# Формулировка задания

Разрядам регистров PORTA, PORTB и PORTC ставится в соответствие отрезок [0–23] (0-7 ~ PA0–PA7, 8-15 ~ PB0-PB7, 16-23 ~ PC0-PC7). На данном отрезке отображается «бегущий огонь» из набора горящих светодиодов. Для гирлянды существует базовое значение (набор одновременно горящих светодиодов), которое с определённой частотой циклически сдвигается на определённый шаг. Изменяемыми параметрами гирлянды являются:

 величина шага (h), на которую циклически сдвигается базовое значение каждый такт гирлянды. Допустимый диапазон величины шага [-12;12];

 частота смены состояний (p). Допустимый диапазон частоты смены состояний [1-15] (раз / в 2 секунды);

Переключение между режимами должно осуществляться циклически с помощью кнопки PD2 (прерывание INT0). Переключение между настраиваемыми параметрами должно осуществляться циклически с помощью кнопки PD3 (прерывание INT1). Имя изменяемого параметра должно отображаться на первом семисегментном индикаторе, на последующих индикаторах должно отображаться значение соответствующего параметра в шестнадцатеричной системе счисления. На последнем индикаторе имени параметра в качестве разделителя должна гореть точка. Имя параметра должно отображаться постоянно, а значение циклически загораться и гаснуть с периодами этих состояний 0,5 с. Изменение значений должно осуществляться с помощью потенциометра, подключённого к выводу PA5, следующим образом:

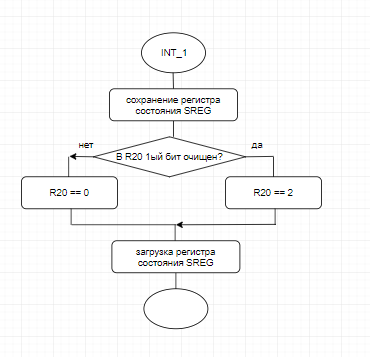
 крайнее левое положение соответствует нижней границе допустимого диапазона, крайнее правое – верхней;

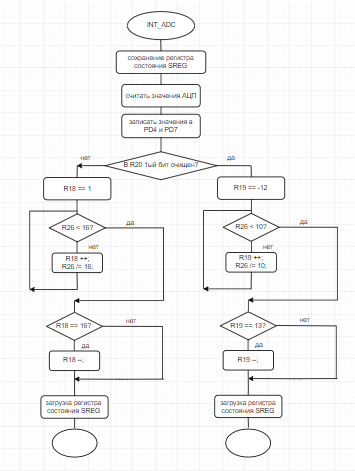
 при повороте потенциометра значение на семисегментном индикаторе изменяется незамедлительно.

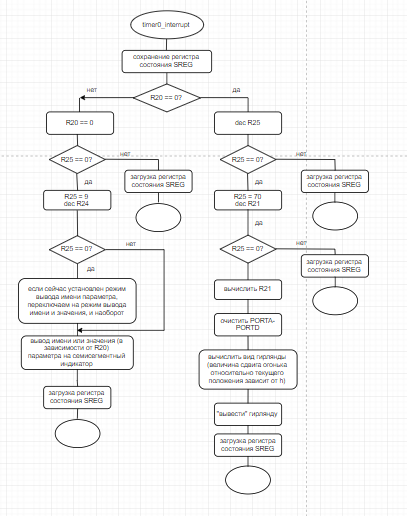
Яркость светодиодов PD7 и PD4 при помощи значения скважности (широтно-импульсная модуляция, режим fast-PWM таймеров T1 и T2) должна показываться степень отдалённости настраиваемого параметра от крайних значений.

# Изображение выглядит как текст, седзи Автоматически созданное описаниеСхема лабораторной установки

# Блок-схема алгоритма работы программы







# Временные диаграммы логических сигналов на портах МК (фрагмент)

В момент времени 7 с было произведено прерывание int0, перешли к настройке гирлянды, теперь пол секунды на семисигментном индикаторе мигает имя параметра (“h.”), пол секунды – имя и значение параметра (“h. -C”);

в 9,25 с был повернут потенциометр, соответственно изменилось значение параметра h (на индикаторе стало высвечиваться “h. -1”);

в 10,75 с было произведено прерывание int1, перешли к выбору значения параметра p (на индикаторе высвечивается “р. 7”);

в 12,25 с был повернут потенциометр, соответственно изменилось значение параметра р (на индикаторе стало высвечиваться “р. 4”);

в 14,25 с было произведено прерывание int0, перешли к «выводу» гирлянды с новыми параметрами.

