Министерство образования и науки Российской Федерации

Санкт-Петербургский Политехнический Университет Петра Великого

—

Институт кибербезопасности и защиты информации

ЛАБОРАТОРНАЯ РАБОТА № 2

Вариант 10

1. «Работа с таймерами-счетчиками»
2. по дисциплине «Аппаратные средства вычислительной техники»
3. Выполнил
4. студент гр. 4851004/10001 Тоцкий В.

<*подпись*>

Проверил:

1. Старший преподаватель Макаров И. Д.

<*подпись*>

1. Санкт-Петербург
2. 2023

# Формулировка задания

Разрядам регистров PORTA и PORTB ставится в соответствие отрезок [0–15] (0-7 ~ PA0–PA7, 8-15 ~ PB0-PB7). На данном отрезке отображается «бегущий огонь» из нескольких горящих светодиодов (восьмиразрядное число y и -y\*, например, если y=53=0x35=0b0011.0101, то в варианте с обратным кодом «бегущий огонь» выглядит как 0b0011.0101.1100.1010). Для каждого состояния активные светодиоды загорается на 0,2 с, после чего активным становится другие светодиоды. Номер следующего активного светодиода определяется по формуле: ni+1 = ni+1 + x (mod 16), где ni+1 – номер текущего светодиода, ni+1 – номер следующего светодиода, x – величина шага, лежащая в интервале [-3;+3]. Если величина шага x > 0, то PD0 = 0 и PD1 = 1, если величина шага x = 0, то PD0 = 1 и PD1 = 1, если величина шага x < 0, то PD0 = 0 и PD1 = 1; PD4-PD5 отображают значение |x|.

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.

Кнопки PD2 (прерывание INT0) и PD3 (прерывание INT1) осуществляют изменение величины шага x на +1 (mod 4) и -1 (mod 4) соответственно.

Изменение величины шага должно отображаться на регистре PORTD и сохраняться во внутренний EEPROM МК в момент нажатия на кнопки PD2 и PD3, изменение логики отображения – со следующего шага «бегущего огня». Исходное положение: PORTA = y = 0x71, величина шага x извлекается из EEPROM, место и формат хранения выбирается самостоятельно.

# Схема лабораторной установки

Изображение выглядит как текст, диаграмма, План, линия

Автоматически созданное описание

Рисунок

# Блок схема разработанного алгоритма программы

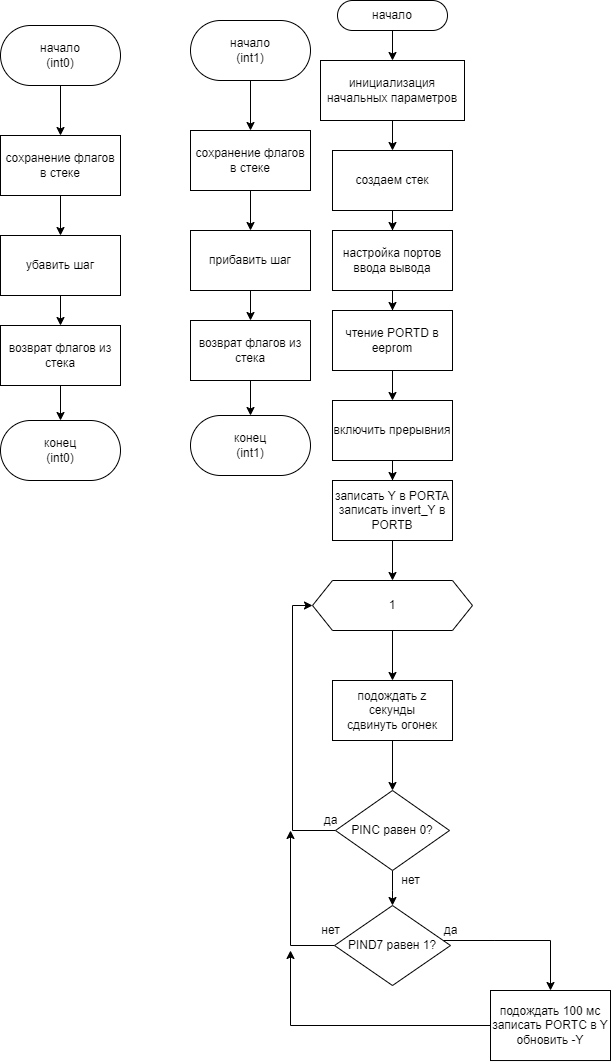


Рисунок – Блок-схема алгоритма

# Временная диаграмма цифровых сигналов на портах ввода-вывода

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок – Схема работы портов вывода

# Ответы на контрольные вопросы

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

Внешние устройства можно подключать к микроконтроллеру ATmega32 с помощью портов ввода-вывода (GPIO). Для этого необходимо подключить провода к соответствующим выводам микроконтроллера и настроить соответствующие регистры. Например, для подключения светодиода необходимо подключить его анод к выводу порта ввода-вывода, а катод - к земле. Для подключения кнопки необходимо подключить один вывод к питанию, а другой - к входу порта ввода-вывода с подтяжкой или без нее.

