Научно-производственное предприятие
«Учтех-Профи»

ПРОГРАММИРОВАНИЕ МИКРОКОНТРОЛЛЕРА ATMEGA 8535 НА АССЕМБЛЕРЕ

Часть 2

Методические указания
к проведению лабораторных работ

Челябинск
2013

комментируются. Частота ШИМ-сигнала устанавливается битами CS22…CS20. Эти биты определяют источник задания частоты и коэффициент делителя (табл. 6).

Табл. 6. Установка источника задания частоты

CS22	CS21	CS20	Описание
0	0	0	Таймер остановлен
0	0	1	Источник – генератор частоты процессора. ($f_{T0} = f_{CLK}/1$)
0	1	0	Источник – генератор частоты процессора. ($f_{T0} = f_{CLK}/8$)
0	1	1	Источник – генератор частоты процессора. ($f_{T0} = f_{CLK}/32$)
1	0	0	Источник – генератор частоты процессора. ($f_{T0} = f_{CLK}/64$)
1	0	1	Источник – генератор частоты процессора. ($f_{T0} = f_{CLK}/128$)
1	1	0	Источник – генератор частоты процессора. ($f_{T0} = f_{CLK}/256$)
1	1	1	Источник – генератор частоты процессора. ($f_{T0} = f_{CLK}/1024$)

Необходимо отметить, что в режиме ШИМ необязательно устанавливать маски и разрешать прерывания по совпадению и переполнению таймера T0 и T2. Эти прерывания следует активировать только при необходимости их использования в программе.

Для запуска режима ШИМ на таймере T0 или T2 необходимо:

– остановить таймер обнулением регистра управления TCCR0(2);

– обнулить регистр сравнения OCR0(2) и счетный регистр TCNT0(2);

– установить в регистре управления TCCR0(2) режим ШИМ и выбрать частоту его работы.

В процессе работы ШИМ в любой момент можно изменять содержимое регистра сравнения OCR0(2) таймеров, при этом среднее значение ШИМ-сигнала на выводе OC0 и OC2 микроконтроллера будет пропорционально изменяться.

Пример. Таймер T0 работает в режиме быстрого ШИМ. В зависимости от состояния бит порта ввода/вывода PORTA регулируется яркость светодиода, подключенного к выходу ШИМ таймера T0. Логическим сигналом PORTD0 ШИМ изменяется с инвертирующего на неинвертирующий.

```
;---------------------------------------------------------------
;ШИМ на таймере T0
;входы:
;     PORTA0...PORTA7      - задание уставки таймера
;     PORTD0               - инвертирующий (1)/неинвертирующий (0) ШИМ
;выходы:
;     PORTB3               - ШИМ на таймере T0
```

```
.include"m8535def.inc"   ;подключение стандартной библиотеки ATmega8535
.cseg                    ;начало сегмента кода
.org$0                   ;по адресу 0
rjmp reset               ;и переход на reset

reset:
cli                      ;запрет всех прерываний
ldi r16,low(RAMEND)      ;запись в указатель стека адреса конца
ldi r17,high(RAMEND)     ;памяти данных
out spl,r16
out sph,r17
ldi r16,0x01   ;инициализация портов ввода вывода.
out PORTD,r16  ;PORTD0 - на ввод информации
clr r16
out DDRD,r16
ldi r16,0xFF
out PORTA,r16  ;PORTA - на ввод информации
clr r16
out DDRA,r16
out PORTB,r16
ldi r16,0x08
out DDRB,r16   ;PORTB3 - на вывод информации
clr r16
out TCCR0,r16  ;сброс регистра управления таймера T0
out OCR0,r16   ;сброс регистра сравнения таймера T0
out TCNT0,r16  ;сброс регистра счета таймера T0
ldi r16,0x69   ;установка режима быстрого неинвертирующего
out TCCR0,r16  ;ШИМ таймера T0
sei            ;разрешение всех прерываний
main:          ;начало основного цикла
in r16,PINA    ;считывание значений порта ввода/вывода PORTA
out OCR0,r16   ;и его отправка в регистр сравнения таймера T0
in r16,PIND    ;считывание значений порта ввода/вывода PORTD
andi r16,0x01  ;и выделение бита PORTD0
cpi r16.0x01   ;Если PORTD0=1
breq met1      ;то переход на метку met1
ldi r17,0x69   ;иначе запись в r17 значения для неинвертирующего ШИМ
rjmp met2      ;и переход на метку met2
met1:          ;По метке met1
ldi r17,0x79   ;запись в r17 значения для инвертирующего ШИМ
met2:          ;По метке met2
out TCCR0,r17  ;установка заданного режима работы таймера T0
rjmp main      ;и возврат на main
;---------------------------------------------------------------------
```

Рассмотрим программу более подробно.

1. Сначала производится подключение необходимых библиотек и определение начала сегмента кода:

```
include"m8535def.inc"
.cseg
.org$0
rjmp reset
```

2. При переходе на reset в указателе стека устанавливается адрес конца памяти данных (вершина стека):

```
reset:
cli
ldi r16,low(RAMEND)
ldi r17,high(RAMEND)
out spl,r16
out sph,r17
```

3. После этого инициализируются порты ввода/вывода в соответствии с поставленным заданием:

```
ldi r16,0x01
out PORTD,r16
clr r16
out DDRD,r16
ldi r16,0xFF
out PORTA,r16
clr r16
out DDRA,r16
out PORTB,r16
ldi r16,0x08
out DDRB,r16
```

4. После установки портов ввода/вывода таймер T0 программируется на заданный режим работы:

```
clr r16
out TCCR0,r16
out OCR0,r16
out TCNT0,r16
ldi r16,0x69
out TCCR0,r16
sei
```

Сначала очищаются все регистры таймера (`clr r16; out TCCR0,r16; out OCR0,r16; out TCNT0,r16`). После этого в регистр управления TCCR0, в соответствии с табл. 1 записываются необходимые комбинации управляющих бит. Установкой WGM00=1, WGM00=1 выбирается режим быстрого ШИМ; установкой COM01=1, COM00=0 выбирается режим неинвертирующего ШИМ; биты CS02:CS01:CS00=001 определяют работу таймера без предделителя частоты процессора clk/1.

5. В начале главного цикла происходит считывание данных с порта A и их запись в регистр сравнения таймера T0:

```
main:
in r16,PINA
out OCR0,r16
```

6. После этого производится опрос порта D и выделение младшего значащего бита:

```
in r16,PIND
andi r16,0x01
```

7. Происходит сравнение PORTD0 с логическими уровнями 0 и 1:

```
cpi r16,0x01
breq met1
ldi r17,0x69
rjmp met2
met1:
ldi r17,0x79
```

Если в результате сравнения (`cpi r16,0x01`) PORTD0=1, то происходит переход на метку met1, по которой в регистр r17 записывается значение, которое необходимо записать в регистр управления TCCR0 таймера T0 для реализации инвертирующего ШИМ (`ldi r17,0x79`). Иначе в r17 производится запись значения TCCR0 для неинвертирующего ШИМ (`ldi r17,0x69`).

8. При переходе на метку met2 происходит запись полученного в r17 значения в регистр управления TCCR0 таймера T0:

```
met2:
out TCCR0,r17
rjmp main
```

После этого производится зацикливание программы (rjmp main).

Подробное рассмотрение программы показывает, что при использовании в таймере режима ШИМ пользователь может не использовать прерывания по совпадению и переполнению таймера, так как в режиме ШИМ все действия таймер делает автоматически. Это существенно облегчает организацию программы и уменьшает ее размер.

В качестве обязательного задания все студентам необходимо написать и опробовать два варианта программа – рассмотренный с использованием режима ШИМ и ручной с использованием прерываний по совпадению и переполнению таймера.

Работа № 2. 16-разрядный таймер Т1. Режим подсчета временных интервалов.

Цель работы

Освоить теоретический и практический материал по работе 16-разрядного таймера Т1 микроконтроллера Atmega 8535 в режиме подсчета временных интервалов. Применить приобретенные навыки при написании программы.

Программа работы

1. Изучить необходимый теоретический материал о регистрах и функционировании таймера Т1.
2. Разобраться в программе по использованию таймера Т1, представленной в лабораторной работе.
3. Написать и отладить собственную программу с использованием таймера в соответствии с вариантом.

Пояснения к работе

Помимо двух восьмиразрядных таймеров/счетчиков Т0 и Т2, микроконтроллер Atmega8535 содержит двухканальный шестнадцатиразрядный таймер/счетчик Т1.

Основным отличием таймера Т1 от таймеров Т0 и Т2 является его 16-разрядная организация.

Все регистры таймера состоят из двух частей – старшей и младшей, обозначаемых, соответственно, буквами H и L. Так, например, регистр счета таймера Т1 состоит из двух 8-разрядных регистров **TCNT1H** и **TCNT1L**.

Таймер/счетчик Т1 служит для подсчета временных интервалов, регистрации внешних событий и работы в режиме ШИМ для вывода цифрового периодического сигнала с регулируемой скважностью и частотой.

Таймер синхронизируется от источника тактового сигнала процессора, либо от источника внешнего сигнала, подаваемого на цифровой вход микроконтроллера.

Рассмотрим работу таймера при его работе от источника тактового сигнала процессора (рис. 1), в качестве которого в лабораторном стенде выступает кварцевый резонатор с тактовой частотой 8 МГц.

Счет импульсов источника частоты ведется в 16-разрядном счетном регистре таймера **TCNT1=TCNT1H:TCNT1L**. Перед тем, как попасть в схему счета импульсов, тактовый сигнал поступает на схему деления частоты, которая, в соответствии с параметрами, установленными при инициализации таймера, производит деление частоты тактового сигнала f_{CLK} в следующем соотношении:

– без делителя частоты тактового сигнала f_{CLK};
– с делителем частоты тактового сигнала $f_{CLK}/8$;
– с делителем частоты тактового сигнала $f_{CLK}/64$;
– с делителем частоты тактового сигнала $f_{CLK}/256$;
– с делителем частоты тактового сигнала $f_{CLK}/1024$.

Каждый импульс с предделителя частоты считается и инкрементирует значение, содержащееся в счетном регистре **TCNT1** (рис. 1).

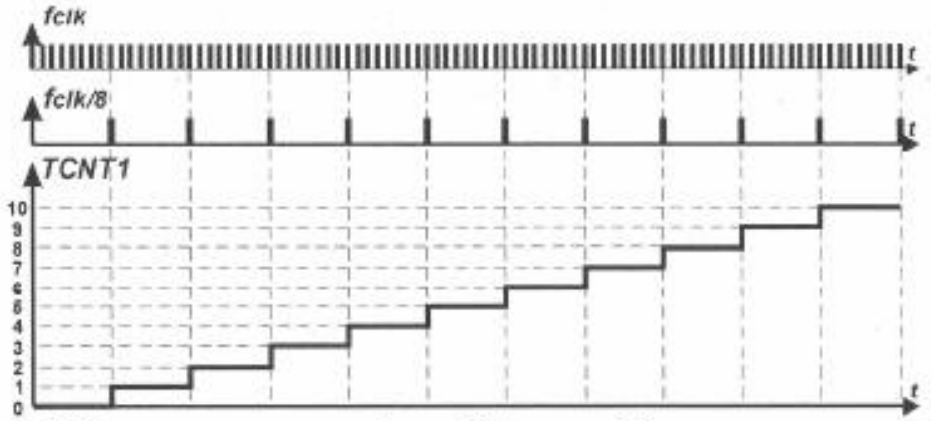


Рис. 1. Работа предделителя таймера T1 при коэффициенте делителя clk/8

Особенностью таймера T1 является наличие двух каналов сравнения. Несмотря на то, что счетный регистр **TCNT1** у таймера – один, он содержит два счетных регистра OCR1 (output compare register): **OCR1A=OCR1AH:OCR1AL**, и **OCR1B=OCR1BH:OCR1BL**.

