Лабораторная работа 4

Формирование временных интервалов с помощью микроконтроллеров (Таймер ATmega16)

Цель работы

Изучить основные приёмы задания временных интервалов с помощью микроконтроллера ATmega16.

Освоить методику построения на основе микроконтроллера ATmega16 генераторов импульсов с заданными длительностями, как автоколебательных, так и ждущих.

Научиться создавать программы на языке ассемблера ATmega16 с использованием подпрограмм.

Формирование временных интервалов заданной длительности на базе микроконтроллера можно организовать с помощью внешнего генератора импульсов, выходной сигнал которого подаётся на входной порт микроконтроллера, и программы-счётчика импульсов. С целью уменьшения количества внешних устройств, в состав микроконтроллеров входят специальные регистры (таймеры), содержание которых инкрементируется не только по приходу импульса от внешнего генератора, но и по импульсу тактового генератора микроконтроллера. Как правило, предусматривается возможность инкрементирования счётчика не по каждому импульса тактового генератора, а по каждому 8-му, или 64-му, или 512-му и т.д. импульсу.

В состав микроконтроллера ATmega16 входят три таймера – два 8-разрядных (T0 и T2) и один 16-разрядный (T1).

В настоящей лабораторной работе рассматриваются структура и основные действия с таймером T0. Состояние таймера T0 хранится в регистре TCNT0, доступ к которому постоянно открыт и для чтения, и для записи. Содержимое этого регистра может изменяться по одному из алгоритмов, который устанавливается с помощью трёх младших разрядов вспомогательного регистра TCCR0 (CS00, CS01, CS02):

CS02	CS01	CS00	
0	0	0	Таймер остановлен
0	0	1	Каждый тактовый импульс микроконтроллера инкрементирует регистр TCNT0
0	1	0	Каждый 8-й тактовый импульс микроконтроллера инкрементирует регистр TCNT0 (коэффициент предделения равен 8)

0	1	1	Каждый 64-й тактовый импульс микроконтроллера инкрементирует регистр TCNT0 (коэффициент предделения равен 64)
1	0	0	Каждый 256-й тактовый импульс микроконтроллера инкрементирует регистр TCNT0 (коэффициент предделения равен 256)
1	0	1	Каждый 1024-й тактовый импульс микроконтроллера инкрементирует регистр TCNT0 (коэффициент предделения равен 1024)
1	1	0	Регистр TCNT0 инкрементируется фронтом внешнего сигнала, подаваемого на вывод T0 микроконтроллера
1	1	1	Регистр TCNT0 инкрементируется спадом внешнего сигнала, подаваемого на вывод T0 микроконтроллера

Оставшиеся 5 разрядов регистра TCCR0 определяют также другие, более сложные алгоритмы работы таймера T0. Подробно изучить эти алгоритмы можно, например, в источнике:
http://chipenable.ru/index.php/programming-avr/item/171-avr-timer-t2-ch1.html

На основе таймера можно создавать генераторы сложно последовательности импульсов, работающие как в автоколебательном, так и в ждущем режимах. Рассмотрим в качестве примера автоколебательный генератор последовательности импульсов, представленной на рис. 1.

Импульс «1», длящийся 10 мс, сменяется паузой «0» той же длительности, после которой следует второй импульс «1» (10 мс) и вторая пауза «0» длительностью уже 20 мс, за которой идёт третий импульс«1» (10 мс) и третья пауза «0» длительностью 60 мс.

