|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

Факультет: «Специальное машиностроение»

Кафедра: СМ11 «Подводные аппараты и роботы»

**Лабораторная работа № 2**

по курсу «Электронные устройства мехатронных и робототехнических систем»

Вариант 8

Выполнил: Ионин Даниил

Группа: СМ11-61Б

Проверил(а):

Москва, 2024 г.

ОГЛАВЛЕНИЕ

[ОГЛАВЛЕНИЕ 2](#_Toc165834569)

[Введение 3](#_Toc165834570)

[Тема работы: 3](#_Toc165834571)

[Цель работы: 3](#_Toc165834572)

[Теоретическая часть 4](#_Toc165834573)

[Память 4](#_Toc165834574)

[Порты ввода/вывода 4](#_Toc165834575)

[Условные обозначения 6](#_Toc165834576)

[Блок схема ATMEGA16 7](#_Toc165834577)

[Работа со стендом EasyAVR5 8](#_Toc165834578)

[Практическая часть 10](#_Toc165834579)

[Постановка задачи: 10](#_Toc165834580)

Введение

Тема работы:

Рассмотрение принципов работы микроконтроллеров при помощи стенда EAZYAVR 5

Цель работы:

Изучить функциональные возможности учебно-отладочного стенда, внутреннюю структуру и систему команд МК ATmega16. Получить первичные навыки программирования МК ATmega16.

Теоретическая часть

Память

Память микроконтроллеров типа AVR распределяется на два ключевых сегмента: область хранения информации и пространство для программного кода. Данная модель предполагает наличие дополнительной памяти EEPROM, которая обеспечивает независимость от источника питания при сохранении данных.

Память микроконтроллера Atmega16 типа AVR представляет собой область, предназначенную для временного хранения переменных и данных, необходимых для выполнения операций, предусмотренных программой. Эта память является энергозависимой, что означает, что её содержимое теряется при отключении питания устройства. Она обладает высокой скоростью доступа, и организована как 8 кбайт х 16.

Порты ввода/вывода

В составе данного МК имеет четыре 8-разрядных двунаправленных порта.

Для обслуживания каждого порта отведено три регистра: регистр данных PORTx, регистр направления данных - DDRx и регистр выводов порта PINx (вместо х подставляется соответственно буквы A, B, C, D). Регистр выводов порта предназначен только для чтения, в то время как регистр данных и регистр направления данных - для чтения/записи.

Все выводы портов имеют отдельно подключаемые подтягивающие резисторы. Выходы порта могут поглощать ток до 20 мА и непосредственно управлять светодиодными индикаторами. Выводы используются как входы и замыкаются на землю, если включены внутренние подтягивающие резисторы, при этом выводы являются источниками тока (IIL).

Все 8 бит каждого из портов при использовании для ввода/вывода одинаковы.

Бит DDxn регистра DDRx выбирает направление передачи данных. Если бит установлен (1), вывод сконфигурирован как выход. Если бит сброшен (0) - вывод сконфигурирован как вход. Если PORTxn установлен и вывод сконфигурирован как вход, включается КМОП подтягивающий резистор. Для отключения резистора, PORTxn должен быть сброшен (0) или вывод должен быть сконфигурирован как выход (см. таблицу 1).

Таблица 1 – Влияние DDxn на выводы порта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DDxn | PORTxn | Вх/Вых | Подт.резист | Комментарий |
| 0 | 0 | Вход | Нет | Третье состояние (Hi-Z) |
| 0 | 1 | Вход | Да | Pxn источник тока IIL, если извне соединен с землей |
| 1 | 0 | Выход | Нет | Выход установлен в 0 |
| 1 | 1 | Выход | Нет | Выход установлен в 1 |

n = 7,6...0 - номер вывода.

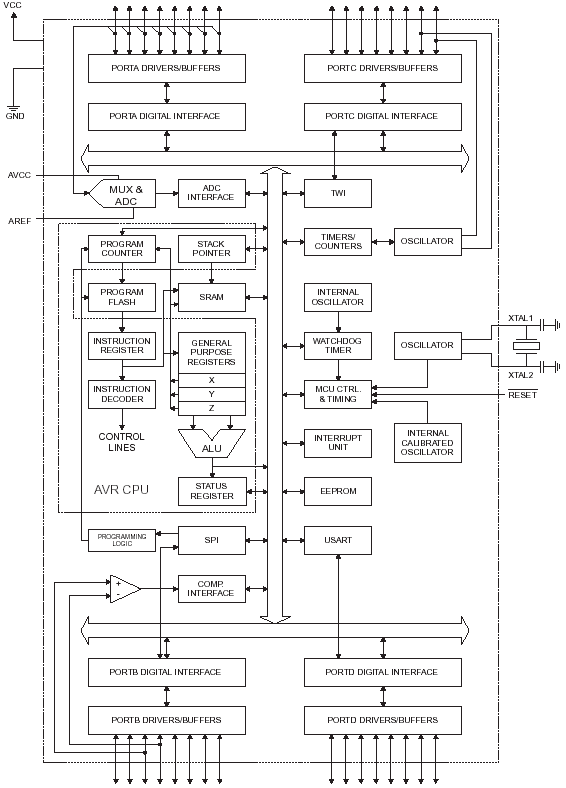
Стенд Easyavr5 представляет собой лабораторный макет с микроконтроллером. К каждому выводу микроконтроллера подключена кнопка и светодиод. На макете установлено четыре семисегментных индикатора. Также имеется возможность подключения к макету внешних устройств через специальные устройства. Питание макета осуществляется либо от внешнего источника напряжения 8–16В, либо через USB-порт. Кроме этого, на макете установлен внутрисхемный программатор.