В эти регистры записываются необходимые уставки срабатывания, то есть числа от 0 до максимального (2^{16}=65535), при достижении которых счетным регистром **TCNT1** формируются флаги прерываний по совпадению таймера T1 по каналам A и B **OCF1A, OCF1B** (Output Compare Flag).

Когда значение, записанное в регистре **TCNT1**, переполняет максимальное значение, которое можно записать в 16-разрядный регистр, формируется флаг прерывания по переполнению таймера **TOV1** (Timer Overflow Flag) (рис. 2).

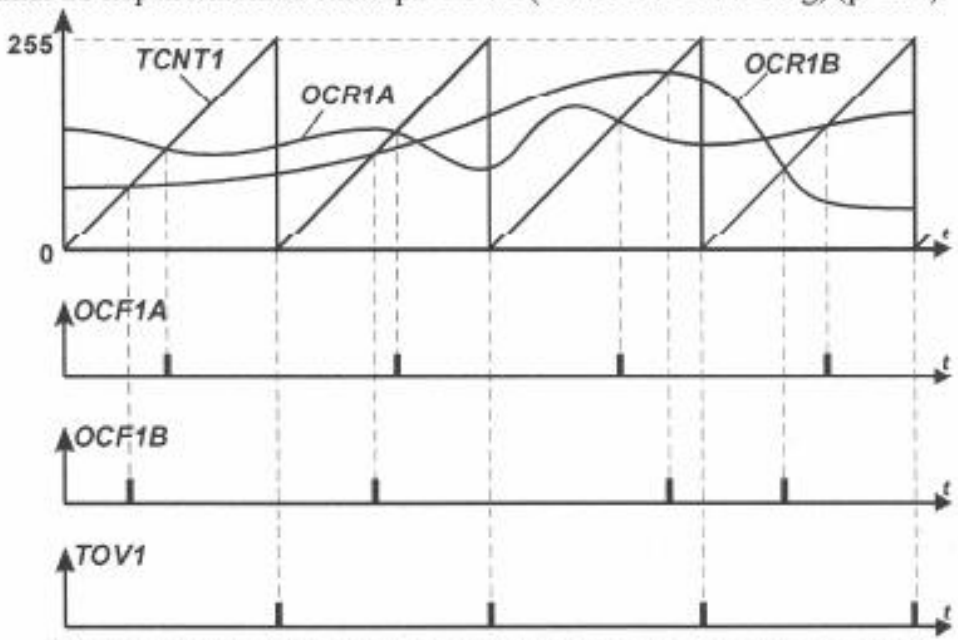


Рис. 2. Принцип работы таймера и формирования прерываний

Управление таймером T1 осуществляется через два 8-разрядных регистра управления **TCCR1A, TCCR1B.**

Регистр управления **TCCR1A** состоит из управляющих бит, пояснение которых приведено в табл. 1.

Табл. 1. Регистр управления таймера T1 TCCR1A

Бит	7	6	5	4	3	2	1	0
Название	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10

Биты 7 и 6 COM1A1 и COM1A0. Микросхема контроллера Atmega 8535 содержит вывод OC1A таймера T1 в качестве альтернативной функции вывода PD5 порта ввода/вывода D. Таймер T1 может управлять этим выводом в соответствии с выбранным режимом работы, определяемым комбинацией бит COM1A1:COM1A0 (табл. 2)

Табл. 2. Функции вывода OC1A таймера T1 в режиме подсчета временных интервалов

COM1A1	**COM1A0**	**Пояснение**
0	0	Вывод OC1A отключен
0	1	Изменение OC1A на противоположное при совпадении OCF1A
1	0	Очистка OC1A при совпадении OCF1A
1	1	Установка OC1A при совпадении OCF1A

Биты 5 и 4 COM1B1 и COM1B0. Микроконтроллер Atmega 8535 содержит вывод OC1B таймера T1 в качестве альтернативной функции вывода PD4 порта ввода/вывода D. Таймер T1 может управлять этим выводом в соответствии с выбранным режимом работы, определяемым комбинацией бит COM1B1:COM1B0 (табл. 3)

Табл. 3. Функции вывода OC1B таймера T1 в режиме подсчета временных интервалов

COM1B1	**COM1B0**	**Пояснение**
0	0	Вывод OC1B отключен
0	1	Изменение OC1B на противоположное при совпадении OCF1B
1	0	Очистка OC1B при совпадении OCF1B
1	1	Установка OC1B при совпадении OCF1B

Биты 3 и 2 FOC1A и FOC1B. Эти биты действуют только с обычном режиме работы при подсчете временных интервалов. При записи в них логической «1» происходит срабатывание прерывания по совпадению соответствующего канала таймера и изменение выводов таймера OC1A и OC1B в соответствии с настройками бит COM1A1:COM1A0, COM1B1:COM1B0.

Биты 1 и 0 WGM11, WGM10. Эти биты определяют режим работы таймера вместе с управляющими битами регистра TCCR1B.

Регистр управления TCCR1B состоит из управляющих бит, пояснение которых приведено в табл. 4.

Табл. 4. Регистр управления таймера T1 TCCR1B

Бит	7	6	5	4	3	2	1	0
Название	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10

Бит 7 – ICNC1. Этот бит включает функцию фильтрации внешнего сигнала, поступающего на вывод PB1 микроконтроллера, когда таймер используется в качестве счетчика. Установка бита увеличивает надежность работы таймера. При активации этого бита между подачей сигнала на вывод PB1 и срабатыванием схемы счета проходит 4 периода тактовой частоты.

Бит 6 – ICES1. Этот бит определяет активный фронт сигнала на входе PB1. При ICES1=0 таймер срабатывает по спадающему фронту сигнала, иначе – по нарастающему фронту.

Бит 5. Зарезервирован и не используется.

Биты 4 и 3 – WGM13, WGM12. Эти биты вместе с битами WGM11, WGM 10 регистра TCCR1A определяют режим работы таймера (табл. 5).

Табл. 5. Задание режима работы таймера T1

Режим	WGM13	WGM12	WGM11	WGM10	Название	Вершина	Обновление OCR1x	Переполнение TOV1
0	0	0	0	0	Нормальный режим	0xFFFF	Немедленно	Максимум
1	0	0	0	1	Фазовый ШИМ, 8 бит	0x00FF	На вершине	Минимум
2	0	0	1	0	Фазовый ШИМ, 9 бит	0x01FF	На вершине	Минимум
3	0	0	1	1	Фазовый ШИМ, 10 бит	0x3FFF	На вершине	Минимум
4	0	1	0	0	Очистка при совпадении	OCR1A	Немедленно	Максимум
5	0	1	0	1	Быстрый ШИМ, 8 бит	0x00FF	На вершине	На вершине
6	0	1	1	0	Быстрый ШИМ, 9 бит	0x01FF	На вершине	На вершине
7	0	1	1	1	Быстрый ШИМ, 10 бит	0x03FF	На вершине	На вершине
8	1	0	0	0	Фазовый и частотный ШИМ	ICR1	Внизу	Минимум
9	1	0	0	1	Фазовый и частотный ШИМ	OCR1A	Внизу	Минимум
10	1	0	1	0	Фазовый ШИМ	ICR1	На вершине	Минимум
11	1	0	1	1	Фазовый ШИМ	OCR1A	На вершине	Минимум
12	1	1	0	0	Очистка при совпадении	ICR1	Немедленно	Максимум
13	1	1	0	1	–	–	–	–
14	1	1	1	0	Быстрый ШИМ	ICR1	На вершине	На вершине
15	1	1	1	1	Быстрый ШИМ	OCR1A	На вершине	На вершине

Биты 3…0 CS12…CS10. Эти биты определяют источник задания частоты таймера T1 и предделитель таймера (табл. 6).

Табл. 6. Функции бит CS02, CS01, CS00 таймера T0

CS02	CS01	CS00	Пояснение
0	0	0	Нет источника. Таймер остановлен.
0	0	1	Предделитель $f_{CLK}/1$
0	1	0	Предделитель $f_{CLK}/8$
0	1	1	Предделитель $f_{CLK}/64$

CS02	CS01	CS00	Пояснение
1	0	0	Предделитель $f_{CLK}/256$
1	0	1	Предделитель $f_{CLK}/1024$
1	1	0	Внешний сигнал на выводе T0. (по спадающему фронту)
1	1	1	Внешний сигнал на T0. (по нарастающему фронту)

Помимо управляющих регистров **TCCR1A** и **TCCR1B**, счетного регистра **TCNT1** и регистров сравнения **OCR1A**, **OCR1B**, в микроконтроллере содержатся регистры флагов и масок прерываний таймеров соответственно **TIFR** и **TIMSK**.

Значение этих регистров следующее. Когда возникает совпадение или переполнение таймера T1, автоматически формируются флаги прерываний **OCF1A**, **OCF1B** или TOV1 в регистре **TIFR**. Если в регистре масок прерываний таймеров **TIMSK** на соответствующее прерывание наложена маска, то вызывается процедура обработки прерывания. Если же маска прерывания не наложена, то при совпадении или переполнении таймера ничего не происходит.

Поскольку регистры **TIMSK** и **TIFR** – общие для всех таймеров, они содержат все флаги и маски таймеров микроконтроллера.

Табл. 7. Описание регистра TIFR

Бит	7	6	5	4	3	2	1	0
Название	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0

В регистре **TIFR** (табл. 7) бит 4 OCF1A – флаг прерывания по совпадению канала A таймера T1; бит 3 OCF1B – флаг прерывания по совпадению канала B таймера T1; бит 2 TOV1 – флаг прерывания по переполнению таймера T1.

Табл. 8. Описание регистра TIMSK

Бит	7	6	5	4	3	2	1	0
Название	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0

В регистре **TIMSK** (табл. 7) бит 4 OCIE1A – маска прерывания по совпадению канала A таймера T1; бит 3 OCIE1B – маска прерывания по совпадению канала B таймера T1; бит 2 TOIE1 – маска прерывания по переполнению таймера T1.

Таймер T1 содержит также шестнадцатиразрядный регистр захвата внешнего сигнала **ICR1H:ICR1L** (input capture register). При использовании в качестве счетного внешнего источника сигнала на входе T1 (PB1) значение этого регистра будет обновляться по каждому заданному событию на входе T1. При этом в него будет записываться значение регистра счета **TCNT1**, содержащееся в данный момент в таймере. Этот режим полезен при подсчете длительности внешних сигналов.

