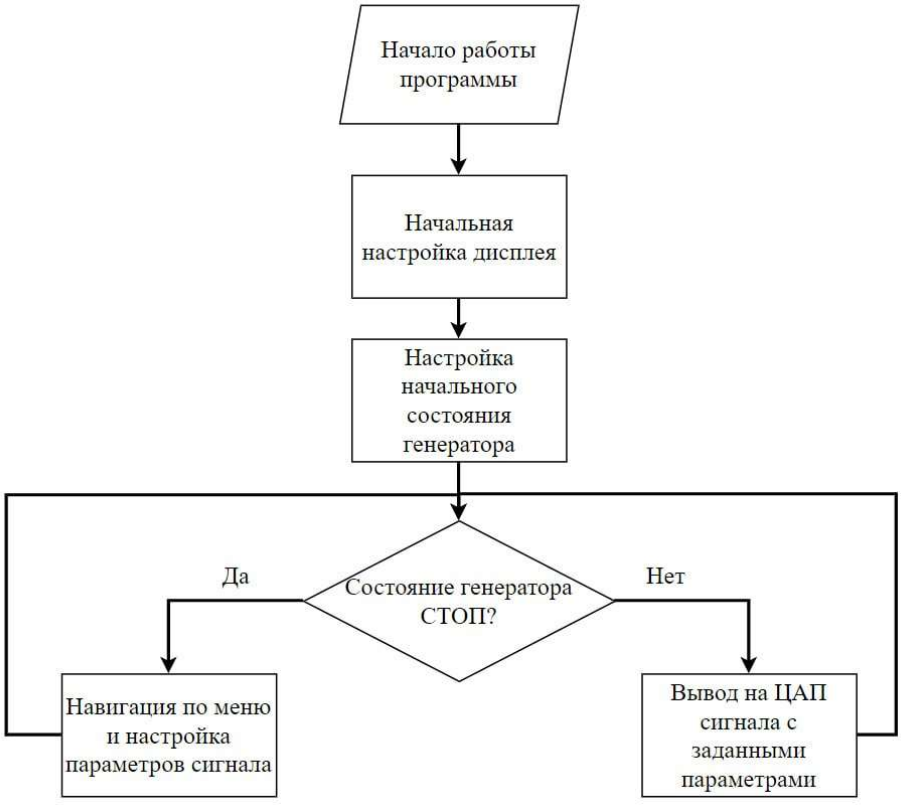

Рисунок 3 – Алгоритм работы программы

Код программы с комментариями приведен в приложении В.

