1 Формулировка задания

Регистры PORTA–PORTB выполняют роль новогодней гирлянды.

Есть следующие 3 режима работы:

1. на каждом регистре два состояния – вывод чисел 0xFF и 0x00, смена состояний с частотой x;
2. на каждом регистре два состояния – вывод чисел 0xAA и 0x55, смена состояний с частотой x;
3. на каждом регистре два состояния – вывод чисел y и -y*, смена состояний с частотой x.

Регистры PD4-PD5 отображают номер режима работы (1-3), регистр PD6 отображают номер состояния в конкретном режиме (0/1), регистры PD0-PD1 отображают номер элемента x в множестве частот.

Ввод числа y должен выполняться на PORTC (одновременным нажатием на несколько кнопок) при нажатой кнопке PD7. Нажатие кнопки PD7 может (а вернее должно) приводить к остановке основного функционала (вывод информации на светодиоды) до момента отпускания всех кнопок на PORTC или отпускания PD7, после чего основной функционал возобновиться с того же режима и состояния, но с новым значением y.

Смена режима работы должна производиться циклически с помощью кнопки PD2 (прерывание INT0), при этом номер активного состояния должен сохраняться.

С помощью кнопки PD3 (прерывание INT1) циклически изменяется величина x в следующем множестве {0,25 Гц; 0,5 Гц; 1 Гц}. Изменения должны отображаться на регистре PORTD и сохраняться во внутренний EEPROM МК в момент нажатия на кнопки PD2 и PD3. Исходное положение: первый режим и первое состояние (0xFF на регистрах PORTA-PORTC), величина x извлекается из EEPROM, место и формат хранения выбирается самостоятельно, y = 0x55.

2 Схема лабораторной установки

Схема лабораторной установки показана на Рисунок 1.

