Министерство образования Республики Беларусь

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

Кафедра «Автоматизированный электропривод»

В. А. Савельев, И. В. Дорощенко

МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА В АВТОМАТИЗИРОВАННОМ ЭЛЕКТРОПРИВОДЕ

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

Гомель 2021

Библиотека ГГТУ им.П.О.Сухого

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 № 10 от 01.06.2020 г.*)

Рецензент: доц. каф. «Информационные технологии» ГГТУ им. П. О. Сухого
канд. техн. наук, доц. *В. С. Захаренко*

Савельев, В. А.

С12 Микропроцессорные средства в автоматизированном электроприводе : практикум по одноим. дисциплине для студентов специальности 1-53 01 05 «Автоматизированные электроприводы» днев. формы обучения / В. А. Савельев, И. В. Дорощенко. – Гомель : ГГТУ им. П. О. Сухого, 2021. – 46 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 примеры решения типовых задач программирования микроконтроллеров по дисциплине «Микропроцессорные средства в автоматизированном электроприводе», а также задания для самостоятельной работы.

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

© Учреждение образования «Гомельский государственный технический университет имени П. О. Сухого», 2021

Библиотека ГГТУ им. П.О.Сухого

Практическая работа № 1
Подсчёт и вывод на 7-сегментные индикаторы количества внешних событий

1.1. Цель работы

1. Работа по внешним прерываниям.
2. Работа с динамической индикацией.
3. Работа с массивом данных.
4. Работа с макрокомандами.

1.2. Постановка задачи

Для схемы, представленной на рис. 1.1, необходимо написать программу, производящую подсчёт числа нажатий кнопки SB1, подключенной ко входу внешнего прерывания INT0 (линия 2 порта D), в пределах от 0 до 99, с последующим отображением на светодиодных индикаторах методом динамической индикации.

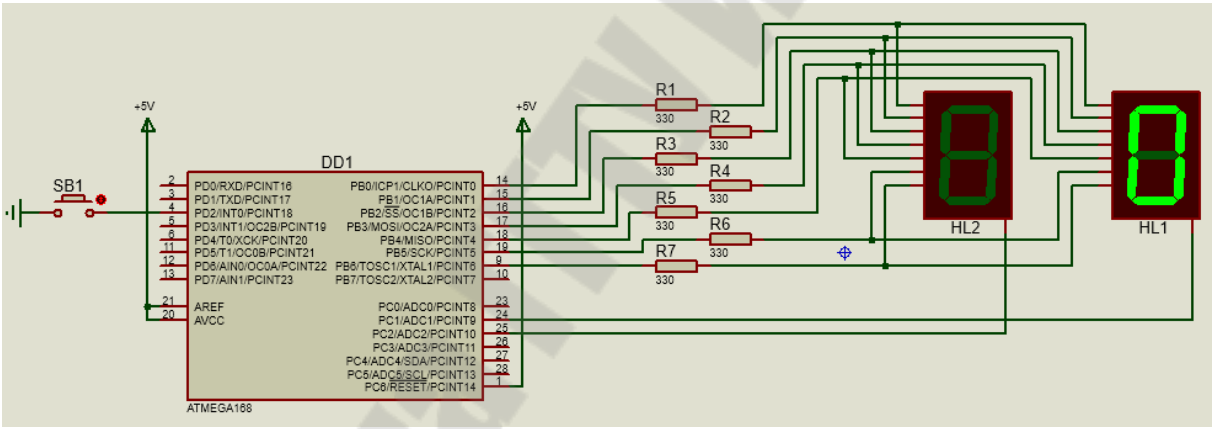


Рис. 1.1. Схема включения индикаторов при динамической индикации

1.3. Пример решения задачи

Для решения поставленной задачи будем использовать внешнее прерывание по входу INT0, к которому и подключен нормально разомкнутый контакт SB1. Когда контакт разомкнут, на вход INT0 будет поступать логическая «1», а когда замкнут – логический «0». Прерывание настроим на срабатывание по переходу от логической «1» к логическому «0», то есть на момент замыкания контакта.

Для правильной работы линии PD2 (INT0) нужно настроить ее на ввод. Для того, чтобы при разомкнутом состоянии контакта на входе INT0 присутствовала стабильная логическая «1», этот вход не-

Библиотека ГГТУ им. П.О.Сухого

обходимо «подтянуть» к шине питания +5В путем включения подтягивающего резистора.

При работе с механическим контактом возникает такая проблема, как дребезг контактов. При нажатии кнопку, перед тем, как контакты плотно соприкоснутся, они будут колебаться (т.е. «дребезжать»), порождая множество срабатываний вместо одного. Аналогичная ситуация возникнет и при размыкании контакта.

Соответственно, микроконтроллер «поймает» все эти нажатия, потому что для микроконтроллера дребезг не отличим от настоящего нажатия на кнопку. Пример осциллограммы, отображающей реальный процесс переключения с логического «0» на логическую «1» приведен на рис. 1.2.

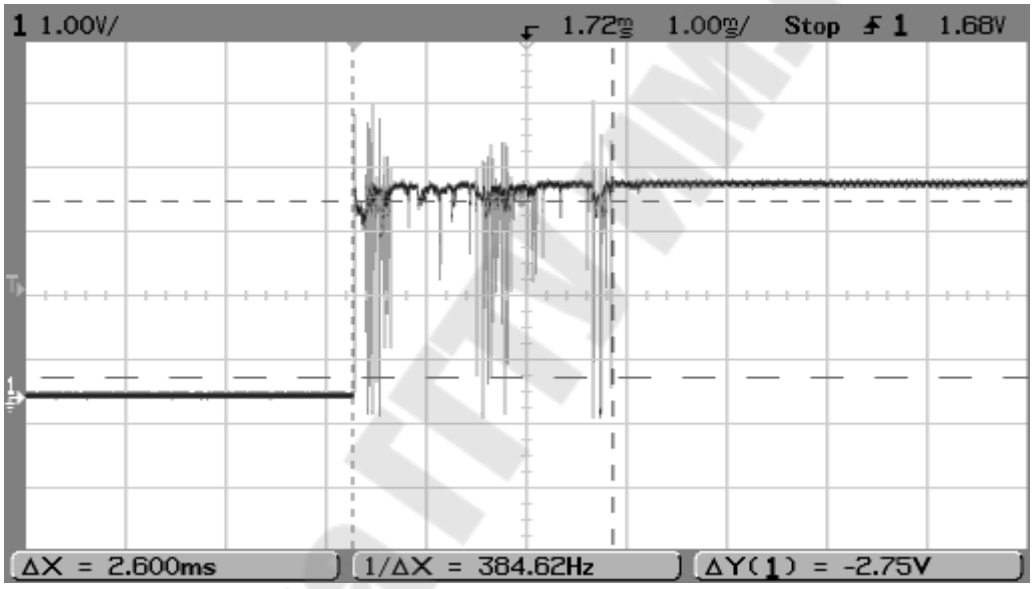


Рис. 1.2. Осциллограмма переключения с «0» на «1»

Суть программного способа, позволяющего избежать дребезга, сводится к следующему. После нажатия кнопки переходим к обработке прерывания, внутри которого сразу же локально запрещаем данный вид прерывания и начинаем проверку состояний линии порта, куда подключена кнопка. Делаем 3…5 проверок с интервалом 0,5…1 мс. Если при каждой проверке будет получен идентичный результат, а именно логический «0», то можно считать, что контакт замкнулся.

Далее выполняем необходимые в прерывании действия.

Перед выходом из прерывания снова производим проверку, но в данном случае уже на логическую «1». Только после этого разрешаем данное прерывание и выходим из прерывания.

Библиотека ГГТУ им. П.О. Сухого

Внутри обработчика прерывания необходимо произвести увеличение числа нажатий кнопки на 1. Проверить, не произошло ли превышение значения 99. Произвести перевод двоичного кода числа нажатий кнопки в двоично-десятичный код.

В основной программе будем выводить информацию (BCD-код) на индикаторы. Как видно из схемы, переключение 7-сегментных индикаторов HL1, HL2 производится через порт C по линиям PC2 и PC1.

Рассмотрим текст программы.

В начале программы выполняем директивы присоединения файла описания выбранной модели контроллера (.include), присвоения удобных имен регистрам общего назначения (.def), присвоения символьному имени числового значения (.set).

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

.def nul = r12                ; присвоили имя регистру r12
.def del1 = r13               ; присвоили имя регистру r13
.def del2 = r14               ; присвоили имя регистру r14
.def del3 = r15               ; присвоили имя регистру r15
.def temp = r16               ; присвоили имя регистру r16
.def cnt = r17                ; присвоили имя регистру r17
.def st_raz = r18             ; присвоили имя регистру r18
.def ml_raz = r19             ; присвоили имя регистру r19
.def schet = r20              ; присвоили имя регистру r20

.set INT0_pin = 2             ; номер линии INT0
;==========================================================================
```

Далее следует секция макрокоманд.

Макрокоманда «uout A, Rd» из РОН в любой РВВ не зависимо от адреса последнего (вместо команд OUT и STS).

Макрокоманда «uin Rd, A» позволяет выполнять пересылку из любого РВВ в РОН, не зависимо от адреса РВВ (вместо команд IN и LDS).

Макрокоманда «usbis A, b» – пропуск следующей команды, если бит I/O(A).b = 1. Макрос работает со всеми РВВ, в то время, как команда sbis работает только с первыми 31-м РВВ.

Макрокоманда «usbic A, b» – пропуск следующей команды, если бит I/O(A).b = 0. Макрос работает со всеми РВВ, в то время, как команда sbic работает только с первыми 31-м РВВ.

Библиотека ГГТУ им. П.О.Сухого

И, наконец, макрокоманда «ldil Rd, k» непосредственной записи в регистры R0-R15. Как известно, младшие 16 регистров изначально не поддерживают непосредственную адресацию.

```
;*************************** 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
;=============================================================================
```

Библиотека ГГТУ им.П.О.Сухого

Теперь указываем сегмент памяти для хранения программы – в нашем случае это сегмент *C* (flash-память).

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

Далее следует секция векторов прерываний. Напомню, что таблица прерываний индивидуальна для каждого контроллера и ее можно найти в инструкции к контроллеру. По сути, нам необходимо только внешнее прерывание INT0, вектор прерывания которого имеет адрес 0x0002 в таблице прерываний. Однако, для нормальной работы программы практически всегда должно быть указано прерывание по сбросу (адрес 0x0000). Оно имеет наивысший приоритет и выполняется сразу после включения контроллера или после нажатия кнопки сброса.

В векторе прерывания прописывается команда безусловного перехода (rjmp) на метку, связанную с расположением обработчика прерывания. В нашем случае, для прерывания по сбросу это метка «reset», а для внешнего прерывания INT0 это метка «int_0». Кстати, назвать метку «int0» не получится, потому что «int0» - это внутренне имя одного из битов регистра EIMSK.

В прерываниях, которые не используются, ставят «заглушки» в виде команд reti.