5. Экспериментальные результаты

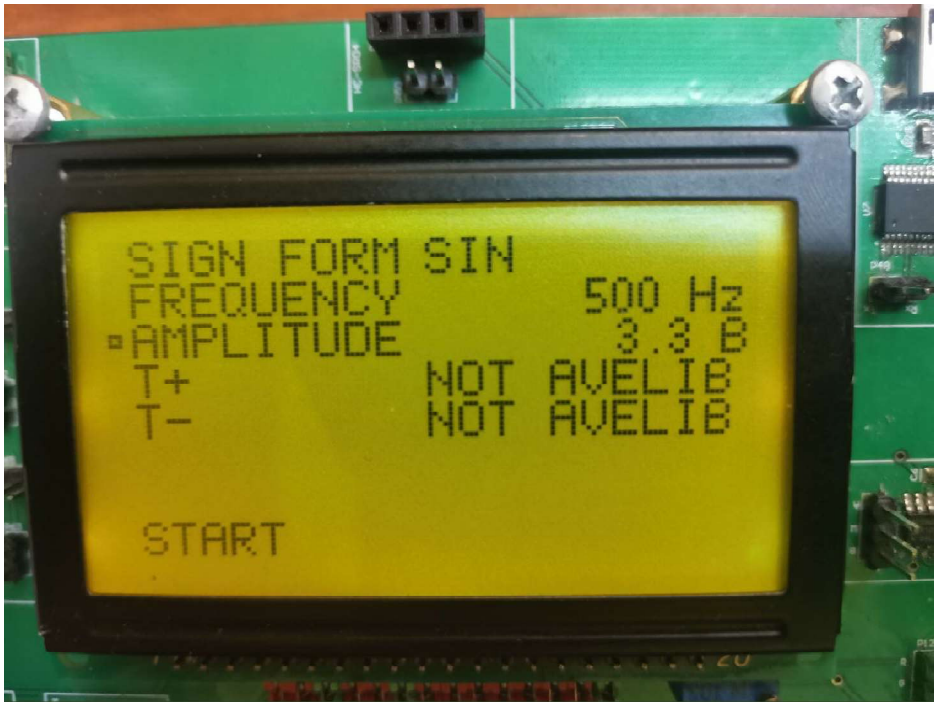

Рисунок 4 – Меню с выбранным сигналом синус 500 Гц 3,3 В

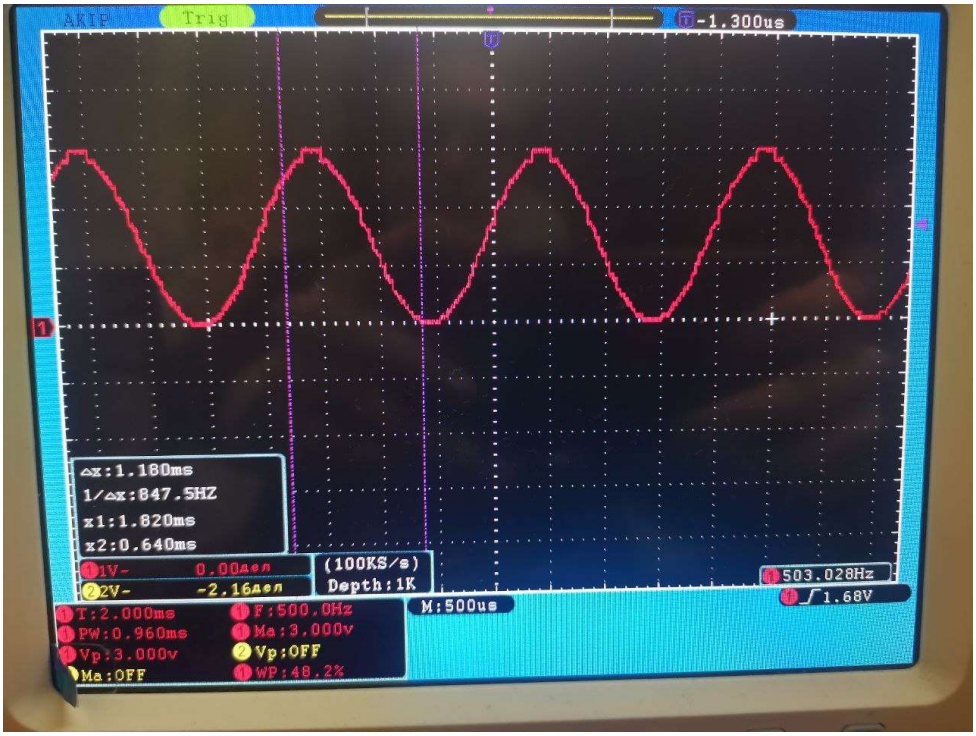

Рисунок 5 – Сигнал на выходе ЦАП синус 500 Гц 3,3 В

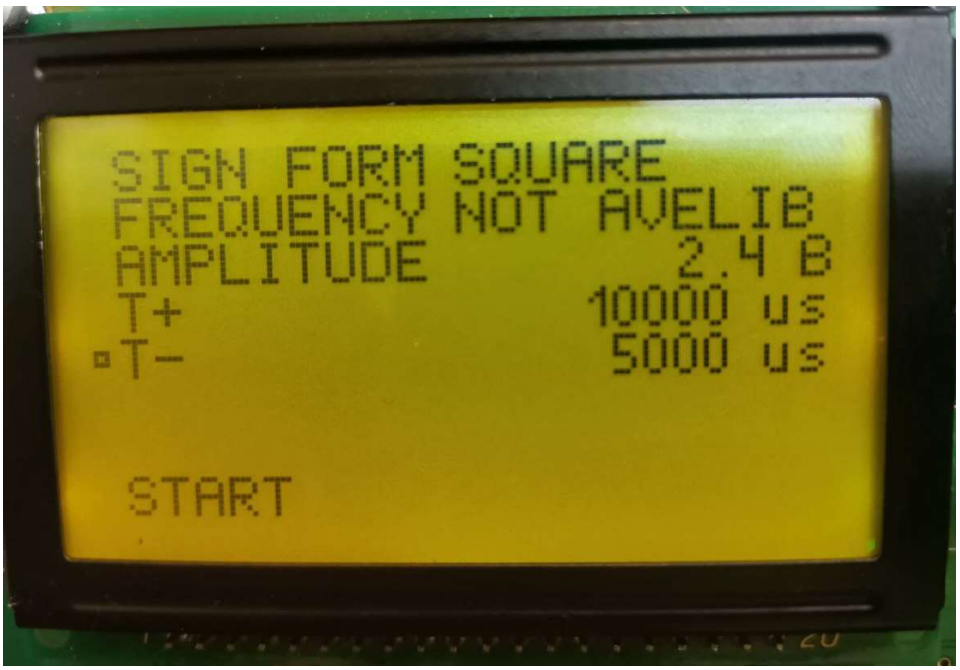

Рисунок 6 – Меню с выбранным сигналом меандр время импульса 10 мс, время паузы 5 мс 2,4 В

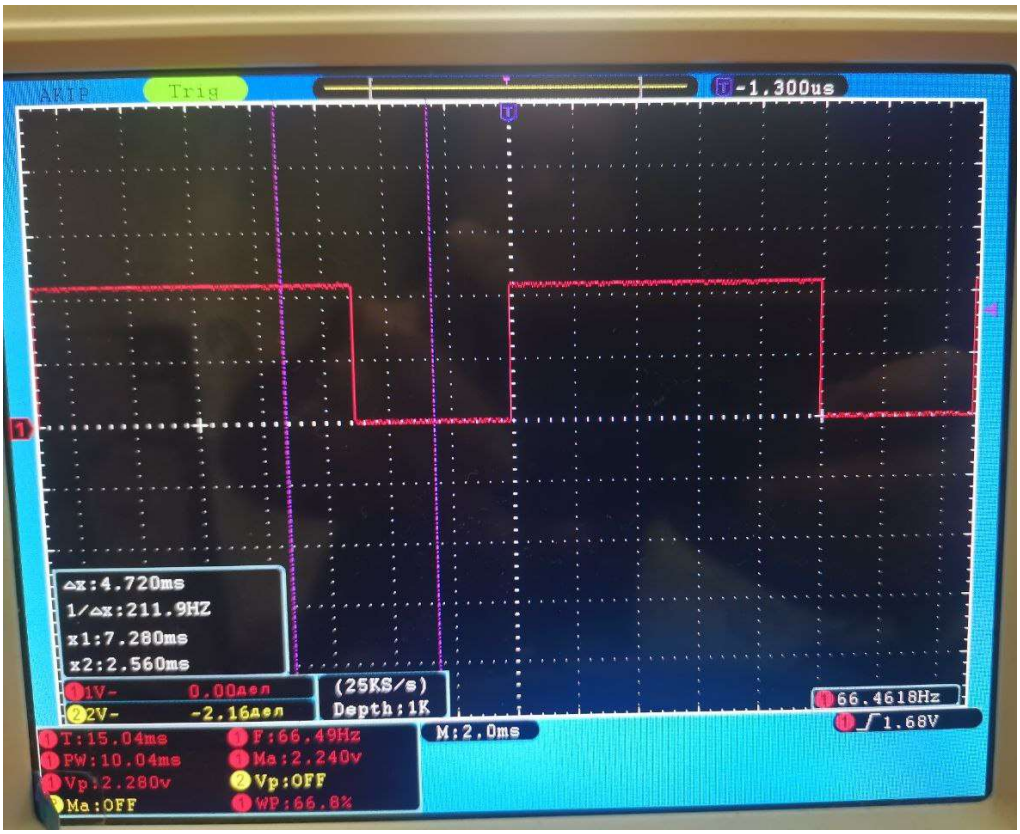

Рисунок 7 – Сигнал на выходе ЦАП меандр время импульса 10 мс, время паузы 5 мс 2,4 В