В лабораторных работах будет использоваться микроконтроллер фирмы ATMEL – ATmega16.

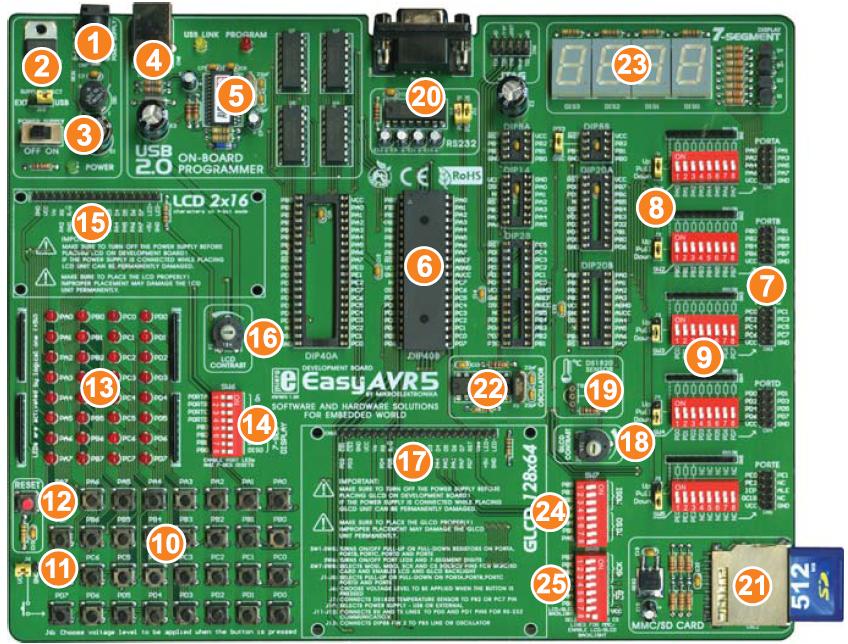
Условные обозначения

* Flash ROM - объем энергонезависимой памяти программ (в килобайтах);
* EEPROM - объем энергонезависимой памяти данных (в байтах);
* RAM - объем статической памяти данных (в байтах);
* External RAM - возможность подключения к микроконтроллеру дополнительной микросхемы внешней статической памяти данных (в килобайтах);
* ISP - возможность программирования микроконтроллера в системе (на целевой плате) при основном напряжении питания;
* SPM - функция самопрограммирования Flash ROM памяти микроконтроллера в системе без участия внешнего программатора;
* JTAG - встроенный JTAG - интерфейс;
* I/O (pins) - максимальное количество доступных линий ввода / вывода;
* Timer(s) 8/16 bit - количество и разрядность таймеров/счетчиков;
* USI - универсальный коммуникационный интерфейс;
* AC - аналоговый компаратор;
* ADC (channels) - количество каналов аналого-цифрового преобразования;
* Internal RC - наличие внутренней RC-цепочки для автономной работы микроконтроллера (без внешнего источника опорной частоты);
* WDT - сторожевой таймер;
* BDC - аппаратный программируемый блок защиты от сбоев при внезапном (в том числе и кратковременном) пропадании напряжения питания микроконтроллера;
* UART - асинхронный последовательный приемопередатчик;
* SPI - синхронный трехпроводной последовательный интерфейс;
* I2C - двухпроводной последовательный интерфейс;
* RTC - система реального времени;
* PWM (channels) - количество независимых каналов широтно - импульсной модуляции;
* Command Set - количество различных инструкций в системе команд микроконтроллера;
* Vcc - диапазон рабочих напряжений питания (в Вольтах);
* Clock - диапазон рабочих частот (в мегагерцах);

Блок схема ATMEGA16

 Рисунок 1 – Блок схема ATmega16

Работа со стендом EasyAVR5

 Рисунок 2 – Внешний вид стенда EasyAVR5

1 – разъем для подключения внешнего питания 8–16В;