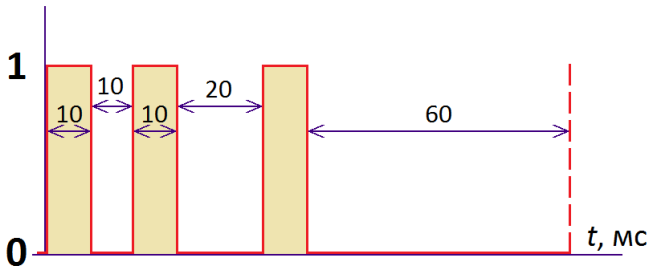

Рис. 1. Пример импульсного периодического сигнала

При установке трёх младших разрядов регистра TCCR0 в значение 101_2 значение таймера TCNT0 будет инкрементироваться с частотой $f_{такт}/1024_{10}$, которая при $f_{такт}$ = 1 МГц (по умолчанию для ATmega16 с использованием внутреннего RC-генератора) принимает значение 0,9765625 кГц. Есть возможность подкорректировать это значение значением специального регистра OCSCAL. По умолчанию значение OCSCAL равно 100_{16}, его уменьшение ведёт к уменьшению тактовой частоты, а увеличение – к её увеличению. В микроконтроллере ATmega16 также предусмотрена возможность изменять значение тактовой частоты по умолчанию с помощью значения конфигурационного регистра CKSEL – при использовании внутреннего RC-генератора значение $f_{такт}$ определяется четырьмя младшими **регистрами** CKSEL:
CKSEL = 0001_2 – $f_{такт}$ = 1 МГц;

CKSEL = 0010_2 – $f_{такт}$ = 2 МГц;
CKSEL = 0011_2 – $f_{такт}$ = 4 МГц;
CKSEL = 0100_2 – $f_{такт}$ = 8 МГц.

При тактовой частоте 1 МГц, значении коэффициента предделения 1024 и OCSCAL = 135_{10} значения таймеров ATmega16 инкрементируются с частотой, с высокой степенью точности равной 1 кГц.

Алгоритм формирования периодического сигнала, представленного на рис. 1 можно представить в линейном виде (рис. 2):

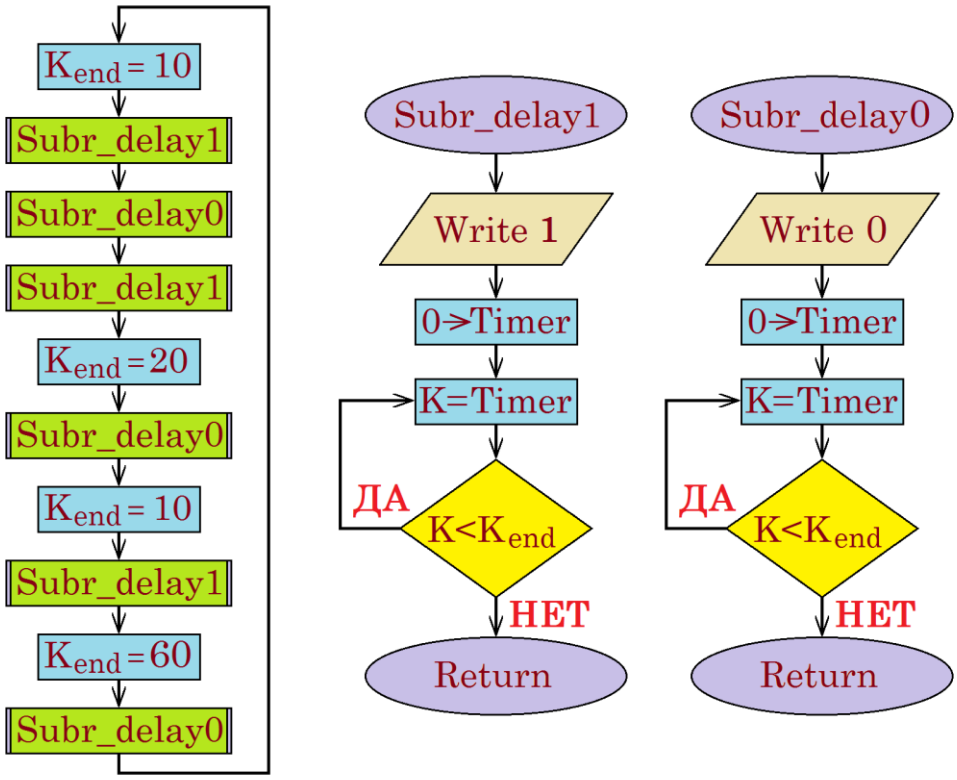

Рис. 2. Схема алгоритма для формирования сигнала рис. 1.