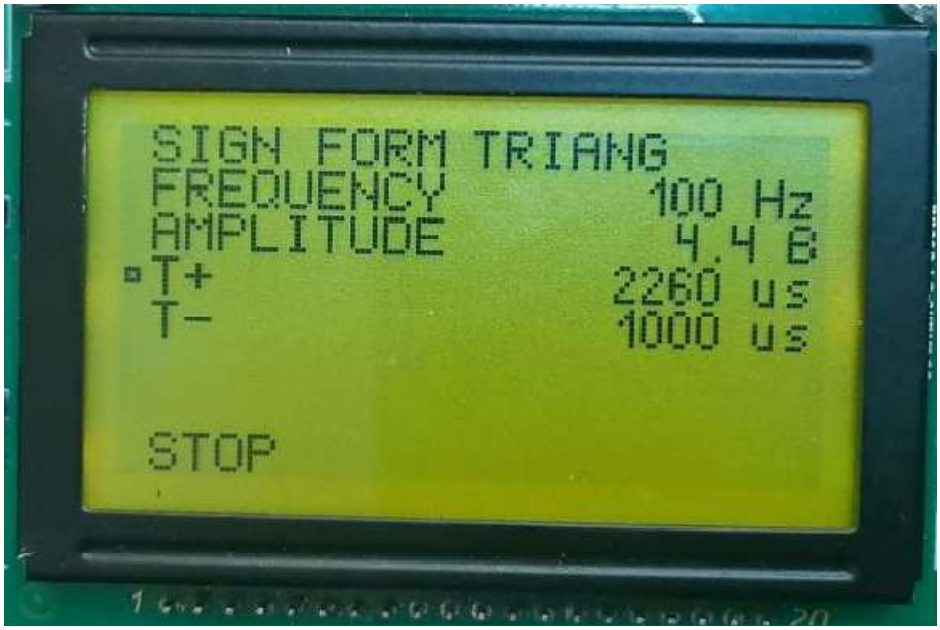

Рисунок 8 – Меню с выбранным сигналом треугольник 100 Гц время положительного фронта 2,26 мс, время отрицательного фронта 1 мс 4,4 В

Рисунок 9 – Сигнал на выходе ЦАП треугольник 100 Гц время положительного фронта 2,26 мс, время отрицательного фронта 1 мс 4,4 В

Заключение

В ходе выполнения курсовой работы был разработан генератор электрических сигналов различной формы. Взаимодействие пользователя с устройством осуществляется по средствам меню, выводимого на графическом дисплее, и матричной клавиатуры для ввода параметров сигнала.

В разработанном генераторе есть ограничения по частотам, так для синуса максимальная частота равна 750 Гц, для меандра минимальное время импульса и паузы 10 мкс, для треугольника минимальный фронт и спад занимает 820 мкс, однако предусмотрен режим пилообразного сигнала, при котором один из фронтов занимает 5 мкс.

Список использованной литературы

1. Datasheet STM8S207xx ARM-based MCUs. STMicroelectronics. — URL: https://stud.lms.tpu.ru/pluginfile.php/1673828/mod_resource/content/1/stm8s207m8.pdf (дата обращения: 05.06.2023). — Текст: электронный.
2. Reference manual STM8S207xx ARM - based MCUs.STMicroelectronics.2015. — Режим доступа: https://stud.lms.tpu.ru/pluginfile.php/1673829/mod_resource/content/1/rm0016-stm8s-series-and-stm8afseries-8bit-microcontrollers-stmicroelectronics.pdf (дата обращения: 07.06.2023). — Текст: электронный.
3. Техническая документация на дисплей MT–12864J. – URL: https://stud.lms.tpu.ru/pluginfile.php/1675180/mod_resource/content/1/MT-1286J%20datasheet.pdf (дата обращения 10.06.2023) – Текст: электронный.
4. Datasheet Texas Instruments DAC7512N/3K. – URL: https://www.rlocman.ru/datasheet/data.html?di=243783&/DAC7512N (дата обращения 15.06.23). — Текст: электронный.