2 – выбор между внешним питанием и питанием от USB;

3 – выключатель;

4 – разъем USB;

5 – встроенный внутрисхемный программатор;

6 – панельки для установки различных типов МК;

7 – разъемы для доступа к портам МК;

8 – джамперы для выбора типа подтягивающих резисторов (к 1 или к 0);

9 – группы переключателей для подключения подтягивающих резисторов к выводам МК;

10 – 32 кнопки, подключенные к выводам МК;

11 – джампер выбора высокого или низкого состояния при нажатии кнопки:

12 – кнопка сброса;

13 – светодиоды, каждый подключенный к выводу МК;

14 – группа переключателей, подключающая или отключающая светодиоды к МК;

15 – разъем для подключения символьной ЖК-матрицы;

16 – потенциометр для регулировки яркости символьной ЖК-матрицы;

17 – разъем для подключения графической ЖК-матрицы;

18 – потенциометр для регулировки яркости графической ЖК-матрицы;

19 – разъем для подключения датчика температуры DS1820;

20 – разъем порта RS232;

21 – слот для подключения MMC/SD карт памяти;

22 – встроенный генератор частоты;

23 – 4 семисегментных индикатора;

24 – группа переключателей для разрешения внутрисхемного программирования;

25 – группа переключателей для разрешения работы с MMC/SD картами памяти.

Практическая часть

Постановка задачи:

1. Выполните проверку работы программы в симуляторе AVR Studio. Измерьте в симуляторе время, в течение которого поочередно включен каждый семисегментный индикатор;
2. Выполните: программирование макетной платы. Проверьте работу платы в 3-х режимах:

- нажать кнопку PD1-«готовность» после загорания CИД(PB7) нажать PD2 "реакция"

- нажать кнопку PD1-«готовность», до загорания CИД(PB7) нажать PD2 "реакция"

- нажать кнопку PD1-«готовность» после загорания CИД(PB7) не нажимать PD2 "реакция".

1. Найдите и исправьте ошибку в программе;
2. Доработайте программу, увеличив счетчик дисплея Disp\_Count до 2-х байт с максимальным значением 0xFFFF, введя дополнительный байт Disp\_CountH. Запрограммируйте макетную плату;
3. Объясните работу схемы. Экспериментально подберите константу Val\_dispCount, определяющую время включения индикатора, когда глаз не воспринимает мигание индикаторов. Измерьте в симуляторе время, в течение которого поочередно включен каждый семисегментный индикатор.