Пример. На базе таймера T1 создать программу работы светофора по следующему алгоритму: горит зеленый сигнал (15 секунд) → мигает зеленый сигнал (5 секунд) → горит желтый сигнал (5 секунд) → горит красный сигнал (15 секунд) → горят красный и желтый сигналы (5 секунд). Далее процесс повторяется.

```
;-----------------------------------------------------------------
;Программа работы светофора.
;Выходы:
;PC0 - зеленый
;PC1 - желтый
;PC2 - красный

.include"m8535def.inc"   ;Подключение библиотеки контроллера Atmega8535
.cseg                    ;Начало сегмента кода - по адресу 0
.org$0
rjmp reset               ;Переход на метку reset
.org$06                  ;При наступлении прерывания OC1A
rjmp T1A_compare         ;переход на метку T1A_compare

reset:                   ;При переходе на reset:
cli                      ;глобальный запрет прерываний
ldi r16,low(RAMEND)      ;запись в регистр указателя стека адреса
ldi r17,high(RAMEND)     ;конца памяти данных
out spl,r16
out sph,r17
clr r16                  ;инициализация портов ввода/вывода
out PORTC,r16
ldi r16,0x07
out DDRC,r16
clr r16                  ;инициализация таймера T1
out TCCR1B,r16
out TCCR1A,r16
out TCNT1H,r16
out TCNT1L,r16
ldi r16,0x1E
out OCR1AH,r16
ldi r16,0x84
out OCR1AL,r16
clr r16
out TCCR1A,r16
ldi r16,0x05
out TCCR1B,r16
ldi r16,0x10
out TIMSK,r16            ;установка маски прерывания по совпадению T1A
clr r16                  ;очистка необходимых регистров
clr r17                  ;общего назначения
clr r18
sbi PORTC,0              ;включение зеленого сигнала
sei                      ;глобальное разрешение прерываний

main:                    ;основной цикл выполнен пустым
rjmp main

T1A_compare:             ;При наступлении прерывания T1CA
inc r17                  ;инкремент счетчика прерываний
clr r18                  ;обнуление счетного регистра таймера T1
out TCNT1H,r18
out TCNT1L,r18
cpi r16,0x00             ;если r16=0x00, горит зеленый (15 сек)
```

Хусаинов Р.З., Качалов А.В. Программирование микроконтроллера Atmega 8535 на ассемблере: Методические указания к выполнению лабораторных работ. Челябинск, Учтех-Профи, 2013.

Методические указания предназначены для студентов средних и высших учебных заведений, изучающих дисциплины по архитектуре и программированию микропроцессорных систем. Методические указания также могут быть использованы для обучения учащихся профессионально-технических училищ и слушателей отраслевых учебных центров повышения квалификации инженерно-технических работников.

```
breq m1                 ;осуществляется переход на m1
cpi r16,0x01            ;если r16=0x01, мигает зеленый (5 сек)
breq m2                 ;осуществляется переход на m2
cpi r16,0x02            ;если r16=0x02, горит желтый (5 сек)
breq m3                 ;осуществляется переход на m3
cpi r16,0x03            ;если r16=0x03, горит красный (15 сек)
breq m4                 ;осуществляется переход на m4
cpi r16,0x04            ;если r16=0x04, горят красный и желтый (5 сек)
breq m5                 ;осуществляется переход на m5

m1:                     ;По метке m1
cpi r17,0x0F            ;сравнение счетчика прерываний r17 с числом 15
brne quit               ;если r17<15, переход на quit
clr r17                 ;иначе очистка r17
cbi PORTC,0             ;выключение зеленого сигнала
inc r16                 ;включение режима мигания зеленого сигнала
rjmp quit               ;и переход на quit

m2:                     ;По метке m2
in r18,PINC             ;считывание состояния порта C
andi r18,0x01           ;и выделение бита PC0
cpi r18,0x01            ;если результат равен «1»,
breq m2_1               ;то переход на метку m2_1
sbi PORTC,0             ;иначе установка бита PC1
rjmp m2_2               ;и переход на метку m2_2
m2_1:                   ;По метке m2_1
cbi PORTC,0             ;очистка бита PC0
m2_2:                   ;По метке m2_2
cpi r17,0x05            ;сравнение счетчика прерываний r17 с числом 5
brne quit               ;Если r17<5, то переход на quit
clr r17                 ;иначе очистка r17
ldi r18,0x02            ;выключение PC0 и установка PC1 (желтый)
out PORTC,r18
inc r16                 ;включение режима работы желтого сигнала
rjmp quit               ;и переход на quit

m3:                     ;По метке m3
cpi r17,0x05            ;сравнение счетчика прерываний r17 с числом 5
brne quit               ;если r17<5, то переход на quit
clr r17                 ;иначе очистка r17
ldi r18,0x04            ;выключение PC1 и включение PC2 (красный).
out PORTC,r18
inc r16                 ;включение режима работы красного сигнала
rjmp quit               ;и переход на quit

m4:                     ;По метке m4
cpi r17,0x0F            ;сравнение счетчика прерываний r17 с числом 15
brne quit               ;если r17<15, то переход на quit
clr r17                 ;иначе очистка r17
ldi r18,0x06            ;включение PC2 (красный) и PC1 (желтый).
out PORTC,r18
inc r16                 ;включение режима работы красного и желтого
rjmp quit               ;сигналов и переход на quit
```

```
m5:                     ;По метке m5
cpi r17,0x05            ;сравнение счетчика прерываний r17 с числом 5
brne quit               ;если r17<5, то переход на quit
clr r17                 ;иначе очистка r17
ldi r18,0x01            ;включение зеленого сигнала
out PORTC,r18
clr r16                 ;возврат r16 в исходное состояние
quit:                   ;По метке quit
reti                    ;осуществляется выход из прерывания.
```

Рассмотрим программу более подробно.

1. Сначала подключается стандартная библиотека контроллера Atmega8535:

```
.include"m8535def.inc"
```

2. Затем инициализируется адрес начала сегмента кода при выполнении основного цикла программы (адрес 0) и при наступлении прерывания по совпадению канала А таймера Т1 (адрес 6):

```
.cseg
.org$0
rjmp reset
.org$06
rjmp T1A_compare
```

При переходе программы по адресу 0 происходит переход на метку reset (`rjmp reset`). При наступлении прерывания по совпадению канала А таймера Т1 происходит переход на метку T1A_compare (`rjmp T1A_compare`).

3. При переходе на метку reset осуществляется инициализация стека, портов ввода/вывода и таймера Т1:

```
reset:
cli
ldi r16,low(RAMEND)
ldi r17,high(RAMEND)
out spl,r16
out sph,r17
clr r16
out PORTC,r16
ldi r16,0x07
out DDRC,r16
```

Сначала инициализируется указатель стека. Для этого в регистры SPH:SPL записывается адрес вершины стека, находящейся в конце памяти данных. После этого производится инициализация порта ввода/вывода **PORTC**. В данном случае используются три младших бита, которые инициализируются на вывод информации (`ldi r16,0x07; out DDRC,r16`).

Инициализация таймера Т1:

```
clr r16
out TCCR1B,r16
out TCCR1A,r16
out TCNT1H,r16
out TCNT1L,r16
ldi r16,0x1E
out OCR1AH,r16
ldi r16,0x84
out OCR1AL,r16
clr r16
```

```
out TCCR1A,r16
ldi r16,0x05
out TCCR1B,r16
ldi r16,0x10
out TIMSK,r16
```

Производится остановка и обнуление таймера очисткой его регистров управления **TCCR1A**, **TCCR1B** и регистра счета **TCNT1H:TCNT1L**. Заранее необходимо рассчитать необходимую уставку таймера.

Принимаем, что таймер должен срабатывать с частотой 1 Гц (раз в секунду). Ориентируясь по данным, представленным в табл.6, можно увидеть, что в таймере доступны следующие предделители частоты: div = 1, 8, 64, 256, 1024. При предделителе div=1 необходимое число, которое требуется записать в регистр сравнения канала А для выдержки времени T=1 секунда:

$$N = \frac{T}{\left[\frac{f_{CLK}}{div}\right]^{-1}} = \frac{1}{\left[\frac{8 \cdot 10^6}{1}\right]^{-1}} = 8\,000\,000.$$

Поскольку максимально число, которое можно записать в регистр **TCNT1H:TNCT1L**, составляет $N_{MAX} = 2^{16} = 65536$, число N невозможно записать в регистр сравнения. При использовании делителей частоты N принимает следующие значения:

– при делителе clk/8 N=1 000 000;
– при делителе clk/64 N=125 000
– при делителе clk/256 N=31 250;
– при делителе clk/1024 N=7 812;

В программе выбран предделитель div=1024. При этом предделителе число, которое необходимо записать в регистр сравнения:

$$N = \frac{T}{\left[\frac{f_{CLK}}{div}\right]^{-1}} = \frac{1}{\left[\frac{8 \cdot 10^6}{1024}\right]^{-1}} = 7812 = 0x1E84.$$

Это число заносится в регистр сравнения канала А таймера Т1 (ldi r16,0x1E; out OCR1AH,r16; ldi r16,0x84; out OCR1AL,r16). Далее в таймере инициализируется режим счета временных интервалов (табл. 5, режим 0) и производится запуск таймера установкой регистра **TCCR1B**. Дополнительно в регистре **TIMSK** устанавливается маска на прерывание по совпадению канала А таймера Т1 (`ldi r16,0x10; out TIMSK,r16`).

После установки всех устройств необходимые в дальнейшем РОН очищаются (`clr r16; clr r17;clr r18`) и включается зеленый сигнал светофора (`sbi PORTC,0`).

```
clr r16
clr r17
clr r18
sbi PORTC,0
sei
```

Командой `sei` производится глобальное разрешение прерываний. Основной цикл программы выполнен пустым, он замкнут сам на себя:

```
main:
rjmp main
```

4. При наступлении прерывания по совпадению канала А таймера Т1 происходит переход на метку `T1A_compare`. В подпрограмме по обработке прерывания введен

счетчик прерываний, реализованный на РОН r17. Этот счетчик производит подсчет срабатывания таймера и таким образом контролирует заданное время работы каждого сигнала светофора. Поскольку таймер настроен на частоту 1 Гц, то есть он срабатывает 1 раз в секунду, то, например, для выдержки времени 5 секунд значение, накопленное в r17, равняется 5.

```
T1A_compare:
inc r17
clr r18
out TCNT1H,r18
out TCNT1L,r18
cpi r16,0x00
breq m1
cpi r16,0x01
breq m2
cpi r16,0x02
breq m3
cpi r16,0x03
breq m4
cpi r16,0x04
breq m5
```

При срабатывании таймера счетчик прерываний r17 инкрементируется (`inc r17`), а содержимое счетного регистра таймера каждый раз очищается (`clr r18; out TCNT1H,r18; out TCNT1L,r18`). В программе регистр r16 служит для выбора режима работы светофора:

– если r16=0, то горит зеленый сигнал светофора;
– если r16=1, то зеленый сигнал светофора мигает;
– если r16=2, то горит желтый сигнал светофора;
– если r16=3, то горит красный сигнал светофора;
– если r16=4, то горит красный и желтый сигналы светофора.