Приложение А. Принципиальная схема

Приложение Б. Перечень элементов

Приложение В. Код программы

```
#include <iostm8s207.h>
#include <main.h>

int delay_number; //Переменная времени delay
int delay_count; //Переменная счетчик для delay
int counter_64; //Переменная счетчик для clean
int counter_pages; // Переменная счетчик для clean
int x_coord; //Координата X для установки курсора
int y_coord; //Координата У для установки курсора
int symbol_number; //Переменная, хранящая число для вывода
int write_counter; //Переменная счетчик для вывода числа
int crystal_numb; //Переменная выбора кристала
int symb_type; //Переменная для выбора типа символа
int cursor; // Переменная текущего положения курсора
int pre_cursor; // Переменная предыдущего положения курсора
int form_status; // Переменная выбора формы
int pre_form_status; // Переменная предыдущей формы
int amplitude_value_10;//Переменная амплитудного значения * 10
int pre_amplitude_value_10;// Переменная пердыдущего амплитудного значения * 10
long int tplus_value; //Переменная, хранящая значение T+
long int pre_tplus_value; //Переменная, хранящая предыдущее значение T+
long int tminus_value;//Переменная, хранящая значение T-
long int pre_tminus_value; //Переменная, хранящая предыдущее значение T-
long int frequency_value; // Переменная, хранящая значение частоты
long int pre_frequency_value; // Переменная, хранящая значение предыдущей частоты
int number_button;// Переменная номера нажатой кнопки
int raw_number_button;// Переменная с сырым номером нажатой кнопки
int counter_raws; //Переменная счетчика строк матричной клавиатуры
int counter_columns; //Переменная счетчика колонок матричной клавиатуры
int is_it_this_button; //Переменная для определения какая кнопка нажата на строки
int left_right_menue_flag; //Переменная флаг для определения какое меню сейчас работает
int start_stop_flag; //Переменная флаг старт стоп
int counter_exponent; // Переменная счетчик порядков числа
int current_exponent_value; // Переменная, хранящая в себе значение текущего порядка в числе
long int current_exponent_check; // Переменная для проверки последний ли порядок у числа
long int timer_delay_us; //Переменная хранящая значение задержки в микросекундах
long int triang_pause; //Переменная времени паузы в треугольнике
int sin_pause; //Переменная времени паузы в синусе
int print_value; // Переменная для вывода на ЦАП
int chip_select; // Переменная для выбора ЦАПа
int dac_counter; // Переменная счетчик ЦАПа
int dac_msb; // Переменная для старших битов в выводе ЦАПа
int dac_lsb; // Переменная для младших битов в выводе ЦАПа
int raw_dac_value; // Переменная для вывода сырых значений ЦАПа 0 - 4095
int triang_counter; // Переменная счетчик для вывода треугольника
int sin_counter; // Переменная счетчик для синуса
long int dac_sin_value; // Переменная для вывода синуса

int numb_arr[11][5] ={{0x3E,0x41,0x41,0x3E,0x00}, //0
```

```
                        {0x00,0x04,0x02,0x7F,0x00}, //1
                        {0x62,0x51,0x49,0x46,0x00}, //2
                        {0x22,0x49,0x49,0x36,0x00}, //3
                        {0x0F,0x08,0x08,0x7F,0x00}, //4
                        {0x4F,0x49,0x49,0x31,0x00}, //5
                        {0x3E,0x49,0x49,0x32,0x00}, //6
                        {0x01,0x71,0x09,0x07,0x00}, //7
                        {0x36,0x49,0x49,0x36,0x00}, //8
                        {0x26,0x49,0x49,0x3E,0x00}, //9
                        {0x00,0x00,0x40,0x00,0x00}  //.
                        };
int Hsymb_arr[36][6] ={{0x7E, 0x09, 0x09, 0x09, 0x7E, 0x00}, //A0
                        {0x7E, 0x49, 0x49, 0x49, 0x36, 0x00}, //B1
                        {0x3E, 0x41, 0x41, 0x41, 0x22, 0x00}, //C2
                        {0x7F, 0x41, 0x41, 0x41, 0x3E, 0x00}, //D3
                        {0x7F, 0x49, 0x49, 0x49, 0x41, 0x00}, //E4
                        {0x7F, 0x09, 0x09, 0x09, 0x01, 0x00}, //F5
                        {0x3E, 0x41, 0x49, 0x49, 0x3A, 0x00}, //G6
                        {0x7F, 0x08, 0x08, 0x08, 0x7F, 0x00}, //H7
                        {0x00, 0x41, 0x7F, 0x41, 0x00, 0x00}, //I8
                        {0x20, 0x41, 0x3F, 0x01, 0x00, 0x00}, //J9
                        {0x7F, 0x08, 0x14, 0x22, 0x41, 0x00}, //K10
                        {0x7F, 0x40, 0x40, 0x40, 0x40, 0x00}, //L11
                        {0x7F, 0x02, 0x04, 0x02, 0x7F, 0x00}, //M12
                        {0x7F, 0x04, 0x08, 0x10, 0x7F, 0x00}, //N13
                        {0x3E, 0x41, 0x41, 0x41, 0x3E, 0x00}, //O14
                        {0x7F, 0x09, 0x09, 0x09, 0x06, 0x00}, //P15
                        {0x3E, 0x41, 0x41, 0x21, 0x5E, 0x00}, //Q16
                        {0x7F, 0x09, 0x19, 0x29, 0x46, 0x00}, //R17
                        {0x26, 0x49, 0x49, 0x49, 0x32, 0x00}, //S18
                        {0x01, 0x01, 0x7F, 0x01, 0x01, 0x00}, //T19
                        {0x3F, 0x40, 0x40, 0x40, 0x3F, 0x00}, //U20
                        {0x1F, 0x20, 0x40, 0x20, 0x1F, 0x00}, //V21
                        {0x3F, 0x40, 0x30, 0x40, 0x3F, 0x00}, //W22
                        {0x43, 0x24, 0x18, 0x24, 0x43, 0x00}, //X23
                        {0x07, 0x08, 0x70, 0x08, 0x07, 0x00}, //Y24
                        {0x61, 0x51, 0x49, 0x45, 0x43, 0x00}, //Z25
                        {0x08, 0x08, 0x08, 0x08, 0x08, 0x00}, //-26
                        {0x08, 0x08, 0x3E, 0x08, 0x08, 0x00}, //+27
                        {0x14, 0x14, 0x14, 0x14, 0x14, 0x00}, //=28
                        {0x00, 0x00, 0x22, 0x00, 0x00, 0x00},//:29
                        {0x00, 0x1C, 0x1C, 0x1C, 0x00, 0x00},//.30
                        {0x00, 0x00, 0x00, 0x00, 0x00, 0x00},//" "31
                        {0x3C, 0x40, 0x40, 0x7C, 0x00, 0x00},//u32
                        {0x48, 0x54, 0x54, 0x24, 0x00, 0x00},//s33
                        {0x44, 0x64, 0x54, 0x4C, 0x00, 0x00},//z34
                        {0x00, 0x1C, 0x14, 0x1C, 0x00, 0x00},//.35
                        };
int sin_arr[60] = {0,0,0,1,1,2,3,4,5,6,7,9,10,12,13,
                        15,17,18,20,21,22,24,25,26,27,28,
                        29,29,30,30,30,30,30,29,29,28,27,