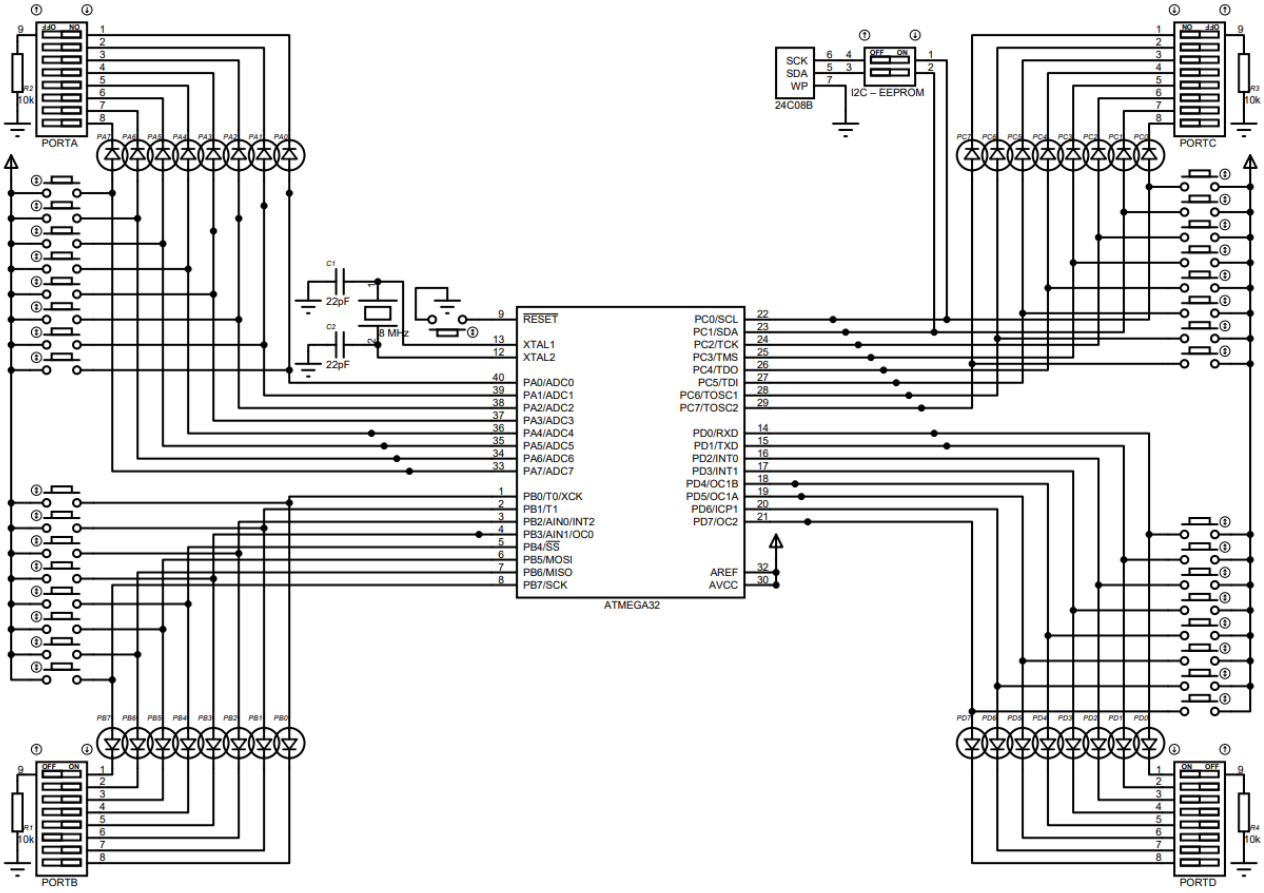


Рисунок 1 – Схема лабораторной установки

3 Блок-схема алгоритма работы программы

Блок-схема алгоритма работы программы показана на Рисунок 2. Блок-схемы подпрограмм, используемых в алгоритме, показаны на Рисунок 3-7.

