Министерство образования и науки Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого

—

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

Отчёт

по лабораторной работе №2

**ОРГАНИЗАЦИЯ АНАЛОГОВОГО ВВОДА-ВЫВОДА: АЦП И ШИМ**

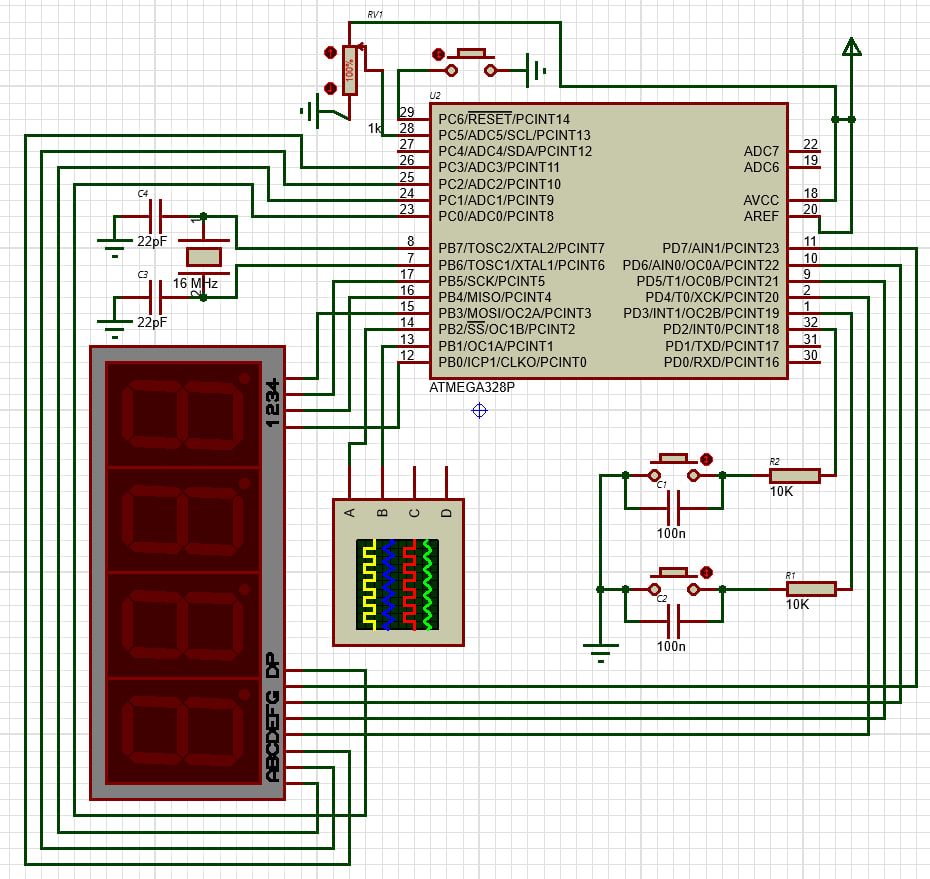
по дисциплине «Аппаратные средства вычислительной техники»

|  |  |  |  |
| --- | --- | --- | --- |
| Выполнили: | студенты группы 4831001/10003 |  | Г. А. Улановский |
|  |  | (подпись, дата) |  |
|  |  |  | Г. Г. Фидаров |
|  |  | (подпись, дата) |  |
|  |  |  |  |
|  |  |  |  |
| Проверил: | доцент, к.т.н. |  | П. О. Семенов |
|  |  | (подпись, дата) |  |

