Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический университет имени П. О. Сухого»

Кафедра «Автоматизированный электропривод»

В. А. Савельев

МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА В АВТОМАТИЗИРОВАННОМ ЭЛЕКТРОПРИВОДЕ

ПРАКТИКУМ
по выполнению лабораторных работ по одноименной дисциплине для студентов специальности 1-53 01 05 «Автоматизированные электроприводы» дневной формы обучения

Гомель 2020

УДК 62-83-52:004.315(075.8)
ББК 31.291+32.844.150.2я73
С12

Рекомендовано научно-методическим советом
факультета автоматизированных и информационных систем
ГГТУ им. П. О. Сухого
(*протокол № 9 от 03.05.2019 г.*)

Рецензент: доц. каф. «Физика и электротехника» ГГТУ им. П. О. Сухого,
канд. техн. наук, доц. *А. В. Козлов*

Савельев, В. А.

С12 Микропроцессорные средства в автоматизированном электроприводе : практикум по выполнению лаборатор. работ по одноим. дисциплине для студентов специальности 1-53 01 05 «Автоматизированные электроприводы» днев. формы обучения / В. А. Савельев. – Гомель : ГГТУ им. П. О. Сухого, 2020. – 26 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: https://elib.gstu.by. – Загл. с титул. экрана.

Содержит краткие теоретические сведения, задания и программы лабораторных работ дисциплины «Микропроцессорные средства в автоматизированном электроприводе».

Для студентов специальности 1-53 01 05 «Автоматизированные электроприводы» дневной формы обучения.

УДК 62-83-52:004.315(075.8)
ББК 31.291+32.844.150.2я73

МЕРЫ БЕЗОПАСНОСТИ

До начала лабораторных работ студенты обязаны ознакомиться с организацией их выполнения, правилами техники безопасности в лаборатории и пройти инструктаж по технике безопасности с распиской об этом в соответствующем журнале.

Работы в лаборатории могут проводиться только с разрешения профессорско-преподавательского состава кафедры.

При выполнении студентами лабораторной работы на рабочем месте могут находиться только материалы, необходимые для выполнении данной работы: инструкции, руководства и необходимое оборудование.

К выполнению лабораторных работ с использованием учебных стендов допускаются студенты, ознакомившиеся с конструкцией и принципом действия стенда.

З А П Р Е Щ А Е Т С Я

1. Включать учебный стенд в сеть, не подключив заземление.
2. Работать с учебным стендом при снятых кожухах.
3. Бросать шнур во избежание поломок вилки.
4. Производить смену предохранителей не отсоединив шнур питания от сетевой розетки.
5. Производить подключение к учебному стенду каких-либо посторонних электрических устройств без согласования с преподавателем.

Студенты, виновные в повреждении приборов и аппаратов, несут за это материальную ответственность.

Лабораторная работа № 1
Регулирование скорости вращения двигателя постоянного тока в системе ШИП-Д

1.1. Цель работы

1. Ознакомиться с устройством драйвера ШИП типа L298.
2. Исследовать работу АЦП в составе микроконтроллера Atmega.
3. Исследовать работу 8-битного таймера/счётчика в составе микроконтроллера Atmega.
4. Ознакомиться с процедурой усреднения результатов измерений.

1.2. Краткие сведения из теории

Функциональная схема микросхемы L298N приведена на рис.1.1. Микросхема L298N представляет собой сдвоенный мостовой драйвер, предназначенный для управления ДПТ и шаговыми двигателями. Данная микросхема находит очень широкое применение в роботостроительстве. Одна микросхема L298N способна управлять двумя двигателями и обеспечивает максимальную нагрузку до 2А на каждый двигатель, а если задействовать параллельное включение для одного двигателя, то можно поднять максимальный ток до 4А.

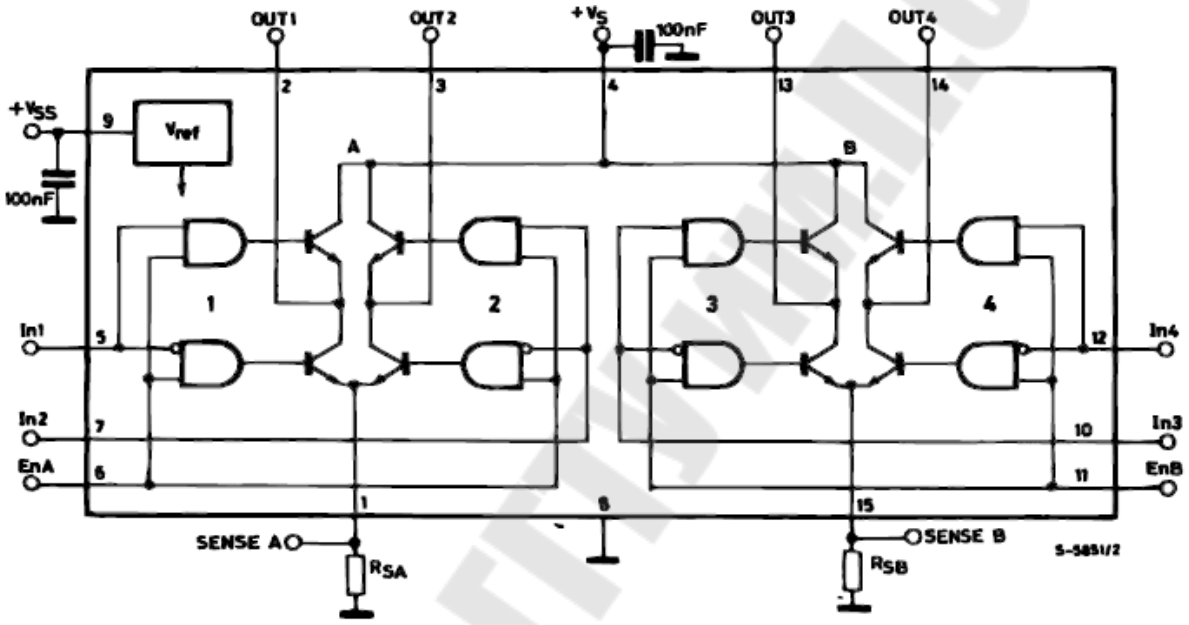


Рис. 1.1. Функциональная схема микросхемы L298N

Микросхема L298N имеет следующие характеристики:

– номинальное напряжение питания до 46 В;

– суммарный постоянный ток до 4 А;

– малое напряжение насыщения;

– защита от перегрева;

– входное напряжение логического нуля до 1,5 В (высокая помехозащищенность).

Функциональное назначение выводов микросхемы драйвера L298N приведено в табл. 1.1, а в табл. 1.2 приведена логика работы драйвера.