На схеме рис. 2 используются подпрограммы Subr_delay1 и Subr_delay0. На языке ассемблера ATmega16 подпрограммы вызываются командой rcall subr_name, где subr_name является меткой начала подпрограммы, а сама подпрограмма записывается как

```
subr_name:
…
ret
```

Необходимо напомнить, что использование подпрограмм предполагает сохранение текущего состояния основной программы в стеке микроконтроллера, поэтому перед вызовом первой подпрограммы необходимо инициализировать стек, то есть занести в указатель стека адрес самой старшей ячейки разработанной программы. На языке ассемблера ATmega16 эта процедура выглядит следующим образом:

```
ldi             temp,low(RAMEND)   ; инициализация стека
out             spl,temp
ldi             temp,high(RAMEND)
out             sph,temp
```

Полный текст программы, реализующей алгоритм рис. 2:

```
.include "m16def.inc" ; подключение библиотеки для работы с ATmega16
.list ; включение листинга
.def temp=r16 ; определение главного рабочего регистра
.def k__z=r17
.def k___=r18
.def s___=r19
;-----------------------------------------
.cseg ; выбор сегмента программного кода
.org 0 ; установка текущего адреса на ноль
;-----------------------------------------
ldi temp,0x80 ; выключение компаратора
out acsr,temp
;-----------------------------------------
ldi temp,0x00    ; 0 --> temp
out ddrd,temp  ; Назначаем порт rd на ввод (00000000 --> ddrd)
ldi temp,0xFF    ; 0xff --> temp
out ddrb,temp  ; Назначаем порт rb на вывод (11111111 --> ddrb)
;-----------------------------------------
ldi temp, 0b101; Предделение 1024
out tccr0, temp
ldi temp, 135    ; Коррекция тактовой частоты
out osccal, temp

ldi         temp,low(RAMEND)   ; инициализация стека
out         spl,temp
ldi         temp,high(RAMEND)
out         sph,temp

ldi         temp, 0

met:

ldi         k__z, 10
rcall       subr_delay1

rcall       subr_delay0

rcall       subr_delay1

ldi         k__z, 20
rcall       subr_delay0

ldi         k__z, 10
```

```
rcall     subr_delay1

ldi       k__z, 60
rcall     subr_delay0

jmp       met

subr_delay1:          ; "1" длится k__z тактов с предделением
ldi       s___, 1     ; 1 --> s___
out       portb, s___ ; s___ --> pb
out       tcnt0, temp ; 0 --> tcnt0 Обнуление таймера
ccc1:                 ; повтор цикла
in        k___, tcnt0 ; считали таймер
cp        k___, k__z  ; сравнили k__ и k__z
brlo      ccc1        ; если k___<k__z, ушли в начало
ret                   ; конец подпрограммы subr_delay1

subr_delay0:          ; "0" длится k__z тактов с предделением
ldi       s___, 0     ; 0 --> s___
out       portb, s___ ; s___ --> pb
out       tcnt0, temp ; 0 --> tcnt0
ccc0:                 ; повтор цикла
in        k___, tcnt0 ; считали таймер
cp        k___, k__z  ; сравнили k__ и k__z
brlo      ccc0        ; если k___<k__z, ушли в начало
ret                   ; конец подпрограммы subr_delay
```

Программа, прошедшая трансляцию в AVR_Studio и загруженная в микроконтроллер ATmega16, обеспечивает результат, приведённый на рис. 3.