1. Как реализуется подсистема прерываний в микроконтроллере AVR?

Подсистема прерываний в микроконтроллере AVR ATmega32 реализуется с помощью векторной таблицы прерываний. Когда происходит прерывание, микроконтроллер автоматически переходит к соответствующему обработчику прерывания, который находится по адресу, указанному в соответствующей ячейке векторной таблицы.

1. Как программно разрешить или запретить выполнение конкретного прерывания?

Для разрешения или запрещения выполнения конкретного прерывания необходимо настроить соответствующий бит в регистре управления прерываниями (например, SREG или TIMSK). Если бит установлен, то прерывание разрешено, если сброшен - запрещено.

1. Какие источники прерываний есть в микроконтроллерах AVR?
   1. INT0 и INT1: Это внешние прерывания, которые могут быть настроены на падающий, нарастающий или оба фронта сигнала на входе. INT0 соответствует пину PD2, а INT1 - пину PD3.
   2. INT2: Это еще одно внешнее прерывание, которое может быть настроено на падающий, нарастающий или оба фронта сигнала на входе. INT2 соответствует пину PB2.
   3. PCINT0-7: Это прерывания по изменению состояния на пине, которые могут быть настроены для каждого пина в портах PCINT0 и PCINT1.
   4. PCINT8-15: Это также прерывания по изменению состояния на пине, которые могут быть настроены для каждого пина в портах PCINT2 и PCINT3.
   5. TIMER1 COMPA и TIMER1 COMPB: Эти прерывания могут быть настроены на совпадение счетчика таймера 1 с определенными значениями, заданными в регистрах OCR1A и OCR1B.
   6. TIMER1 OVF: Это прерывание возникает при переполнении таймера 1.
   7. TIMER0 OVF: Это прерывание возникает при переполнении таймера 0.
   8. USART RX Complete: Это прерывание возникает при завершении приема данных через USART.
2. Как настраиваются внешние прерывания?

Выбрать вход, который будет использоваться для внешнего прерывания. Например, для использования входа INT0 необходимо подключить устройство к выводу PD2. Настроить регистр MCUCR (MCU Control Register) для выбора источника прерывания и настройки режима работы. Для входа INT0 можно выбрать режим срабатывания по фронту или спаду сигнала, а также выбрать уровень срабатывания (например, низкий или высокий). Настроить регистр GICR (General Interrupt Control Register) для разрешения прерывания от выбранного входа. Например, для входа INT0 необходимо установить бит INT0 в этом регистре. Написать обработчик прерывания, который будет вызываться при срабатывании внешнего прерывания. Обработчик должен содержать необходимые действия по обработке сигнала от устройства. Установить флаг прерывания после обработки сигнала, чтобы микроконтроллер знал, что прерывание было обработано и можно перейти к основной программе.

# Выводы по лабораторной работе

В ходе работы изучены основы работы с цифровыми портами ввода-вывода микроконтроллера ATmega32. Получены практические навыки по обработке внешних прерываний и орган изации ввода-вывода с помощью механизма прерываний.

Приложение А

Листинг программы «бегущий огонь.asm»

;C:\Users\Денис\Documents\Atmel Studio\7.0\running\_fire\running\_fire\Debug\running\_fire.hex

.org $000

JMP zeroing

.def NULL = R28

.def TMP = R27

.def TMP2 = R26

.def COUNT = R25

.def \_Y = R22

.def invert\_Y = R21

.def \_x = R20

.org INT0addr

JMP INT0\_

.org INT1addr

JMP INT1\_

zeroing:

CLR NULL ; 0x00

ldi \_x, 0x1

OUT PINA, NULL

OUT PINB, NULL

OUT PIND, NULL

OUT PINC, NULL

OUT PORTA, NULL

OUT PORTB, NULL

OUT PORTD, NULL

OUT PORTC, NULL

OUT DDRA, NULL

OUT DDRB, NULL

OUT DDRC, NULL

OUT DDRD, NULL

reset:

CLR NULL

ldi \_Y, 0x71

mov invert\_Y, \_Y

SER TMP ; 0xFF

eor invert\_Y, TMP ; ислючающее или

; настройка портов ввода-вывода

SER TMP ; 0xFF

OUT DDRA, TMP ; Вывод

OUT DDRB, TMP ; Вывод

LDI TMP, 0b00110011 ; там где 0 там ввод, там где 1 вывод

OUT DDRD, TMP

CLR TMP ;0x00

OUT DDRC, TMP; ввод

LDI TMP, 0x0F

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

LDI TMP, 0xC0

OUT GICR, TMP ; Разрешение прерываний int0 и int1

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

; Установка вершины стека в конец ОЗУ

LDI TMP, HIGH(RAMEND) ; Старшие разряды адреса

OUT SPH, TMP

LDI TMP, LOW(RAMEND) ; Младшие разряды адреса

OUT SPL, TMP

ldi R16,0 ; Загружаем адрес нулевой ячейки

ldi R17,0 ; EEPROM

rcall EERead ; вызываем процедуру чтения.