# Алгоритм выполнения задействованных команд (конструкций) ассемблера

sbrc reg, i – Пропуск следующей команды, если бит i в регистре reg очищен

sbic PORT, i - Пропуск следующей команды, если бит i в порту очищен

brlo delay - Переход если меньше

brlt delay - Переход если меньше, для чисел со знаком

muls reg1, reg2 - Умножение чисел со знаком

lsl reg - Логический сдвиг влево

cbr reg, 0b00000100 – Очистка 2ого бита регистра reg

sbr reg, 0b00000100 – Установка 2ого бита регистра reg

# Результаты работы

Написали программу работы гирлянды с возможностью настройки скорости переключения мигающего огонька и шага (в том числе направление сдвига огонька). Переключение огонька происходит по прерываниям от таймера, работающего в обычном режиме. Смена режимов «вывод только имени параметра» и «вывод имени и значения параметра» в режиме настройки, а также переключение между индикаторами для вывода разной информации происходит по тем же прерываниям. Считывание значения аналогового сигнала производится с помощью прерывания АЦП (ADC). Вывода на светодиоды PD4 и PD7 производится сразу после считывания значения из АЦП.

# Ответы на контрольные вопросы

1. В каких режимах может работать АЦП?
2. Одиночное преобразование: АЦП сработает единожды при инициализации АЦП (при записи бита ADSC в регистр ADCSRA).
3. Free Running mode («свободный полет»): значения регистров ADCH и ADCL обновляются постоянно. АЦП выполняется последовательно (новое считывание сигнала начинается сразу после окончания обработки предыдущего считывания) независимо от флагов прерываний.
4. Auto Triggering: обработка входного сигнала производится через фиксированные промежутки времени.
5. Какого назначение бита ADATE регистра ADCSRA?

Если ADATE установлен, допускаются преобразования по прерыванию от периферийных устройств микроконтроллера. Иначе не допускаются.

1. Поясните назначение предделителя частоты в модуле АЦП МК ATmega32.

Предделитель корректирует частоту. Чем выше предделитель, тем ниже частота, ниже скорость обработки результата, но выше точность считанного значения.

1. В чём заключаются отличия режима Fast PWM от Phase correct PWM?

Fast PWM mode – режим с самой высокой частотой среди всех режимов работы ШИМ. Отличается от других режимов односкатностью операции. Всегда проходит все значения от BOTTON до TOP, затем начинает считать снова с BOTTOM. В неинвертирующем режиме бит OC1x устанавливается в BOTTOM и очищается в точке сравнения OCR1x (соответственно горит и светодиод). В инвертирующем режиме устанавливается в точке сравнения и очищается в BOTTOM.

Рhase correct PWM mode – режим с высокой разрешающей способностью и меньшей частотой. Базируется на двускатных операциях. Счетчик считает от BOTTOM до ТОР и обратно. В неинвертирующем режиме бит OC1x очищается при достижении OCR1x, пока счетчик считает «вверх», и устанавливается при достижении OCR1x при счете «вниз». В инвертирующем режиме OC1x устанавливается при достижении OCR1x при свете «вверх», и очищается при достижении OCR1x при счете «вниз».

1. Перечислите все возможности по настройке ШИМ для 16-разрядного таймера-счётчика.

Настройка идет с помощью битов регистров TCCR1A, TCCR1B, TIMSK путем установки определенных битов в 1. В этих регистрах содержатся биты определения режима вывода при сравнении, управления направлением счета, управления настройкой режима (Fast PWM Mode, Phase Correct PWM Mode, Phase and Frequency Correct PWM Mode), определения режима генерации сигнала, определения источника тактирования.

# Выводы по лабораторной работе

В ходе данной лабораторной работы изучили структуры данных, пересылаемых через последовательный порт, ознакомились с принципом работы АЦП в микроконтроллере, приобрели навыки программирования встроенного в микроконтроллер UART модуля. Реализовали предложенную преподавателем программу, в которой смоделировали работающую гирлянду с переменными скоростью и шагом мигания.