```
;************************ ВЕКТОРЫ ПРЕРЫВАНИЙ ************************
.org 0
            rjmp reset          ; сброс
.org 2
            rjmp int_0          ; внешний запрос на прерывание 0
.org 4
            reti                ; внешний запрос на прерывание 1
.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
.org 14
            reti                ; совпадение в канале A таймера/счётчика T2
.org 16
            reti                ; совпадение в канале B таймера/счётчика T2
.org 18
            reti                ; переполнение таймера/счётчика T2
```


Библиотека ГГТУ им. П.О.Сухого

```
.org 20
        reti            ; захват фронта таймером T1
.org 22
        reti            ; совпадение в канале A таймера/счётчика T1
.org 24
        reti            ; совпадение в канале B таймера/счётчика T1
.org 26
        reti            ; переполнение таймера/счётчика T1
.org 28
        reti            ; совпадение в канале A таймера/счётчика T0
.org 30
        reti            ; совпадение в канале B таймера/счётчика T0
.org 32
        reti            ; переполнение таймера/счётчика T0
.org 34
        reti            ; передача по SPI завершена
.org 36
        reti            ; приём по USART завершён
.org 38
        reti            ; регистр данных USART пуст
.org 40
        reti            ; передача по USART завершена
.org 42
        reti            ; окончание преобразования АЦП
.org 44
        reti            ; готовность EEPROM
.org 46
        reti            ; аналоговый компаратор
.org 48
        reti            ; двухпроводный последовательный интерфейс
.org 50
        reti            ; готовность SPM
;=====================================================================
```

Далее следует секция инициализации. Секция начинается меткой «reset:». В начале данной секции производим обнуление всех регистров общего назначения (РОН) и всех ячеек оперативной памяти (ОЗУ).

После этого выполняем инициализацию стека и портов ввода/вывода.

При настройке портов линию PD2, к которой присоединена кнопка, настраиваем на ввод и подключаем подтягивающий резистор.

Линии порта B, к которым подключены аноды индикаторов, настраиваем на вывод и подаем в порт B начальное значение, соответствующее отображению на индикаторах «00».

Линии PC2 и PC1, подключенные к общим катодам индикаторов, также настраиваем на вывод.

Библиотека ГГТУ им. П.О. Сухого

В этой же секции программы производим настройку внешнего прерывания. Вначале задаем режим прерывания – по спадающему фронту, а затем разрешаем прерывание INT0 локально.

И, наконец, производим глобальное разрешение прерываний.

```
;*************************** 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
      cbi ddrd, 2                   ; настройка линии PD2 на ввод
      sbi portd, 2                  ; подключение подтягивающего резистора
      ldi temp, 0b11111111          ; настройка порта B на вывод
      uout ddrb, temp
      ldi temp, 1<<PC2|1<<PC1       ; настройка линий PC2 и PC1 на вывод
      uout ddrc, temp
      ldi st_raz, 0b00111111        ; начальный вывод «00» на индикаторы
      ldi ml_raz, 0b00111111
;=====================================================================
; инициализация прерывания INT0
      ldi temp, 1<<ISC01|0<<ISC00   ; настройка срабатывания по спадающему
                                    ; фронту (по замыканию контакта)
      uout eicra, temp
      ldi temp, 1<<INT0             ; локальное разрешение прерывания INT0
      uout eimsk, temp
      sei                           ; глобальное разрешение прерываний
```


Библиотека ГГТУ им. П.О.Сухого

Теперь следует секция основной программы (метка «main:»).

Не смотря на название секции, в ней может вообще ничего не происходить. Но в нашем случае, в основной программе будет производиться вывод информации на индикаторы. Поскольку в программе используется динамическая индикация, алгоритм вывода будет следующий.

1. Выводим в порт B код для старшего разряда.
2. Включаем старший разряд (PC2 = 0).
3. Задаем задержку времени.
4. Выключаем старший разряд (PC2 = 1).
5. Повторяем п.1 - п.4 для младшего разряда.

Контроллер будет всё время работать в основной программе, до того момента, пока не сработает прерывание.

```
;************************ ОСНОВНАЯ ПРОГРАММА ***************************
main:
; индикатор ст. разряда
      out portb, st_raz             ; выводим значение ст. разряда
      ldi temp, 0<<PC2|1<<PC1       ; включаем индикатор ст. разряда
      uout portc, temp
      rcall del_100ms               ; задержка времени
      ldi temp, 1<<PC2|1<<PC1       ; отключаем индикатор ст. разряда
      uout portc, temp
;=======================================================================
; индикатор мл. разряда
      out portb, ml_raz             ; выводим значение мл. разряда
      ldi temp, 1<<PC2|0<<PC1       ; включаем индикатор мл. разряда
      uout portc, temp
      rcall del_100ms               ; задержка времени
      ldi temp, 1<<PC2|1<<PC1       ; отключаем индикатор мл. разряда
      uout portc, temp

      rjmp main                     ; завершаем цикл
;=======================================================================
```

Теперь следует секция обработчиков прерываний. В данной секции присутствует всего один обработчик (метка «int_0:»).

Обработчик вначале сохраняет важную информацию (состояние регистров *temp* и *sreg*) в стек (*push*). Это необходимо в связи с тем, что прерывание может произойти в любой точке основной программы, а внутри прерывания состояние указанных регистров может меняться и меняется. После окончания прерывания в основную программу вернутся совершено другие значения регистров temp и sreg, чем были до прерывания, и ход основной программы будет нарушен.

Библиотека ГГТУ им. П.О.Сухого

Затем происходит локальный запрет прерываний INT0, чтобы дребезг контактов не влиял на работу программы. Далее следует блок защиты от дребезга. Пять раз с интервалом 500 мкс проверяем состояние линии INT0 на равенство нулю. Если дребезга нет, переходим к основной процедуре.

В основной процедуре происходит приращение счётного регистра *cnt*, сравнение его содержимого с максимальным значением и перевод в BCD-код.

Процедура перевода состоит в том, что из значения счётного регистра, которое не превышает 99, вычитают число 10 и подсчитывают количество удавшихся вычитаний. Например, если в счётном регистре будет значение 72_{10}, то число удачных вычитаний будет равно 7. Это количество и составляет значение старшего разряда, и временно (до окончания преобразования) хранится в регистре *cnt*. Остаток же составляет значение младшего разряда.

Однако, вывести BCD-коды разрядов непосредственно на индикаторы не получится. Вначале необходимо используя BCD-коды взять из таблицы соответствующие коды для 7-сегментного индикатора.

Выбор кода из массива производится путем задания адреса в массиве, равного значению метки массива («massiv:») плюс значение BCD-кода соответствующего разряда.

После преобразования возвращаем из стека значение регистра *cnt*.

Завершаем процедуру прерывания проверкой отпускания кнопки, разрешением прерывания INT0 и восстановлением значений регистров *temp* и *sreg*.

```
;********************* ОБРАБОТЧИКИ ПРЕРЫВАНИЙ ************************
int_0:
      push temp                     ; сохраняем temp и SREG в стек
      uin temp, sreg
      push temp
      clr temp                      ; запрещаем прерывание INT0
      uout eimsk, temp
;=======================================================================
; защита от дребезга контактов
      ldi schet, 0x05               ; число проверок нажатия
del_int0:
      rcall del_500us
      usbic pind, INT0_pin          ; если INT0 = 0, продолжаем проверку
      rjmp end_int_0                ; иначе выходим из прерывания
      dec schet                     ; уменьшаем счётчик проверок на 1
      brne del_int0                 ; если счётчик не обнулился,
                                    ; продолжаем проверки
```


Библиотека ГГТУ им. П.О. Сухого

```
; основная процедура
      inc cnt                    ; приращение счётчика нажатий на 1
      cpi cnt, 100               ; сравнение счётчика с числом 100
      brne conv                  ; если счётчик меньше 100, то переходим
                                 ; к переводу счётчика в BCD
      ldi cnt, 0                 ; если счётчик переполнен,
                                 ; то его обнуляем
;=============================================================================
; преобразование двоичного кода в BCD-код
conv:
      push cnt                   ; сохраняем cnt в стек
      clr temp                   ; обнуляем temp
loop:
      subi cnt, 10               ; вычитаем 10 из числа нажатий
      brcs back                  ; если меньше 10, то переходим дальше
      inc temp                   ; переносим 1 в разряд десятков
      rjmp loop                  ; переходим к очередному вычитанию 10-и
back:
      subi cnt, -10              ; возвращаем назад значение счёта,
                                 ; если остаток меньше 10
;=============================================================================
; получаем код младшего разряда ml_raz для LED
      clr nul                    ; сброс nul
      ldi zl, low(massiv*2)      ; инициализация массива BCD-кодов
      ldi zh, high(massiv*2)
      add zl, cnt                ; сложение начального адреса
                                 ; массива с числом cnt
      adc zh, nul
      lpm ml_raz, z              ; загрузка в ml_raz значения из таблицы
;=============================================================================
; получаем код старшего разряда st_raz для LED
      clr nul                    ; сброс nul
      ldi zl, low(massiv*2)      ; инициализация массива BCD-кодов
      ldi zh, high(massiv*2)
      add zl, temp               ; сложение начального адреса массива
                                 ; с числом temp
      adc zh, nul
      lpm st_raz, z              ; загрузка в st_raz значения из таблицы
      pop cnt                    ; восстанавливаем cnt из стека
;=============================================================================
; перед выходом из прерывания INT0 ждем, когда установится "1"
end_int_0:
      usbis pind, INT0_pin       ; если INT0 = 1, продолжаем проверку
      rjmp end_int_0             ; иначе, продолжаем ждать
      ldi schet, 0x0a            ; задаем число проверок размыкания
                                 ; контакта
del2_int0:
      rcall del_500us            ; длительность одной проверки
      usbis pind, INT0_pin       ; если INT0 = 1, продолжаем проверку
      rjmp end_int_0             ; иначе, продолжаем ждать
      dec schet                  ; уменьшаем счётчик проверок на 1
      brne del2_int0             ; если счётчик не обнулился,
                                 ; продолжаем проверки
```


Библиотека ГГТУ им. П.О.Сухого

```
      ldi temp, 1<<INT0            ; разрешение прерывания INT0
      uout eimsk, temp
      pop temp                     ; восстанавливаем temp и SREG из стека
      uout sreg, temp
      pop temp
      reti                         ; возврат из прерывания
;=====================================================================
```

При размещении элементов в массиве необходимо помнить, что в строке должно быть чётное количество элементов (кроме последней строки), либо все элементы должны быть в одной строке.