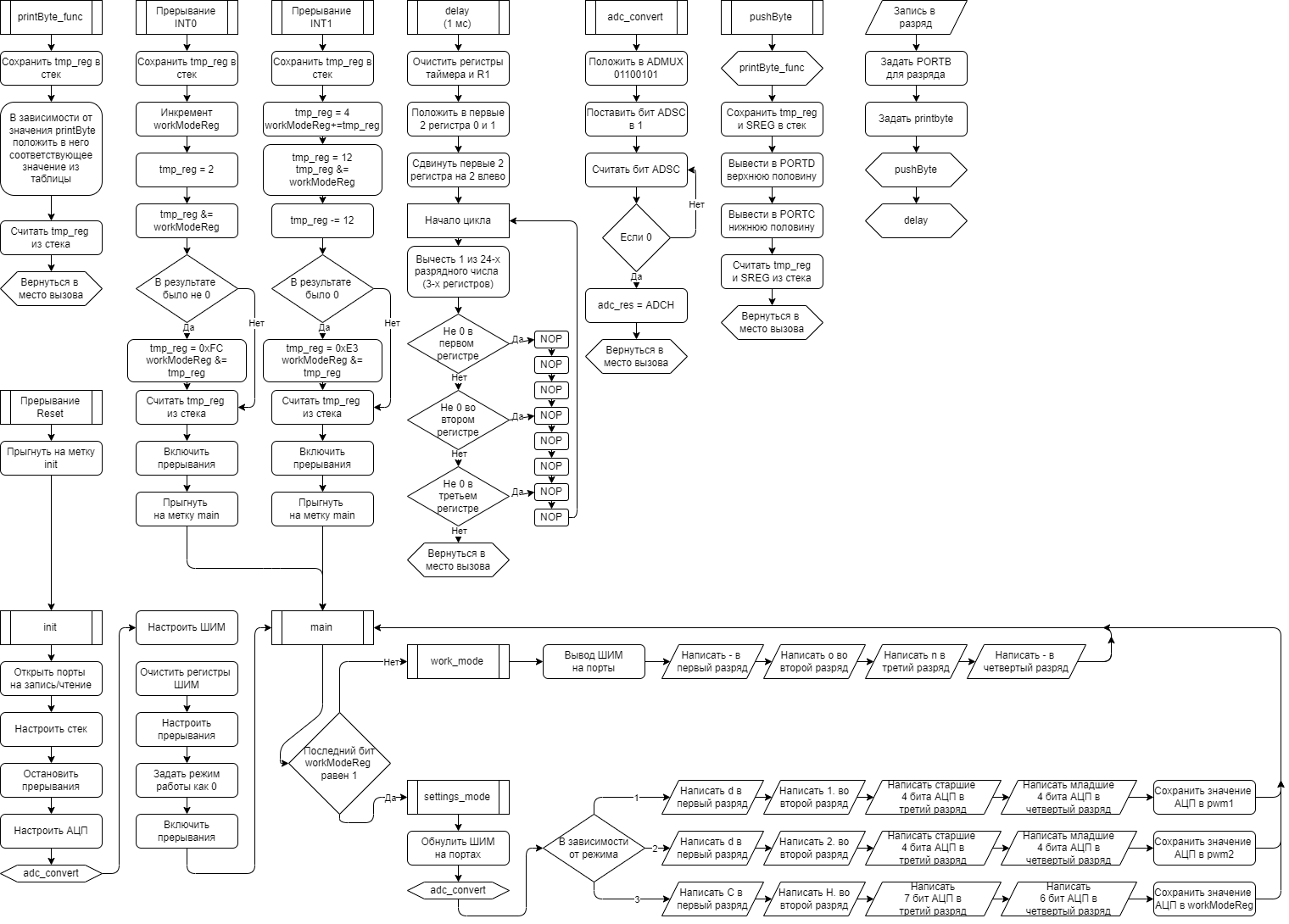
# Формулировка задания

1. Программа должна осуществлять два режимы работы:
   1. демонстрация работы;
   2. настройка параметров работы.
2. Вывод PA5 должен быть настроен на ввод (DDRA=0xdf), для работы он игнорируется (изменения в логику работы не вносятся). Считывание значения аналогового сигнала должно производиться с помощью прерывания АЦП (ADC).
3. 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 первых (одном, двух или трёх, в зависимости от параметра) семисегментных индикаторах, на последующих индикаторах должно отображаться значение соответствующего параметра в шестнадцатеричной системе счисления. На последнем индикаторе имени параметра в качестве разделителя должна гореть точка.
4. Изменение значений должно осуществляться с помощью потенциометра, подключённого к выводу PA5, следующим образом:
   1. крайнее левое положение соответствует нижней границе допустимого диапазона, крайнее правое – верхней;
   2. при повороте потенциометра значение на семисегментном индикаторе изменяется незамедлительно.