# Приложение 1 Комментированный листинг программы для МК на языке ассемблера

.org 0x0000

RJMP init

.org 0x0002 ;ининциализация прерывания INT0

RJMP INT\_0

.org 0x0004 ;ининциализация прерывания INT1

RJMP INT\_1

.org 0x0016

RJMP timer0\_interrupt

.org 0x020

RJMP INT\_ADC

.org 0x0030 ;сдвигаем весь код, чтоб сторонние прерывания не мешали

init:

cli

;Настраиваем порты A, B, C на выход

ldi R16, 0b11011111 ;PA5 - на вход

out DDRA, R16

ser R16 ;установить все биты регистра

out DDRB, R16

out DDRC, R16

ldi R16, 0b11110011 ;PD2, PD3 - на вход

out DDRD, R16

;УСТАНОВКА СТЕКА

ldi R16,low(RAMEND) ;скопируем в R16 младщий байт из константы RAMEND, которая хранит размер SRAM

out SPL,R16 ;скопируем значение из R16 в SPL

ldi R17,high(RAMEND) ;скопируем в R17 старший байт из константы RAMEND

out SPH,R17 ;скопируем значение из R17 в SPH

;Определяем начальные значения параметров

ldi R17, 1 ;вывод гирлянды

ldi R18, 1 ;частота р

ldi R19, 1 ;сдвиг h

ldi R20, 1 ;режим (1 - гирлянда, 0 - настройка)

;вывод названия параметра (х0 - вывод h, х1 - вывод р)

;вывод значения параметра (хх0 - вывод и названия, и значения, хх1 - вывод только названия)

ldi R21, 60 ;m = 8\*(15-p)

ldi R23, 1 ;номер порта, на котором сейчас горит огонек гирлянды

ldi R24, 200 ;счетчик для таймера на 0,5 секунд

ldi R25, 70

;Разрешение прерываний

ldi R16, 0b00001111 ;биты 0 и 1 отвечают за прерывание INT0, биты 2 и 3 - за INT1 (по нарастающему фронту; по любому изменению - 00000101)

out MCUCR, R16

ldi R16, 0b11000000 ;бит 0 отвечает за прерывание INT0, 1 - за INT1

out GICR, R16

;Работа с таймерами

ldi R16, 0b01101001 ;неинвертирующий режим. 0bхх11хххх - инвертирующий

out TCCR2, R16

ldi R16, 0b00001001

out TCCR1B, R16

ldi R16, 0b00110001

out TCCR1A, R16

ldi R16, 0b00000010

out TCCR0, R16 ;инициализация таймера 0; normal = x0xx0xxx; СТС = x0xx1xxx; последние 3 - коэффициент предделителя

ldi R16, 0b00000001

out TIMSK, R16

;инициализация АЦП

ldi R16, 0b00100101

out ADMUX, R16 ;изменение порядка следования битов на лево-ориентированное (5ый бит); xxx00101 - AD5

ldi R16, 0b11101101 ;101 предделитель - частота мигания

out ADCSRA, R16

rcall fill\_r21

ldi R17, 1

out PORTA, R17

sei

loop\_1:

rjmp loop\_1

; ВСЕ ПРЕРЫВАНИЯ

timer0\_interrupt:

ldi R26, 1

cpse R20, R26

rjmp set\_decision ;переключать ли настройки?

rjmp lights\_decision ;переключать ли гирлянду?

INT\_0:

in R2, SREG

ldi R26, 1

cpse R20, R26

rjmp lights

rjmp settings

INT\_1:

in R2, SREG

sbrc R20, 0

rjmp out\_of\_interrupt

sbrc R20, 1

rjmp change\_to\_h ;Если сейчас идет вывод h, то надо выводить р

ldi R20, 0b00000010

out SREG, R2

reti

INT\_ADC:

in R2, SREG

sbrc R20, 0 ;Если нулевой бит очищен, пропустить следующую команду. То есть в режиме гирлянды мы игнорируем прерывание.

rjmp out\_of\_interrupt

in R26, ADCH ;Считать значение из АЦП

out OCR2, R26

out OCR1BL, R26

sbrc R20, 1 ;Если первый бит очищен, пропустить следующую команду. То есть в режиме настройки h мы игнорируем переход в change\_p\_value.