Рисунок 2 – Блок-схема алгоритма работы программы

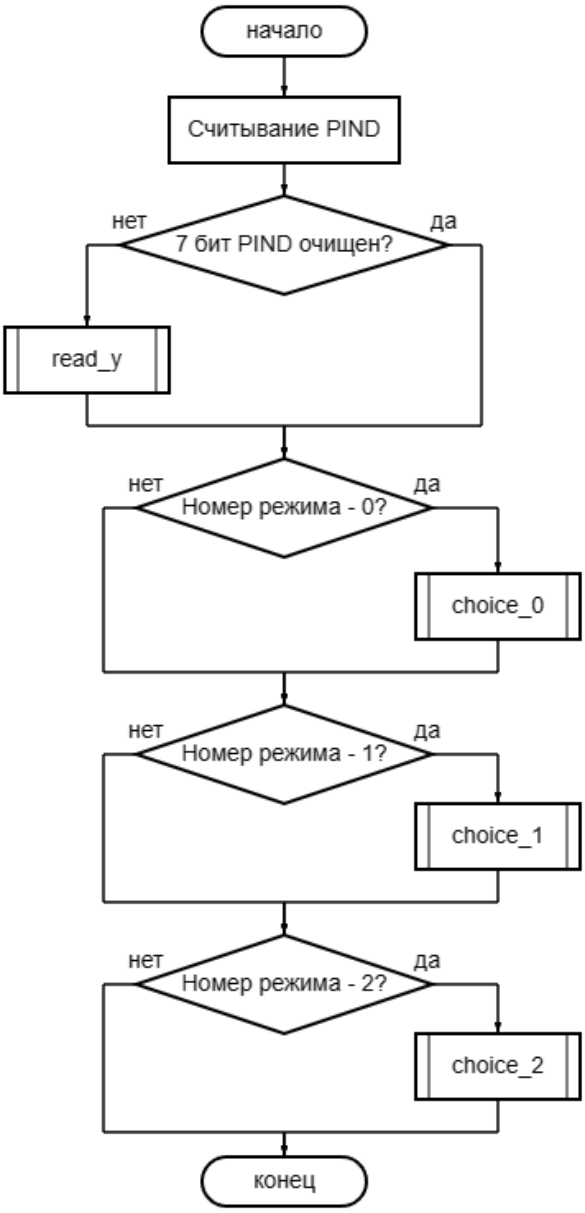


Рисунок 3 – Блок-схема подпрограммы choice выбора режима работы

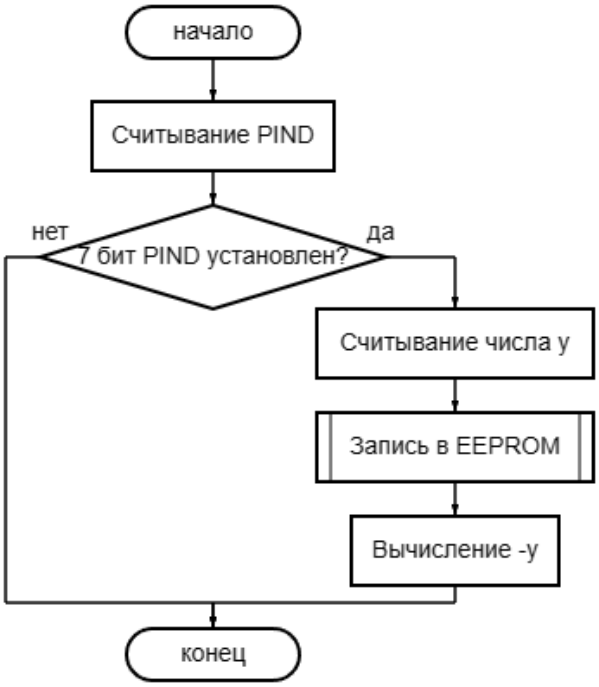


Рисунок 4 – Блок-схема подпрограммы read_y чтения числа y

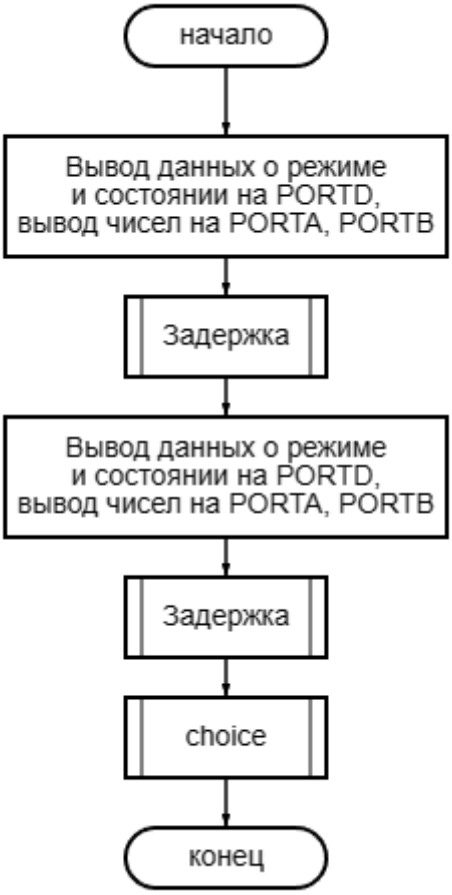


Рисунок 5 – Блок-схема подпрограмм choice_X работы в режиме X

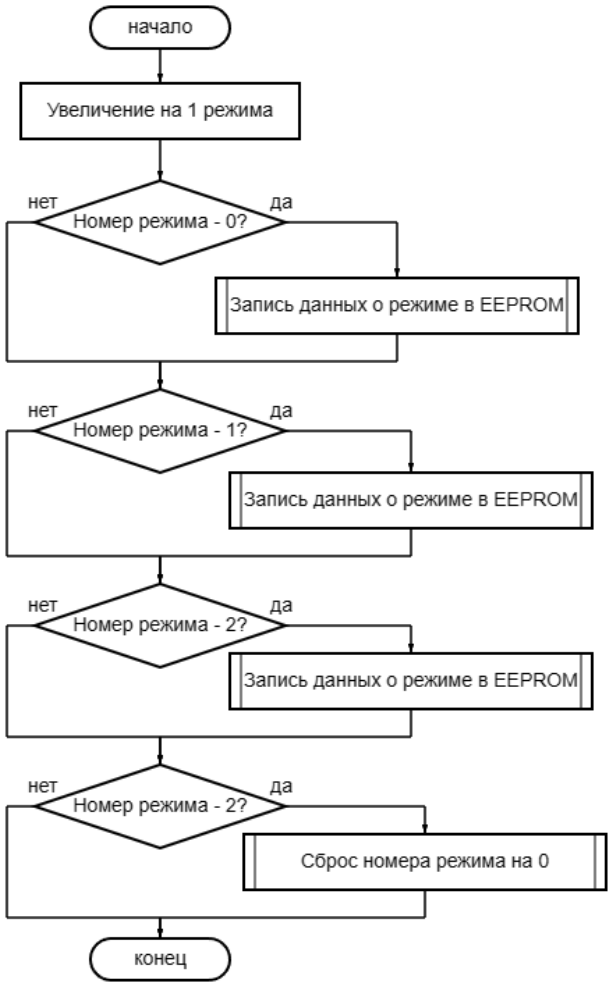


Рисунок 6 – Блок-схема подпрограммы INT0_HANDLER обработки прерывания int0 (изменение пользователем режима)

