МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«МИРЭА - Российский технологический университет»

РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения
Кафедра КБ-6 «Приборы и информационно-измерительные системы»

КУРСОВАЯ РАБОТА

по дисциплине Методы и средства автоматизации проектирования интеллектуальных измерительных приборов

Тема курсовой работы Разработка и отладка встраиваемого программного обеспечения интеллектуального измерительного прибора

Студента _____ А.А.Атюнькин
дата, подпись инициалы и фамилия

Группа БПБО-02-20 шифр 20Б0917

Обозначение работы КР–02068717–12.03.01–КБ-6–20–21

Работа защищена на оценку _____

Руководитель курсовой работы _____ С.А. Канаев
дата, подпись инициалы и фамилия

Члены комиссии _____ О.В. Москаленко
подпись инициалы и фамилия

_____
подпись инициалы и фамилия

Работа представлена к защите «___» _____20__ г.

Допущен к защите «___» _____20__ г.

Москва 2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "МИРЭА - РОССИЙСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ"

ИНСТИТУТ КОМПЛЕКСНОЙ БЕЗОПАСНОСТИ И СПЕЦИАЛЬНОГО ПРИБОРОСТРОЕНИЯ
КАФЕДРА КБ-6 «ПРИБОРЫ И ИНФОРМАЦИОННО-ИЗМЕРИТЕЛЬНЫЕ СИСТЕМЫ»

УТВЕРЖДАЮ

Заведующий кафедрой КБ-6

_____ /А.Б. Снедков/
подпись ФИО

«___»_____ 20___г.

ЗАДАНИЕ

на выполнение курсовой работы по дисциплине

Методы и средства автоматизации проектирования интеллектуальных измерительных приборов

Студент Атюнькин Алексей Андреевич
Ф.И.О

шифр студенческого билета 20Б0917 группа БПБО-02-20

1. Тема: Разработка и отладка встраиваемого программного обеспечения интеллектуального измерительного прибора
2. Срок сдачи студентом законченной работы **27 мая 2021г**.
3. Исходные данные для проектирования вариант№34 Разработать управляющую программу на языке Си для устройства на базе микроконтроллера с ядром AVR. Тип микроконтроллера ATmega32. Базовая платформа - EasyAVR v7 Development System фирмы mikroElektronika. Наименование: часы реального времени, с функцией секундомера и таймера на базе DS1307, с сенсорной панелью. Отображаемые и корректируемые (с панели управления) параметры выводимые на индикатор: время (часы, минуты); дата (число, месяц). Тип индикатора - графический ЖКИ (ME-GLCD 128x64). Тип сенсорной панели управления - ME-TOUCH SCREEN. Реализовать режим секундомера и таймера(в качестве параметра таймера служит время, установленное пользователем).

Руководитель работы _____ / Канаев С.А. /
Подпись Ф.И.О.

Задание принял к исполнению **25 марта 2021г**.

Студент _____ /_____/
Подпись Ф.И.О.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Основная цель курсовой работы – научиться работать с микроконтроллерами и получить навыки в разработке программ на языке Си, реализовать функции секундомера, таймера и часов на базе модуля часов реального времени DS1307 на отладочной плате EasyAVR V7 Development System фирмы mikroElektronika. Необходимые изменяемые параметры: время часов, дата, время таймера. Данные выводятся на графический жидкокристаллический дисплей (ME-GLCD 128x64). Управление прибором происходит с помощью сенсорной панели - ME-TOUCH SCREEN.

Использованный пакет для написания программы – Microchip Studio.

Отладка производилась на отладочной плате EasyAVR V7 Development System фирмы mikroElektronika.

1 ОПИСАНИЕ АППАРАТНОЙ ЧАСТИ И ОБЩЕГО АЛГОРИТМА РАБОТЫ ПРИБОРА

1.1 Краткое описание аппаратной части прибора

Цифровые часы реального времени с возможностью секундомера, таймера спроектированы на основе платы EasyAVRV7 Development System фирмы mikroElektronika [1]. Данная плата является отличным инструментом для обучения и изучения того, как работать с микроконтроллерами. На данной отладочной плате имеются встроенный программатор, светодиоды, разъемы с выведенными портами I/O, имеет возможность подключать дисплеи, датчики температуры и реального времени. Согласно схеме на рисунке 1 микроконтроллер ATmega32 обращается к датчику часов реального времени (DS1307), дисплею ME-GLCD и сенсорной панели ME-TOUCH-SCREEN. Для связи между микроконтроллером и датчиком часов реального времени используется последовательная асимметричная шина. Для начала приема данных от датчика часов, сначала по шине передается адрес устройства, после чего происходит формирование повторного старта, после которого, микроконтроллер работает, как устройство для приема данных, а датчик часов реального времени как главное устройство, которое посылает данные. После завершения приема данных, полученные данные обрабатываются и передаются по шине в порт данных жидкокристаллического дисплея, где предварительно был установлен режим приема данных и их записи. Управление дисплеем осуществляется с помощью сенсорной панели.

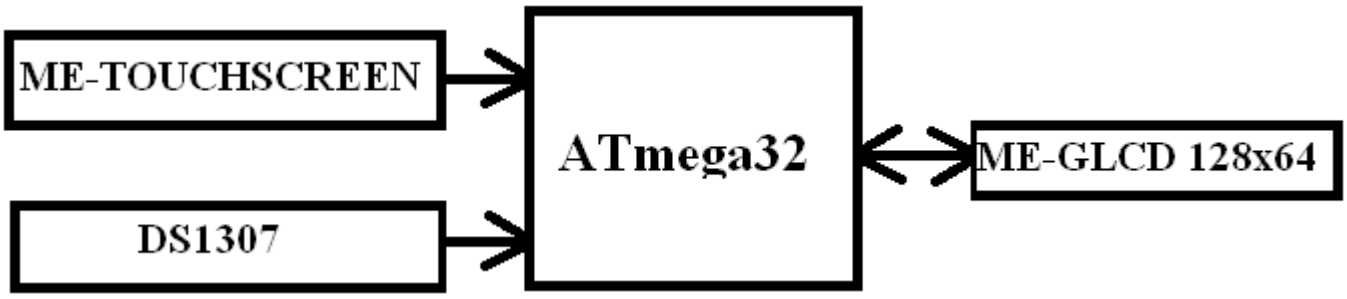

Рисунок 1 – Схема устройства

1.2 Описание компонентов устройства

1.2.1 Микроконтроллер ATmega32

Микроконтроллер Atmel ATmega32 [2] содержит в себе 32 рабочих регистра, 32 килобайта встроенной программируемой флэш-памяти общего назначения линий ввода/вывода, три гибких таймера/счетчика с режимами сравнения, байт-ориентированный двухпроводной последовательный интерфейс, 8-канальный 10-разрядный АЦП. Atmel ATmega32 представляет собой мощный микроконтроллер, который обеспечивает очень гибкое и экономичное решение для многих встроенных приложений управления. Схема расположения выходов микроконтроллера представлена на рисунке 1.

Основные характеристики:

- расширенная архитектура RISC;
- сегменты энергонезависимой памяти повышенной надежности;
- байтовый двухпроводной последовательный интерфейс;
- 32 Кбайт внутрисистемной самопрограммируемой флэш-памяти.

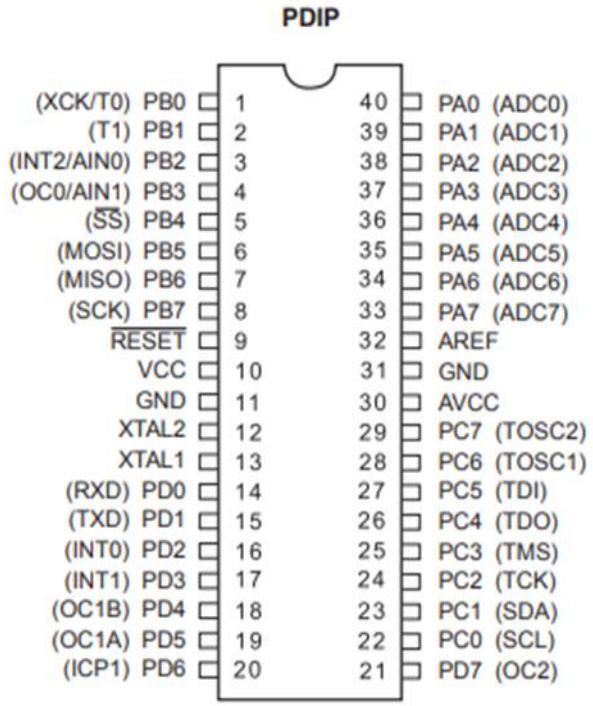

Рисунок 2 – Схема выводов микроконтроллера микроконтроллер ATmega32

1.2.2 Часы реального времени DS1307

Часы реального времени с последовательным интерфейсом DS1307 [3] – это малопотребляющие полные двоично-десятичные часы-календарь, включающие 56 байтов энергонезависимой статической ОЗУ. Адреса и данные передаются последовательно по двухпроводной двунаправленной шине. Часы-календарь отсчитывают секунды, минуты, часы, день, дату, месяц и год. Последняя дата месяца автоматически корректируется для месяцев с количеством дней меньше 31, включая коррекцию високосного года. Часы работают как в 24-часовом, так и в 12-часовом режимах с индикатором AM/PM. DS1307 имеет встроенную схему наблюдения за питанием, которая обнаруживает перебои питания и автоматически переключается на питание от батареи.

Процедура отправки данных устройству, подключенному к микроконтроллеру по шине I2C, начинается с формирования сигнала старт, который производится путем установки напряжения на линии SDA (Serial Data) в 0 В, при значении напряжения на лини SCL (Serial Clock) равному 5 В. После формирования сигнала старт, посылается адрес выбираемого устройства. Если устройство с выбранным адресом существует, то оно возвращает бит подтверждения. Дальше производится последовательная отправка данных выбранному устройству. Для завершения отправки данных, необходимо сформировать сигнал стоп, который формируется при переходе лини SDA из напряжения 0 В к напряжению 5 В при напряжении на линии SCL равному 5 В. В случае, если необходимо получить данные с датчика, необходимо после получения бита подтверждения, что устройство, с выбранным адресом, было обнаружено, произвести повторный старт, после чего микроконтроллер будет считывать данные с датчика.

DS1307 работает как ведомое устройство. Доступ к нему достигается установкой условия старт и передачей устройству идентификационного кода, за которым следует адрес регистра и формирования повторного старта. К следующим за ним регистрам доступ осуществляется последовательно, пока не будет выполнено

условие STOP. Часы реального времени и соединение его выводов с микроконтроллером представлены на рисунке 3.

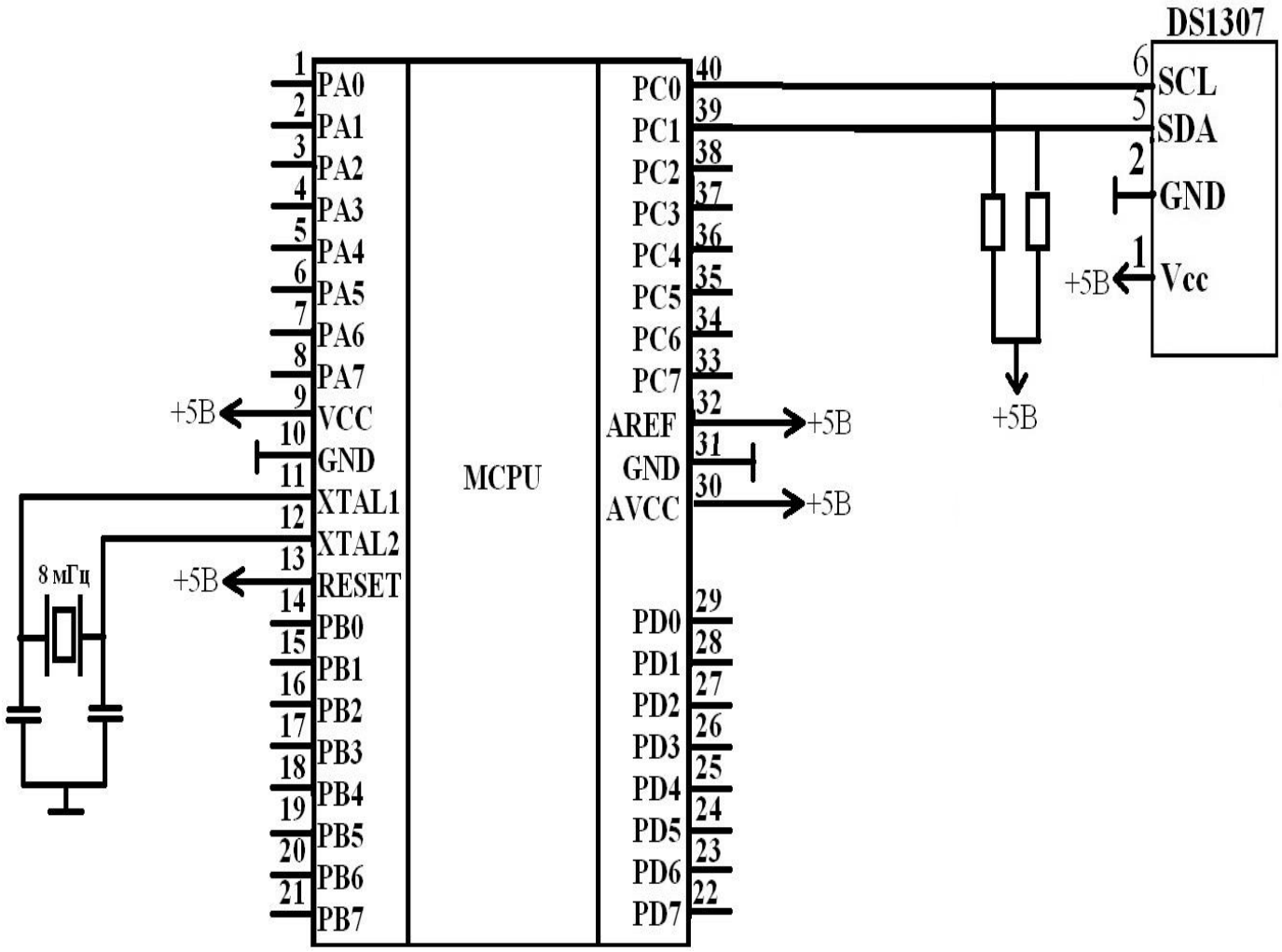

Рисунок 3 – Интегральная схема часов реального времени

1.2.3 Графический ЖКИ ME-GLCD 128x64

ME-GLCD128x64 [4] – графический жидкокристаллический дисплей размером 128 на 64 пикселя. Этот дисплей предусмотрен для установки на платы лабораторных стендов для изучения микроконтроллеров от фирмы mikroElektronika. Совместим с сенсорным экраном ME-TOUCH SCREEN, разъем для подключения которого установлен на отладочных стендах. Данные дисплея находятся в портах D0, D1 … D7, которые подключены к портам PC0, PC1 ... PC7 микроконтроллера соответственно. Дисплей содержит порты D/I и R/W, которые подключены к порту микроконтроллера PORTA. Порт D/I отвечает за то, как жидкокристаллический

дисплей будет интерпретировать полученный байт, в случае если D/I устанавливается в значение единицы, дисплей воспринимает полученный байт, как данные, которые в дальнейшем можно будет отобразить на экране, если D/I равно нулю, то дисплей воспринимает полученный байт как команду, которая в дальнейшем будет исполнена. Порт R/W отвечает за то, будут данные, расположенные в порте данных дисплея считываться или записываться. Установка порта в значение единицы означает что, данные будут считываться, если в значение ноль, то будет произведена запись данных, расположенных в порте данных дисплея. Дисплей имеет два порта CS1 и CS2, подключенные к порту PORTB микроконтроллера, которые отвечают за выбор половины экрана, на которой будет производится запись данных. Для того, чтобы произвести выбор половины, необходимо установить соответствующий порт в значение ноль. Графический ЖКИ и связь его выводов с микроконтроллером показаны на рисунке 4.

1.2.4 Сенсорная панель ME-TOUCHSCREEN

Сенсорная панель представляет собой стеклянную панель и гибкой пластиковой мембраны, поверхность которой покрыта двумя слоями резистивного материала. При нажатии на экран внешний слой надавливается на внутренний слой, и микроконтроллер с помощью аналогово-цифрового преобразователя (АЦП) может измерить это давление и точно определить его местоположение и преобразовать их в координаты X и Y. Для определения координаты Y на верхний электрод подается напряжение 5 В, а нижний заземляется. Левый и правый электрод соединяются и происходит проверка соответствующего коду АЦП. После чего, для определения координаты по оси X, верхний и правый электроды соединяются, на правый электрод подается напряжение 5 В, а левый заземляется.

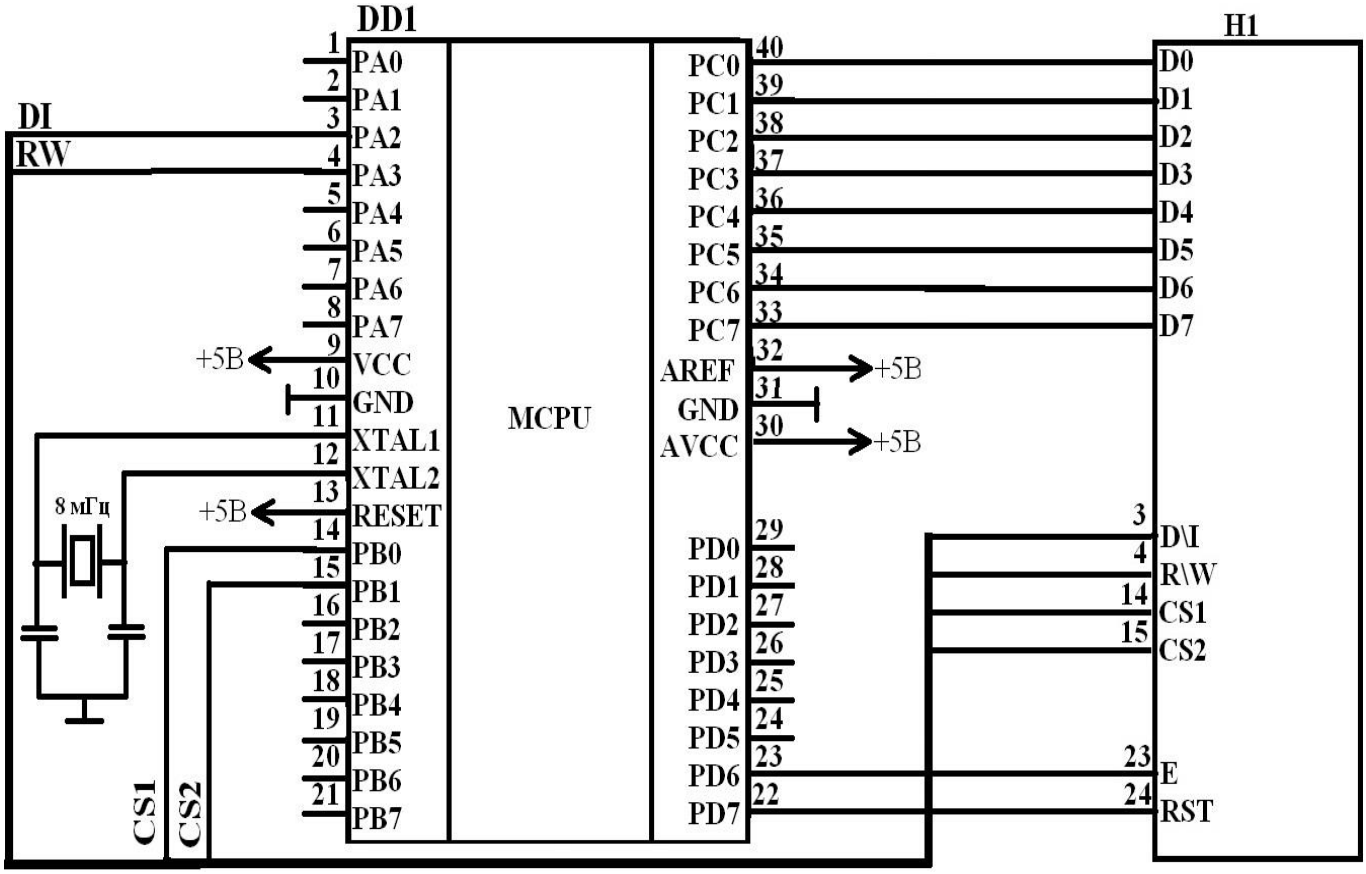

Рисунок 4 - Графический ЖКИ ME-GLCD 128x64

1.3 Общий алгоритм работы устройства

Программа разрабатывалась для отладочной платы EasyAVR V7 фирмы mikroElektronika. При включении питания, происходит настройка портов и инициализация всех аппаратных средств и отрисовка стартового меню с выбором языка, представленного на рисунке 5, после нажатия на кнопку “Русский” или “English”, происходит отрисовка меню настройки времени и даты, представленного на рисунке 6 в соответствии с выбранным языком. После окончания настройки, установленная дата и время записываются в датчик часов реального времени DS1307 и происходит отрисовка главного меню, представленного на рисунке 7. Интерфейс пользователя разработан самостоятельно. При нажатии на кнопку настроек в правом верхнем углу, происходит переход в меню настроек прибора, в котором можно выбрать стиль даты и язык системы. При нажатии на одну из кнопок “1 мая”, “01.05” или “05.01”, происходит выбор того, как будет отображаться дата в

главном меню. При нажатии на кнопку “РУ” или “ENG”, происходит выбор языка прибора. При нажатии на кнопку “Назад”, происходит возвращение в главное меню. Меню настроек прибора показано на рисунке 8.