```
;***************************** МАССИВЫ *********************************
; массив кодов десятичных чисел
massiv:
.db 0b00111111, 0b00000110         ; код 0,1
.db 0b01011011, 0b01001111         ; код 2,3
.db 0b01100110, 0b01101101         ; код 4,5
.db 0b01111101, 0b00000111         ; код 6,7
.db 0b01111111, 0b01101111         ; код 8,9
;=====================================================================
```

Секция подпрограмм задержки содержит две подпрограммы, построенные на принципе циклического уменьшения константы на 1. При этом константа задается в регистрах del1, del2, del3. Ее величина определяет длительность задержки.

```
;***************************** ЗАДЕРЖКИ *********************************
; подпрограмма задержки на 500 мкс
del_500us:
            ldil del1, 51
            ldil del2, 6
loop_1:
            dec del1
            brne loop_1
            dec del2
            brne loop_1
            ret
;=====================================================================
; подпрограмма задержки на 100 мс
del_100ms:
            ldil del1, 245
            ldil del2, 15
            ldil del3, 5
loop_2:
            dec del1
            brne loop_2
            dec del2
            brne loop_2
```


Библиотека ГГТУ им. П.О. Сухого

```
            dec del3
            brne loop_2
            ret
;=====================================================================
```

Расчёт значений констант указанных регистров основан на величине необходимой задержки, значении тактовой частоты и числа тактов, необходимых для выполнения каждой команды подпрограммы. Кроме того, расчёт точных значений может быть выполнен при помощи утилиты «AVR Delay» для Androuid.

1.4. Задание для самостоятельной работы

1. Модифицируйте рассмотренную выше программу для вывода на индикаторы трёхразрядного числа.
2. Дополните схему вторым кнопочным выключателем, подключенным ко входу INT1 и выполняющим функцию сброса информации, отображаемой на индикаторах.

Практическая работа № 2
Передача байта данных по интерфейсу SPI

2.1. Цель работы

1. Работа с интерфейсом SPI.
2. Программирование таймера/счётчика.
3. Программирование аналого-цифрового преобразователя.
4. Работа с таблицей данных.

2.2. Постановка задачи

Для схемы соединения двух микроконтроллеров, приведенной на рис. 2.1, написать управляющую программу, которая позволит передавать информацию о состоянии переключателей SB1 на светодиодный индикатор VD1.

2.3. Пример решения задачи

Для решения поставленной задачи необходимо настроить один микроконтроллеров в режим передатчика (master), а второй – в режим приемника (slave). Передатчик будет осуществлять считывание состояния подключенных к порту D переключателей, и формировать последовательный двоичный код, который будет передан приемнику.

Библиотека ГГТУ им. П.О. Сухого

Приемник, в свою очередь, будет выполнять приём и дешифрацию полученного кода, и формировать параллельный двоичный код, который будучи отправлен в порт D приемника, приведет к включению светодиодов индикатора VD1.

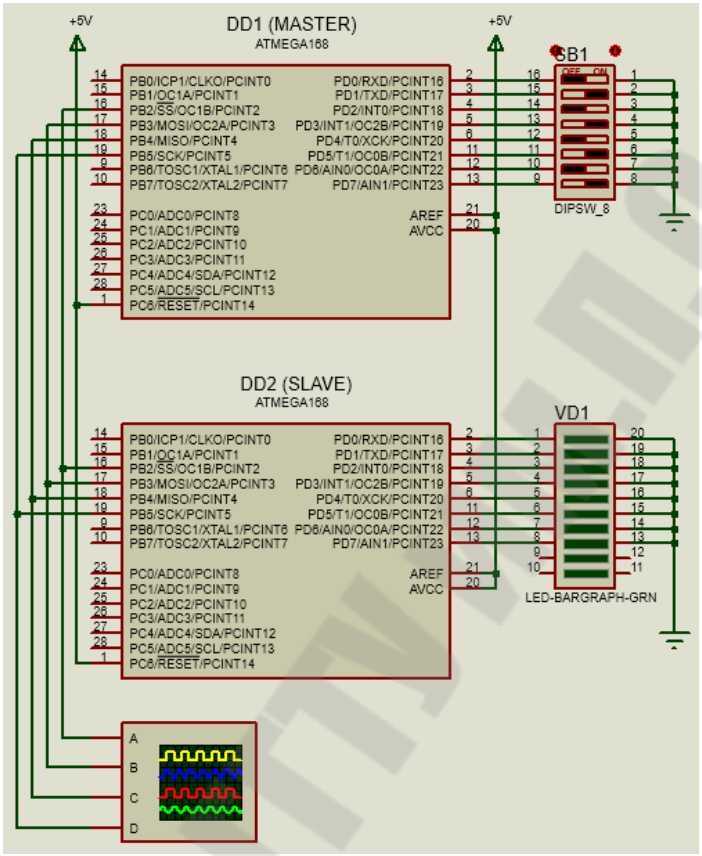


Рис. 2.1. Схема соединения контроллеров по интерфейсу SPI

Важно помнить, что передачу информации можно контролировать двумя способами. Во-первых, через прерывания по завершению передачи через SPI (адрес 34), а во-вторых, путем контроля флага SPIF окончания передачи через SPI, который расположен в регистре SPRS. В данном примере воспользуемся вторым способом.

Рассмотрим текст программы для передающего устройства.

В начале программы выполняем директивы присоединения файла описания выбранной модели контроллера (.include) и присвоения удобных имен регистрам общего назначения (.def).

```
;************************ ЗАДАНИЕ ПАРАМЕТРОВ ************************
.include "m168def.inc"        ; присоединение файла описания ATmega168
.def temp = r16               ; временный регистр
.def del = r17                ; регистр задержки
```


Библиотека ГГТУ им. П.О.Сухого

Далее следует секция макрокоманд.

```
;*************************** МАКРОКОМАНДЫ ****************************
; Макрокоманда "uout A,Rd" пересылки из РОН в любой РВВ
; не зависимо от адреса РВВ (вместо команд OUT и STS)
.macro uout
      .if @0 < 0x40
            out @0, @1
      .else
            sts @0, @1
      .endif
.endm
;=========================================================================

; Макрокоманда "uin Rd,A" пересылки из любого РВВ в РОН
; не зависимо от адреса РВВ (вместо команд IN и LDS)
.macro uin
      .if @1 < 0x40
            in @0, @1
      .else
            lds @0, @1
      .endif
.endm
;=========================================================================
```

Указываем сегмент памяти для хранения программы.

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

Далее следует секция векторов прерываний. В данной секции активен только один вектор (reset), однако при необходимости контролировать передачу информации через прерывания можно активировать ещё и вектор прерывания по завершению передачи SPI.

```
;************************ 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 (PCINT0)
.org 8
      reti        ; прерывание 1 по изменению состояния выводов (PCINT1)
.org 10
      reti        ; прерывание 2 по изменению состояния выводов (PCINT2)
.org 12
```


Библиотека ГГТУ им.П.О.Сухого

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

Далее следует секция инициализации. Кроме обнуления регистров и памяти здесь выполняется настройка порта D передатчика на прием сигналов и к нему подключаются подтягивающие резисторы. Далее конфигурируются четыре линии порта B, задействованные в работе интерфейса SPI.

И, наконец, настраиваем сам интерфейс SPI: включаем SPI, запрещаем прерывания по SPI, устанавливаем порядок передачи дан-

Библиотека ГГТУ им.П.О.Сухого

ных, делаем контроллер ведущим, задаем полярность и фазу тактового сигнала, а также скорость передачи данных.

При этом порядок передачи, полярность и фаза тактового сигнала особого значения не имеют. Необходимо лишь, чтобы эти настройки совпадали у передатчика и приемника.

```
;*************************** 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
; установка порта D на ввод с включенными подтягивающими резисторами
      ldi temp, 0b11111111
      uout portd, temp
      ldi temp, 1<<PB5|1<<PB3|1<<PB2
      uout ddrb, temp         ; MOSI, SCK и SS на вывод
      sbi portb, 2            ; устанавливаем высокий уровень на SS
      sbi portb, 4            ; включаем подтяг. резистор на MISO
;=====================================================================
; инициализация SPI
      ldi temp, 0b01010011
; SPIE    0b[0]xxxxxxx - разрешение прерывания от SPI
; SPE     0bx[1]xxxxxx - включение SPI
; DORD    0bxx[0]xxxxx - порядок передачи данных
; MSTR    0bxxx[1]xxxx - ведущий или ведомый (ведущий)
; CPOL    0bxxxx[0]xxx - полярность тактового сигнала (положительная)
; CPHA    0bxxxxx[0]xx - фаза тактового сигнала (по нараст. фронту)
```


Библиотека ГГТУ им. П.О.Сухого

```
; SPR1:0  0bxxxxxx[11] - скорость передачи (fclk/4)
      uout spcr, temp
;sei                          ; разрешение прерываний
```

Основная программа посредством порта D осуществляет приём информации в параллельном двоичном коде с блока микропереключателей SB1. Далее следует отправка полученного байта информации по последовательному каналу, начинающаяся установкой низкого уровня на линии SS контроллера. При этом процесс передачи контролируется не по прерываниям, а путём проверки флага окончания отправки – 7-го бита регистра SPSR. Как только флаг окончания передачи будет установлен, на линии SS необходимо установить высокий уровень. Далее процедура отправки пакета может быть повторена.