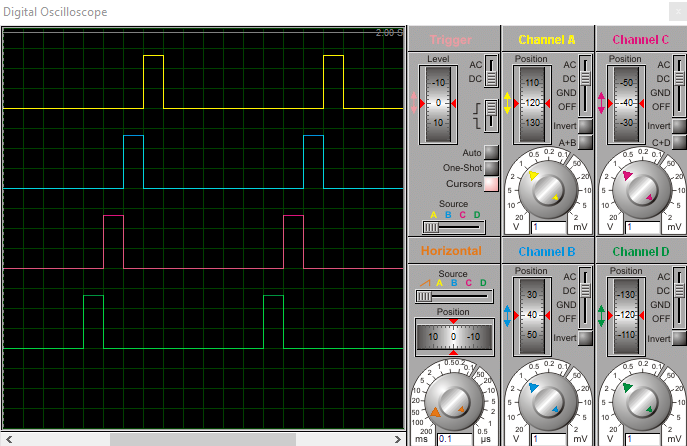
# Схема лабораторной установки



# Блок-схема алгоритма работы программы



# Временные диаграммы логических сигналов на портах МК (фрагмент)



+ см pdf с примером оформления

# Результаты работы

### Определение зависимости количества тактов от констант x и y

СМ .ру файл

### Анализ lst-файла

СМ lst-файл и первый лист .xlsx

### Анализ hex-файла

CM hex-файл

### Алгоритм выполнения команды ассемблера

Вариант 3а: SUB R21, R1

Этапы выполнения команд:

1. Передача памяти программ (ПЗУП) адреса текущей команды

2. Извлечение команды из ПЗУП и запись в регистр команды

3. Декодирование команды – определение типа операции и формата

Код операции: 0001 10rd dddd rrrr

4. Извлечение адресов операндов (номеров регистров общего

назначения (РОН)) -> d.dddd, r.rrrr

5. Извлечение операндов из команды (непосредственная адресация)

6. Передача операндов в сумматор

7. Выполнение операции вычитания

8. Передача результата операции вычитания в регистр Rd

9. Увеличение значения счётчика команд

# Ответы на контрольные вопросы

1. Посредством каких регистров производится настройка АЦП?
   1. ADMUX – регистр настройки мультиплексора; В программе установлено на 0x65, т.е. 0b011\*0101.
      1. 01 - Внешний источник питания на AVCC, с внешним конденсатором на AREF
      2. 1 – Обратный порядок байт в регистрах ADCH и ADCL
      3. 0101 – выбор ADC5 (A5 пин) на чтение
   2. ADCSRA – управляющий и статусный регистр; В программе установлено на 0x83, т.е. 0b10000011. В подпрограмме считывания АЦП происходит установка второго бита и ожидание его снятия
      1. 1 – включение/выключение АЦП.
      2. 0 – запуск преобразования АЦП. В режиме одиночного преобразования нужно записать единицу в этот бит, чтобы начать преобразование. В режиме свободного запуска нужно записать единицу в этот бит, чтобы начать первое преобразование.
      3. 0 – включение автоматического запуска АЦП. Когда этот бит записывается в единицу, включается автоматическая работа АЦП, то есть значения будут считываться постоянно.
      4. 0 – флаг прерывания АЦП. Этот бит устанавливается, когда преобразование АЦП завершается и регистры данных обновляются.
      5. 0 – активация прерывания АЦП.
      6. 011 – выбор предделителя АЦП, в данном случае – 8
   3. ADCSRB – управляющий регистр; В данном регистре настраивается автоматическое считывание АЦП. Например, от прерываний таймеров или внешнего INTR0
   4. ADCH и ADCL – регистры, в которых записывается значение АЦП.
2. В каких режимах может работать АЦП?
   1. Режим одиночного преобразования – в этом режиме АЦП может преобразовывать аналоговый сигнал с одного входа, после чего прекращает работу.
   2. Режим автозапуска – в этом режиме АЦП постоянно преобразует сигналы с заданного входа и выводит результаты в соответствующий регистр.
   3. Режим захвата – в этом режиме АЦП используется для захвата сигнала, если он превышает или находится ниже определенного уровня. Результаты преобразования выводятся в регистры ADCH и ADCL.
3. Какие порты и разряды портов микроконтроллера ATmega328p могут обрабатывать входящие аналоговые сигналы?
   1. По datasheet на микроконтроллер ATmega328P, входы для аналоговых сигналов находятся на портах С (ADC0-ADC5). Кроме того, порты С могут использоваться для работы с внешними прерываниями, включая возможность настройки на работу с изменениями уровня аналоговых сигналов.
   2. Компаратор доступен только на пинах PD6 (AIN0) и PD7 (AIN1).
4. Какими способами реализуется ШИМ?
   1. Аппаратный ШИМ (PWM):