                        26,25,24,22,21,20,18,17,15,13,12,
                        10,9,8,6,5,4,3,2,1,1,0,0};
```

```
void delay(int delay_number); // Функция простой задержки
void coord(int x_coord, int y_coord); // Функция выставления кординаты на экране
void write_numb(int symbol_numb); // Функция для вывода числа (5 ширина)
void write_symb(int symbol_numb); // Функция для вывода символа (6 ширина)
void crystal_select(int crystal_numb); // Функция выбора кристала
void clean_symb(int symb_type); // Функция очистки символа symb_type = 0 - numb, 1 - symb, 2 - string
void set_cursor(int cursor); // Выставляет знак курсора на указанную строку
void clean_cursor(int pre_cursor);// Очищает предыдущий курсор
void title_form(void);// Пишет строку SIGN FORM
void title_freq(void);// Пишет строку FREQUENCY
void title_ampl(void);// Пишет строку AMPLITUDE
void title_tplus(void);// Пишет строку T+
void title_tminus(void);// Пишет строку T-
void chose_form(int form_status);// Пишет на втором кристале тип формы
void title_not_avelible(int form_status);// Пишет NOT AVELIB в зависимости от формы
void print_freq(long int frequency_value, int form_status);// Пишет значение частоты
void print_ampl(int ampitude_value_10);// Пишет значение амплитуды
void print_tplus(long int tplus_value, int form_status);// Пишет значение T+
void print_tminus(long int tminus_value, int form_status); //Пишет значение T-
void print_start_stop(int start_stop_flag); // Пишет старт или стоп
void spi_write(int print_value); // Функция для вывода по SPI
void dac_write_raw(int raw_dac_value); // Функция для сырыз значений ЦАПа
void delay_us(long int timer_delay_us); // Функция задержки по таймеру в мкс
void strob(void); // Функция строба
void start(void); // Функция подготовки экрана
void clean(void); // Фунция полной очистки экрана

main()
{
        CLK_CKDIVR = 0x00; // Разгон частоты контроллера до 16 МГц
        SPI_CR1 = 0x05;
        SPI_CR2 = 0x03;
        SPI_CR1 ^= 0x40; // Включение SPI
        PA_DDR = 0xFF;
        PA_CR1 = 0xFF;
        PA_CR2 = 0xFF;
        PB_DDR = 0xFF;
        PF_DDR |= 0xF9;
        PB_CR1 = 0xFF;
        PB_CR2 = 0xFF;
        PF_CR1 |= 0xF9;
        PF_CR2 |= 0xF9;
        PE_DDR = 0xF0;
        PE_CR1 = 0xFF;
        PE_CR2 = 0x0F;
        EXTI_CR2 = 0x02;
        _asm("rim");

        start();
        clean();
        start_stop_flag = 0;
        cursor =0;
```

```
	pre_cursor = 7;
	left_right_menue_flag = 0;
	form_status = 0;
	pre_form_status = 1;
	frequency_value = 0;
	amplitude_value_10 = 0;
	tplus_value = 0;
	tminus_value = 0;
	title_form();
	title_freq();
	title_ampl();
	title_tplus();
	title_tminus();
	chose_form(form_status);
	print_start_stop(start_stop_flag);

	while (1){
		PE_CR2 = 0x0F;
		while(start_stop_flag == 0){ //Настройка параметров
			PE_CR2 = 0x0F;
			dac_write_raw(0);
			set_cursor(cursor);
			clean_cursor(pre_cursor);
			print_start_stop(start_stop_flag);
			if((pre_frequency_value != frequency_value) | (pre_form_status != form_status)|
			(pre_amplitude_value_10 != amplitude_value_10) | (pre_tplus_value != tplus_value) | (pre_tminus_value != tminus_value)){
				pre_form_status = form_status;
				pre_frequency_value = frequency_value;
				pre_amplitude_value_10 = amplitude_value_10;
				pre_tplus_value = tplus_value;
				pre_tminus_value = tminus_value;
				chose_form(form_status);
				print_freq(frequency_value, form_status);
				print_ampl(amplitude_value_10);
				print_tplus(tplus_value, form_status);
				print_tminus(tminus_value, form_status);
			}
		}
		while(start_stop_flag == 1){ //Вывод сигнала
			PE_CR2 = 0x0F;
			print_start_stop(start_stop_flag);
			while(start_stop_flag){
				if(form_status == 2){ //Вывод треугольника
					if((tplus_value == 0) & (tminus_value != 0)){
						if(tminus_value<810){
							triang_pause = ((1000000/frequency_value)- 810);
						}
						else{
							triang_pause = ((1000000/frequency_value)- tminus_value);
						}
						for(triang_counter = (amplitude_value_10*81); triang_counter>-1; triang_counter-=((amplitude_value_10*81)/30)){
```

```
                dac_write_raw(triang_counter);
                delay_us((tminus_value-810)/30);
            }
            dac_write_raw(0);
            if(triang_pause>65534){
                triang_pause = 65534;
            }
            delay_us(triang_pause);
        }
        else if((tplus_value != 0) & (tminus_value == 0)){
            if(tplus_value<810){
                triang_pause = ((1000000/frequency_value)- 810);
            }
            else{
                triang_pause = ((1000000/frequency_value)- tplus_value);
            }

            for(triang_counter = 0; triang_counter<(amplitude_value_10*81);
triang_counter+=((amplitude_value_10*81)/30)){
                dac_write_raw(triang_counter);
                delay_us((tplus_value-810)/30);
            }
            dac_write_raw(0);
            if(triang_pause>65534){
                triang_pause = 65534;
            }
            delay_us(triang_pause);
        }
        else{
            if((tminus_value == 0) & (tplus_value == 0)){
                triang_pause = ((1000000/frequency_value) - 1620);
            }
            if((tminus_value < 810) & (tplus_value > 810)){
                triang_pause = ((1000000/frequency_value) - 810 - tplus_value);
            }
            if((tplus_value < 810) & (tminus_value > 810)){
                triang_pause = ((1000000/frequency_value) - 810 - tminus_value);
            }
            if((tplus_value > 810) & (tminus_value > 810)){
                triang_pause = ((1000000/frequency_value) - tplus_value - tminus_value);
            }
            for(triang_counter = 0; triang_counter<(amplitude_value_10*81);
triang_counter+=((amplitude_value_10*81)/30)){
                dac_write_raw(triang_counter);
                delay_us((tplus_value-810)/30);
            }
            for(triang_counter = (amplitude_value_10*81); triang_counter>-1; triang_counter-
=((amplitude_value_10*81)/30)){
                dac_write_raw(triang_counter);
                delay_us((tminus_value-810)/30);