Рисунок 5 – Стартовое меню выбора языка

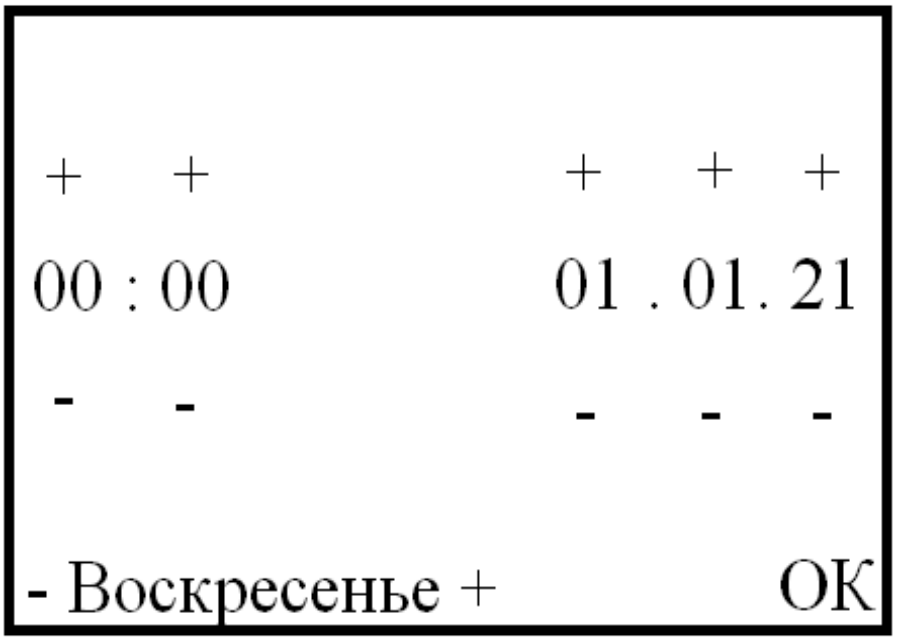

Рисунок 6 – Стартовое меню настройки времени и даты

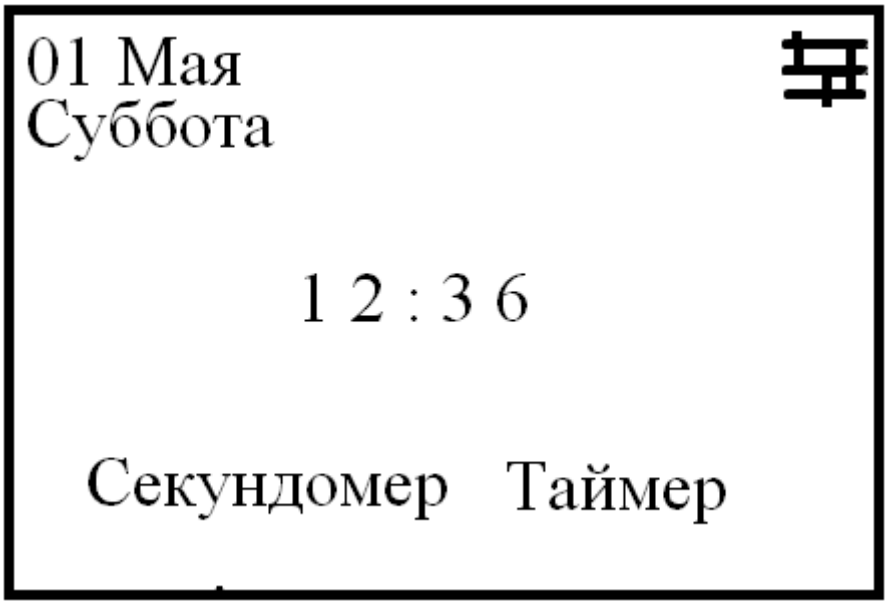

Рисунок 7 – Главное меню

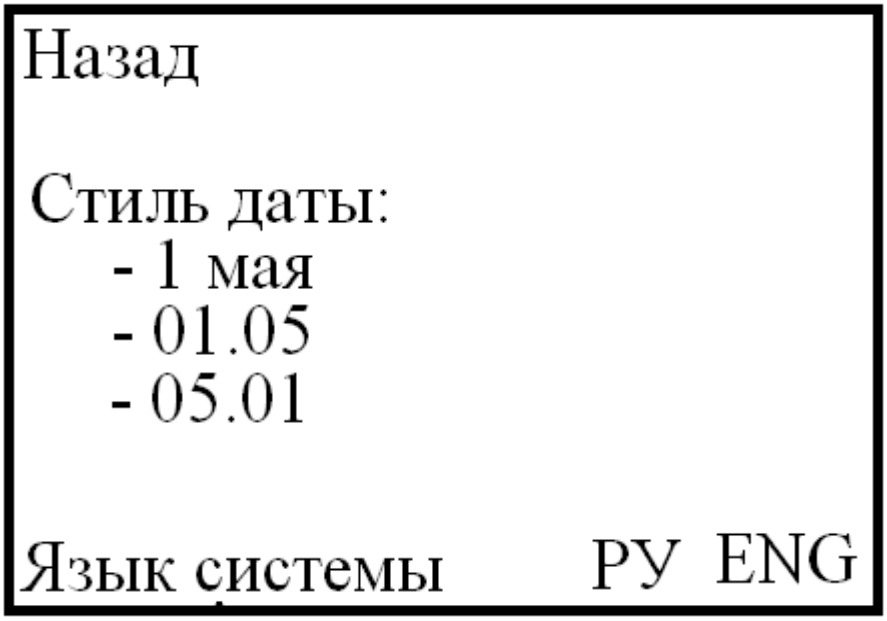

Рисунок 8 – Настройки прибора

В главном меню после нажатия на кнопку “Секундомер”, происходит переход в меню секундомера, показанного на рисунке 9. При нажатии на кнопку “Назад” происходит возвращение в главное меню прибора. Управление секундомером происходит путем нажатия на кнопки “Старт и “Стоп”, которые запускают и останавливают секундомер соответственно. После нажатия кнопки “Старт” происходит перерисовка меню секундомера, в котором добавляется кнопка “Сброс”.

После нажатия кнопки “Сброс”, время на секундомере останавливается и сбрасывается до первоначального значения.

При нажатии на кнопку “Таймер” в главном меню, происходит переход в меню таймера, показанного на рисунке 10. В правой части экрана находятся кнопки управления таймером, при нажатии на которые, происходит прибавление или убавление времени таймера. При нажатии на кнопку “Старт” или “Назад”, происходит запуск или возвращение в главное меню соответственно. Меню таймера после запуска показано на рисунке 11.

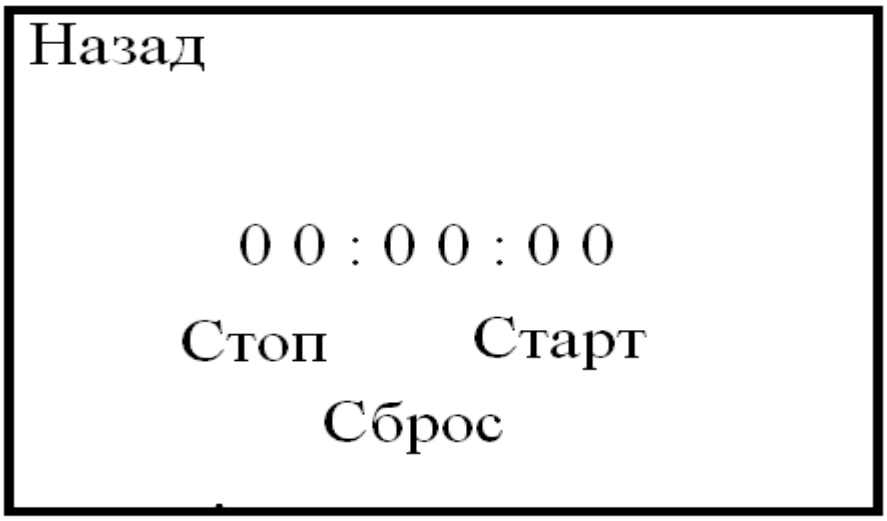

Рисунок 9 – Полное меню секундомера

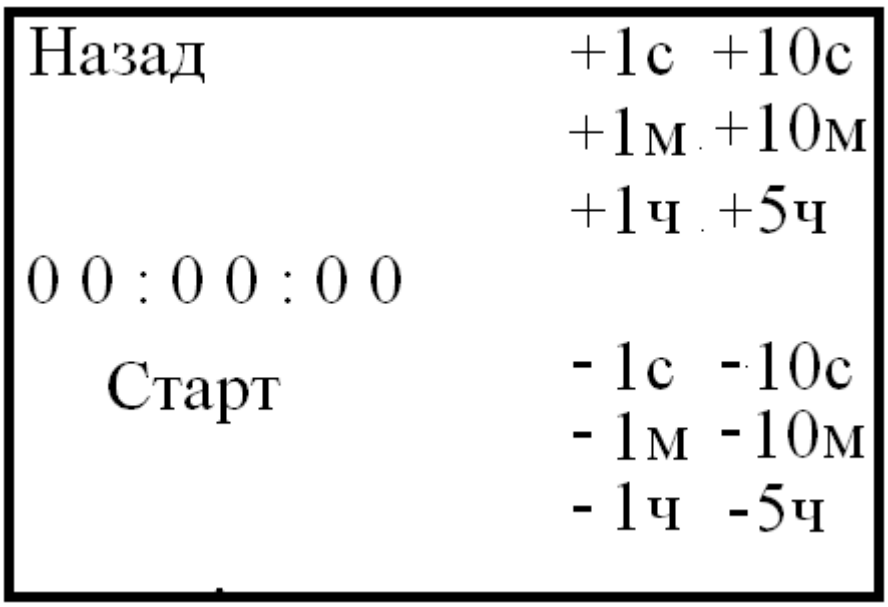

Рисунок 10 – Меню таймера до запуска

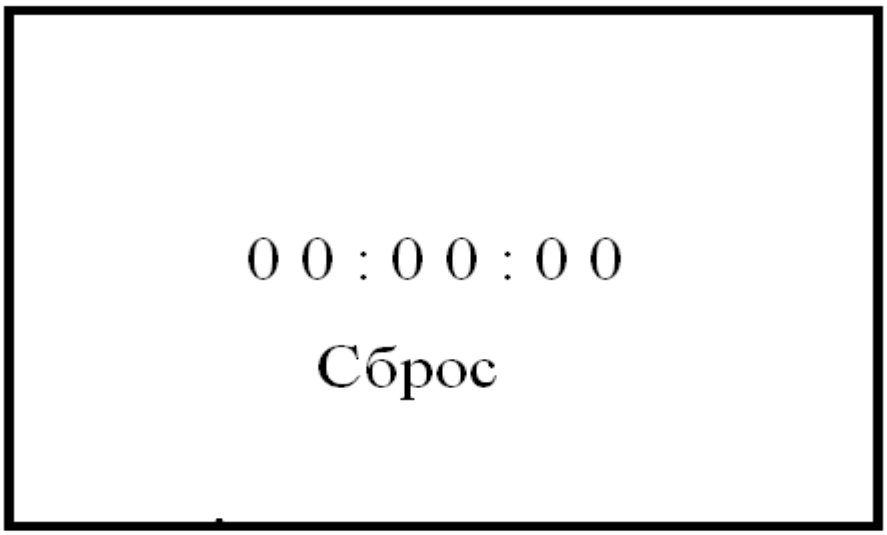

Рисунок 11 – Меню таймера после запуска

2 РАЗРАБОТКА ПРОГРАММЫ

2.1 Инициализация таймера-счетчика 1

При инициализации таймера-счетчика 1 первоначально происходит обнуление регистра TCNT1, чтобы избежать неточности в прерывании при включении и перезапуске прибора. Значение, записываемое в регистр OCR1A, соответствует такому значению, чтобы прерывание таймер-счетчика один происходило каждую секунду.

Расчет значения регистра OCR1A происходит по формуле 1, в соответствии с официальной документацией к микроконтроллеру ATmega32 [2]:

$$\text{OCR1A} = \frac{Fcpu}{Prescaler} \ . \qquad (1)$$

За значение Prescaler из формулы (1) отвечает бит CS12 регистра TCCR1B, устанавливаемый в значение единицы, и значение Prescaler равно 256. Для установки режима CTC (Clear Timer on Compare), необходимо установить биты WGM11 и WGM10 регистра TCCR1A в значение ноль, а также бит WGM12 регистра TCCR1B в значение единицы. Для того чтобы таймер-счетчик один производил сравнение с регистром OCR1A, необходимо установить бит OCF1A регистра TIFR в значение единицы, а для разрешения прерывания по достижении значения из регистра OCR1A, необходимо установить бит OCIE1A регистра TIMSK в значение единицы.

2.2 Функция прерывания таймера-счетчика 1

Прерывание таймера-счетчика 1 настроено так, что оно происходит каждую секунду, поэтому прерывание таймера-счетчика 1 используется для точного подсчета времени секундомера и таймера, а также для получения данных о дате и

времени. Если секундомер запущен, значение переменной isStartedSWT равно единице, тогда каждую секунду значение переменной stopwatchTime[0] будет увеличиваться на единицу. Если таймер запущен, значение переменной isStartedTMR равно единице, тогда каждую секунду значение переменной timerTime[0] будет уменьшаться на единицу. Прием данных с датчика часов реального времени происходит при условии, что отображается главный экран. Функция прерывания таймера-счетчика 1 представлена на рисунке 13.

```
void Time_1_init()
{
        TCNT1=0x0000;
        OCR1A=31250;
        // Обнуление битов WGM10 и WGM11 для выбора режима CTC
        TCCR1A &= ~((1<<WGM11)|(1<<WGM10));
        // Установка 1 в бит WGM12 для установки режима CTC и установка значения предделителя в значение 256
        TCCR1B |= (1<<WGM12)|(1<<CS12);
        //Флаг сравнения значения TCNT1 с регистром OCR1A
        TIFR |= (1<<OCF1A);
        //Разрешение прерывания по сравнению с регистром OCR1A
        TIMSK |= (1<<OCIE1A);
}
```

Рисунок 12 – Функция инициализации таймера-счетчика 1

```
ISR (TIMER1_COMPA_vect)
{
        //Увеличиваем каждую секунду содержимое stopwatchTime[0]
        if(isStartedSWT) stopwatchTime[0]++;
        //Уменьшаем каждую секунду содержимое timerTime[0]
        if(isStartedTMR) timerTime[0]--;
        if(scene == 1)
        {
                //Получаем текущее время
                getCurrentTime();
                //Получаем текующую дату
                getCurrentDate();
        }
}
```

Рисунок 13 – Функция прерывания таймера-счетчика 1

2.3 Функция инициализации таймера-счетчика 0

При инициализации таймера-счетчика 0 первоначально происходит обнуление регистра TCNT0, чтобы избежать неточности в прерывании при включении и перезапуске прибора. Значение, записываемое в регистр OCR0, соответствует такому значению, чтобы прерывание таймера-счетчика ноль происходило каждые

8 миллисекунд.

Расчет значения регистра OCR0 производится по формуле 2, в соответствии с официальной документацией [2]:

$$OCR0 = \frac{Fcpu}{Prescaler} \quad . \tag{2}$$

За значение Prescaler из формулы (2) отвечает бит CS02 регистра TCCR0, устанавливаемый в значение единицы, и значение Prescaler равно 256. Для установки режима CTC (Clear Timer on Compare), необходимо установить биты WGM00 и WGM01 регистра TCCR0 в значение ноль и один соответственно. Для того чтобы таймер-счетчик 0 производил сравнение с регистром OCR0, необходимо установить бит OCF0 регистра TIFR в значение единицы

```
void Timer0_init()
{
        TCNT0 = 0;
        OCR0=249;
        TCCR0 |= (1<<WGM01)|(1<<CS02); //Установка Prescaler = 256 и режима CTC
        TIMSK|=(1<<OCIE0);
        TIFR|=(1<<OCF0)|(1<<TOV0);
}
```

Рисунок 14 – Инициализация таймера-счетчика 0

2.4 Функция прерывания таймера-счетчика 0

Функция прерывания таймера-счетчика 0, представленная на рисунках 15, 16…27, используется для определения координаты X и Y при нажатии на сенсорный экран. При прерывании, сначала происходит определение координаты по оси X, для этого биты PA2 и PA3 порта PORTA устанавливаются в значение единицы и нуля соответственно, что соответствует определению положения нажатия на экране. Для того, чтобы получить значение кода АЦП, который определяет, где произошло нажатие по оси X в регистр ADMUX записывается число ноль. После в регистры

ADCSRA биты ADEN, ADSC, ADP2, ADP1 и ADP0 записывается единица. Дальше производится считывание значения АЦП, полученное при нажатии на экран, и его запись в переменную “x”. Для того, чтобы получить значение кода АЦП, который определяет, где произошло нажатие по оси Y в регистр ADMUX записывается единица. После в регистры ADCSRA биты ADEN, ADSC, ADP2, ADP1 и ADP0 записывается единица. Дальше производится считывание значения АЦП, полученное при нажатии на экран, и его запись в переменную “y”. После чего происходит проверка нажатия кнопки для каждой сцены. Если нажатие произошло на стартовом экране и соответствует координатам положения кнопок “Русский” и “English”, то осуществляется выбор соответствующего языка системы и переход на сцену настройки даты и времени. При нажатии на кнопки плюс или минус настроек времени, производится учет выбираемого времени. По превышении максимального допустимого значения для часов или минут, произойдет присваивание минимального допустимого значения соответствующей переменной. По превышении минимального допустимого значения для часов и минут, произойдет переход к максимально возможному значению, допустимого для данной переменной. Для настроек даты производится автоматический учет максимального значения дня в выбираемом месяце. По превышении максимального допустимого значения для значения дня, месяца или года, произойдет присваивание минимального допустимого значения для переполненной переменной. По превышении минимального допустимого значения для значения дня, месяца или года, произойдет присваивание максимального допустимого значения для переменной. Если координаты нажатия соответствуют координатам положения кнопки “ОК”, произойдет запись установленной даты и времени с последующим переходом на сцену главного меню. Если текущая сцена соответствует значению “1”, и нажатие соответствует координатам кнопки секундомера, таймера или настроек, то происходит переход в меню секундомера, таймера или настроек соответственно. Если текущая сцена соответствует значению “2”, и нажатие соответствует координатам кнопки “назад”, “старт”, “стоп” или “сброс”, то происходит переход в главное меню, запуск, остановка или сброс значения

секундомера соответственно. Если текущая сцена соответствует значению “3”, и нажатие соответствует координатам кнопки “старт”, “назад” или одной из кнопок настройки времени таймера, то происходит запуск с последующей отрисовкой меню таймера после запуска, возвращение в главное меню или настройка времени таймера. Если текущая сцена соответствует значению “4”, и нажатие соответствует координатам кнопки выбора даты, языка системы или кнопки назад, то происходит установка стиля даты, языка системы или переход в главное меню.