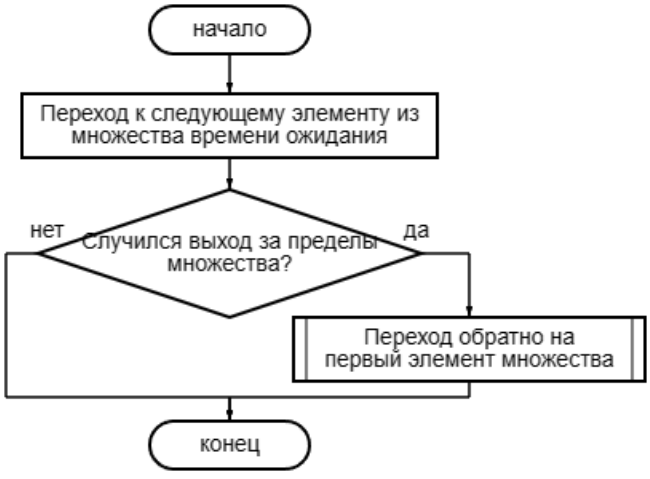
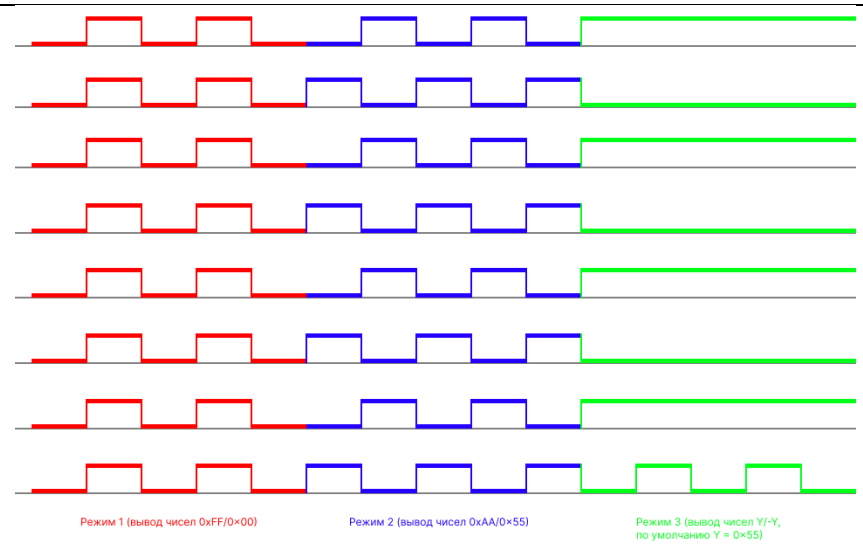
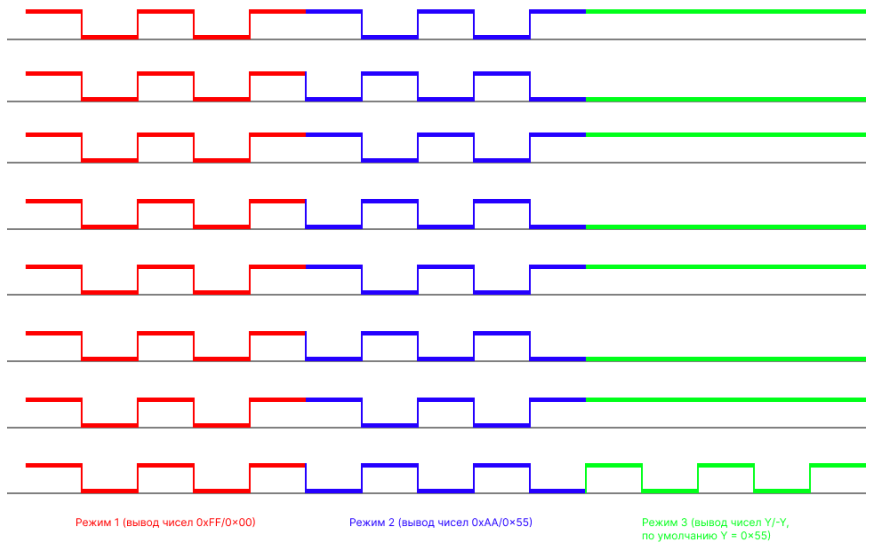
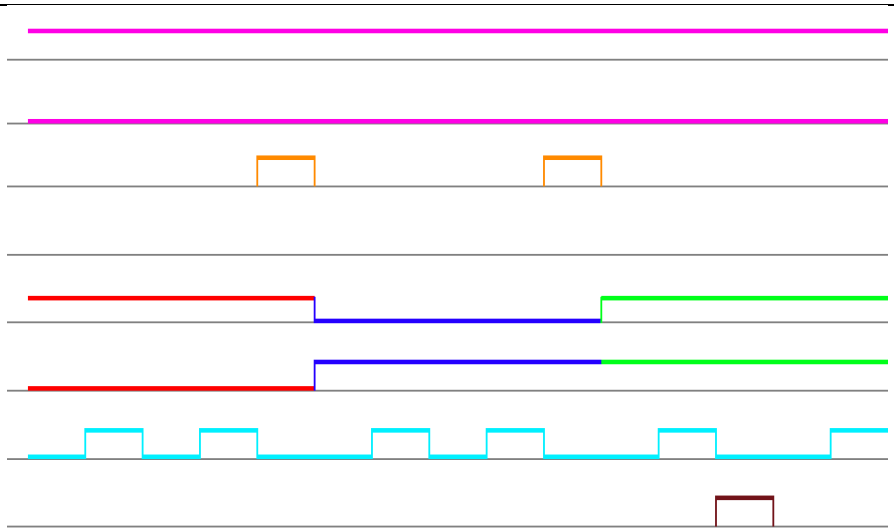