В соответствии с вышеизложенным в программе происходит переход на соответствующие метки m1 … m5.

5. При переходе на метку m1 программа переключается на режим работы зеленого сигнала светофора:

```
m1:
cpi r17,0x0F
brne quit
clr r17
cbi PORTC,0
inc r16
rjmp quit
```

Сначала происходит сравнение РОН r17 с числом 15=0x0F, которое соответствует 15-секундной выдержке времени (`cpi r17,0x0F`). Если это число не достигнуто, ничего не происходит, а программа переходит на метку quit (`brne quit`). При достижении r17=15 РОН очищается (`clr r17`), зеленый сигнал выключается (`cbi PORTC,0`), а программа переходит в режим мигания зеленого сигнала (`inc r16`).

6. При переходе на метку m2 программа переключается на мигающий режим работы зеленого сигнала светофора:

```
m2:
```

```
in r18,PINC
andi r18,0x01
cpi r18,0x01
breq m2_1
sbi PORTC,0
rjmp m2_2
m2_1:
cbi PORTC,0
m2_2:
cpi r17,0x05
brne quit
clr r17
ldi r18,0x02
out PORTC,r18
inc r16
rjmp quit
```

Сначала происходит опрос состояния зеленого сигнала светофора (`in r18,PINC; andi r18,0x01`). Если он горит, то сигнал гаснет (`cbi PORTC,0`), иначе – загорается (`sbi PORTC,0`). После этого происходит проверка времени работы светофора в режиме мигания зеленого сигнала, подобная описанной выше для режима работы зеленого сигнала.

7. При переходе на метку m3 программа переключается на режим работы желтого сигнала светофора:

```
m3:
cpi r17,0x05
brne quit
clr r17
ldi r18,0x04
out PORTC,r18
inc r16
rjmp quit
```

8. При переходе на метку m4 программа переключается на режим работы красного сигнала светофора:

```
m4:
cpi r17,0x0F
brne quit
clr r17
ldi r18,0x06
out PORTC,r18
inc r16
rjmp quit
```

9. При переходе на метку m5 программа переключается на режим работы красного и желтого сигналов светофора:

```
m5:
cpi r17,0x05
brne quit
clr r17
ldi r18,0x01
out PORTC,r18
clr r16
```

По окончании выдержки времени работы данного режима РОН r16 очищается и цикл работы светофора замыкается.

10. По метке quit происходит выход из подпрограммы обработки прерывания по совпадению канала A таймера T1:

```
quit:
reti
```

Работа № 3. 16-разрядный таймер Т1. Режим широтно-импульсной модуляции.

Цель работы

Освоить теоретический и практический материал по работе 16-разрядного таймера Т1 микроконтроллера Atmega 8535 в режиме широтно-импульсной модуляции. Применить приобретенные навыки при написании программы.

Программа работы

1. Изучить необходимый теоретический материал о регистрах и функционировании таймера Т1 в режиме ШИМ.
2. Разобраться в программе по использованию таймера Т1, представленной в лабораторной работе.
3. Написать и отладить собственную программу с использованием таймера по заданию преподавателя.

Пояснения к работе

Таймер Т1 микроконтроллера Atmega8535 может работать как в режиме подсчета временных интервалов, так и в режиме широтно-импульсной модуляции (ШИМ).

Поскольку таймер Т1 содержит два канала и два регистра сравнения OCR1A и OCR1B, то он может сформировать два ШИМ-сигнала одновременно. Для этого два канала таймера подключены к соответствующим выводам микроконтроллера. Так, к выходу T1A таймера подключен вывод PB5, а к выходу канала T1B таймера Т1 подключен вывод PD4 микроконтроллера.

В микроконтроллере Atmega 8535 таймер Т1, как и таймеры Т0 и Т2, работает в двух режимах широтно-импульсной модуляции: быстрый ШИМ и фазовый ШИМ.

В режиме быстрого ШИМ счетный регистр **TCNT1** таймера производит формирование пилообразной развертки (рис. 1), инкрементируя свое значение по каждому импульсу с предделителя, который устанавливается в регистре управления **TCCR1B** таймера. При достижении счетным регистром максимального значения, определяемого в настройках регистров управления **TCCR1A**, **TCCR1B** происходит его автоматическое обнуление.

В режиме фазового ШИМ обеспечивается ШИМ с высокой разрешающей способностью. Отличительной особенностью режима является сигнал развертки с двумя пологими фронтами – нарастающим и спадающим. Счетчик непрерывно производит счет импульсов с выхода предделителя таймера от минимального до максимального значения, а затем – от максимального до минимального. Принцип работы схемы формирования ШИМ пояснен на примере канала А таймера на рис. 2.

При неинвертирующем ШИМ соответствующий выход микроконтроллера (PD4 – для канала сравнения B, PD5 – для канала сравнения A таймера) очищается при совпадении уставки, записанной в регистрах сравнения **OCR1A**, **OCR1B**, с величиной сигнала развертки, формируемой в регистре **TCNT1** при счете от минимального до максимального значения и устанавливается при счете от

максимального до минимального значения (рис. 2). В случае инвертирующего ШИМ логика переключения вывода микроконтроллера инверсная.

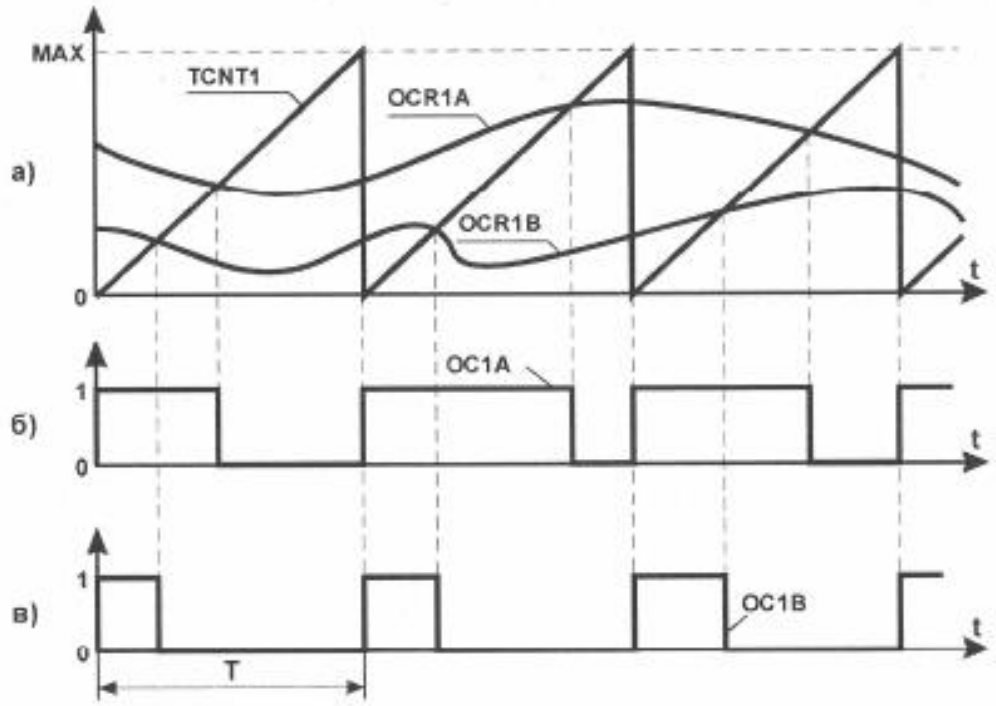


Рис. 1. Принцип работы таймера Т1 в режиме «Быстрый ШИМ»

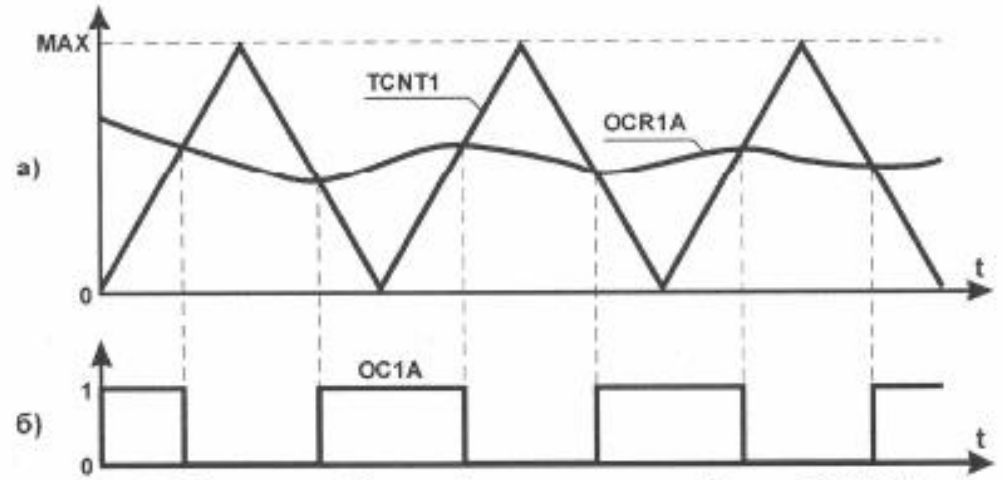


Рис. 2. Принцип работы таймера в режиме «Фазовый ШИМ»

Фазовый ШИМ, реализованный на таймере Т1 чрезвычайно полезен при практическом применении микроконтроллера в качестве устройства для управления, например, транзисторными преобразователями напряжения, когда необходимо тщательно выдерживать паузу между выключением одного сигнала и включением другого. Это требование легко реализуется использованием двух каналов сравнения таймера (рис. 3).

Канал сравнения А таймера инициализируется на неинвертирующий режим ШИМ, а канал сравнения В – на инвертирующий. В этом случае выходы OC1A и

OC1B микроконтроллера будут работать в противофазе. Для реализации паузы Δt между изменением состояния каналов сравнения A и B в регистры сравнения **OCR1A** и **OCR1B** необходимо занести уставки срабатывания, отличающиеся друг от друга на небольшую величину, соответствующую требуемой временной задержке (рис. 3).

