Министерство образования и науки Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого

—

**Институт** **кибербезопасности и защиты информации**

Отчёт

по лабораторной работе №5

**Организация ввода-вывода по прерываниям**

**Вариант 4**

по дисциплине «Аппаратные средства вычислительной техники»

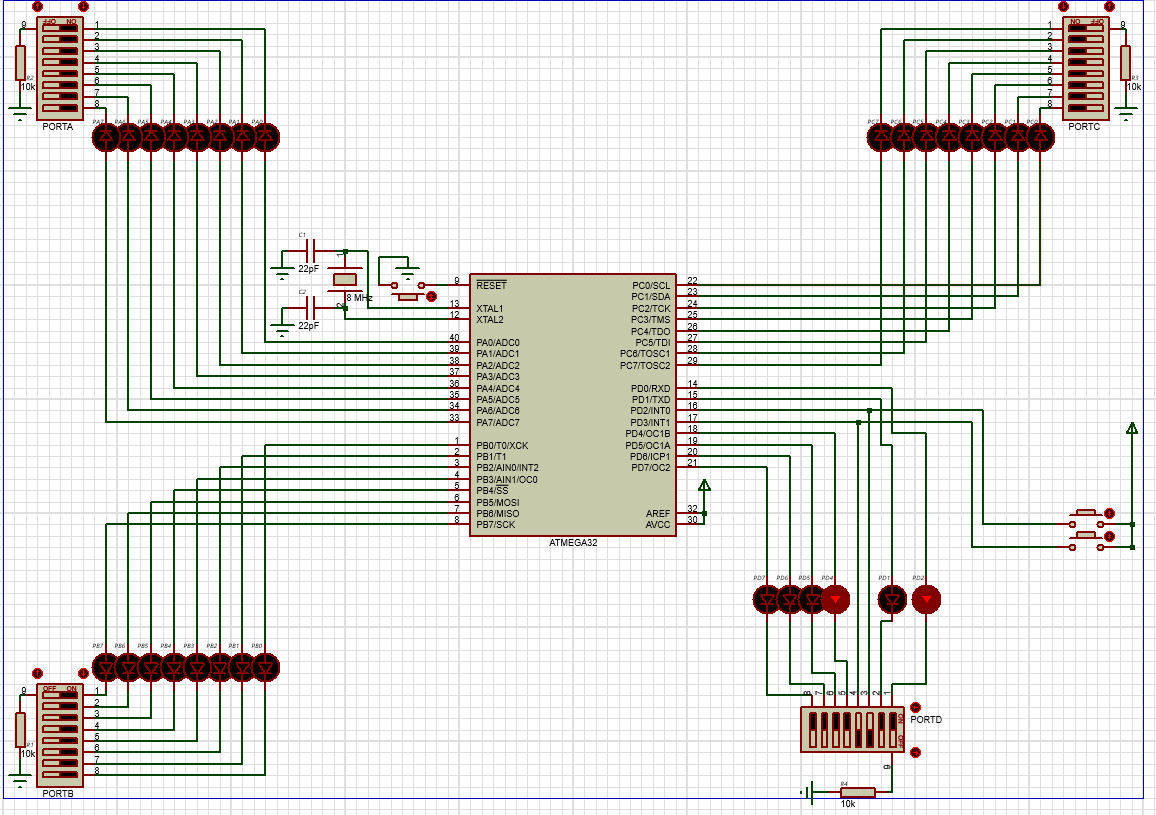
|  |  |  |  |
| --- | --- | --- | --- |
| Выполнили: | студенты группы 4851003/00002 |  | И.А. Скрипко |
|  |  | (подпись, дата) |  |
|  |  |  | А.П. Калугина |
|  |  | (подпись, дата) |  |
|  |  |  | Д.А. Данилов |
|  |  | (подпись, дата) |  |
| Проверил: | старший преподаватель |  | П.О. Семенов |
|  |  | (подпись, дата) |  |

# Формулировка задания

Получение практических навыков по обработке внешних прерываний и по организации ввода-вывода с помощью механизма прерываний.