2.5 Функция выбора меню

Функция выбора меню представлена на рисунке 28. За то, какое меню будет отображаться на экране, отвечает переменная scene. Если переменная scene равна 0, то на экране будет отображаться стартовое меню. Если единице, то будет производится отрисовка главного меню.

2.6 Функция для получения текущего времени

Функция для получения текущего времени, представленная на рисунке 29, считывает значения по адресу 1 и 2 датчика часов реального времени DS1307, которые впоследствии записываются в переменные minutesи hour. По адресу 1 в датчике DS1307 находится текущее значение минут, а по адресу 2 в датчике DS1307 находится текущее значение часов.

2.7 Функция получения текущей даты

Функция для получения текущей даты, представленная на рисунке 30, считывает значения, по адресу 3, 4 и 5, датчика реального времени DS1307, которые впоследствии записываются в переменные weekDay, day и month. По адресу 3 в датчике DS1307 находится значение текущего дня недели, по адресу 4 в датчике DS1307 находится значение текущего дня, а по адресу 5 значение текущего месяца.

```
ISR (TIMER0_COMP_vect)
{
	ms_counter++;
	if(ms_counter==30)
	{
		ms_counter=0;

		PORTA=0b00000100;
		_delay_ms(2);
		ADMUX=0b00000000;
		ADCSRA=0b11000111;
		while((ADCSRA&0b01000000)!=0);
		x=ADCL|(ADCH<<8);

		PORTA=0b00001000;
		_delay_ms(2);
		ADMUX=0b00000001;
		ADCSRA=0b11000111 ;
		while((ADCSRA&0b01000000)!=0);
		y=ADCL|(ADCH<<8);

	}

	if (scene == 0) //Проверка нажатия кнопок для стартового меню
	{
		if (startScreenScene == 0) //Проверка нажатия кнопок на стартовом экране выбора языка
		{
			if((x>=120)&&(x<=340)&&(y>=380)&&(y<=480)) //Выбран русский язык
			{
				language = 0;
				startScreenScene = 1;
				isSceneChanged++;		//Флаг смены экрана
			}
			else if((x>=530)&&(x<=760)&&(y>=380)&&(y<=480)) //Выбран английский язык
			{
				language = 1;
				startScreenScene = 1;
				isSceneChanged++; //Флаг смены экрана
			}
		}
```

Рисунок 15 – Функция прерывания таймера-счетчика 0

```
		if (startScreenScene == 1)
		{
			if ((x>=80)&&(x<=130)&&(y>=680)&&(y<=730) && (isPressed == 0)) //Нажата кнопка плюс для часов
			{
				isPressed++;
				hour++;
				if(hour > 23) hour = 0;
			}

			else if ((x>=78)&&(x<=150)&&(y>=310)&&(y<=370) && (isPressed == 0))//Нажата кнопка минус для часов
			{
				isPressed++;
				hour--;
				if (hour < 0) hour = 23;
			}
```

Рисунок 16 – Функция прерывания таймера-счетчика 0

```
if ((x>=190)&&(x<=260)&&(y>=640)&&(y<=720) && (isPressed == 0)) //Нажата кнопка плюс для минут
{
        isPressed++;
        minutes++;
        if(minutes > 59) minutes = 0;
}
else if ((x>=185)&&(x<=275)&&(y>=320)&&(y<=390) && (isPressed == 0))//Нажата кнопка минус для минут
{
        isPressed++;
        minutes--;
        if (minutes < 0) minutes = 59;
}
```

Рисунок 17 – Функция прерывания таймера-счетчика 0

```
if ((x>=410)&&(x<=510)&&(y>=635)&&(y<=710) && (isPressed == 0)) //Нажата кнопка плюс для дня
{
        isPressed++;
        day++;
        if ((month == 1) || (month == 3) || (month == 5) || (month == 7) || (month == 8) || (month == 10) || (month == 12))
        {
                if (day > 31) day = 1;
        }
        else if( month == 2 && month % 4 == 0 && day > 29) day = 1;
        else if( month == 2 && month % 4 != 0 && day > 28) day = 1;
        else if(day > 30) day = 1;
}
```

Рисунок 18 – Функция прерывания таймера-счетчика 0

```
else if ((x>=420)&&(x<=520)&&(y>=300)&&(y<=360) && (isPressed == 0)) //Нажата кнопка минус для дня
{
        isPressed++;
        day--;
        if (day < 1)
        {
                if ((month == 1) || (month == 3) || (month == 5) || (month == 7) || (month == 8) || (month == 10) || (month == 12))
                {
                        day = 31;
                }
                else if( month == 2 && month % 4 == 0) day = 29;
                else if( month == 2 && month % 4 != 0) day = 28;
                else day = 30;
        }
}
if ((x>=550)&&(x<=650)&&(y>=615)&&(y<=690) && (isPressed == 0))//Нажата кнопка плюс для месяца
{
        isPressed++;
        month++;
        if(month > 12) month = 1;
        //Проверка коректности даты для выбранного месяца
        if( month == 2 && month % 4 == 0 && day > 29) day = 29;
        if( month == 2 && month % 4 != 0 && day > 28) day = 28;
        else if(((month == 4) || (month == 6) || (month == 9) || (month == 11)) && day > 30) day = 30;
}
```

Рисунок 19 – Функция прерывания таймера-счетчика 0

```
                        }
                        if ((x>=570)&&(x<=670)&&(y>=260)&&(y<=400) && (isPressed == 0))//Нажата кнопка минус для месяца
                        {
                                isPressed++;
                                month--;
                                if (month < 1) month = 12;
                                //Проверка коректности даты для выбранного месяца
                                if( month == 2 && month % 4 == 0 && day > 29) day = 29;
                                if( month == 2 && month % 4 != 0 && day > 28) day = 28;
                                else if(((month == 4) || (month == 6) || (month == 9) || (month == 11)) && day > 30) day = 30;
                        }
                        if ((x>=700)&&(x<=790)&&(y>=600)&&(y<=670) && (isPressed == 0))//Нажата кнопка плюс для года
                        {
                                isPressed++;
                                year++;
                                if (year > 99) year = 0;
                        }
                        if ((x>=730)&&(x<=815)&&(y>=300)&&(y<=360) && (isPressed == 0))//Нажата кнопка минус для года
                        {
                                isPressed++;
                                year--;
                                if (year < 0) year = 99;
                        }
                        if ((x>=505)&&(x<=556)&&(y>=132)&&(y<=215) && (isPressed == 0))//Нажата кнопка плюс дня недели
                        {
                                isPressed++;
                                weekDay++;
                                if(weekDay > 6) weekDay = 0;

                                LCD_GotoXY(7, 0);
                                for (int i; i<85;i++)
                                {
                                        LCD_GotoXY(7, i);
                                        LCD_Data(0);
                                }
                        }
```

Рисунок 20 – Функция прерывания таймера-счетчика 0

```
                        if ((x>=72)&&(x<=106)&&(y>=140)&&(y<=215) && (isPressed == 0)) //Нажата кнопка минус дня недели
                        {
                                isPressed++;
                                weekDay--;
                                if(weekDay < 0) weekDay = 6;

                                for (int i=6; i<76;i++)
                                {
                                        LCD_GotoXY(7, i);
                                        LCD_Data(0);
                                }
                        }
                        if ((x>=725)&&(x<=825)&&(y>=120)&&(y<=205)) //Нажата кнопка OK
                        {
                                //Запись установленных данных в датчик часов реального времени DS1307
                                write_DS1307(0x00, 0);
                                _delay_us(50);
                                write_DS1307(0x01, minutes);
                                _delay_us(50);
                                write_DS1307(0x02, hour);
                                _delay_us(50);
                                write_DS1307(0x03, weekDay);
                                _delay_us(50);
                                write_DS1307(0x04, day);
                                _delay_us(50);
                                write_DS1307(0x05, month);
                                _delay_ms(3);
                                //Переход на сцену главного меню
                                scene = 1;
                                isSceneChanged++; //Флаг смены экрана
                        }
                }
        }
```

Рисунок 21 – Функция прерывания таймера-счетчика 0

```
else if (scene == 1) //Проверка нажатия кнопок для главного меню
{
        if ((x>=725)&&(x<=800)&&(y>=670)&&(y<=730)) //Нажата кнопка настройки
        {
                scene = 4;
                isSceneChanged++; //Флаг смены экрана
        }
        if ((x>=100)&&(x<=440)&&(y>=190)&&(y<=305)) //Нажата кнопка секундомера
        {
                scene = 2;
                isSceneChanged++; //Флаг смены экрана
        }
        if ((x>=530)&&(x<=750)&&(y>=190)&&(y<=290)) //Нажата кнопка таймера
        {
                scene = 3;
                isSceneChanged++; //Флаг смены экрана
        }

}
```

Рисунок 22 – Функция прерывания таймера-счетчика 0

```
else if (scene == 2) //Проверка нажатия кнопок для секундомера
{
        if ((x>=75)&&(x<=210)&&(y>=718)&&(y<=790)) //Нажата кнопка назад
        {
                scene = 1;
                isSceneChanged++; //Флаг смены экрана
        }

        if ((x>=430)&&(x<=610)&&(y>=275)&&(y<=375)) isStartedSWT = 1;          //Нажата кнопка старт
        if ((x>=195)&&(x<=360)&&(y>=290)&&(y<=390)) isStartedSWT = 0;          //Нажата кнопка стоп
        if ((x>=305)&&(x<=510)&&(y>=120)&&(y<=220))                            //Нажата кнопка сброс
        {
                isStartedSWT = 0;
                stopwatchTime[0] = 0;
                stopwatchTime[1] = 0;
                stopwatchTime[2] = 0;
        }
}
```

Рисунок 23 – Функция прерывания таймера-счетчика 0

```
else if (scene == 3) //Проверка нажатия кнопок для таймера
{
        if (!isStartedTMR)
        {
                if ((x>=75)&&(x<=210)&&(y>=718)&&(y<=790)) //Нажата кнопка назад
                {
                        scene = 1;
                        isSceneChanged++; //Флаг смены экрана
                }


                else if ((x>=430)&&(x<=570)&&(y>=680)&&(y<=760) && (isPressed == 0)) //Нажата кнопка +1 секунда
                {
                        timerTime[0]++;
                        isPressed++;
                }
```

Рисунок 24 – Функция прерывания таймера-счетчика 0

```
                    else if ((x>=590)&&(x<=770)&&(y>=670)&&(y<=740) && (isPressed == 0)) //Нажата кнопка +10 секунд
                    {
                              timerTime[0]+=10;
                              isPressed++;
                    }
                    else if ((x>=430)&&(x<=580)&&(y>=605)&&(y<=700) && (isPressed == 0)) //Нажата кнопка +1 минута
                    {
                              timerTime[1]++;
                              isPressed++;
                    }
                    else if ((x>=605)&&(x<=775)&&(y>=580)&&(y<=690) && (isPressed == 0)) //Нажата кнопка +10 минут
                    {
                              timerTime[1]+=10;
                              isPressed++;
                    }
                    else if ((x>=450)&&(x<=590)&&(y>=525)&&(y<=620) && (isPressed == 0)) //Нажата кнопка +1 час
                    {
                              timerTime[2]++;
                              isPressed++;
                    }
                    else if ((x>=620)&&(x<=770)&&(y>=500)&&(y<=600) && (isPressed == 0)) //Нажата кнопка +5 часов
                    {
                              timerTime[2]+=5;
                              isPressed++;
                    }
                    else if ((x>=445)&&(x<=600)&&(y>=360)&&(y<=470) && (isPressed == 0)) //Нажата кнопка -1 секунда
                    {
                              timerTime[0]--;
                              isPressed++;
                    }
                    else if ((x>=600)&&(x<=790)&&(y>=350)&&(y<=450) && (isPressed == 0)) //Нажата кнопка -10 секунд
                    {
                              timerTime[0]-=10;
                              isPressed++;
                    }
```

Рисунок 25 – Функция прерывания таймера-счетчика 0

```
                    else if ((x>=450)&&(x<=600)&&(y>=190)&&(y<=290) && (isPressed == 0)) //Нажата кнопка -1 час
                    {
                              timerTime[2]--;
                              isPressed++;
                    }
                    else if ((x>=600)&&(x<=790)&&(y>=170)&&(y<=270) && (isPressed == 0)) //Нажата кнопка -5 часов
                    {
                              timerTime[2]-=5;
                              isPressed++;
                    }
                    else if ((x>=100)&&(x<=325)&&(y>=265)&&(y<=415)) //Нажата кнопка старт
                    {
                              setTMR = 0;
                              isStartedTMR = 1;
                              isSceneChanged++; //Флаг смены экрана
                    }
          }
          else if ((x>=300)&&(x<=550)&&(y>=190)&&(y<=330)) //Нажата кнопка сброс
          {
                    setTMR = 1;
                    isStartedTMR = 0;
                    isSceneChanged++; //Флаг смены экрана
          }

}
```

Рисунок 26 – Функция прерывания таймера-счетчика 0

```
        else if (scene == 4) //Проверка нажатия кнопок для настроек
        {
                if ((x>=75)&&(x<=210)&&(y>=718)&&(y<=790)) //Нажата кнопка назад
                {
                        scene = 1;
                        isSceneChanged++; //Флаг смены экрана
                }

                else if ((x>=155)&&(x<=360)&&(y>=475)&&(y<=535)) dataStyle = 0;//Нажата кнопка 1 мая
                else if ((x>=155)&&(x<=360)&&(y>=380)&&(y<475)) dataStyle = 1;  //Нажата кнопка 01.05
                else if ((x>=155)&&(x<=360)&&(y>=280)&&(y<380)) dataStyle = 2;  //Нажата кнопка 05.01
                else if ((x>=590)&&(x<=680)&&(y>=130)&&(y<=215)) language = 0;//Нажата кнопка РУ
                else if ((x>=710)&&(x<=825)&&(y>=130)&&(y<=215)) language = 1;//Нажата кнопка РУ
        }
        if ((x<70) && (y<70) && isPressed)
        {
                isPressed = 0;
        }
}
```

Рисунок 27 – Функция прерывания таймера-счетчика 0

```
void mainMenu()
{
        if (isSceneChanged) //Очистка экрана при смене сцены
        {
                LCD_Clear();
                isSceneChanged = 0;
        }
        if (scene == 0) //Стартовое меню
        {
                printStartMenu();
        }
        else if(scene == 1) //Главное меню
        {
                printMainMenu();
        }
        else if (scene == 2) //Меню секундомера
        {
                printStopwatchMenu();
        }
        else if(scene == 3) //Меню таймера
        {
                printTimerMenu();
        }
        else if(scene == 4) //Меню Настроек
        {
                printSettings();
        }
}
```

Рисунок 28 – Функция выбора меню

```
void getCurrentTime()
{
        minutes = read_DS1307(0x01);
        hour = read_DS1307(0x02);
}
```

Рисунок 29 – Функция получения текущего времени

```
void getCurrentDate()
{
        weekDay = read_DS1307(0x03);
        day = read_DS1307(0x04);
        month = read_DS1307(0x05);
}
```

Рисунок 30 – Функция получения текущей даты

2.8 Функция отправки данных дисплею

Для отправки данных дисплею используется функция LCD_Data, которая при вызове функции помещает, отправленные данные, в порт LCD_DATA_PORT, после чего в порте DIRW_PORT, биты DI и RW устанавливаются в 1 и 0 соответственно, что устанавливает режим приема данных и их записи. Полностью функция отправки данных показана на рисунке 31.

2.9 Функция отправки команд дисплею

Для отправки команд дисплею используется функция LCD_Data, которая при вызове функции помещает, отправленную команду, в порт LCD_DATA_PORT, после чего в порте DIRW_PORT, биты DI и RW устанавливаются в 0, что устанавливает режим приема команд и их записи. Полностью функция отправки команд показана на рисунке 32.

```
void LCD_Data(char Data)
{
      RSRW_PORT |=  (1 << RS);    //Выставляем бит RS в 1 для формирвоания состояния принятия данных
      RSRW_PORT &= ~(1 << RW);    //Выставляем бит RW в 0 для формирования состояния записи в жки
      LCD_DATA_PORT = Data;       //Помещаем данные в порт данных жки
      trigger();
}
```

Рисунок 31 – Функция отправки данных дисплею

```
void LCD_Command(char Command)
{
	RSRW_PORT &= ~(1 << RS);	//Выставляем в регистре RS бит 0 для состояния принятия команды
	RSRW_PORT &= ~(1 << RW);	//Формируем состояние записи выставив бит RW в 1
	LCD_DATA_PORT = Command;	//Помещаем команду в порт для данных жки
	trigger();			//Выставляем бит EN в 1 для формирования состояния начала отправки команды
}
```

Рисунок 32 – Функция отправки команд

2.10 Функция инициализации жидкокристаллического дисплея

Для инициализации дисплея используется функция LCD_init. Порт LCD_DATA_DDR, отвечающий за направление данных в порте PORTC микроконтроллера ATmega32, устанавливается на выход. Порты RSRW_DDR и ERST_DDR, отвечающие за направление данных в портах PORTA и PORTD соответственно, устанавливаются на выход. Порт CS_DDR, отвечающий за направление данных в порте PORTC, устанавливается на выход. Дальше дисплею отправляется команда LCD_Command(0x3F), которая указывает на то, что данные на дисплее должны отображаться. После этого дисплею посылаются команды LCD_Command(0x40) и LCD_Command(0xB8), которые выбирают положение по оси Y и оси X соответственно равными 0.

2.11 Функция очистки дисплея

Для очистки данных на экране используется функция LCD_Clear, показанная на рисунке 34. Сначала через порт CS_PORT происходит выбор левого контроллера экрана, отвечающего за работу левой половины дисплея. После чего используются два цикла для последовательной очистки данных в дисплее. Затем тоже самое делается с правым контроллером для очистки данных в правой части экрана.

