|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

Институт кибернетики

*(наименование института, филиала)*

Кафедра проблем управления

*(наименование кафедры)*

**КУРСОВОЙ ПРОЕКТ (РАБОТА)**

по дисциплине Микропроцессорная техника в мехатронике и робототехнике

(*наименование дисциплины)*

**Тема курсового проекта** (работы) Реализация устройства «Часы с будильником»

**Студент группы** КРБО-02-19 Гаджиев Даниял Стефанович **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*(учебная группа, фамилия, имя, отчество студента*) *(подпись студента)*

**Руководитель курсового проекта (работы)** Трипольский П.А., доцент, к.т.н.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*должность, звание, ученая степень (подпись руководителя)*

**Рецензент** (при наличии) **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*должность, звание, ученая степень (подпись рецензента)*

Работа представлена к защите «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20\_\_г.

Допущен к защите «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20\_\_г.

Оглавление

[**Постановка задачи** 3](#_Toc88233267)

[**Алгоритм работы часов** 4](#_Toc88233268)

[**Аппаратные средства** 7](#_Toc88233269)

[**Описание регистров для таймеров-счетчиков** 9](#_Toc88233270)

[**Описание регистров для внешних прерываний** 10](#_Toc88233271)

[**Описание регистров для работы с EEPROM** 11](#_Toc88233272)

[**Общие сведения о программе** 12](#_Toc88233273)

[**Описание программы:** 12](#_Toc88233274)

[**Использование программой внешних прерываний:** 13](#_Toc88233275)

[**Структура данных Регистры:** 14](#_Toc88233276)

[**Память программ:** 14](#_Toc88233277)

[**Структура подпрограмм и их назначения** 15](#_Toc88233278)

[**Структура программного обеспечения** 17](#_Toc88233279)

[**Методика и результаты тестирования** 18](#_Toc88233280)

[ **Часы точно отсчитывают время** 18](#_Toc88233281)

[ **Часы могут настраиваться пользователем через соответствующие кнопки** 19](#_Toc88233282)

[ **Будильник работает и может быть настроен пользователем и его значение сохраняется при отключении питания контроллера** 21](#_Toc88233283)

[**Вывод:** 23](#_Toc88233284)

[**Список использованной литературы:** 24](#_Toc88233285)

[**Листинг программы.** 25](#_Toc88233286)

# 

# **Постановка задачи**

Требуется реализовать устройство типа “Часы” с функцией будильника. Для вывода информации будильника и часов необходимо использовать: 4 семисегментных индикатора для отображения часов и минут, 3 кнопки - одну для изменения значения часов, вторую для изменения значения минут и третью для настройки будильника. Данные о конфигурации будильника сохраняются в  EEPROM. Во время срабатывания будильника должен загораться светодиод. Для тактирования необходимо использовать кварцевый резонатор на 32.768 КГц.

# **Алгоритм работы часов**

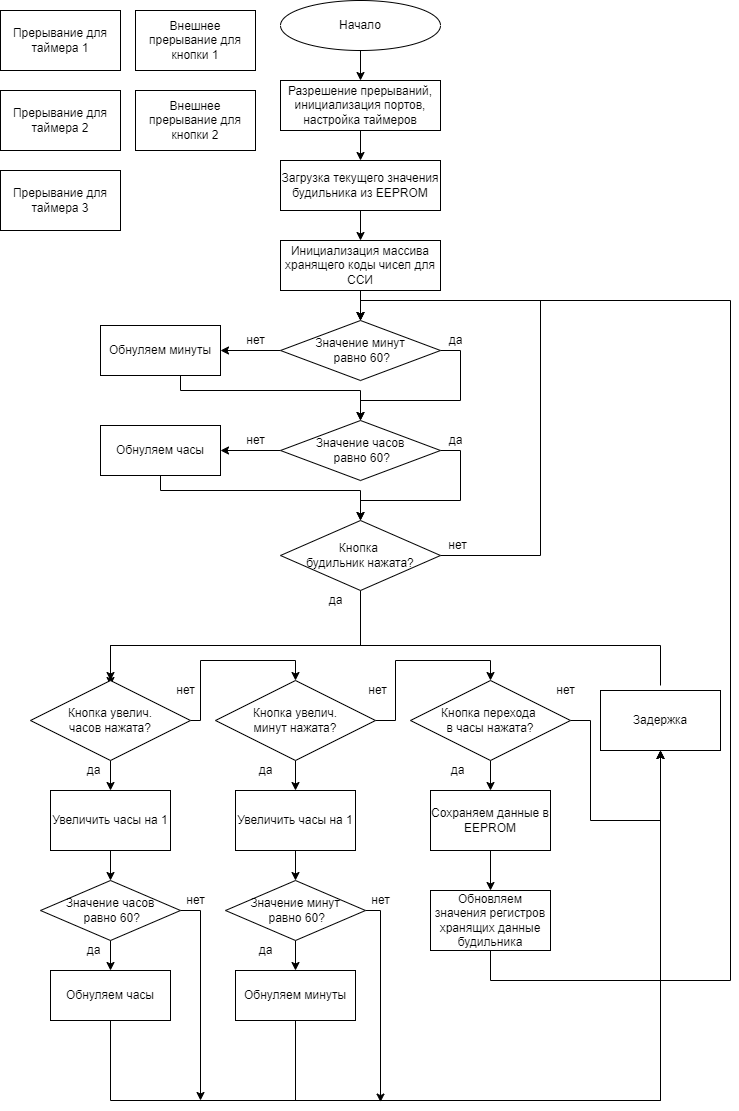


Рис. 1. Блок-схема алгоритма решения задачи

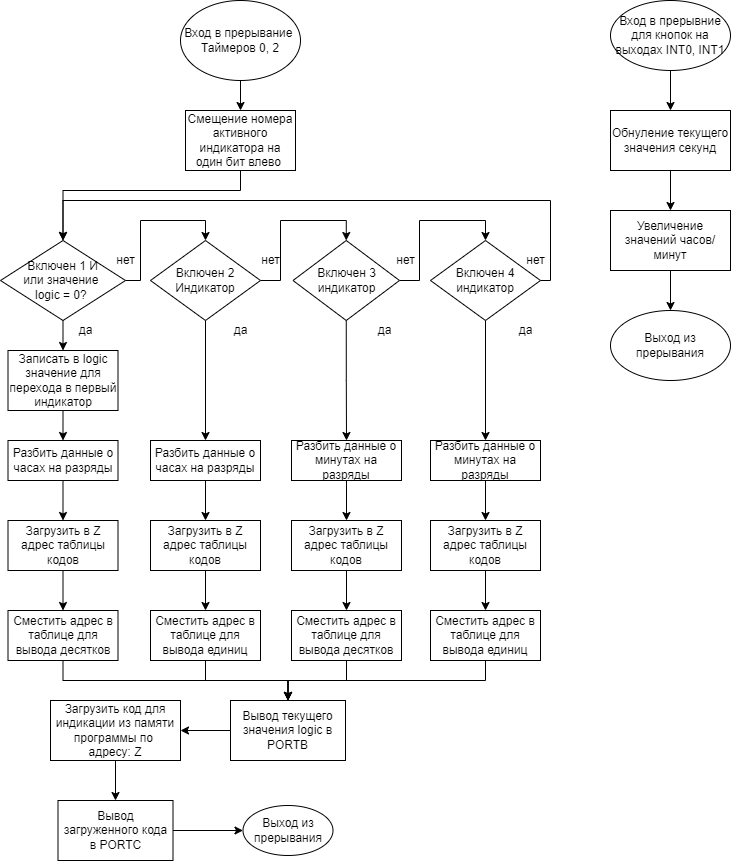
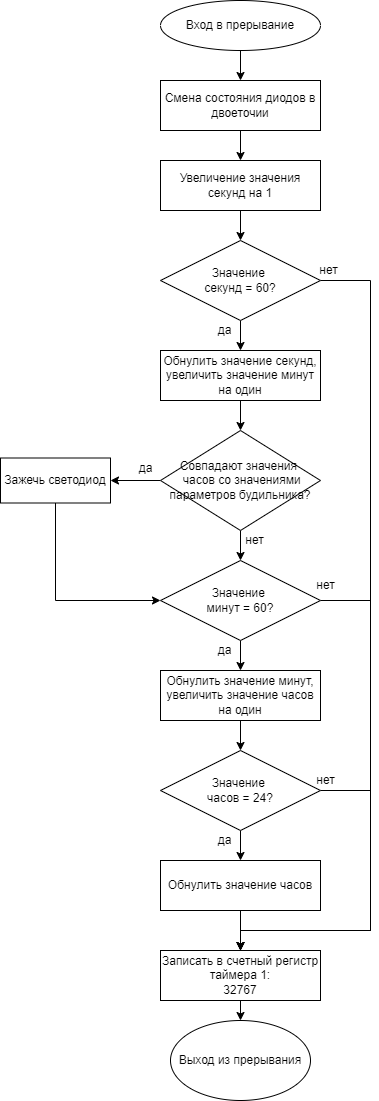
****