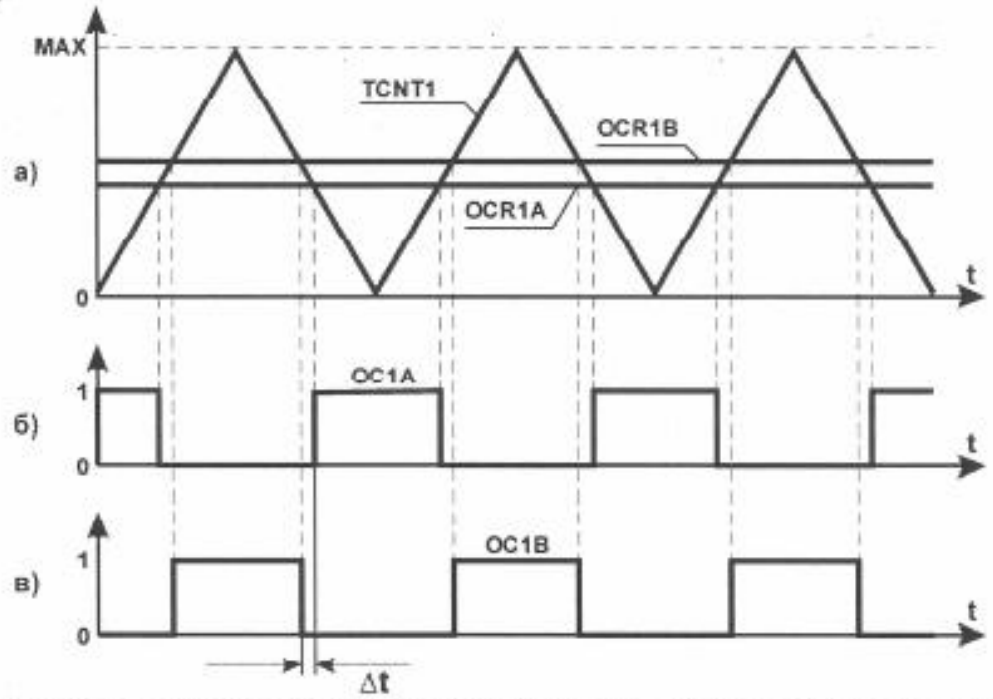


Рис. 3. Пример реализации задержки между двумя сигналами при использовании фазового ШИМ таймера T1

Разновидностью фазового ШИМ является частотно-фазовый ШИМ. Отличие от фазового ШИМ в этом режиме заключается в моменте обновления регистров сравнения OCR1A и OCR1B. Если в режиме фазового ШИМ обновление регистров происходит при достижении счетным регистром TCNT1 максимального значения, то при использовании частотно-фазового ШИМ обновление регистров происходит при достижении счетным регистром минимального значения (рис. 4). Это позволяет получить в пределах периода T ШИМ одинаковые по длительности участки включенного или выключенного состояния выходов OC1A, OC1B микроконтроллера $t1=t3$ (рис. 4).

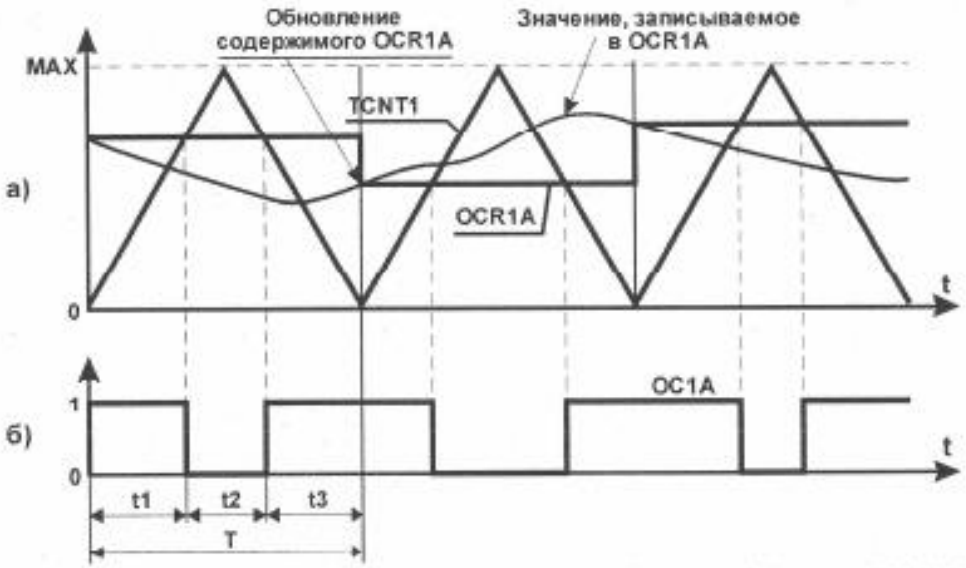


Рис. 4. Принцип работы таймера в режиме частотно-фазового ШИМ

Для управления режимами работы таймера используются регистры управления **TCCR1A, TCCR1B**.

Регистр управления **TCCR1A** состоит из управляющих бит, пояснение которых приведено в табл. 1.

Табл. 1. Регистр управления таймера T1 TCCR1A

Бит	7	6	5	4	3	2	1	0
Название	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10

Биты 7 и 6 COM1A1 и COM1A0. В режиме ШИМ эти биты служат для управления выводом OC1A микроконтроллера (PD5). Комбинацией управляющих бит можно либо отключить вывод от таймера, либо инициализировать инвертирующий или неинвертирующий режим ШИМ.

Табл. 2. Функции вывода OC1A таймера T1 в режиме ШИМ

COM1A1	**COM1A0**	**Пояснение**
0	0	Вывод OC1A отключен от таймера
0	1	При установке бита WGM13 в регистре TCCR1B WGM13=0 вывод отключен от таймера. При WGM13=1 изменение состояния OC1A на противоположное при совпадении.
1	0	Неинвертирующий ШИМ. Очистка вывода при совпадении при счете вверх и установка при совпадении при счете вниз.
1	1	Инвертирующий ШИМ. Установка вывода при совпадении при счете вверх и обнуление при совпадении при счете вниз.

Биты 5 и 4 COM1B1 и COM1B0. Назначение этих бит аналогично вышеописанному для бит COM1A1, COM1A0. В режиме COM1B1=0, COM1B0=1 вывод отключен от таймера.

ОГЛАВЛЕНИЕ

Биты 3 и 2 FOC1A и FOC1B. Эти биты действуют только в обычном режиме работы при подсчете временных интервалов. При записи в них логической «1» происходит срабатывание прерывания по совпадению соответствующего канала таймера и изменение выводов таймера OC1A и OC1B в соответствии с настройками бит COM1A1:COM1A0, COM1B1:COM1B0.

Биты 1 и 0 WGM11, WGM10. Эти биты определяют режим работы таймера вместе с управляющими битами регистра TCCR1B.

Регистр управления **TCCR1B** состоит из управляющих бит, пояснение которых приведено в табл. 3.

Табл. 3. Регистр управления таймера T1 TCCR1B

Бит	7	6	5	4	3	2	1	0
Название	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10

Бит 7 – ICNC1. Этот бит включает функцию фильтрации внешнего сигнала, поступающего на вывод PB1 микроконтроллера, когда таймер используется в качестве счетчика. Установка бита увеличивает надежность работы таймера. При активации этого бита между подачей сигнала на вывод PB1 и срабатыванием схемы счета проходит 4 периода тактовой частоты.

Бит 6 – ICES1. Этот бит определяет активный фронт сигнала на входе PB1. При ICES1=0 таймер срабатывает по спадающему фронту сигнала, иначе – по нарастающему фронту.

Бит 5. Зарезервирован и не используется.

Биты 4 и 3 – WGM13, WGM12. Эти биты вместе с битами WGM11, WGM 10 регистра TCCR1A определяют режим работы таймера (табл. 4).

Табл. 4. Задание режима работы таймера T1

Режим	**WGM13**	**WGM12**	**WGM11**	**WGM10**	**Название**	**Вершина**	**Обновление OCR1x**	**Переполнение TOV1**
0	0	0	0	0	Нормальный режим	0xFFFF	Немедленно	Максимум
1	0	0	0	1	Фазовый ШИМ, 8 бит	0x00FF	На вершине	Минимум
2	0	0	1	0	Фазовый ШИМ, 9 бит	0x01FF	На вершине	Минимум
3	0	0	1	1	Фазовый ШИМ, 10 бит	0x3FFF	На вершине	Минимум
4	0	1	0	0	Очистка при совпадении	OCR1A	Немедленно	Максимум
5	0	1	0	1	Быстрый ШИМ, 8 бит	0x00FF	На вершине	На вершине
6	0	1	1	0	Быстрый ШИМ, 9 бит	0x01FF	На вершине	На вершине
7	0	1	1	1	Быстрый ШИМ, 10 бит	0x03FF	На вершине	На вершине
8	1	0	0	0	Частотно-фазовый ШИМ	ICR1	Внизу	Минимум
9	1	0	0	1	Частотно-фазовый ШИМ	OCR1A	Внизу	Минимум
10	1	0	1	0	Фазовый ШИМ	ICR1	На вершине	Минимум
11	1	0	1	1	Фазовый ШИМ	OCR1A	На вершине	Минимум
12	1	1	0	0	Очистка при совпадении	ICR1	Немедленно	Максимум
13	1	1	0	1	–	–	–	–
14	1	1	1	0	Быстрый ШИМ	ICR1	На вершине	На вершине
15	1	1	1	1	Быстрый ШИМ	OCR1A	На вершине	На вершине

Биты 3…0 CS12…CS10. Эти биты определяют источник задания частоты таймера T1 и предделитель таймера (табл. 5).

Табл. 5. Функции бит CS02, CS01, CS00 таймера T0

CS02	CS01	CS00	Пояснение
0	0	0	Нет источника. Таймер остановлен.
0	0	1	Предделитель $f_{CLK}/1$
0	1	0	Предделитель $f_{CLK}/8$
0	1	1	Предделитель $f_{CLK}/64$
1	0	0	Предделитель $f_{CLK}/256$
1	0	1	Предделитель $f_{CLK}/1024$
1	1	0	Внешний сигнал на выводе T0. (по спадающему фронту)
1	1	1	Внешний сигнал на T0. (по нарастающему фронту)

В режиме широтно-импульсной модуляции нет необходимости инициализировать прерывания по совпадению или переполнению таймера T1, поэтому описание регистров TIFR и TIMSK в данной работе не приводится (см. лабораторную работу №7).

Пример. Перевести таймер T1 в режим 10-битного ШИМ и управлять скоростью вращения электродвигателя постоянного тока модуля «Микроконтроллер». Предусмотреть 8 скоростей вращения электродвигателя, сигнал задания задавать дискретно с помощью тумблеров, подключенных к порту ввода/вывода A. Организовать предустановленные скорости необходимо в виде таблицы, хранящейся во FLASH-памяти контроллера.

```
//-----------------------------------------------------------------
;Программа для управления электродвигателем постоянного тока.
;Входы:
;         PA0…PA2 - дискретные сигналы задания скорости
;Выходы:
          PD5 - ШИМ-сигнал на выходе микроконтроллера
.include"m8535def.inc"   ;подключение стандартной библиотеки Atmega8535
.cseg                    ;начало сегмента кода
.org$0                   ;По адресу 0
rjmp reset               ;осуществляется переход на метку reset
.org$100                 ;По адресу 100
.db 0x00, 0x7F, 0x00, 0xFE, 0x01, 0x7D, 0x01, 0xFC, 0x02, 0x7B, 0x02,
0xFA, 0x03, 0x79, 0x03, 0xF8       ;осуществляется запись скоростей
                                   ;электродвигателя. Каждая скорость
                                   ;занимает 2 байта памяти.
reset:                   ;По метке reset происходит:
ldi r16,low(RAMEND)      ;Запись адреса вершины стека
ldi r17,high(RAMEND)     ;В конце памяти данных
out spl,r16
out sph,r17
ldi r16,0x00             ;Инициализация портов ввода/вывода
out DDRA,r16             ;Порт A - на ввод информации
ldi r16,0xFF
out PORTA,r16
out DDRD,r16             ;Порт D   на вывод информации.
clr r16
```

```
out OCR1AH,r16          ;Установка в таймере T1 быстрого ШИМ
out OCR1AL,r16          ;в 10-битном режиме.
out TCNT1H,r16
out TCNT1L,r16
ldi r16,0xC3
out TCCR1A,r16
ldi r16,0x1A
out TCCR1B,r16

main:                   ;по метке main
in r18,PINA             ;происходит считывание данных, поступающих на
andi r18,0x07           ;порт A и выделение 3-х младших бит
lsl r18                 ;и формирование в соответствии с комбинацией
mov r30,r18             ;этих бит адреса FLASH для считывания данных.
ldi r31,0x02            ;Передача адреса FLASH в Z-регистр контроллера
lpm r17,Z+              ;Считывание старшего байта данных из FLASH
lpm r16,Z               ;Считывание младшего байта данных из FLASH
out OCR1AH,r17          ;Передача считанных данных в регистр сравнения
out OCR1AL,r16          ;канала A таймера T1
rjmp main               ;Возврат на метку main
//---------------------------------------------------------------
```

Рассмотрим программу более подробно.