2.12 Функция выбора положения курсора на экране

Для выбора положения на экране используется функция LCD_GotoXY,

представленная на рисунке 35. Сначала осуществляется проверка по выбранному положению на оси Y. Если выбранное значение на оси меньше 64, то выбирается левый контроллер, иначе выбирается правый контроллер. После чего осуществляется переход в выбранное положение по оси X и оси Y командами LCD_Command(0x8B + x) и LCD_Command(0x40 + y) соответственно.

```
void LCD_Init()
{
	LCD_DATA_DDR = 0xFF;
	RSRW_DDR |= (1<<RS)|(1<<RW);
	ERST_DDR |= (1<<EN)|(1<<RST);
	CS_DDR |= (1<<CS1)|(1<<CS2);

	CS_PORT &= ~((1 << CS1) | (1 << CS2)); //включаем оба чип
	ERST_PORT |= (1 << RST);
	_delay_us(20);
	LCD_Command(0x3E);		//Отправка команды ВЫКЛ
	LCD_Command(0x40);		//Установка положения по оси Y=0
	LCD_Command(0xB8);		//Установка положения по оси X=0
	LCD_Command(0xC0);		//Установка положения по оси Z=0
	LCD_Command(0x3F);		//Отправка команды ВКЛ
}
```

Рисунок 33 – Функция инициализации дисплея

```
void LCD_Clear()
{
	//Выбираем первую половину дисплея
	CS_PORT &= ~(1 << CS1);
	CS_PORT |= (1 << CS2);
	for(int i = 0; i < 8; i++)
	{

		LCD_Command((0xB8) + i);
		for(int j = 0; j < 64; j++)
		{
			LCD_Data(0); //Очищаем каждый пиксель первой половины дисплея
		}
	}
	//Выбираем вторую половину дисплея
	CS_PORT |= (1 << CS1);
	CS_PORT &= ~(1 << CS2);
	for(int i = 0; i < 8; i++)
	{

		LCD_Command((0xB8) + i);
		for(int j = 0; j < 64; j++)
		{
			LCD_Data(0);//Очищаем каждый пиксель второй половины дисплея
		}
	}
	LCD_Command(0x40);
	LCD_Command(0xB8);
}
```

Рисунок 34 – Функция очистки данных экрана

```
void LCD_GotoXY(uint8_t x, uint8_t y)// Установить курсор в позицию x, y
{
        if(y<64)//Опредления контроллера на котором находится наши координаты
        {
                CS_PORT &= ~(1<<CS1);
                CS_PORT |= (1<<CS2);
        }
        else
        {
                CS_PORT &= ~(1<<CS2);
                CS_PORT |= 1<<CS1;
                y=y-64;
        }
        LCD_Command(0xB8+x);
        LCD_Command(0x40+y);
        LCD_Command(0x3F);
        LCD_Command(0xC0);
}
```

Рисунок 35 – Функция выбора положение курсора на экране

2.13 Функция вывода байта на экран

Для вывода байта на экран используется функция LCD_drawByte, показанная на рисунке 36. При вызове функции производится переход в выбранные координаты и устанавливается режим приема данных и их записи. Дальше в порт данных дисплея записываются переданный байт данных.

```
void LCD_drawByte(uint8_t x, uint8_t y, unsigned char byte)//ставит пиксель в координаты x  и y
{
      LCD_GotoXY(x,y);
      LCD_Data(byte);
}
```

Рисунок 36 – Функция вывода байта на экран

2.14 Функция вывода символа на экран

Для вывода символа на экран используется функция LCD_Char, представленная на рисунке 37. При вызове функции производится выбор соответствующего номера символа. После чего производится последовательная запись 5 байтов данных, которые представляют собой выбранный символ.

```
void LCD_Char(uint8_t  x1, uint8_t y1, char code){
        if((code >= 0x20) && (code < 0x80)) code -= 32;
        else  code -= 96;

        for(uint8_t i = 0; i < 5; i++)
        {
                LCD_drawByte(x1, y1 + i, symbol[code][i]);
        }
}
```

Рисунок 37 – Функция вывода символа на экран

2.15 Функция записи строки

Для записи строки, используется функция LCD_String, представленная на рисунке 38. При выводе строки используется последовательный вывод каждого символа отдельно с помощью команды LCD_Char.

```
void LCD_String(uint8_t x1, uint8_t y1, char *string) {
        while(*string) {
                LCD_Char(x1, y1, *string++);//Посимвольное отображение каждого символа строки
                y1+=6;
        }
}
```

Рисунок 38 – Функция выводы строки на дисплей

2.16 Функция инициализации часов реального времени

Для инициализации часов реального времени используется функция init_DS1307, представленная на рисунке 39. Сначала происходит настройка частоты SCL равной 100 кГц.

Расчет значения частоты SCL рассчитывается по формуле 3:

$$F = \frac{Fcpu\ freq}{16+2*TWBR*2^{TWSP}}. \quad (3)$$

Для того, чтобы добиться частоты 100 кГц, в регистр TWBR, записывается

число 32, а в регистре TWSR биты TWPS1 и TWPS0 устанавливаются в 0, чтобы установить число TWSP из формулы 3 в значение 0.

```
void init_DS1307(void)
{
        TWBR = 32; //При частоте 1 МГц
        TWSR = (0 << TWPS1)|(0 << TWPS0);
}
```

Рисунок 39 – Функция инициализации часов реального времени

2.17 Функция чтения данных из датчика часов реального времени

Для чтения данных из часов реального времени используется функция read_DS1307, представленная на рисунке 40. После вызова функции происходит формирование режима старт, путем установки битов TWINT, TWSTA и TWEN в значение единицы для того, чтобы начать работу с устройствами по шине TWI. После этого по шине передается байт, содержащий адрес часов реального времени. Когда устройство выбрано, формируется повторный старт, чтобы перейти в режим получения данных от устройства. После чего происходит чтение данных из адреса, переданного при использовании функции.

2.18 Функция записи данных в датчик часов реального времени

Для записи данных в часы реального времени используется функция write_DS1307, представленная на рисунке 41. После вызова функции происходит формирование режима старт, путем установки битов TWINT, TWSTA и TWEN в положение 1 для того, чтобы начать работу с устройствами по шине TWI. После этого по шине передается байт, содержащий адрес часов реального времени. Когда устройство выбрано, ему отправляется адрес, куда будут записываться данные. После чего происходит запись данных в адрес.

2.19 Функция вывода стартового меню

Функция для вывода стартового меню, представленная на рисунке 42 и 43 используется единожды за включение прибора, чтобы настроить дату, время и язык системы. Первоначально производится отрисовка меню выбора языка системы. После выбора языка, производится отрисовка меню настройки времени и даты.

```
uint8_t read_DS1307(uint8_t addr) //Передаем адрес регистра
{
        uint8_t time;
        TWCR |= (1<<TWEN);
        while(!(TWCR&(1<<TWEN)));
        //формируем состояние СТАРТ
        TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
        while (!(TWCR & (1<<TWINT)));
        TWDR = 0xd0; //передаем адрес и бит означающий режим записи
        TWCR = (1<<TWINT)|(1<<TWEN);
        while (!(TWCR & (1<<TWINT)));
        TWDR = addr; // Устанавливаем необходимый адрес для чтения данных
        TWCR = (1<<TWINT)|(1<<TWEN);
        while (!(TWCR & (1<<TWINT)));
        //Формируем повторно сигнал старта, чтобы считать данные с необходимого регистра
        TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
        while (!(TWCR & (1<<TWINT)));
        TWDR = 0xd1; //Передаем адрес и бит, означающий режим чтения
        TWCR = (1<<TWINT)|(1<<TWEN);
        while (!(TWCR & (1<<TWINT)));
        //считываем данные
        TWCR = (1<<TWINT)|(1<<TWEN);
        while(!(TWCR & (1<<TWINT)));
        time = TWDR;
        time = (((time & 0xF0) >> 4)*10)+(time & 0x0F);
        //формируем состояние СТОП
        TWCR = (1<<TWINT)|(1<<TWSTO)|(1<<TWEN);
        while(TWCR&(1<<TWSTO));
        TWCR &= (1<<TWEN); //Отключаем шину TWI, для корректной работы экрана с портом C
        _delay_us(50);
        TWCR &= (1<<TWINT);
        return time;
}
```

Рисунок 40 – Функция чтения данных из датчика часов реального времени

```
void write_DS1307 (uint8_t reg, uint8_t time)
{
	TWCR |= (1<<TWEN);
	while(!(TWCR&(1<<TWEN)));
	TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);//формируем состояние СТАРТ
	while (!(TWCR & (1<<TWINT)));
	TWDR = 0xd0; //передаем адрес и бит означающий режим записи
	TWCR = (1<<TWINT)|(1<<TWEN);
	while (!(TWCR & (1<<TWINT)));
	TWDR = reg;// Устанавливаем необходимый адрес для записи данных
	TWCR = (1<<TWINT)|(1<<TWEN);
	while (!(TWCR & (1<<TWINT)));
	time = ((time/10)<<4) + time%10;
	TWDR = time; //Записываем данные
	TWCR = (1<<TWINT)|(1<<TWEN);
	while(!(TWCR & (1<<TWINT)));
	TWCR = (1<<TWINT)|(1<<TWSTO)|(1<<TWEN);
	while(TWCR&(1<<TWSTO));
	TWCR &= (1<<TWEN); //Отключаем шину TWI, для корректной работы экрана с портом С
	_delay_us(50);
	TWCR &= (1<<TWINT);
}
```

Рисунок 41 – Функция записи данных в датчик часов реального времени

```
void printStartMenu()
{
	//Отрисовка стартового меню до выбора языка
	if (startScreenScene == 0)
	{
		LCD_String(4, 10, "Русский");
		LCD_String(4, 78, "English");
	}
	//Отрисовка стартового меню после выбора языка
	if (startScreenScene == 1)
	{
		//Отображение кнопок прибавления и убавление времени для настройки времени
		LCD_Char(1,4, '+');
		LCD_Char(5,4, '-');
		LCD_Char(1,26, '+');
		LCD_Char(5,26, '-');
		LCD_Char(3,0, hour/10 + '0');
		LCD_Char(3,8, hour%10 + '0');
		LCD_Char(3,16, ':');
		LCD_Char(3,24, minutes/10 + '0');
		LCD_Char(3,32, minutes%10 + '0');

		//Отображение кнопок прибавления и убавления для настройки даты
		LCD_Char(1,67, '+');
		LCD_Char(5,67, '-');
		LCD_Char(1,91, '+');
		LCD_Char(5,91, '-');
```

Рисунок 42 – Функция отрисовки стартового меню с настройкой языка

```
                LCD_Char(5,91, '-');
                LCD_Char(1, 115, '+');
                LCD_Char(5,115, '-');
                LCD_Char(3,63, day/10 + '0');
                LCD_Char(3,71, day%10 + '0');
                LCD_Char(3,79, '.');
                LCD_Char(3,87, month/10 + '0');
                LCD_Char(3,95, month%10 + '0');
                LCD_Char(3,103, '.');
                LCD_Char(3,111, year/10 + '0');
                LCD_Char(3,119, year%10 + '0');

                //Отображение дня недели и кнопок смены дня недели на русском языке
                if (!language)
                {
                        LCD_Char(7, 0, '-');
                        LCD_String(7,6 + (70 - strlen(DaysRU[weekDay])*6)/2, DaysRU[weekDay]);
                        LCD_Char(7, 76, '+');
                }
                //Отображение дня недели и кнопок смены дня недели на английском языке
                else
                {
                        LCD_Char(7, 0, '-');
                        LCD_String(7,6 + (70 - strlen(DaysENG[weekDay])*6)/2, DaysENG[weekDay]);
                        LCD_Char(7, 76, '+');
                }
                //Отображение кнопки окончания стартовых настроек
                LCD_String(7, 114, "OK");
        }

}
```

Рисунок 43 – Функция отрисовки стартового меню с настройкой даты и времени

2.20 Функция вывода главного меню

Для вывода главного меню используется функция printMainMenu, представленная на рисунках 44, 45 и 46. При вызове функции, сначала выводится время, после чего в зависимости от того, какой выбран стиль даты, выводится различный вид текущей даты. После того как напечатана дата, в зависимости от выбранного языка печатается день недели и месяц на выбранном языке. Дальше выводятся кнопки “Секундомер”, “Таймер” и кнопка настроек в соответствии с выбранным языком.

2.21 Функция вывода меню секундомера

Для вывода меню секундомера используется функция printStopwatchMenu, представленная на рисунке 47 и 48. При вызове функции, происходит проверка корректности времени, которое будет выводиться на экран. После проверки,

производится вывод времени на экран. Далее производится вывод кнопок “Старт”, “Стоп” в соответствии с выбранным языком. Если секундомер запущен, то к кнопкам “Старт” и “Стоп” добавляется вывод кнопки “Сброс”.

```
void printMainMenu()
{
	//Вывод времени
	LCD_Char(3,42, hour/10 + '0');
	LCD_Char(3,51, hour%10 + '0');
	LCD_Char(3,60, ':');
	LCD_Char(3,68, minutes/10 + '0');
	LCD_Char(3,76, minutes%10 + '0');

	//Вывод соответствующего стиля отображения даты в соответствии с выбранным языком
	if(dataStyle == 0)
	{
		if(!language)
		{
			LCD_Char(0,0, day/10+'0');
			LCD_Char(0,6, day%10+'0');
			LCD_String(0,12, monthsRU[month-1]);
			LCD_String(1,0,DaysRU[weekDay-1]);
		}
```

Рисунок 44 – Функция отрисовки главного меню

```
		else
		{
			LCD_String(0,0, monthsENG[month - 1]);
			LCD_Char(0,strlen(monthsENG[month-1])*6+3, day/10+'0');
			LCD_Char(0,strlen(monthsENG[month-1])*6 + 9, day%10+'0');
			LCD_String(1,0,DaysENG[weekDay-1]);
			if ((day%10 == 1) && (day != 11))
			{
				LCD_Char(0, strlen(monthsENG[month-1])*6 + 15, 's');
				LCD_Char(0, strlen(monthsENG[month-1])*6 + 21, 't');
			}
			else if ((day%10 == 2) && (day != 12))
			{
				LCD_Char(0, strlen(monthsENG[month-1])*6 + 15, 'n');
				LCD_Char(0, strlen(monthsENG[month-1])*6 + 21, 'd');
			}
			else if((day%10 == 3) && (day != 13))
			{
				LCD_Char(0, strlen(monthsENG[month-1])*6 + 15, 'r');
				LCD_Char(0, strlen(monthsENG[month-1])*6 + 21, 'd');
			}
			else
			{
				LCD_Char(0, strlen(monthsENG[month-1])*6 + 15, 't');
				LCD_Char(0, strlen(monthsENG[month-1])*6 + 21, 'h');
			}
		}
	}
```

Рисунок 45 – Функция отрисовки главного меню

```
        else if(dataStyle == 1)
        {
                LCD_Char(1,0, day/10+'0');
                LCD_Char(1,6, day%10+'0');
                LCD_Char(1,12,'.');
                LCD_Char(1,18, month/10+'0');
                LCD_Char(1,24, month%10+'0');
                if(!language)       LCD_String(0,0,DaysRU[weekDay-1]);
                else LCD_String(0,0,DaysENG[weekDay-1]);
        }
        else if (dataStyle == 2)
        {
                LCD_Char(1,0, month/10+'0');
                LCD_Char(1,6, month%10+'0');
                LCD_Char(1,12,'.');
                LCD_Char(1,18, day/10+'0');
                LCD_Char(1,24, day%10+'0');
                if(!language)       LCD_String(0,0,DaysRU[weekDay-1]);
                else LCD_String(0,0,DaysENG[weekDay-1]);
        }
        //Вывод кнопок Настроек секундомера и таймера в соответствии я зыком
        if (!language)
        {
                LCD_String(6,8,"Секундомер"); //Вывод кнопки Секундомера
                LCD_String(6,80,"Таймер"); //Вывод кнопки Таймера
        }
        else
        {
                LCD_String(6,8,"Stopwatch"); //Вывод кнопки Секундомера
                LCD_String(6,80,"Timer"); //Вывод кнопки Таймера
        }
        for (int i = 0; i<10; i++)
        {
                LCD_drawByte(0, 117+i, settingsImage[i]);
        }
}
```

Рисунок 46 – Функция отрисовки главного меню

```
void printStopwatchMenu()
{
        if (stopwatchTime[0] == 60)
        {
                stopwatchTime[0] = 0;
                stopwatchTime[1]++;
                if(stopwatchTime[1] == 60)
                {
                        stopwatchTime[1] = 0;
                        stopwatchTime[2]++;
                }
        }
        LCD_Char(3,75,stopwatchTime[0]/10+'0');
        LCD_Char(3,82,stopwatchTime[0]%10+'0');
        LCD_Char(3,68,':');
        LCD_Char(3,54,stopwatchTime[1]/10+'0');
        LCD_Char(3,61,stopwatchTime[1]%10+'0');
        LCD_Char(3,33,stopwatchTime[2]/10+'0');
        LCD_Char(3,40,stopwatchTime[2]%10+'0');
        LCD_Char(3,47,':');
```

Рисунок 47 – Функция вывода меню секундомера

```
        if (!language)
        {
                LCD_String(0,0,"назад");
                LCD_String(5,27,"Стоп");
                LCD_String(5,64,"Старт");
                if (isStartedSWT)
                {
                        LCD_String(6,45, "Сброс");
                }
        }
        else
        {
                LCD_String(0,0,"back");
                LCD_String(5,24,"Stop");
                LCD_String(5,64,"Start");
                if (isStartedSWT)
                {
                        LCD_String(6,45, "Reset");
                }
        }

}
```

Рисунок 48 – Функция вывода меню секундомера

2.22 Функция вывода меню таймера

Для вывода меню таймера используется функция printTimerMenu, представленная на рисунках 49, 50, 51 и 52. При вызове функция первоначально производится проверка корректности времени, выводимого на экран. После чего осуществляется проверка для остановки времени таймер, которая выполняется при условии, что таймер запущен и время таймера равно нулю. После этой проверки идет проверка, отвечающая за отображение меню таймера. Если идет идёт настройка таймера, то переменная setTMR равна единице и производится вывод времени таймер и кнопок для точной настройки таймера. Если переменная setTMR равна нулю, то осуществляется вывод меню таймера без кнопок, для настройки времени.