Рис. 2. Блок-схема алгоритма прерываний динамической индикации (Таймеры 0 и 2)

и регистров INT0, INT1 по нажатию кнопки

****Рис. 3. Блок-схема алгоритма прерываний для таймера 1

# **Аппаратные средства**

Для реализации проекта использовался микроконтроллер Atmega16. Решение выбора данного микроконтроллера было обусловлено наличием большого количества портов ввода-вывода, которые потребовались для разработки устройства.

Устройство состоит из подключенных к микроконтроллеру четырех семисегментрых индикаторов с транзисторными ключами на NPN транзисторах, трех кнопок управления(BUT1, BUT2, BUT3), зеленого светодиода LED1, который будет служить индикацией работы будильника, ключа отключения будильника KEY, двух параллельно подсоединенных красных светодиодов LED2 и LED3, которые будут отвечать за индикацию секунд, и часового кварца QUARTZ на 32768 Гц.

Для подавления дребезга контактов кнопок было использовано аппаратное решение с применением RC-цепочек. Суть данного решения заключается в том, что к портам контроллера подключены кнопки с резисторами номиналом 10кОм, а параллельно с кнопками в схему включены конденсаторы. Расчет емкости конденсаторов основан на формуле:

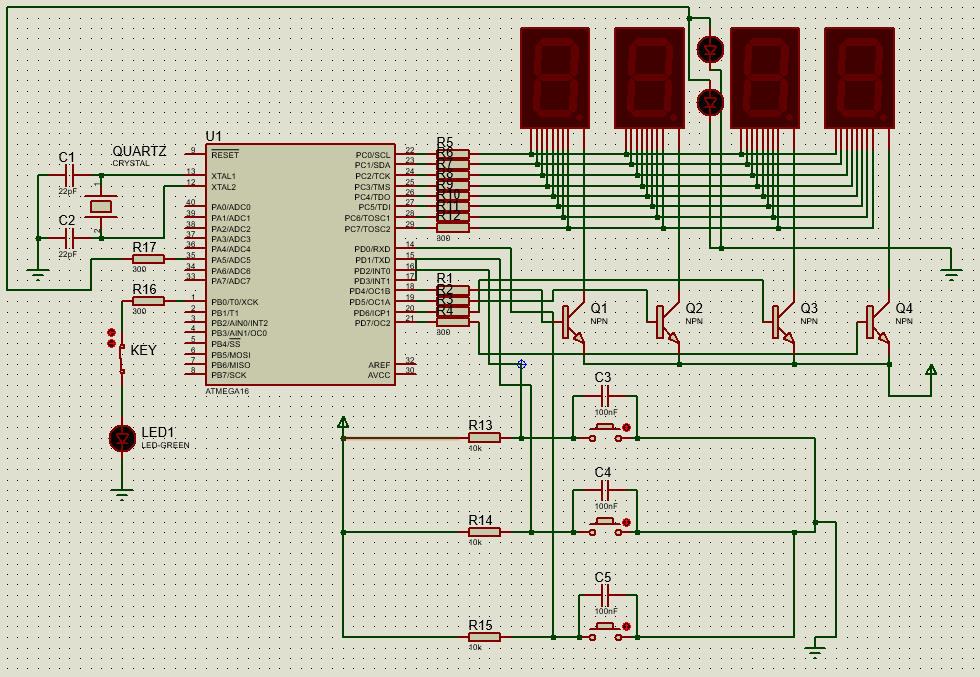
Где τ - время зарядки конденсатора, R - сопротивление резистора, C - емкость конденсатора. В данном случае время зарядки конденсатора составит 0.01 с. Для подключения светодидов и семисегментных индикаторов были использованы резисторы номиналом 300 Ом. Для расчета была использована формула.

Где – напряжение питания, равное в микроконтроллере 5В, и – падение напряжения и ток работы светодиода. В среде Proteus мы можем увидеть, что для использованных светодиодов = 2.2В, = 10 мА, а пиковое напряжение 4В (В данной программе установленные параметры напряжения и силы тока равняются 2.2 и 10мА соответственно.

Также к тактовому кварцу для стабильной работы подключены два нагрузочных конденсатора емкостью 22 пФ. (Электрическая схема устройства показана на Рис. 2. Используемые выходы микроконтроллера показаны в таблице 1)

Таблица 1. Список использованных выводов

|  |  |
| --- | --- |
| Port С  (PС7:PС0) | Шина данных для вывода чисел на семисегментные индикаторы |
| PD0/RXD | Пин приема сигнала с кнопки 3 (Вход в настройку будильника) |
| PD2/INT0 | Пин приема сигнала с кнопки 1 (Изменение количества часов) |
| PD3/INT1 | Пин приема сигнала с кнопки 2 (Изменение количества минут) |
| Port D  (PD7:PD4) | Выводы порта D для переключения активного семисегментного индикатора |
| XTAL1, XTAL2 | Вход и выход инвертирующего усилителя |
| PA5/ADC5 | Вывод контроля междуиндикаторного диодного двоеточия |
| PB0/T0/XCK | Вывод контроля будильника |

****** Рис. 4. Принципиальная схема устройства в Proteus

# **Описание регистров для таймеров-счетчиков**

Для работы с динамической индикацией и правильной работы часов требуется настроить таймеры и их прерывания. В данном проекте использованы все 3 таймера-счетчика, которые есть в микроконтроллере ATmega16 - два 8-битных (T/C0, T/C2) и один 16- битный(T/C1), 16 битный таймер подключен к кварцевому резонатору таким образом, что микроконтроллер тактируется от кварца, а как следствие остальные 2 таймера тоже.

**Необходимые регистры и их настройка:**

**TCCR1B** - Управляющий регистр B таймера 1. Используемые биты:

* **CS12:10** - три бита для настройки, отвечающие за установку предделителя и выбор источника тактового сигнала. Записываем биты следующим образом: “001” — предделитель равен 1, то есть выключен, а таймер считает тактовые импульсы.

**TCCR0** - Управляющий регистр таймера 0. Используемые биты:

* **CS02:00** - три бита для настройки, отвечающие за установку предделителя и выбор источника тактового сигнала. Записываем биты следующим образом: “001” — предделитель равен 1, то есть выключен, а таймер считает тактовые импульсы.

**TCCR2** - Управляющий регистр таймера 2. Используемые биты:

* **CS22:20** - три бита для настройки, отвечающие за установку предделителя и выбор источника тактового сигнала. Записываем биты следующим образом: “001” — предделитель равен 1, то есть выключен, а таймер считает тактовые импульсы.

**TIMSK** - Регистр маски прерываний таймеров. Используем биты:

* **TOIE0** - бит разрешения прерываний таймера-счетчика 0 по прерыванию
* **TOIE1** - бит разрешения прерываний таймера-счетчика 1 по прерыванию
* **TOIE2** - бит разрешения прерываний таймера счетчика 2 по прерыванию

**TCNT0** - Регистр счетчик таймера 0.

**TCNT1 (TCNT1L и TCNT1H)** - Младший и старший регистры счетчики таймера 1

**TCNT2 -** Регистр счетчик таймера 2

# **Описание регистров для внешних прерываний**

Для работы с изменением значений часов/минут были задействованы внешние прерывания (INT0/INT1) по нажатию на соответсвующие кнопки (BUT1/BUT2).