Рисунок 7 – Блок-схема подпрограммы INT1_HANDLER обработки прерывания int1 (изменение пользователем времени ожидания)

4 Ход работы

4.1 Временная диаграмма цифровых сигналов на портах ввода-вывода (фрагмент)

Вывод	Диаграмма	Назначение
PA0 PA1 PA2 PA3 PA4 PA5 PA6 PA7		Вывод чисел
PB0 PB1 PB2 PB3 PB4 PB5 PB6 PB7		Вывод чисел

PC0 PC1 PC2 PC3 PC4 PC5 PC6 PC7		Ввод числа Y
PD0 PD1 PD2 PD3 PD4 PD5 PD6 PD7		Вывод номера текущей частоты в множестве (0-1) Изменение режима (2) Изменение частоты (3) Вывод номера режима (4-5) Вывод текущего состояния (6) Начать ввод Y (7)

5 Ответы на контрольные вопросы

1. Какими способами можно подключить внешние устройства (светодиод, кнопку) к микроконтроллеру?

Подключить светодиод можно двумя способами, как показано на Рисунок 8.

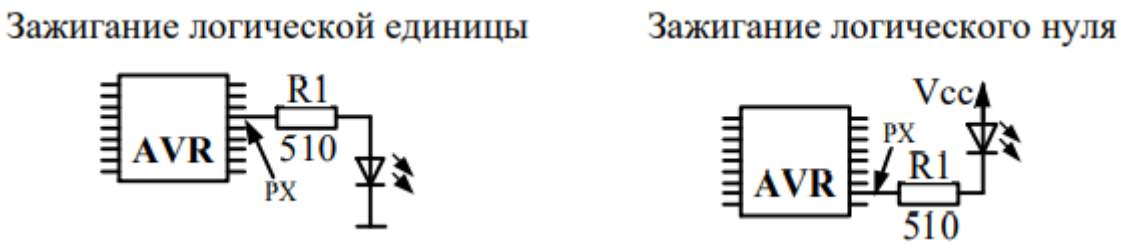


Рисунок 8 – Способы подключения светодиода

В первом случае для включения светодиода нужно сконфигурировать порт на вывод, и записать в него единицу. Во втором случае нужно сконфигурировать порт на ввод, и записать в него ноль.

Вариант подключения кнопки изображён на Рисунок 9.

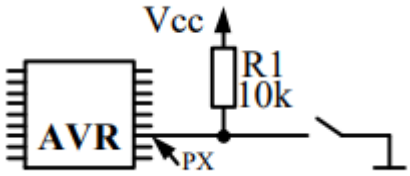


Рисунок 9 – Вариант подключения кнопки

Для корректной работы кнопки регистр ввода-вывода должен быть сконфигурирован на ввод. Когда кнопка отпущена, с входа микроконтроллера будет считываться логическая единица, т. к. вход подтянут резистором к линии питания. Когда кнопка нажата, то линия питания соединяется с землёй через резистор, при этом из регистра PINx будет считан логический ноль.

2. Как реализуется подсистема прерываний в микроконтроллере AVR?