2.23 Функция вывода меню настроек

Для отображения меню настроек используется функция printSettings, представленная на рисунках 53 и 54. Вывод соответствует выбранному языку.

```
void printTimerMenu()
{
	//Проверка корректности времени таймера
	if (timerTime[0] == -1)
	{
		timerTime[0] = 59;
		timerTime[1]--;
		if(timerTime[1] == -1)
		{
			timerTime[1] = 59;
			timerTime[2]--;
		}
	}

	//Проверка условия для остановки таймера
	if (isStartedTMR && (timerTime[2] == 0) && (timerTime[1] == 0) && (timerTime[0] == 0))
	{
		LCD_Clear();
		isStartedTMR = 0;
		timerTime[0] = 0;
		timerTime[1] = 0;
		timerTime[2] = 0;
	}

	//Отрисовка меню таймера
	if (setTMR)
	{
```

Рисунок 49 – Функция отрисовки меню таймера

```
		if (!language)
		{
			LCD_String(0,0,"Назад");
			LCD_Char(3,2, timerTime[2]/10+'0');
			LCD_Char(3,9,timerTime[2]%10+'0');
			LCD_Char(3,16,':');
			LCD_Char(3,23, timerTime[1]/10+'0');
			LCD_Char(3,30,timerTime[1]%10+'0');
			LCD_Char(3,37,':');
			LCD_Char(3,44, timerTime[0]/10+'0');
			LCD_Char(3,51,timerTime[0]%10+'0');

			LCD_String(0,69,"+1с");
			LCD_String(0,95,"+10с");
			LCD_String(1,69,"+1м");
			LCD_String(1,95,"+10м");
			LCD_String(2,69,"+1ч");
			LCD_String(2,101,"+5ч");

			LCD_String(4,69,"-1с");
			LCD_String(4,95,"-10с");
			LCD_String(5,69,"-1м");
			LCD_String(5,95,"-10м");
			LCD_String(6,69,"-1ч");
			LCD_String(6,101,"-5ч");

			LCD_String(5,13,"Старт");
		}
```

Рисунок 50 – Функция отрисовки меню таймера

```
                else
                {
                        LCD_String(0,0,"back");
                        LCD_Char(3,2, timerTime[2]/10+'0');
                        LCD_Char(3,9,timerTime[2]%10+'0');
                        LCD_Char(3,16,':');
                        LCD_Char(3,23, timerTime[1]/10+'0');
                        LCD_Char(3,30,timerTime[1]%10+'0');
                        LCD_Char(3,37,':');
                        LCD_Char(3,44, timerTime[0]/10+'0');
                        LCD_Char(3,51,timerTime[0]%10+'0');

                        LCD_String(0,69,"+1s");
                        LCD_String(0,95,"+10s");
                        LCD_String(1,69,"+1m");
                        LCD_String(1,95,"+10m");
                        LCD_String(2,69,"+1h");
                        LCD_String(2,101,"+5h");

                        LCD_String(4,69,"-1s");
                        LCD_String(4,95,"-10s");
                        LCD_String(5,69,"-1m");
```

Рисунок 51 – Функция отрисовки меню таймера

```
                        LCD_String(5,95,"-10m");
                        LCD_String(6,69,"-1h");
                        LCD_String(6,101,"-5h");

                        LCD_String(5,13,"Start");
                }
        }
        else
        {
                LCD_Char(3,75,timerTime[0]/10+'0');
                LCD_Char(3,82,timerTime[0]%10+'0');
                LCD_Char(3,68,':');
                LCD_Char(3,54,timerTime[1]/10+'0');
                LCD_Char(3,61,timerTime[1]%10+'0');
                LCD_Char(3,33,timerTime[2]/10+'0');
                LCD_Char(3,40,timerTime[2]%10+'0');
                LCD_Char(3,47,':');

                if (!language) LCD_String(6,45,"Сброс");
                else LCD_String(6,45,"Reset");
        }
}
```

Рисунок 52 – Функция отрисовки меню таймера

```
void printSettings()
{
        if (!language)
        {
                LCD_String(0,0,"назад");

                LCD_String(2,0,"Стиль даты:");
                LCD_String(3,10, "- 1 Мая");
                LCD_String(4,10, "- 01.05");
                LCD_String(5,10, "- 05.01");

                LCD_String(7,0, "Язык системы");
                LCD_String(7, 92, "РУ");
                LCD_String(7, 110, "ENG");
        }
```

Рисунок 53 – Функция вывода меню настроек

```
        else
        {
                LCD_String(0,0,"back");

                LCD_String(2,0,"Data style:");
                LCD_String(3,10, "- May 1st");
                LCD_String(4,10, "- 01.05");
                LCD_String(5,10, "- 05.01");

                LCD_String(7,0, "System");
                LCD_String(7, 92, "РУ");
                LCD_String(7, 110, "ENG");
        }
}
```

Рисунок 54 – Функция вывода меню настроек

3 ОТЛАДКА ПРОГРАММЫ

Microchip Studio был использован для написания и правки кода программы. Microchip Studio [5] – основанная на базе Visual Studio бесплатная интегрированная среда разработки, содержащая компилятор GNUC/C++, для создания приложения для микроконтроллеров семейства AVR и ARM. Microchip Studio показан на рисунке 55.

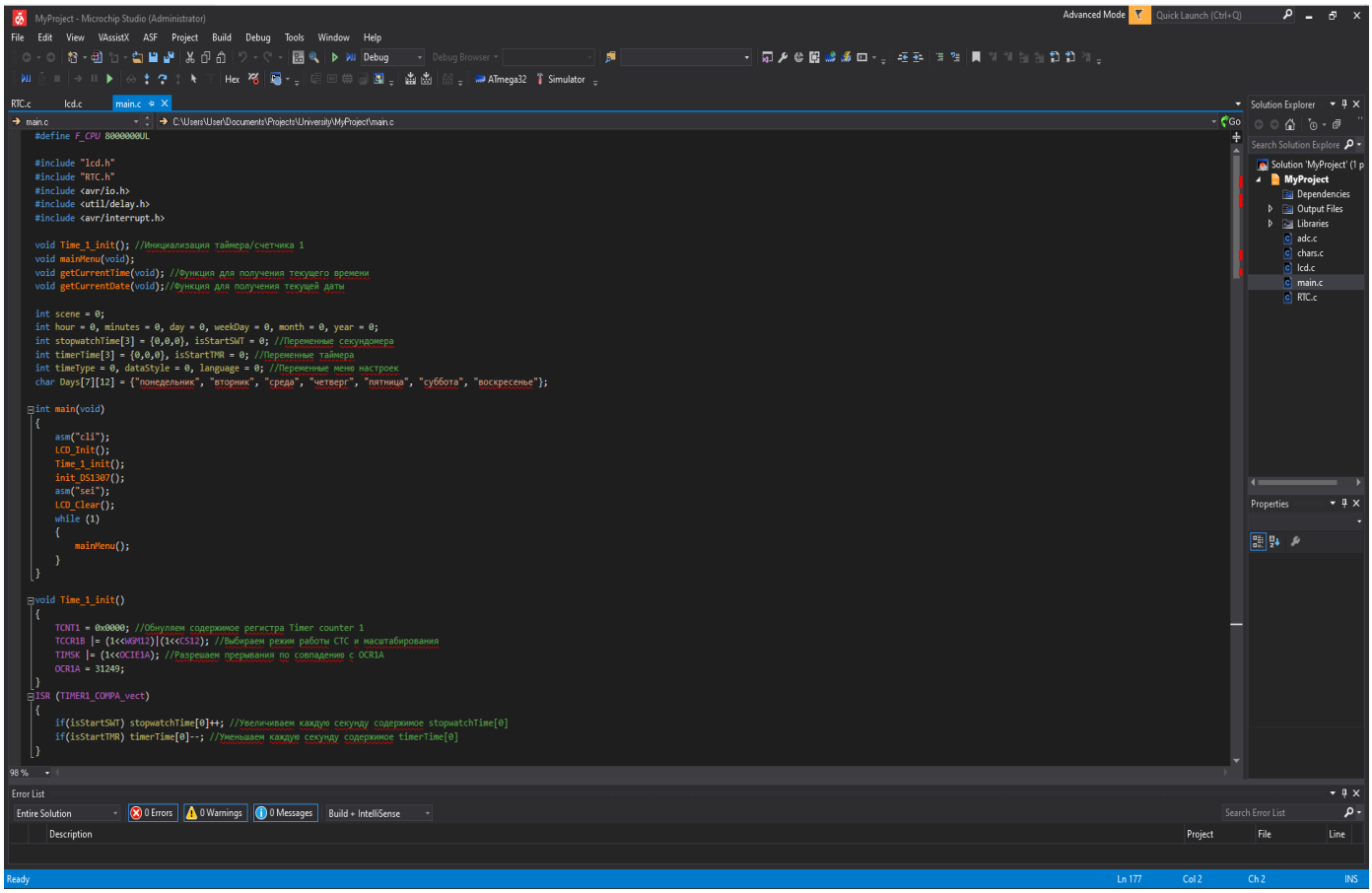

Рисунок 55 – интегрированная среда разработки Microchip Studio

AVRflash– программа созданная компанией microElektronika. Использовалась для загрузки исполняемой программы в отладочную платуEasyAVR V7 Development System фирмы microElektronika.Внешний вид панели управления AVRflash показан на рисунке 56.

Рисунок 56 – Внешний вид панели управления AVRFlash

Для отладки отрисовки меню прибора, был использован Proteus [6]. Proteus — это пакет программ для автоматизированного проектирования электронных схем. Пакет представляет собой систему схемотехнического моделирования, базирующуюся на основе моделей электронных компонентов. Есть возможность просмотреть макет печатной платы компонента отдельно при выборе компонента. Proteus опережает в моделировании схем, содержащих микроконтроллеры, где мы можем смоделировать схему, загрузив шестнадцатеричный код в микроконтроллер. Отличительной чертой пакета Proteus Professional является возможность моделирования работы программируемых устройств: микроконтроллеров, микропроцессоров, DSP и прочее. Proteus Professional может симулировать работу следующих микроконтроллеров: 8051, ARM7, AVR, Motorola, PIC, Basic Stamp. Библиотека компонентов содержит справочные данные.

Стартовое меню прибора до выбора языка, отрисованное в Proteus и на реальном экране ME-GLCD 128x64, представлено на рисунке 57. Стартовое меню, после выбора языка системы, отрисованное в Proteus и на экране ME-GLCD

представлено на рисунке 58. Главное меню прибора отрисованное в Proteus и на экране ME-GLCD представлено на рисунке 59. Меню секундомера до запуска, отрисованное в Proteus и на экране ME-GLCD представлено на рисунке 60. Меню секундомера после запуска, отрисованное в Proteus и на экране ME-GLCD представлено на рисунке 61. Меню таймера до запуска, отрисованное в Proteus и на экране ME-GLCD представлено на рисунке 62. Меню таймера после запуска, отрисованное в Proteus и на экране ME-GLCD представлено на рисунке 63. Меню настроек, отрисованное в Proteus и на экране ME-GLCD представлено на рисунке 64. Схема подключения микроконтроллера ATmega32, жидкокристаллического дисплея ME-GLCD 128x64 и датчика часов реального времени DS1307 представлена на рисунке 65

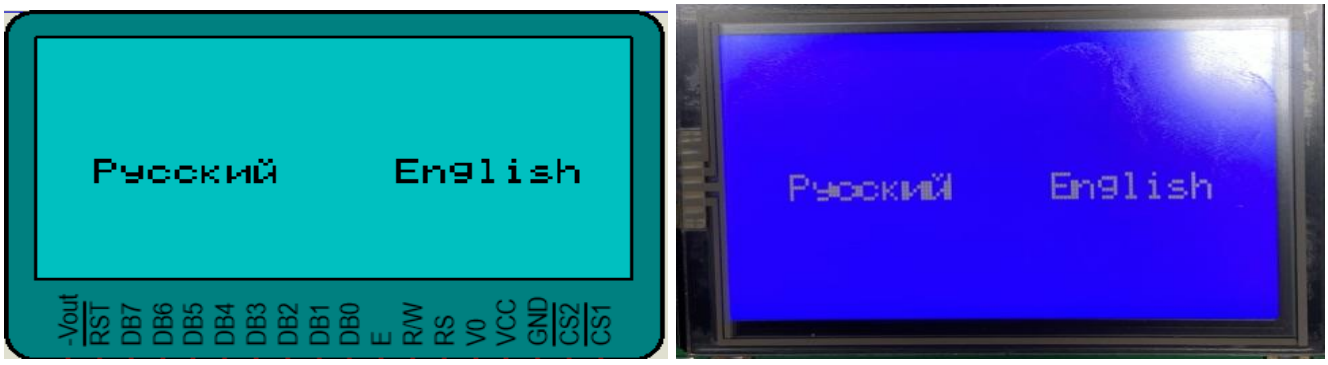

Рисунок 57 – Стартовое меню, отрисованное в Proteus и на экране ME-GLCD 128x64

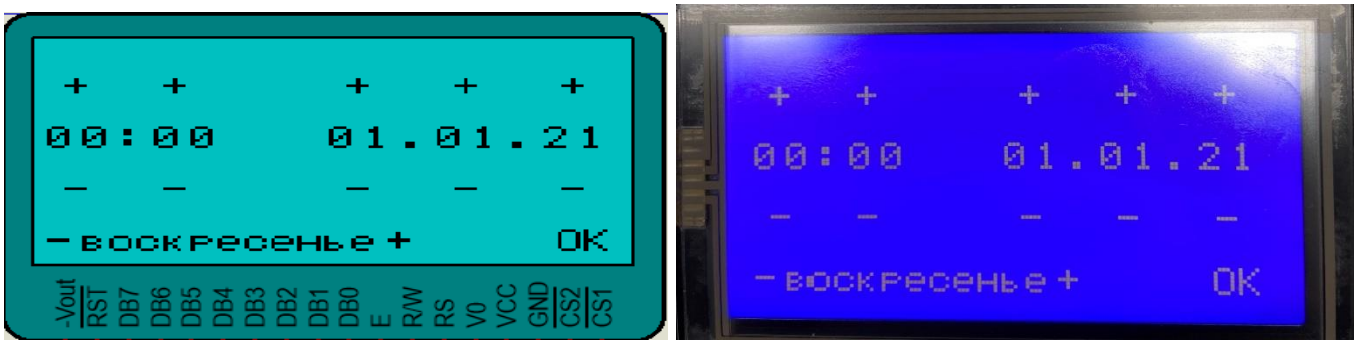

Рисунок 58 – Стартовое меню настройки даты и времени, отрисованное в Proteus и на экране ME-GLCD 128x64

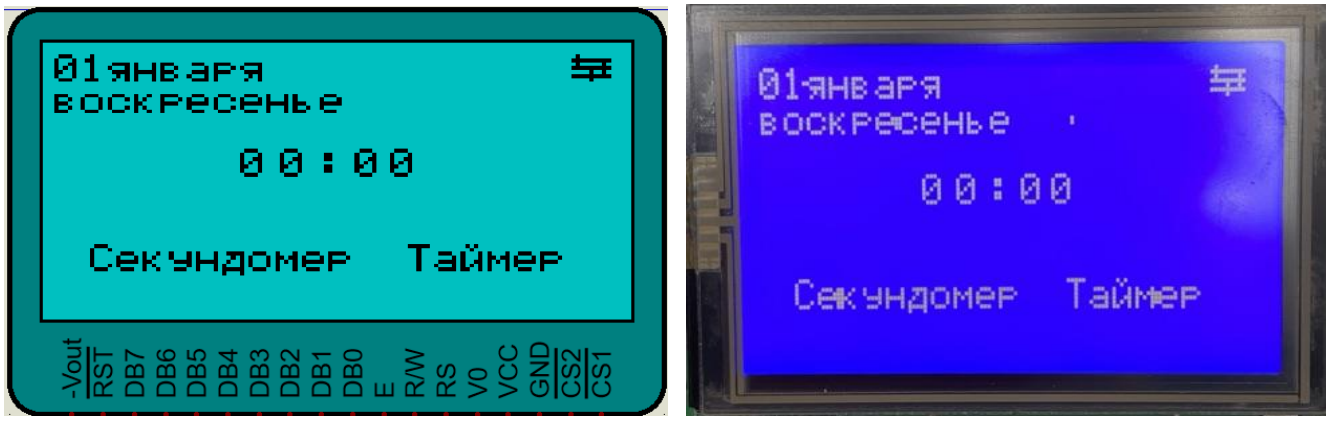

Рисунок 59 – Главное меню прибора, отрисованное в Proteus и на экране ME-GLCD

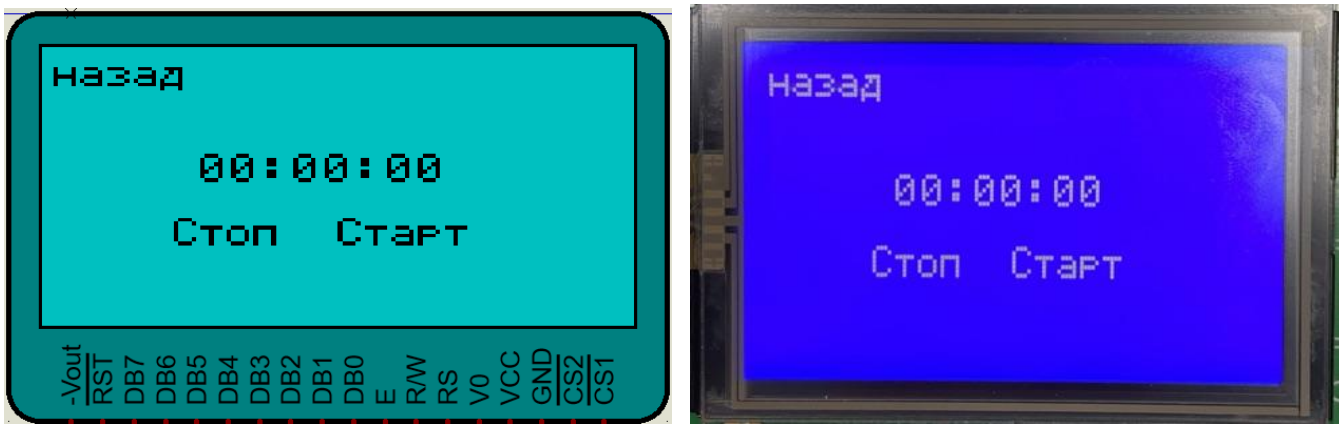

Рисунок 60 – Меню секундомера до запуска отрисованное в Proteus и на ME-GLCD

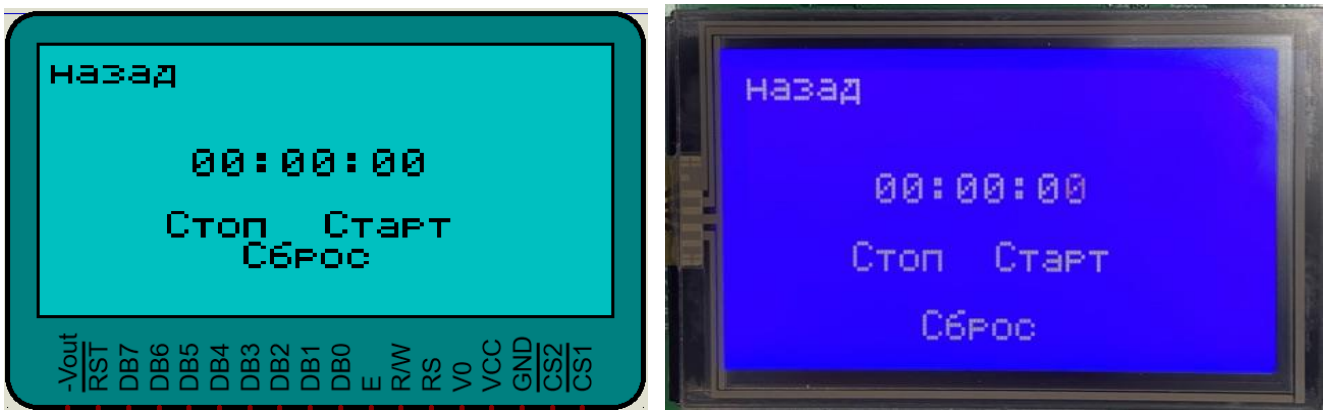

Рисунок 61 – Меню секундомера после запуска, отрисованное в Proteus и на экране ME-GLCD