Таблица 1.1

Функциональное назначение выводов микросхемы L298N

1	SENS-A	Контроль тока 1 и 2 каналов
2	OUT2	Выход 2 канала
3	OUT3	Выход 3 канала
4	Vs	Питание нагрузки
5	IN1	Вход 1 канала
6	EN-A	Разрешение работы 1 и канала 2
7	IN2	Вход 2 канала
8	GND	Общий вывод
9	Vss	Питание логики (+5В)
10	IN3	Вход 3 канала
11	EN-B	Разрешение работы 3 и 4 канала
12	IN4	Вход 4 канала
13	OUT3	Выход 3 канала
14	OUT4	Выход 4 канала
15	SENS-B	Контроль тока 3 и 4 каналов

Таблица 1.2

Логика работы драйвера L298N

Входы		Выходы
IN	**EN**	**OUT**
H	H	H
L	H	L
X	L	Z
H – высокий уровень сигнала L – низкий уровень сигнала X –любое состояние Z –высокий импенданс (выключено)		

Микросхема объединяет два выходных мощных каскада (A; B). Выходной каскад — это мостовая структура ШИМ. Его выходы могут управлять индуктивной нагрузкой обычным и дифференциальным методом в зависимости от состояния входов.

Эмиттеры нижних транзисторов каждого моста соединены вместе, а соответствующий внешний вывод может использоваться для подключения внешнего измерительного резистора R_{SA} или R_{SB}. Ток,

проходящий через нагрузку, выходит из моста и протекает по внешнему резистору (R_{SA} или R_{SB}), который позволяет определить силу этого тока.

Для включения и выключения устройства независимо от входных сигналов предусмотрены два входа EN-A и EN-B.

Дополнительный вход питания V_{SS} предусмотрен таким образом, что логическая схема работает при малом напряжении (+5В).

1.3. Задание к лабораторной работе

В системе проектирования Proteus создать модель подключения двух двигателей постоянного тока к микроконтроллеру Atmega при помощи драйвера L298. Каждый двигатель должен иметь возможность вращаться независимо от другого. Программа должна считывать значение потенциометра, подключенного к АЦП и на основе полученной информации изменять скорость и направление двигателя.

Исследовать программу управляющую скоростью и направлением вращения двигателя.

В качестве примера рассмотрим программу, позволяющую управлять скоростью двигателя при помощи потенциометра, подключенного ко входу ADC0 контроллера Atmega168 (рис. 1.2).

В начале следует стандартный набор процедур: присоединение файла описания контроллера, присвоение имен регистрам, макрокоманды, определение сегмента памяти для хранения программы.

```
;************************* ЗАДАНИЕ ПАРАМЕТРОВ **************************
.include "m168def.inc"       ; присоединение файла описания ATmega168
.def temp = r16              ; временный регистр
.def rezl = r17              ; младший байт АЦП
.def rezh = r18              ; старший байт АЦП
.def col = r19               ; константа усреднения

; FLASH ===============================================================
.cseg                   ; выбор сегмента памяти для хранения программы
```

В секции векторов прерывания указываем метки перехода для прерываний по сбросу «reset», по срабатыванию компаратора канала А таймера/счётчика Т2 «timer2_int» и по окончанию преобразования АЦП «adc_int».

```
;************************* ВЕКТОРЫ ПРЕРЫВАНИЙ **************************
.org 0
          rjmp reset         ; переход на обработку сброса
.org 2
```

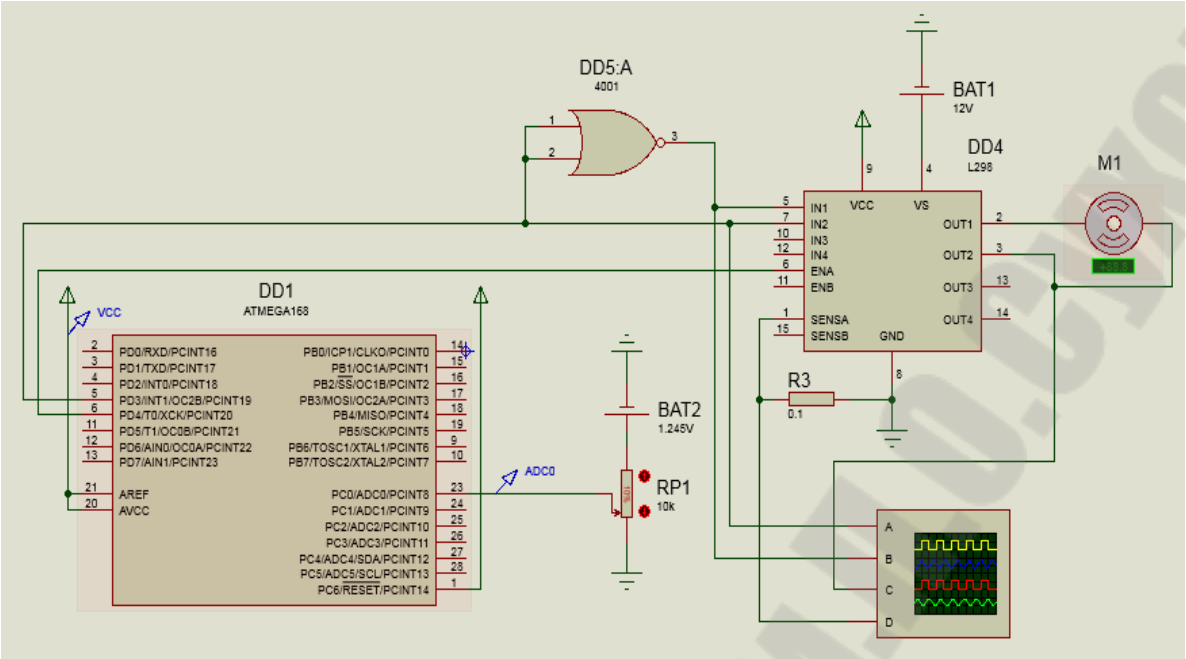


Рис.1.2. Модель привода постоянного тока в Proteus

```
        reti            ; внешнее прерывание IRQ0 (INT0)
.org 4
        reti            ; внешнее прерывание IRQ1 (INT1)
.org 6
        reti            ; прерывание 0 по изменению состояния выводов
.org 8
        reti            ; прерывание 1 по изменению состояния выводов
.org 10
        reti            ; прерывание 2 по изменению состояния выводов
.org 12
        reti            ; тайм-аут сторожевого таймера (WatchDog)
.org 14
        reti            ; срабатывание компаратора A таймера T2
.org 16
        reti            ; срабатывание компаратора B таймера T2
.org 18
        reti            ; переполнение таймера T2
.org 20
        reti            ; захват фронта таймера T1
.org 22
        reti            ; срабатывание компаратора A таймера T1
.org 24
        reti            ; срабатывание компаратора B таймера T1
.org 26
        reti            ; переполнение таймера T1
.org 28
        reti            ; срабатывание компаратора A таймера T0
.org 30
        reti            ; срабатывание компаратора B таймера T0
```

```
.org 32
            reti              ; переполнение таймера T0
.org 34
            reti              ; передача по SPI завершена
.org 36
            reti              ; приём по USART завершен (USART Rx)
.org 38
            reti              ; регистр данных USART пуст
.org 40
            reti              ; передача по USART завершена (USART Tx)
.org 42
            rjmp adc_int      ; окончание преобразования АЦП
.org 44
            reti              ; готовность EEPROM
.org 46
            reti              ; аналоговый компаратор
.org 48
            reti              ; 2-х проводной последовательный интерфейс
.org 50
            reti              ; готовность SPM
;=====================================================================
```