1. Сначала происходит подключение библиотеки контроллера Atmega 8535. Далее по адресу 0 осуществляется переход на метку reset, по которой начинается программа:

```
.include"m8535def.inc"
.cseg
.org$0
rjmp reset
```

2. По адресу 100 FLASH-памяти осуществляется запись предустановленных скоростей электродвигателя, в качестве которых выступает значение регистра сравнения OCR1A таймера T1:

```
.org$100
.db 0x00, 0x7F, 0x00, 0xFE, 0x01, 0x7D, 0x01, 0xFC, 0x02, 0x7B, 0x02,
0xFA, 0x03, 0x79, 0x03, 0xF8
```

Память разделена на страницы, в которых содержатся 2 байта информации – старший и младший. В случае, если необходимо считать один байт, необходимо умножать адрес страницы на 2. Так, если по адресу $100 содержатся два байта: $100(H) – 0x00 и $100(L) – 0x7F, то каждый из этих байт имеет следующий адрес: $200 – 0x00 и $201 – 0x7F.

Во FLASH – памяти записаны следующие ставки скорости:

- скорость 1: 0x007F;
- скорость 2: 0x00FE;
- скорость 3: 0x017D;
- скорость 4: 0x01FC;
- скорость 5: 0x027B;
- скорость 6: 0x02FA;
- скорость 7: 0x0379;
- скорость 8: 0x03F8.

3. По метке reset осуществляется инициализация портов ввода/вывода и таймера/счетчика T1. В соответствии с заданием порт ввода/вывода A инициализируется на ввод, а порт D – на вывод информации:

```
reset:
ldi r16,low(RAMEND)
ldi r17,high(RAMEND)
out spl,r16
out sph,r17
ldi r16,0x00
out DDRA,r16
ldi r16,0xFF
out PORTA,r16
out DDRD,r16
clr r16
out OCR1AH,r16
out OCR1AL,r16
out TCNT1H,r16
out TCNT1L,r16
ldi r16,0xC3
out TCCR1A,r16
ldi r16,0x1A
out TCCR1B,r16
```

Таймер/счетчик T1 инициализируется на режим быстрого ШИМ, при этом задействуется канал сравнения OCR1A, вывод которого работает в режиме неинвертирующего ШИМ.

4. По метке main осуществляется переход на основной цикл программы. В этом цикле осуществляется опрос порта ввода/вывода A, формирование адреса FLASH-памяти в соответствии с комбинацией управляющих входов, а также передача данных из FLASH в регистры сравнения таймера T1:

```
main:
in r18,PINA
andi r18,0x07
lsl r18
mov r30,r18
ldi r31,0x02
lpm r17,Z+
lpm r16,Z
out OCR1AH,r17
out OCR1AL,r16
rjmp main
```

Сначала осуществляется считывание данных порта A (`in r18,PINA`) и выделение трех младших бит считанных данных (`andi r18,0x07`). Считанные данные преобразуются в адрес FLASH памяти контроллера. Работа с FLASH-памятью контроллера реализуется через специальный Z-регистр, состоящий из двух РОН r31(H):r30(L). Для чтения из FLASH необходимо в Z-регистр записать адрес в памяти контроллера, а затем специальной командой lpm считать данные из памяти по указанному адресу.

Поскольку начало массива с предустановленными скоростями соответствует адресу $100, то в регистр Z при необходимости считывания данных из FLASH необходимо записывать адреса от $200 до $20F. По этой причине в регистр r31 записывается

число 0x02, а в регистр r30 записывается адрес, формируемый умножением кода, считанного из PINA, на число 2 (`lsl r18`). Таким образом, при комбинации PINA=0000 0000 в Z-регистр будет записан адрес 0x0200, при комбинации PINA=0000 0001 в Z-регистр будет записан адрес 0x0202 и т.д. То есть адрес FLASH будет всегда указывать на старший байт предустановленной скорости.

При считывании данных из FLASH командой `lpm r17,Z+` в регистр r17 записывается значение FLASH по указанному адресу, то есть старший байт уставки по скорости, а затем адрес FLASH автоматически инкрементируется и при следующем обращении к FLASH командой `lpm r16,Z` в регистр r16 будет записано уже содержимое памяти по адресу, указанному в Z+1, то есть младший байт уставки по скорости.

После считывания данных из памяти они передаются в регистр сравнения канала A таймера T1 (`out OCR1AH,r17;out OCR1AL,r16`). После этого процесс повторяется (`rjmp main`).

Работа № 4. Динамическая индикация символов

Цель работы

Освоить теоретический и практический материал по реализации динамической индикации с использованием микроконтроллера Atmega 8535. Применить приобретенные навыки при написании программы.

Программа работы

1. Изучить необходимый теоретический материал о принципах индикации символов на светодиодном семисегментном индикаторе.
2. Разобраться в программах, представленных в лабораторной работе.
3. Написать и отладить собственную программу динамической индикации.

Пояснения к работе

В настоящее время подавляющее число промышленных приборов оснащаются цифровыми индикаторами для отображения различных величин. В данной работе рассматриваются методы работы с семисегментными индикаторами и рассматриваются примеры программ.

Простейший семисегментный индикатор (рис. 1) представляет набор отдельных сегментов (светодиодов), при зажигании которых в определенной последовательности можно получить набор цифр и определенных символов.

Каждый сегмент индикатора имеет унифицированное буквенное обозначение в соответствии с латинским алфавитом. Верхний горизонтальный сегмент имеет обозначение «A», все последующие сегменты по часовой стрелке имеют обозначения «B», «C», «D», «E», «F», «G», «H» (рис. 1).

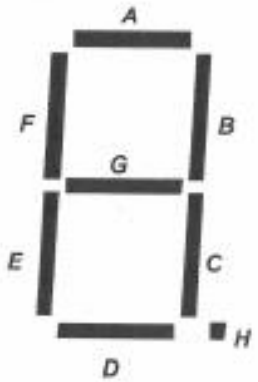
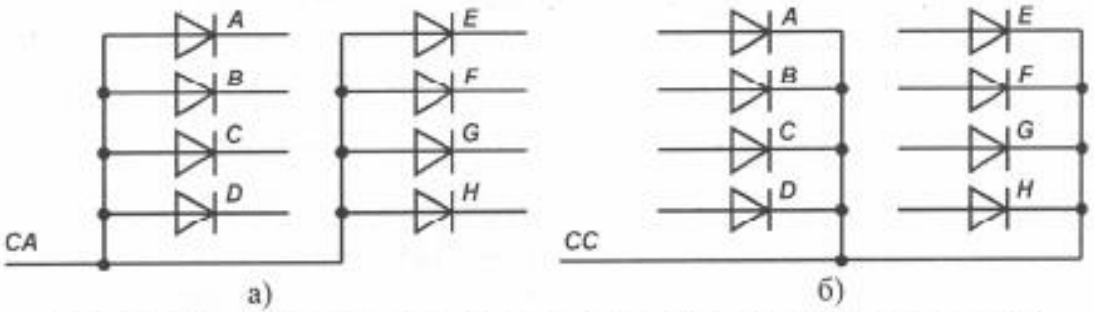


Рис. 1. Внешний вид и структура семисегментного индикатора

Для того, чтобы зажечь сегмент индикатора, необходимо подать напряжение на соответствующий светодиод A … H. Семисегментные индикаторы выпускаются двух типов – с общим анодом (рис. 2, а) и с общим катодом (рис. 2, б). Это делается для упрощения работы с индикатором и уменьшения количества выводов его

микросхемы. Действительно, при использовании схемы с общим анодом аноды всех светодиодов объединяются, на них подается положительное напряжение. Для того, чтобы зажечь, например, сегмент «С», необходимо катод соответствующего светодиода через токоограничивающий резистор присоединить к общему проводу.



а) б)

Рис. 2. Семисегментные индикаторы с общим анодом и общим катодом

При использовании индикаторов с общим катодом на него подключается шина с нулевым потенциалом, а на аноды светодиодов через токоограничивающие элементы подключается напряжение питания.

Таким образом, для того, чтобы зажечь, например, цифру 3, необходимо осуществить подачу напряжения на светодиоды A, B, C, D, G.

На практике для упрощения работы с индикаторами применяют схемы промежуточного усиления, предназначенные для усиления сигналов управления индикаторами и удобного управления их сегментами. Пример такой схемы приведен на рис. 3.

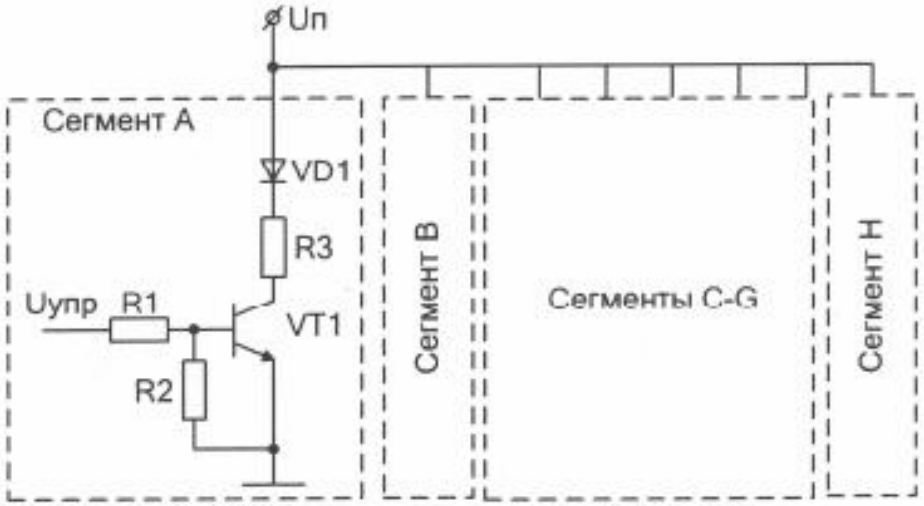


Рис. 3. Схема для управления индикатором

На общие аноды сегментов подается напряжение электропитания $U_п$. Катоды сегментов через токоограничивающий резистор и транзистор подключаются к общей шине. Очевидно, что для зажигания сегмента необходимо включить транзистор (для сегмента A – транзистор VT1)

Включение транзистора VT1 осуществляется подачей на его базу тока управления, который появляется при подаче напряжения управления $U_{УПР}$ на токоограничивающий резистор R1. Таким образом, при подаче от микроконтроллера сигнала логической «1» транзистор VT1 открывается и светодиод VD1 начинает светиться. Применение рассмотренной схемы позволяет включать сегменты сигналами логической «1», а не логического «0».