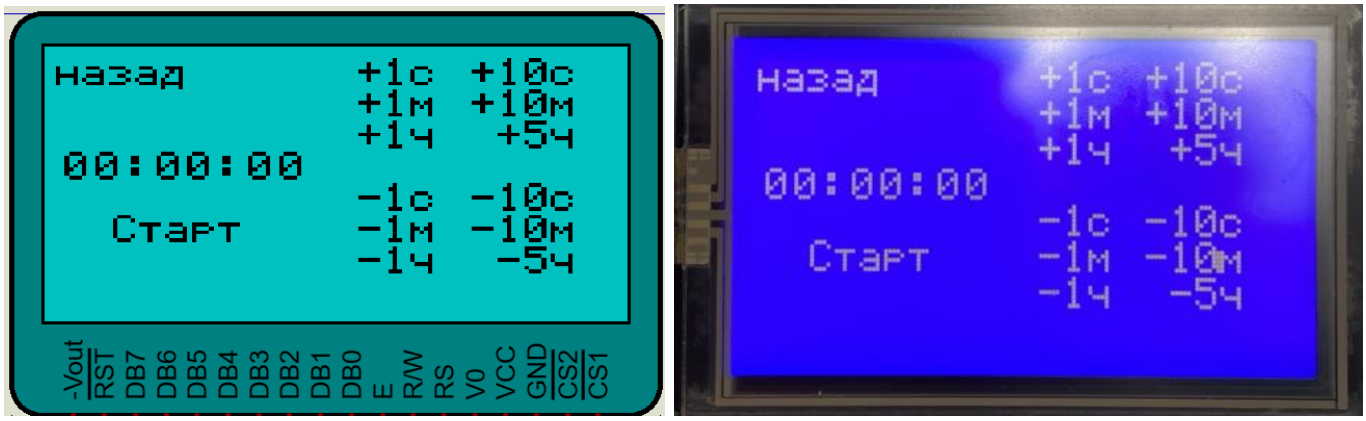

Рисунок 62 – Меню таймера до запуска, отрисованное в Proteus и на ME-GLCD

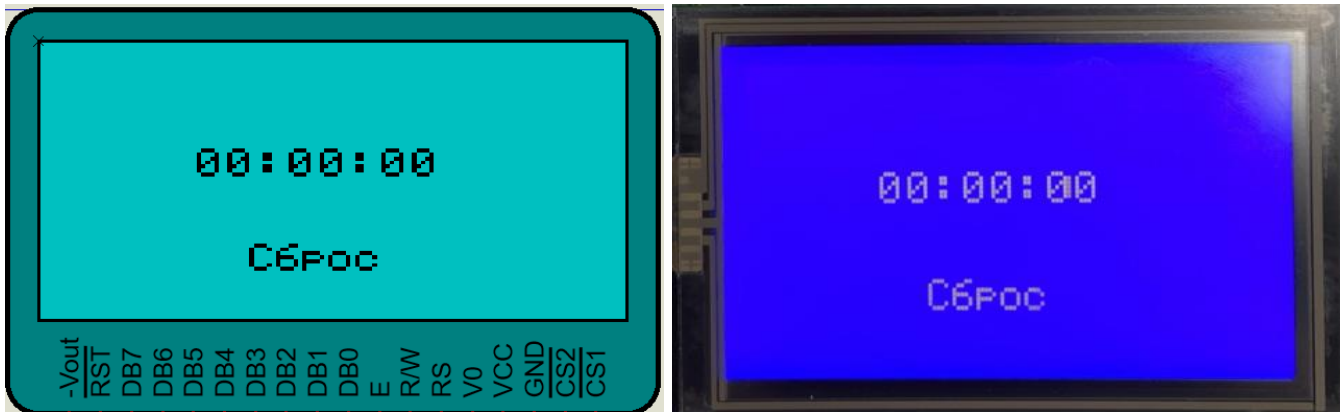

Рисунок 63 – Меню таймера после запуска, отрисованное в Proteus и на ME-GLCD

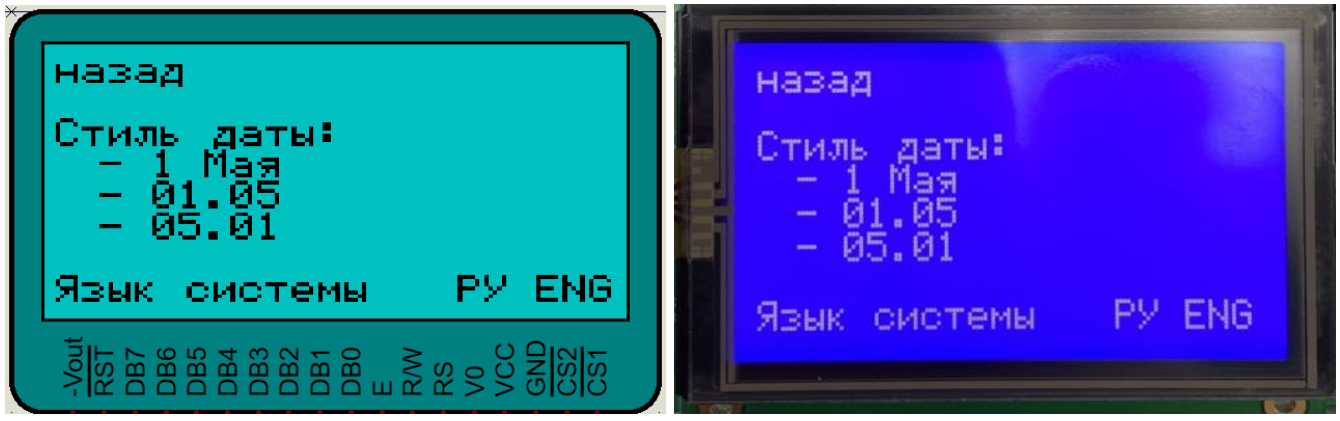

Рисунок 64 – Меню настроек прибора, отрисованное в Proteus и на ME-GLCD

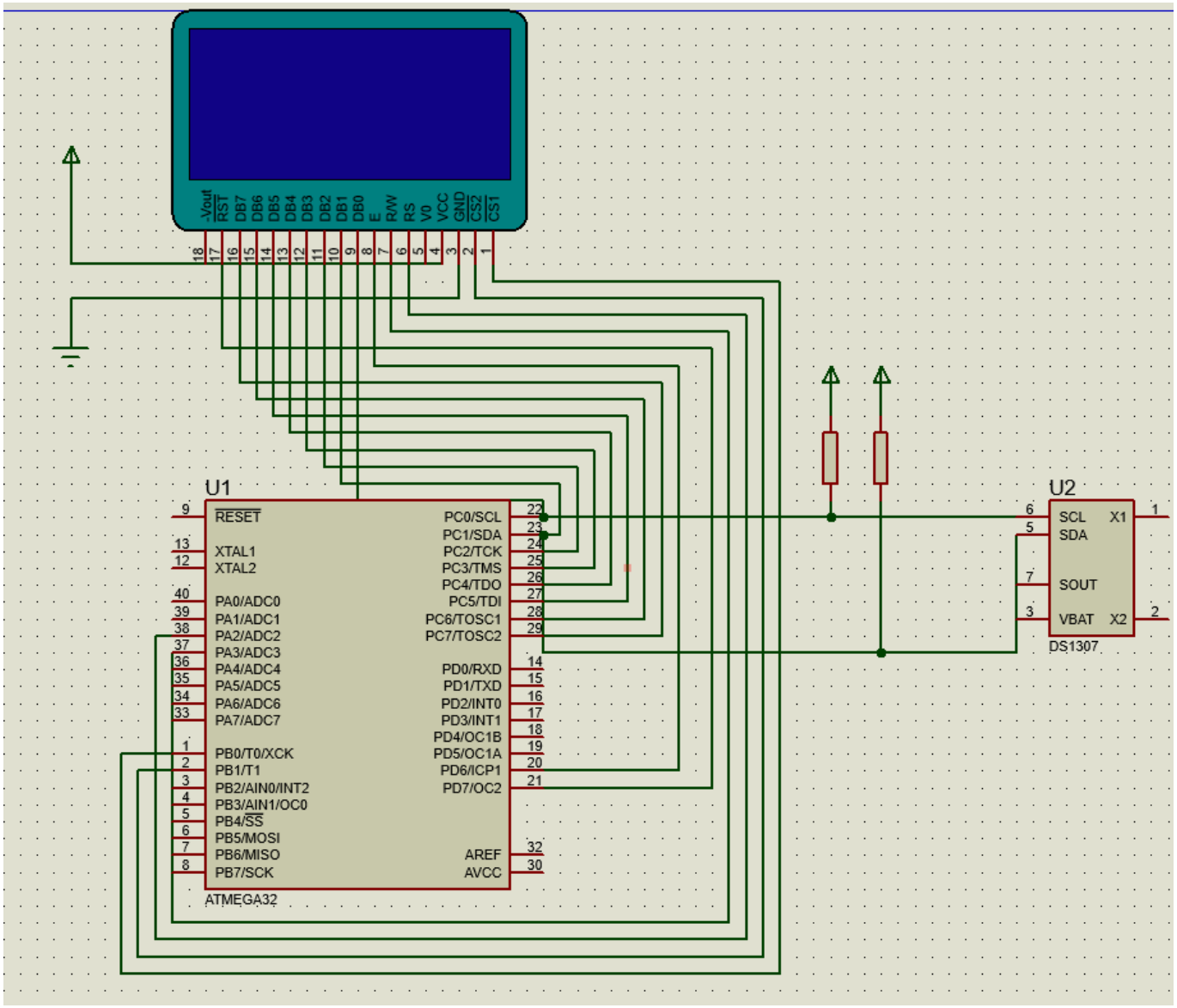

Рисунок 65 – Схема подключения ME-GLCD и DS1307 к ATmega32

ЗАКЛЮЧЕНИЕ

За время выполнения курсовой работы были получены навыки в работе с микроконтроллерами, жидкокристаллическими дисплеями, были получены навыки в разработке и отладки программ на языке Си для микроконтроллера ATmega32. Получены знания о том, как устроен микроконтроллер ATmega32, о том, как работать с резистивным сенсорной панелью и как работать с датчиком часов реального времени DS1307. Получены навыки в работе с устройствами с помощью интерфейса I2C.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация для EasyAVR V7 [Интернет ресурс] https://www.mikroe.com/easyavr (дата обращения 02.04.2021)
2. Документация к микроконтроллеру ATmega32 [интернет-ресурс] https://ww1.microchip.com/downloads/en/DeviceDoc/2503S.pdf (дата обращения 02.04.2021)
3. Часы реального времени DS1307 [интернет ресурс] https://datasheets.maximintegrated.com/en/ds/DS1307.pdf (дата обращения 02.04.2021)
4. Графический ЖКИ ME-GLCD 128x64 [интернет ресурс] https://static.chipdip.ru/lib/583/DOC000583796.pdf (дата обращения 03.04.2021)
5. Официальный сайт Mircochip Studio [Интернет ресурс] https://www.microchip.com/en-us/development-tools-tools-and-software/microchip-studio-for-avr-and-sam-devices (дата обращения 24.05.2021)
6. Официальный сайт Proteus [Интернет ресурс] https://www.labcenter.com/ (дата обращения 26.05.2021)