Секция инициализации начинается с процедур обнуления РОН и ОЗУ, затем следует стандартная процедура загрузки указателя стека и настройка портов ввода/вывода. Кроме того, в данном примере, происходит отключение цифрового буфера входа ADC0 (хотя это не обязательно), а также загружается начальный адрес таблицы состояний ШД в индексный регистр Z.

```
;*************************** ИНИЦИАЛИЗАЦИЯ ****************************
reset:
; чистка ОЗУ
      Ldi zl,low(sram_start)        ; адрес начала ОЗУ в индекс
      Ldi zh, high(sram_start)
      clr temp                      ; очищаем temp
clr_ram:
      st z+, temp                   ; пишем 0 в ячейку памяти
      cpi zh, high(ramend+1)        ; достиг ли СБ индекса конца ОЗУ?
      brne clr_ram                  ; продолжаем чистку
      cpi zl,low(ramend+1)          ; достиг ли МБ индекса конца ОЗУ?
      brne clr_ram                  ; продолжаем чистку
      clr zl                        ; чистим индекс
      clr zh
;=====================================================================
; чистка РОН
      Ldi zl, 30
      clr zh
clr_ron:
      dec zl
      st z, zh
```

```
      brne clr_ron
;=====================================================================
; инициализация стека
      ldi temp, high(ramend)
      out sph, temp
      ldi temp, low(ramend)
      out spl, temp
;=====================================================================
; инициализация портов
      usbis didr0, ADC0D      ; отключение цифрового буфера линии ADC0
      ldi temp, 0b00011000    ; установка PD3, PD4 на вывод
      out ddrd, temp
      ldi col, 64             ; константа усреднения
;=====================================================================
```

Секция инициализации продолжается настройкой режима работы таймера/счётчика T2. Таймер настраивается на режим «быстрая ШИМ». Здесь также задается делитель тактовой частоты.

Далее следует инициализация АЦП. Выбираем аналоговый вход ADC0, смещение результата вправо, разрешаем работу АЦП, однократное преобразование с автоматическим перезапуском. Обязательно разрешаем прерывания от АЦП и выбираем делитель, такой, чтобы частота работы АЦП составляла от 50 до 200 кГц.

После всего разрешаем прерывания глобально.

```
;=====================================================================
; инициализация таймера Timer_2
      ldi temp, 0b01111111    ; начальная установка скважности
      sts ocr2b, temp         ; 50%
      ldi temp, 0b00100001

; COM2A1:0  0b[00]xxxxxx -    режим формирования выходного сигнала
;                             OC2A (отключен)
; COM2B1:0  0bxx[10]xxxx -    режим формирования выходного сигнала
;                             OC2B (сброс в 0 при равенстве)
;           0bxxxx[00]xx -    зарезервирован
; WGM21:0   0bxxxxxx[11] -    режим работы таймера

      sts tccr2a, temp
      ldi temp, 0b00000010

; FOC2A     0b[0]xxxxxxx -    принудительное изменение состояния OC2A
; FOC2B     0bx[0]xxxxxx -    принудительное изменение состояния OC2B
;           0bxx[00]xxxx -    зарезервированы
; WGM22     0bxxxx[0]xxx -    режим работы таймера (быстрая ШИМ)
; CS22:0    0bxxxxx[010] -    выбор тактового источника с
;                             предделителем частоты, равным 8 при
;                             тактовой частоте контроллера 8МГц и
;                             8-бит быстрой ШИМ имеем
;                             частоту 8 МГц/8/256 = 3906 Гц
```

```
      sts tccr2b, temp

;=====================================================================
; инициализация АЦП
      ldi temp, 0b00000000

; REFS1:0   0b[00]xxxxxx -    выбор ИОН (напряжение AREF)
; ADLAR     0bxx[0]xxxxx -    смещение 10-битного результата вправо
;           0bxxx[0]xxxx -    зарезервирован
; MUX3:0    0bxxxx[0000] -    выбор аналогового входа (ADC0)

      sts admux, temp
      ldi temp, 0b11101111

; EDEN      0b[1]xxxxxxx -    разрешение работы АЦП
; ADSC      0bx[1]xxxxxx -    однократное преобразование
; ADFR      0bxx[1]xxxxx -    разрешение автоматического запуска АЦП
; ADIF      0bxxx[0]xxxx -    сброс флага прерывания АЦП
; ADIE      0bxxxx[1]xxx -    разрешение прерывания от АЦП
; ADPS2:0   0bxxxxx[110] -    выбор делителя частоты 8 МГц/64=125 Гц

      sei                     ;разрешение прерываний
;=====================================================================
```

Теперь следует секция основной программы (метка «main:»), которая представляет собой пустой цикл.

```
;********************** ОСНОВНАЯ ПРОГРАММА ***************************
main:
      rjmp main
;=====================================================================
```

Основные действия происходят в обработчиках прерываний. Поскольку при пуске с максимальным напряжением возникают броски тока, для их исключения блокируем драйвер до тех пор, пока потенциометр не будет установлен в среднее положение, после этого устанавливается флаг T и разрешается работа драйвера со скважностью 50%.

```
;=====================================================================
; обработка прерывания от АЦП
adc_int:
      lds r20, adcl           ; вывод младшего байта АЦП в
                              ; регистр R20
      lds r21, adch           ; вывод старшего байта АЦП
                              ; в регистр R21
      brbs 6, next
; проверка флага T (если он установлен, переходим по метке)
      cpi r20, 0b01111111
```

```
; сравнение содержимого АЦП с числом 127 (50%)
      brne endi
; если начальное значение не установлено, окончание прерывания
      set
; установка флага T
      sbi portd, 4
; установка линии PD4 в «1» (разрешение работы драйвера)
      ldi temp, 0b01111111
      sts ocr2b, temp
; запись в регистр сравнения числа 127 (50%)

;=====================================================================
; суммируем результат 64-х измерений
next:
      add rezl, r20
      adc rezh, r21
      dec col
      brne endi
; усредняем результат (делим на 64 путем сдвига 6 раз)
      ldi temp, 6                ; число сдвигов 2^6=64
sdvig:
      clc                        ; сброс флага C
      ror rezh                   ; сдвиг вправо через перенос
                                 ; ст.байта
      ror rezl                   ; сдвиг вправо через перенос
                                 ; мл.байта
      dec temp                   ; уменьшаем число сдвигов
      brne sdvig                 ; если сдвиг не завершен,
                                 ; переходим по метке
      ldi col, 64                ; восстанавливаем константу
                                 ; усреднения
      sts ocr2b, rezl
; запись усредненного результата в регистр сравнения OCR2B Timer_2
      clr rezl                   ; чистим регистры результата
      clr rezh
endi:
      reti
```

На рис. 1.3 показаны осциллограммы работы схемы, полученные при симуляции в программе Proteus. На рис. 1.4 показаны осциллограммы полученные при экспериментальном исследовании.