Пример 1. Написать программу, осуществляющую вывод на индикатор числа от 0 до F в шестнадцатеричном коде в соответствии с комбинацией сигналов на входе порта ввода/вывода A.

```
//-----------------------------------------------------------------------
;Программа вывода чисел 0…F на один разряд семисегментного индикатора
;Входы:
;     PA3…PA0 - задание кода числа
;Выходы:
;     PD0 - управление подачей напряжения на индикатор
;     PC7…PC0   - управление сегментами индикатора

.include"m8535def.inc"   ;Подключение библиотеки Atmega8535
.cseg                    ;Начало сегмента кода.
.org$0                   ;По адресу 0
rjmp reset               ;переход на метку reset
.org$100                 ;По адресу 100
.db 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F, 0x77,
0x7C, 0x39, 0x5E, 0x79, 0x71
                         ;во Flash записывается таблица кодов символов
reset:                   ;По метке reset осуществляется:
ldi r16,low(RAMEND)      ;Инициализация стека. Вершина стека - в
ldi r17,high(RAMEND)     ;конце памяти данных
out spl,r16
out sph,r17
clr r16                  ;Инициализация портов ввода/вывода
out PORTC,r16            ;Порт C - на вывод
out PORTD,r16            ;Порт D - на вывод
ser r16
out DDRC,r16
out DDRD,r16
ldi r16,0x0F
out PORTA,r16            ;младшие 4 бита порта A - на ввод
clr r16
out DDRA,r16
ldi r16,0x01             ;Установка младшего бита порта D с целью
out PORTD,r16            ; подачи напряжения на общие аноды индикатора

main:                    ;По метке main
in r16,PINA              ;Происходит считывание данных с порта A
andi r16,0x0F            ;и выделение бит PA3…PA0.
mov r30,r16              ;формирование адреса flash исходя из
ldi r31,0x02             ;считанных данных
lpm r16,Z                ;извлечение в r16 данных из flash
out PORTC,r16            ;и вывод их на порт C (катоды индикатора).
rjmp main                ;переход на main и зацикливание программы
//-----------------------------------------------------------------------
```

Рассмотрим программу более подробно.

1. Сначала производится подключение библиотеки используемого контроллера Atmega 8535. После этого объявляется адрес начала сегмента кода и адрес Flash-памяти, по которому записывается таблица кодов символов:

```
.include"m8535def.inc"
.cseg
.org$0
rjmp reset
.org$100
.db 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F, 0x77,
0x7C, 0x39, 0x5E, 0x79, 0x71
```

Таблица кодов символов включает 16 кодовых комбинаций, каждая из которых соотвествует выводимому на индикатор символу в шестнадцатеричном коде. Таблица символов от 0 до F приведена в табл. 1.

Табл. 1. Соответствие символа и его шестнадцатеричного и двоичного кодов

Символ	Двоичный код	Шестнадцатеричный код
0	0b00111111	0x3F
1	0b00000110	0x06
2	0b01011011	0x5B
3	0b01001111	0x4F
4	0b01100110	0x66
5	0b01101101	0x6D
6	0b01111101	0x7D
7	0b00000111	0x07
8	0b01111111	0x7F
9	0b01101111	0x6F
A	0x01110111	0x77
b	0b01111100	0x7C
C	0b00111001	0x39
d	0b01011110	0x5E
E	0b01111001	0x79
F	0b01110001	0x71

2. По метке reset сначала происходит инициализация стека, затем – инициализация периферийных устройств микроконтроллера. В данном случае из периферийных устройств используются только порты ввода/вывода.

```
reset:
ldi r16,low(RAMEND)
ldi r17,high(RAMEND)
out spl,r16
out sph,r17
clr r16
out PORTC,r16
out PORTD,r16
ser r16
out DDRC,r16
out DDRD,r16
ldi r16,0x0F
out PORTA,r16
clr r16
```

```
out DDRA,r16
ldi r16,0x01
out PORTD,r16
```

В регистр указателя стека записывается адрес вершины стека, который соответствует концу памяти данных (`ldi r16,low(RAMEND); ldi r17,high(RAMEND); out spl,r16; out sph,r17`). Это необходимо для правильной работы программы при использовании подпрограмм и переходов. Порт С в программе будет подключен к катодам индикатора, поэтому он инициализируется на вывод, младший бит порта D по заданию управляет подачей напряжения питания на общие аноды индикатора, поэтому порт инициализируется на вывод, биты PA3...PA0 порта ввода/вывода A инициализируются на ввод информации.

3. В основном цикле программы сначала происходит опрос порта ввода/вывода A:

```
main:
in r16,PINA
andi r16,0x0F
```

Производится опрос всего порта (`in r16,PINA`), после чего путем побитового умножения результата на число 0xFF (`andi r16,0x0F`) происходит выделение младших значащих бит порта A.

4. По результату, считанному с порта A, производится формирование адреса во Flash-памяти, по которому хранится соответствующий символ. Поскольку во Flash память данные хранятся словами (2 байта), то для доступа к отдельному байту указывается двойной адрес. Поскольку адрес начала массива данных 0x100 (адрес в словах), то адрес первого байта – 0x200 (в байтах).

Для чтения данных из Flash в Z-регистре (r31:r30) указывается адрес памяти, а затем командой lpm происходит считывание данных:

```
mov r30,r16
ldi r31,0x02
lpm r16,Z
```

5. После считывание данных из Flash они выводятся на порт C, соединенный с индикатором:

```
out PORTC,r16
rjmp main
```

Далее программа зацикливается (`rjmp main`) для постоянного опроса порта ввода/вывода A.

Динамическая индикация символов

Пример 1 показывает, как осуществляется подача символов на один индикатор. Однако часто требуется осуществлять индикацию больших чисел.

Рассмотрим варианты индикации числа 1234 на четырех семисегментных индикаторах. В примере будет использован индикатор с общим анодом (рис. 2, а).

В простейшем случае к общим анодам разрядов индикатора необходимо подключить напряжение питания, а на выводы A...H каждого разряда подать соответствующую кодовую комбинацию (рис. 4).

Модуль «Микроконтроллер»

Методические указания написаны для проведения лабораторных работ с использованием модуля «Микроконтроллер», входящего в состав лабораторного стенда.

Модуль «Микроконтроллер» предназначен для программирования и изучения функций микроконтроллера ATmega8535 семейства AVR, выпускаемого фирмой Atmel. Внешний вид модуля приведен на рис. 1.

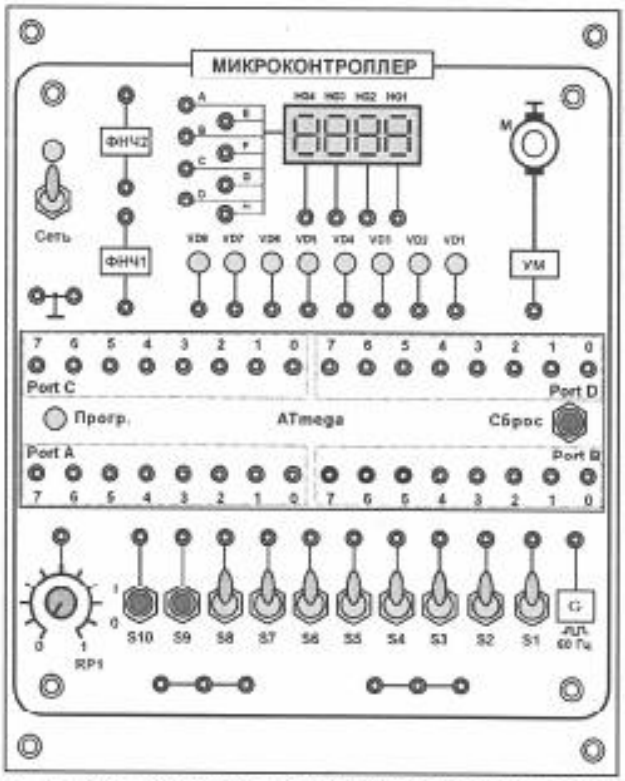


Рис. 1. Внешний вид модуля «Микроконтроллер»

На лицевой панели модуля расположены:

– переключатель «Сеть» со светодиодом индикации наличия напряжения. Переключатель осуществляет коммутацию напряжения, подаваемого на модуль;

– мнемосхему микроконтроллера с клеммами, связанными с портами ввода/вывода микроконтроллера;

– переключатели S1-S8 с выходными клеммами для подачи логических сигналов на микроконтроллер;

– кнопки S9, S10 с выходными клеммами для подачи логических сигналов на микроконтроллер;

– потенциометр RP1 с выходной клеммой для подачи аналогового напряжения на микроконтроллер;

– мнемосхема генератора низкочастотного прямоугольного сигнала 50 Гц и клемма выхода генератора;

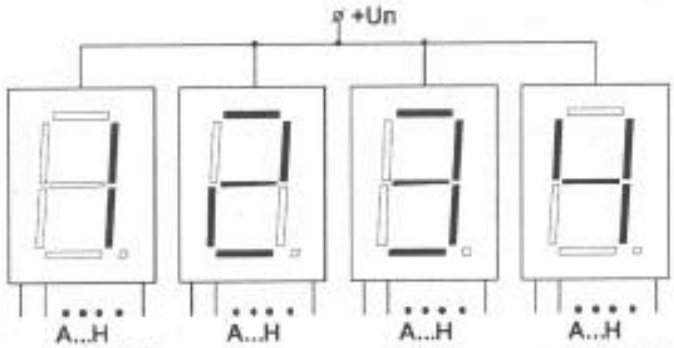


Рис. 4. Пример возможной реализации индикации символов на многоразрядном индикаторе

При подобной реализации многоразрядного индикатора возникает существенная проблема – при использовании четырех разрядов количество выводов микроконтроллера, используемых для индикации, составляет 32 шт, при этом всего рабочих выводов у микроконтроллера Atmega 8535 32, то есть ресурсы контроллера будут использованы полностью.

Для того, чтобы минимизировать ресурсоемкость процесса индикации, предлагается использовать метод динамической индикации. Этот метод основан на свойстве инерции человеческого зрения, при котором глаз не воспринимает разницу между быстро сменяющимися картинками, если они меняются с частотой, превышающей 25 Гц. Схема динамической индикации представлена на рис. 5.