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ ТЕКСТ ПРОГРАММЫ

```
#define F_CPU 8000000UL

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <string.h>

#define LCD_DATA_PORT PORTC
#define LCD_DATA_DDR DDRC
#define LCD_DATA_PIN PINC
#define CS_PORT PORTB //Порт cs1/cs2
#define CS_DDR DDRB
#define RSRW_PORT PORTA //Порт rs и rw
#define RSRW_DDR DDRA
#define ERST_PORT PORTD //Порт E и RST
#define ERST_DDR DDRD
#define RS 2 // Данные/команда
#define RW 3 // Чтение/запись
#define EN 6 // Строб
#define CS1 0 // Контроллер 1
#define CS2 1 // Контроллер 2
#define RST 7 // Сброс адреса

void Time_1_init(); //Инициализация таймера/счетчика 1
void Timer0_init();
```

```
void mainMenu(void);
void getCurrentTime(void); //Функция для получения текущего времени
void getCurrentDate(void);//Функция для получения текущей даты
void printStartMenu(); //Функция отрисовки стартового меню
void printMainMenu(); //Функция отрисовки главного меню
void printStopwatchMenu(); //Функция отрисовки меню секундомера
void printTimerMenu(); //Функция отрисовки меню таймера
void printSettings(); //Функция отрисовки меню настроек

void LCD_Command(char Command);//Посылает команду жки
void LCD_Data(char Data);//Посылает данные жки
void LCD_Init();//Функция инициализации жки
void LCD_Clear();//Функция очистки дисплея
void LCD_GotoXY(uint8_t x, uint8_t y);//Функция установки положения курсора на экране
void LCD_drawByte(uint8_t x, uint8_t y, unsigned char byte);//ставит пиксель в координаты x  и y
void LCD_Char(uint8_t  x1, uint8_t y1, char code);//Функция для отображения символа на экране в координатах x и y
void LCD_String(uint8_t x1, uint8_t y1, char *string);//Функция для отображения строки на экране в позиции x и y

void init_DS1307(void);  //Инициализация часов
uint8_t read_DS1307(uint8_t addr);//Получение значений из часов

void init_adc(void); //Инициализация АЦП

char symbol[][5] =
{
        //стандарнтные символы начинаются с 0x20
```

```
{0x00,0x00,0x00,0x00,0x00},		//пробел
{0x00,0x00,0x4f,0x00,0x00},		//!
{0x00,0x07,0x00,0x07,0x00},		//"
{0x14,0x7f,0x14,0x7f,0x14},		//#
{0x24,0x2a,0x7f,0x2a,0x12},		//$
{0x23,0x13,0x08,0x64,0x62},		//%
{0x36,0x49,0x55,0x22,0x40},		//&
{0x00,0x05,0x03,0x00,0x00},		//,
{0x00,0x1c,0x22,0x41,0x00},		//(
{0x00,0x41,0x22,0x1c,0x00},		//)
{0x14,0x08,0x3E,0x08,0x14},		//*
{0x08,0x08,0x3E,0x08,0x08},		//+
{0x00,0x50,0x30,0x00,0x00},		//,
{0x08,0x08,0x08,0x08,0x08},		//-
{0x00,0x60,0x60,0x00,0x00},		//.
{0x20,0x10,0x08,0x04,0x02},		///

{0x3e,0x51,0x49,0x45,0x3e},		//0
{0x00,0x42,0x7f,0x40,0x00},		//1
{0x42,0x61,0x51,0x49,0x46},		//2
{0x21,0x41,0x45,0x4b,0x31},		//3
{0x18,0x14,0x12,0x7f,0x10},		//4
{0x27,0x45,0x45,0x45,0x39},		//5
{0x3c,0x4a,0x49,0x49,0x30},		//6
{0x01,0x71,0x09,0x05,0x03},		//7
{0x36,0x49,0x49,0x49,0x36},		//8
{0x06,0x49,0x49,0x29,0x1e},		//9
{0x00,0x36,0x36,0x00,0x00},		//:
{0x00,0x56,0x36,0x00,0x00},		//;
{0x08,0x14,0x22,0x41,0x00},		//<
```

```
{0x14,0x14,0x14,0x14,0x14},  //=
{0x00,0x41,0x22,0x14,0x08},  //>
{0x02,0x01,0x51,0x09,0x06},  //?

{0x32,0x49,0x71,0x41,0x3e},  //@
{0x7e,0x11,0x11,0x11,0x7e},  //A
{0x7f,0x49,0x49,0x49,0x36},  //B
{0x3e,0x41,0x41,0x41,0x22},  //C
{0x7f,0x41,0x41,0x22,0x1c},  //D
{0x7f,0x49,0x49,0x49,0x41},  //E
{0x7f,0x09,0x09,0x09,0x01},  //F
{0x3e,0x41,0x49,0x49,0x3a},  //G
{0x7f,0x08,0x08,0x08,0x7f},  //H
{0x00,0x41,0x7f,0x41,0x00},  //I
{0x20,0x40,0x41,0x3f,0x01},  //J
{0x7f,0x08,0x14,0x22,0x41},  //K
{0x7f,0x40,0x40,0x40,0x40},  //L
{0x7f,0x02,0x0c,0x02,0x7f},  //M
{0x7f,0x04,0x08,0x10,0x7f},  //N
{0x3e,0x41,0x41,0x41,0x3e},  //O

{0x7f,0x09,0x09,0x09,0x06},  //P
{0x3e,0x41,0x51,0x21,0x5e},  //Q
{0x7f,0x09,0x19,0x29,0x46},  //R
{0x46,0x49,0x49,0x49,0x31},  //S
{0x01,0x01,0x7f,0x01,0x01},  //T
{0x3f,0x40,0x40,0x40,0x3f},  //U
{0x1f,0x20,0x40,0x20,0x1f},  //V
{0x3f,0x40,0x70,0x40,0x3f},  //W
{0x63,0x14,0x08,0x14,0x63},  //X
```

```
{0x07,0x08,0x70,0x08,0x07},     //Y
{0x61,0x51,0x49,0x45,0x43},     //Z
{0x00,0x7F,0x41,0x41,0x00},     //[
{0x02,0x04,0x08,0x10,0x20},     // - обратный слэш
{0x00,0x41,0x41,0x7F,0x00},     //]
{0x04,0x02,0x01,0x02,0x04},     //^
{0x40,0x40,0x40,0x40,0x40},     //_

{0x00,0x01,0x02,0x04,0x00},     //'
{0x20,0x54,0x54,0x54,0x78},     //a
{0x7F,0x48,0x44,0x44,0x38},     //b
{0x38,0x44,0x44,0x44,0x28},     //c
{0x38,0x44,0x44,0x48,0x7F},     //d
{0x38,0x54,0x54,0x54,0x18},     //e
{0x08,0x7E,0x09,0x01,0x02},     //f
{0x0C,0x52,0x52,0x52,0x3E},     //g
{0x7F,0x08,0x04,0x04,0x78},     //h
{0x00,0x44,0x7D,0x40,0x00},     //i
{0x20,0x40,0x44,0x3D,0x00},     //j
{0x7F,0x10,0x28,0x44,0x00},     //k
{0x00,0x41,0x7F,0x40,0x00},     //l
{0x7C,0x04,0x18,0x04,0x78},     //m
{0x7C,0x08,0x04,0x04,0x78},     //n
{0x38,0x44,0x44,0x44,0x38},     //o

{0x7C,0x14,0x14,0x14,0x08},     //p
{0x08,0x14,0x14,0x18,0x7C},     //q
{0x7C,0x08,0x04,0x04,0x08},     //r
{0x48,0x54,0x54,0x54,0x20},     //s
{0x04,0x3F,0x44,0x40,0x20},     //t
```

```
{0x3C,0x40,0x40,0x20,0x7C},     //u
{0x1C,0x20,0x40,0x20,0x1C},     //v
{0x3C,0x40,0x30,0x40,0x3C},     //w
{0x44,0x28,0x10,0x28,0x44},     //x
{0x0C,0x50,0x50,0x50,0x3C},     //y
{0x44,0x64,0x54,0x4C,0x44},     //z
{0x00,0x08,0x36,0x41,0x00},     //{
{0x00,0x00,0x7f,0x00,0x00},     //|
{0x00,0x41,0x36,0x08,0x00},     //}
{0x02,0x01,0x02,0x02,0x01},     //~
{0x00,0x00,0x00,0x00,0x00},     //резерв
// 0x7f
//0xc0 и далее русские (0x60-0xa0)
{0x7e,0x11,0x11,0x11,0x7e},     //А
{0x7f,0x49,0x49,0x49,0x33},     //Б
{0x7f,0x49,0x49,0x49,0x36},     //В
{0x7f,0x01,0x01,0x01,0x03},     //Г
{0xe0,0x51,0x4f,0x41,0xff},     //Д
{0x7f,0x49,0x49,0x49,0x41},     //Е
{0x77,0x08,0x7f,0x08,0x77},     //Ж
{0x41,0x49,0x49,0x49,0x36},     //З
{0x7f,0x10,0x08,0x04,0x7f},     //И
{0x7c,0x21,0x12,0x09,0x7c},     //Й
{0x7f,0x08,0x14,0x22,0x41},     //К
{0x20,0x41,0x3f,0x01,0x7f},     //Л
{0x7f,0x02,0x0c,0x02,0x7f},     //М
{0x7f,0x08,0x08,0x08,0x7f},     //Н
{0x3e,0x41,0x41,0x41,0x3e},     //О
{0x7f,0x01,0x01,0x01,0x7f},     //П
```

```
{0x7f,0x09,0x09,0x09,0x06},          //Р
{0x3e,0x41,0x41,0x41,0x22},          //С
{0x01,0x01,0x7f,0x01,0x01},          //Т
{0x47,0x28,0x10,0x08,0x07},          //У
{0x1c,0x22,0x7f,0x22,0x1c},          //Ф
{0x63,0x14,0x08,0x14,0x63},          //Х
{0x7f,0x40,0x40,0x40,0xff},          //Ц
{0x07,0x08,0x08,0x08,0x7f},          //Ч
{0x7f,0x40,0x7f,0x40,0x7f},          //Ш
{0x7f,0x40,0x7f,0x40,0xff},          //Щ
{0x01,0x7f,0x48,0x48,0x30},          //Ъ
{0x7f,0x48,0x30,0x00,0x7f},          //Ы
{0x00,0x7f,0x48,0x48,0x30},          //Э
{0x22,0x41,0x49,0x49,0x3e},          //Ь
{0x7f,0x08,0x3e,0x41,0x3e},          //Ю
{0x46,0x29,0x19,0x09,0x7f},          //Я

{0x20,0x54,0x54,0x54,0x78},          //а
{0x3c,0x4a,0x4a,0x49,0x31},          //б
{0x7c,0x54,0x54,0x28,0x00},          //в
{0x7c,0x04,0x04,0x04,0x0c},          //г
{0xe0,0x54,0x4c,0x44,0xfc},          //д
{0x38,0x54,0x54,0x54,0x18},          //е
{0x6c,0x10,0x7c,0x10,0x6c},          //ж
{0x44,0x44,0x54,0x54,0x28},          //з
{0x7c,0x20,0x10,0x08,0x7c},          //и
{0x7c,0x41,0x22,0x11,0x7c},          //й
{0x7c,0x10,0x28,0x44,0x00},          //к
{0x20,0x44,0x3c,0x04,0x7c},          //л
{0x7c,0x08,0x10,0x08,0x7c},          //м
```

```
        {0x7c,0x10,0x10,0x10,0x7c},          //н
        {0x38,0x44,0x44,0x44,0x38},          //о
        {0x7c,0x04,0x04,0x04,0x7c},          //п

        {0x7C,0x14,0x14,0x14,0x08},          //р
        {0x38,0x44,0x44,0x44,0x28},          //с
        {0x04,0x04,0x7c,0x04,0x04},          //т
        {0x0C,0x50,0x50,0x50,0x3C},          //у
        {0x30,0x48,0xfc,0x48,0x30},          //ф
        {0x44,0x28,0x10,0x28,0x44},          //х
        {0x7c,0x40,0x40,0x40,0xfc},          //ц
        {0x0c,0x10,0x10,0x10,0x7c},          //ч
        {0x7c,0x40,0x7c,0x40,0x7c},          //ш
        {0x7c,0x40,0x7c,0x40,0xfc},          //щ
        {0x04,0x7c,0x50,0x50,0x20},          //ъ
        {0x7c,0x50,0x50,0x20,0x7c},          //ы
        {0x7c,0x50,0x50,0x20,0x00},          //ь
        {0x28,0x44,0x54,0x54,0x38},          //э
        {0x7c,0x10,0x38,0x44,0x38},          //ю
        {0x08,0x54,0x34,0x14,0x7c},          //я
        //0xff

};

char settingsImage[] = {0x2A,0x2F,0x2A, 0x2A, 0x7A, 0x2A, 0x2A, 0x3E, 0x2A};
//Переменные отвечающие за выбор экрана и  меню настроек
int scene = 0, isSceneChanged = 0, isPressed = 0;
char dataStyle = 0, language = 0, startScreenScene = 0;
```

```
//Переменные для времени и даты
int hour = 0, minutes = 0, day = 1, weekDay =0, month = 1, year = 21;

//Переменные секундомера и таймера
int stopwatchTime[3] = {0,0,0}, isStartedSWT = 0, resetSWT = 0;
int timerTime[3] = {0,0,0}, isStartedTMR = 0, setTMR = 1;

//Переменные для дней недели на русском и английском языках
char *DaysRU[7] = {"воскресенье", "понедельник", "вторник", "среда",
"четверг", "пятница", "суббота"};
char *DaysENG[7] = {"Sunday", "Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday"};

//Переменные для месяцев на русском и английском языках
char *monthsRU[12] = {"января", "февраля", "марта", "апреля", "мая", "июня",
"июля", "августа", "сентября", "октября", "ноября", "декабря"};
char *monthsENG[12] = {"January", "February", "March", "April", "May",
"June", "July", "August", "September", "October", "November", "December"};

volatile unsigned char sec_flag=0;
volatile unsigned int ms_counter=0;

uint16_t temp_code_ADC;
volatile unsigned long long int code_ADC;
volatile unsigned char temp_ADC;
uint16_t x,y;

int main(void)
{
```

```
	//Инициализация ЖКИ, таймер-счетчика 1, ацп и датчика часов реального времени DS1307
	LCD_Init();
	Time_1_init();
	init_adc();
	Timer0_init();
	init_DS1307();
	LCD_Clear();
	//Ассемблеровская команда, для разрешения прерываний
	asm("sei");
	while (1)
	{
		mainMenu();
	}
}

//Настройка таймер-счетчика 1
void Time_1_init()
{
	TCNT1 = 0;
	OCR1A = 31250;
	// Обнуление битов WGM10 и WGM11 для выбора режима CTC
	TCCR1A &= ~((1<<WGM11)|(1<<WGM10));
	// Установка 1 в бит WGM12 для установки режима CTC и установка значения предделителя в значение 256
	TCCR1B |= (1<<WGM12)|(1<<CS12);
	//Флаг сравнения значения TCNT1 с регистром OCR1A
	TIFR |= (1<<OCF1A);
	//Разрешение прерывания по сравнению с регистром OCR1A
	TIMSK |= (1<<OCIE1A);
```

```
}
//Функция прерывания таймер-счетчика 1
ISR (TIMER1_COMPA_vect)
{
	//Увеличиваем каждую секунду содержимое stopwatchTime[0]
	if(isStartedSWT) stopwatchTime[0]++;
	//Уменьшаем каждую секунду содержимое timerTime[0]
	if(isStartedTMR) timerTime[0]--;
	if(scene == 1)
	{
		//Получаем текущее время
		getCurrentTime();
		//Получаем текующую дату
		getCurrentDate();
	}
}

//Инициализации АЦП
void init_adc(void)
{
	DDRA|=0b00001100;
	PORTA=0b00000000;
	ADMUX=0b00000000;
	ADCSRA=(0b10000111|0b01000000);
}

void Timer0_init()
{
	TCNT0=0b00000000;
	OCR0=249;
```

```
	TCCR0=0b00001100;
	TIMSK|=0b00000010;
	TIFR|=0b00000011;
}
ISR (TIMER0_COMP_vect)
{
	ms_counter++;
	if(ms_counter==30)
	{
		ms_counter=0;
		sec_flag=1;

		PORTA=0b00000100;
		_delay_ms(2);
		ADMUX=0b00000000;
		ADCSRA=0b11000111;
		while((ADCSRA&0b01000000)!=0){}
		x=ADCL|(ADCH<<8);

		PORTA=0b00001000;
		_delay_ms(2);
		ADMUX=0b00000001;
		ADCSRA=0b11000111 ;
		while((ADCSRA&0b01000000)!=0){}
		y=ADCL|(ADCH<<8);

	}

	if (scene == 0) //Проверка нажатия кнопок для стартового меню
	{
```

```
if (startScreenScene == 0) //Проверка нажатия кнопок на стартовом экране выбора языка
{
	if((x>=120)&&(x<=340)&&(y>=380)&&(y<=480))   //Выбран русский язык
	{
		language = 0;
		startScreenScene = 1;
		isSceneChanged++;	//Флаг смены экрана
	}
	else	if((x>=530)&&(x<=760)&&(y>=380)&&(y<=480)) //Выбран английский язык
	{
		language = 1;
		startScreenScene = 1;
		isSceneChanged++; //Флаг смены экрана
	}
}
if (startScreenScene == 1)
{
	if	((x>=80)&&(x<=130)&&(y>=680)&&(y<=730)	&& (isPressed == 0)) //Нажата кнопка плюс для часов
	{
		isPressed++;
		hour++;
		if(hour > 23) hour = 0;
	}

	else	if	((x>=78)&&(x<=150)&&(y>=310)&&(y<=370)	&& (isPressed == 0))//Нажата кнопка минус для часов
```

```
                {
                        isPressed++;
                        hour--;
                        if (hour < 0) hour = 23;
                }
                if      ((x>=190)&&(x<=260)&&(y>=640)&&(y<=720)      &&
(isPressed == 0)) //Нажата кнопка плюс для минут
                {
                        isPressed++;
                        minutes++;
                        if(minutes > 59) minutes = 0;
                }
                else  if  ((x>=185)&&(x<=275)&&(y>=320)&&(y<=390)  &&
(isPressed == 0))//Нажата кнопка минус для минут
                {
                        isPressed++;
                        minutes--;
                        if (minutes < 0) minutes = 59;
                }
                if      ((x>=410)&&(x<=510)&&(y>=635)&&(y<=710)      &&
(isPressed == 0)) //Нажата кнопка плюс для дня
                {
                        isPressed++;
                        day++;
                        if ((month == 1) || (month == 3) || (month == 5) || (month
== 7) || (month == 8) || (month == 10) || (month == 12))
                        {
                                if (day > 31) day = 1;
                        }
```

```
                else if( month == 2 && month % 4 == 0 && day > 29) day = 1;
                else if( month == 2 && month % 4 != 0 && day > 28) day = 1;
                else if(day > 30) day = 1;
            }
            else  if  ((x>=420)&&(x<=520)&&(y>=300)&&(y<=360)  &&  (isPressed == 0)) //Нажата кнопка минус для дня
            {
                isPressed++;
                day--;
                if (day < 1)
                {
                    if ((month == 1) || (month == 3) || (month == 5) || (month == 7) || (month == 8) || (month == 10) || (month == 12))
                    {
                        day = 31;
                    }
                    else if( month == 2 && month % 4 == 0) day = 29;
                    else if( month == 2 && month % 4 != 0) day = 28;
                    else day = 30;
                }
            }
            if      ((x>=550)&&(x<=650)&&(y>=615)&&(y<=690)      &&  (isPressed == 0))//Нажата кнопка плюс для месяца
            {
                isPressed++;
                month++;
                if(month > 12) month = 1;
                //Проверка коректности даты для выбранного месяца
```

```
                    if( month == 2 && month % 4 == 0 && day > 29) day =
29;
                    if( month == 2 && month % 4 != 0 && day > 28) day =
28;
                    else if(((month == 4) || (month == 6) || (month == 9) ||
(month == 11)) && day > 30) day = 30;
                }
                if      ((x>=570)&&(x<=670)&&(y>=260)&&(y<=400)      &&
(isPressed == 0))//Нажата кнопка минус для месяца
                {
                    isPressed++;
                    month--;
                    if (month < 1) month = 12;
                    //Проверка коректности даты для выбранного месяца
                    if( month == 2 && month % 4 == 0 && day > 29) day =
29;
                    if( month == 2 && month % 4 != 0 && day > 28) day =
28;
                    else if(((month == 4) || (month == 6) || (month == 9) ||
(month == 11)) && day > 30) day = 30;
                }
                if      ((x>=700)&&(x<=790)&&(y>=600)&&(y<=670)      &&
(isPressed == 0))//Нажата кнопка плюс для года
                {
                    isPressed++;
                    year++;
                    if (year > 99) year = 0;
                }
                if      ((x>=730)&&(x<=815)&&(y>=300)&&(y<=360)      &&
(isPressed == 0))//Нажата кнопка минус для года
```

```
{
        isPressed++;
        year--;
        if (year < 0) year = 99;
}
if      ((x>=505)&&(x<=556)&&(y>=132)&&(y<=215)      &&
(isPressed == 0))//Нажата кнопка плюс дня недели
{
        isPressed++;
        weekDay++;
        if(weekDay > 6) weekDay = 0;

        LCD_GotoXY(7, 0);
        for (int i; i<85;i++)
        {
                LCD_GotoXY(7, i);
                LCD_Data(0);
        }
}
if      ((x>=72)&&(x<=106)&&(y>=140)&&(y<=215)      &&
(isPressed == 0)) //Нажата кнопка минус дня недели
{
        isPressed++;
        weekDay--;
        if(weekDay < 0) weekDay = 6;

        for (int i=6; i<76;i++)
        {
                LCD_GotoXY(7, i);
                LCD_Data(0);
```

```
                }
            }
            if ((x>=725)&&(x<=825)&&(y>=120)&&(y<=205)) //Нажата кнопка OK
            {
                //Запись установленных данных в датчик часов реального времени DS1307
                write_DS1307(0x00, 0);
                _delay_us(50);
                write_DS1307(0x01, minutes);
                _delay_us(50);
                write_DS1307(0x02, hour);
                _delay_us(50);
                write_DS1307(0x03, weekDay);
                _delay_us(50);
                write_DS1307(0x04, day);
                _delay_us(50);
                write_DS1307(0x05, month);
                _delay_ms(3);
                //Переход на сцену главного меню
                scene = 1;
                isSceneChanged++; //Флаг смены экрана
            }
        }
    }
    else if (scene == 1)        //Проверка нажатия кнопок для главного меню
    {
        if      ((x>=725)&&(x<=800)&&(y>=670)&&(y<=730))      //Нажата кнопка настройки
        {
```

```
                scene = 4;
                isSceneChanged++; //Флаг смены экрана
            }
            if ((x>=100)&&(x<=440)&&(y>=190)&&(y<=305)) //Нажата
кнопка секундомера
            {
                scene = 2;
                isSceneChanged++; //Флаг смены экрана
            }
            if ((x>=530)&&(x<=750)&&(y>=190)&&(y<=290)) //Нажата
кнопка таймера
            {
                scene = 3;
                isSceneChanged++; //Флаг смены экрана
            }

        }
        else if (scene == 2) //Проверка нажатия кнопок для секундомера
        {
            if ((x>=75)&&(x<=210)&&(y>=718)&&(y<=790)) //Нажата кнопка
назад
            {
                scene = 1;
                isSceneChanged++; //Флаг смены экрана
            }

            if ((x>=430)&&(x<=610)&&(y>=275)&&(y<=375)) isStartedSWT =
1;      //Нажата кнопка старт
            if ((x>=195)&&(x<=360)&&(y>=290)&&(y<=390)) isStartedSWT =
0;      //Нажата кнопка стоп
```

```
            if ((x>=305)&&(x<=510)&&(y>=120)&&(y<=220))
                  //Нажата кнопка сброс
            {
                  isStartedSWT = 0;
                  stopwatchTime[0] = 0;
                  stopwatchTime[1] = 0;
                  stopwatchTime[2] = 0;
            }
      }
      else if (scene == 3) //Проверка нажатия кнопок для таймера
      {
            if (!isStartedTMR)
            {
                  if   ((x>=75)&&(x<=210)&&(y>=718)&&(y<=790))   //Нажата
кнопка назад
                  {
                        scene = 1;
                        isSceneChanged++; //Флаг смены экрана
                  }


                  else  if  ((x>=430)&&(x<=570)&&(y>=680)&&(y<=760)  &&
(isPressed == 0)) //Нажата кнопка +1 секунда
                  {
                        timerTime[0]++;
                        isPressed++;
                  }
```

```
                else if ((x>=590)&&(x<=770)&&(y>=670)&&(y<=740) &&
(isPressed == 0)) //Нажата кнопка +10 секунд
                {
                        timerTime[0]+=10;
                        isPressed++;
                }