rjmp change\_p\_value ;Определяем, какую именно переменную надо корректировать.

rjmp change\_h\_value

; ВСЯ РАБОТА С ГИРЛЯНДОЙ: РАСЧЕТ СДВИГА, РАСЧЕТ ПОРТА ВЫВОДА, РАСЧЕТ ПОДОБНЫХ ПАРАМЕТРОВ

lights\_decision:

dec R25

cpi R25, 0

brne out\_of\_interrupt

ldi R25, 70

dec R21

cpi R21, 0

breq change\_lights

out SREG, R2

reti

out\_of\_interrupt:

out SREG, R2

reti

change\_lights:

rcall fill\_r21

cpi R19, 0

breq out\_of\_interrupt

ldi R16, 0

out PORTA, R16

out PORTB, R16

out PORTC, R16

mov R16, R19

rcall absolute\_value\_r16

circle:

rcall shift\_lights

dec R16

brne circle

rjmp print\_lights

fill\_r21:

mov R16, R18 ;m = 8\*(16-р): раз в m прерываний от таймера 2 будет случаться переключение гирлянды

ldi R22, 16

sub R22, R16

ldi R16, 8

mul R16, R22

mov R21, R0

ret

absolute\_value\_r16:

sbrs R16, 7 ;пропуск если бит установлен. Если 7ой бит установлен, значит, R16 < 0, значит, не надо возвращаться (следующая команда не выполняется)

ret

ldi R22, -1

muls R16, R22

clr R1

mov R16, R0

clr R0

ret

shift\_lights:

cpi R19, 0

brlt shift\_lights\_negative ;если h меньше 0

lsl R17

cpi R17, 0

breq change\_print\_port\_positive ;если досдвигали до обнуления, перескакиваем на следующий регистр

ret

shift\_lights\_negative:

lsr R17

cpi R17, 0

breq change\_print\_port\_negative

ret

change\_print\_port\_negative:

dec R23

ldi R17, 0b10000000

ldi R26, 0

cpse R23, R26

ret

ldi R23, 3

ret

change\_print\_port\_positive:

inc R23

ldi R17, 1

ldi R26, 4

cpse R26, R23

ret

ldi R23, 1

ret

lights:

rcall fill\_r21

ldi R16, 0

out PORTA, R16

out PORTB, R16

out PORTC, R16

ldi R20, 1

rjmp print\_lights

; ВСЕ ПРО РЕЖИМ НАСТРОЙКИ

set\_decision:

dec R25

ldi R26, 0

cpse R25, R26

rjmp out\_of\_interrupt

ldi R25, 9

dec R24

brne set\_continue

ldi R24, 200 ;Если счетчик обнулился, возвращаем ему значение 200.

ldi R26, 4

sbrc R20, 2 ;Смотрим, какой режим установлен, и меняем его.

ldi R26, -4

add R20, R26

set\_continue:

sbrc R20, 2 ;Сморим, какой режим установлен.

rjmp out\_name ;Если установлен режим "выводить всё", то пропускаем эту строку. Иначе - сразу переходим к выводу названия.

mov R16, R19 ;Запомним значение параметра h.

sbrc R20, 1 ;Какой параметр надо выводить?

mov R16, R18 ;Если надо выводить h, то пропускаем эту команду. Иначе - запоминаем значение параметра p.

sbic PORTA, 3 ;Если выведено имя переменной,

rjmp out\_sign ;то переходим к выводу знака (минус или ничего).

sbic PORTA, 1 ;Если выведен знак,

rjmp out\_value ;то переходим к выводу значения параметра.

rjmp out\_name ;Иначе переходим к выводу имени переменной.

out\_name:

sbrc R20, 1 ;Какой параметр надо выводить?

rjmp out\_p ;Если надо выводить h, то пропускаем эту команду

rjmp out\_h

out\_value: ;вывод значения текущей переменной

ldi R16, 0

out PORTC, R16

ldi R16, 0b00000001

out PORTA, R16

mov R16, R19 ;Запомним значение параметра h.

sbrc R20, 1 ;Какой параметр надо выводить?

mov R16, R18 ;Если надо выводить h, то пропускаем эту команду. Иначе - запоминаем значение параметра p.