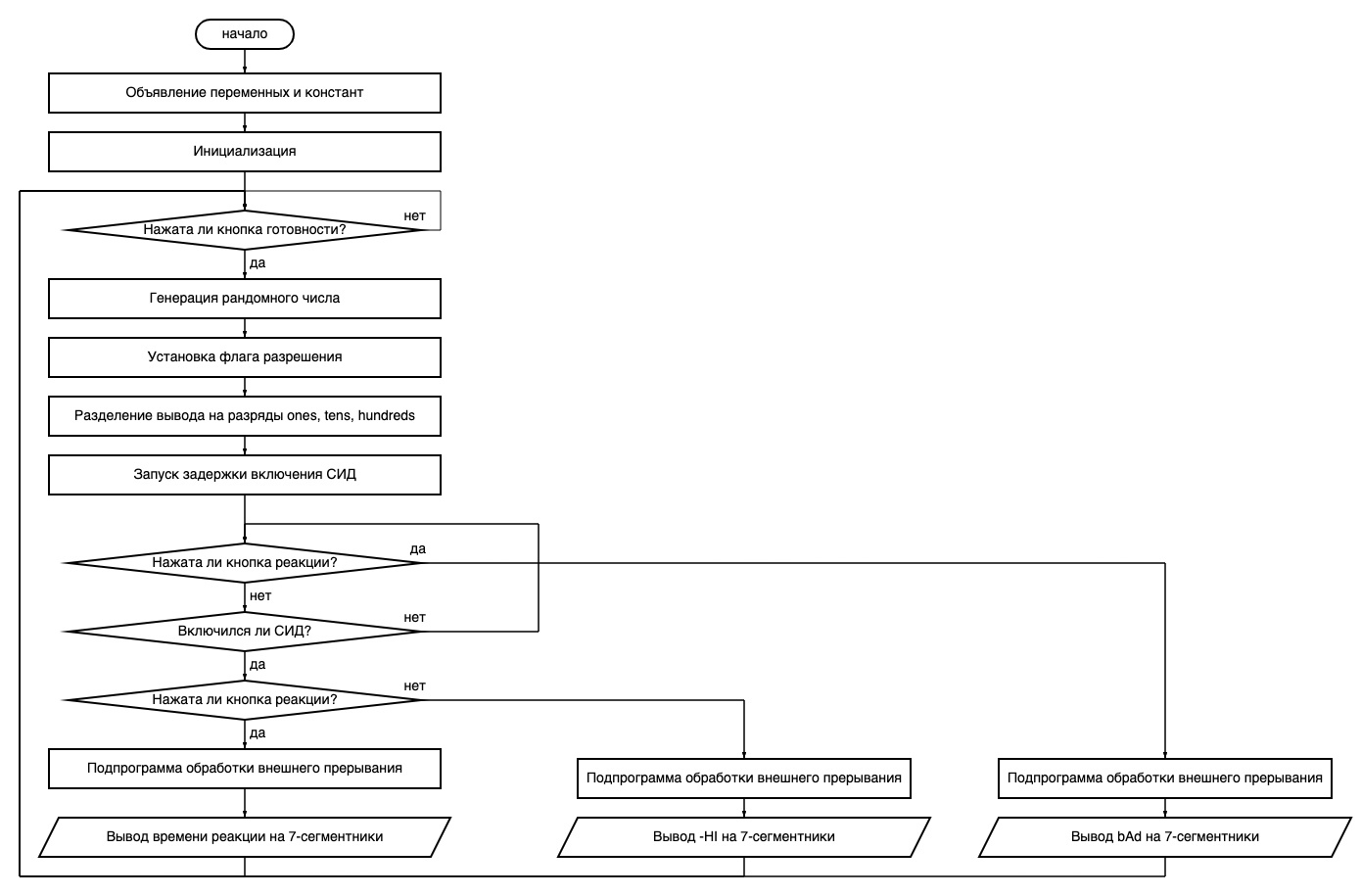


Рисунок 2 – блок схема программного кода.

Схема электрическая принципиальная и перечень элементов в приложении 1.

Код програмы reaction.asm:

|  |
| --- |
| ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;\* Designer Mechtcherjakowa R.I.  ;\* Version: 1.0  ;\* Date 18.03.2009  ;\* Title: Reaction.asm  ;\* Device ATmega16  ;\* Clock frequency:Частота кв.резонатора 8 мГц  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ; учебная  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;(Назначение - программа определяет время реакции пользователя и  ;отображает время реакции пользователя в мсек на дисплее (3 семиcегмент.индикатора).  ;после нажатия кнопки "Готовность" через случайный промежуток времени (4...12с)  ;загорается светодиод (СИД),при его включении пользователь должен  ;нажать кнопку "Реакция")  ;Кнопка "Готовность" подключена к PD1,кнопка "Реакция" подключена к PD2(INT0) входы,  ;Cветодиод СИД подключенк PB7 - выход  ;семигегментные индикаторы PC0-PC7-выходы,семигегментный индикатор  ;PC0-a,PC1-b,PC2-c,PC3-d,PC4-e,PC5-f,PC6-g,PC7-h  ;сигналы выбора индикатора PB2 - сотни,PB1-десятки,PB0 - единицы - выходы  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;Задание  ;1.Выполните проверку работы программы в симуляторе AVR Studio.Измерьте в симуляторе  ; время,в течении которого поочередно включен каждый семисегментный индикатор  ;2.Выполните программирование макетной платы. Проверьте работу платы в 3-х режимах:  ;- нажать кнопку PD1-"готовность",после загорания CИД(PB7) нажать PD2-"реакция"  ;- нажать кнопку PD1-"готовность",до загорания CИД(PB7) нажать PD2-"реакция"  ;- нажать кнопку PD1-"готовность",после загорания CИД(PB7) не нажимать PD2-"реакция"  ;3.Найдите и исправьте ошибку в программе  ;4.Доработайте программу,увеличив счетчик дисплея Disp\_Count до 2-х байт с максимальным  ;значением 0xFFFF,введя дополнительный байт Disp\_CountH. Запрограммируйте макет.плату  ;5.Объясните работу схемы.Экспериментально подберите константу Val\_dispCount,опре-  ;деляющую время включения индикатора, когда глаз не воспринимает мигание индикаторов.  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  .include "m16def.inc";"C:\Program Files (x86)\Atmel\AVR Tools\AvrAssembler\Appnotes\m16def.inc"  ; присоединение файла описаний;  .list ;включение листинга  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ; Register Variables  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  .def temp =R16;регистр для хранения временных данных, исп. для промежуточного хранения данных при обращении к регистрам РВВ;  .def Random =R15;буфер случайного числа  .def Hundreds =R18;адреса символов для семисегментных индикаторов по таблице перекодировки  .def Tens =R19;  .def Ones =R20;  .def CountX =R14;счетчика числа переполнений таймера Т0 при формировании псевдослучайной задержки  .def Disp\_Numb =R22;указатель включенного интервала  .def Disp\_Count =R23; счетчик обновления дисплея  .def TimeH =R17;счетчик переполнений Т0  .def TimeL =R25;  .def Count3 =R13;счетчик для организации процедуры умножения на 16 логическим сдвигом влево на 4 разряда;  .def tempH =R21;преобразованое значение, хранящегося в регистрах TimeH:TimeL, в миллисекунды  .def tempH2 =R26;  .def temp2 =R27;  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  .def Byte\_fl =R24;байт флагов  ;------------------  .equ F\_ready =1;бит флага "Готовность" (бит 1 в байте Byte\_fl)  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ; Constants  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  .equ Val\_dispCount=50;величина константы,опр. время вкл индикатора  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  .cseg  .org $0000  rjmp Init  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  .org INT0addr;=$002 ;External Interrupt0 Vector Address  rjmp IN\_INT0; Переход на процедуру обработки прерывания по нажатию кнопки "Реакция"  ;----------------  .org INT1addr;=$004 ;External Interrupt1 Vector Address  reti  .org OC2addr; =$006 ;Output Compare2 Interrupt Vector Address  reti  .org OVF2addr;=$008 ;Overflow2 Interrupt Vector Address  reti  .org ICP1addr;=$00A ;Input Capture1 Interrupt Vector Address  reti  .org OC1Aaddr;=$00C ;Output Compare1A Interrupt Vector Address  reti  .org OC1Baddr;=$00E ;Output Compare1B Interrupt Vector Address  reti  .org OVF1addr;=$010 ;Overflow1 Interrupt Vector Address  reti  .org OVF0addr;=$012 ;Overflow0 Interrupt Vector Address  rjmp IN\_T0ovf; Переход на процедуру обработки прерывания переполнение таймера-счетчика  ;--------------------  .org SPIaddr; =$014 ;SPI Interrupt Vector Address  reti  .org URXCaddr;=$016 ;UART Receive Complete Interrupt Vector Address  reti  .org UDREaddr;=$018 ;UART Data Register Empty Interrupt Vector Address  reti  .org UTXCaddr; =$01A ;UART Transmit Complete Interrupt Vector Address  reti  .org ADCCaddr; =$01C ;ADC Interrupt Vector Address  reti  .org ERDYaddr; =$01E ;EEPROM Interrupt Vector Address  reti  .org ACIaddr; =$020 ;Analog Comparator Interrupt Vector Address  reti  .org TWIaddr; =$022 ;Irq. vector address for Two-Wire Interface  reti  .org INT2addr; =$024 ;External Interrupt2 Vector Address  reti  .org OC0addr; =$026 ;Output Compare0 Interrupt Vector Address  reti  .org SPMRaddr; =$028 ;Store Program Memory Ready Interrupt Vector Address  reti  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ; Start Of Main Program  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Init: ;Инициализация стека (Устанавливаем указатель в начало стекаЫ)  ldi temp,LOW(RAMEND);выбор вершины стека  out SPL, temp;Указатель стека  ldi temp,HIGH(RAMEND)  out SPH,temp  ; ------Инициализация портов В/B  Init\_B: ldi temp,0b11111111;(PB0-PB2)-выходы управления семисегментными индикаторами + PB7-СИД  out DDRB,temp  ldi temp,0b00000100;выбраны сотни (1-й индикатор)  out PORTB,temp  Init\_C: ser temp; (PC0-PC7) - выходы, задающие значение на семисегментном индикаторе  out DDRC,temp  ;  Init\_D: ldi temp,0b11111001;PD1,PD2-входы, остальные-выходы (Кнопка "Готовность" PD1,кнопка "Реакция" PD2(INT0))  out DDRD,temp  ldi temp,0b00000110;Вкл подтяжка на кнопках.  