```
;********************** ОСНОВНАЯ ПРОГРАММА ****************************
main:
      uin temp, pind          ; чтение порта D
      cbi portb, 2            ; устанавливаем низкий уровень на SS
      uout spdr, temp         ; отправляем 8 бит
; ожидаем окончания отправки
wait:
      uin temp, spsr          ; проверяем флаг окончания отправки
      sbrs temp, 7
      rjmp wait
      sbi portb, 2            ; устанавливаем высокий уровень на SS
; пауза между пакетами
      rcall del_50us          ; пауза между пакетами
      rjmp main               ; переходим к отправке нового пакета

;***************************** ЗАДЕРЖКИ *******************************
; подпрограмма задержки на 50 мкс
del_50us:
      ldi del, 134
loop:
      dec del
      brne loop
      nop
      ret
```

2.4. Задание для самостоятельной работы

1. Напишите программу для контролера-приёмника.

2. Напишите эту же программу, используя систему прерываний для контроля передачи информации.

3. Используйте АЦП для задания передаваемой величины и ЦАП для вывода полученного результата на принимающей стороне.

Библиотека ГГТУ им. П.О. Сухого

Практическая работа № 3
Управление униполярным шаговым двигателем

3.1. Цель работы

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

3.2. Постановка задачи

Для схемы управления униполярным шаговым двигателем (ШД), приведенной на рис. 3.1, написать управляющую программу, которая позволит регулировать частоту «шагания» двигателя с помощью потенциометра, подключенного ко входу аналого-цифрового преобразователя (АЦП) микроконтроллера.

3.3. Пример решения задачи

Анализируя схему, приведенную на рис. 3.1, 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».

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.

Библиотека ГГТУ им. П.О.Сухого

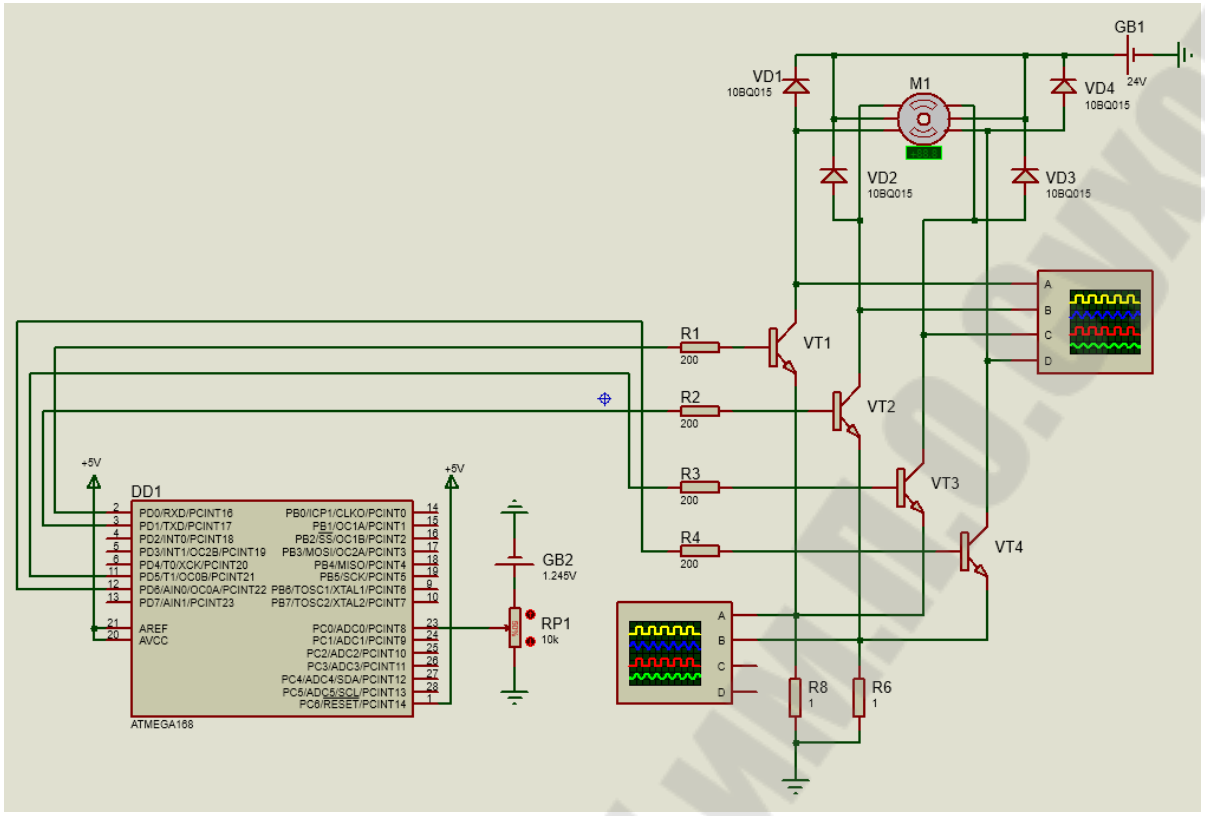


Рис. 3.1. Схема управления униполярным шаговым двигателем

```
;************************** 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
```


Библиотека ГГТУ им. П.О.Сухого

```
;=====================================================================
.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
```


Библиотека ГГТУ им.П.О.Сухого

```
.org 12
        reti            ; тайм-аут сторожевого таймера (WatchDog)
.org 14
        rjmp timer2_int ; 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:
```


Библиотека ГГТУ им.П.О.Сухого

```
; 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.

Библиотека ГГТУ им.П.О.Сухого

```
; 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:
```


Библиотека ГГТУ им.П.О.Сухого

```
      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, сбрасываем 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 полуоб-

Библиотека ГГТУ им. П.О. Сухого

мотки) 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
```

3.4. Задание для самостоятельной работы

1. Напишите программу для управления биполярным шаговым двигателем с использованием драйвера L298.

2. Напишите программу, позволяющую сделать двигателю заданное число шагов в одном из направлений, а затем столько же в противоположном.

3. Напишите программу, в которой шаговый двигатель будет совершать один шаг при каждом замыкании кнопочного выключателя, соединенного со входом INT0.

Практическая работа № 4
Получение периодической функции при помощи таблицы

4.1. Цель работы

1. Приобрести навыки формирования периодических функций путем задания их в виде таблиц.

2. Приобрести навыки программирования и применения 8-битного таймера для отсчёта интервалов времени.

4.2. Постановка задачи

Для схемы, приведенной на рис. 4.1, написать управляющую программу, которая позволит получить на выходе цифро-аналогового преобразователя (ЦАП) периодическую функцию, частоту которой можно регулировать с помощью потенциометра, подключенного ко входу аналого-цифрового преобразователя (АЦП) микроконтроллера.

Библиотека ГГТУ им. П.О.Сухого

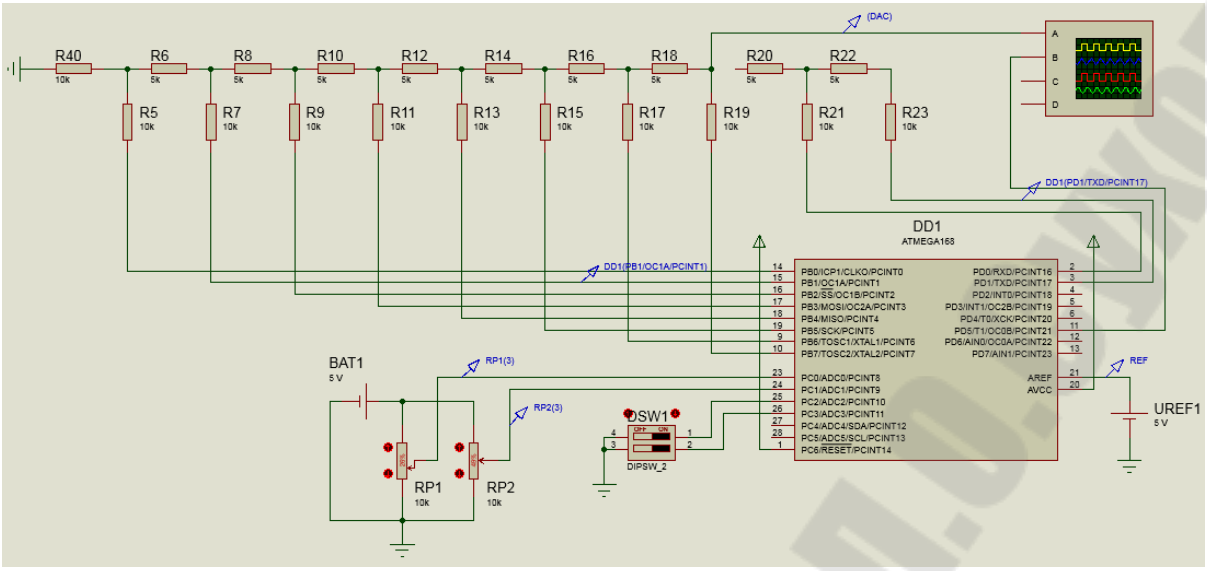


Рис. 4.1. Схема генератора периодической функции

4.3. Пример решения задачи

Формирование периодической функции рассмотрим на примере синусоидальной функции вида $y(t) = A \cdot \sin(щ \cdot t)$, где A – амплитудное значение функции, ω – циклическая частота.

Для задания функции в виде таблицы необходимо заранее рассчитать значения этой функции при различных значениях переменной t. Чем больше точек будет задано, тем более «гладкой» получится результирующая функция. В данном примере я ограничусь числом точек, равным 36.

Учитывая то, что на выходе ЦАП будет получено напряжение синусоидальной формы, но постоянное по знаку, необходимо предварительно сместить синусоиду относительно горизонтальной оси координат, как показано на рис. 4.2.

Таким образом, не трудно рассчитать 36 значений на интервале от 0 до 2π, а именно: 0,5; 0,578; 0,654; 0,725; 0,789; 0,845; 0,89; 0,923; 0,943; 0,95; 0,943; 0,923; 0,89; 0,845; 0,789; 0,725; 0,654; 0,578; 0,5; 0,422; 0,346; 0,275; 0,211; 0,155; 0,11; 0,077; 0,057; 0,05; 0,057; 0,077; 0,11; 0,155; 0,211; 0,275; 0,346; 0,422.

Алгоритм работы программы следующий. В регистр Z заносим адрес первого элемента массива синуса, а в счётный регистр – нуль. В каждом цикле из массива извлекается очередное значение функции и пересылается в порт B, к которому подключен ЦАП. Далее содержимое регистра Z (адрес очередного элемента массива) увеличивается на

Библиотека ГГТУ им. П.О. Сухого

1. После чего запускается задержка времени. Величина задержки времени определяет частоту генерируемой функции. Затем содержимое счётчика также увеличивается на 1. Когда содержимое счётчика достигнет 36-и, его значение сбрасывается, при этом восстанавливается значение регистра Z.