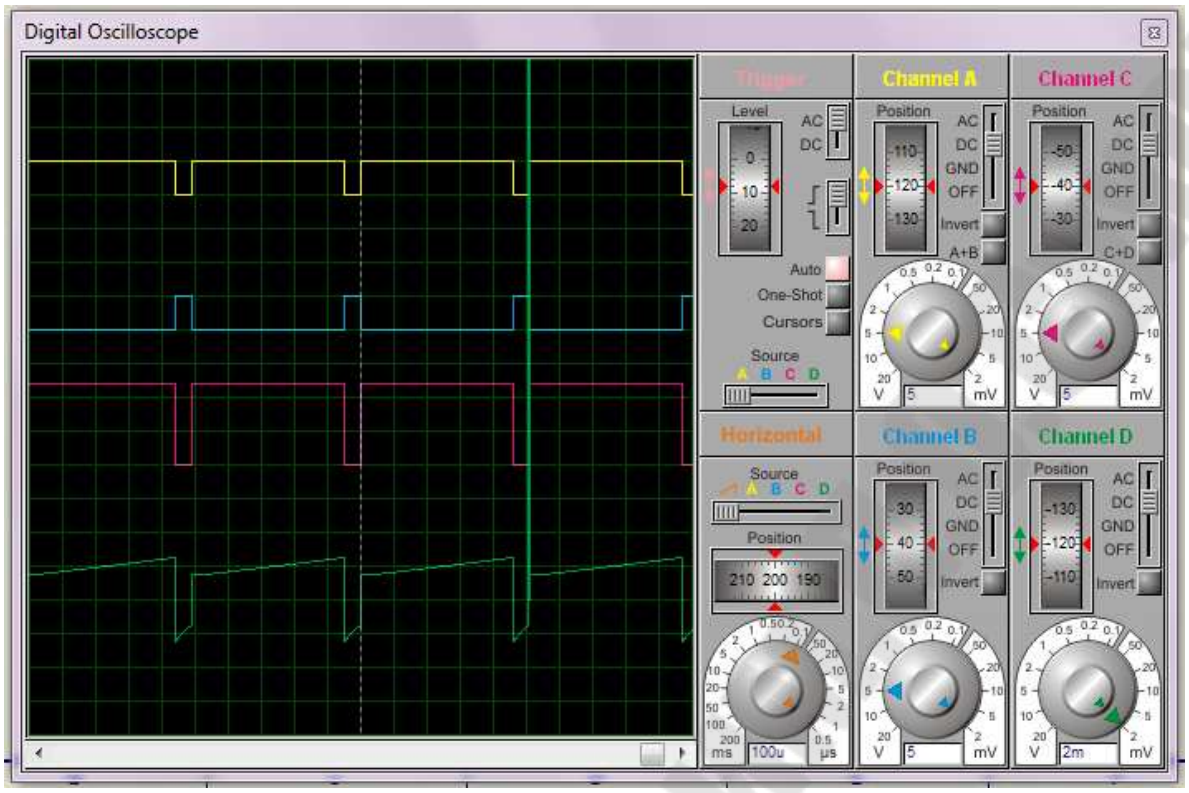


Рис.1.3. Осциллограммы работы электропривода в Proteus

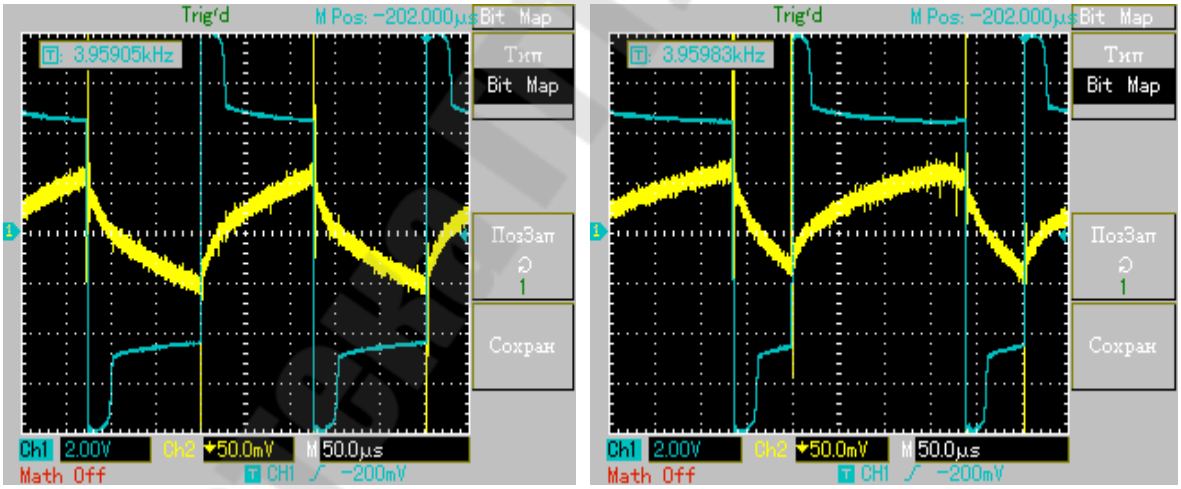


Рис.1.4 Осциллограммы работы электропривода:
желтый – ток якоря, голубой – напряжение на якоре

1.4. Содержание отчета

В отчете привести наименование и цель работы, схему модели подключения двигателей в Proteus, текст программы с комментариями к командам и директивам, осциллограммы работы модели.

Сделать выводы по существу полученных результатов.

1.5. Контрольные вопросы

1. Что представляет собой микросхема драйвера L298?
2. Какова логика работы драйвера L298?
3. Как производится инициализация АЦП в составе микроконтроллера Atmega?
4. Как производится инициализация 8-битного таймера-счётчика в составе микроконтроллера Atmega?
5. В чём состоит алгоритм усреднения полученного с АЦП результата?
6. Прокомментируйте осциллограммы работы схемы, приведенные на рис. 5.3 и 5.4.

Лабораторная работа № 2
Управление шаговым двигателем

2.1. Цель работы

1. Управление шаговым двигателем.
2. Программирование таймера/счётчика.
3. Программирование аналого-цифрового преобразователя.
4. Работа с таблицей данных.

2.2. Краткие сведения из теории

Шаговый двигатель (рис. 2.1) – это электромеханическое устройство, которое преобразует электрические импульсы в дискретные механические перемещения. Внешне он практически ничем не отличается от двигателей других типов. Чаще всего это круглый корпус, вал, несколько выводов.