ATmega328P имеет три 8-битных таймера/счетчика (Timer0, Timer1, Timer2), которые могут генерировать широтно-импульсную модуляцию (PWM).

Timer0 может быть использован для генерации ШИМ на пине OC0A (PD6) и OC0B (PD5), Timer2 - на пине OC2A (PB3) и OC2B (PD3), а Timer1 - на пинах OC1A (PB1) и OC1B (PB2). Для генерации ШИМ на каждом из этих пинов есть специальные регистры TCCR0A, TCCR2A и TCCR1A.

* 1. Фазово-корректирующий режим (PWM Phase Correct):

ATmega328P также поддерживает фазово-корректирующий режим ШИМ, который может быть использован для более точного управления сервоприводами и другими устройствами. В этом режиме ШИМ может быть генерируем на пинах OC1A и OC1B (Timer1), и на пинах OC2A и OC2B (Timer2). Для настройки этого режима есть специальные регистры TCCR1A и TCCR2A.

* 1. ШИМ на программном уровне (Software PWM):

Также возможна генерация ШИМ на программном уровне при помощи использования обычных digital-пинов с помощью прерываний или искусственных задержек.

1. Как настроить ШИМ с помощью таймера-счётчика?

Настройка аппаратного ШИМ на ATmega328P с помощью таймера-счетчика производится путем настройки соответствующих регистров TCCRnA, TCCRnB, а также регистров сравнения OCRnA, OCRnB (где n – номер таймера)

# Выводы по лабораторной работе

# Приложение 1 Комментированный листинг программы для МК на языке ассемблера

.device atmega328p ; Необходимо для компилятора

.def TMP = R20 ; Установка R20 как регистра для временного хранения

.org $000

**JMP** reset ; Указатель на начало программы

; Функция паузы

delay:

LDI R29, 20 ; Установка регистра R29 (x) в значение 20

LDI R30, 250 ; Установка регистра R30 (y) в значение 250

delay\_sub:

**INC** R29 ; Инкрементируем x

**NOP** ; Много nop

**NOP**

**NOP**

**NOP**

**NOP**

**NOP**

**NOP**

**NOP**

**NOP**

**NOP**

**NOP**

**NOP**

**NOP**

**NOP**

**NOP**

**NOP**

**NOP**

**NOP**

**NOP**

**NOP**

**NOP**

**NOP**

BRNE delay\_sub ; Если x стал 0, пропускаем. Иначе переход

**NOP**

**DEC** R30 ; �"екриментируем y

BRNE delay\_sub ; Если y стал 0, пропускаем. Иначе переход

**RET** ; �'озврат по адресу возврата в основной цикл

; Начальная настройка

reset:

; настройка исходных значений

LDI TMP, 0x01 ; Положили 1 в temp

**MOV** R0, TMP ; Положили temp в R0

CLR TMP ; Очистили temp

; настройка портов ввода-вывода

SER TMP ; Положили 0xFF в temp (-//- LDI 0xFF)