                else if ((x>=430)&&(x<=580)&&(y>=605)&&(y<=700) &&
(isPressed == 0)) //Нажата кнопка +1 минута
                {
                        timerTime[1]++;
                        isPressed++;
                }
                else if ((x>=605)&&(x<=775)&&(y>=580)&&(y<=690) &&
(isPressed == 0)) //Нажата кнопка +10 минут
                {
                        timerTime[1]+=10;
                        isPressed++;
                }
                else if ((x>=450)&&(x<=590)&&(y>=525)&&(y<=620) &&
(isPressed == 0)) //Нажата кнопка +1 час
                {
                        timerTime[2]++;
                        isPressed++;
                }
                else if ((x>=620)&&(x<=770)&&(y>=500)&&(y<=600) &&
(isPressed == 0)) //Нажата кнопка +5 часов
                {
                        timerTime[2]+=5;
                        isPressed++;
                }
```

```
                else if ((x>=445)&&(x<=600)&&(y>=360)&&(y<=470) &&
(isPressed == 0)) //Нажата кнопка -1 секунда
                {
                        timerTime[0]--;
                        isPressed++;
                }
                else if ((x>=600)&&(x<=790)&&(y>=350)&&(y<=450) &&
(isPressed == 0)) //Нажата кнопка -10 секунд
                {
                        timerTime[0]-=10;
                        isPressed++;
                }
                else if ((x>=450)&&(x<=600)&&(y>=270)&&(y<=370) &&
(isPressed == 0)) //Нажата кнопка -1 минута
                {
                        timerTime[1]--;
                        isPressed++;
                }
                else if ((x>=600)&&(x<=790)&&(y>=260)&&(y<=360) &&
(isPressed == 0)) //Нажата кнопка -10 минут
                {
                        timerTime[1]-=10;
                        isPressed++;
                }
                else if ((x>=450)&&(x<=600)&&(y>=190)&&(y<=290) &&
(isPressed == 0)) //Нажата кнопка -1 час
                {
                        timerTime[2]--;
                        isPressed++;
                }
```

```
                else if ((x>=600)&&(x<=790)&&(y>=170)&&(y<=270) &&
(isPressed == 0)) //Нажата кнопка -5 часов
                {
                        timerTime[2]-=5;
                        isPressed++;
                }
                else     if     ((x>=100)&&(x<=325)&&(y>=265)&&(y<=415))
//Нажата кнопка старт
                {
                        setTMR = 0;
                        isStartedTMR = 1;
                        isSceneChanged++; //Флаг смены экрана
                }
        }
        else if ((x>=300)&&(x<=550)&&(y>=190)&&(y<=330)) //Нажата
кнопка сброс
        {
                setTMR = 1;
                isStartedTMR = 0;
                isSceneChanged++; //Флаг смены экрана
        }

    }
    else if (scene == 4) //Проверка нажатия кнопок для настроек
    {
        if ((x>=75)&&(x<=210)&&(y>=718)&&(y<=790)) //Нажата кнопка
назад
        {
                scene = 1;
                isSceneChanged++; //Флаг смены экрана
```

```
            }

            else if ((x>=155)&&(x<=360)&&(y>=475)&&(y<=535)) dataStyle =
0;      //Нажата кнопка 1 мая
            else if ((x>=155)&&(x<=360)&&(y>=380)&&(y<475)) dataStyle =
1;      //Нажата кнопка 01.05
            else if ((x>=155)&&(x<=360)&&(y>=280)&&(y<380)) dataStyle =
2;      //Нажата кнопка 05.01
            else if ((x>=590)&&(x<=680)&&(y>=130)&&(y<=215)) language =
0;      //Нажата кнопка РУ
            else if ((x>=710)&&(x<=825)&&(y>=130)&&(y<=215)) language =
1;      //Нажата кнопка РУ
        }
        if ((x<70) && (y<70) && isPressed)
        {
            isPressed = 0;
        }
}


//Функция вывода главного меню
void mainMenu()
{
        if (isSceneChanged) //Очистка экрана при смене сцены
        {
            LCD_Clear();
            isSceneChanged = 0;
        }
        if (scene == 0) //Стартовое меню
        {
```

```
            printStartMenu();
        }
        else if(scene == 1) //Главное меню
        {
            printMainMenu();
        }
        else if (scene == 2) //Меню секундомера
        {
            printStopwatchMenu();
        }
        else if(scene == 3) //Меню таймера
        {
            printTimerMenu();
        }
        else if(scene == 4) //Меню Настроек
        {
            printSettings();
        }
}

//Функция отрисовки стартового меню
void printStartMenu()
{
        //Отрисовка стартового меню до выбора языка
        if (startScreenScene == 0)
        {
            LCD_String(4, 10, "Русский");
            LCD_String(4, 78, "English");
        }
        //Отрисовка стартового меню после выбора языка
```

```
if (startScreenScene == 1)
{
    //Отображение кнопок прибавления и убавление времени для настройки времени
    LCD_Char(1,4, '+');
    LCD_Char(5,4, '-');
    LCD_Char(1,26, '+');
    LCD_Char(5,26, '-');
    LCD_Char(3,0, hour/10 + '0');
    LCD_Char(3,8, hour%10 + '0');
    LCD_Char(3,16, ':');
    LCD_Char(3,24, minutes/10 + '0');
    LCD_Char(3,32, minutes%10 + '0');

    //Отображение кнопок прибавления и убавления для настройки даты
    LCD_Char(1,67, '+');
    LCD_Char(5,67, '-');
    LCD_Char(1,91, '+');
    LCD_Char(5,91, '-');
    LCD_Char(1, 115, '+');
    LCD_Char(5,115, '-');
    LCD_Char(3,63, day/10 + '0');
    LCD_Char(3,71, day%10 + '0');
    LCD_Char(3,79, '.');
    LCD_Char(3,87, month/10 + '0');
    LCD_Char(3,95, month%10 + '0');
    LCD_Char(3,103, '.');
    LCD_Char(3,111, year/10 + '0');
    LCD_Char(3,119, year%10 + '0');
```

```
            //Отображение дня недели и кнопок смены дня недели на русском
языке
            if (!language)
            {
                  LCD_Char(7, 0, '-');
                  LCD_String(7,6 + (70 - strlen(DaysRU[weekDay])*6)/2,
DaysRU[weekDay]);
                  LCD_Char(7, 76, '+');
            }
            //Отображение дня недели и кнопок смены дня недели на
английском языке
            else
            {
                  LCD_Char(7, 0, '-');
                  LCD_String(7,6 + (70 - strlen(DaysENG[weekDay])*6)/2,
DaysENG[weekDay]);
                  LCD_Char(7, 76, '+');
            }
            //Отображение кнопки окончания стартовых настроек
            LCD_String(7, 114, "OK");
      }

}

//Функция получения текущего времени
void getCurrentTime()
{
      minutes = read_DS1307(0x01);
      hour = read_DS1307(0x02);
```

```
}

//Функция получения текущей даты, дня недели и месяца
void getCurrentDate()
{
	weekDay = read_DS1307(0x03);
	day = read_DS1307(0x04);
	month = read_DS1307(0x05);
}
//Функция отрисовки главного меню
void printMainMenu()
{
	//Вывод времени
	LCD_Char(3,42, hour/10 + '0');
	LCD_Char(3,51, hour%10 + '0');
	LCD_Char(3,60, ':');
	LCD_Char(3,68, minutes/10 + '0');
	LCD_Char(3,76, minutes%10 + '0');

	//Вывод даты в стиле "1 мая"/"May 1st"
	if(dataStyle == 0)
	{
		//Вывод даты на русском языке
		if(!language)
		{
			LCD_Char(0,0, day/10+'0');
			LCD_Char(0,6, day%10+'0');
			LCD_String(0,12, monthsRU[month-1]);
			LCD_String(1,0,DaysRU[weekDay]);
		}
```

```
        //Вывод даты на английском языке
        else
        {
            LCD_String(0,0, monthsENG[month-1]);
            LCD_Char(0,strlen(monthsENG[month-1])*6+3, day/10+'0');
            LCD_Char(0,strlen(monthsENG[month-1])*6 + 9,
day%10+'0');
            LCD_String(1,0,DaysENG[weekDay-1]);

            //Проверка необходимого окончания для даты
            if ((day%10 == 1) && (day != 11))
            {
                LCD_Char(0, strlen(monthsENG[month-1])*6 + 15, 's');
                LCD_Char(0, strlen(monthsENG[month-1])*6 + 21, 't');
            }
            else if ((day%10 == 2) && (day != 12))
            {
                LCD_Char(0, strlen(monthsENG[month-1])*6 + 15, 'n');
                LCD_Char(0, strlen(monthsENG[month-1])*6 + 21, 'd');
            }
            else if((day%10 == 3) && (day != 13))
            {
                LCD_Char(0, strlen(monthsENG[month-1])*6 + 15, 'r');
                LCD_Char(0, strlen(monthsENG[month-1])*6 + 21, 'd');
            }
            else
            {
                LCD_Char(0, strlen(monthsENG[month-1])*6 + 15, 't');
                LCD_Char(0, strlen(monthsENG[month-1])*6 + 21, 'h');
```

```
                }
            }
        }
        //Вывод даты в стиле "01.05"
        else if(dataStyle == 1)
        {
            LCD_Char(1,0, day/10+'0');
            LCD_Char(1,6, day%10+'0');
            LCD_Char(1,12,'.');
            LCD_Char(1,18, month/10+'0');
            LCD_Char(1,24, month%10+'0');
            if(!language)       LCD_String(0,0,DaysRU[weekDay]);
            else LCD_String(0,0,DaysENG[weekDay]);
        }
        //Вывод даты в стиле "05.01"
        else if (dataStyle == 2)
        {
            LCD_Char(1,0, month/10+'0');
            LCD_Char(1,6, month%10+'0');
            LCD_Char(1,12,'.');
            LCD_Char(1,18, day/10+'0');
            LCD_Char(1,24, day%10+'0');
            if(!language)       LCD_String(0,0,DaysRU[weekDay]);
            else LCD_String(0,0,DaysENG[weekDay]);
        }
        //Вывод кнопок Настроек секундомера и таймера в соответствии я
зыком
        if (!language)
        {
            LCD_String(6,8,"Секундомер"); //Вывод кнопки Секундомера
```

```
            LCD_String(6,80,"Таймер"); //Вывод кнопки Таймера
        }
        else
        {
            LCD_String(6,8,"Stopwatch"); //Вывод кнопки Секундомера
            LCD_String(6,80,"Timer"); //Вывод кнопки Таймера
        }
        //Вывод знака настроек
        for (int i = 0; i<10; i++)
        {
            LCD_drawByte(0, 117+i, settingsImage[i]);
        }
}

//Функция отрисовки меню секундомера
void printStopwatchMenu()
{
        //Проверка коректности времени секундомера
        if (stopwatchTime[0] == 60)
        {
            stopwatchTime[0] = 0;
            stopwatchTime[1]++;
            if(stopwatchTime[1] == 60)
            {
                stopwatchTime[1] = 0;
                stopwatchTime[2]++;
            }
        }
        //Отрисовка времени секундомера
        LCD_Char(3,75,stopwatchTime[0]/10+'0');
```

```
LCD_Char(3,82,stopwatchTime[0]%10+'0');
LCD_Char(3,68,':');
LCD_Char(3,54,stopwatchTime[1]/10+'0');
LCD_Char(3,61,stopwatchTime[1]%10+'0');
LCD_Char(3,33,stopwatchTime[2]/10+'0');
LCD_Char(3,40,stopwatchTime[2]%10+'0');
LCD_Char(3,47,':');
//Отрисовка управляющих кнопок назад, старт и стоп на русском языке
if (!language)
{
	LCD_String(0,0,"назад");
	LCD_String(5,27,"Стоп");
	LCD_String(5,64,"Старт");
	if (isStartedSWT)
	{
		LCD_String(7,45, "Сброс");
	}
}
//Отрисовка управляющих кнопок назад, старт и стоп на английском языке
else
{
	LCD_String(0,0,"back");
	LCD_String(5,24,"Stop");
	LCD_String(5,64,"Start");
	if (isStartedSWT)
	{
		LCD_String(7,45, "Reset");
	}
}
```

```
}

//Функция отрисовки меню таймера
void printTimerMenu()
{
	//Проверка корректности времени таймера
	if (timerTime[0] == -1)
	{
		timerTime[0] = 59;
		timerTime[1]--;
		if(timerTime[1] == -1)
		{
			timerTime[1] = 59;
			timerTime[2]--;
		}
	}

	//Проверка условия для остановки таймера
	if (isStartedTMR && (timerTime[2] == 0) && (timerTime[1] == 0) && (timerTime[0] == 0))
	{
		LCD_Clear();
		isStartedTMR = 0;
		timerTime[0] = 0;
		timerTime[1] = 0;
		timerTime[2] = 0;
	}

	//Отрисовка меню таймера
```

```
if (setTMR)
{
	//Отрисовка меню таймера до запуска на русском языке
	if (!language)
	{
		LCD_String(0,0,"назад");
		LCD_Char(3,2, timerTime[2]/10+'0');
		LCD_Char(3,9,timerTime[2]%10+'0');
		LCD_Char(3,16,':');
		LCD_Char(3,23, timerTime[1]/10+'0');
		LCD_Char(3,30,timerTime[1]%10+'0');
		LCD_Char(3,37,':');
		LCD_Char(3,44, timerTime[0]/10+'0');
		LCD_Char(3,51,timerTime[0]%10+'0');

		LCD_String(0,69,"+1с");
		LCD_String(0,95,"+10с");
		LCD_String(1,69,"+1м");
		LCD_String(1,95,"+10м");
		LCD_String(2,69,"+1ч");
		LCD_String(2,101,"+5ч");

		LCD_String(4,69,"-1с");
		LCD_String(4,95,"-10с");
		LCD_String(5,69,"-1м");
		LCD_String(5,95,"-10м");
		LCD_String(6,69,"-1ч");
		LCD_String(6,101,"-5ч");

		LCD_String(5,13,"Старт");
```

```
}
//Отрисовка меню таймера до запуска на английском языке
else
{
	LCD_String(0,0,"back");
	LCD_Char(3,2, timerTime[2]/10+'0');
	LCD_Char(3,9,timerTime[2]%10+'0');
	LCD_Char(3,16,':');
	LCD_Char(3,23, timerTime[1]/10+'0');
	LCD_Char(3,30,timerTime[1]%10+'0');
	LCD_Char(3,37,':');
	LCD_Char(3,44, timerTime[0]/10+'0');
	LCD_Char(3,51,timerTime[0]%10+'0');

	LCD_String(0,69,"+1s");
	LCD_String(0,95,"+10s");
	LCD_String(1,69,"+1m");
	LCD_String(1,95,"+10m");
	LCD_String(2,69,"+1h");
	LCD_String(2,101,"+5h");

	LCD_String(4,69,"-1s");
	LCD_String(4,95,"-10s");
	LCD_String(5,69,"-1m");
	LCD_String(5,95,"-10m");
	LCD_String(6,69,"-1h");
	LCD_String(6,101,"-5h");

	LCD_String(5,13,"Start");
}
```

```
	}
	else
	{
		LCD_Char(3,75,timerTime[0]/10+'0');
		LCD_Char(3,82,timerTime[0]%10+'0');
		LCD_Char(3,68,':');
		LCD_Char(3,54,timerTime[1]/10+'0');
		LCD_Char(3,61,timerTime[1]%10+'0');
		LCD_Char(3,33,timerTime[2]/10+'0');
		LCD_Char(3,40,timerTime[2]%10+'0');
		LCD_Char(3,47,':');

		if (!language) LCD_String(6,45,"Сброс");
		else LCD_String(6,45,"Reset");
	}
}
//Функция отрисовки меню настроек
void printSettings()
{
	if (!language)
	{
		LCD_String(0,0,"назад");

		LCD_String(2,0,"Стиль даты:");
		LCD_String(3,10, "- 1 Мая");
		LCD_String(4,10, "- 01.05");
		LCD_String(5,10, "- 05.01");

		LCD_String(7,0, "Язык системы");
		LCD_String(7, 92, "РУ");
```

```
            LCD_String(7, 110, "ENG");
        }
        else
        {
            LCD_String(0,0,"back");

            LCD_String(2,0,"Data style:");
            LCD_String(3,10, "- May 1st");
            LCD_String(4,10, "- 01.05");
            LCD_String(5,10, "- 05.01");

            LCD_String(7,0, "System");
            LCD_String(7, 92, "РУ");
            LCD_String(7, 110, "ENG");
        }
}


void trigger()
{
        ERST_PORT |= (1<<EN);
        _delay_us(50);
        ERST_PORT &= ~(1<<EN);
        _delay_us(50);
}
//Функция для отправки команд
void LCD_Command(char Command)
{
        //Выставляем в регистре RS бит 0 для состояния принятия команды
        RSRW_PORT &= ~(1 << RS);
```

```
	//Формируем состояние записи выставив бит RW в 1
	RSRW_PORT &= ~(1 << RW);
	//Помещаем команду в порт для данных жки
	LCD_DATA_PORT = Command;
	//Выставляем бит EN в 1 для формирования состояния начала отправки команды
	trigger();
}
//Функция для отправки данных
void LCD_Data(char Data)
{
	//Выставляем бит RS в 1 для формирвоания состояния принятия данных
	RSRW_PORT |=  (1 << RS);
	//Выставляем бит RW в 0 для формирования состояния записи в жки
	RSRW_PORT &= ~(1 << RW);
	//Помещаем данные в порт данных жки
	LCD_DATA_PORT = Data;
	trigger();
}

//Функция инициализции дисплея
void LCD_Init()
{
	//Настраиваем все порты на выход
	LCD_DATA_DDR = 0xFF;
	RSRW_DDR |= (1<<RS)|(1<<RW);
	ERST_DDR |= (1<<EN)|(1<<RST);
	CS_DDR |= (1<<CS1)|(1<<CS2);
	//включаем оба контроллера
```

```
	CS_PORT &= ~((1 << CS1) | (1 << CS2));
	ERST_PORT |= (1 << RST);
	_delay_us(20);
	//Отправка команды ВЫКЛ
	LCD_Command(0x3E);
	//Установка положения по оси Y=0
	LCD_Command(0x40);
	//Установка положения по оси X=0
	LCD_Command(0xB8);
	//Установка положения по оси Z=0
	LCD_Command(0xC0);
	//Отправка команды ВКЛ
	LCD_Command(0x3F);
}

//Функция очистки дисплея
void LCD_Clear()
{
	//Выбираем первую половину дисплея
	CS_PORT &= ~(1 << CS1);
	CS_PORT |= (1 << CS2);
	for(int i = 0; i < 8; i++)
	{

		LCD_Command((0xB8) + i);
		for(int j = 0; j < 64; j++)
		{
			LCD_Data(0); //Очищаем каждый пиксель первой половины
дисплея
		}
```

```
	}
	//Выбираем вторую половину дисплея
	CS_PORT |= (1 << CS1);
	CS_PORT &= ~(1 << CS2);
	for(int i = 0; i < 8; i++)
	{

		LCD_Command((0xB8) + i);
		for(int j = 0; j < 64; j++)
		{
			LCD_Data(0);//Очищаем каждый пиксель второй половины дисплея
		}
	}
	//Установка курсора в позицию (0;0)
	LCD_Command(0x40);
	LCD_Command(0xB8);
}

// Установить курсор в позицию x, y
void LCD_GotoXY(uint8_t x, uint8_t y)
{
	//Опредления контроллера на котором находится наши координаты
	if(y<64)
	{
		//Выбор левого контроллера для отображения данных
		CS_PORT &= ~(1<<CS1);
		CS_PORT |= (1<<CS2);
	}
	else
```

```
	{
		//Выбор правого контроллера, для отображения данных
		CS_PORT &= ~(1<<CS2);
		CS_PORT |= 1<<CS1;
		y=y-64;
	}
	//Установка курсора в выбранное положение на оси X
	LCD_Command(0xB8+x);
	//Установка курсора в выбранное положение на оси Y
	LCD_Command(0x40+y);
	//Отправка команды ВКЛ
	LCD_Command(0x3F);
	LCD_Command(0xC0);
}

//Функция отображения байта данных на экране в позиции X и Y
void LCD_drawByte(uint8_t x, uint8_t y, unsigned char byte)
{
	LCD_GotoXY(x,y);
	LCD_Data(byte);
}

//Функция для отображения символа на экране в координатах x и y
void LCD_Char(uint8_t  x1, uint8_t y1, char code)
{	//Преобразование кода символа в соответствии с предопределенным пользовательским алфавитом
	if((code >= 0x20) && (code < 0x80)) code -= 32;
	else  code -= 96;

	for(uint8_t i = 0; i < 5; i++)
```

```
	{
		LCD_drawByte(x1, y1 + i, symbol[code][i]);
	}
}

//Функция для отображения строки на экране в позиции x и y
void LCD_String(uint8_t x1, uint8_t y1, char *string)
{
	while(*string)
	{
		//Посимвольное отображение каждого символа строки
		LCD_Char(x1, y1, *string++);
		y1+=6;
	}
}

//Функция инициализации датчика часов реального времени
void init_DS1307(void)
{
	TWBR = 32;
	TWSR = (0 << TWPS1)|(0 << TWPS0);
}

//Функция чтения данных из DS1307
uint8_t read_DS1307(uint8_t addr) //Передаем адрес регистра
{
	uint8_t time;
	TWCR |= (1<<TWEN);
	//формируем состояние СТАРТ
	while(!(TWCR&(1<<TWEN)));
```

```
TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
while (!(TWCR & (1<<TWINT)));
//передаем адрес и бит означающий режим записи
TWDR = 0xd0;
TWCR = (1<<TWINT)|(1<<TWEN);
while (!(TWCR & (1<<TWINT)));
// Устанавливаем необходимый адрес для чтения данных
TWDR = addr;
TWCR = (1<<TWINT)|(1<<TWEN);
//Формируем повторно сигнал старта, чтобы считать данные с необходимого регистра
while (!(TWCR & (1<<TWINT)));
TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
while (!(TWCR & (1<<TWINT)));
//Передаем адрес и бит, означающий режим чтения
TWDR = 0xd1;
TWCR = (1<<TWINT)|(1<<TWEN);
//считываем данные
while (!(TWCR & (1<<TWINT)));
TWCR = (1<<TWINT)|(1<<TWEN);
while(!(TWCR & (1<<TWINT)));
time = TWDR;
time = (((time & 0xF0) >> 4)*10)+(time & 0x0F);
//формируем состояние СТОП
TWCR = (1<<TWINT)|(1<<TWSTO)|(1<<TWEN);
while(TWCR&(1<<TWSTO));
//Отключаем шину TWI, для корректной работы экрана с портом C
TWCR &= (1<<TWEN);
_delay_us(50);
TWCR &= (1<<TWINT);
```

```
	return time;
}

//Функция записи данных в DS1307
void write_DS1307 (uint8_t reg, uint8_t time) {
	//формируем состояние СТАРТ
	TWCR |= (1<<TWEN);
	while(!(TWCR&(1<<TWEN)));
	TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
	while (!(TWCR & (1<<TWINT)));
	//передаем адрес и бит означающий режим записи
	TWDR = 0xd0;
	TWCR = (1<<TWINT)|(1<<TWEN);
	while (!(TWCR & (1<<TWINT)));
	// Устанавливаем необходимый адрес для записи данных
	TWDR = reg;
	TWCR = (1<<TWINT)|(1<<TWEN);
	while (!(TWCR & (1<<TWINT)));
	time = ((time/10)<<4) + time%10;
	//Записываем данные
	TWDR = time;
	TWCR = (1<<TWINT)|(1<<TWEN);
	while(!(TWCR & (1<<TWINT)));
	TWCR = (1<<TWINT)|(1<<TWSTO)|(1<<TWEN);
	while(TWCR&(1<<TWSTO));
	//Отключаем шину TWI, для корректной работы экрана с портом C
	TWCR &= (1<<TWEN);
	_delay_us(50);
	TWCR &= (1<<TWINT);
}
```