            }
            dac_write_raw(0);
            if(triang_pause>65534){
```

```
                                        triang_pause = 65534;
                                    }
                                    delay_us(triang_pause);
                                }
                            }
                            if(form_status == 1){ //Вывод меандра
                                dac_write_raw(amplitude_value_10*81);
                                delay_us(tplus_value-15);
                                dac_write_raw(0);
                                delay_us(tminus_value-15);
                            }
                            if(form_status == 0){ //Вывод синуса
                                if(frequency_value> 665){

                                    for(sin_counter = 0; sin_counter<60; sin_counter++){
                                        dac_sin_value = (sin_arr[sin_counter] * amplitude_value_10)/30;
                                        dac_write_raw(dac_sin_value*81);
                                        delay_us(0);

                                    }
                                }
                                else {
                                    sin_pause = ((16666/(frequency_value+1))-28);
                                    for(sin_counter = 0; sin_counter<60; sin_counter++){
                                        dac_sin_value = (sin_arr[sin_counter] * amplitude_value_10)/30;
                                        dac_write_raw(dac_sin_value*81);
                                        delay_us(sin_pause);
                                    }

                                }
                            }
                        }
                    }
                }
            }
        }
    }

    @far @interrupt void EXTI_PortE (void)
    {
        PE_CR2 = 0x00;
        number_button = 0;
        raw_number_button = PE_IDR;
        raw_number_button ^= 0x0F;
        for (counter_raws = 0; counter_raws < 4; counter_raws++){
            raw_number_button /= 2;
            if (raw_number_button == 0){
            break;
            }
        }
        raw_number_button = 0x10;
        for(counter_columns = 0; counter_columns<4; counter_columns++){
            PE_ODR = raw_number_button;
            raw_number_button = raw_number_button*2;
            is_it_this_button = ((~PE_IDR<<4)&0xFF)>>4;
```

```
            if (is_it_this_button == 0){
            break;
            }
}
number_button = counter_raws*4 + counter_columns;

if(number_button == 12){
            start_stop_flag ^=0x01;
}

if(start_stop_flag == 0x00){
            if(number_button==15){
                        left_right_menue_flag ^=0x01;
            }
            if (left_right_menue_flag == 0){
                        if(number_button==11){
                                    pre_cursor = cursor;
                                    cursor++;
                                    if(cursor > 4){
                                                cursor = 0;
                                    }
                        }
            }
            if (left_right_menue_flag == 1){
                        if(cursor == 0){
                                    if(number_button==11){
                                                form_status++;
                                                if(form_status > 2){
                                                            form_status = 0;
                                                }
                                    }
                        }
            }
            if (left_right_menue_flag == 1){
                        if((cursor == 1)&((form_status == 0)|(form_status == 2))){
                                    if(number_button == 0){
                                                frequency_value++;
                                    }
                                    else if(number_button == 1){
                                                frequency_value += 10;
                                    }
                                    else if(number_button == 2){
                                                frequency_value += 100;
                                    }
                                    else if(number_button == 3){
                                                frequency_value += 1000;
                                    }
                                    else if(number_button == 4){
                                                frequency_value += 10000;
                                    }
                                    else if(number_button == 5){
                                                frequency_value += 100000;
                                    }
```

```
                else if(number_button == 6){
                        frequency_value = 0;
                }
                if(frequency_value > 999999){
                        frequency_value = 0;
                }
        }
}

if (left_right_menue_flag == 1){
        if(cursor == 2){
                if(number_button == 0){
                        amplitude_value_10 +=1;
                }
                else if(number_button == 1){
                        amplitude_value_10 +=10;
                }
                else if(number_button == 6){
                        amplitude_value_10 = 0;
                }

                if(amplitude_value_10 > 50){
                        amplitude_value_10 = 0;
                }
        }
}
if(left_right_menue_flag == 1){
        if((cursor == 3)&((form_status == 1)|(form_status == 2))){
                if(number_button == 0){
                        tplus_value++;
                }
                else if(number_button == 1){
                        tplus_value += 10;
                }
                else if(number_button == 2){
                        tplus_value += 100;
                }
                else if(number_button == 3){
                        tplus_value += 1000;
                }