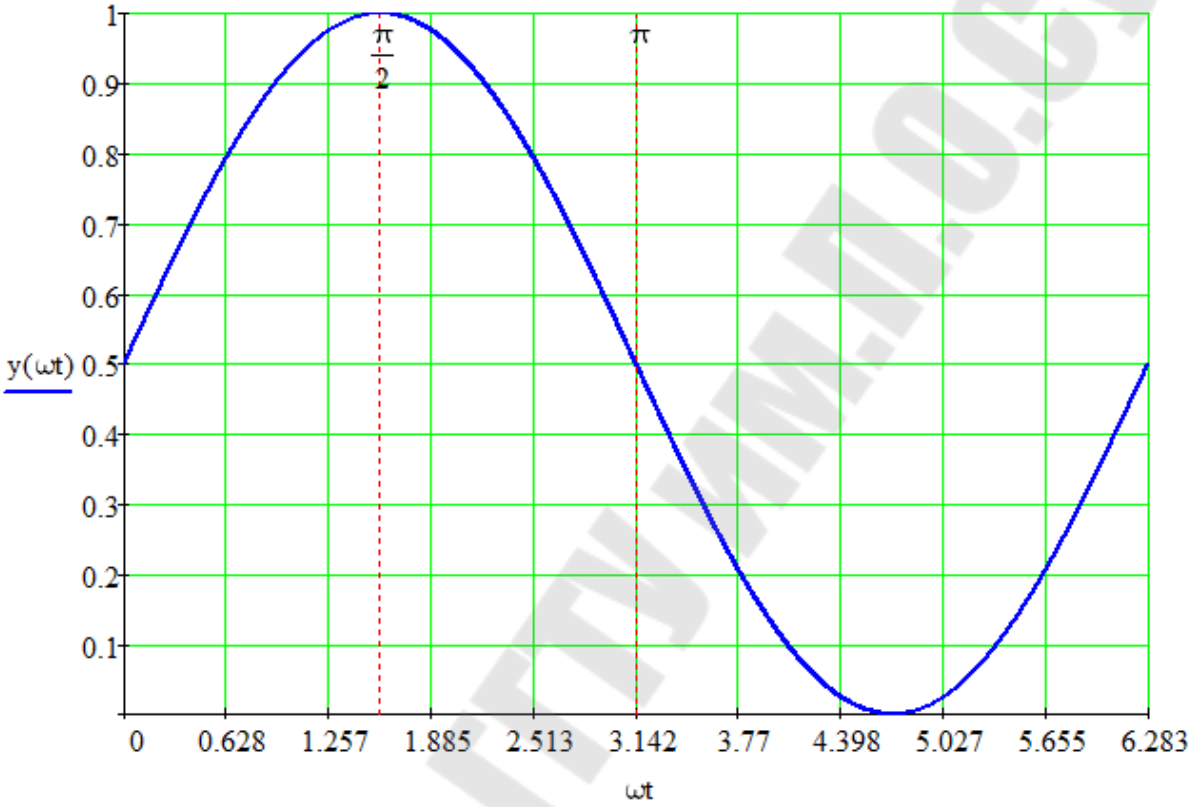


Рис. 4.2. Целевая функция

```
;=====================================================================
; Генератор синусоиды с программной задержкой.
; Контроллер ATmega168.
; Источник опорного напряжения AREF.
; Тактовая контроллера 8 МГц.

.include"m168def.inc"   ; присоединение файла описания ATmega168
.def temp = r16         ; временный регистр
.def cnt = r17          ; счётчик элементов массива

;=====================================================================
; Макрокоманда задержки
.macro delay
      push temp         ; сохраняем в стеке temp, yl, yh
      push yl
      push yh
      ldi temp, @1      ; указываем число повторов счёта
m1:   ldi yl, low(@0)   ; указываем константу счёта (2 байта)
```

Библиотека ГГТУ им. П.О.Сухого

```
      ldi yh, high(@0)
m2:   sbiw yl, 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,
                          ; переходим на метку m1
      pop yh              ; восстанавливаем из стека xh, xl, temp
      pop yl
      pop temp
.endm

; Макрокоманда "UOUT @0, @1" пересылки из РОН в любой РВВ
; не зависимо от адреса РВВ (вместо команд OUT и STS)
.macro uout
      .if @0 < 0x40
            out @0, @1
      .else
            sts @0, @1
      .endif
.endm

; Макрокоманда "UIN @0, @1" пересылки из любого РВВ в РОН
; не зависимо от адреса РВВ (вместо команд IN и LDS)
.macro uin
      .if @1 < 0x40
            in @0, @1
      .else
            lds @0, @1
      .endif
.endm
.cseg                     ; выбор сегмента памяти для хранения программы

;=====================================================================
; векторы прерываний
.org 0
; инициализация стека
      ldi temp, high(ramend)
      uout sph, temp
      ldi temp, low(ramend)
      uout spl, temp
; начальная настройка портов
      ldi temp, 0b11111111
      uout ddrb, temp         ; установка порта B на вывод

;=====================================================================
; основная программа
main:
      ldi zl, low(sinus*2)    ; загрузка адреса таблицы синусов
      ldi zh, high(sinus*2)
      clr cnt                 ; сброс счётчика
metka:
      lpm temp, z+            ; чтение очередного элемента таблицы
```


Библиотека ГГТУ им.П.О.Сухого

```
        out portb, temp         ; и отправка на ЦАП
        delay 0x00ff, 1         ; запуск задержки
        inc cnt                 ; увеличение счётчика
        cpi cnt, 36             ; сравнение счётчика с числом 36
        brne metka              ; повтор цикла, если в счётчике не 36
        rjmp main               ; переход к следующему периоду

;=====================================================================
; таблица синусов
sinus:
        .db 0.5*256, 0.578*256, 0.654*256, 0.725*256
        .db 0.789*256, 0.845*256, 0.89*256, 0.923*256
        .db 0.943*256, 0.95*256, 0.943*256, 0.923*256
        .db 0.89*256, 0.845*256, 0.789*256, 0.725*256
        .db 0.654*256, 0.578*256, 0.5*256, 0.422*256
        .db 0.346*256, 0.275*256, 0.211*256, 0.155*256
        .db 0.11*256, 0.077*256, 0.057*256, 0.05*256
        .db 0.057*256, 0.077*256, 0.11*256, 0.155*256
        .db 0.211*256, 0.275*256, 0.346*256, 0.422*256
;=====================================================================
```

Как видно из рис. 4.3, получена синусоида с размахом несколько меньше 5 В и с частотой примерно 220 Гц.

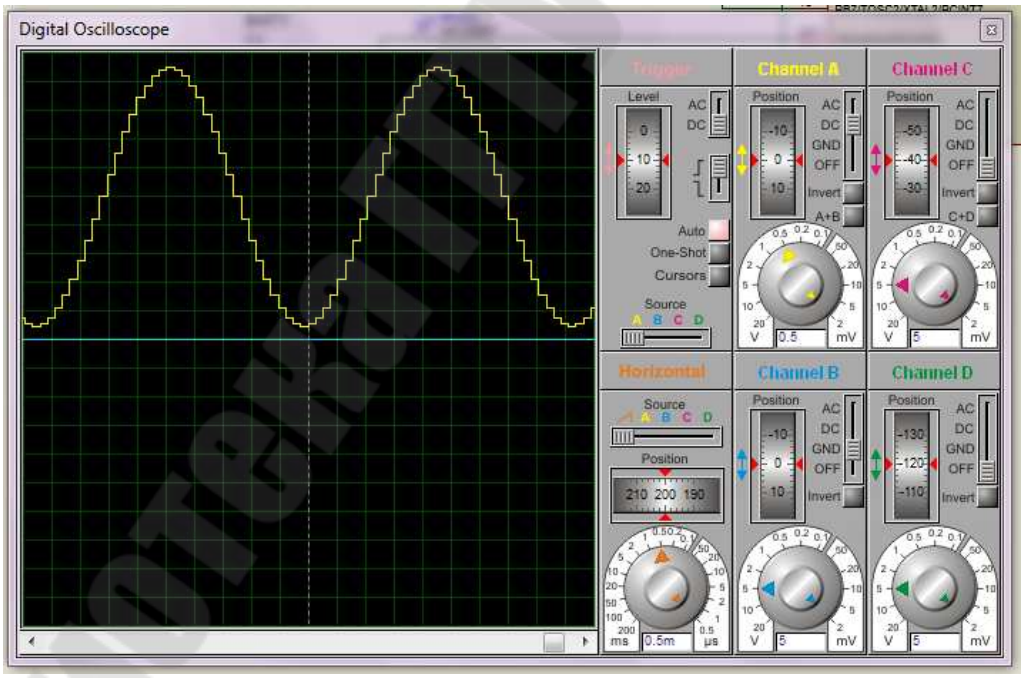


Рис. 4.3. Осциллограмма полученного синусоидального напряжения

Недостатком рассмотренного метода является то, что, во-первых, ресурс центрального процессора направлен на формирование

Библиотека ГГТУ им. П.О. Сухого

временной задержки, во-вторых, нет возможности регулировать частоту синусоиды аппаратно.

Рассмотрим альтернативный метод формирования синусоиды с использованием АЦП и 8-битного таймера.

Используем таймер T0, который будет работать в режиме сброса по совпадению (CTC). Значение регистра сравнения OCR0A будет определять момент сброса таймера, а, следовательно, и частоту прерываний от таймера T0. Программа обработки прерываний от таймера будет выполнять функцию, которую выполняла основная программа, рассмотренная выше.

Для управления частотой синусоиды используем АЦП, результат преобразования (младший байт, т.к. таймер 8-битный) которого используем для задания значения регистра сравнения OCR0A. АЦП будет работать в автоматическом режиме не зависимо от таймера.

```
;=====================================================================
; Генератор синусоиды с использованием таймера T0 и АЦП.
; Контроллер ATmega168.
; Источник опорного напряжения AREF.
; Тактовая частота контроллера 8 МГц.

.include"m168def.inc"   ; присоединение файла описания ATmega168
.def temp = r16         ; временный регистр
.def cnt = r17          ; счётчик элементов массива

;=====================================================================
; Макрокоманда "UOUT @0, @1" пересылки из РОН в любой РВВ
; не зависимо от адреса РВВ (вместо команд OUT и STS)
.macro uout
      .if @0 < 0x40
            out @0, @1
      .else
            sts @0, @1
      .endif
.endm

; Макрокоманда "UIN @0, @1" пересылки из любого РВВ в РОН
; не зависимо от адреса РВВ (вместо команд IN и LDS)
.macro uin
      .if @1 < 0x40
            in @0, @1
      .else
            lds @0, @1
      .endif
.endm
;=====================================================================
```


Библиотека ГГТУ им. П.О.Сухого

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

;=====================================================================
; векторы прерываний
.org 0
      rjmp reset        ; переход на обработку сброса
...