out PORTD,temp;  ; --- Инициализация таймера TCNT0  Init\_T0: ldi temp,(1<<TOIE0);TOIE0-Timer/Counter0 Overflow Interrupt Enable (разрешается прерывание по переполнению таймера)  out TIMSK,temp  ; --- Инициализация внешнего прерывания INT0  ;  ldi temp,(1<<ISC01); ;прерывание по спаду ISC01=1 ISC00=0  out MCUCR,temp  ldi temp,(1<<INT0);INT0: External Interrupt Request 0 Enable (разрешение прерывания)  out GIMSK,temp  ;  Init\_R: ldi Disp\_Count,Val\_dispCount; загружаем в регистр время вкл индикатора  clr Disp\_Numb ; устанавливаем нулевое значение текущего индикатора  clr Byte\_fl; сбрасываем байт флагов (Сигнализирует о возникновения некоторого события)  ldi temp,1 ;любое число  mov Random,temp;в регистр Random (1)  clr Hundreds; обнуление регистров значений индикаторов  clr Tens ;(0)  clr Ones  ;==================================================  ;начало цикла  ;==================================================  Start: rcall Display; вызов процедуры динамической индикации  sbrc Byte\_fl,F\_ready; проверяется состояние флага F\_ready (если бит1=0 пропускаем start) устанавливается после нажатия готов  rjmp Start ;внутренний цикл  sbic PIND,1 ;ждем нажатия кнопки "Готов"  rjmp Start  ;Формирование псевдослучайного числа  Rand\_st: mov temp,Random;Вычисляем следующее случайное число  add Random,temp;умножаем на 5 сложением  add Random,temp;случайное число в диапазоне(0-255)  add Random,temp  add Random,temp  add Random,temp  inc Random  ;  mov temp,Random  ;Т0 - 8 разр таймер. Counter0 оverflow соотвествует  ;времени 256\*1024/8000000=0,033с.Интервал (4...8,4с)->  ;Формируем интервал(127+Random/2)  lsr temp; деление на 2    subi temp,-127;прибавляем случайное число (от 90 до 256)  Rand\_end: mov CountX,temp    ldi temp,(1<<CS02)|(1<<CS00);частота TCNT0 Clk/1024,(CS02,CS01,CS00) установка предделителя таймера  out TCCR0,temp;включаем таймер  ldi temp,(1<<INTF0)  out GIFR,temp;сбрасываем флаг прерыв.INT0 записью 1  ldi temp,(1<<TOV0)  out TIFR,temp;сбрасываем флаг прерыв TOV0 (флаг прерыв. по переполнению)  sei ;разрешаем прерывания  clr TimeH ;сброс счетчика переполнений Т0  clr Hundreds;обнуление регистров значений индикаторов  clr Tens  clr Ones  set  bld Byte\_fl,F\_ready;Флаг F\_ready (бит1) в байте Byte\_fl установить в 1  rjmp Start  ;==================================================  ; конец цикла  ;==================================================  ; Подпрограмма Display работы с дисплеем (динам. индикация)  ;==================================================  Display: dec Disp\_Count; уменьшаем счетчик обновления дисплея  ;Индикатор активен, пока регистр Disp\_Count уменьшается от значения Val\_dispCount до 0, затем происходит смена активного индикатора  brne ex\_displ; проверка нуля Z =0 (если верно, то переход)  ;  ldi Disp\_Count,Val\_dispCount; загружаем в регистр время вкл индикатора  ;  inc Disp\_Numb;выбираем следующий дисплей (от 0 до 2)  cpi Disp\_Numb,3  brne Out\_disp; если дисплей не последний (Z =0 ) переход  clr Disp\_Numb; если дисплей посдений обнуляем и опять идем с нулевого  rjmp Out\_disp  Out\_disp: ldi ZL,18; ;указатель на Hundreds  ldi ZH,0  add ZL,Disp\_Numb  ld temp,Z  ; преобразуем в семисегментный код  ;----------------------------------------------------  ldi ZL,TABLE\*2;загружаем адрес начала  ldi ZH,0x00 ;таблицы в памяти программ (\*2 - для байтовой  add ZL,temp ;адресации)  ;  lpm temp,Z ;читаем семисегментный код значения ;  out PortC,temp; передаем на индикатор  ;  in temp,PINB  in temp2,PINB  andi temp,0b00000111;маскируем биты выбора индик.PB2 - сотни,PB1-десятки,PB0 - един.  andi temp2,0b10000000;маскируем бит светодиода (СИД)  lsr temp  brcc PC+2; Проверка переноса (перед сдвигом вправо PB0 = 1)  ldi temp,0b00000100; в начало (Hundreds)  or temp, temp2;восстанавливаем СИД  out PORTB,temp    ;  ex\_displ: ret  ;  ;==================================================  ;------- Таблица перекодировки символов  TABLE: .db 0b00111111,0b00000110; коды "0","1"  .db 0b01011011,0b01001111; коды "2","3"  .db 0b01100110,0b01101101;;коды "4","5"  .db 0b01111101,0b00000111;;коды "6","7"  .db 0b01111111,0b01101111;;коды "8","9"  .db 0b01111100,0b01110111; коды "b","A"  .db 0b01011110,0b01000000; коды "d","-"  .db 0b01110110,0b00110000; коды "H","I"    ;==================================================  ;Подпрограмма обработки внешнего прерывания INT0  ;Т0 - 8 разр таймер. Counter0 оverflow соотвествуетвремени 256\*1024/8000000=0,033с  ;1мсек=7,8125отсчета. т.