Министерство образования Республики Беларусь

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

Кафедра «Автоматизированный электропривод»

В. А. Савельев, М. Н. Погуляев

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

Электронный аналог печатного издания

Гомель 2019

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

Рекомендовано к изданию научно-методическим советом
факультета автоматизированных и информационных систем
ГГТУ им. П. О. Сухого
(*протокол № 8 от 05.03.2018 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, М. Н. Погуляев. – Гомель : ГГТУ им. П. О. Сухого, 2019. – 48 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.

ISBN 978-985-535-398-1.

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

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

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

ISBN 978-985-535-398-1

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.

ЗАПРЕЩАЕТСЯ:

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

Знакомство с работой в среде программирования микроконтроллеров AVR Studio

1.1. Цель работы

1. Изучить особенности работы со средой программирования микроконтроллеров AVR Studio.
2. Ознакомиться с работой портов микроконтроллера.
3. Получить навыки написания простейшей программы на языке ассемблера.
4. Исследовать работу простейшей программы с помощью встроенного в AVR Studio эмулятора работы микроконтроллера.

1.2. Краткие сведения из теории

1.2.1. Среда разработки программ AVR Studio

Разработка программ в AVR Studio начинается с создания проекта. После установки программы это легче всего сделать через менеджер создания проектов во вкладке Project/Project Wizard.

После нажатия экранной кнопки New Project появится окно, приведенное на рис. 1.1, в котором надо задать название и директорию размещения проекта, например 168_LED (168, потому что для Atmega168, а LED, потому что программа будет управлять светодиодами). В качестве типа проекта необходимо выбрать Atmel AVR Assembler (проекты AVR GCC используют Си-компилятор WinAVR). Сейчас и в дальнейшем очень важно размещать все файлы проекта в одной папке, что избавит от многих проблем при редактировании и переносе программ.

ВНИМАНИЕ! Путь к программе и папке с сохраненными проектами не должен содержать имен, написанных кириллицей.

На следующем этапе необходимо выбрать модель микроконтроллера и тип отладочного средства, как показано на рис. 1.2 (в данном случае это ATmega168 и AVR Simulator соответственно), после чего откроется окно текстового редактора, где и будет непосредственно происходить создание программы (рис. 1.3).

Рис. 1.1. Окно Project Wizard AVR Studio

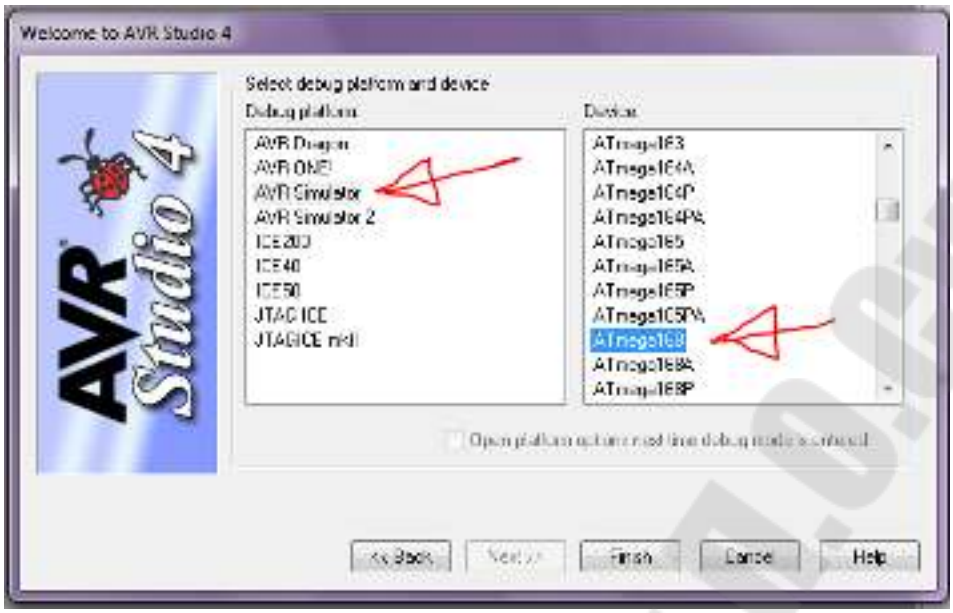


Рис. 1.2. Выбор отладчика (debug platform) и модели микроконтроллера

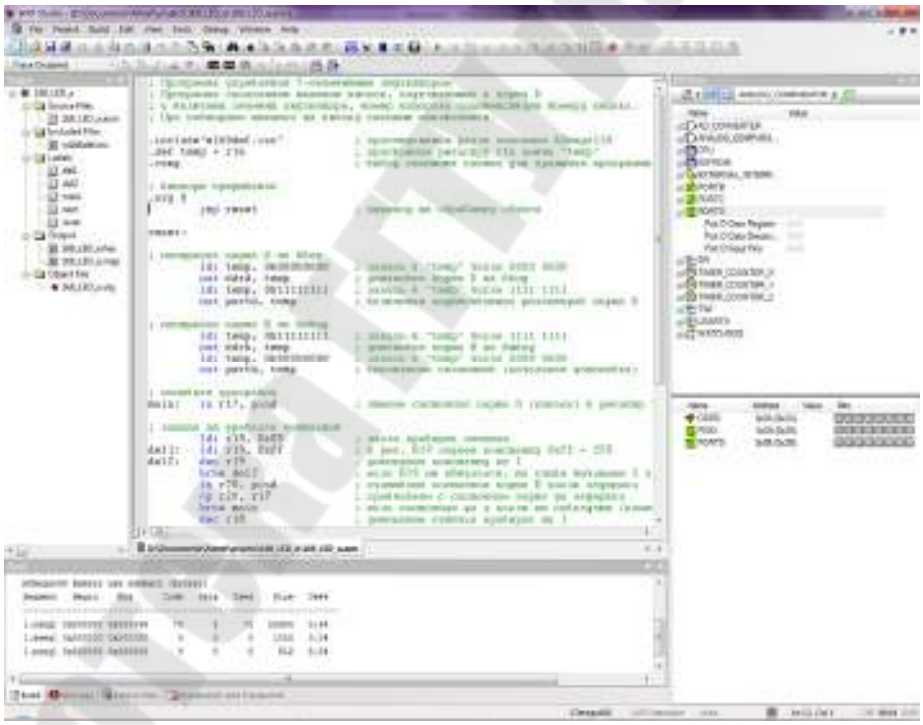


Рис. 1.3. Текстовый редактор AVR Studio

В окне проекта Project можно видеть все компоненты программы: файл с исходным текстом 168_LED_a.asm, файл описания контроллера m168def.inc, а также выходные файлы с расширениями. map, .hex и .obj с одноименными именами. В разделе Labels находятся символьные имена меток, встречающиеся в программе.

Компиляция проекта осуществляется после нажатия на иконку Assemble либо Assemble and Run. В последнем случае сразу же запускается и программа отладчика. Если в исходном тексте были допущены ошибки, то. hex, естественно, создан не будет, а в окне Build появится описание всех ошибок и строки, где они находятся. После внесения необходимых исправлений и успешной сборки в окне Build отобразится статистика о проекте в виде диапазонов адресов и размеров секций FLASH, SRAM и EEPROM.

Проверить работоспособность программы можно в симуляторе либо с помощью любого другого отладчика. Его запуск происходит после нажатия иконки Start Debagging (иконка в виде зеленого треугольника на панели инструментов).

На рис. 1.4 показана работа в симуляторе и вид основных отладочных окон, в которых можно наблюдать за состоянием содержимого различных областей памяти и символьными именами, объявленных в тексте программы. Все эти данные доступны для редактирования на ходу.

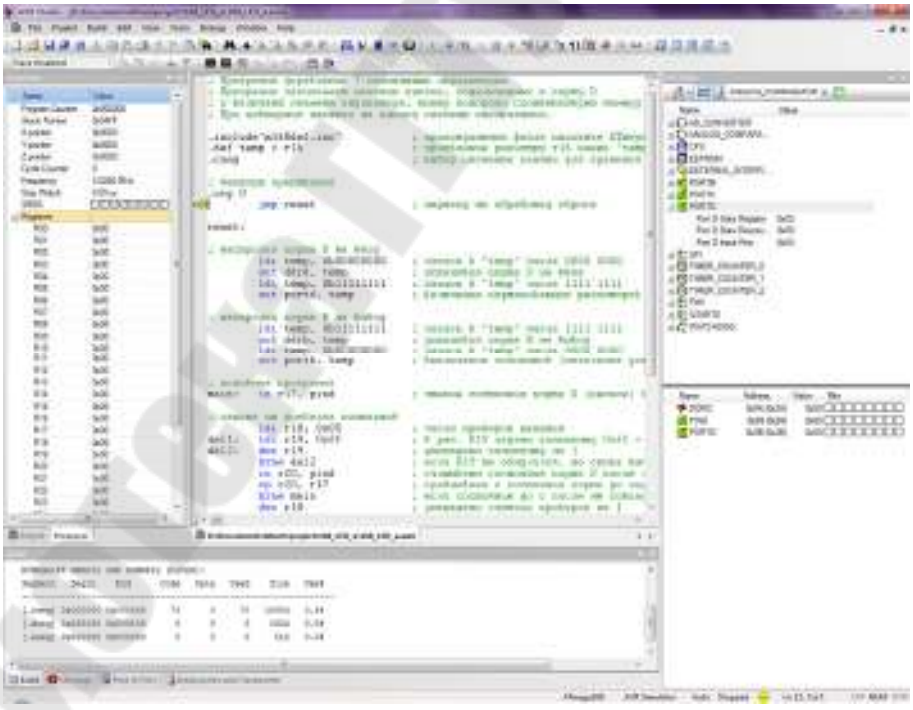


Рис. 1.4. Работа в симуляторе AVR Studio

Отладку можно вести как в пошаговом (кнопки Step Into, Step Over, Step Out), так в автоматическом (Auto Step) или ускоренном (Run) режимах. Имеется возможность использовать также **точки останова**.

Симулятор, встречая строку, в которой находится точка останова, принудительно останавливает свое выполнение, после чего можно детально изучить содержимое отладочных окон. Управление точками останова производится кнопками Toggle Brekpoint (включение точки останова) и Remove all Program Brekpoints (удаление всех точек останова). В окне Disassembler можно видеть соответствие машинных кодов командам ассемблера AVR.

Детальное описание всех компонентов AVR Studio можно найти во встроенной справочной системе.