**Необходимые регистры:**

- **GICR** - Регистр управления маскированием внешних прерываний и размещением таблицы векторов прерываний. Используемые биты:

- **INT0** - Бит, разрешающий запрос внешнего прерывания INT0

- **INT1** - Бит, разрешающий запрос внешнего прерывания INT1

- **MCUCR** - Регистр, содержащий биты управления режимами пониженного энергопотребления MCU и распознаванием сигналов внешних прерываний INT1 и INT0. Используемые биты:

- **ISC01:0** - два бита, определяющие форму сигналов запроса внешнего прерываний INT0. Записываем биты следующим образом: “10" — запрос прерывания происходит по спадающему фронту на входе INT0.

- **ISC11:0** - два бита, определяющие форму сигналов запроса внешнего прерываний INT1. Записываем биты следующим образом: “10" — запрос прерывания происходит по спадающему фронту на входе INT1.

# **Описание регистров для работы с EEPROM**

Запись значений часов/минут будильника будет происходить в EEPROM для сохранения их после выключения питания микроконтроллера.

**Необходимые регистры:**

**EEAR(EEARL и EEARH)** - Регистр, предназначенный для адресации ячейки EEPROM, к которой будет производиться обращение.

**EEDR** - Регистр, в который при записи помещаются данные, которые требуется записать в ячейку памяти EEPROM, а при чтении помещаются данные, которые были прочитаны из ячейки. EECR - Регистр управления EEPROM. Используемые биты:

- **EERE** - Бит разрешения чтения данных EEPROM.

- **EEMWE** - Бит, определяющий, будет ли установленный в состояние 1 бит EEWE разрешать запись в EEPROM. При записи в бит “1” установка бита EEWE приведет к записи в EEPROM. Если же бит EEMWE будет находиться в состоянии “0”, то установка бита EEWE никакого эффекта не окажет.

- **EEWE** - Бит разрешения записи данных в EEPROM.

# **Общие сведения о программе**

Название: Clock

Автор: Гаджиев Даниял Стефанович

Группа: КРБО-02-19.

Дата создания: 19.11.2021.

Программа была написана на языке Assembler с использованием программных комплексов VMLab и Proteus 8 Professional. Proteus8 - использовалась для написания кода и детальной работы с регистрами и для моделирования принципиальной электрической схемы, сборки и отладки конечной версии устройства, VMLab - для отладки динамической индикации и работы с EEPROM.

## **Описание программы:**

Для начала в памяти программы создается массив, хранящий в себе ключи индикаций для ССИ. После запуска программы устанавливаются разрешения для прерывания таймеров 0 и 1 и внешние прерывания INT0 и INT1 по спадающему фронту (по нажатию). После инициализации портов из ячеек памяти EEPROM в регистры XH, XL загружается конфигурация будильника (сохраненные минуты и часы), затем идет переход в основной цикл программы.

Основной цикл программы выполняет две функции:

1) Проверяет переполнения регистров хранящих значения часов/минут при ручной установке, при достижении максимума регистрами обнуляет их.

2) Проверяет не нажата ли кнопка входа в режим конфигурации будильника. Если происходит регистрация нажатия кнопки, происходит переход в режим настройки будильника, последовательно запрещая прерывания INT0, INT1 и прерывания таймера 0. Включаются прерывания таймера 2, который в режиме установки будильника отвечает за динамическую индикацию. Сделано это потому, что таймер 0 жестко привязан к регистрам часов, а для индикации и установки будильника используются другие регистры, на которые и настроен таймер 2

После этого происходит вызов подпрограммы, являющейся вечным циклом и постоянно считывающей состояния пинов PD0, PD2, PD3. В случае высокого логического уровня на пине (нажатия на кнопки BUT1(PD2)/BUT3(PD3) соответственно), будет происходить инкремент часов/минут будильника. После инкремента программа также проверяет параметры на переполнения, в данном случае они будут происходить при превышении значений (24/60), и по достижении обнуляет соответствующие регистры. Если же пользователь нажмет на кнопку BUT2(PD0) произойдет последовательность возврата в основной цикл - разрешение прерываний INT0, INT1 по спадающему фронту, разрешение прерываний для таймера 0 и запрет для таймера 2, а также запись текущих параметров будильника в EEPROM.

**Использование программой таймеров:**

Таймер 0 используется для динамической индикации и переполняется 128 раз в секунду (128 Герц), так как он 8-ми битный, то переполняется раз в 255 тактов, а секунда проходит спустя 32768 тактов, следовательно он переполнится 128 раз за секунду, переключая индикатор и меняя выводящееся число, предварительно получив информацию и разбив полученное число на десятки и единицы.

Таймер 2 аналогичен таймеру 0, однако использует источником индикации другие регистры, так как его предназначение - индикация во время конфигурации будильника.

Таймер 1 используется для отсчета секунд. Счетный регистр данного таймера является 16-битным, вмещая 65535 тиков до переполнения. Для того чтобы прерывание при тактовании от кварцевого резонатора с частотой 32768 Гц выполнялось раз в секунду, в счетный регистр таймера 1 помещается значение 32767 (65535 - 32768 = 32767). По прерыванию таймера 1 происходит включение или отключение двух диодов между семисегментными индикаторами , затем инкремент значения секунд. После данной операции происходит постепенная проверка регистров секунд/ минут/часов на достижение максимального значения (60/60/24 соответственно). При достижении максимального значения секунд/минут инкрементируются минуты/часы. При достижении максимального значения часов все регистры обнуляются. Помимо работы со временем на таймер 1 возложена задача проверки соответствия текущего времени с временем установленного будильника. Если значения и часов, и минут совпадают, то через подпрограмму на светодиод LED1 подается питание. Таким образом, будильник будет работать ровно минуту, а потом отключится. Последним действием в прерывании в счетный регистр таймера 1 заново записывается значение 32767

## **Использование программой внешних прерываний:**

Внешние прерывания INT0 и INT1 используются, чтобы настраивать время в основном режиме работы часов. По спадающему фронту на соответствующих пинах PD2, PD3 происходит вызов прерывания и инкремент регистра содержащего часы или минуты, в зависимости от нажатой кнопки.

# **Структура данных Регистры:**

1. r0 - регистр, куда из памяти программ загружаются данные для отображения чисел
2. r15 - регистр для проверки в подпрограмме задержки
3. temp (r16) - многоцелевой регистр для временного хранения данных
4. dots (r17) - регистр контроля состояния двоеточия между индикаторами
5. digit (r18) - регистр-счетчик, использующийся в подпрограмме задержки
6. copy (r19) - регистр, использующийся для копирования неразбитого на разряды числа
7. ones (r20) - регистр для подсчета десятков при разделении чисел на разряды
8. logic (r21) - регистр для контроля логики динамической индикации
9. compare(r22) - регистр, куда помещаются максимальные возможные значения для часов
10. hrs (r23) - регистр для хранения текущего значения часов
11. min (r24) - регистр для хранения текущего значения минут
12. scd (r25) - регистр для хранения текущего значения секунд
13. XL (r26) - регистр для хранения текущего значения часов будильника
14. XH (r27) - регистр для хранения текущего значения минут будильника
15. YL (r28) - регистр для хранения числа 10, нужного в процессе разделения чисел на разряды
16. YH (r29) - регистр для подсчета единиц при разделении чисел на разряды
17. Z (r30:r31) - регистр для загрузки из памяти программы кодов для индикации
18. DDR(A, B, C, D) - регистры направления портов ввода-вывода
19. PORT(A, B, C, D) - регистры для вывода байта на порт ввода-вывода
20. PIN(A, B, C, D) – регистры для чтения логического уровня портов

## **Память программ:**

1. data - массив кодов для динамической индикации размером 10 байт
2. Ячейка EEPROM ($0000) - ячейка для хранения значения часов будильника
3. Ячейка EEPROM ($0001) - ячейка для хранения значения минут будильника

# **Структура подпрограмм и их назначения**

**Reset** - начало программы, в него происходит переход по прерыванию $000, здесь происходит инициализация стека, портов, необходимых регистров, настройка внешних прерываний и прерываний таймеров

**alarmdwnd** - подпрограмма загрузки конфигурации будильника из EEPROM

**data** - подпрограмма, на этапе прошивки загружающая данные о кодах индикации в память программ. Не может быть размещена по адресу $000, так как, по сути, не содержит программу (По сути массив).