# Схема лабораторной установки

Схема лабораторной установки представлена на рисунке 1.

Рисунок 1 – схема лабораторной установки

# Блок-схема алгоритма работы программы

На рисунке 2 представлена блок-схема подготовительной части программы.

На рисунке 3 представлена блок-схема основной части программы.

На рисунке 4 представлена блок-схема прерывания int1.

На рисунке 5 представлена блок-схема прерывания int0.

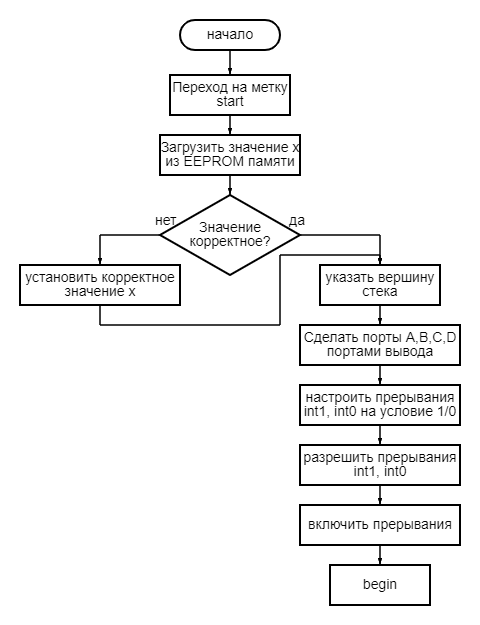


Рисунок 2 – Блок-схема подготовительной части программы

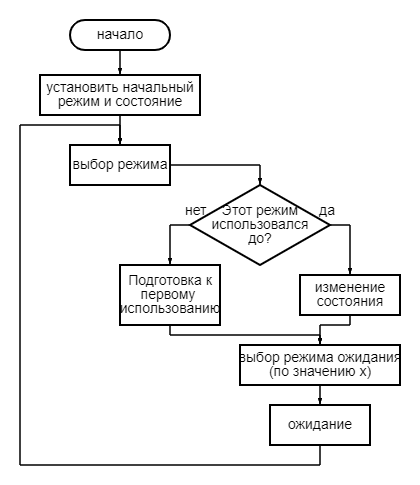


Рисунок 3 – блок-схема основной части программы



Рисунок 4 – Блок-схема INT1\_HANDLER

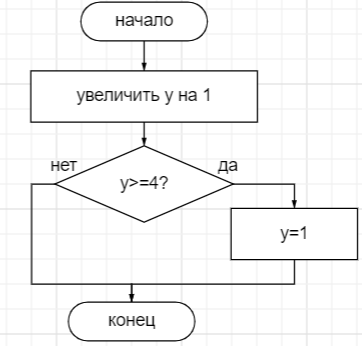


Рисунок 5 – Блок-схема INT0\_HANDLER

# Временные диаграммы логических сигналов на портах МК (фрагмент)

На рисунках 6-11 представлены временные диаграммы логических сигналов.

Изображение выглядит как текст, здание

Автоматически созданное описание

Рисунок 6 – Временная диаграмма на портах PORTA, PORTB, PORTC при х=2 (второе состояние – 0,5 Гц), y=1 (0xff или 0х00)

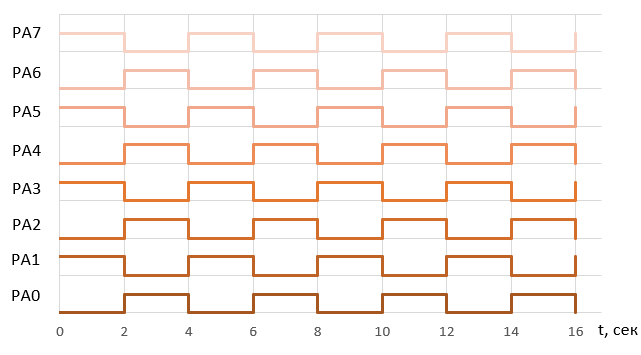


Рисунок 7 – Временная диаграмма на портах PORTA, PORTB, PORTC при х=2 (второе состояние – 0,5 Гц), y=2 (0xaa или 0x55)