1.2.2. Порты микроконтроллера

С внешним миром микроконтроллер общается через порты ввода/вывода. Каждый порт микроконтроллера AVR (обычно имеют имена A, B и т. д.) имеет 8 разрядов, каждый из которых привязан к определенной ножке (выводу, пину) корпуса. Каждый порт имеет три специальных регистра DDR*x*, PORT*x* и PIN*x* (где *x* соответствует букве порта A, B, C и т. д.). Назначение регистров следующее:

PIN*x* – это регистр чтения. Из него можно только читать. В регистре PIN*x* содержится информация о реальном текущем логическом уровне на выводах порта. Вне зависимости от настроек порта. Поэтому если необходимо узнать, что у нас на входе, то читаем соответствующий бит регистра PIN*x*.

DDR*x* – это регистр направления порта. Отдельные разряды порта в конкретный момент времени могут быть либо входом, либо выходом (но для состояния битов PIN*x* это значения не имеет. Читать из PIN*x* реальное значение можно всегда).

Если DDRx_n = 0 – *n*-й разряд порта *x* работает как ВХОД (ввод данных).

Если DDRx_n = 1 – *n*-й разряд порта *x* работает как ВЫХОД (вывод данных).

PORT*x* – режим управления состоянием разрядов порта. Когда мы настраиваем какие-либо разряды порта на ввод данных, то от PORT*x* зависит тип входа (Hi-Z или PullUp).

Когда *n*-й разряд порта настроен на ввод и при этом PORTx_n = 0, то данный разряд находится в режиме Hi-Z (без подтягивающего резистора).

Если PORTx_n = 1, то *n*-й разряд порта переходит в режим *PullUp* – внутри микроконтроллера включается подтягивающий резистор величиной 100 кОм между выводом порта и шиной питания.

Когда *n*-й разряд порта настроен как выход, то значение соответствующего бита в регистре PORT*x* определяет значение этого разряда порта. Если PORTx_n = 1, то на выводе логическая «1», если PORTx_n = 0, то на выводе логический «0».

1.3. Задания к лабораторной работе

В окне текстового редактора AVR Studio ввести текст программы управления светодиодами. Восемь светодиодов подключены к порту *B* микроконтроллера Atmega168. Восемь нормально разомкнутых контактов подключены к порту *D*. Светодиоды должны светиться при нажатии на соответствующий контакт.

Текст программы:

```
.include "m168def.inc"        ; подключаем файл описания
.def  temp = r16              ; присваиваем имя регистру R16
.cseg                         ; указываем сегмент памяти программ
.org 0                        ; начальный адрес программы
      rjmp reset              ; переход к метке «reset»

reset:
      ldi temp, 0b00000000    ; настраиваем порт D на ввод и
      out ddrd, temp          ; подключаем подтягивающие резисторы
      ldi temp, 0b11111111
      out portd, temp

      ldi temp, 0b11111111    ; настраиваем порт B на вывод
      out ddrb, temp          ; и выключаем сегменты
      ldi temp, 0b00000000
      out portb, temp

main:                         ; основная программа
      in temp, pind           ; читаем состояние порта D
      com temp                ; инвертируем полученные значения
      out portb, temp         ; записываем результат в порт B
      rjmp main               ; переходим к началу основной программы.
```

Привести комментарии к командам и директивам программы.

Произвести отладку программы.

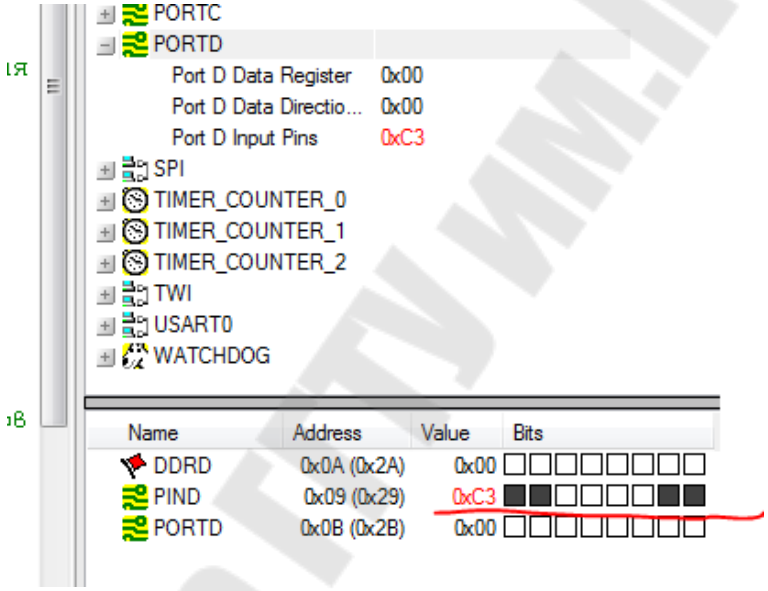
Выполняя программу в пошаговом режиме, после выполнения каждого шага, записывать в табл. 1.1 содержимое регистров, занятых в выполнении программы.

Таблица 1.1

Состояние регистров контроллера

Шаг №	PC	temp	DDRD	PORTD	PIND	DDRB	PORTB	PINB	Комментарии
0	0	0	0	0	0	0	0		Исходные значения
1									
…									

ПРИМЕЧАНИЕ! Перед выполнением команды *in temp*, *pind* необходимо вручную установить произвольное значение разрядов регистра PIND, кликнув по пустым битам левой кнопкой мыши, например, так:



1.4. Содержание отчета

В отчете привести наименование и цель работы, текст программы с комментариями к командам и директивам, а также заполненную в соответствии с заданием табл. 1.1 с подробными комментариями.

Сделать выводы по существу полученных результатов.

1.5. Контрольные вопросы

1. Для чего предназначена среда разработки AVR Studio?
2. Как создать новый проект в AVR Studio?
3. Как пользоваться эмулятором AVR Studio?
4. Что такое порт микроконтроллера?
5. Опишите назначение регистров, относящихся к портам микроконтроллеров AVR.

Лабораторная работа № 2

Статическая индикация. Изучение команд перехода

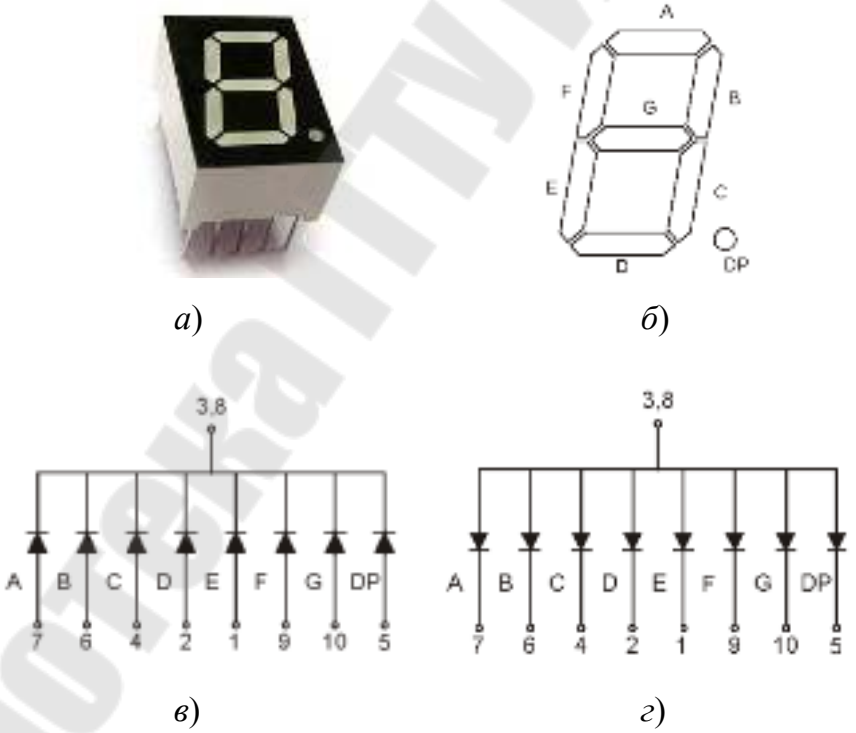
2.1. Цель работы

1. Ознакомиться со статическим управлением светодиодным индикатором.
2. Исследовать действие команд перехода.
3. Получить практические навыки составления программ.

2.2. Краткие сведения из теории

2.2.1. Семисегментные светодиодные индикаторы

Семисегментный индикатор выглядит примерно так, как показано на рис. 2.1, *а*. Почему семисегментный, когда сегментов на самом деле – восемь? Потому что 8-й сегмент – это десятичная точка, которая не входит в изображение цифры и вообще является необязательной. Бывают индикаторы и без точек.



а) *б*)

в) *г*)

Рис. 2.1. Семисегментный индикатор:
а – внешний вид; *б* – нумерация сегментов;
в – схема с общим катодом; *г* – схема с общим анодом

Для нумерации сегментов используются латинские буквы от *a* до *h*. У всех семисегментных индикаторов сегменты нумеруются в том порядке, как это показано на рис. 2.1, *б*.

Индикаторы бывают с общим катодом (ОК) и с общим анодом (ОА). Поэтому нужно быть внимательными, чтобы не перепутать. Хотя в данной ситуации это не имеет значения, так как индикаторами будет управлять контроллер, а его можно запрограммировать как на работу с ОК, так и с ОА.

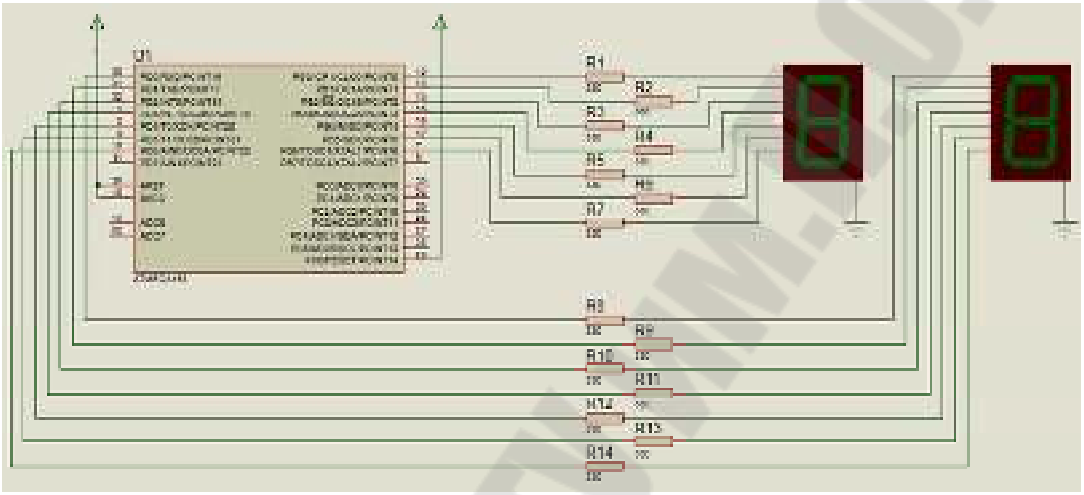


Рис. 2.2. Схема включения индикаторов
при статической индикации

Семисегментным индикатором можно управлять статически или динамически. При статическом управлении (рис. 2.2) разряды индикатора подключены к микроконтроллеру независимо друг от друга, и информация на них выводится постоянно. Этот способ управления проще динамического, но без использования таких дополнительных элементов, как сдвиговые регистры, подключить многоразрядный семисегментный индикатор к микроконтроллеру будет проблематично – может не хватить выводов.