е за временной интеввал считаем не более 7812,5 отсчета два байта  ;для преобразование в мсек делим на 7,8125 (или \*0,128) что можно получить (16/125)  ;-> (8/125)\*2)  ;на 3 семисегмент. индикатора можно вывести максимальное число 999,Этому соответствует  ; ограничение в отсчетах  ;за время 999мсек имеем Х\_отсч\*(16/125), где Х\_отсч=7804 (1E7C)h. Контролируем лишь старший байт  ; TimeH (1Fh), добавив 84h,а затем вычитая его из результата.  ;.  ;===================================================  IN\_INT0: push temp  in temp,SREG;сохранение регистра статуса в стек  push temp  ;  sbis PINB,7;проверяем СИД  rjmp Cheat;если 0 (не горит), то кнопка "Реакция" была нажата преждевременно и переходим в Cheat  clr temp  out TCCR0,temp;остaнов Т0  in TimeL,TCNT0;счетный регистр таймера Т0 TCNT0 сохраняется в регистре TimeL  ;В регистре TimeH уже храниться число переполнений таймера Т0 (см обр прерывания IN\_T0ovf)  in temp,TIFR ;проверяем Т0 на переполнение  sbrc temp,1  inc TimeH  subi TimeL,0x84 ;вычитаем 84h  sbci TimeH,0  ;цикл, который осуществляет 4 логических сдвига влево для числа, образованного регистрами tempH2:tempH:temp  ldi temp,4  mov Count3,temp; умножаем время реакции на 16.логическим сдвигом влево 4 раза  clr tempH2  mov temp,TimeL  mov tempH,TimeH  shl3: lsl temp  rol tempH  rol tempH2  dec Count3  brne shl3  ;  clr TimeL  clr TimeH  ;  Divide12: subi temp,125;деление вычитанием  sbci tempH,0  sbci tempH2,0  brcs DoneDividing  inc TimeL  brne Divide12  inc TimeH  rjmp Divide12  ;  DoneDividing:  rcall digitConvert;преобразует число в значения регистров Hundreds, Tens и Ones  ;  ex\_INT0: clt;сбрасываем пользовательский бит Т  bld Byte\_fl,F\_ready;загрузить бит в регистре из флага Т  clr TimeH ;сброс счетчика переполнений Т0  clr temp  out TCCR0,temp;остaнов Т0  cbi PORTB,7;выключаем СИД  pop temp ;выход без разрешения глоб. прерывания  out SREG,temp; возвращаем регистр статуса из стека  pop temp  ret  ;=================================================  Cheat: ldi Hundreds,10;коды "b",  ldi Tens,11 ; "A"  ldi Ones,12 ; "d"  rjmp ex\_INT0  ;  ;===================================================  ;Конвертирует двоичное 2-х байтное число в двоично-десятичный код  ;===================================================  digitConvert:  clr Hundreds  clr Tens  clr Ones  ;Конвертация осуществляется последовательным вычитанием сначала числа 100, а затем числа 10 из пары регистров TimeH:TimeL  ;Остаток записывается в регистр Ones.  FindHundreds:  subi TimeL,100  sbci TimeH,0  brcs FindTens  inc Hundreds  rjmp FindHundreds  ;  FindTens: subi TimeL,-100  subi TimeL,10  brcs FindOnes  inc Tens  rjmp FindTens+1  ;  FindOnes: subi TimeL,-10  mov Ones,TimeL  ret  ;===================================================  ;Подпрограмма обработки прерывания переполнения таймера Т0  ;Т0 - 8 разр таймер. Counter0 оverflow соотвествуетвремени 256\*1024/8000000=0,033с  ;1cек =(1024/8000000)\*N\_отсчетов.(N отсч =7812,5 за секунду)  ;1мсек=7,8125отсчета. т.е за временной интеввал считаем не более 7812,5 отсчета два байта  ;для преобразование в мсек делим на 7,8125 (или \*0,128) что можно получить (16/125)  ;за время 999мсек имеем Х\_отсч\*(16/125), где Х\_отсч=7804 (1E7C)h. Контролируем TimeH (1Fh)  ; добавив 84h,а затем вычитая.  ;===================================================  IN\_T0ovf: push temp  in temp,SREG;SREG сохраняется в стеке  push temp  ;  sbic PINB,7;проверяем СИД  rjmp LEDon;Если бит не равен нулю (светодиод горит), переходим на LEDon  dec CountX  brne ex\_T0ovf  Start\_m: ldi temp,0x84  out TCNT0,temp  sbi PORTB,7;включаем СИД  ;  ex\_T0ovf: pop temp ;выход c разрешениtv глоб. прерывания  out SREG,temp  pop temp  reti  ;таймер находится в режиме подсчета времени реакции  LEDon: inc TimeH ;инкрементируем ст байт  cpi TimeH,0x1F ;проверяем на максимальное время  brlo ex\_T0ovf  ; превышение времеми > 999 мсек  ;Если значение этого регистра меньше 0х1F, значит 999 мс еще не прошло и можно продолжать счет.  ;Если же оно стало равно или больше 0х1F, значит время ожидания истекло  out\_HI: clt  bld Byte\_fl,F\_ready  clr temp  out TCCR0,temp;остaнов Т0  ldi Hundreds,13 ;"-"  ldi Tens,14 ;"H"  ldi Ones,15 ;"I"  cbi PORTB,7;выключаем СИД  pop temp  out SREG,temp  pop temp  ret ;выход без разрешения глоб. прерывания  ;=====================================================  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ; ldi ZL,18; ;указатель на Hundreds  ; ldi ZH,0  ;----------------------------------------------------  ; ldi ZL,low(TABLE\*2);загружаем адрес начала  ; ldi ZH,high(TABLE\*2);таблицы в памяти программ (\*2 - для байтовой  ; add ZL,temp ;адресации)  ; clr temp  ; adc ZH,temp  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;.def Disp\_CountH=R1  ;---------------------  ; Constants  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;.equ Val\_dispCountH=255;величина константы,опр. время вкл индикатора  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ; ldi Disp\_Count,Val\_dispCount  ; ldi temp,Val\_dispCountH  ; mov Disp\_CountH,temp  ;  ;----------------------------------------  ; subi Disp\_Count,0x01  ; ldi temp,0x00  ; sbc Disp\_CountH,temp  ; ldi temp,0x00  ; cp Disp\_Count,temp  ; brne ex\_displ  ; cp Disp\_CountH,temp  ; brne ex\_displ  ;  ; ldi Disp\_Count,Val\_dispCount  ; ldi temp,Val\_dispCountH  ; mov Disp\_CountH,temp  ;  ;---------------------------------------- |