Изображение выглядит как текст, седзи, здание

Автоматически созданное описание

Рисунок 8 – Временная диаграмма на порте PORTA при х=2 (второе состояние – 0,5 Гц), y=3 (0xff – на одном, на остальных – 0x00)

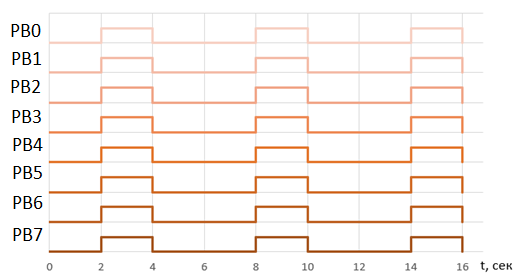


Рисунок 9 – Временная диаграмма на порте PORTB при х=2 (второе состояние – 0,5 Гц), y=3 (0xff – на одном, на остальных – 0x00)

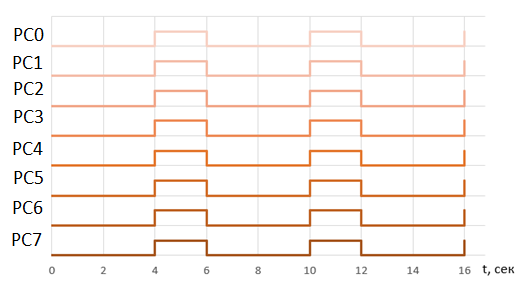


Рисунок 10 – Временная диаграмма на порте PORTC при х=2 (второе состояние – 0,5 Гц), y=3 (0xff – на одном, на остальных – 0x00)