2.2.2. Команды перехода

Команды перехода позволяют изменять последовательность выполнения команд программы. Существуют два способа изменения этой последовательности.

Первый из них называется безусловным (табл. 2.1). Согласно этому способу последовательность выполнения программы подвергается изменению всякий раз, когда реализуется команда «безусловный переход».

Таблица 2.1

Некоторые команды безусловного перехода

Мнемокод	Описание	Операция	Число тактов
RJMP k	Относительный безусловный переход	PC = PC + k + 1	2
JMP k	Абсолютный переход	PC = K	3

В соответствии со вторым (условным) способом последовательность выполнения программы определяется некоторыми условиями, т. е. изменяется в том случае, если значение проверяемого условия совпадает с заданным (табл. 2.2 и 2.3).

Таблица 2.2

Некоторые команды условного перехода

Мнемокод	Описание	Операция	Число тактов
BRBC s, k	Переход, если флаг S регистра SREG сброшен	Если S = 0, то PC = PC + k + 1	1/2
BRBS s, k	Переход, если флаг S регистра SREG установлен	Если S = 1, то PC = PC + k + 1	1/2
BRCS k	Переход по переносу	Если C = 1, то PC = PC + K + 1	1/2
BRCC k	Переход, если нет переноса	Если C= 0, то PC = PC + K + 1	1/2
BREQ k	Переход по «равно»	Если Z = 1, то PC = PC + K + 1	1/2
BRNE k	Переход по «не равно»	Если Z = 0, то PC = PC + K + 1	1/2
BRTS k	Переход, если флаг T установлен	Если T = 1, то PC = PC + K + 1	1/2
BRTC k	Переход, если флаг T сброшен	Если T = 0, то PC = PC + K + 1	1/2

Таблица 2.3

Некоторые команды проверки-пропуска

Мнемокод	Описание	Операция	Число тактов
CPSE Rd, Rr	Сравнение и пропуск следующей команды при равенстве	Если Rd = Rr, то PC = PC + 2(3)	1/2/3

Окончание табл. 2.3

Мнемокод	Описание	Операция	Число тактов
SBRC Rr, b	Пропуск следующей команды, если бит РОН* сброшен	Если Rd, b = 0, то PC = PC + 2(3)	1/2/3
SBRS Rr, b	Пропуск следующей команды, если бит РОН установлен	Если Rd, b = 0, то PC = PC + 2(3)	1/2/3
SBIC a, b	Пропуск следующей команды, если бит РВВ** сброшен	Если a, b = 0, то PC = PC + 2(3)	1/2/3
SBIS a, b	Пропуск следующей команды, если бит РВВ установлен	Если a, b = 1, то PC = PC + 2(3)	1/2/3

*РОН – регистр общего назначения;
**РВВ – регистр ввода/вывода.

2.3. Задание к лабораторной работе

Исследовать программу, управляющую семисегментным индикатором. Принципиальная схема приведена на рис. 2.3.

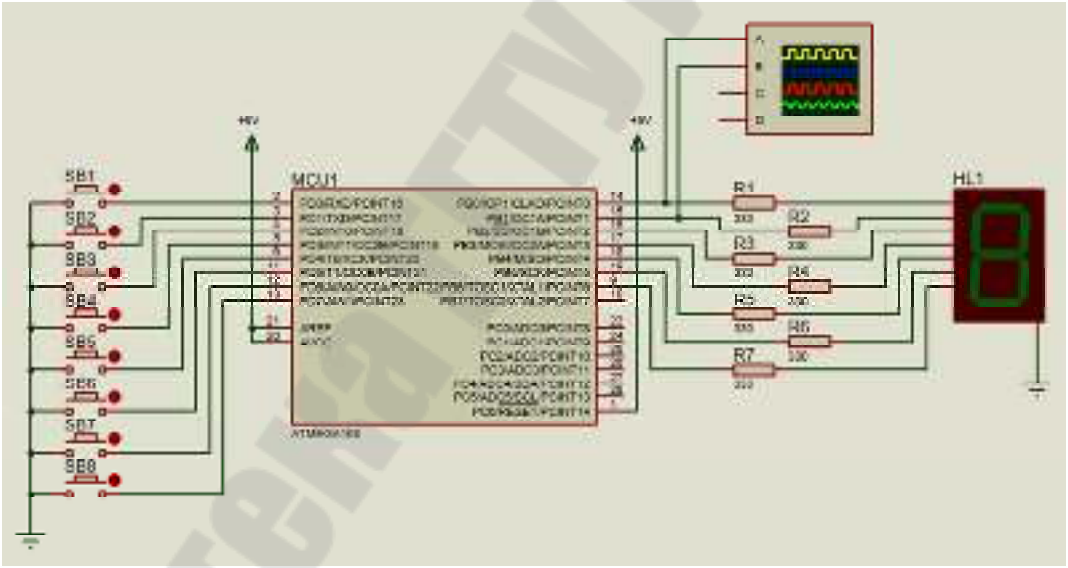


Рис. 2.3. Смеха подключения семисегментного индикатора к контроллеру в программе Proteus 8 Professional

Программа запоминает нажатие контактов, подключенных к порту *D*, и включает сегмент индикатора, номер которого соответствует номеру контакта. При повторном нажатии на контакт сегмент отключается. Нажатие на контакт, подключенный к старшему разряду порта *D*, отключает все сегменты индикатора.

Текст программы:

```
.include "m168def.inc"        ; подключаем файл описания
.def  temp = r16              ; присваиваем имена регистрам R16,
.def  knop = r17              ; R17, R18
.def  pam = r18
.cseg                         ; указываем сегмент памяти программ,
.org 0                        ; начальный адрес программы,
      jmp reset               ; адрес перехода по сбросу.

reset:
      ldi temp, 0b00000000    ; настраиваем порт D на ввод
      out ddrd, temp          ; и подключаем подтягивающие
      ldi temp, 0b11111111    ; резисторы
      out portd, temp
      ldi temp, 0b11111111    ; настраиваем порт B на вывод
      out ddrb, temp
      ldi temp, 0b00000000
      out portb, temp

main:                         ; основная программа
      in knop, pind           ; читаем порт D в регистр «knop»
      cp pam, knop            ; сравниваем предыдущее состояние
                              ; контактов с текущим
      breq main               ; если состояние контактов не изменилось,
                              ; переходим на метку «main»
      mov pam, knop           ; в противном случае запоминаем новое
                              ; состояние контактов
      rol knop                ; проверяем старший разряд:
      brcs next               ; если кнопка сброса не нажата,
                              ; переходим на метку «next»,
      ldi temp, 0             ; в противном случае выводим в порт B
      out portb, temp         ; нули – гасим сегменты
      jmp main                ; и возвращаемся в начало основной
                              ; программы.
next:
      ror knop                ; восстанавливаем значение
                              ; регистра «knop»
```

```
eor temp, knop     ; объединяем информацию
                   ; регистров «temp» и «knop»
com temp           ; включаем сегменты
out portb, temp
jmp main           ; завершаем цикл
```

Произвести отладку программы. Выполняя программу в пошаговом режиме в среде AVR Studio, после выполнения каждого шага записывать в табл. 2.4 содержимое регистров, занятых в выполнении программы.

Таблица 2.4

Состояние регистров контроллера

Шаг №	PC	PIND	PORTB	temp	pam	knop	Флаг C	Флаг Z	Комментарии
0	0	0	0	0	0	0	0	0	Исходные значения
1									
…									

!При исследовании программы протестировать ее работу до нажатия кнопки сброса без нажатых кнопок и с любой нажатой кнопкой, кроме кнопки сброса, а также после нажатия кнопки сброса.

2.4. Содержание отчета

В отчете привести наименование и цель работы, текст программы с комментариями к командам и директивам, а также заполненную в соответствии с заданием табл. 2.4 с подробными комментариями.

Сделать выводы по существу полученных результатов.

2.5. Контрольные вопросы

1. Что представляет семисегментный светодиодный индикатор, для чего предназначен?
2. Какие виды семисегментных светодиодных индикаторов существуют?
3. Что такое статическая индикация? Ее достоинства и недостатки.
4. В чем отличие команд условного и безусловного перехода?
5. Опишите процесс выполнения команд перехода.
6. Чем отличаются команды перехода от команд пропуска?

Лабораторная работа № 3

Вывод информации на семисегментные индикаторы. Динамическая индикация

3.1. Цель работы

1. Изучить схему сопряжения микроконтроллера с многоразрядным семисегментным индикатором в режиме динамической индикации.

2. Разработать и отладить программу вывода информации на устройство динамической индикации.

3.2. Краткие сведения из теории

3.2.1. Отображение информации на семисегментных индикаторах

Давайте разберемся, каким образом получаются двоичные коды, соответствующие десятичным цифрам. Проще всего для определения двоичного кода воспользоваться программой SegCodeGen. Окно программы представлено на рис. 3.1.

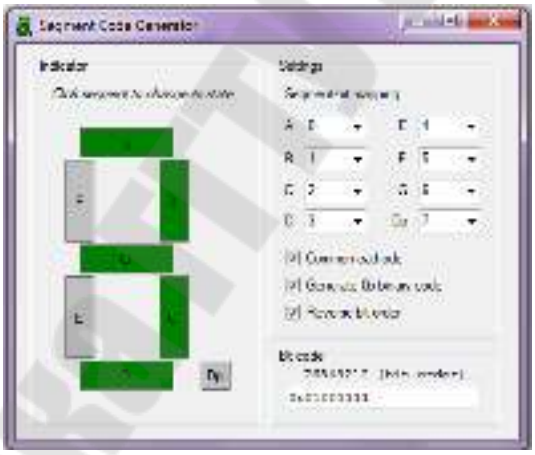


Рис. 3.1. Окно программы SegCodeGen

Во-первых, необходимо определиться, к каким выводам порта подключены сегменты индикатора. Для этого используется настройка «Segment-bit mapping». Здесь цифры обозначают номера линий порта, а буквы – подключенные к этим линиям сегменты (сегмент A подключен к линии PB0, сегмент B – к линии PB1 и т. д.).

Во-вторых, необходимо указать какой тип индикатора используется – ОА или ОК (если ОК, то поставить галочку «Common cathode»). Ведь при использовании индикаторов с ОА для включения сегмента на соответствующей линии порта необходим «0», а при использовании индикатора с ОК – нужна «1».

В-третьих, необходимо указать порядок следования цифр в двоичном коде – прямой или обратный (если младший разряд должен быть справа, то нужно поставить галочку «Reverse bit order»).