                else if(number_button == 4){
                        tplus_value += 10000;
                }
                else if(number_button == 5){
                        tplus_value += 100000;
                }
                else if(number_button == 6){
                        tplus_value = 0;
                }
                if(tplus_value > 999999){
                        tplus_value = 0;
                }
        }
```

```
                }
                        if(left_right_menue_flag == 1){
                        if((cursor == 4)&((form_status == 1)|(form_status == 2))){
                                if(number_button == 0){
                                        tminus_value++;
                                }
                                else if(number_button == 1){
                                        tminus_value += 10;
                                }
                                else if(number_button == 2){
                                        tminus_value += 100;
                                }
                                else if(number_button == 3){
                                        tminus_value += 1000;
                                }
                                else if(number_button == 4){
                                        tminus_value += 10000;
                                }
                                else if(number_button == 5){
                                        tminus_value += 100000;
                                }
                                else if(number_button == 6){
                                        tminus_value = 0;
                                }
                                if(tminus_value > 999999){
                                        tminus_value = 0;
                                }
                        }
                }
        }
                //number_button = 0;

                //delay(20000);
                //PE_CR2 = 0x0F;
                PE_ODR = 0x00;
}




void delay(int delay_number){
        for(delay_count=0; delay_count<delay_number; delay_count++);
}

void strob(void){
        PF_ODR |= 0x80;
        delay(5);
        PF_ODR &= 0x7F;
}

void start(void){
        PF_ODR &=0xEF;
```

```
	PF_ODR |=0x10;
	delay(1);
	PF_ODR &=0x7F;
	PF_ODR &=0xBF;
	PF_ODR &=0xDF;
	PF_ODR |= 0x09;
	PB_ODR = 0x3F;
	strob();
}

void clean (void){
	PF_ODR |= 0x09;		//Оба кристалла активны;
	for (counter_pages = 0xB8; counter_pages < 0xC0; counter_pages++){
		PF_ODR &= 0xBF; //Режим команд
		PB_ODR = counter_pages;
		strob();
		PB_ODR = 0x40; //установка строки в 0 на всякий случай
		strob();
		PF_ODR |= 0x40; //Режим данных
		PB_ODR = 0x00;
		for (counter_64 = 0; counter_64 < 64; counter_64++){
			strob();
		}
	}
	PF_ODR &= 0xBF; //Режим команд
}

void coord(int x_coord, int y_coord) {
	PF_ODR &= 0xBF;  //Режим команд
	PB_ODR = x_coord + 0x40;
	strob();
	PB_ODR = y_coord + 0xB8;
	strob();
	PF_ODR |= 0x40;  //Режим данных
}

void write_numb(int symbol_numb){
	PF_ODR |= 0x40;  //Режим данных
	for(write_counter = 0; write_counter < 5; write_counter++) {
		PB_ODR = numb_arr[symbol_numb][write_counter];
		strob();
	}
}

void write_symb(int symbol_numb){
	PF_ODR |= 0x40;  //Режим данных
	for(write_counter = 0; write_counter < 6; write_counter++) {
		PB_ODR = Hsymb_arr[symbol_numb][write_counter];
		strob();
	}
}

void crystal_select(int crystal_numb){
```

```
	if (crystal_numb == 0){
			PF_ODR |= 0x09;		//Оба кристалла активны;
	}
	if (crystal_numb == 1){
			PF_ODR |= 0x01; //E1 = 1 Кристалл 1 активен
			PF_ODR &= 0xF7; //E2 = 0 Кристалл 2 неактивен
	}
		if (crystal_numb == 2){
			PF_ODR &= 0xFE; //E1 = 0 Кристалл 1 неактивен
			PF_ODR |= 0x08; //E2 = 1 Кристалл 2 активен
	}
}

void clean_symb(int symb_type){
	PF_ODR |= 0x40;  //Режим данных
	if(symb_type == 0){
		write_counter = 0;
	}
	if(symb_type == 1){
		write_counter = -1;
	}
	if(symb_type == 2){
		write_counter = -59;
	}
	for(write_counter; write_counter < 5; write_counter++) {
		PB_ODR = 0x00;
		strob();
	}
}

void clean_cursor(int pre_cursor){
	crystal_select(1);
	coord(0, pre_cursor);
	clean_symb(1);
}

void set_cursor(int cursor){
	if(left_right_menue_flag == 0){
		crystal_select(1);
		coord(0, cursor);
		write_symb(30);
	}
	if(left_right_menue_flag == 1){
		crystal_select(1);
		coord(0, cursor);
		write_symb(35);
	}
}

void title_form(void){
	crystal_select(1);
	coord(6, 0);
	write_symb(18);//S
```

```
	write_symb(8);//I
	write_symb(6);//G
	write_symb(13);//N
	write_symb(31);//_
	write_symb(5);//F
	write_symb(14);//O
	write_symb(17);//R
	write_symb(12);//M
}

void title_freq(void){
	crystal_select(1);
	coord(6, 1);
	write_symb(5);//F
	write_symb(17);//R
	write_symb(4);//E
	write_symb(16);//Q
	write_symb(20);//U
	write_symb(4);//E
	write_symb(13);//N
	write_symb(2);//C
	write_symb(24);//Y
}

void title_ampl(void){
	crystal_select(1);
	coord(6, 2);
	write_symb(0);//A
	write_symb(12);//M
	write_symb(15);//P
	write_symb(11);//L
	write_symb(8);//I
	write_symb(19);//T
	write_symb(20);//U
	write_symb(3);//D
	write_symb(4);//E
}

void title_tplus(void){
	crystal_select(1);
	coord(6, 3);
	write_symb(19);//T
	write_symb(27);//+
}

void title_tminus(void){
	crystal_select(1);
	coord(6, 4);
	write_symb(19);//T
	write_symb(26);//-
}

void chose_form(int form_status){
```

```
	crystal_select(2);
	coord(0, 0);
	if (form_status == 0){
		write_symb(18);//S
		write_symb(8);//I
		write_symb(13);//N
		clean_symb(1);
		clean_symb(1);
		clean_symb(1);
		title_not_avelible(form_status);
	}
	else if (form_status == 1){
		write_symb(18);//S
		write_symb(16);//Q
		write_symb(20);//U
		write_symb(0);//A
		write_symb(17);//R
		write_symb(4);//E
		title_not_avelible(form_status);
	}
	else if (form_status == 2){
		write_symb(19);//T
		write_symb(17);//R
		write_symb(8);//I
		write_symb(0);//A
		write_symb(13);//N
		write_symb(6);//G
		title_not_avelible(form_status);
	}
}

void title_not_avelible(int form_status){
	crystal_select(2);
	if (form_status == 0){
		coord(0, 3);
		clean_symb(2);
		coord(0, 3);
		write_symb(13);//N
		write_symb(14);//O