.org 30
      rjmp timer_0      ; совпадение в канале B таймера T0
...

.org 42
      rjmp adc_int      ; окончание преобразования АЦП
...

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

; начальная настройка портов
      ldi temp, 0b11111111
      uout ddrb, temp         ; установка порта B на вывод
      uout ddrd, temp         ; установка порта D на вывод

      ldi zl, low(sinus*2)    ; загрузка адреса таблицы синусов
      ldi zh, high(sinus*2)
      clr cnt                 ; сброс счётчика

;=====================================================================
; инициализация таймера T0
      ldi temp, 0b11111111    ; начальная установка частоты
      uout ocr0a, temp        ; !!! без загрузки ocr0a не работает
                              ; режим CTC
      ldi temp, 0b00010010
; COM0A1:0 0b[00]xxxxxx - режим формирования выходного сигнала
;                         OC0A (отключен)
; COM0B1:0 0bxx[01]xxxx - режим формирования выходного сигнала
;                        OC0B (переключение на противоположное)
; -        0bxxxx[00]xx - зарезервирован
; WGM01:0  0bxxxxxx[10] - режим работы таймера (CTC)
      uout tccr0a, temp
      ldi temp, 0b00000100
; FOC0A  0b[0]xxxxxxx - принудительное изменение состояния OC0A
; FOC0B  0bx[0]xxxxxx - принудительное изменение состояния OC0B
; -      0bxx[00]xxxx - зарезервированы
; WGM02  0bxxxx[0]xxx - режим работы таймера (CTC)
; CS02:0 0bxxxxx[100] - выбор тактового источника с предделителем
;                           частоты 8МГц/256=31250 кГц
```


Библиотека ГГТУ им.П.О.Сухого

```
      uout tccr0b, temp
      ldi temp, 0b00000100    ; разрешение прерываний при
      uout timsk0, temp       ; совпадении в канале B

;=====================================================================
; 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
      uout admux, temp
      ldi temp, 0b11101110
; 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 8МГц/64=125 кГц
      uout adcsra, temp
      sei                     ; разрешение прерываний

;=====================================================================
; основная программа
main:
      rjmp main               ; пустой цикл

;=====================================================================
; программа обработки прерывания от таймера T0
timer_0:
      lpm temp, z+            ; чтение очередного элемента таблицы
      out portb, temp         ; и отправка на ЦАП
      inc cnt                 ; увеличение счётчика
      cpi cnt, 36             ; сравнение счётчика с числом 36
      brne endi               ; окончание прерывания, если в
                              ; счётчике не 36
      ldi zl, low(sinus*2)    ; загрузка адреса таблицы синусов
      ldi zh, high(sinus*2)
      clr cnt                 ; сброс счётчика
endi:
      reti

;=====================================================================
; программа обработки прерываний от АЦП
adc_int:
      uin r18, adcl           ; чтение мл.байта АЦП в регистр R20
      uin r19, adch           ; чтение ст.байта АЦП в регистр R21
      uout ocr0a, r18         ; запись мл.байта АЦП в регистр OCR0A
      reti

;=====================================================================
; таблица синусов
sinus:
```


Библиотека ГГТУ им. П.О.Сухого

```
        .db 0.5*256, 0.578*256, 0.654*256, 0.725*256
        .db 0.789*256, 0.845*256, 0.89*256, 0.923*256
        .db 0.943*256, 0.95*256, 0.943*256, 0.923*256
        .db 0.89*256, 0.845*256, 0.789*256, 0.725*256
        .db 0.654*256, 0.578*256, 0.5*256, 0.422*256
        .db 0.346*256, 0.275*256, 0.211*256, 0.155*256
        .db 0.11*256, 0.077*256, 0.057*256, 0.05*256
        .db 0.057*256, 0.077*256, 0.11*256, 0.155*256
        .db 0.211*256, 0.275*256, 0.346*256, 0.422*256
;=====================================================================
```

Рассчитаем частоту синусоиды при напряжении на входе канала 0 АЦП, равном 0,622 В. При таком напряжении двоичный код на выходе АЦП (младший байт) будет равен

$$\mathrm{ADC} = \frac{V_{\text{in}}}{V_{\text{ref}}} \cdot N = \frac{0{,}622}{5} \cdot 1024 = 127_{10} = 01111111_2 .$$

Это число, будучи записанным в регистр OCR0A, определит частоту синусоиды.

За один период синусоиды 8-битный таймер, генерирует 36 задержек. Время каждой задержки зависит от числа, записанного в регистр сравнения и от выбранного предделителя тактовой частоты. В рассмотренном примере в регистр OCR0A записано число 127, а предделитель равен 256. Таким образом, имеем частота прерываний таймера составит

$$f_{\text{int}} = \frac{f_{\text{т}}}{k_{\text{дел}} \cdot (\mathrm{ADC}+1)} = \frac{8 \cdot 10^6}{256 \cdot (127+1)} = 244 \text{ Гц}.$$

Частота синусоиды составит

$$f_{\sin} = \frac{f_{\text{int}}}{n_{\sin}} = \frac{244}{36} = 6{,}8 \text{ Гц}.$$

Как видно из рис. 4.4, получена синусоида с размахом несколько меньше 5 В и с частотой примерно

$$f_{\sin} = \frac{1}{14{,}7 \text{кл.} \cdot 10 \frac{\text{мс}}{\text{кл.}}} = 6{,}8 \text{ Гц}.$$

4.4. Задание для самостоятельной работы

1. Напишите программу получения треугольной функции.

2. Напишите программу, где с помощью переключателя SW1 можно выбирать тип генерируемой периодической функции.

Библиотека ГГТУ им. П.О. Сухого

3. Напишите программу, в которой можно менять частоту и амплитуду периодической функции с помощью двух потенциометров RP1 и RP2.

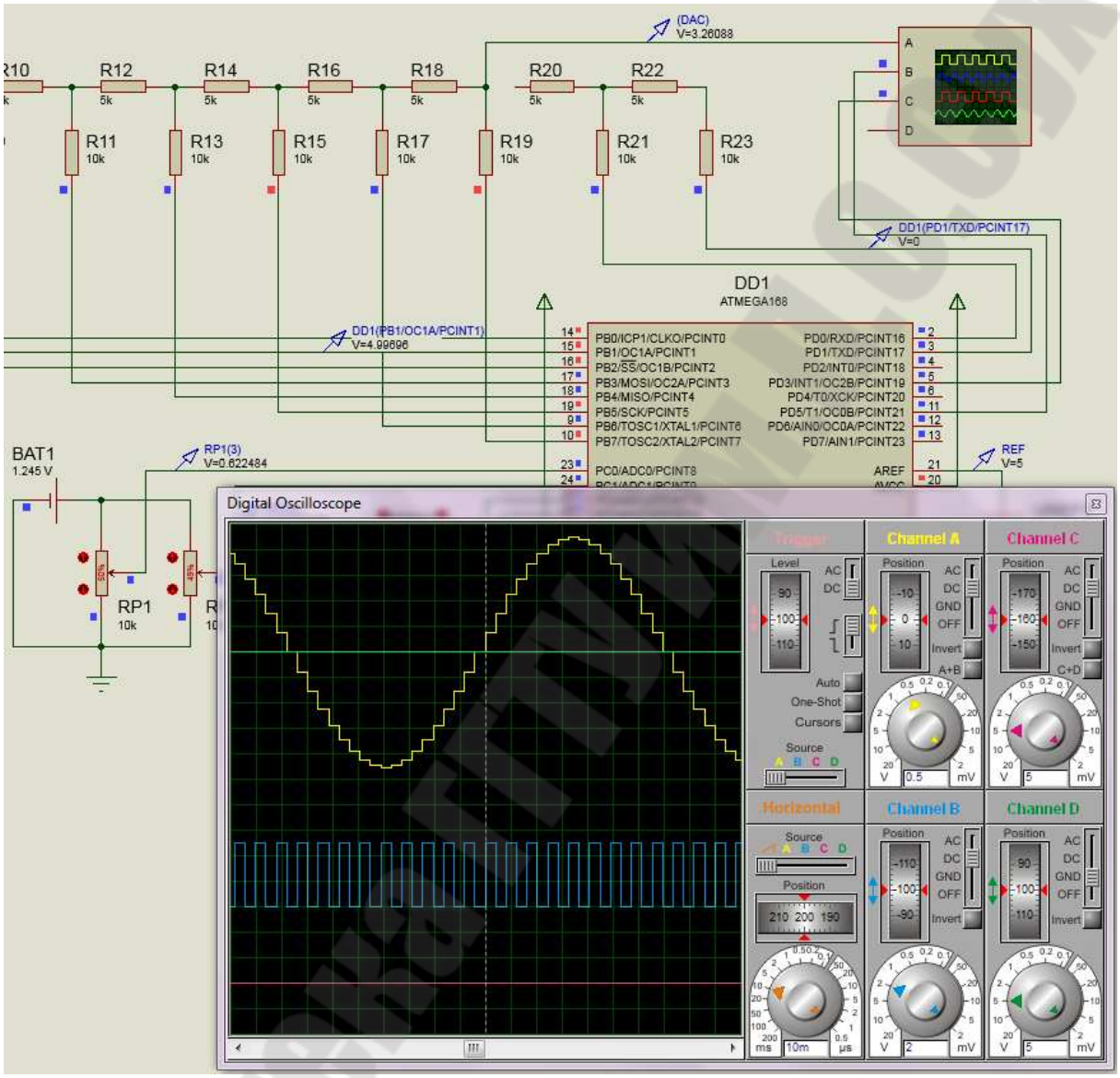


Рис. 4.4. Осциллограмма полученного синусоидального напряжения

Практическая работа № 5
Программирование USART

5.1. Цель работы

1. Работа с интерфейсом USART.
2. Программирование аналого-цифрового преобразователя.

Библиотека ГГТУ им. П.О. Сухого

5.2. Постановка задачи

Для схемы соединения двух микроконтроллеров, приведенной на рис. 5.1, написать управляющую программу, которая позволит передавать информацию о состоянии потенциометра RP1 и восстановить аналоговое значение на выходе АЦП.

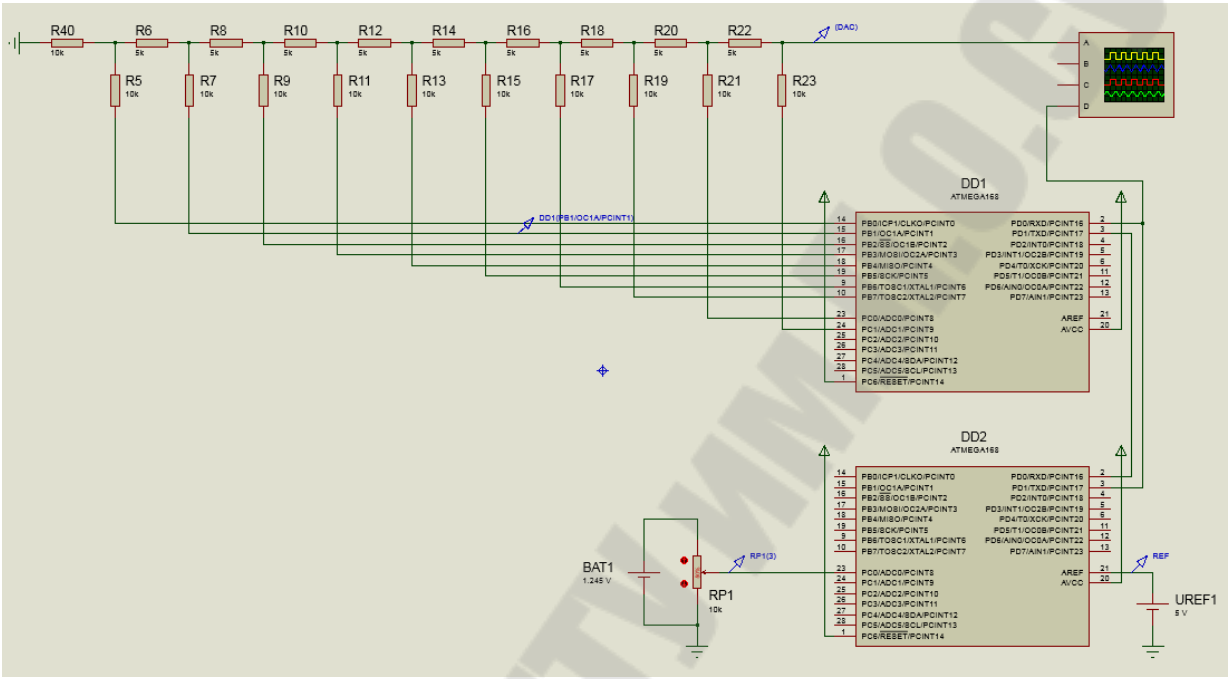


Рис. 5.1. Схема включения микроконтроллеров, связанных через USART

5.3. Пример решения задачи

Ранее была рассмотрена задача преобразования аналогового напряжения в двоичный код при помощи встроенного в микроконтроллер АЦП, с последующим восстановлением аналогового напряжения с помощью внешнего ЦАП, выполненного по схеме R-2R.

Теперь усложним задачу, разделив функции ФЦП и ЦАП между двумя микроконтроллерами. Таким образом, возникает проблема передачи двоичного кода из одного контроллера в другой. Для решения этой проблемы воспользуемся встроенным в микроконтроллер универсальным синхронно-асинхронным приёмо-передатчиком (USART).

Библиотека ГГТУ им. П.О. Сухого

Как видно из рис. 5.1, микроконтроллер DD1 выполняет функцию ЦАП, а микроконтроллер DD2 выполняет функцию АЦП. Для связи контроллеров при помощи последовательного интерфейса необходимо соединить выход передатчика USART DD2 (вывод 3 – *TxD*) со входом приемника USART DD1 (вывод 2 – *RxD*) и наоборот – вывод 3 DD1 соединить с выводом 2 DD2.

Для «оживления» схемы придётся написать две программы: одну для контроллера DD1, вторую – для DD2.

Программа для контроллера DD1 работает через прерывание при окончании приёма USART. Основная программа контролера DD1 является пустым циклом. Как только прием байта данных, переданных контроллером DD2 завершен, происходит прерывание. В результате этого прерывания выполняется обработчик прерывания, который копирует байт данных из регистра данных UDR USART в порт B, то есть на ЦАП. Ниже приведен текст этой программы.