Исправленные фрагменты кода

3) Чтобы исправить ошибку программы, был уточнен регистр считывания значений из таблицы TABLE в temp

|  |
| --- |
| Out\_disp: ldi ZL,18; ;указатель на Hundreds  ldi ZH,0  add ZL,Disp\_Numb  ld temp,Z  ; преобразуем в семисегментный код  ;----------------------------------------------------  ldi ZL,LOW(TABLE\*2);загружаем адрес начала  ldi ZH,HIGH(TABLE\*2) ;таблицы в памяти программ (\*2 - для байтовой  add ZL,temp ;адресации)  clr temp  adc ZH,temp |

4) Чтобы доработать программу была оптимизирована подпрограмма Display

|  |
| --- |
| Display: subi Disp\_Count,0x01  ldi temp,0x00  sbc Disp\_CountH,temp  ldi temp,0x00  cp Disp\_Count,temp  brne ex\_displ  cp Disp\_CountH,temp  brne ex\_displ  ldi Disp\_Count,Val\_dispCount  ldi temp,Val\_dispCountH  mov Disp\_CountH,temp    inc Disp\_Numb;выбираем следующий дисплей (от 0 до 2)  cpi Disp\_Numb,3  brne Out\_disp; если дисплей не последний (Z =0 ) переход  clr Disp\_Numb; если дисплей посдений обнуляем и опять идем с нулевого  rjmp Out\_disp |

Для того чтобы обеспечить работу программы (4 пункт) пришлось объявить новый регистр:

|  |
| --- |
| .def temp =R16;регистр для хранения временных данных, исп. для ;промежуточного хранения данных при обращении к регистрам РВВ;  .def Random =R15;буфер случайного числа  .def Hundreds =R18;адреса символов для семисегментных индикаторов по ;таблице перекодировки  .def Tens =R19;  .def Ones =R20;  .def CountX =R14;счетчика числа переполнений таймера Т0 при ;формировании псевдослучайной задержки  .def Disp\_Numb =R22;указатель включенного интервала  .def Disp\_Count =R23; счетчик обновления дисплея  .def Disp\_CountH =R28  .def TimeH =R17;счетчик переполнений Т0  .def TimeL =R25;  .def Count3 =R13;счетчик для организации процедуры умножения на 16 ;логическим сдвигом влево на 4 разряда;  .def tempH =R21;преобразованое значение, хранящегося в регистрах ;TimeH:TimeL, в миллисекунды  .def tempH2 =R26;  .def temp2 =R27;  .def Byte\_fl =R24;байт флагов  ;------------------  .equ F\_ready =1;бит флага "Готовность" (бит 1 в байте Byte\_fl)  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

5) Значение константы ValDisplH было определено экспериментально

|  |
| --- |
| ; Constants  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  .equ Val\_dispCount=50;величина константы,опр. время вкл индикатора  .equ Val\_dispCountH=15 |