		write_symb(19);//T
		write_symb(31);//" "
		write_symb(0);//A
		write_symb(21);//V
		write_symb(4);//E
		write_symb(11);//L
		write_symb(8);//I
		write_symb(1);//B
		coord(0, 4);
		clean_symb(2);
		coord(0, 4);
		write_symb(13);//N
		write_symb(14);//O
		write_symb(19);//T
```

```
                write_symb(31);//" "
                write_symb(0);//A
                write_symb(21);//V
                write_symb(4);//E
                write_symb(11);//L
                write_symb(8);//I
                write_symb(1);//B

        }
        else if (form_status == 1){
                        coord(0, 1);
                        clean_symb(2);
                        coord(0, 1);
                        write_symb(13);//N
                        write_symb(14);//O
                        write_symb(19);//T
                        write_symb(31);//" "
                        write_symb(0);//A
                        write_symb(21);//V
                        write_symb(4);//E
                        write_symb(11);//L
                        write_symb(8);//I
                        write_symb(1);//B

        }
        else if (form_status == 2){
        }
}


void print_freq(long int frequency_value, int form_status){
        if((form_status == 0)|(form_status == 2)){

                crystal_select(2);
                coord(0, 1);
                clean_symb(2);
                coord(51, 1);
                write_symb(7);//H
                write_symb(34);//z

                current_exponent_check = frequency_value;
                if (frequency_value == 0){
                        coord(41 ,1);
                        write_numb(0);
                }
                else{
                        for(counter_exponent = 0; counter_exponent < 6; counter_exponent++){
                                current_exponent_value = current_exponent_check % 10;
                                current_exponent_check = current_exponent_check /10;
                                if ((current_exponent_value == 0) & (current_exponent_check == 0)){
                                        break;
                                }
                                coord(41 - (counter_exponent * 5),1);
```

```
                write_numb(current_exponent_value);
            }
        }
    }
}

void print_ampl(int ampitude_value_10){
        crystal_select(2);
        coord(0, 2);
        clean_symb(2);
        coord(57, 2);
        write_symb(1);//B

        if (ampitude_value_10 == 0){
            coord(47 ,2);
            write_numb(0);
        }
        else{
            current_exponent_value = ampitude_value_10;
            coord(47 ,2);
            write_numb(current_exponent_value % 10);
            coord(42 ,2);
            write_numb(10);
            coord(37 ,2);
            write_numb(current_exponent_value / 10);
        }
}

void print_tplus(long int tplus_value, int form_status){
    if((form_status == 1)|(form_status == 2)){

        crystal_select(2);
        coord(0, 3);
        clean_symb(2);
        coord(51, 3);
        write_symb(32);//u
        write_symb(33);//s

        current_exponent_check = tplus_value;
        if (tplus_value == 0){
            coord(41 ,3);
            write_numb(0);
        }
        else{
            for(counter_exponent = 0; counter_exponent < 6; counter_exponent++){
                current_exponent_value = current_exponent_check % 10;
                current_exponent_check = current_exponent_check /10;
                if ((current_exponent_value == 0) & (current_exponent_check == 0)){
                    break;
                }
                coord(41 - (counter_exponent * 5),3);
                write_numb(current_exponent_value);
            }
```

```
                }
        }
}

void print_tminus(long int tminus_value, int form_status){
        if((form_status == 1)|(form_status == 2)){

                crystal_select(2);
                coord(0, 4);
                clean_symb(2);
                coord(51, 4);
                write_symb(32);//u
                write_symb(33);//s

                current_exponent_check = tminus_value;
                if (tminus_value == 0){
                        coord(41 ,4);
                        write_numb(0);
                }
                else{
                        for(counter_exponent = 0; counter_exponent < 6; counter_exponent++){
                                current_exponent_value = current_exponent_check % 10;
                                current_exponent_check = current_exponent_check /10;
                                if ((current_exponent_value == 0) & (current_exponent_check == 0)){
                                        break;
                                }
                                coord(41 - (counter_exponent * 5),4);
                                write_numb(current_exponent_value);
                        }
                }
        }
}
void print_start_stop(int start_stop_flag){
        crystal_select(1);
        coord(6,7);
        if (start_stop_flag==1){
                write_symb(18);//S
                write_symb(19);//T
                write_symb(0);//A
                write_symb(17);//R
                write_symb(19);//T
        }
        else if (start_stop_flag == 0){
                write_symb(18);//S
                write_symb(19);//T
                write_symb(14);//O
                write_symb(15);//P
                write_symb(31);//" "
        }
}


void spi_write(int print_value){
```

```
	PE_CR2 = 0x00;
	while(!((SPI_SR&0x02)>>1));
	SPI_DR = print_value;
	while(!((SPI_SR&0x80)>>7));
	PE_CR2 = 0x0F;
}



void dac_write_raw( int raw_dac_value){
	dac_msb =  (raw_dac_value >> 8);
	dac_lsb = (raw_dac_value & 0xFF);
	PA_ODR = 0x00;
	spi_write(dac_msb);
	spi_write(dac_lsb);
	PA_ODR = 0xFF;
}


void delay_us(long int timer_delay_us){
	if(timer_delay_us > 0){
		TIM1_PSCRL = 0x0F;
		TIM1_ARRH = timer_delay_us >> 8;
		TIM1_ARRL = (timer_delay_us & 0xFF) ;
		TIM1_CR1 |= 0x09;
		TIM1_SR1 &=0xFE;
		while((TIM1_SR1 & 0x01) == 0x00){}
		TIM1_SR1 &=0xFE;
	}
}
```