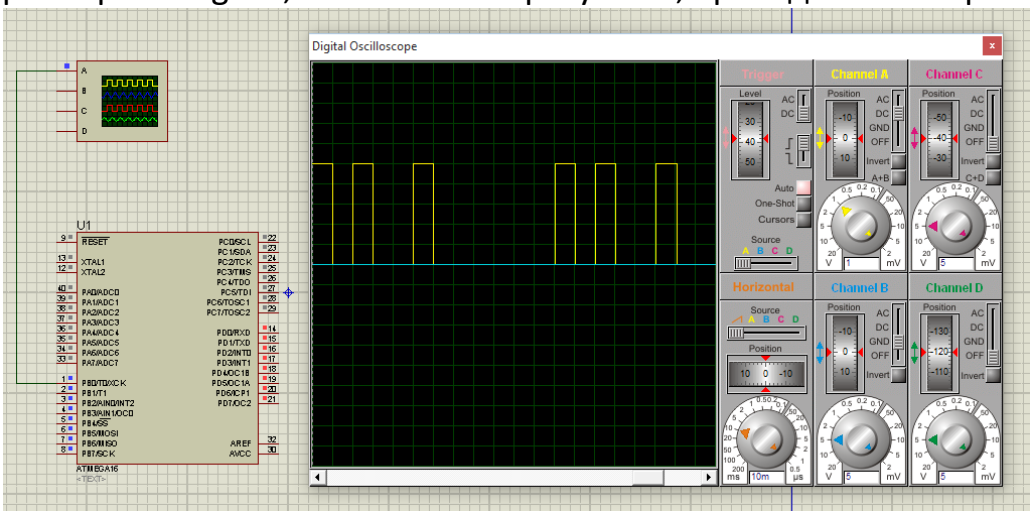

Рис. 3. Работа автоколебательного генератора импульсной последовательности

Построение генератора, работающего в ждущем режиме, также не представляет особых трудностей. Для этого необходимо запрограммировать один из регистров микроконтроллера на вход, обеспечить непрерывный опрос этого регистра, и при реализации входного перепада 0→1 стартует выполнение заданной импульсной последовательности. Необходимый перепад можно реализовать с помощью кнопки, в этом случае следует предусмотреть подключение подтягивающих резисторов, а можно использовать для этого специальный импульсный генератор, без подтягивающих резисторов. Схема алгоритма для построения ждущего импульсного генератора приведена на рис. 4.

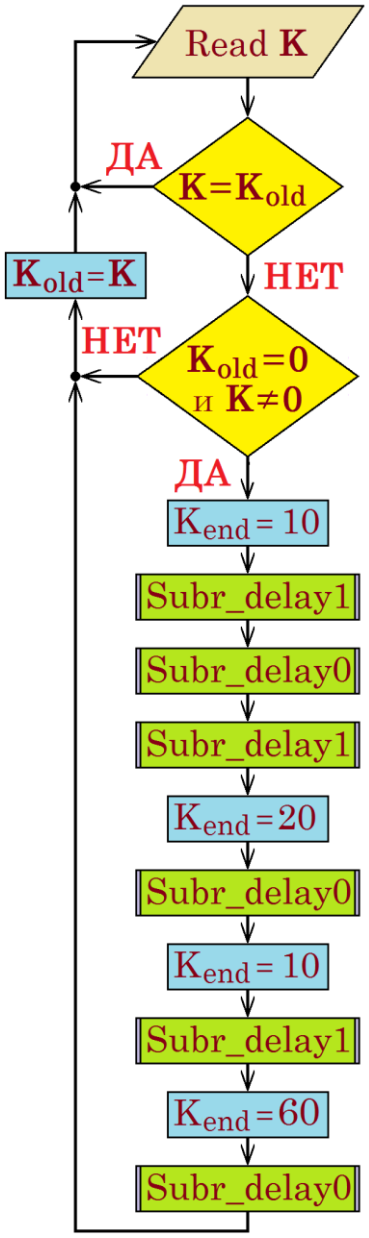

Рис. 4. Схема алгоритма для импульсного генератора в ждущем режиме

Применение микроконтроллеров для построения генераторов заданных импульсных последовательностей открывает массу возможностей – можно нажатием на различные кнопки вызывать различные импульсные последовательности, можно добиться кварцевой стабилизации длительностей импульсов (при использовании кварцевой стабилизации тактовой частоты), можно реализовать несколько абсолютно синхронный импульсных последовательностей и т.д. Некоторым слабым местом является сравнительно невысокие частоты импульсных последовательностей – так, микроконтроллер ATmega16 имеет максимальное значение тактовой частоты только 16 МГц.

В настоящей лабораторной работе предлагается на выбор два варианта:

Вариант А: Создать в системе PROTEUS генераторы импульсных последовательностей, работающие в автоколебательном и в ждущем режимах. В этом случае исходными данными является импульсная последовательность, рассмотренный пример (рис. 1) можно задать как **10** 10 **10** 20 **10** 60.

Вариант В: Создать в системе PROTEUS генератор музыкальных мелодий. В этом случае исходными данными является нотная запись мелодии. По нотной записи необходимо построить последовательность звуков, каждый из которых имеет некоторую частоту и некоторую длительность. Для вывода звука следует использовать элемент SPEAKER, к которому подключаются гене-

ратор синусоидальных колебаний с различными частотами. Подключение осуществляется с помощью управляемых ключей VSWITCH, переключение которых осуществляется с помощью микроконтроллера. Общая схема генератора показана на рис. 5.

Как нетрудно догадаться, по схеме рис. 5 можно реализовать генератор музыкальных мелодий, которые содержат не более восьми нот. Если мелодия содержит большее количество нот, следует использовать дешифратор.