call write\_POTRD

SEI ; Включение прерываний

jmp loop

loop:

call print

call delay

read\_number: ; считывание операнда

SBIS PIND, 7 ; Пропуск следующей команды, если бит в порту установлен

jmp loop

IN TMP, PINC

CP TMP, NULL ; PINC = 0?

breq read\_number ; если PINC = 0 считывание операции

read\_PORTD:

call delay ; ожидание нажатия комбинации кнопок

IN TMP, PINC

CP TMP, NULL

breq stop\_reading

mov \_Y, TMP

SER TMP ; 0xFF

mov invert\_Y, \_Y

eor invert\_Y, TMP ; ислючающее или

out PORTA, \_Y

out PORTB, invert\_Y

jmp read\_PORTD

stop\_reading: ; обеспечение однократного ввода

SBIS PIND, 7 ; Пропуск следующей команды, если бит в порту установлен

jmp loop

jmp stop\_reading

INT0\_:

push TMP

in TMP, SREG

push TMP

ldi TMP, 3

cpse \_x, TMP

inc \_x

call write\_POTRD

ldi R16,0 ; Загружаем адрес нулевой ячейки

ldi R17,0 ; EEPROM

rcall EEWrite ; вызываем процедуру записи.

pop TMP

out SREG, TMP

pop TMP

RETI

INT1\_:

push TMP

in TMP, SREG

push TMP

ldi TMP, -3

cpse \_x, TMP

dec \_x

call write\_POTRD

ldi R16,0 ; Загружаем адрес нулевой ячейки

ldi R17,0 ; EEPROM

rcall EEWrite ; вызываем процедуру записи.

pop TMP

out SREG, TMP

pop TMP

RETI

write\_POTRD:

clr TMP ; 0x00

cp \_x, TMP ; сравниваем значение регистра r17 с 0

breq write\_11 ; переходим на метку write\_11, если значение равно 0

brge write\_01 ; переходим на метку write\_01, если значение больше или равно 0

jmp write\_10 ; переходим на метку write\_10, если значение меньше 0

write\_01: ; больше нуля

mov TMP, \_x

lsl TMP

lsl TMP

lsl TMP

lsl TMP

ori TMP, 0b00000001

out PORTD, TMP

mov TMP, \_x

RET

write\_10: ; меньше нуля

ser TMP ; 0xFF

muls TMP, \_x

mov TMP, R0

lsl TMP

lsl TMP

lsl TMP

lsl TMP

ori TMP, 0b00000010

out PORTD, TMP

RET

write\_11: ; равно нулю

ldi TMP, 0b00000011

out PORTD, TMP

RET

EEWrite:

SBIC EECR,EEWE ; Ждем готовности памяти к записи. Крутимся в цикле

RJMP EEWrite ; до тех пор пока не очистится флаг EEWE

CLI ; Затем запрещаем прерывания.

OUT EEARL,R16 ; Загружаем адрес нужной ячейки

OUT EEARH,R17 ; старший и младший байт адреса

OUT EEDR,\_x ; и сами данные, которые нам нужно загрузить

SBI EECR,EEMWE ; взводим предохранитель

SBI EECR,EEWE ; записываем байт

SEI ; разрешаем прерывания

RET

EERead:

SBIC EECR,EEWE ; Ждем пока будет завершена прошлая запись.

RJMP EERead ; также крутимся в цикле.

OUT EEARL, R16 ; загружаем адрес нужной ячейки

OUT EEARH, R17 ; его старшие и младшие байты

SBI EECR,EERE ; Выставляем бит чтения

IN \_x, EEDR ; Забираем из регистра данных результат

RET

print:

mov TMP, invert\_Y

mov TMP2, \_Y

mov COUNT, \_x

loop\_shift:

cp COUNT, NULL ; сравниваем COUNT с 0

breq continue\_loop\_shift ; переходим на метку continue\_loop\_shift, если значение равно 0

brge minus ; переходим на метку minus, если значение больше или равно 0

jmp plus ; переходим на метку plus, если значение меньше 0

plus:

lsl invert\_Y

rol TMP

brcs continue\_plus1 ;Переход при установленном флаге переноса (C=1)

ori invert\_Y, 0b00000001

continue\_plus1:

lsl \_Y

rol TMP2

brcs continue\_plus2 ;Переход при установленном флаге переноса (C=1)

ori \_Y, 0b00000001

continue\_plus2:

inc COUNT

jmp loop\_shift

minus:

lsr invert\_Y

ror TMP

brcs continue\_minus1 ;Переход при установленном флаге переноса (C=1)

ori invert\_Y, 0b10000000

continue\_minus1:

lsr \_Y

ror TMP2

brcs continue\_minus2 ;Переход при установленном флаге переноса (C=1)

ori \_Y, 0b10000000

continue\_minus2:

dec COUNT

jmp loop\_shift

continue\_loop\_shift:

out PORTA, \_Y

out PORTB, invert\_Y

ret

delay: ; задержка 0.2 мс

LDI R31, 8

LDI R30, 99

LDI R29, 88

delay\_sub:

DEC R29

BRNE delay\_sub

DEC R30

BRNE delay\_sub

DEC R31

BRNE delay\_sub

NOP

RET