Наконец, в левой части окна программы необходимо выделить сегменты, соответствующие нужной цифре.

Таким образом, для схемы на рис. 3.1, для индикаторов с ОК имеем код цифры «3» – 0b01001111, который и используем в тексте программы.

Допустим, мы хотим отображать 3-разрядное число. Для этого нам необходимо 3 индикатора, у каждого из которых 8 выводов (считая точку). То есть нам потребуется 24 вывода портов. Но ведь следует еще куда-то подключить кнопки, датчики и т. п. В этой ситуации нам и поможет динамическая индикация. Динамическая индикация – это метод отображения многоразрядного числа через быстрое последовательное отображение отдельных разрядов этого числа. Причем «целостность» восприятия получается благодаря инерционности человеческого зрения.

Делаем так, как показано на рис. 3.2: подключаем все индикаторы параллельно, точнее, соединяем выводы сегментов на общую шину (в данном случае к выводам PB6…PB0 порта *B*), а общие провода индикаторов подключаем раздельно (в данном случае к выводам PC3…PC1 порта *C*).

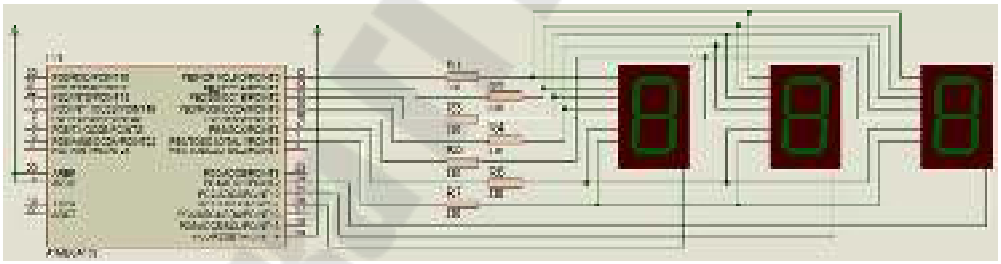


Рис. 3.2. Схема включения индикаторов
при динамической индикации

!Несмотря на то что каждый из выходов микроконтроллера AVR способен работать при силе выходного тока до 20 мА, в отношении предельных параметров устройства действуют определенные ограничения. Так, сумма всех выходных токов не должна превышать 200 мА, а сумма выходных токов портов *B* и *C* – 100 мА.

Мы последовательно подаем напряжение на адресные (общие) выводы индикаторов, и одновременно выдаем в порт *B* семисегментный код, соответствующий индикатору, активному в данный момент.

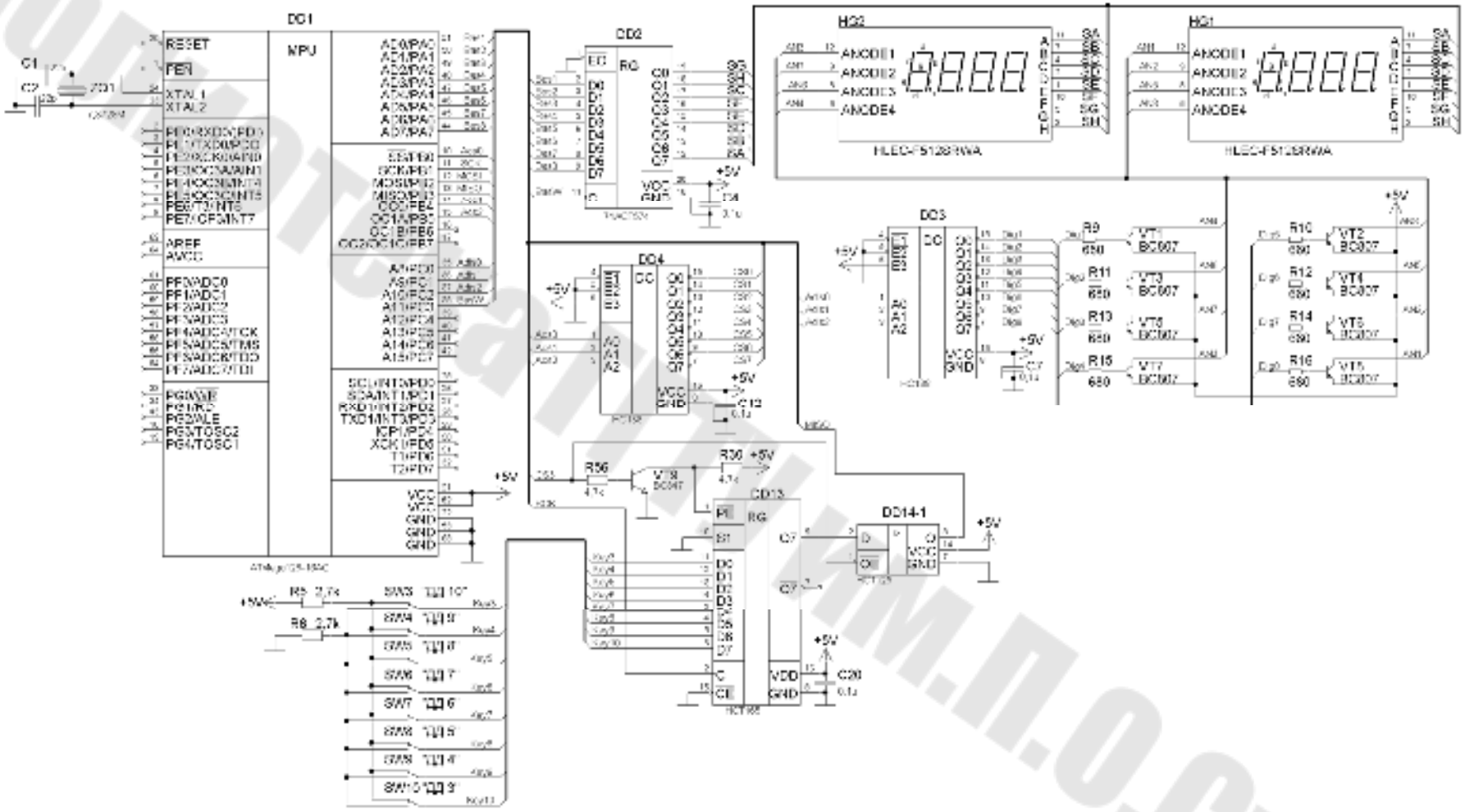


Рис. 3.3. Электрическая принципиальная схема динамической индикации стенда

3.2.2. Описание принципиальной схемы сопряжения индикаторов и микроконтроллера

В схеме (рис. 3.3) для подключения к микроконтроллеру сегментов символьного светодиодного индикатора HG1-HG2 использован параллельный регистр DD2 1594ИР37 (74ACT574).

Для организации параллельной шины данных используется 8-битный порт *PA* микроконтроллера. Для защелкивания информации в регистре используется линия PC3 порта *C*. Для переключения разрядов индикаторов используются транзисторные ключи VT1–VT8, управляемые выходами микросхемы DD3 дешифратора HC138.

Для управления дешифратором используются линии PC0 – PC2 порта *C* микроконтроллера.

3.3. Задание к лабораторной работе

Написать программу, позволяющую выводить информацию на восьмиразрядный семисегментный индикатор учебного стенда (см. рис. 3.3) с использованием динамической индикации.

Собрать модель схемы учебного стенда в системе проектирования Proteus и исследовать работу схемы с использованием инструмента «логический анализатор».

Проверить работоспособность программы на учебном стенде.

В качестве примера рассмотрим текст программы для схемы, приведенной на рис. 3.2:

```
.include "m168def.inc"        ; подключаем файл описания
.def  temp = r16              ; присвоили имя регистру r16

; макрокоманда задержки
.macro delay
        push temp             ; сохраняем в стеке temp, xl, xh
        push xl
        push xh
        ldi temp, @1          ; указываем число повторов счета
m1:
        ldi xl, low(@0)       ; указываем константу счета
        ldi xh, high(@0)
m2:
        sbiw xl, 1            ; уменьшаем константу на 1
        brne m2               ; если константа не равна 0,
```

```
                        ; переходим на метку «m2»
        dec r16         ; уменьшаем число повторов на 1
        brne m1         ; если число повторов не равно 0,
                        ; то переходим на метку «m1»
        pop xh          ; восстанавливаем из стека
        pop xl          ; xh, xl, temp
        pop temp
.endm

; макрокоманда загрузки PBB
macro outp
        push temp       ; сохраняем в стеке temp
        ldi temp, @1    ; записываем в «temp» значение
        out @0, temp    ; и отправляем это значение в PBB
        pop temp        ; восстанавливаем из стека temp
.endm

.cseg                   ; указываем сегмент памяти
.org 0                  ; и адрес начала программы
        rjmp reset

reset:
        outp spl, low(ramend)       ; инициализация стека
        outp sph, high(ramend)
        outp ddrb, 0xff             ; настройка порта B  на вывод
        outp ddrc, 0x0E             ; настройка разрядов PC3...PC1
                                    ; порта C на вывод

main:                               ; основная программа
; включаем 1-й индикатор
        outp portc, (0<<pc3)|(1<<pc2)|(1<<pc1)
        outp portb, 0b00000111      ; отправляем код «7»
        delay 0xffff, 2             ; первая задержка времени
        outp portb, 0b00000000      ; удаляем код «7»
; включаем 2-й индикатор
        outp portc, (1<<pc3)|(0<<pc2)|(1<<pc1)
        outp portb, 0b01101101      ; отправляем код «5»
        delay 0xffff, 2             ; вторая задержка времени
        outp portb, 0b00000000      ; удаляем код «5»
```

```
; включаем 3-й индикатор
        outp portc, (1<<pc3)|(1<<pc2)|(0<<pc1)
        outp portb, 0b01001111      ; отправляем код «3»
        delay 0xffff, 2             ; третья задержка времени
        outp portb, 0b00000000      ; удаляем код «3»
        rjmp main                   ; зацикливаем программу
```

Для исследования динамической индикации к схеме (см. рис. 3.2), собранной в системе проектирования Proteus, подключим логический анализатор и загрузим приведенную выше программу (hex-файл). Как видно из рис. 3.4, индикаторы включаются поочередно, при этом на них отображается одна из трех цифр десятичного числа «753». Если время задержки, задаваемое в программе уменьшить до 10 мс, то наши глаза будут наблюдать одновременное свечение трех индикаторов.