**Proga** - основной цикл, постоянно проверяющий не достигли ли минуты и часы основного режима предельных значений, а также проверяющий не была ли нажата кнопка перехода в режим конфигурации будильника.

**mz**, **hz** - подпрограммы обнуления минут и часов в основном режиме

**alarm** - подпрограмма входа в режим настройки будильника, запрещающая внешние прерывания и прерывания таймеров 1 и 0, но разрешающая их для таймера 2

**loop** - вечный цикл, проверяющий нажатия на кнопки BUT1, BUT2, BUT3

**inch** - подпрограмма, увеличивающая количество часов в режиме будильника и проверяющая не достигли ли они конечного значения

**incm** - подпрограмма, увеличивающая количество минут в режиме будильника и проверяющая не достигли ли они конечного значения

**hzero** - подпрограмма, обнуляющая значение часов в режиме будильника

**mzero** - подпрограмма, обнуляющая значение минут в режиме будильника

**delay** – подпрограмма, инициализирующая цикл подпрограммы задержки

**long\_delay** - цикл задержки, проверяющий не отсчитал ли таймер заданное количество тиков

**refresh** - подпрограмма для инкремента счетчика, требующегося подпрограмме задержки

**away** - подпрограмма, инициализирующая выход из режима будильника посредством разрешения внешних прерываний и прерываний таймеров 1 и 0 и запретом прерываний таймера 2, а также записывающая установленные данные будильника в EEPROM

**TIM0\_OVF** - прерывание, вызывающееся по переполнение таймера 0

**TIM1\_OVF** - прерывание, вызывающееся по переполнение таймера 1 17

**TIM2\_OVF** - прерывание, вызывающееся по переполнение таймера 2

**led1, led2, led3, led4** - подпрограммы, отвечающие за отображение нужного разряда числа на семисегментных индикаторах в основном режиме

**aled1, aled2, aled3, aled4** - подпрограммы, отвечающие за отображение нужного разряда числа на семисегментных индикаторах в режиме настройки будильника

**split\_func** - подпрограмма, инициализирующая переменную-счетчик и вызывающая цикл разбиения числа на разряды

**div** - цикл разбиения числа на разряды и получения числа десятков

**result** - подпрограмма, производящая вычисление числа единиц

**Zix** - подпрограмма, выполняющая переключение активного индикатора и вывод нужного числа по коду на данный индикатор

**minutes** - подпрограмма, проверяющая не достигли ли минуты основного режима предельного значения, а также вызывающая подпрограмму проверки будильника

**hours** - подпрограмма, проверяющая не достигли ли часы основного режима предельного значения **alarmcheck** - подпрограмма, проверяющая совпадают ли минуты основного режима с минутами установленного будильника

**sndstep** - подпрограмма, проверяющая совпадают ли часы основного режима с часами установленного будильника

**Vix** - подпрограмма загрузки необходимого значения в счетный регистр таймера 1

**EXT\_INT0** - прерывание, вызывающееся в основном режиме по спадающему фронту на PIND2. В прерывании производится инкремент значения часов основного режима

**EXT\_INT1** - прерывание, вызывающееся в основном режиме по спадающему фронту на PIND3. В прерывании производится инкремент значения минут основного режима

# **Структура программного обеспечения**

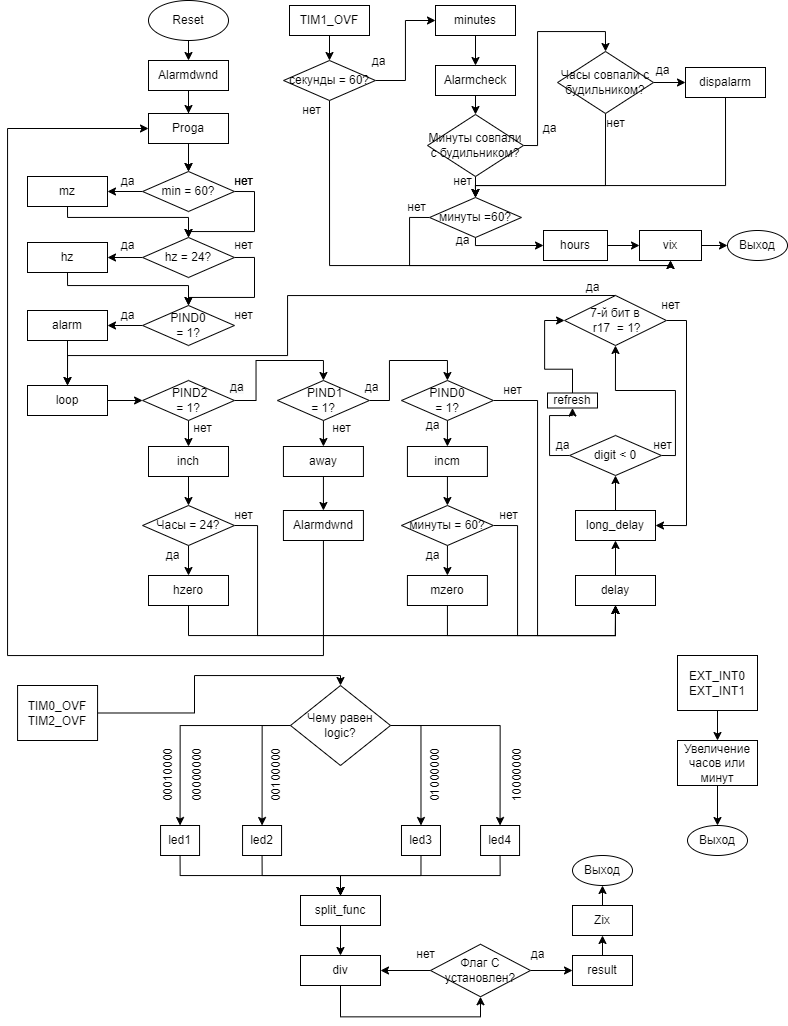


Рис. 5. Структурная схема программы

# **Методика и результаты тестирования**

Тестирование проекта производилось в среде Proteus 8. Для проверки устройства необходимо проверить работоспособность основных функций часов:

* **Часы точно отсчитывают время**

Для проверки запустим часы и проверим изменится ли значение минут на индикаторах по прошествии одной минуты. Для контроля времени был использован внутренний таймер Proteus:

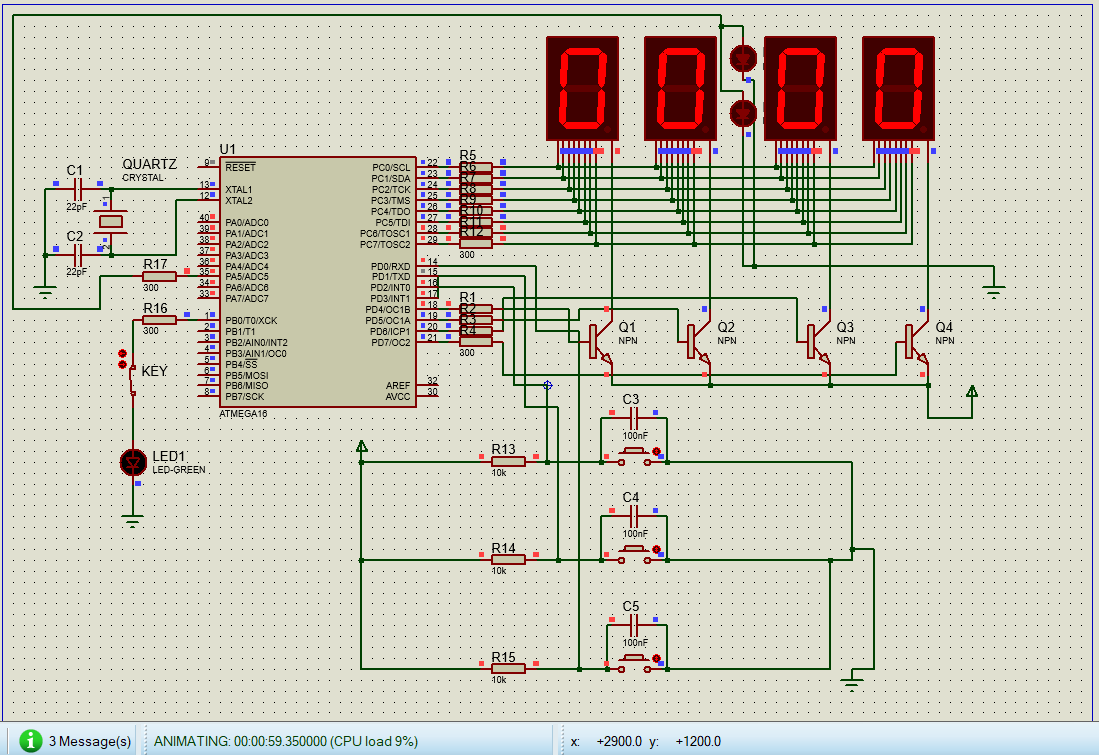
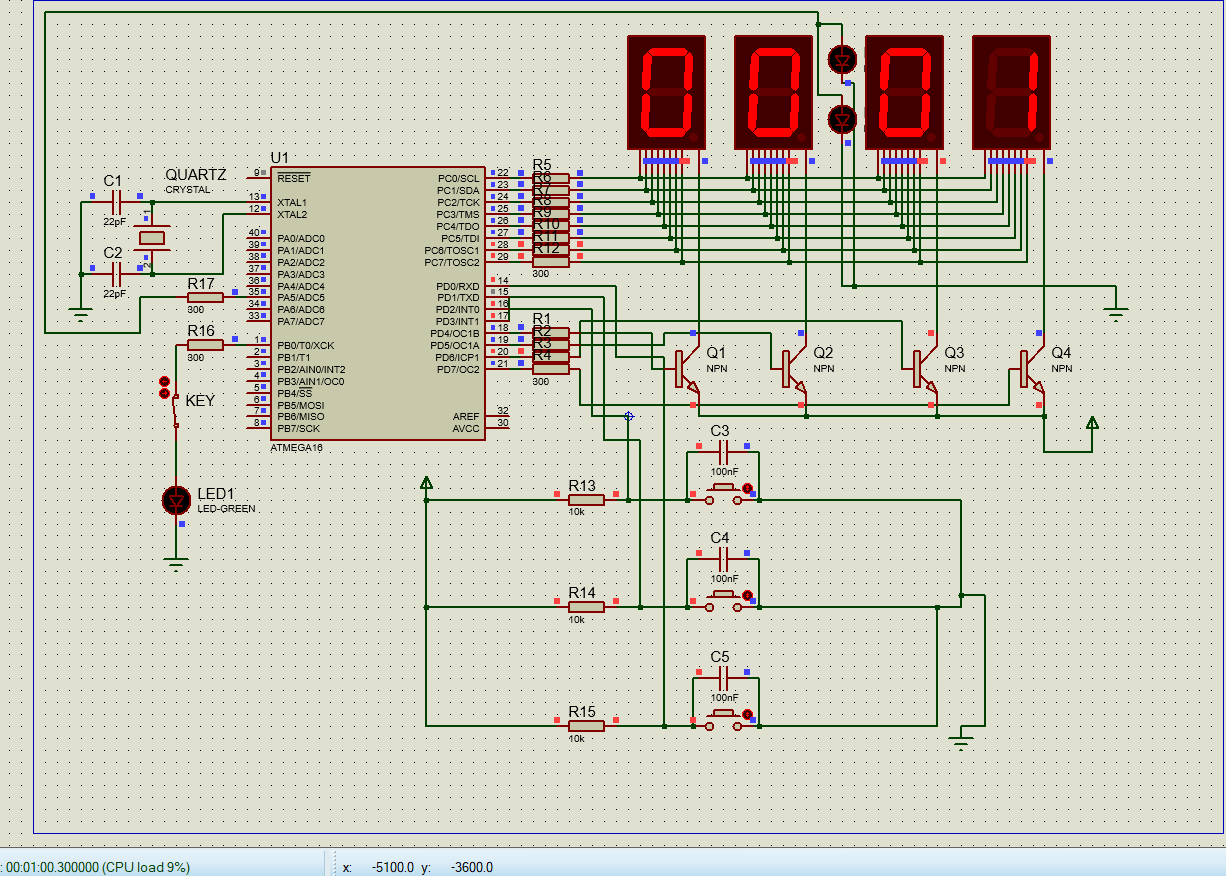


Рис. 6. Внутренние часы отсчитали чуть больше 59 секунд, а значение на часах не изменилось.

Как видно на Рис.7, при достижении минуты, число, выводимое на индикаторы, изменилось корректно, также можно заметить, что двоеточие через секунду изменило свое состояние в противоположное (из активного в неактивное), что говорит о том, что оно меняет свое состояние каждую секунду

******Рис. 7. Внутренние часы отсчитали 60 секунд, значение на часах поменялось корректно.

* **Часы могут настраиваться пользователем через соответствующие кнопки**

Для проверки запустим часы и проверим изменится ли значение часов при нажатии на BUT1 и значение минут при нажатии на BUT2:

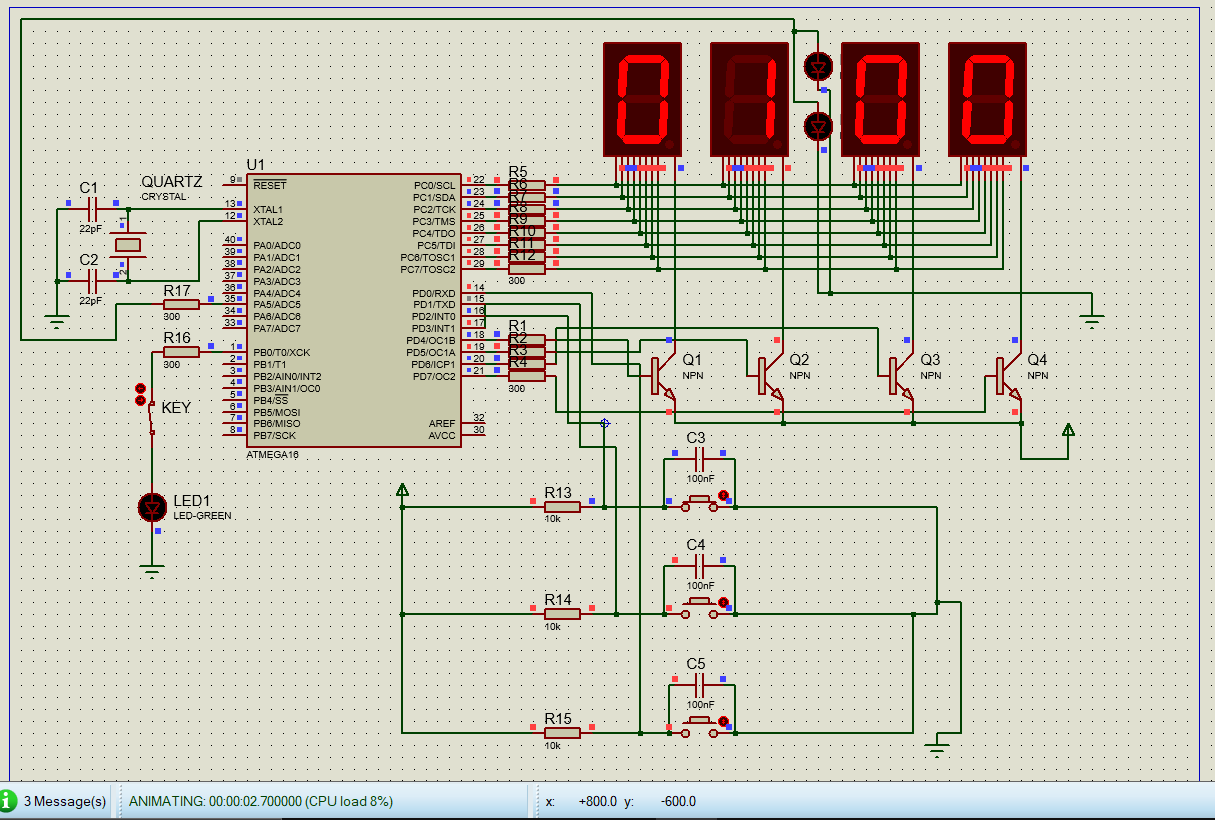
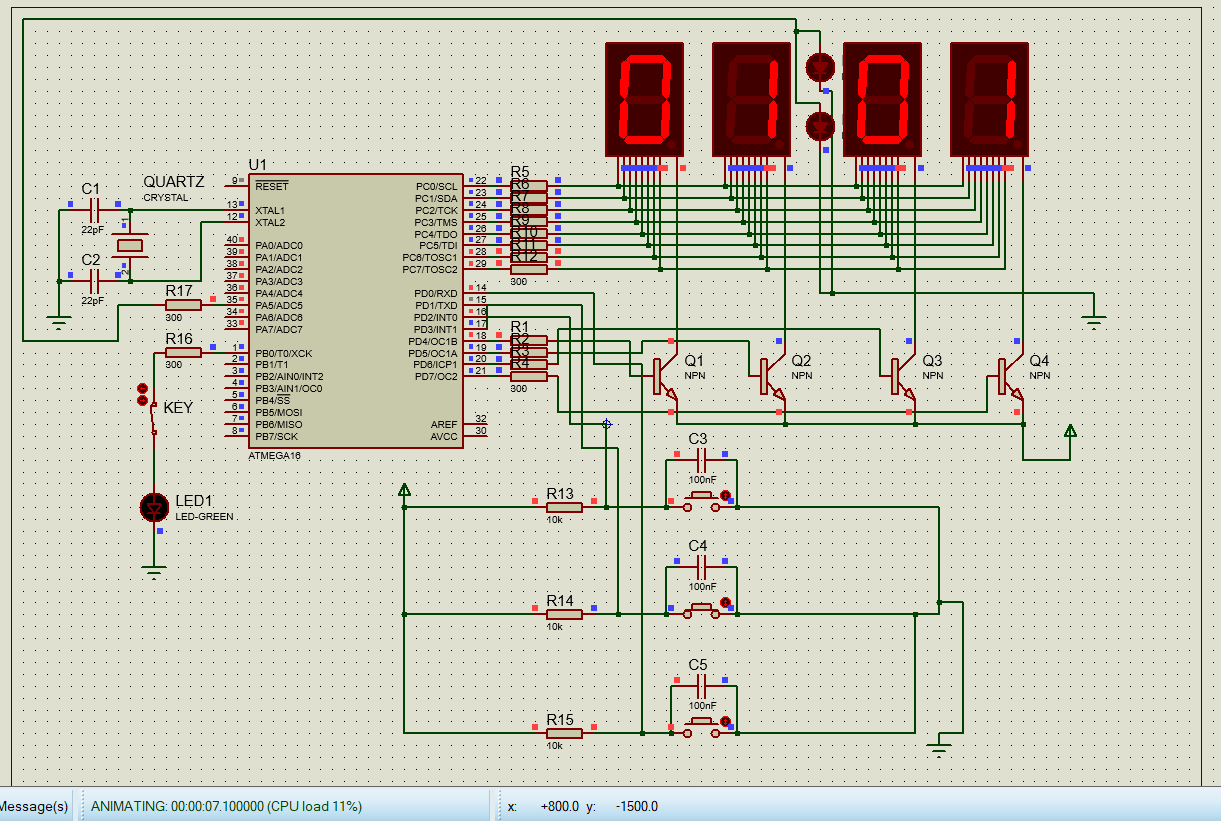
Рис. 8. При нажатии BUT1 значение часов увеличилось

Рис. 9. При нажатии BUT2 значение минут увеличилось

* **Будильник работает и может быть настроен пользователем и его значение сохраняется при отключении питания контроллера**

Для этого запустим устройство, нажмем на BUT3 для входа в режим конфигурации будильника и кнопками установим будильник, например, на 12:15 . После этого выйдем в основной режим и установим время на 12:14. По прошествии минуты время изменится и сработает индикатор будильника - зеленый светодиод:

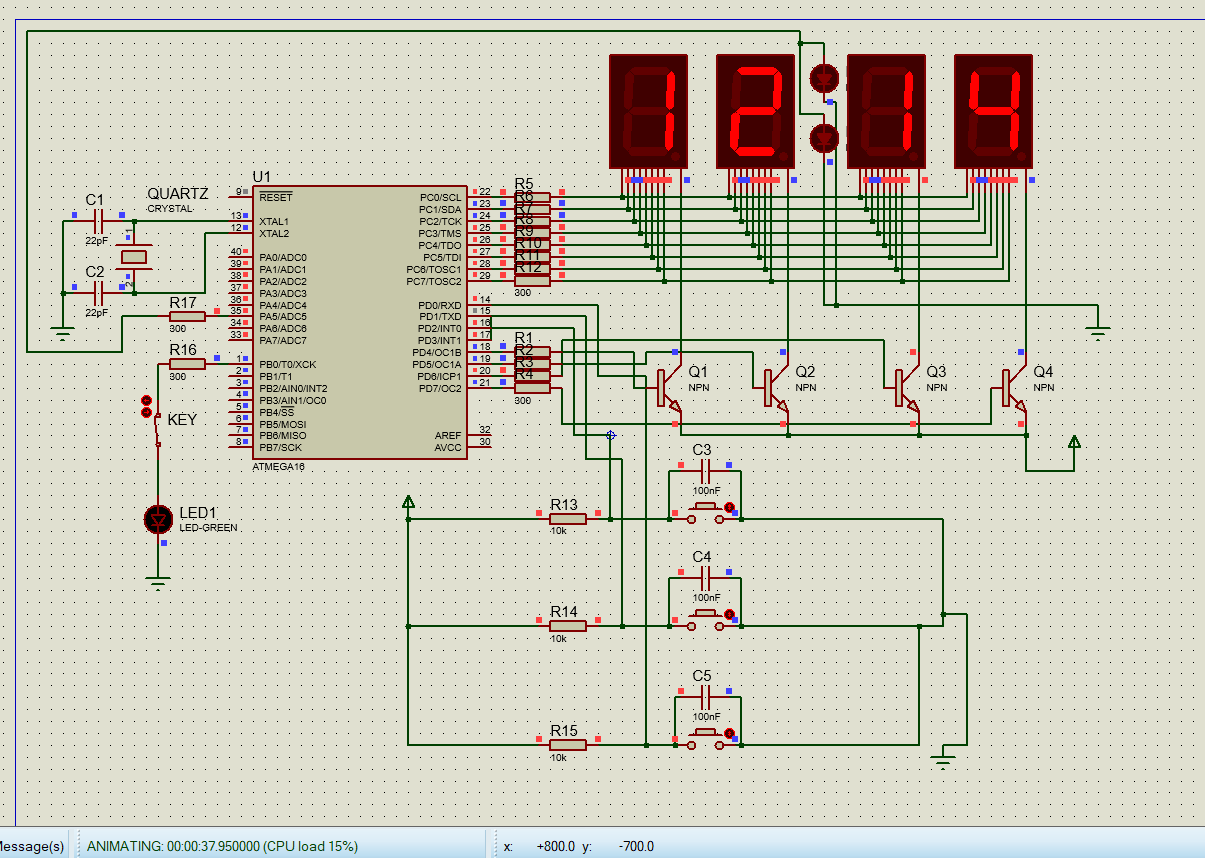


Рис. 10. Время часов не совпало со временем будильника

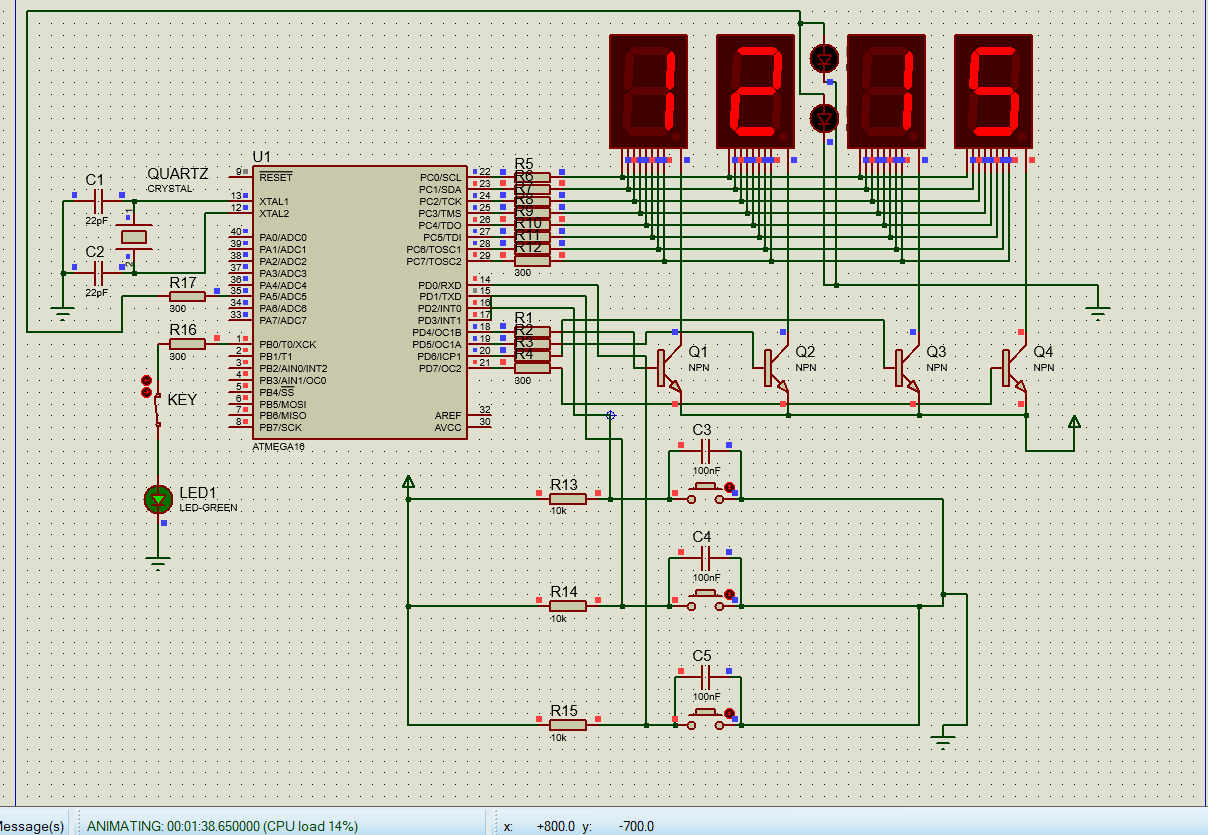


Рис. 11. При нажатии BUT2 значение минут увеличилось

Также, поскольку каждый раз при выходе из настроек конфигурации будильника мы сохраняем данные в EEPROM, то данные сохраняются в памяти даже после отключения питания, что можно увидеть на рисунке 12

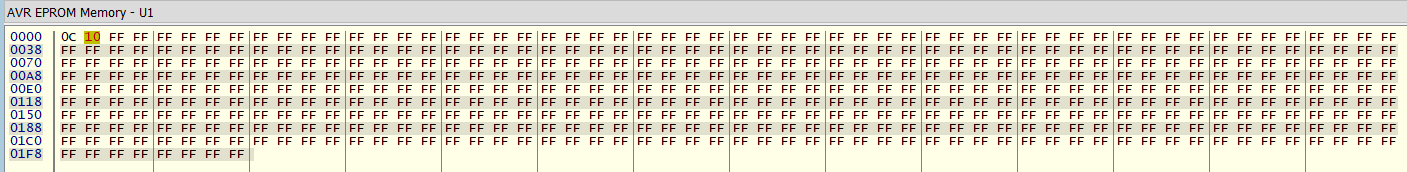


Рис. 12. Данные сохраненные в EEPROM

# **Вывод:**

В результате было разработано аппаратное и программное решение для устройства «Часы с будильником». Проект включает в себя код написанный на языке программирования Assembler, отвечающую за обработку нажатий кнопок, настройкку будильника и часов, а ткже вывод правильных значений на 4 семисегментных индикатора, и собранную в Proteus8 схему, отвечающую за подачу напряжения на соответствующие порты, а также корректную работу таймеров. Схема была реализована на основе микроконтроллера ATMEGA16 и прочих внешних устройств.

# **Список использованной литературы:**

1. ATmega16 datasheet [Электронный ресурс]. –URL: http://ww1.microchip.com/

downloads/en/devicedoc/doc2466.pdf

1. Accessing The Timer Count [Электронный ресурс]. –URL: https://www.avrfreaks.net/

forum/noob-question-accessing-timer-count

1. AVR Timers - Timer 2 [Электронный ресурс]. –URL: http://maxembedded.com/

2011/06/avr-timers-timer2/

1. Loading data from programs memory [Электронный ресурс]. –URL: <http://www.rjhcoding.com>

/avr-asm-pm.php

1. Conditional branching [Электронный ресурс]. –URL: http://www.rjhcoding.com/avrasm-program-

flow.php

1. Crystal Oscillator 32kHz [Электронный ресурс]. –URL: https://www.avrfreaks.net/

forum/crystal-oscillator-32-khz

1. Low frequency oscillator [Электронный ресурс]. –URL: https://www.avrfreaks.net/

forum/low-frequency-external-oscillator

# 

# **Листинг программы.**

.include "m16def.inc"

.def temp = r16

.def dots = r17

.def digit = r18

.def copy = r19

.def ones = r20

.def logic = r21

.def compare = r22

.def hrs = r23

.def min = r24

.def scd = r25

.cseg

.org $000

jmp Reset

.org $002

jmp EXT\_INT0

.org $004

jmp EXT\_INT1

.org $008

jmp TIM2\_OVF

.org $010

jmp TIM1\_OVF

.org $012

jmp TIM0\_OVF

Reset:

ldi temp, high(RAMEND) ; Stack init

out sph, temp

ldi temp, low(RAMEND)

out spl, temp

ldi temp, 0xFF ; Ports settings

out DDRA, temp

out DDRB, temp

out DDRC, temp

out PORTC, temp

ldi temp, 0 ;**Вспомогательный регистр проверки**

mov r15, temp

ldi temp, 0b11110000; **Говорим какие порты будут работать на подачу напряжения ССИ**

out DDRD, temp

ldi temp, 0b00000001 ;**Все три таймера работают в нормальном режиме**

out TCCR1B, temp

out TCCR0, temp

out TCCR2, temp

ldi temp, 0b00000101 ;**Разрешаем прерывания таймеров 1, 0**

out TIMSK, temp

ldi temp, 0x7F ;**Помещаем в счетчик значение 32767 чтобы до переполнения(значения**

out TCNT1H, temp ;**65535 он совершил 32768 тактов) прошла секунда**

ldi temp, 0xFF

out TCNT1L, temp

ldi scd, 0 ;**Начальные значения сек, минут, часов**

ldi min, 0

ldi hrs, 0

ldi dots, 0xFF ; for PORTA

ldi logic, 0b00010000 ; **Для Динамической индикации стартовое значение**

out PORTD, logic

ldi temp, 0b11000000 ; **Внешние прерывания INT1 - INT0 разрешены**

out GICR, temp

ldi temp, 0b00001010 ;**Прерывния INT0, INT1 по спадающему фронту**

out MCUCR, temp

rcall alrmdwnd

sei

rjmp data

rjmp Proga

;-------------------------------------------------------------------------------------------------

alrmdwnd: **; Подпрограмма для записи данных будильника в XL, XH**

ldi temp, 0x00

out EEARH, temp

out EEARL, temp

sbi EECR, EERE

in XL, EEDR

rcall wait

inc temp

out EEARL, temp

sbi EECR, EERE

in XH, EEDR

rcall wait

ret

data: ; **массив данных**

.db 0b11000000, 0b11111001, 0b10100100, 0b10110000,0b10011001, 0b10010010, 0b10000010, 0b11111000,0b10000000, 0b10010000

ret

;-------------------------------------------------**Основная чать программы**

Proga:

cpi min, 60

breq mz

cpi hrs, 24

breq hz

sbis PIND, 0 ;**Переход врежим часов**

rcall alarm

rjmp Proga

hz: ;Сброс часов

clr hrs

rjmp Proga

mz: ; Сброс минут

clr min

rjmp Proga

;