ldi R26, 0

cpse R16, R26

rcall absolute\_value\_r16 ;взять по модулю

rjmp print\_value

out\_sign: ;вывод знака

cpi R16, 0 ;Если значение параметра < 0,

brlt print\_minus;выводим минус.

rjmp out\_value ;Иначе переходим к выводу значения.

out\_p: ;вывод названия p

ldi R16, 0b00001000

out PORTA, R16

ldi R16, 0b11110011

out PORTC, R16

out SREG, R2

reti

out\_h: ;вывод названия h

ldi R16, 0b00001000

out PORTA, R16

ldi R16, 0b11110100

out PORTC, R16

out SREG, R2

reti

settings:

ldi R16, 0

out PORTA, R16

out PORTB, R16

out PORTC, R16

ldi R20, 0

ldi R16, 0b00001000

out PORTA, R16

rjmp out\_h

change\_h\_value:

ldi R19, -12

change\_h\_loop:

cpi R26, 10

brlo change\_h\_continue

inc R19

subi R26, 10

rjmp change\_h\_loop

change\_h\_continue:

cpi R19, 13

breq dec\_r19

out SREG, R2

reti

dec\_r19:

dec R19

out SREG, R2

reti

change\_p\_value:

ldi R18, 1

change\_p\_loop:

cpi R26, 16

brlo change\_p\_continue

inc R18

subi R26, 16

rjmp change\_p\_loop

change\_p\_continue:

cpi R18, 16

breq dec\_r18

out SREG, R2

reti

dec\_r18:

dec R18

rcall fill\_r21

out SREG, R2

reti

change\_to\_h:

ldi R20, 0

out SREG, R2

reti

; ВСЕ, ЧТО КАСАЕТС ВЫВОДА НА ИНДИКАТОРЫ И СВЕТОДИОДЫ

print\_minus:

ldi R16, 0b00000010

out PORTA, R16

ldi R16, 0b01000000

out PORTC, R16

out SREG, R2

reti

print\_value:

cpi R16, 1

brlt print\_0

breq print\_1

cpi R16, 2

breq print\_2

cpi R16, 3

breq print\_3

cpi R16, 4

breq print\_4

cpi R16, 5

breq print\_5

cpi R16, 6

breq print\_6

cpi R16, 7

breq print\_7

cpi R16, 8

breq print\_8

cpi R16, 9

breq print\_9

cpi R16, 0xA

breq print\_A

cpi R16, 0xB

breq print\_B

cpi R16, 0xC

breq print\_C

cpi R16, 0xD

breq print\_D

cpi R16, 0xE

breq print\_E

brne print\_F

print\_0:

ldi R16, 0b00111111

rjmp print\_on\_port

print\_1:

ldi R16, 0b00000110

rjmp print\_on\_port

print\_2:

ldi R16, 0b01011011

rjmp print\_on\_port

print\_3:

ldi R16, 0b01001111

rjmp print\_on\_port

print\_4:

ldi R16, 0b01100110

rjmp print\_on\_port

print\_5:

ldi R16, 0b01101101

rjmp print\_on\_port

print\_6:

ldi R16, 0b01111101

rjmp print\_on\_port

print\_7:

ldi R16, 0b00000111

rjmp print\_on\_port

print\_8:

ldi R16, 0b01111111

rjmp print\_on\_port

print\_9:

ldi R16, 0b01101111

rjmp print\_on\_port

print\_A:

ldi R16, 0b01110111

rjmp print\_on\_port

print\_B:

ldi R16, 0b01111100

rjmp print\_on\_port

print\_C:

ldi R16, 0b00111001

rjmp print\_on\_port

print\_D:

ldi R16, 0b01011110

rjmp print\_on\_port

print\_E:

ldi R16, 0b01111001

rjmp print\_on\_port

print\_F:

ldi R16, 0b01110001

rjmp print\_on\_port

print\_on\_port:

out PORTC, R16

out SREG, R2

reti

print\_lights:

cpi R23, 1

breq print\_lights\_a

cpi R23, 2

breq print\_lights\_b

out PORTC, R17

out SREG, R2

reti

print\_lights\_a:

out PORTA, R17

out SREG, R2

reti

print\_lights\_b:

out PORTB, R17

out SREG, R2

reti