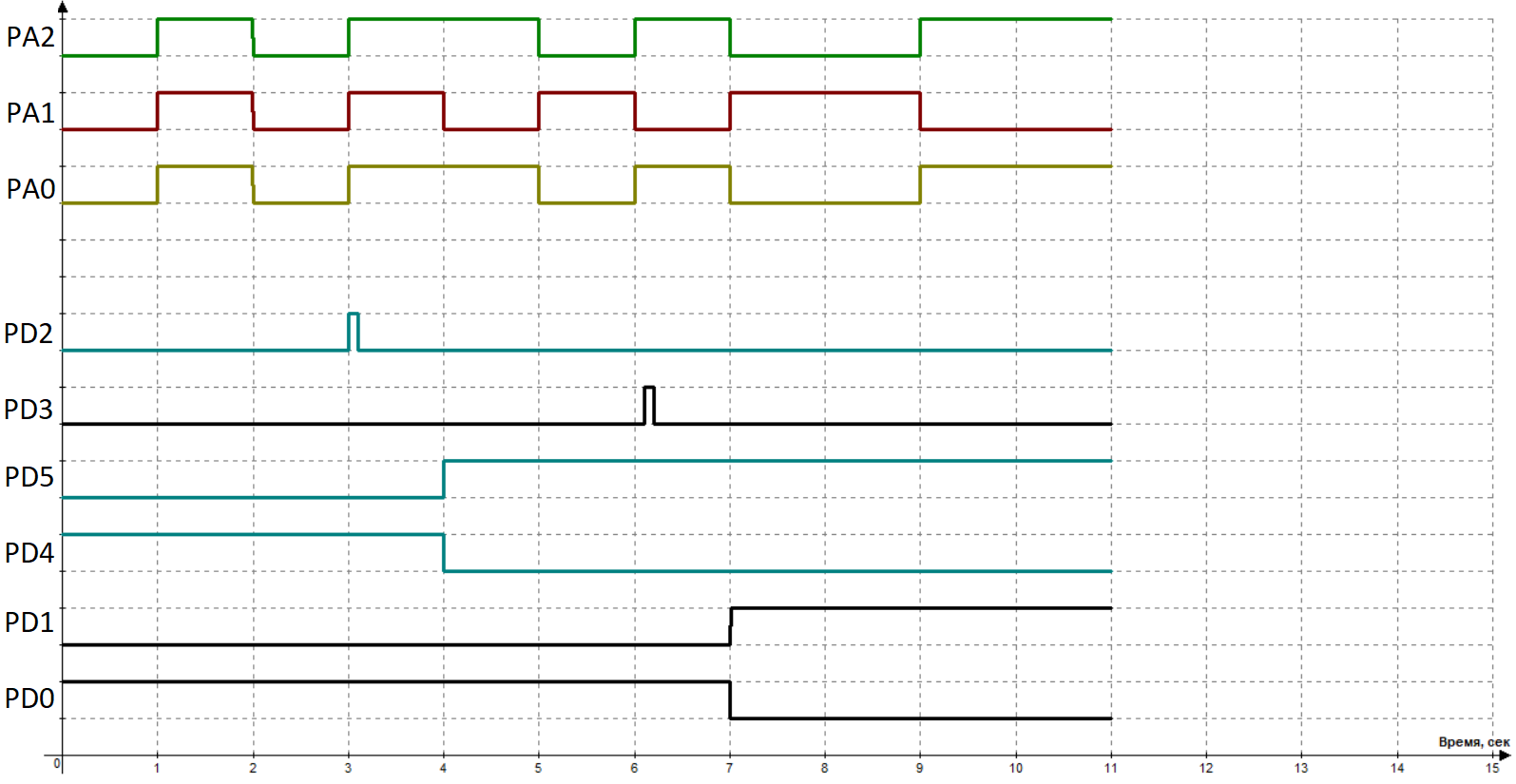


Рисунок 11 – Изменение работы программы при срабатывании прерываний

# Алгоритм выполнения задействованных команд (конструкций) ассемблера

В таблице 1 представлены задействованные команды ассемблера и их описание.

Таблица 1 – Задействованные команды ассемблера

|  |  |  |
| --- | --- | --- |
| Команда | Описание | Количество тактов |
| SER | Установка всех битов регистра | 1 |
| CLR | Очистка всех битов регистров | 1 |
| OUT | Запись значения регистра в порт | 1 |
| COM | Инвертирование | 1 |
| LDI | Загрузка константы | 1 |
| CP | Сравнение двух операндов. Возвращает 1, если равны, иначе 0 | 1 |
| BRNE | Выполняет переход, если флаг Z = 0 (Z = 1 при нулевом результате прошлой операции) | 1 – если нет перехода 2 – если есть переход |
| RJMP | Выполняет относительный безусловный переход | 2 |
| JMP | Выполняет прямой безусловный переход | 3 |
| LPM | Считывание из ПЗУП в регистр | 3 |
| BRLO | Переход, если меньше | 1/2 |
| CPI | Сравнение регистра с константой | 1 |
| BRSH | Переход, если больше либо равно | 1/2 |
| SEI | Разрешение прерываний | 1 |
| SBI | Установить бит в порту ввода/вывода | 2 |
| CBI | Очистить бит в порту ввода/вывода | 2 |
| CALL | Прямой вызов подпрограммы | 4 |
| RET | Возврат из подпрограммы | 4 |
| INC | Инкремент | 1 |
| RETI | Возврат из обработчика прерывания | 4 |

# Результаты работы

Была написана программа для микроконтроллера ATmega32, реализующая вывод информации на порты вывода PORTA, PORTB, PORTC, PORTD. Информация может выводиться в 3 разных вариантах, для каждого варианта можно настроить частоту изменения состояний (3 возможных частоты). Изменения можно производить с помощью прерываний int1 и int0.