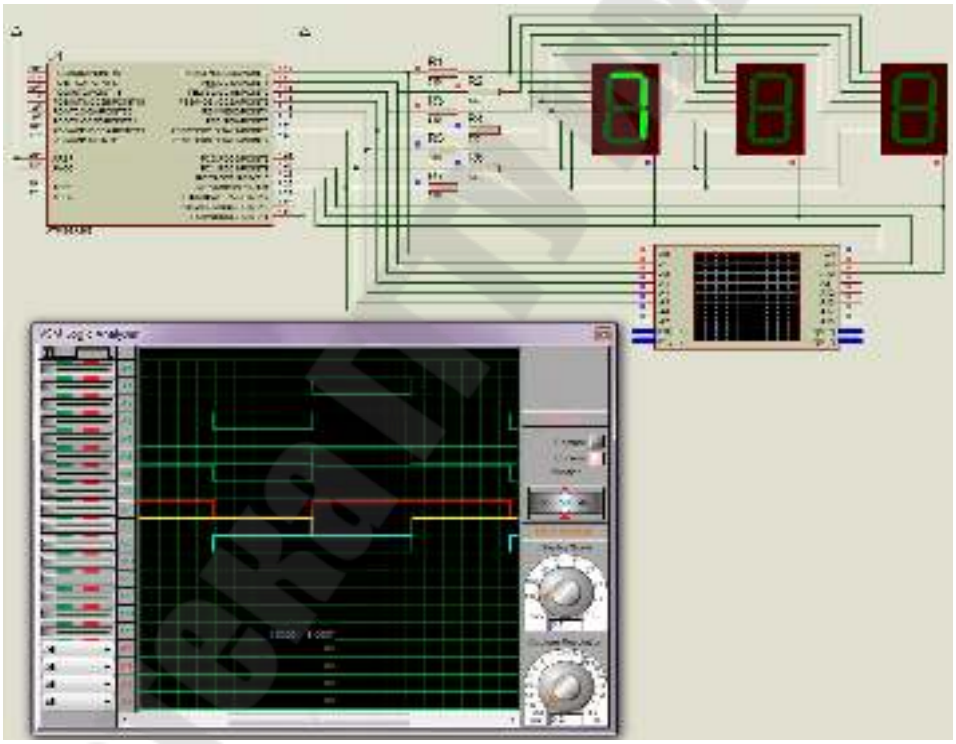


Рис. 3.4. Анализ работы программы динамической индикации

Как видно из диаграмм логического анализатора, при нулевом уровне на линии PC3 (диаграмма красного цвета) активен индикатор старшего разряда, а на линиях порта *B* присутствует код 00000111 (диаграммы зеленого цвета – снизу вверх), что соответствует цифре «7». Нулевому уровню на диаграмме желтого цвета соответствует включение среднего индикатора и код 01101101 на линиях порта *B*,

т. е. цифра «5». Нулевому уровню на диаграмме голубого цвета соответствует включение индикатора младшего разряда и код 01001111 на линиях порта *B*, т. е. цифра «3»

3.4. Содержание отчета

В отчете привести наименование и цель работы, текст программы с комментариями к командам и директивам, результаты анализа работы программы в системе Proteus с подробными комментариями.

Сделать выводы по существу полученных результатов.

3.5. Контрольные вопросы

1. Что представляет собой динамическая индикация? Какие она имеет преимущества перед статической?
2. Каким образом формируется двоичный (шестнадцатеричный) код символов, выводимых на семисегментный индикатор?
3. Что такое макрокоманда и каковы принципы ее написания?
4. Каков алгоритм вывода информации на индикаторы при динамической индикации?
5. Дайте описание принципиальной схемы учебного стенда.
6. Как работает инструмент «логический анализатор» системы проектирования Proteus?

Лабораторная работа № 4

Исследование устройства матричной жидкокристаллической индикации

4.1. Цель работы

1. Ознакомиться с документацией на устройство матричной жидкокристаллической индикации.
2. Изучить принципиальную электрическую схему сопряжения микроконтроллера с устройством матричной индикации учебного стенда.
3. Разработать и отладить программу вывода информации на устройство матричной жидкокристаллической индикации.

4.2. Краткие сведения из теории

По отношению к обыкновенным семисегментным ЖКИ модули на базе контроллера HD44780 обладают на порядок большими возможностями. Количество строк на экране у разных моделей – 1, 2 или 4;

число символов в строке: 8, 10, 16, 20, 24, 30, 32 или 40. Каждое знакоместо на дисплее представляет собой матрицу размером 5 × 8 точек. Индикатор может иметь светодиодную или люминесцентную подсветку практически любого цвета свечения.

Индикатор SC-1602 представляет собой жидкокристаллический матричный индикатор (ЖКИ), содержащий 2 строки по 16 символов со встроенным контроллером. Внешний вид и условное обозначение индикатора приведено на рис. 4.1. Функциональное назначение выводов индикатора приведено в табл. 4.1.

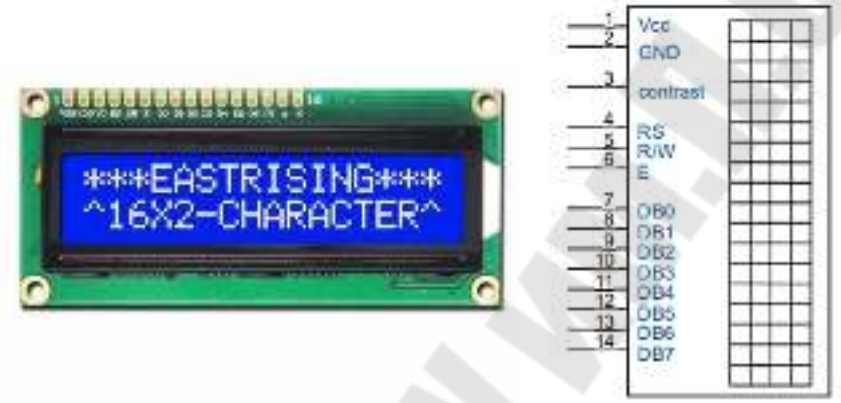


Рис. 4.1. Внешний вид и условное обозначение индикатора SC-1602

Таблица 4.1

Назначение выводов индикатора SC-1602

Обозначение выводов	Номер	Назначение
V_{CC}	1	Напряжение питания
GND	2	Общий провод
Contrast	3	Регулятор контраста
RS	4	Выбор регистра (1 – регистр данных, 0 – регистр команд)
R/W	5	1 – чтение данных, 0 – запись данных
E	6	Сигнал разрешения чтения/записи
DB0-DB7	7–14	Биты данных

Изначально HD44780 имеет предопределенную таблицу символов, размещенную в ОЗУ знакогенератора CGRAM (Character Generator RAM). Для отображения любого из них программа микроконтроллера должна передать координаты позиции и, непосредственно за ними, сам адрес символа из CGRAM. Пример таблицы CGRAM

приведен на рис. 4.2. Заглавные и прописные буквы латинского алфавита, числовые знаки, а также большинство знаков препинания совпадают в ней с кодами ASCII. Набор символов, размещенных по адресам 0xA0…0xFF, содержит национальный алфавит (в данном случае кириллицу) того региона, где предполагается его использование. Первые 16 ячеек CGRAM имеют особое значение. В них могут быть записаны любые пользовательские символы, которых нет в таблице (сразу после включения модуля в них находится случайная информация).

Рис. 4.2. Таблица ASCII-символов индикатора

На рис. 4.3 приведена принципиальная электрическая схема подключения индикатора SC-1602 в учебном стенде. Для управления жидкокристаллическим индикатором используются линии PF0…PF3, PE2, PD5, PD7 микроконтроллера.

В табл. 4.2 приведена система команд индикатора SC-1602.

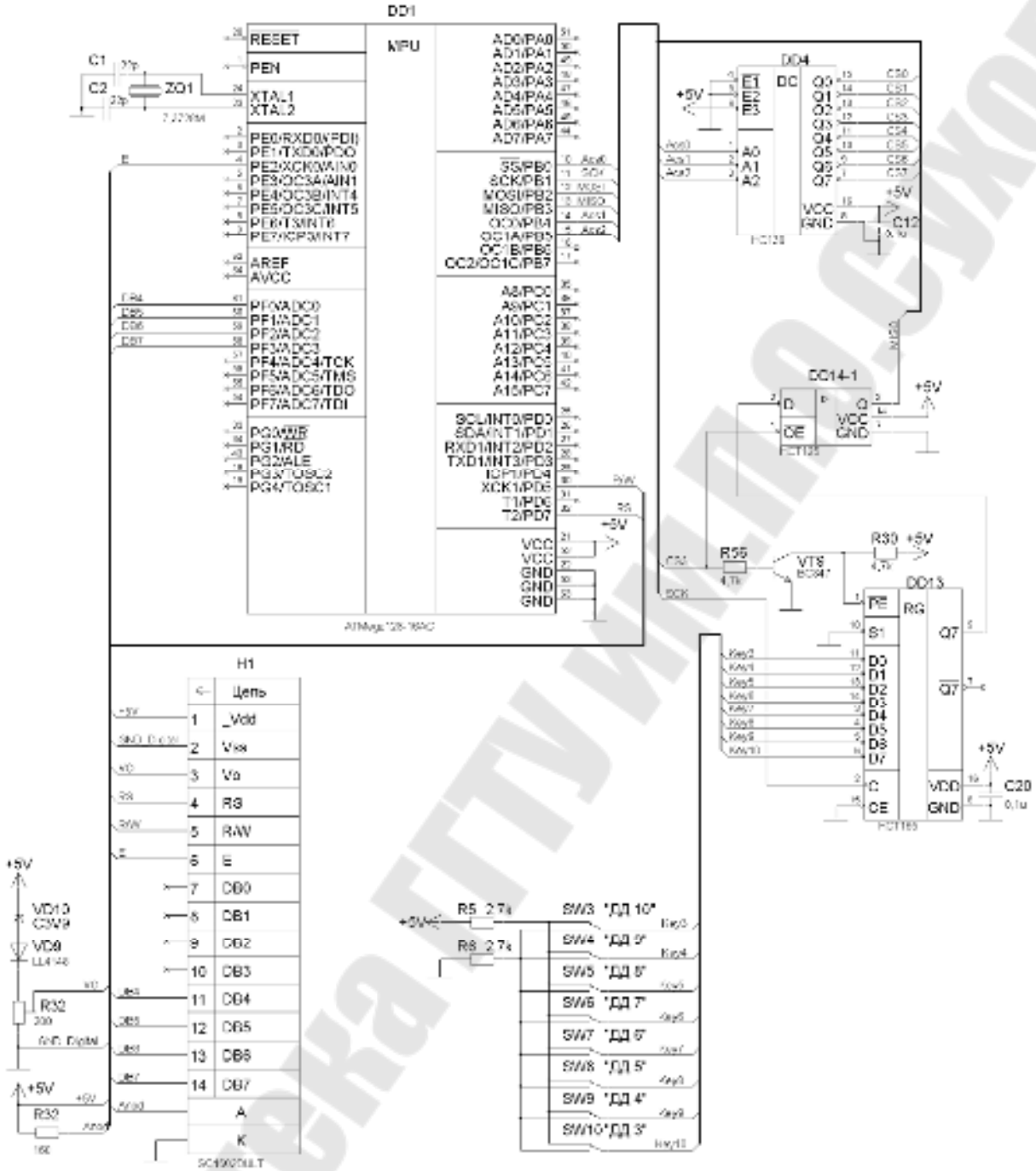


Рис. 4.3. Схема электрическая принципиальная подключения индикатора SC-1602 в учебном стенде