Все микроконтроллеры AVR имеют многоуровневую систему прерываний. Подсистема прерываний состоит из нескольких объектов: источники прерываний, векторы прерываний, таблица векторов прерываний, биты активации прерываний и глобальный бит активации прерываний.

Прерывания приостанавливают выполнение программы для обработки приоритетных задач. У каждого периферийного устройства есть как минимум один источник прерывания, которому соответствует вектор в таблице векторов прерываний. Чтобы использовать прерывание, необходимо установить его бит активации. Для начала обработки прерываний также необходимо разрешить

глобальные прерывания в регистре состояния SREG. Чтобы запретить прерывания, необходимо сбросить бит глобального разрешения прерываний.

3. Как программно разрешить или запретить выполнение конкретного прерывания?

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.

4. Какие источники прерываний есть в микроконтроллерах AVR?

Перечень источников прерываний для МК ATmega32:

- RESET – Сигнал сброса.
- INT0 – Внешний запрос на прерывание по входу INT0
- INT1 – Внешний запрос на прерывание по входу INT1
- INT2 – Внешний запрос на прерывание по входу INT2
- TIMER2_COMP – Совпадение с регистром сравнения таймера T/C2
- TIMER2_OVF – Переполнение счётчика T/C2
- TIMER1_CAPT – Захват по таймеру T/C1
- TIMER1_COMPA – Совпадение с регистром сравнения A таймера T/C1
- TIMER1_COMPB – Совпадение с регистром сравнения B таймера T/C1
- TIMER1_OVF – Переполнение счётчика T/C1
- TIMER0_COMP – Совпадение с регистром сравнения таймера T/C0

- TIMER0_OVF – Переполнение счётчика T/C0
- SPI_STC – Передача данных по интерфейсу SPI завершена
- UART_RXC – Приём данных приёмопередатчиком UART завершён
- UART_UDRE – Регистр данных UART пуст
- UART_TXC – Передача данных приёмопередатчиком UART завершена
- TWI – Прерывание от интерфейса I2C
- ADC – Завершено преобразование АЦП
- EE_RDY – EEPROM готов
- ANA_COMP – Прерывание от аналогового компаратора
- SPM_RDY – Запись программной памяти (Flash) готова

5. Как настраиваются внешние прерывания?

За управление внешними прерываниями в ATmega32 отвечают четыре регистра:

– GICR (он же GIMSK) – запрет/разрешение прерываний по сигналам на входах INT0, INT1;

– MCUCR – выбор условия срабатывания прерываний int0 и int1;

– GIFR – управление внешними прерываниями;

6 Выводы

В ходе лабораторной работы были изучены основы работы с цифровыми портами ввода-вывода микроконтроллера ATmega32 и получены практические навыки по обработке внешних прерываний и организации ввода-вывода с помощью механизма прерываний.