# Ответы на контрольные вопросы

1. Как реализуется подсистема прерываний в микроконтроллере AVR?

Прерывание прекращает нормальный ход программы для выполнения приоритетной задачи, определяемой внутренним или внешним событием.

Для каждого такого события разрабатывается отдельная программа, которую называют подпрограммой прерывания, и размещается в памяти программ.

У каждого периферийного устройства, что входит в состав AVR микроконтроллеров, есть как минимум один источник прерывания.

За каждым прерыванием, строго закреплён вектор (ссылка), указывающий на процедуру обработки прерывания (Interrupt service routine). Все векторы прерываний, располагаются в самом начале памяти программ и вместе формируют «таблицу векторов

прерываний» (Interrupt vectors table). Каждому прерыванию соответствует определённый «бит активации прерывания» (Interrupt Enable bit).

Таким образом, чтобы использовать определённое прерывание, следует записать в его «бит активации прерывания» – логическую единицу. Далее, независимо от того активированы ли определённые прерывания, микроконтроллер не начнёт обработку этих прерываний, пока в «бит всеобщего разрешения прерываний» (Global Interrupt Enable bit в регистре состояния SREG) не будет записана логическая единица. Также, чтобы запретить все прерывания (на неопределённое время), в бит всеобщего разрешения прерываний следует записать – логический ноль.

1. Как программно разрешить или запретить выполнение прерываний?

Записать в «бит всеобщего разрешения прерываний» логическую единицу или ноль. Для этого есть команды SEI (разрешить) и CLI (запретить). После этого нужно настроить и разрешить нужные прерывания.

1. Какие источники прерываний есть в микроконтроллерах AVR?

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Адрес** | **Источник** | **Описание** |
| 1 | 0x000 | RESET | Сигнал сброса |
| 2 | 0x002 | INT0 | Внешний запрос на прерывание по входу INT0 |
| 3 | 0x004 | INT1 | Внешний запрос на прерывание по входу INT1 |
| 4 | 0x006 | INT2 | Внешний запрос на прерывание по входу INT2 |
| 5 | 0x008 | TIMER2\_COMP | Совпадение с регистром сравнения таймера T/C2 |
| 6 | 0x00A | TIMER2\_OVF | Переполнение счётчика T/C2 |
| 7 | 0x00C | TIMER1\_CAPT | Захват по таймеру T/C1 |
| 8 | 0x00E | TIMER1\_COMPA | Совпадение с регистром сравнения A таймера T/C1 |
| 9 | 0x010 | TIMER1\_COMPB | Совпадение с регистром сравнения B таймера T/C1 |
| 10 | 0x012 | TIMER1\_OVF | Переполнение счётчика T/C1 |
| 11 | 0x014 | TIMER0\_COMP | Совпадение с регистром сравнения таймера T/C0 |
| 12 | 0x016 | TIMER0\_OVF | Переполнение счётчика T/C0 |
| 13 | 0x018 | SPI\_STC | Передача данных по интерфейсу SPI завершена |
| 14 | 0x01A | UART\_RXC | Приём данных приёмопередатчиком UART завершён |
| 15 | 0x01C | UART\_UDRE | Регистр данных UART пуст |
| 16 | 0x01E | UART\_TXC | Передача данных приёмопередатчиком UART завершена |
| 17 | 0x020 | ADC | Завершено преобразование АЦП |
| 18 | 0x022 | EE\_RDY | EEPROM готов |
| 19 | 0x024 | ANA\_COMP | Прерывание от аналогового компаратора |
| 20 | 0x026 | TWI | Прерывание от интерфейса I2C |
| 21 | 0x028 | SPM\_RDY | Запись программной памяти (Flash) готова |

1. Как настраиваются внешние прерывания?

У ATMEGA32 3 внешних прерывания – INT0, INT1, INT2. Эти прерывания жестко «привязаны» к определенным выводам и переназначить их на другие выводы нельзя.