Шаговые двигатели обладают некоторыми уникальными особенностями, что делает порой их исключительно удобными для применения или даже незаменимыми. Некоторые особенности, характеризующие шаговые двигатели:

– угол поворота ротора определяется числом импульсов, которые поданы на двигатель;

– двигатель обеспечивает полный момент в режиме остановки (если обмотки запитаны);

Рис. 2.1. Внешний вид шагового двигателя

– прецизионное позиционирование и повторяемость; хорошие шаговые двигатели имеют точность 3–5 % от величины шага, причем эта ошибка не накапливается от шага к шагу;

– возможность быстрого старта/остановки/реверсирования;

– высокая надежность, связанная с отсутствием щеток; срок службы шагового двигателя фактически определяется сроком службы подшипников;

– однозначная зависимость положения от входных импульсов обеспечивает позиционирование без обратной связи;

– возможность получения очень низких скоростей вращения для нагрузки, присоединенной непосредственно к валу двигателя без промежуточного редуктора;

– скорость пропорциональна частоте входных импульсов, таким образом, может быть перекрыт довольно большой диапазон скоростей.

В зависимости от конфигурации обмоток двигатели делятся на биполярные и униполярные. Биполярный двигатель (рис. 2.2, *а*) имеет одну обмотку в каждой фазе, которая для изменения направления магнитного поля должна переполюсовываться драйвером. Для такого типа двигателя требуется мостовой драйвер, или полумостовой с двухполярным питанием. Всего биполярный двигатель имеет две обмотки и, соответственно, четыре вывода.

Униполярный двигатель (рис. 2.2, *б*) также имеет одну обмотку в каждой фазе, но от середины обмотки сделан отвод. Это позволяет изменять направление магнитного поля, создаваемого обмоткой, про-

стым переключением половинок обмотки. При этом существенно упрощается схема драйвера. Драйвер должен иметь только 4 простых ключа. Таким образом, в униполярном двигателе используется другой способ изменения направления магнитного поля. Средние выводы обмоток могут быть объединены внутри двигателя, поэтому такой двигатель может иметь 5 или 6 выводов.

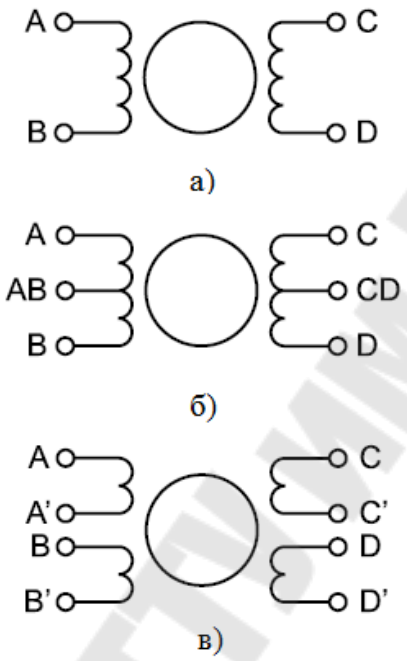


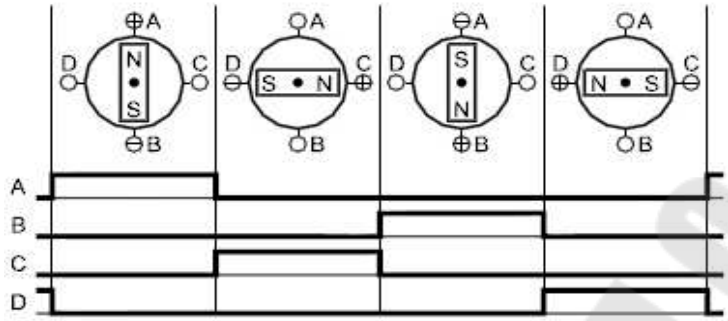
Рис. 2.2. Биполярный двигатель (*а*), униполярный (*б*) и четырехобмоточный (*в*)

Иногда униполярные двигатели имеют раздельные 4 обмотки (рис. 6.2, *в*), по этой причине их ошибочно называют 4-фазными двигателями. Каждая обмотка имеет отдельные выводы, поэтому всего выводов 8. При соответствующем соединении обмоток такой двигатель можно использовать как униполярный или как биполярный. Униполярный двигатель с двумя обмотками и отводами тоже можно использовать в биполярном режиме, если отводы оставить неподключенными. В любом случае ток обмоток следует выбирать так, чтобы не превысить максимальной рассеиваемой мощности.

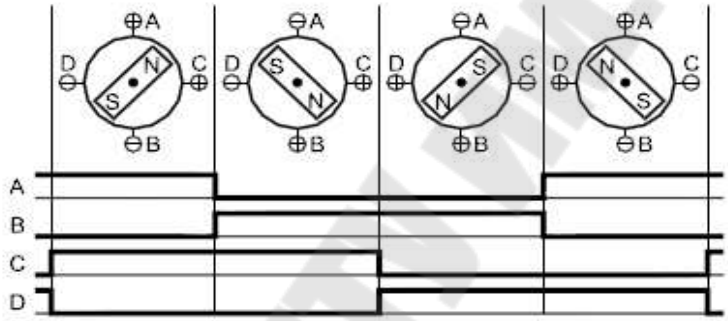
В настоящее время существуют специализированные микросхемы драйверов для биполярных двигателей, с использованием которых драйвер получается не сложнее, чем для униполярного двигателя. Например, это микросхемы L293E, L298N или L6202 фирмы SGS-Thomson, PBL3770, PBL3774 фирмы Ericsson, NJM3717, NJM3770,

NJM3774 фирмы JRC, A3957 фирмы Allegro, LMD18T245 фирмы National Semiconductor.

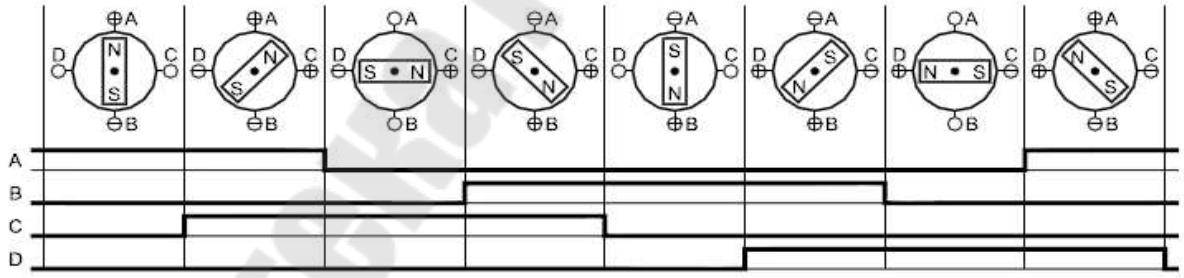
Существует несколько способов управления фазами шагового двигателя.