;------------------------------------------------------------**Будильник**

alarm:

cli; **отключаем внешние прерывания, сделано для отключения INT0, INT1**

ldi temp, 0b00000000 ;**Запрещаем внешние прерывания INT0, INT1**

out GICR, temp

ldi temp, 0b01000100 ; **Прерывания для 0 таймера запрещены, для 1 и 2 разрешены**

out TIFR, temp

sei

rcall loop

ret

loop: ;**Обрабатываем нажати на кнопки**

sbis PIND, 2

rcall inch

sbis PIND, 3

jmp away

sbis PIND, 0

rcall incm

rcall delay

rjmp loop

inch:

inc XL

cpi XL, 24

breq hzero

rcall delay

ret

incm:

inc XH

cpi XH, 60

breq mzero

rcall delay

ret

hzero:

clr XL

ret

mzero:

clr XH

ret

delay:

ldi digit, 0x00

mov r15, digit

ldi digit, 0xff

rcall long\_delay

long\_delay: **;Цикл задержки между обработчиком нажатий кнопки и нажатием кнопки**

dec digit ;**Происходит уменьшение с 255 до -1, и увеличение r15 с 0 до 128, всего**

cpi digit, 0 ; **128\*256 32768 тактов, так что секунда.**

**breq refresh**

sbrc r15, 7

ret

rjmp long\_delay

refresh:

mov digit, r15

inc digit

mov r15, digit

ldi digit, 0xff

ret

away:

cli

ldi temp, 0b11000000 ; **Внешние прерывания INT1 - INT0 разрешены**

out GICR, temp

ldi temp, 0b00001010 **; Прерывания по спадающему фронту для INT0, INT1**

out MCUCR, temp

ldi temp, 0b00000101 ; **Interrupts for timer 1 and 0 are ON**

out TIMSK, temp

;-----------------------------**EEPROM Запись конфигурации будильника**

out EEDR, XL ;**Назначаем адреса записи в еепром**

ldi XL, 0x00

out EEARH, XL

ldi XL, 0x00

out EEARL, XL

sbi EECR, EEMWE ;**Ставим флаг разрешения записи данных**

sbi EECR, EEWE

rcall wait

out EEDR, XH

ldi XH, 0x00

out EEARH, XH

ldi XH, 0x01

out EEARL, XH

sbi EECR, EEMWE

sbi EECR, EEWE

rcall wait

rcall alrmdwnd ; **Вызов подпрограммы для обновления данных в регистрах, XH, XL**

sei ;которые временно использовались как вспомогательные.

ret

wait: ;**Подпрограмма ожидания записи данных в EEPROM**

sbic EECR, EEWE

rjmp wait

ret

alarmcheck: ;**Проверяем совпадение данных часов с данными будильника**

cp min, XH ;**Проверяем совпадение минут**

breq sndstep

ret

sndstep:

cp hrs, XL ;**Проверяем совпадение часов**

breq dispalarm

ret

dispalarm:

ldi temp, 0xff

out PORTB, temp ;**Зажигаем светодиод на минуту**

ret

;---------------------------------------------------------------------------------

TIM1\_OVF:

cli

out PORTA, dots : ;**Смена состояния двоеточия с 1 на 0 на 1**

ldi temp, 0xff

eor dots, temp

inc scd

ldi compare, 60 **;Проверка секунд на переполнение**

cp scd, compare

breq minutes

Vix: ; **Перезаписываем в 32767 т.к они обнуляются**

ldi temp, 0x7F

out TCNT1H, temp

ldi temp, 0xFF

out TCNT1L, temp

sei

ret

minutes:

ldi temp, 0x00 ;**Поле срабатывания будильника через минуту гасим светодиод**

out PORTB, temp

clr scd

inc min

rcall alarmcheck ;**Проверка на совпадение текущих значений минут и часов с данными**

cp min, compare ; **будильника сохраненными в EEPROM**

breq hours

rjmp Vix

hours: ;**Проверка на переполнение часов**

clr min

inc hrs

ldi compare, 24

cp hrs, compare

breq zeroing

rjmp Vix

zeroing:

clr hrs

;------------------------------------------------------------------------------------------

TIM0\_OVF:

cli

lsl logic ;**Делаем сдвиг для смены индикатора который будем отображать**

cpi logic, 0b00010000 ;**Проверяем индикацию какого ССИ будем обновлять**

breq led1

cpi logic, 0b00100000

breq led2

cpi logic, 0b01000000

breq led3

cpi logic, 0b10000000

breq led4

cpi logic, 0b00000000

breq led1

out PORTC, temp

sei

reti

Zix:

out PORTD, logic

lpm ;**Сохраняем в r0 значение из дата по адресу ZH, ZL**

out PORTC, r0 ; **Выводим число на ССИ соответсвующее коду в r0**

sei

ret

led1:

ldi logic, 0b00010000

mov copy, hrs

mov r29, copy

rcall split\_func

ldi ZH, high(data\*2) ;**Достаем значения из массива Data по адресу ZH, ZL**

ldi ZL, low(data\*2)

add ZL, ones ;**Адрес ZH всегда нулевой, так как одна размерность у массива ZL зависит от**

rjump Zix ; **цифры которую надо индицировать**

led2:

mov copy, hrs

mov r29, copy

rcall split\_func

ldi ZH, high(data\*2)

ldi ZL, low(data\*2)

add ZL, r29

rjmp Zix

led3:

mov copy, min

mov r29, copy

rcall split\_func

ldi ZH, high(data\*2)

ldi ZL, low(data\*2)

add ZL, ones

rjmp Zix

led4:

mov copy, min

mov r29, copy

rcall split\_func

ldi ZH, high(data\*2)

ldi ZL, low(data\*2)

add ZL, r29

rjmp Zix

;----------------------------------------**Разбиение на десятки и единицы**

split\_func:

ldi ones, 0

rcall div

ret

div:

subi copy, 10 ;**Вычитаем 10 пока не получим отрицательное число,**

brcs result ;**при получении отрицательного числа сохраняем кол-во десятков**

inc ones

rjmp div

result:

ldi r28, 10

mul ones, r28

sub r29, r0 ;**Вычитаем из полного значения десятки чтобы в r29 созранить единицы.**

ret

;-------------------------------------------------------------------

TIM2\_OVF:

cli

lsl logic

cpi logic, 0b00010000

breq aled1

cpi logic, 0b00100000

breq aled2

cpi logic, 0b01000000

breq aled3

cpi logic, 0b10000000

breq aled4

cpi logic, 0b00000000

breq aled1

out PORTC, temp

reti

sei

reti

aled1:

ldi logic, 0b00010000

mov copy, XL

mov r29, copy

rcall split\_func

ldi ZH, high(data\*2)

ldi ZL, low(data\*2)

add ZL, ones

rjmp Zix

aled2:

mov copy, XL

mov r29, copy

rcall split\_func

ldi ZH, high(data\*2)

ldi ZL, low(data\*2)

add ZL, r29

rjmp Zix

aled3:

mov copy, XH

mov r29, copy

rcall split\_func

ldi ZH, high(data\*2)

ldi ZL, low(data\*2)

add ZL, ones

rjmp Zix

aled4:

mov copy, XH

mov r29, copy

rcall split\_func

ldi ZH, high(data\*2)

ldi ZL, low(data\*2)

add ZL, r29

rjmp Zix

reti

;----------------------------------------**Кнопки**

EXT\_INT0:

cli

inc hrs

ldi scd, 0

sei

reti

EXT\_INT1:

cli

inc min

ldi scd, 0

sei

reti