Таблица 4.2

Система команд индикатора SC-1602

Команда	Код										Описание	Время выполнения
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Очистка индикатора	0	0	0	0	0	0	0	0	0	1	Очистка дисплея и перевод курсора в начальную позицию	1,5 мс
Возврат в начало	0	0	0	0	0	0	0	0	1	–	Перевод курсора в начальную позицию	40 мкс
Установка режима ввода	0	0	0	0	0	0	0	1	I/D	S	Установка направления перемещения курсора и разрешение дисплея	40 мкс
Вкл./выкл. индикатора	0	0	0	0	0	0	1	D	C	B	Вкл./выкл. дисплея (D) или курсора (C) и установка режима мерцания символов	40 мкс
Сдвиг курсора	0	0	0	0	0	1	S/C	R/L	–	–	Перемещение курсора и сдвиг дисплея без изменения DDRAM	40 мкс
Установка режимов	0	0	0	0	1	DL	N	F	–	–	Установка 8/4-битного режима, количества строк и шрифта	40 мкс
Установка адреса CGRAM	0	0	0	1	A_{CG}						Установка адреса CGRAM	40 мкс
Установка адреса DDRAM	0	0	1	A_{DD}							Установка адреса DDRAM	40 мкс

Окончание табл. 4.2

Команда	Код										Описание	Время выполнения
	RS	**R/W**	**DB7**	**DB6**	**DB5**	**DB4**	**DB3**	**DB2**	**DB1**	**DB0**		
Чтение флага BUSY и адреса	0	1	BF	AC							Чтение флага BUSY и содержимого счетчика	0
Запись данных в CG или DDRAM	1	0	A_{CG} или A_{DD}								Запись данных в DDRAM или CGRAM	40 мкс
Чтение данных из CG или DDRAM	1	1	A_{CG} или A_{DD}								Чтение данных в DDRAM или CGRAM	40 мкс

I/D = 1(0) – увеличение (уменьшение) S = 1 – сопровождает сдвиг дисплея S/C = 1 – сдвиг дисплея S/C = 0 – смещение курсора R/L = 1 – сдвиг вправо R/L = 0 – сдвиг влево DL = 1 – 8 бит DL = 0 – 4 бит N = 1 – 2 строки N = 0 – 1 строка F = 1 – 5 × 10 точек F = 0 – 5 × 7 точек BF = 1 – индикатор занят BF = 0 – доступ разрешен	DDRAM – RAM данных дисплея; CGRAM – RAM генератора символов; A_{CG} – CGRAM адрес; A_{DD} – DDRAM адрес (соответствует адресу курсора); AC – счетчик адреса

Последовательность операций, необходимых для инициализации дисплея SC-1602 в 4-битном режиме, показана на рис. 4.4.

Рис. 4.4. Последовательность инициализации дисплея SC-1602 в 4-битном режиме

4.3. Задание к лабораторной работе

Разработать программу для учебного стенда, позволяющую отображать на ЖК-дисплее заданную надпись в виде бегущей строки.

Произвести отладку программы в системе проектирования Proteus и на учебном стенде.

Результат продемонстрировать преподавателю.

В качестве примера рассмотрим программу, позволяющую вывести на дисплей статическую надпись в две строки: «ЧАСТОТА ВРАЩЕНИЯ» и «об/мин». Поскольку дисплей, примененный в учебном стенде не имеет кириллицы, нам придется создать необходимые кириллические символы.