**OUT** DDRD, TMP ; Открыли все восемь пинов порта D на запись

; Установка вершины стека в конец ОЗУ

LDI TMP, HIGH(RAMEND) ; Старшие разряды адреса

**OUT** SPH, TMP ; Установили старшие разряды ESP

LDI TMP, LOW(RAMEND) ; Младшие разряды адреса

**OUT** SPL, TMP ; Установили младшие разряды ESP

; Основной цикл

loop:

; Циклический сдвиг 8-разрядного числа R0

**ROR** R0 ; циклический сдвиг вправо

; Вывод 8-разрядного числа R0 на порт PORTD

**OUT** PORTD, R0

; Пауза

**CALL** delay;

; Возврат в начало основного цикла

RJMP **loop** ;

# Приложение 2 Фрагмент lst-файла

gavrasm Gerd's AVR assembler version 5.4 (C)2022 by DG4FAC

----------------------------------------------------------

Source file: main2.asm

Hex file: main2.hex

Eeprom file: main2.eep

Compiled: 08.04.2023, 17:10:41

Pass: 2

1: .device atmega328p ;

2:

3: .def TMP = R20 ;

4:

5: .org $000

6: 000000 940C **JMP** reset ;

000001 0020

7:

8: ;

9: delay:

10: 000002 E1D4 LDI R29, 20 ;

11: 000003 EFEA LDI R30, 250 ;

12:

13: delay\_sub:

14: 000004 95D3 **INC** R29 ;

15: 000005 0000 **NOP** ;

16: 000006 0000 **NOP**

17: 000007 0000 **NOP**

18: 000008 0000 **NOP**

19: 000009 0000 **NOP**

20: 00000A 0000 **NOP**

21: 00000B 0000 **NOP**

22: 00000C 0000 **NOP**

23: 00000D 0000 **NOP**

24: 00000E 0000 **NOP**

25: 00000F 0000 **NOP**

26: 000010 0000 **NOP**

27: 000011 0000 **NOP**

28: 000012 0000 **NOP**

29: 000013 0000 **NOP**

30: 000014 0000 **NOP**

31: 000015 0000 **NOP**

32: 000016 0000 **NOP**

33: 000017 0000 **NOP**

34: 000018 0000 **NOP**

35: 000019 0000 **NOP**

36: 00001A 0000 **NOP**

37: 00001B F741 BRNE delay\_sub ;

38: 00001C 0000 **NOP**

39: 00001D 95EA **DEC** R30 ;

40: 00001E F729 BRNE delay\_sub ;

41: 00001F 9508 **RET** ;

42:

43: ;

44: reset:

45: ;

46: 000020 E041 LDI TMP, 0x01 ;

47: 000021 2E04 **MOV** R0, TMP ;

48: 000022 2744 CLR TMP ;

49: ;

50: 000023 EF4F SER TMP ;

51: 000024 B94A **OUT** DDRD, TMP ;

52: ;

53: 000025 E048 LDI TMP, HIGH(RAMEND) ;

54: 000026 BF4E **OUT** SPH, TMP ;

55: 000027 EF4F LDI TMP, LOW(RAMEND) ;

56: 000028 BF4D **OUT** SPL, TMP ;

57:

58: ;

59: **loop**:

60: ;

61: 000029 9407 **ROR** R0 ;

62: ;

63: 00002A B80B **OUT** PORTD, R0

64: ;

65: 00002B 940E **CALL** delay;

00002C 0002

66: ;

67: 00002D CFFB RJMP **loop** ;

Program : 46 words.

Constants : 0 words.

Total program memory: 46 words.

Eeprom space : 0 bytes.

Data segment : 0 bytes.

Compilation completed, no errors.

Compilation ended 08.04.2023, 17:10:41

# Приложение 3 hex-файл

:020000020000FC

:100000000C942000D4E1EAEFD3950000000000003A

:1000100000000000000000000000000000000000E0

:1000200000000000000000000000000000000000D0

:1000300000000000000041F70000EA9529F708954C

:1000400041E0042E44274FEF4AB948E04EBF4FEF3E

:0C0050004DBF07940BB80E940200FBCFCC

:00000001FF