С помощью схемы рис. 5 можно воспроизводить также полифонические мелодии.

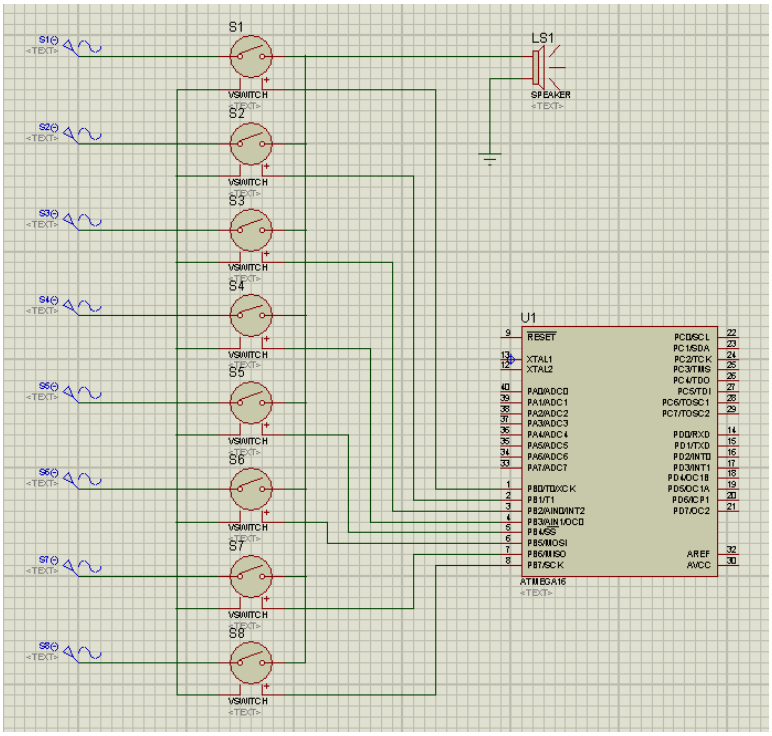

Рис. 5. Схема генератора музыкальных мелодий

Вариант А

Порядок выполнения работы.

1. Получить у преподавателя задание – последовательность импульсов.

2. Составить схемы алгоритмов работы генератора в ждущем и в автоколебательном режимах.

3. Написать на языке ассемблера ATMEGA16 программы, реализующие алгоритмы п.2.

4. С помощью программы AVR_Studio осуществить трансляцию программ (получить hex-файлы).

5. Собрать в системе PROTEUS схему (или схемы) на основе микроконтроллера ATMEGA16, реализующую разрабатываемые генераторы.

6. Убедиться в правильном функционировании разработанных генераторов.

Содержание отчёта.

Отчёт должен содержать:

1. Задание лабораторной работы – последовательность импульсов.

2. Алгоритм (алгоритмы) работы системы

3. Программу (программы) для микроконтроллера ATMEGA16, реализующую разработанный алгоритм.

4. Функционирование разработанных генераторов должно быть продемонстрировано в программе PROTEUS.

Вариант В

Порядок выполнения работы.

1. Получить у преподавателя задание – нотную запись мелодии.

2. Составить последовательность частот и длительностей нот в мелодии.

3. Собрать в системе PROTEUS схему на основе микроконтроллера ATMEGA16, управляющую генераторами синусоидальных напряжений требуемых частот.

4. Написать на языке ассемблера ATMEGA16 программу, реализующую последовательность нот п.2.

Следует обратить внимание, что описанным алгоритмом будут воспроизводиться ноты, амплитуды звучания которых абсолютно постоянны. Поэтому для их отделения друг от друга следует предусмотреть короткую паузу (это, конечно, не относится к нотам, объединённым легато).

5. С помощью программы AVR_Studio осуществить трансляцию программ (получить hex-файлы).

6. Убедиться в правильном функционировании разработанного генератора музыкальной мелодии.

Содержание отчёта

Отчёт должен содержать:

1. Задание лабораторной работы – нотную запись мелодии.

2. Последовательность частот и длительностей нот в мелодии.

3. Программу для микроконтроллера ATMEGA16, реализующую заданную мелодию.

4. Функционирование разработанного генератора должно быть продемонстрировано в программе PROTEUS.