Для разрешения или запрещения внешних прерываний предназначен управляющий регистр GICR (General Interrupt Control Register).

Установка битов INT1, INT0 или INT2 разрешает прерывания при возникновении события на соответствующем выводе микроконтроллера AVR, а сброс – запрещает.

Также нужно установить еще и флаг глобального разрешения прерываний – I, который расположен в регистре SREG.

Внешние прерывания можно настроить на определенное условие. Например, каждый раз, когда на выводе INT0 устанавливается логический 0.

# Выводы по лабораторной работе

Был изучен механизм внешних прерываний. Также были изучены новые команды ассемблера. Получены знания по работе с ПЗУП (запись и чтение). Изучена работа изменения информации в порты с помощью внешних прерываний.

# Приложение 1 Комментированный листинг программы для МК на языке ассемблера

.org $000

JMP RESET ; Указатель на начало программы

.org INT0addr

JMP EXT\_INT0 ; Указатель на обработчик прерывания int0

.org INT1addr

JMP EXT\_INT1 ; Указатель на обработчик прерывания int1

RESET:

EEPROM\_read:

sbic EECR,EEWE

rjmp EEPROM\_read

out EEARH, r18

out EEARL, r17

sbi EECR,EERE

in r16,EEDR

;проверяем на корректность считанное значение

cpi R19, 0x01

brlo incorrect;если меньше 1 - там некорректное значение

cpi R19, 0x04

brsh incorrect;если больше или равно 4 - там некорректное значение

rjmp correct

incorrect:

LDI R19, 0x03;если в памяти находится некорректное значение (программа запущена впервые), то ставим 1 состояние

correct:

;LDI R19, 0x02;

LDI R20, HIGH(RAMEND) ; Старшие разряды адреса

OUT SPH, R20 ; Установка вершины стека в конец ОЗУ

LDI R20, LOW(RAMEND) ; Младшие разряды адреса

OUT SPL, R20 ; Установка вершины стека в конец ОЗУ

SER R20;Регистр R20 заполняется единицами

OUT DDRA, R20;Сделать порты A, B, C, D - портами вывода

OUT DDRB, R20

OUT DDRC, R20

LDI R20, 0xf3

OUT DDRD, R20

LDI R16, 0x0F

OUT MCUCR, R16;Настройка прерываний int0 и int1 на условие 0/1

LDI R16, 0xC0

OUT GICR, R16;Разрешение прерываний int0 и int1

OUT GIFR, R16 ; Предотвращение срабатывания int0 и int1 при включении прерываний

SEI;Включить прерывания

begin:

;исходное положение: первый режим и первое состояние

;LDI R19, 0x03;состояние x: 1 - 0.25, 2 - 0.5, 3 - 1.0 Гц

LDI R18, 0x01;режим: 1 - все горят/не горят с частотой x

;2 - горят 10101010/01010101 с частотой x

;3 - на одном порту горят все лампочки, на остальных ни одной. Может быть 3 состояния

LDI R25, 0x03;состояние для третьего режима работы: показывает, какой светодиод горит

LDI R29, 0x00;регист для очистки состояния в PORTD

ldi R24, 0x00;номер предыдущего режима работы

ldi R22, 0x01;номер состояния в конкретном режиме (горят/горят и какие именно горят)

choose\_mode:

out PORTD, R29;тушим все светодиоды - они будут установлены заново

CPI R18, 0x01

BRNE next\_mode1

cpi R24, 0x01

BREQ do\_1

;подготовка к началу использования режима

ldi R22, 0x01;подготовка=начало работы, поэтому первое состояние режима

LDI R24, 0x01;показываем, что режим используется

CLR R23;подать во все порты 0х00

OUT PORTA, R23

OUT PORTB, R23

OUT PORTC, R23

do\_1:;если режим уже используется и подготовку проводить не надо

sbi PORTD, 4;устанавливаем 4 разряд - режим работы 1

cpi R22, 0x01

brne check\_1

cbi PORTD, 7;устанавливаем бит, указывающий состояние 2 в 0

sbi PORTD, 6;устанавливаем бит, указывающий состояние 1 в 1

ldi R22, 0x02;//для следующего захода ставим 2, чтобы PORTD поменял состояние

call mode1

rjmp choose\_mode

check\_1:

cbi PORTD, 6;устанавливаем бит, указывающий состояние 1 в 0

sbi PORTD, 7;устанавливаем бит, указывающий состояние 2 в 1

ldi R22, 0x01;//для следующего захода ставим 1, чтобы PORTD поменял состояние

call mode1

rjmp choose\_mode

next\_mode1:

CPI R18, 0x02

BRNE next\_mode2

cpi R24, 0x02

BREQ do\_2

;подготовка к началу использования режима

ldi R22, 0x01;подготовка=начало работы, поэтому первое состояние режима

LDI R24, 0x02;показываем, что режим используется

LDI R23, 0x55;подать во все порты 0х55

OUT PORTA, R23

OUT PORTB, R23

OUT PORTC, R23

do\_2:;режим уже используется

sbi PORTD, 5;устанавливаем 5 разряд - режим работы 2

cpi R22, 0x01

brne check\_2

cbi PORTD, 7;устанавливаем бит, указывающий состояние 2 в 0

sbi PORTD, 6;устанавливаем бит, указывающий состояние 1 в 1

ldi R22, 0x02;//для следующего захода ставим 2, чтобы PORTD поменял состояние

call mode2

rjmp choose\_mode

check\_2:

cbi PORTD, 6;устанавливаем бит, указывающий состояние 1 в 0

sbi PORTD, 7;устанавливаем бит, указывающий состояние 2 в 1

ldi R22, 0x01;//для следующего захода ставим 1, чтобы PORTD поменял состояние

call mode2

rjmp choose\_mode

next\_mode2:

cpi R24, 0x03

BREQ do\_3

;подготовка к началу использования режима

ldi R25, 0x03;//при этом числе программа потушит PORTC и зазжет PORTA

LDI R24, 0x03;показываем, что режим используется

LDI R23, 0xFF;подать на первый порт 0х55

OUT PORTA, R23

LDI R23, 0x00;подать на второй и третий порты 0х00

OUT PORTB, R23

OUT PORTC, R23

do\_3:

sbi PORTD, 5;устанавливаем 5 и 4 разряды - режим работы 3

sbi PORTD, 4

call mode3;в результате в R23 0х00

rjmp choose\_mode

mode1:

;меняем состояние (горят/не горят)

COM R23

;ldi R23, 0xff

OUT PORTA, R23

OUT PORTB, R23

OUT PORTC, R23

call choose\_time

ret

mode2:

;меняем состояние (горят/не горят)

COM R23

OUT PORTA, R23

OUT PORTB, R23

OUT PORTC, R23

call choose\_time

ret

mode3:

clr R23

cpi R25, 0x01

brne two

;нужно потушить первую восьмерку и зажечь вторую

out PORTA, R23;тушим светодиоды

com R23;ставим 0хff

out PORTB, R23;зажигаем вторую восьмерку

inc R25;указываем, что теперь горит второй светодиод

;показываем, что теперь второй режим работы

sbi PORTD, 7;устанавливаем бит, указывающий состояние 2 в 1

cbi PORTD, 6;устанавливаем бит, указывающий состояние 1 в 0

call choose\_time

ret

two:

cpi R25, 0x02

brne three

;нужно потушить вторую восьмерку и зажечь третью

out PORTB, R23;тушим светодиоды

com R23;ставим 0хff

out PORTC, R23;зажигаем третью восьмерку

inc R25;указываем, что теперь горит третий светодиод

;показываем, что теперь третий режим работы

sbi PORTD, 7;устанавливаем бит, указывающий состояние 2 в 1

sbi PORTD, 6;устанавливаем бит, указывающий состояние 1 в 0

call choose\_time

ret

three:

;нужно потушить третью восьмерку и зажечь первую

out PORTC, R23;тушим светодиоды

com R23;ставим 0хff

out PORTA, R23;зажигаем третью восьмерку

ldi R25, 0x01;указываем, что теперь горит первый светодиод

;показываем, что теперь первый светодиод

cbi PORTD, 7;устанавливаем бит, указывающий состояние 2 в 0

sbi PORTD, 6;устанавливаем бит, указывающий состояние 1 в 1

call choose\_time

ret

choose\_time:

;узнаем значение x, чтобы лампочки горели нужного время

CPI R19, 0x03

BRNE next\_1

CALL wait\_1;ждем 1 сек

ret

next\_1:

CPI R19, 0x02

BRNE next\_2

CALL wait\_0\_5;ждем 2 сек

ret

next\_2:

CALL wait\_0\_25;ждем 4 сек

ret

wait\_1:;z=41 y=205 x=127

sbi PORTD, 1;устанавливаем бит, указывающий состояние 2 в 1

sbi PORTD, 0;устанавливаем бит, указывающий состояние 1 в 1

LDI R28, 0x29

LDI R27, 0xcd

LDI R26, 0x7f

NOP;для того, чтобы ожидание было ровно секунду

CALL waiting

RET

wait\_0\_5:;z=82 y=140 x=0

sbi PORTD, 1;устанавливаем бит, указывающий состояние 2 в 1

cbi PORTD, 0;устанавливаем бит, указывающий состояние 1 в 0

LDI R28, 0x52

LDI R27, 0x8c

LDI R26, 0x00

CALL waiting

RET

wait\_0\_25:;z = 164 y = 9 x = 2

;показываем, что теперь первый режим работы

cbi PORTD, 1;устанавливаем бит, указывающий состояние 2 в 0

sbi PORTD, 0;устанавливаем бит, указывающий состояние 1 в 1

LDI R28, 0xa4

LDI R27, 0x09

LDI R26, 0x02

CALL waiting

RET

waiting:

DEC R26

BRNE waiting

DEC R27

CPI R27, 0x0f

BRNE waiting

DEC R28

BRNE waiting

RET

EXT\_INT1:

in R15, SREG

inc R19;уменьшить время смены состояний

cpi R19, 0x04;проверяем, что номер режима не вышел за пределы

BREQ change\_x;если вышел, то устанавливаем состояние 1

;Z уже подготовлен. он сначала принимает значение и оно константное

;clr R1;Очистить старший байт регистровой пары

;mov R0, R19;Установить младший байт Z - значение состояния

;spm;записываем в eeprom по адресу для x значение Z

call EEPROM\_write

out SREG, R15

reti

change\_x:

ldi R19, 0x01

;Z уже подготовлен. он сначала принимает значение и оно константное

call EEPROM\_write

out SREG, R15

reti

EXT\_INT0:

in R15, SREG

inc R18;изменяем режим работы

cpi R18, 0x04;проверяем, что режим работы не вышел за границы

BREQ change\_y//если вышел, то устанавливаем режим 1

out SREG, R15

reti

change\_y:

ldi R18, 0x01

out SREG, R15

reti

EEPROM\_write:

;Дождитесь завершения предыдущей записи

;EEWE - бит, установка которого даёт команду контроллеру записать данные из регистра данных в определенный адрес EEPROM

;EECR - регистр управления EEPROM

sbic EECR,EEWE

rjmp EEPROM\_write

;Настройка адреса (r18:r17) в регистре адресов

;EEARH отвечает за старший байт адреса ячейки памяти, EEARL за младший байт

out EEARH, r18

out EEARL, r17

;Запись данных (r16) в регистр данных

out EEDR,r19

sbi EECR,EEMWE

sbi EECR,EEWE

ret