ПРИЛОЖЕНИЕ А

Листинг программы на языке ассемблера

```
.def TMP = R20
.def N1R = R0
.def N11R = R1
.def N2R = R2
.def N22R = R5
.def N3R = R3
.def N33R = R4
.def SR_REG = R12

.equ NUM1 = 0xFF ; режим 1
.equ NUM11 = 0x00
.equ NUM2 = 0xAA ; режим 2
.equ NUM22 = 0x55
; режим 3: y/-y 0x55 по умолчанию, прямой код

.equ ADDR_MODE = 0 ; адрес ячейки EEPROM для хранения режима


.org $000
JMP  reset; Указатель на начало программы


.org INT0addr
JMP INT0_HANDLER ; Указатель на обработчик прерывания int0
.org INT1addr
JMP INT1_HANDLER ; Указатель на обработчик прерывания int1


EEPROM_read:
; Wait for completion of previous write
SBIC EECR,EEWE
RJMP EEPROM_read
; Set up address (r18:r17) in address register
```

```
OUT EEARH, r18
OUT EEARL, r17
; Start eeprom read by writing EERE
SBI EECR,EERE
; Read data from data register
IN r16,EEDR
RET

EEPROM_write:
; Wait for completion of previous write
SBIC EECR,EEWE
RJMP EEPROM_write
; Set up address (r18:r17) in address register
OUT EEARH, r18
OUT EEARL, r17
; Write data (r16) to data register
OUT EEDR, r16
; Write logical one to EEMWE
SBI EECR,EEMWE
; Start eeprom write by setting EEWE
SBI EECR,EEWE
RET

delay:
     CPI R31, 0
     BREQ base_delay
     CPI R31, 1 ; сравнение
     BREQ first_time
     CPI R31, 2
     BREQ second_time
     CPI R31, 3
     BREQ third_time
     ret

base_delay:;2Гц начальная частота
   LDI R25, 10;  y
```

```
   LDI R28, 255; x
   LDI R29, 255
delay_sub:
   DEC R29
   BRNE delay_sub
   DEC R28
   BRNE delay_sub
   DEC R25
   BRNE delay_sub
   NOP
   NOP
   RET

first_time:;0.25Гц
LDI R16, 1
LDI R18, 0
LDI R17, 1
call EEPROM_write
LDI R21, 0b00000001
OR R21, R19; + смотрим состояние в R19
OUT PORTD, R21
LDI R25, 5
 LDI R28, 223
 LDI R29, 188
delay_sub_250:
 DEC R29
 NOP
 NOP
 NOP
 BRNE delay_sub_250
 DEC R28
 BRNE delay_sub_250
 DEC R25
 BRNE delay_sub_250
 NOP
 NOP
```

```
RET

second_time:;0.5Гц
LDI R16, 2
LDI R18, 0
LDI R17, 1
call EEPROM_write
LDI R23, 0b00000010
OR R23, R19; + смотрим состояние в R19
OUT PORTD, R23
LDI R25, 10
 LDI R28, 240
 LDI R29, 188
delay_sub_500:
 DEC R29
 NOP
 NOP
 NOP
 BRNE delay_sub_500
 DEC R28
 BRNE delay_sub_500
 DEC R25
 BRNE delay_sub_500
 NOP
 NOP
RET

third_time:;1Гц
LDI R16, 3
LDI R18, 0
LDI R17, 1
call EEPROM_write
LDI R24, 0b00000011
OR R24, R19; + смотрим состояние в R19
OUT PORTD, R24
 LDI R25, 20
```

```
 LDI R28, 240
 LDI R29, 188
delay_sub_1000:
 DEC R29
 NOP
 NOP
 NOP
 BRNE delay_sub_1000
 DEC R28
 BRNE delay_sub_1000
 DEC R25
 BRNE delay_sub_1000
 NOP
 NOP
RET

; Начальная настройка
reset:
; Запись чисел для работы в регистры, оттуда их будет читать программа в соответствующем режиме
 LDI TMP, NUM1
 MOV N1R, TMP
 LDI TMP, NUM2
 MOV N2R, TMP
 LDI TMP, NUM22
 MOV N22R, TMP
 CLR TMP
 LDI TMP, NUM22;
 MOV N3R, TMP; y
 MOV N33R, TMP
 LDI TMP, 0b10000000
 ADD N33R, TMP
 CLR TMP
  ; настройка портов ввода-вывода
 LDI TMP, NUM11 ; 0x00
 MOV N11R, TMP ; Вход
```

```
 OUT DDRC, TMP
 CLR TMP;
 SER TMP ; 0xFF
 OUT DDRA, TMP ; Выход
 OUT DDRB, TMP ; Выход
 CLR TMP;
 LDI TMP, 0b01110011
 OUT DDRD, TMP ; Часть битов на вход, часть на выход (см. задание)
 ;Настройка указателя стека
 LDI R22, 0b00000000
LDI R22, HIGH(RAMEND) ; Старшие разряды адреса. HIGH(RAMEND)- адрес вершины стека
OUT SPH, R22 ; SPH (Stack Pointer High) - старший регистр указателя стека
LDI R22, LOW(RAMEND)  ; Младшие разряды адреса
OUT SPL, R22 ;SPL (Stack Pointer Low) - младший регистр указателя стека
; Настройка регистров для работы с прерываниями
LDI R16, 0x0A
OUT MCUCR, R16 ; int0 и int1 работают на перепаде 0/1