Текст программы:

```
.include "m128def.inc"          ; присоединение файла описания

; макрокоманда пересылки из РОН в любой РВВ
.macro uout
            .if @0 < 0x40
                  out @0, @1
            .else
                  sts @0, @1
            .endif
.endm

; макрокоманда пересылки из любого РВВ в РОН
.macro uin
      .if @1 < 0x40
                  in @0, @1
            .else
                  lds @0, @1
            .endif
.endm

; указываем номера пинов контроллера, соединенных с ЖК-дисплеем
.set RS = 7
.set E = 2
.set RW = 5
.set D4 = 0
.set D5 = 1
```

```
.set D6 = 2
.set D7 = 3
.def col = r9                   ; количество байт в массиве
                                ; пользовательских символов
.def del1 = r12                 ; регистры задержки
.def del2 = r13
.def temp = r16                 ; временный регистр
.cseg                           ; выбор сегмента памяти
                                      ; для хранения программы

        ; таблица векторов прерываний
        .org 0
              rjmp reset        ; переход на обработку сброса
        .org 2
              reti              ; переход на обработку запроса IRQ0
        …
        …                       ; таблица прерываний
        …
        .org 68
              reti              ; переход на обработку прерывания при
                                ; готовности выполнения команды SPM

reset:
; инициализация стека
        ldi temp, high(ramend)
        uout sph, temp
        ldi temp, low(ramend)
        uout spl, temp

; начальная настройка портов
        ldi temp, 0b00001111        ; настройка линий порта F
        uout ddrf, temp
        ldi temp, (1<<E)            ; настройка линий порта E
        uout ddre, temp
        ldi temp, (1<<RS)|(1<<RW)   ; настройка линий порта C
        uout ddrd, temp

; ******** ИНИЦИАЛИЗАЦИЯ ЖК-ДИСПЛЕЯ HD44780 *********
```

```
; 1. Пауза после подачи питания (20 мс)
        rcall del_5ms      ; 4 раза вызываем подпрограмму задержки
        rcall del_5ms      ; на 5 мс
        rcall del_5ms
        rcall del_5ms

; 2. Установка разрядности (8 бит!)
; отправка только старшего полубайта команды
        ldi temp, 0b00110000
                           ; D7  0b[0]xxxxxxx
                           ; D6  0bx[0]xxxxxx
                           ; D5  0bxx[1]xxxxx – 1 – признак команды
                           ; D4  0bxxx[1]xxxx – 1 = 8-битный режим
        swap temp          ; меняем местами младший
                           ; и старший полубайты
        rcall send_com     ; отправляем код команды в индикатор
        rcall del_5ms      ; формируем задержку 5 мс

; 3. Установка разрядности (4 бита!), числа строк и размера
; символов
; отправка только старшего полубайта
        ldi temp, 0b00100000
                           ; D7  0b[0]xxxxxxx
                           ; D6  0bx[0]xxxxxx
                           ; D5  0bxx[1]xxxxx – 1 – признак команды
                           ; D4  0bxxx[0]xxxx – 0 = 4-битный режим
        swap temp          ; меняем местами младший и старший
                           ; полубайты
        rcall send_com     ; отправляем код команды в индикатор
        rcall del_5ms      ; формируем задержку 5 мс
; отправка всей команды (так как после отправки предыдущего
; полубайта дисплей сразу переходит в режим 4 бита и ожидает
; новую команду)
        ldi temp, 0b00101100
                           ; D7  0b[0]xxxxxxx
                           ; D6  0bx[0]xxxxxx
                           ; D5  0bxx[1]xxxxx – 1 – признак команды
                           ; D4  0bxxx[0]xxxx – 0 = 4-битный режим
                           ; N   0bxxxx[1]xxx – 1 = 2 строки
```

```
                        ; F    0bxxxxx[1]xx – 1 = шрифт 5x10
                        ; D1   0bxxxxxx[0]x
                        ; D0   0bxxxxxxx[0]
        rcall com_out   ; отправляем код команды в индикатор
        rcall del_5ms   ; формируем задержку 5 мс

; 4. Очистка дисплея и установка курсора в нулевую позицию
        ldi temp, 0b00000001
        rcall com_out   ; отправляем код команды в индикатор
        rcall del_5ms   ; формируем задержку 5 мс

; 5. Направление сдвига курсора и записи символов
        ldi temp, 0b00000110
                        ; D7  0b[0]xxxxxxx
                        ; D6  0bx[0]xxxxxx
                        ; D5  0bxx[0]xxxxx
                        ; D4  0bxxx[0]xxxx
                        ; D3  0bxxxx[0]xxx
                        ; D2  0bxxxxx[1]xx – 1 – признак команды
                        ; I/D 0bxxxxxx[1]x – 1 = сдвиг вправо
                        ; S    0bxxxxxxx[0] – 0 = сдвиг экрана запрещен
        rcall com_out   ; отправляем код команды в индикатор

; 6. Включение дисплея, курсор не виден
        ldi temp, 0b00001100
                        ; D7  0b[0]xxxxxxx
                        ; D6  0bx[0]xxxxxx
                        ; D5  0bxx[0]xxxxx
                        ; D4  0bxxx[0]xxxx
                        ; D3  0bxxxx[1]xxx – 1 – признак команды
                        ; D    0bxxxxx[1]xx – 1 = дисплей включен
                        ; C    0bxxxxxx[0]x – 0 = курсор не виден
                              ; B    0bxxxxxxx[0] – 0 = курсор не мигает
        rcall com_out   ; отправляем код команды в индикатор

;**********ПОЛЬЗОВАТЕЛЬСКИЕ СИМВОЛЫ**********

; команда записи в CGRAM по адресу 0
        ldi temp, 0b01000000
```

```
                        ; D7  0b[0]xxxxxxx
                        ; D6  0bx[1]xxxxxx – 1 – признак команды
                        ; D5  0bxx[0]xxxxx – бит адреса CGRAM
                        ; D4  0bxxx[0]xxxx – бит адреса CGRAM
                        ; D3  0bxxxx[0]xxx – бит адреса CGRAM
                        ; D2  0bxxxxx[0]xx –  бит адреса CGRAM
                        ; D1  0bxxxxxx[0]x – бит адреса CGRAM
                        ; D0  0bxxxxxxx[0] –  бит адреса CGRAM
        rcall com_out   ; отправляем код в индикатор
; записываем 8 пользовательских символов
        ldi temp, 64            ; задаем количество байт в массиве
        mov col, temp           ; пользовательских символов
        ldi zh, high(2*symbol) ; задаем адрес массива
        ldi zl, low(2*symbol)   ; пользовательских символов
        line:
        lpm temp, z+            ; читаем очередной элемент
                                ; массива с постинкрементом
        rcall data_out          ; и выводим его в CGRAM
        dec col                 ; уменьшаем счетчик на 1
        tst col                 ; проверяем, не обнулился ли счетчик
        brne line               ; если массив не окончен, переходим к
                                ; выводу следующего символа,
                                        ; иначе переходим к выводу текста

; ******************** ВЫВОД ТЕКСТА ********************

; установка курсора на первую строку (стр.1, поз.0)
        ldi temp, 0b10000000
                        ; D7  0b[1]xxxxxxx – 1 – признак команды
                        ; D6  0bx[0]xxxxxx – бит адреса DDRAM
                        ; D5  0bxx[0]xxxxx – бит адреса DDRAM
                        ; D4  0bxxx[0]xxxx – бит адреса DDRAM
                        ; D3  0bxxxx[0]xxx – бит адреса DDRAM
                        ; D2  0bxxxxx[0]xx – бит адреса DDRAM
                        ; D1  0bxxxxxx[0]x – бит адреса DDRAM
                        ; D0  0bxxxxxxx[0] – бит адреса DDRAM
        rcall com_out

; вывод текста первой строки: «Частота вращения»
```

```
        ldi zh, high(2*text_1)   ; задаем адрес массива кодов
        ldi zl, low(2*text_1)    ; символов 1-й строки
line_1:
        lpm temp, z+             ; читаем очередной элемент
                                 ; массива с постинкрементом
        tst temp                 ; проверяем на нулевой символ,
                                 ; означающий конец массива
        brne brk_1               ; если массив не окончен,
                                 ; переходим по метке «brk_1»,
        rjmp end_1               ; переходим к выводу второй строки
brk_1:
        rcall data_out           ; выводим текущий символ
        rjmp line_1              ; переход к следующему символу
end_1:

; установка курсора на вторую строку (стр.2, поз.47)
        ldi temp, 0b11000111
                                 ; D7  0b[1]xxxxxxx – 1 – признак команды
                                 ; D6  0bx[1]xxxxxx – бит адреса DDRAM
                                 ; D5  0bxx[0]xxxxx – бит адреса DDRAM
                                 ; D4  0bxxx[0]xxxx – бит адреса DDRAM
                                 ; D3  0bxxxx[0]xxx – бит адреса DDRAM
                                 ; D2  0bxxxxx[0]xx – бит адреса DDRAM
                                 ; D1  0bxxxxxx[0]x – бит адреса DDRAM
                                 ; D0  0bxxxxxxx[0] – бит адреса DDRAM
        rcall com_out

; вывод текста 2-й строки «об/мин»
        ldi zh, high(2*text_2)   ; задаем адрес массива кодов
        ldi zl, low(2*text_2)    ; символов 2-й строки
line_2:
        lpm temp, z+             ; читаем очередной элемент
                                 ; массива с постинкрементом
        tst temp                 ; проверяем на нулевой символ,
                                 ; означающий конец массива
        brne brk_2               ; если массив не окончен,
                                 ; переходим к выводу символа
        rjmp end_2               ; иначе переходим к пустому циклу
brk_2:
```

```
        rcall data_out
        rjmp line_2             ; переход к следующему символу
end_2:

; *************** ОСНОВНАЯ ПРОГРАММА **************

main:
        rjmp main               ; пустой цикл

; массив пользовательских символов (8 символов)
symbol:
.db 0b00000000, 0b00000000, 0b00000000, 0b00000000
.db 0b00000000, 0b00000000, 0b00000000, 0b00000000  ; «»

.db 0b00010001, 0b00010001, 0b00010001, 0b00001111
.db 0b00000001, 0b00000001, 0b00000001, 0b00000000  ; «Ч»

.db 0b00010101, 0b00010101, 0b00010101, 0b00010101
.db 0b00010101, 0b00010101, 0b00011111, 0b00000001  ; «Щ»

.db 0b00010001, 0b00010001, 0b00010011, 0b00010101
.db 0b00011001, 0b00010001, 0b00010001, 0b00000000  ; «И»

.db 0b00001111, 0b00010001, 0b00010001, 0b00001111
.db 0b00000101, 0b00001001, 0b00010001, 0b00000000  ; «Я»

.db 0b00001111, 0b00010000, 0b00001110, 0b00010001
.db 0b00010001, 0b00010001, 0b00001110, 0b00000000  ; «б»

.db 0b00000000, 0b00000000, 0b00010001, 0b00011011
.db 0b00010101, 0b00010101, 0b00010001, 0b00000000  ; «м»

.db 0b00000000, 0b00000000, 0b00010001, 0b00010001
.db 0b00011111, 0b00010001, 0b00010001, 0b00000000  ; «н»

; массив кодов символов 1-й строки
text_1:
.db 0x01, 0x41, 0x43, 0x54, 0x4f, 0x54, 0x41, 0x20
.db 0x42, 0x50, 0x41, 0x02, 0x45, 0x48, 0x03, 0x04, 0
```

```
; массив кодов символов 2-й строки
text_2:
.db 0x6f, 0x05, 0x2f, 0x06, 0x75, 0x07, 0

; ********** ПОДПРОГРАММЫ ВЫВОДА В LCD **********

; подпрограмма вывода команд
com_out:
        push temp               ; делаем копию кода команды в стек
        swap temp               ; меняем местами младший и старший
                                ; полубайты
        andi temp, (1<<D7|1<<D6|1<<D5|1<<D4)
                                ; маскируем старший полубайт кода
        rcall send_com          ; отправляем старший полубайт в LCD
        pop temp                ; достаем копию кода команды из стека
        andi temp, (1<<D7|1<<D6|1<<D5|1<<D4)
                                ; маскируем старший полубайт кода
        rcall send_com          ; отправляем байт в LCD
        rcall del_50us          ; пауза не менее 40 мкс
        ret
; подпрограмма пересылки команд в ЖК-дисплей
send_com:
        cbi portd, RW           ; устанавливаем RW = 0
        cbi portd, RS           ; устанавливаем RS = 0
        sbi porte, E            ; устанавливаем E = 1 (передний фронт)
        uout portf, temp        ; вывод байта в порт F
        ldi temp, 3             ; формирование длительности
loop:                           ; строб-импульса (1 мкс)
        dec temp
        brne loop
        nop
        cbi porte, E            ; задний фронт строб-импульса
        ret

; подпрограмма вывода символов
data_out:
        push temp               ; делаем копию кода символа в стек
        swap temp               ; меняем местами младший
                                ; и старший полубайты
```

```
        andi temp, (1<<D7|1<<D6|1<<D5|1<<D4)
                                ; маскируем старший полубайт кода
        rcall send_data         ; отправляем полубайт в LCD
        pop temp                ; достаем копию кода символа из стека
        andi temp, (1<<D7|1<<D6|1<<D5|1<<D4)
                                ; маскируем старший полубайт кода
        rcall send_data         ; отправляем байт в LCD
        rcall del_50us          ; пауза не менее 40 мкс
        ret

; подпрограмма пересылки данных в ЖК-дисплей
send_data:
        cbi portd, RW           ; устанавливаем RW = 0
        sbi portd, RS           ; устанавливаем RS = 1
        nop                     ; пауза не менее 60 мкс между RS и E
        sbi porte, E            ; устанавливаем E = 1
        uout portf, temp        ; вывод байта в порт D
        ldi temp, 3             ; формирование длительности
loop_1:                         ; строб-импульса (1 мкс)
        dec temp
        brne loop_1
        nop
        cbi porte, E            ; задний фронт строб-импульса
        ret

; ********************** ЗАДЕРЖКИ **********************

; подпрограмма задержки на 50 мкс
del_50us:
        ldi temp, 134
        mov del1, temp
loop_2:
        dec del1
        brne loop_2
        nop
        ret

; подпрограмма задержки на 5 мс
del_5ms:
```

```
        ldi temp, 244
        mov del1, temp
        ldi temp, 52
        mov del2, temp
loop_3:
        dec del1
        brne loop_3
        dec del2
        brne loop_3
        nop
        ret
```

4.4. Содержание отчета

В отчете привести наименование и цель работы, схему электрическую принципиальную подключения индикатора, текст программы с комментариями к командам и директивам, а также результат, полученный в программе Proteus или на учебном стенде.

Сделать выводы по существу полученных результатов.

4.5. Контрольные вопросы

1. Опишите возможности жидкокристаллического дисплея SC-1602.
2. Что такое таблица ASCII-кодов и как ею пользоваться?
3. Что представляет собой система команд дисплея SC-1602?
4. Какова последовательность инициализации дисплея?
5. Опишите алгоритм вывода кодов команд (данных) в дисплей.
6. Опишите алгоритм создания пользовательских символов.

Лабораторная работа № 5

Регулирование скорости вращения двигателя постоянного тока в системе ШИП-Д

5.1. Цель работы

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

5.2. Краткие сведения из теории

Функциональная схема микросхемы L298N приведена на рис. 5.1. 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.

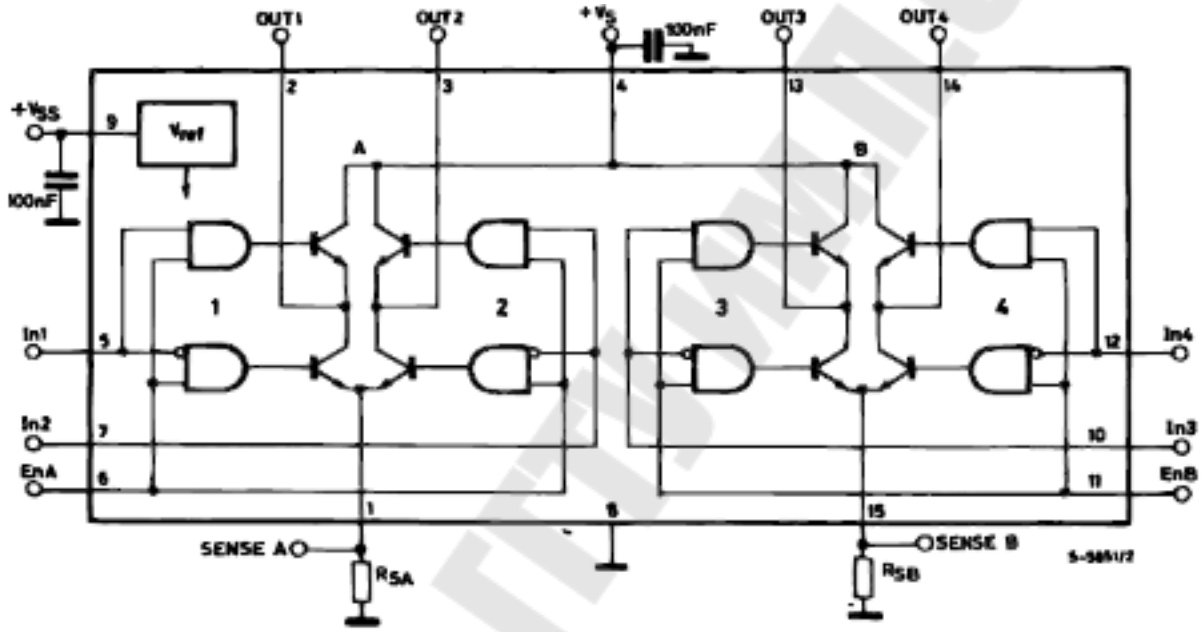


Рис. 5.1. Функциональная схема микросхемы L298N

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. 5.1, а в табл. 5.2 представлена логика работы драйвера.