а) полношаговый режим, включена одна фаза,
"one phase on" full step (wave drive mode)



б) полношаговый режим, включены две фазы,
"two-phase-on" full step (full step mode)



в) полушаговый режим, "one and two-phase-on" half step (half step mode)

Рис.2.3. Способы управления фазами шагового двигателя

Первый способ обеспечивается попеременной коммутации фаз, при этом они не перекрываются, в один момент времени включена только одна фаза (рис. 2.3, *а*). Этот способ называют полношаговым режимом с включением одной фазы. Точки равновесия ротора для каждого шага совпадают с «естественными» точками равновесия ротора у незапитанного двигателя. Недостатком этого способа управле-

ния является то, что для биполярного двигателя в один и тот же момент времени используется 50 % обмоток, а для униполярного – только 25 %. Это означает, что в таком режиме не может быть получен полный момент.

Второй способ – управление фазами с перекрытием: две фазы включены в одно и то же время. Его называют полношаговым режимом с включением двух фаз. При этом способе управления ротор фиксируется в промежуточных позициях между полюсами статора (рис.2.3,б) и обеспечивается примерно на 40 % больший момент, чем в случае одной включенной фазы. Этот способ управления обеспечивает такой же угол шага, как и первый способ, но положение точек равновесия ротора смещено на пол шага.

Третий способ является комбинацией первых двух и называется полушаговым режимом, когда двигатель делает шаг в половину основного. Этот метод управления достаточно распространен, так как двигатель с меньшим шагом стоит дороже и очень заманчиво получить от 100-шагового двигателя 200 шагов на оборот. Каждый второй шаг запитана лишь одна фаза, а в остальных случаях запитаны две (рис. 2.3, *в*). В результате угловое перемещение ротора составляет половину угла шага для первых двух способов управления. Кроме уменьшения размера шага этот способ управления позволяет частично избавиться от явления резонанса. Полушаговый режим обычно не позволяет получить полный момент, хотя наиболее совершенные драйверы реализуют модифицированный полушаговый режим, в котором двигатель обеспечивает практически полный момент, при этом рассеиваемая мощность не превышает номинальной.

2.3. Задание к лабораторной работе

В системе проектирования Proteus создать модель подключения униполярного и биполярного шаговых двигателей к микроконтроллеру Atmega. Напишите программу:

– для управления биполярным шаговым двигателем с использованием драйвера L298;

– позволяющую сделать двигателю заданное число шагов в одном из направлений, а затем столько же в противоположном;

– в которой ШД будет совершать один шаг при каждом замыкании кнопочного выключателя, соединенного со входом INT0.

Исследовать программу управляющую скоростью и направлением вращения двигателей.

В качестве примера рассмотрим программу, позволяющую управлять скоростью шагового двигателя при помощи потенциометра, подключенного ко входу ADC0 контроллера Atmega168.

Анализируя схему, приведенную на рис. 2.4, можно отметить, что каждая полуобмотка ШД подключена к общему проводу схемы через один из транзисторов VT1…VT4, а средние точки обеих обмоток соединены с источником питания +24В. Таким образом, для того, чтобы ток протекал по полуобмотке ШД необходимо открыть соответствующий транзистор путём подачи на его базу от микроконтроллера логической «1».

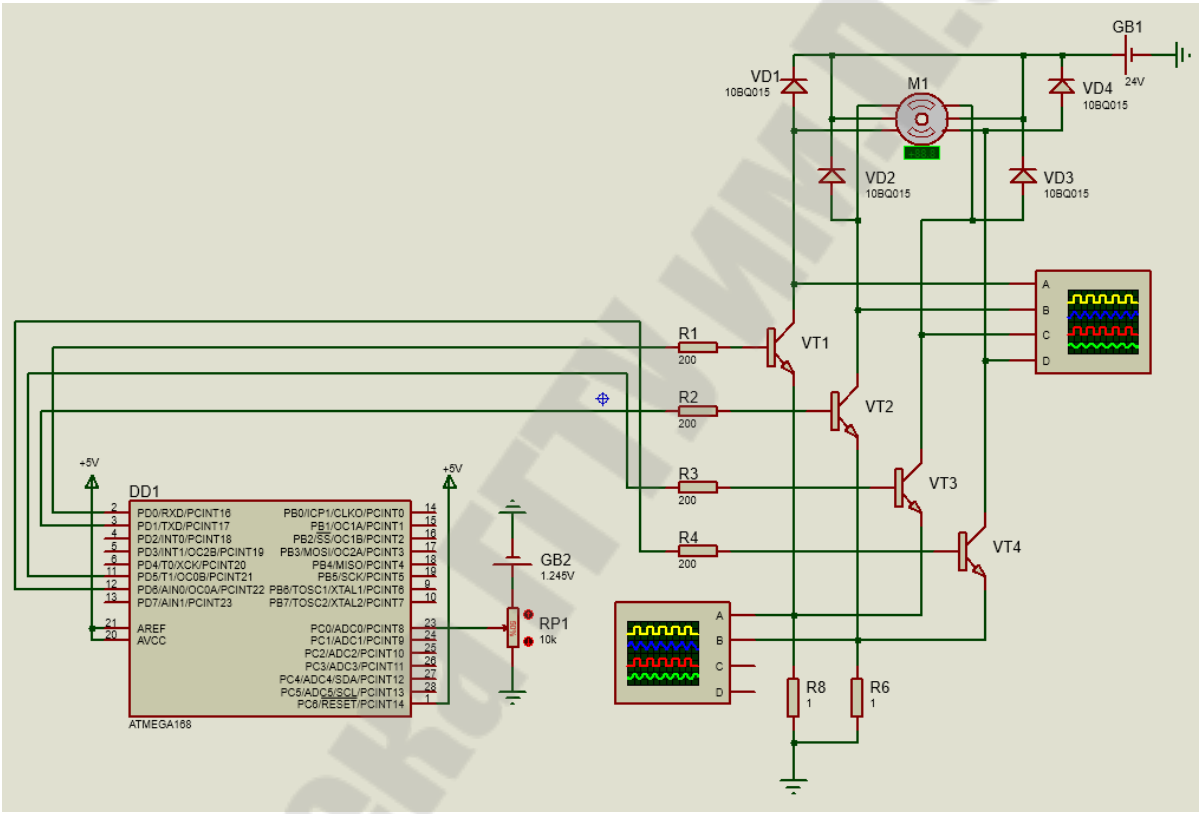


Рис. 2.4. Схема управления униполярным шаговым двигателем

За счёт определённой последовательности отпирания транзисторов можно задать различные режимы работы ШД: полношаговые «1-1» или «2-2» и полушаговый «1-2».

Таким образом, для управления ШД необходимо на каждом его шаге выставлять на линиях PD0, PD1, PD5, PD6 порта D определенный двоичный код.