LDI R16, 0xC0
OUT GICR, R16 ; разрешить int0 и int1
LDI R16, 0x00
OUT GIFR, R16 ; Предотвращение срабатывания int0 и int1 при включении прерываний
SEI ; Глобальное разрешение прерываний

LDI R18, 0
LDI R17, 0 ; (r18:r17) занесется в регистр адреса
call EEPROM_read ; считали данные из EEPROM в R16, потом запишем их в R30 - какой режим?
MOV R30, R16

LDI R18, 0
LDI R17, 1; (r18:r17) занесется в регистр адреса
```

```
call EEPROM_read ; считали данные из EEPROM в R16, потом запишем их в R31 - какая частота?
MOV R31, R16

LDI R18, 0
LDI R17, 2 ; (r18:r17) занесется в регистр адреса
call EEPROM_read ; считали данные из EEPROM в R16 - какое число y?
; тут восстанавливается y из памяти
CLR N3R
MOV N33R, R16
MOV N3R, R16
LDI TMP, 0b10000000
ADD N33R, TMP ; меняем знаковый бит числа - прямой код
CLR TMP


choice:
IN TMP, PIND ; Считывание PIND в TMP
SBRC TMP, 7; Если 7 бит PIND очищен, то следующая команда пропускается
CALL read_y ;
CPI R30, 0
BREQ choice_0
CPI R30, 1
BREQ choice_1
CPI R30, 2
BREQ choice_2

read_y:
   IN TMP, PIND ; Считывание PIND в TMP
   SBRC TMP, 7 ; Если 7 бит PIND установлен, следующая команда выполняется
   RJMP y_in_mode ; Переход если кнопка отпущена
   RET

y_in_mode:
```

```
; считали число
   IN N3R, PINC
   MOV N33R, N3R
   ; сохранили в EEPROM
   MOV R16, N3R
   LDI R18, 0
   LDI R17, 2
   call EEPROM_write
   ; Вычислили -y (как было выше)
   LDI TMP, 0b10000000
   ADD N33R, TMP
   CLR TMP

   RJMP read_y

 choice_0: ; 0xFF/0x00

LDI R19, 0b00010000
OUT PORTD, R19
OUT PORTA, N1R
OUT PORTB, N11R
 call delay
LDI R19, 0b01010000
OUT PORTD, R19
OUT PORTA, N11R
OUT PORTB, N1R
  call delay
 jmp choice

  choice_1: ; 0xAA/0x55

LDI R19, 0b00100000
OUT PORTD, R19
OUT PORTA, N2R
OUT PORTB, N22R
 call delay
```

```
LDI R19, 0b01100000
OUT PORTD, R19
OUT PORTA, N22R
OUT PORTB, N2R
  call delay
 jmp choice

choice_2: ; y/-y

LDI R19, 0b00110000
OUT PORTD, R19
out PORTA, N3R
out PORTB, N33R
 call delay
LDI R19, 0b01110000
OUT PORTD, R19
out PORTA, N33R
out PORTB, N3R
  call delay
 jmp choice

ch_mode_0:
ldi R30, 0
LDI R16, 0
LDI R18, 0
LDI R17, 0
call EEPROM_write

     POP R15 ; Восстановление значения R16 из стека
     OUT SREG, R15 ; Восстановление значения SREG из стека
reti

ch_mode_1:
ldi R30, 1
ldi R16, 1
LDI R18, 0
```

```
LDI R17, 0
call EEPROM_write

     POP R15 ; Восстановление значения R16 из стека
     OUT SREG, R15 ; Восстановление значения SREG из стека
reti

ch_mode_2:
ldi R30, 2
ldi R16, 2
LDI R18, 0
LDI R17, 0
call EEPROM_write

     POP R15 ; Восстановление значения R16 из стека
     OUT SREG, R15 ; Восстановление значения SREG из стека
reti


INT0_HANDLER:
     IN R15, SREG
     PUSH R15 ; Сохранение текущего значения SREG в стеке

    INC R30
     CPI R30, 0
     BREQ ch_mode_0
     CPI R30, 1
     BREQ ch_mode_1
     CPI R30, 2
     BREQ ch_mode_2
     CPI R30, 3
     BREQ ch_mode_0

;int0_new:
;;   LDI R30, 0
     ;RETI
```

```
INT1_HANDLER:
     IN R15, SREG
     PUSH R15 ; Сохранение текущего значения SREG в стеке

     INC R31              ;Изменить время ожидания
     CPI R31, 4           ;Проверить, что время ожидания не вышло за
пределы {0.25, 0.5, 1.0}
     BREQ x_of

     POP R15 ; Восстановление значения R16 из стека
     OUT SREG, R15 ; Восстановление значения SREG из стека

     RETI             ; Возврат из обработчика прерываний

x_of:
     LDI R31, 1

     POP R15 ; Восстановление значения R16 из стека
     OUT SREG, R15 ; Восстановление значения SREG из стека
     RETI             ; Возврат из обработчика прерываний
```