```
;=====================================================================
; Исследование работы USART (программа для приемника DD1)
; Контроллер ATmega168
; Тактовая контроллера 8 МГц

.include"m168def.inc" ; присоединение файла описания ATmega168
.def temp = r16     ; временный регистр

; Расчёт скорости передачи данных USART
.equ fck = 8000000
.equ baud = 9600
.equ ubrr = fck/(16*baud)-1

;=====================================================================
; Макрокоманда "UOUT @0, @1" пересылки из РОН в любой РВВ
; не зависимо от адреса РВВ (вместо команд OUT и STS)
.macro uout
      .if @0 < 0x40
            out @0, @1
      .else
            sts @0, @1
      .endif
.endm

; Макрокоманда "UIN @0, @1" пересылки из любого РВВ в РОН
; не зависимо от адреса РВВ (вместо команд IN и LDS)
.macro uin
      .if @1 < 0x40
            in @0, @1
      .else
            lds @0, @1
```


Библиотека ПГУ www.lib.psu.by

```
      .endif
.endm

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

;=====================================================================
; векторы прерываний
.org 0
      rjmp reset              ; переход на обработку сброса
...
.org 36
      rjmp USART_ready        ; приём по USART завершен
...
.org 50
      reti ; готовность SPM

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

; начальная настройка портов
      ldi temp, 0b11111111
      uout ddrb, temp         ; установка порта B на вывод
      uout ddrc, temp         ; установка порта C на вывод

;=====================================================================
; инициализация USART (только прием)
      ldi temp, low(ubrr)     ; указываем скорость передачи данных
      uout ubrr0l, temp
      ldi temp, high(ubrr)
      uout ubrr0h, temp
      ldi temp, 0b00000000
; RXC  0b[0]xxxxxxx - флаг завершения приема
; TXC  0bx[0]xxxxxx - флаг завершения передачи
; UDRE 0bxx[0]xxxxx - флаг опустошения регистра данных
; FE   0bxxx[0]xxxx - флаг ошибки кадрирования
; DOR  0bxxxx[0]xxx - флаг переполнения
; UPE  0bxxxxx[0]xx - флаг ошибки контроля чётности
; U2X  0bxxxxxx[0]x - удвоение скорости обмена
; MPCM 0bxxxxxxx[0] - режим мультипроцессорного обмена
      uout ucsr0a, temp
      ldi temp, 0b10010000
; RXCIE 0b[1]xxxxxxx - разрешение прерывания по завершению приема
; TXCIE 0bx[0]xxxxxx - разрешение прерывания по завершению передачи
; UDRIE 0bxx[0]xxxxx - разрешение прерывания при очистке регистра данных
; RXEN  0bxxx[1]xxxx - разрешение приема
; TXEN  0bxxxx[0]xxx - резрешение передачи
; UCSZ2 0bxxxxx[0]xx - формат посылок (0 - для 5,6,7,8 бит)
```

Библиотека ГГТУ им.П.О.Сухого

```
; RXB8  0bxxxxxx[0]x - 8-й бит принимаемых данных
; TXB8  0bxxxxxxx[0] - 8-й бит передаваемых данных
      uout ucsr0b, temp
      ldi temp, 0b00000110
; UMSEL1:0 0b[00]xxxxxx - режим работы USART (асинхронный USART)
; UPM1:0   0bxx[00]xxxx - контроль чётности (без контроля чётности)
; USBS     0bxxxx[1]xxx - число стоповых-бит (для приемника - безразлично)
; UCSZ1:0  0bxxxxx[11]x - формат посылок (11 - для 8 бит)
; UCPOL    0bxxxxxxx[0] - полярность тактового сигнала
      uout ucsr0c, temp
      sei                           ; разрешение прерываний

;=====================================================================
; основная программа
main:
      rjmp main

;=====================================================================
; программа обработки прерываний от USART
USART_ready:
      uin temp, udr0                ; читаем регистр данных USART
      uout portb, temp              ; выводим байт данных на ЦАП
      reti
;=====================================================================
```

Программа для контроллера DD2 работает через прерывание при окончании преобразования АЦП. Основная программа контролера DD2 производит проверку флага *UDRE* опустошения регистра данных *UDR* USART. Как только регистр данных оказывается пуст, в него записывается результат очередного преобразования АЦП (8 бит из регистра *buf*), и производится передача этого байта через передатчик USART DD2 к DD1. При окончании очередного преобразования АЦП происходит прерывание, по которому младший байт АЦП копируется во временный регистр *buf*. Ниже приведен текст этой программы.

```
;=====================================================================
; Исследование USART (программа для передатчика DD2)
; Контроллер ATmega168
; Источник опорного напряжения AREF.
; Тактовая контроллера 8 МГц, делитель АЦП 128.