Частота «шагания» ШД будет определяться частотой смены двоичных кодов на линиях порта D. Эту частоту необходимо задавать

с помощью потенциометра, подключенного ко входу АЦП микроконтроллера.

Рассмотрим текст программы.

В начале следует стандартный набор процедур: присоединение файла описания контроллера, присвоение имен регистрам, макрокоманды, определение сегмента памяти для хранения программы.

```
;************************* ЗАДАНИЕ ПАРАМЕТРОВ ***************************
.include "m168def.inc"              ; присоединение файла описания
ATmega168

.def temp = r16                     ; временный регистр
.def temp2 = r17                    ; временный регистр
.def cnt = r18                      ; счётчик состояний

;*************************** МАКРОКОМАНДЫ ******************************
.macro uout
      .if @0 < 0x40
            out @0, @1
      .else
            sts @0, @1
      .endif
.endm
;=============================================================================
.macro uin
      .if @1 < 0x40
            in @0, @1
      .else
            lds @0, @1
      .endif
.endm
;=============================================================================
.macro usbis
      .if @0 < 0x20
            sbis @0, @1
      .else
            push temp
            uin temp, @0
            sbrs temp, @1
            pop temp
      .endif
.endm
;=============================================================================
.macro usbic
      .if @0 < 0x20
            sbic @0, @1
      .else
            push temp
            uin temp, @0
            sbrc temp, @1
```

```
            pop temp
        .endif
.endm
;=====================================================================
.macro ldil
            push temp
            ldi temp, @1
            mov @0, temp
            pop temp
.endm

; FLASH ==============================================================
.cseg                   ; выбор сегмента памяти для хранения программы
```

В секции векторов прерывания указываем метки перехода для прерываний по сбросу «reset», по срабатыванию компаратора канала А таймера/счётчика Т2 «timer2_int» и по окончанию преобразования АЦП «adc_int».

```
;************************ ВЕКТОРЫ ПРЕРЫВАНИЙ ************************
.org 0
            rjmp reset          ; переход на обработку сброса
.org 2
            reti                ; внешнее прерывание IRQ0 (INT0)
.org 4
            reti                ; внешнее прерывание IRQ1 (INT1)
.org 6
            reti                ; прерывание 0 по изменению состояния выводов
.org 8
            reti                ; прерывание 1 по изменению состояния выводов
.org 10
            reti                ; прерывание 2 по изменению состояния выводов
.org 12
            reti                ; тайм-аут сторожевого таймера (WatchDog)
.org 14
            rjmp timer2_int     ; срабатывание компаратора А таймера Т2
.org 16
            reti                ; срабатывание компаратора В таймера Т2
.org 18
            reti                ; переполнение таймера Т2
.org 20
            reti                ; захват фронта таймера Т1
.org 22
            reti                ; срабатывание компаратора А таймера Т1
.org 24
            reti                ; срабатывание компаратора В таймера Т1
.org 26
            reti                ; переполнение таймера Т1
.org 28
            reti                ; срабатывание компаратора А таймера Т0
.org 30
```

```
            reti              ; срабатывание компаратора B таймера T0
.org 32
            reti              ; переполнение таймера T0
.org 34
            reti              ; передача по SPI завершена
.org 36
            reti              ; приём по USART завершен (USART Rx)
.org 38
            reti              ; регистр данных USART пуст
.org 40
            reti              ; передача по USART завершена (USART Tx)
.org 42
            rjmp adc_int      ; окончание преобразования АЦП
.org 44
            reti              ; готовность EEPROM
.org 46
            reti              ; аналоговый компаратор
.org 48
            reti              ; 2-х проводной последовательный интерфейс
.org 50
            reti              ; готовность SPM
;=====================================================================
```

Секция инициализации начинается с процедур обнуления РОН и ОЗУ, затем следует стандартная процедура загрузки указателя стека и настройка портов ввода/вывода. Кроме того, в данном примере, происходит отключение цифрового буфера входа ADC0 (хотя это не обязательно), а также загружается начальный адрес таблицы состояний ШД в индексный регистр Z.

```
;*************************** ИНИЦИАЛИЗАЦИЯ ****************************
reset:
; чистка ОЗУ
      Ldi zl,low(sram_start)        ; адрес начала ОЗУ в индекс
      Ldi zh, high(sram_start)
      clr temp                      ; очищаем temp
clr_ram:
      st z+, temp                   ; пишем 0 в ячейку памяти
      cpi zh, high(ramend+1)        ; достиг ли СБ индекса конца ОЗУ?
      brne clr_ram                  ; продолжаем чистку
      cpi zl,low(ramend+1)          ; достиг ли МБ индекса конца ОЗУ?
      brne clr_ram                  ; продолжаем чистку
      clr zl                        ; чистим индекс
      clr zh
;=====================================================================
; чистка РОН
      Ldi zl, 30
      clr zh
clr_ron:
      dec zl
      st z, zh
```

```
      brne clr_ron
;=====================================================================
; инициализация стека
      ldi temp, high(ramend)
      uout sph, temp
      ldi temp, low(ramend)
      uout spl, temp
;=====================================================================
; инициализация портов
      usbis didr0, ADC0D      ; отключение цифрового буфера линии ADC0
      ldi temp, 0b01100011    ; установка PD0, PD1, PD5, PD6 на вывод
      uout ddrd, temp

      ldi zl, low(full_2*2)   ; загрузка адреса таблицы состояний
      ldi zh, high(full_2*2)
;=====================================================================
```

Секция инициализации продолжается настройкой режима работы таймера/счётчика T2. Таймер настраивается на режим «сброс при совпадении», то есть частота прерываний будет зависеть от константы, записанной в регистр сравнения таймера. Здесь также задается делитель тактовой частоты, и разрешаются прерывания по совпадению в канале A.

Далее следует инициализация АЦП. Выбираем аналоговый вход ADC0, смещение результата вправо, разрешаем работу АЦП, однократное преобразование без автоматического перезапуска. Обязательно разрешаем прерывания от АЦП и выбираем делитель, такой, чтобы частота работы АЦП составляла от 50 до 200 кГц.