Таблица 5.1

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 2 каналов
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 каналов
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 5.2

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).

5.3. 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. 5.2).

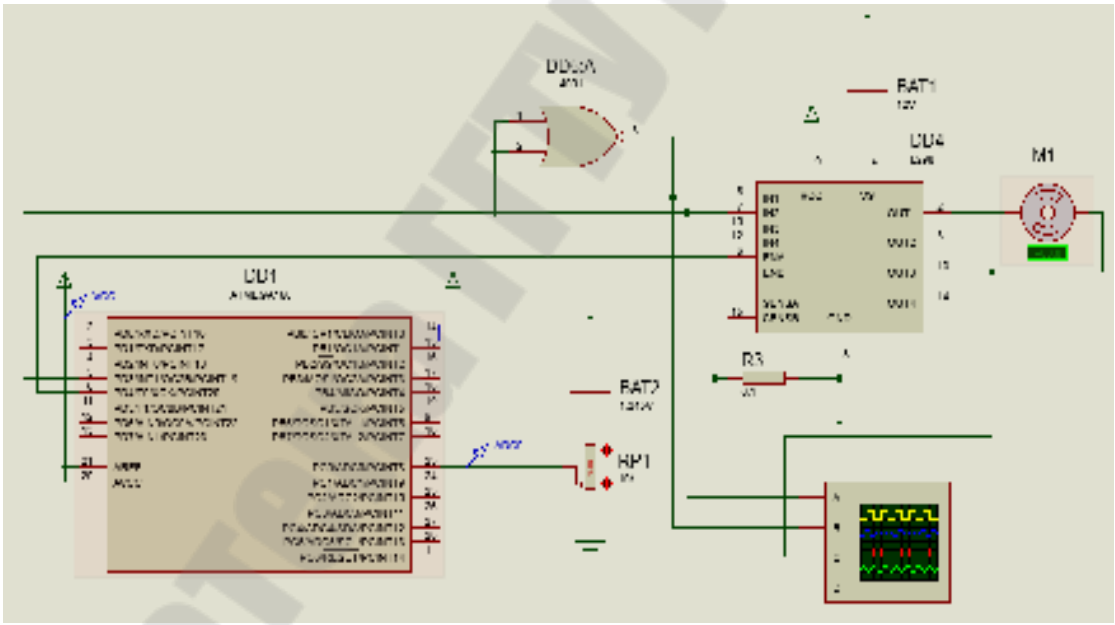


Рис. 5.2. Модель привода постоянного тока в Proteus

Текст программы:

```
.include "m168def.inc"          ; присоединение файла описания
.def  temp = r16                ; временный регистр
.def  rezl = r17                ; младший байт результата АЦП
.def  rezh = r18;               ; старший байт результата АЦП
```

```
.def  col = r19                    ; константа усреднения
.cseg                              ; выбор сегмента памяти

; векторы прерываний
.org 0

        jmp reset                  ; переход на обработку сброса
…
…                                  ; таблица прерываний
…
.org 42
        rjmp adc_int               ; окончание преобразования АЦП

reset:
; инициализация стека
        ldi temp, high(ramend)
        out sph, temp
        ldi temp, low(ramend)
        out spl, temp

; начальная настройка портов
        ldi temp, 0b00000001       ; отключение цифрового буфера
        sts didr0, temp            ; линии ADC0
        ldi temp, 0b00011000       ; установка линии PD3 порта D
        out ddrd, temp             ; на вывод
        ldi col, 64                ;        константа        усреднения

; 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
;                         при тактовой частоте контроллера 8 МГц
;                         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
empty:                                  ; пустой цикл
        rjmp empty

; 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
; поскольку при пуске с максимальным напряжением возникают
; броски тока, для их исключения блокируем драйвер до тех пор,
; после этого устанавливается флаг T и разрешается работа
; драйвера со скважностью 50%
        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. 5.3 показаны осциллограммы работы схемы, полученные при симуляции в программе Proteus. На рис. 5.4 показаны осциллограммы, полученные при экспериментальном исследовании.

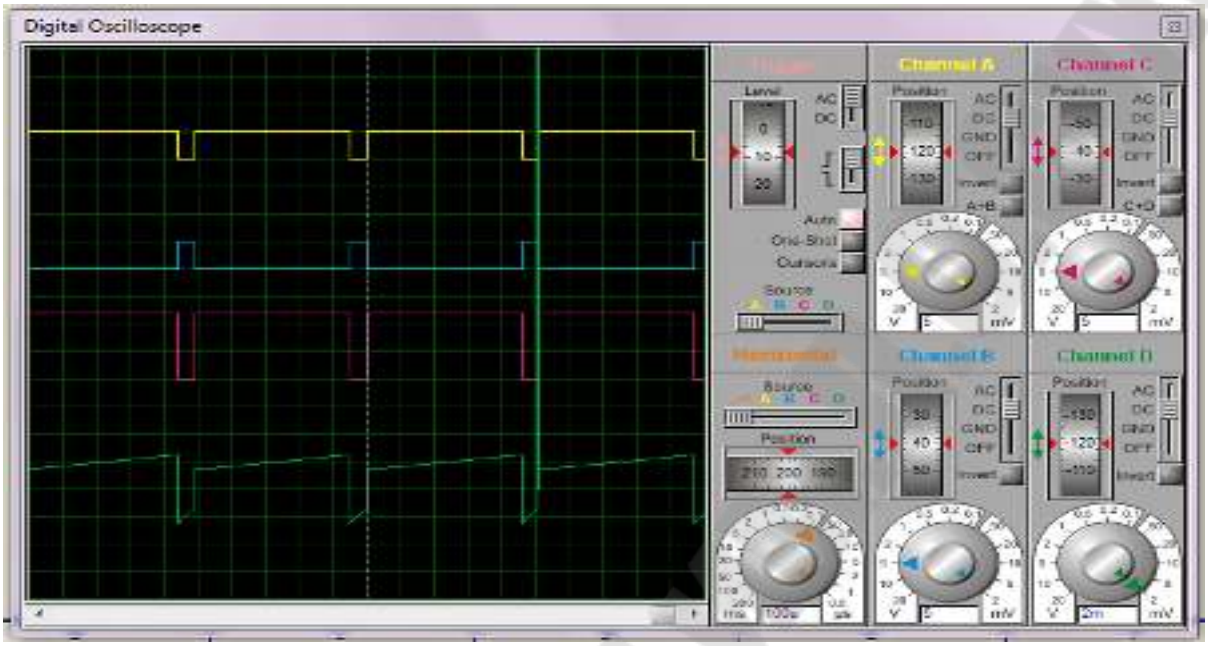


Рис. 5.3. Осциллограммы работы электропривода в Proteus

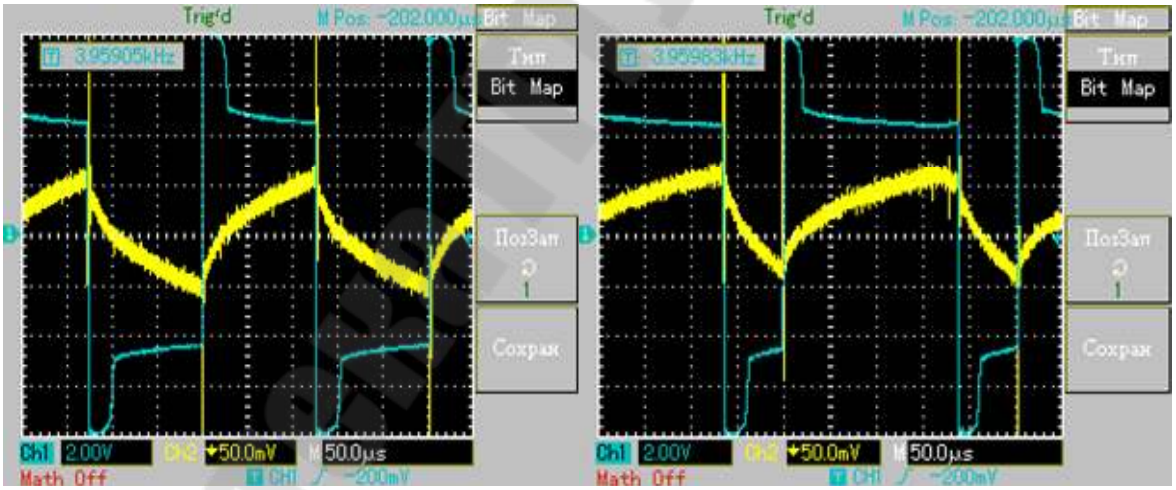


Рис. 5.4. Осциллограммы работы электропривода:
желтый – ток якоря, голубой – напряжение на якоре

5.4. 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.

5.5. 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-счетчика в составе микроконтроллера Atmega?

5. В чем состоит алгоритм усреднения полученного с АЦП результата?

6. Прокомментируйте осциллограммы работы схемы, приведенные на рис. 5.3 и 5.4.

ЛИТЕРАТУРА

1. Евстифеев, А. В. Микроконтроллеры AVR семейства Mega. Руководство пользователя / А. В. Евстифеев. – М. : Додэка-XXI, 2007. – 592 с.

2. Баранов, В. Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы / В. Н. Баранов. – М. : 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: учеб. пособие / А. М. Водовозов. – Вологда : ВоГТУ, 2002. – 123 с.

5. Рюмик, С. М. 1000 и одна микроконтроллерная схема. Кн. 1 / С. М. Рюмик. – М. : Додэка-XXI, 2012. – 356 с.

6. Хартов, В. Я. Микроконтроллеры AVR. Практикум для начинающих : учеб. пособие / В. Я. Хартов. – М. : МГТУ им. Баумана, 2012. – 280 с.

Содержание

Учебное электронное издание комбинированного распространения

Учебное издание

Савельев Вадим Алексеевич
Погуляев Михаил Никифорович

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

Электронный аналог печатного издания

Редактор *Н. Г. Мансурова*
Компьютерная верстка *Н. Б. Козловская*

Подписано в печать 31.01.19.
Формат 60x84/16. Бумага офсетная. Гарнитура «Таймс».
Цифровая печать. Усл. печ. л. 3,02. Уч.-изд. л. 3,16 .
Изд. № 17.
http://www.gstu.by

Издатель и полиграфическое исполнение
Гомельский государственный
технический университет имени П. О. Сухого.
Свидетельство о гос. регистрации в качестве издателя
печатных изданий за № 1/273 от 04.04.2014 г.
пр. Октября, 48, 246746, г. Гомель