.include"m168def.inc"         ; присоединение файла описания ATmega168
.def temp = r16               ; временный регистр
.def buf = r17                ; буфер USART

; расчёт скорости передачи данных USART
.equ fck = 8000000
.equ baud = 9600
```


Библиотека ГГТУ им. П.О. Сухого

```
.equ ubrr = fck/(16*baud)-1

;=====================================================================
; Макрокоманда "UOUT @0, @1" пересылки из РОН в любой РВВ
; не зависимо от адреса РВВ (вместо команд OUT и STS)
.macro uout
      .if @0 < 0x40
            out @0, @1
      .else
            sts @0, @1
      .endif
.endm

; Макрокоманда "UIN @0, @1" пересылки из любого РВВ в РОН
; не зависимо от адреса РВВ (вместо команд IN и LDS)
.macro uin
      .if @1 < 0x40
            in @0, @1
      .else
            lds @0, @1
      .endif
.endm

;=====================================================================
.cseg                         ; выбор сегмента памяти
; векторы прерываний
.org 0
      rjmp reset              ; переход на обработку сброса
...
.org 42
      rjmp adc_int            ; окончание преобразования АЦП
...
.org 50
reti                          ; готовность SPM

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

; начальная настройка портов
      ldi temp, 0b00000001    ; отключение цифрового буфера линии ADC0
      uout didr0, temp
      ldi temp, 0b11111111    ; порт B на вывод
      uout ddrb, temp

;=====================================================================
; инициализация АЦП
      ldi temp, 0b00000000
; REFS1:0   0b[00]xxxxxx - выбор ИОН (напряжение AREF)
```


Библиотека ГГТУ им.П.О.Сухого

```
; ADLAR     0bxx[0]xxxxx - смещение 10-битного результата вправо
; bit4      0bxxx[0]xxxx - зарезервирован
; MUX3:0    0bxxxx[0000] - выбор аналогового входа
      uout 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 - флаг прерывания АЦП
; ADIE      0bxxxx[1]xxx - разрешение прерывания от АЦП
; ADPS2:0   0bxxxxx[111] - выбор делителя частоты 8МГц/128=62,5 кГц
      uout adcsra, temp

;=====================================================================
; инициализация USART (только передача)
      ldi temp, low(ubrr)     ; указываем скрость передачи данных
      uout ubrr0l, temp
      ldi temp, high(ubrr)
      uout ubrr0h, temp
      ldi temp, 0b00000000
; RXC       0b[0]xxxxxxx - флаг завершения приема
; TXC       0bx[0]xxxxxx - флаг завершения передачи
; UDRE      0bxx[0]xxxxx - флаг опустошения регистра данных
; FE        0bxxx[0]xxxx - флаг ошибки кадрирования
; DOR       0bxxxx[0]xxx - флаг переполнения
; UPE       0bxxxxx[0]xx - флаг ошибки контроля чётности
; U2X       0bxxxxxx[0]x - удвоение скорости обмена
; MPCM      0bxxxxxxx[0] - режим мультипроцессорного обмена
      uout ucsr0a, temp
      ldi temp, 0b00001000
; RXCIE     0b[0]xxxxxxx - разрешение прерывания по завершению приема
; TXCIE     0bx[0]xxxxxx - разрешение прерывания по завершению передачи
; UDRIE     0bxx[0]xxxxx - разрешение прерывания при очистке рег. данных
; RXEN      0bxxx[0]xxxx - разрешение приема
; TXEN      0bxxxx[1]xxx - резрешение передачи
; UCSZ2     0bxxxxx[0]xx - формат посылок (0 - для 5,6,7,8 бит)
; RXB8      0bxxxxxx[0]x - 8-й бит принимаемых данных
; TXB8      0bxxxxxxx[0] - 8-й бит передаваемых данных
      uout ucsr0b, temp
      ldi temp, 0b00000110
; UMSEL1:0  0b[00]xxxxxx - режим работы USART (00 - асинхронный USART)
; UPM1:0    0bxx[00]xxxx - контроль чётности (00 - без контроля чётности)
; USBS      0bxxxx[1]xxx - число стоповых-бит (1 - 2 стоп-бита)
; UCSZ1:0   0bxxxxx[11]x - формат посылок (11 - для 8 бит)
; UCPOL     0bxxxxxxx[0] - полярность тактового сигнала
      uout ucsr0c, temp
      sei                     ; разрешение прерываний

;=====================================================================
; основная программа
main:
      uin temp, ucsr0a
```

Библиотека ГГТУ им. П.О.Сухого

```
      sbrc temp, udre0        ; ждем опустошения регистра данных USART
      uout udr0, buf          ; отправляем содержимое буфера в регистр
                              ; данных USART
      rjmp main

;=====================================================================
; обработка прерывания от АЦП
adc_int:
      uin buf, adcl           ; чтение мл.байта АЦП в регистр buf
      uin temp, adch
      uout portb, buf         ; вывод buf в порт B (для контроля)
      reti
```

На рис. 5.2 приведен пример симуляции работы контроллеров в программе Proteus. Аналоговые напряжения на входе АЦП и на выходе ЦАП практически равны и составляют примерно 1,19 В. Двоичный код на выходе порта B обоих контроллеров совпадает и равен $1111\ 0101_2$. На осциллограмме напряжения на выходе передатчика DD2 показан формат посылки, который соответствует передаваемому коду.

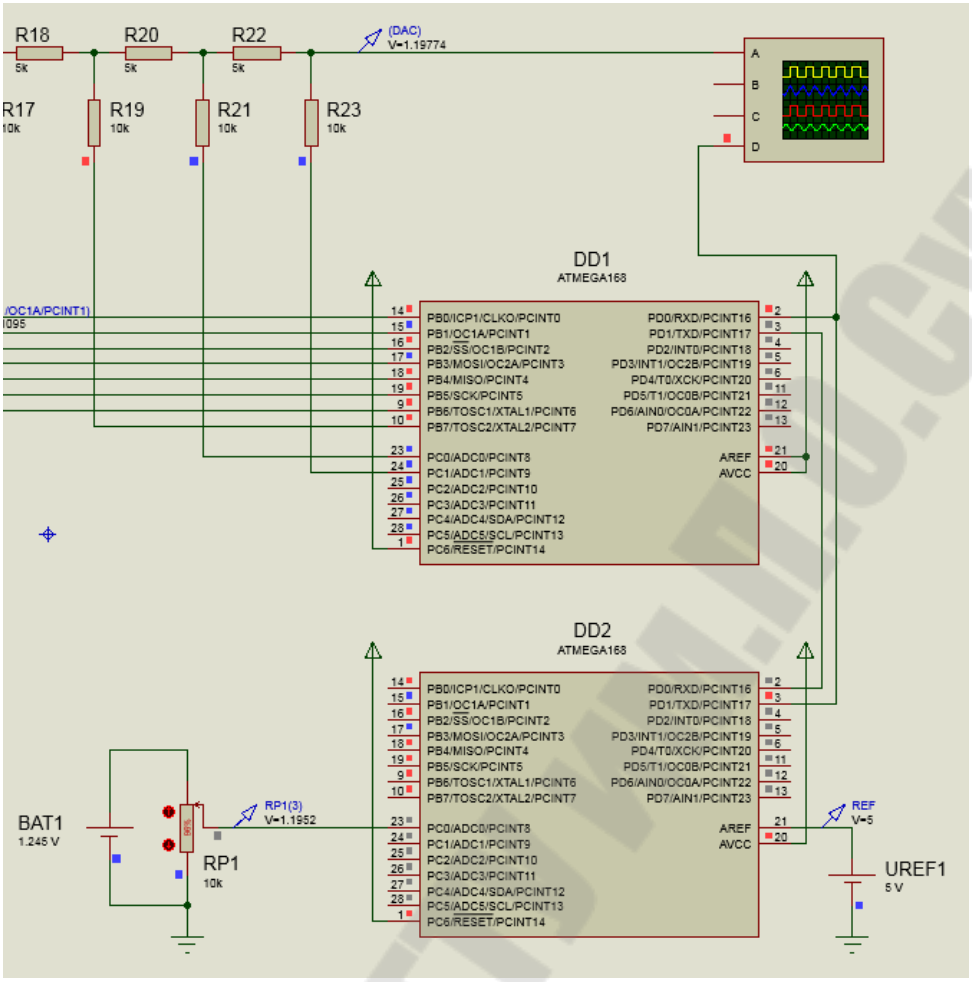
5.4. Задание для самостоятельной работы

1. Напишите программу для контролера-приёмника, работающую по контролю флагов.

2. Напишите программу для контролера–передатчика, работающую по прерываниям.

3. Используйте микропереключатели для задания передаваемой величины и светодиоды для вывода полученного результата на принимающей стороне.

Библиотека ГГТУ им. П.О.Сухого



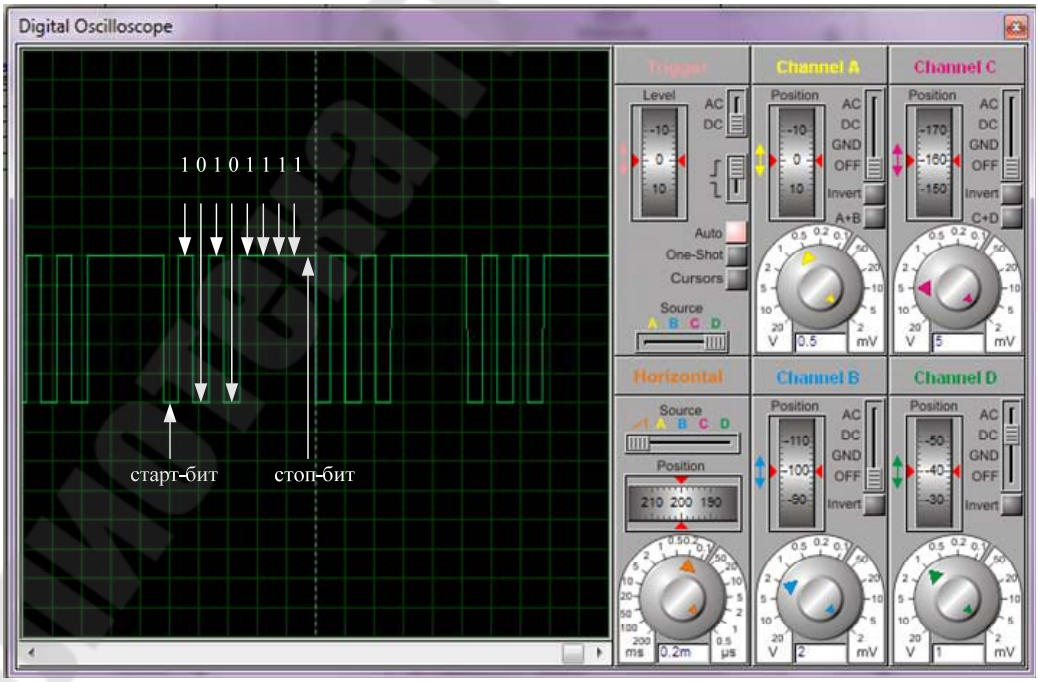


Рис. 5.2. Результат симуляции в Proteus

Литература

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

2. Баранов, В. Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы / В. Н. Баранов. – М. : Додэка-XXI, 2004. – 288 с.

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. Книга 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.

Библиотека ГГТУ им. П.О.Сухого

Содержание

Библиотека ГГТУ им.П.О.Сухого

Савельев Вадим Алексеевич
Дорощенко Игорь Васильевич

МИКРОПРОЦЕССОРНЫЕ СРЕДСТВА В АВТОМАТИЗИРОВАННОМ ЭЛЕКТРОПРИВОДЕ

Практикум
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 16.11.21.

Рег. № 53Е.
http://www.gstu.by

Библиотека ГГТУ им. П.О.Сухого