После всего разрешаем прерывания глобально.

```
;=====================================================================
; инициализация таймера Timer_2
      ldi temp, 0b01111111    ; начальная установка скважности
      uout ocr2a, temp
      ldi temp, 0b00000010
; COM2A1:0 0b[00]xxxxxx - режим формирования выходного сигнала OC2A
; COM2B1:0 0bxx[00]xxxx - режим формирования выходного сигнала OC2B
; -        0bxxxx[00]xx - зарезервирован
; WGM21:0  0bxxxxxx[10] - режим работы таймера (CTC)
      uout tccr2a, temp
      ldi temp, 0b00000111
; FOC2A    0b[0]xxxxxxx - принудительное изменение состояния OC2A
; FOC2B    0bx[0]xxxxxx - принудительное изменение состояния OC2B
; -        0bxx[00]xxxx - зарезервированы
; WGM22    0bxxxx[0]xxx - режим работы таймера (быстрая ШИМ)
; CS22:0   0bxxxxx[111] - выбор тактового источника с предделителем
;                         частоты 8МГц/8=1МГц
```

```
; При тактовой частоте 8МГц с включенным делителем на 8 имеем частоту
1МГц/1024/256=3,8 Гц
      uout tccr2b, temp
      ldi temp, 0b00000010
; -         0b[00000]xxx - зарезервировано
; OCIE2B    0bxxxxx[0]xx - разрешение прерывания по совпадению в канале B
; OCIE2A    0bxxxxxx[1]x - разрешение прерывания по совпадению в канале A
; CS22:0    0bxxxxxxx[0] - разрешение прерывания по переполнению
      uout timsk2, temp

;=====================================================================
; инициализация АЦП
      ldi temp, 0b00000000
; REFS1:0 0b[00]xxxxxx - выбор ИОН (напряжение AREF)
; ADLAR   0bxx[0]xxxxx - смещение 10-битного результата вправо
; -       0bxxx[0]xxxx - зарезервирован
; MUX3:0  0bxxxx[0000] - выбор аналогового входа (ADC0)
      uout admux, temp
      ldi temp, 0b11001011
; EDEN    0b[1]xxxxxxx - разрешение работы АЦП
; ADSC    0bx[1]xxxxxx - однократное преобразование
; ADFR    0bxx[0]xxxxx - разрешение автоматического запуска АЦП
; ADIF    0bxxx[0]xxxx - сброс флага прерывания АЦП
; ADIE    0bxxxx[1]xxx - разрешение прерывания от АЦП
; ADPS2:0 0bxxxxx[011] - выбор делителя частоты 8МГц/8/8=125 кГц
      uout adcsra, temp

      sei                     ; глобальное разрешение прерываний
;=====================================================================
```

Теперь следует секция основной программы (метка «main:»), которая представляет собой пустой цикл.

```
;********************** ОСНОВНАЯ ПРОГРАММА ****************************
main:
      rjmp main
;=====================================================================
```

Основные действия происходят в обработчиках прерываний. В прерывании от АЦП происходит чтение результата преобразования АЦП. Не смотря на то, что результат в данном примере не превышает 8 бит, необходимо читать оба байта регистра ADC (сначала младший, затем старший). Младший байт результата копируется в регистр сравнения таймера.

В прерывании от таймера/счётчика T2 производим считывание байта, определяющего состояние ШД на текущем шаге, из массива и отправляем его в порт D. При этом ведем подсчёт считанных из массива байт (состояний). Если все состояния из массива выбраны, сбра-

сываем адрес элементов массива в индексном регистре Z и счётчик числа элементов массива *cnt*. В конце этого прерывания запускаем новое преобразование АЦП.

```
;********************** ОБРАБОТЧИКИ ПРЕРЫВАНИЙ ***************************
; обработка прерывания от АЦП
adc_int:
      uin temp, adcl          ; вывод младшего байта АЦП в регистр temp
      uin temp2, adch         ; вывод старшего байта АЦП в регистр temp2
      uout ocr2a, temp        ; запись результата из АЦП в регистр
                              ; сравнения OCR2A Timer_2
      reti

;=====================================================================
; обработка прерывания от Timer_2
timer2_int:
      lpm temp, z+            ; чтение очередного элемента таблицы
      uout portd, temp        ; состояний и отправка в порт D
      inc cnt                 ; увеличение счётчика
      cpi cnt, 4              ; сравнение счётчика с числом состояний
                              ; двигателя
      brne end_t2_int         ; окончание прерывания, если все
                              ; состояния пройдены
      ldi zl, low(full_2*2)   ; загрузка адреса таблицы состояний
      ldi zh, high(full_2*2)
      clr cnt                 ; сброс счётчика
end_t2_int:
      ldi temp, 0b11001011    ; запускаем новое преобразование от АЦП
      uout adcsra, temp
      reti
;=====================================================================
```

Массив состояний ШД для полношагового режима «2-2» (режим так называется, потому что на каждом шаге включены две полуобмотки) представляет собой четыре байта, в каждом из которых присутствуют только 2 логические «1». Эти «единицы» соответствуют полуобмоткам ШД, которые необходимо подключить на текущем шаге.

```
;***************************** МАССИВЫ *********************************
; таблица состояний ШД
full_2:
.db 0b00000011, 0b00100010, 0b01100000, 0b01000001
```

2.4. Содержание отчета

В отчете привести наименование и цель работы, схему модели подключения ШД в Proteus, текст программы с комментариями к командам и директивам, осциллограммы работы модели.

Сделать выводы по существу полученных результатов.

2.5. Контрольные вопросы

1. Чем отличаются униполярный и биполярный ШД?
2. В чём отличие полношагового и полушагового способов управления ШД?
3. Какие аппаратные средства необходимы для управления частотой вращения ШД?
4. В чём состоит процесс инициализации АЦП микроконтроллера?
5. Охарактеризуйте режимы работы таймеров и обоснуйте выбор конкретного режима, используемого для управления ШД.
6. Опишите основные правила работы с таблицами данных.

Литература

1. Евстифеев А.В. Микроконтроллеры AVR семейства Mega. Руководство пользователя – М.: Издательский дом «Додэка-XXI», 2007. – 592 с.
2. Баранов В.Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы – М.: Издательский дом «Додэка-XXI», 2004. – 288с.
3. Бродин В.Б. Микроконтроллеры. Архитектура, программирование, интерфейс / В.Б. Бродин, М.И. Шагурин. - М.: ЭКОМ, 1999. - 400 с.
4. Водовозов А.М. Микроконтроллеры для систем автоматики: учебное пособие / А.М. Водовозов. - Вологда: ВоГТУ, 2002. - 123 с.
5. Рюмик С.М. 1000 и одна микроконтроллерная схема. Книга 1 / С.М. Рюмик. - М.: Додэка-XXI, 2012. - 356 с.
6. Хартов В.Я. Микроконтроллеры AVR. Практикум для начинающих: Учебное пособие / В.Я. Хартов. - М.: МГТУ им. Баумана, 2012. - 280 с.

Содержание

МЕРЫ БЕЗОПАСНОСТИ..... 3
Лабораторная работа № 1 Регулирование скорости вращения двигателя постоянного тока в системе ШИП-Д 3
Лабораторная работа № 2 Управление шаговым двигателем 13
Литература 25

Савельев Вадим Алексеевич

МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА В АВТОМАТИЗИРОВАННОМ ЭЛЕКТРОПРИВОДЕ

Практикум
по выполнению лабораторных работ
по одноименной дисциплине для студентов
специальности 1-53 01 05 «Автоматизированные электроприводы» дневной формы обучения

Подписано к размещению в электронную библиотеку
ГГТУ им. П. О. Сухого в качестве электронного
учебно-методического документа 15.02.20.

Рег. № 44Е.
http://www.gstu.by