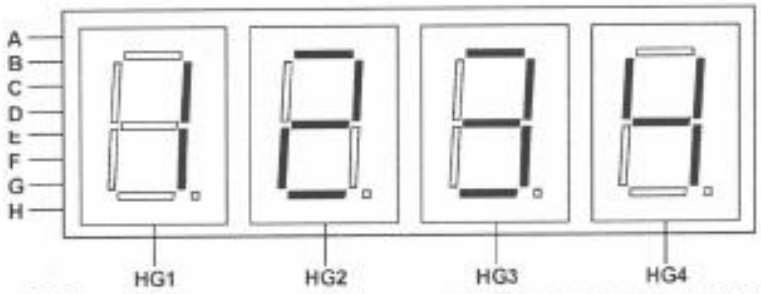


Рис. 5. Схема реализации динамической индикации символов

При динамической индикации соответствующие катоды всех индикаторов объединяются, то есть катоды сегмента А индикаторов HG1…HG4 объединяются в шину А, катоды сегмента В объединяются в шину В и т.д. Общие аноды индикаторов HG1…HG2, наоборот, разъединяются.

Если индикатор реализован подобным образом, то последовательность динамической индикации следующая:

– микроконтроллер выдает код числа 4 на катоды всех индикаторов, при этом напряжение питания подается только на разряд HG4. В результате цифра 4 светится только на индикаторе HG4;

– спустя время, не большее 1/100 секунды, на все сегменты выдается код числа 3, при этом напряжение питания подается только на разряд HG3. В результате цифра 3 светится только на индикаторе HG3;

– на следующем интервале времени на все индикаторы выдается код числа 2, при этом напряжение питания подается только на разряд HG2. В результате цифра 2 светится только на индикаторе HG2;

– в конце цикла на индикаторы подается код числа 1, а напряжение питания – на индикатор HG1. В результате цифра 1 светится только на разряд HG1. Далее процесс повторяется.

Поскольку каждый разряд индикатора обновляется с частотой как минимум 25 Гц, человеческий глаз не замечает мерцания индикатора и процесс отображения числа 1234 кажется постоянным, хотя в один момент времени всегда светится только один разряд индикатора. С увеличением частоты обновления цифр качество индикации улучшается.

Пример 2. Написать программу индикации на четырехразрядном семисегментном индикаторе числа 1234 с использованием динамической индикации. Код числа выдается на порт C, управление разрядами осуществляется с порта D.

```
//-------------------------------------------------------------------
; Программа демострации динамической индикации. На индикатор выводится
;число 1234.
;Выходы:
;     PORTC7…0 - управление сегменты (PC0 - A, PC1 - B, … , PC7 - H);
;     PORTD3 - разряд HG4;
;     PORTD2 - разряд HG3;
;     PORTD1 - разряд HG2;
;     PORTD0 - разряд HG1.

.include"m8535def.inc"    ;Подключение библиотеки Atmega8535
.cseg                     ;Начало сегмента кода.
.org$0                    ;По адресу 0
rjmp reset                ;осуществляется переход на метку reset
.org$100                  ;По адресу 100
.db 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F, 0x77,
0x7C, 0x39, 0x5E, 0x79, 0x71 ;находится массив данных кодов символов

reset:                    ;По метке reset
ldi r16,low(RAMEND)       ;происзводится конфигурация указателя стека
ldi r17,high(RAMEND)
out spl,r16
out sph,r17
clr r16                   ;Инициализируются порты ввода/вывода
out PORTC,r16
out PORTD,r16
ser r16
out DDRC,r16              ;PORTC - на вывод информации
out DDRD,r16              ;PORTD - на вывод информации
ldi r17,0x01              ;Включается младший разряд PORTD
out PORTD,r17             ;Для подачи напряжения на HG1

main:                     ;По метке main
cpi r17,0x01              ;производится опрос, какой разряд HG работает
breq HG1                  ;если первый, то переход на метку HG1
cpi r17,0x02              ;если второй,
breq HG2                  ;то переход на метку HG2
cpi r17,0x04              ;если третий,
```

```
breq HG3                ;то переход на метку HG3
cpi r17,0x08            ;если четвертый,
breq HG4                ;то переход на метку HG4
HG1:                    ;По метке HG1
ldi r31,0x02            ;в Z-регистр задается адрес числа 4 в Flash
ldi r30,0x04
rjmp met1               ;и переход на метку met1
HG2:                    ;По метке HG2
ldi r31,0x02            ;в Z-регистр задается адрес числа 3 в Flash
ldi r30,0x03
rjmp met1               ;и переход на метку met1
HG3:                    ;По метке HG3
ldi r31,0x02            ;в Z-регистр задается адрес числа 2 в Flash
ldi r30,0x02
rjmp met1               ;и переход на метку met1
HG4:                    ;По метке HG4
ldi r30,0x01            ;в Z-регистр задается адрес числа 1 в Flash
rjmp met1           ;и переход на метку met1
met1:               ;По метке met1
lpm r16,Z           ;из Flash по указанному адресу считываются данные
out PORTC,r16       ;и выводятся на PORTC
clr r16             ;Осуществляется программная задержка времени
met2:
inc r16
cpi r16,0xFF
brne met2
lsl r17             ;Значение r17 сдвигается на один разряд влево.
cpi r17,0x10        ;Если r17=0x10, то
brne met3
ldi r17,0x01        ;в r17 записывается 0x01
met3:
clr r16
out PORTC,r16       ;обнуляется PORTC
out PORTD,r17       ;обнуляется PORTD
rjmp main           ;осуществляется переход на main
//--------------------
```

Рассмотрим программу более подробно.

1. Сначала производится подключение библиотеки используемого контроллера Atmega 8535. После этого объявляется адрес начала сегмента кода и адрес Flash-памяти, по которому записывается таблица кодов символов:

```
.include"m8535def.inc"
.cseg
.org$0
rjmp reset
.org$100
.db 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F, 0x77,
0x7C, 0x39, 0x5E, 0x79, 0x71
```

2. По метке reset осуществляется конфигурирование периферийных устройств контроллера и инициализация указателя стека:

```
reset:
ldi r16,low(RAMEND)
ldi r17,high(RAMEND)
```

```
out spl,r16
out sph,r17
clr r16
out PORTC,r16
out PORTD,r16
ser r16
out DDRC,r16
out DDRD,r16
ldi r17,0x01
out PORTD,r17
```

После инициализации устройств в регистр r17 записывается число 0x01 (`ldi r17,0x01`), после чего содержимое регистра выводится на порт управления разрядами индикатора (`out PORTD,r17`). Далее в r17 будет находится значение, соотвесттвующее включенному состоянию PORTD.

3. По метке main сначала производится опрос включенного состояния PORTD путем опроса регистра r17:

```
main:
cpi r17,0x01
breq HG1
cpi r17,0x02
breq HG2
cpi r17,0x04
breq HG3
cpi r17,0x08
breq HG4
```

В зависимости от состояния r17 производится переход на метки HG1...HG4. Так, если в данный момент работает разряд HG1 (`cpi r17,0x01`), то осуществляется переход на метку HG1 (`breq HG1`).

4. Далее, в зависимости от метки, производится запись в Z-регистр значения адреса Flash-памяти, по которому находится код нужного символа:

```
HG1:
ldi r31,0x02
ldi r30,0x04
rjmp met1
HG2:
ldi r31,0x02
ldi r30,0x03
rjmp met1
HG3:
ldi r31,0x02
ldi r30,0x02
rjmp met1
HG4:
ldi r30,0x01
rjmp met1
met1:
lpm r16,Z
out PORTC,r16
```

После записи адреса символа осуществляется переход на метку met1, по которой происходит считывание данных из Flash-памяти в РОН r16 (`lpm r16,Z`) и вывод их на PORTC (`out PORTC,r16`).

5. В момент передачи данных на PORTC на одном из разрядов индикатора зажигается нужный символ, после чего необходимо сделать небольшую задержку времени:

```
clr r16
met2:
inc r16
cpi r16,0xFF
brne met2
```

Для реализации задержки сначала содержимое РОН r16 обнуляется (`clr r16`), после чего происходит его инкремент (`inc r16`) с последующим сравнением с уставкой (`cpi r16,0xFF`). При значении r16, меньшем уставки, происходит возврат на метку met2 (`brne met2`). При достижении уставки программа следует дальше.

6. По прошествии выдержки времени необходимо выключить индикатор и переключиться на индикацию другого разряда:

```
lsl r17
cpi r17,0x10
brne met3
ldi r17,0x01
met3:
clr r16
out PORTC,r16
out PORTD,r17
rjmp main
```

С этой целью содержимое r17 последовательно сдвигается на одни разряд влево (`lsl r17`), после чего производится выдача его содержимого на PORTD (`out PORTD,r17`). Однако при достижении r17 значения 0x10 (`cpi r17,0x10`) в него необходимо записать значение 0x01 (`ldi r17,0x01`) для того, чтобы разряды PD0...PD3 включались циклично, а не происходило включение разрядов PD4...PD7 порта D. Во-избежание ложного отображения информации перед сменой включенного разряда индикатора происходит выключение сегментов индикатора (`clr r16; out PORTC,r16`). После этого программа зацикливается (`rjmp main`).

Работа № 5. АЦП и динамическая индикация

Цель работы

Освоить совместную индикацию аналого-цифрового преобразователя микроконтроллера Atmega8535 и четырехразрядного семисегментного индикатора.

Программа работы

1. Изучить необходимый теоретический материал о принципах индикации символов на светодиодном семисегментном индикаторе и работе АЦП контроллера.
2. Разобраться в программах, представленных в лабораторной работе.
3. Написать и отладить собственную программу по заданию преподавателя.

Пояснения к работе

При работе цифровой техники часто требуется вывести какую-либо переменную на индикацию. Например, если требуется знать точное значение результата преобразования АЦП, проще всего вывести этот результат на индикатор.

Однако здесь возникают определенные сложности. Действительно, результат преобразвоания АЦП микроконтроллера Atmega8535 – 10-битное число, то есть его максимальное значение – $11\ 1111\ 1111_b = 3FF_h = 1023_{10}$.

Если на индикатор требуется вывести значение кода в десятичном виде, требуется осуществить преобразование двоичного кода в десятичный, для чего необходимо выделить число единиц, десятков, сотен и тысяч – по количеству разрядов используемого индикатора.

Процесс преобразования кода графически показан на рис. 1 и заключается в следующем.

Из исходного числа **N** вычитается число, соответствующее старшему разряду индикатора. Если индикатор – трехразрядный, то из исходного числа вычитается число 100.

Если результат вычитания **N-100** больше нуля, то количество сотен в конечном коде **ddd** увеличивается на 1 (рис. 1). В этом случае из результата снова вычитается число 100 и производится оценка результата. Если результат оказывается меньше нуля, то это значит, что количество сотен посчитано и можно переходить к подсчету десятков (**dd**).

При окончании подсчета сотен начальное число N должно уменьшиться. Так, если исходное число N=948, то после подсчета сотен ddd=9, а N=48. Из этого числа производится вычитание числа 10 по вышеописанному алгоритму. После этого производится подсчет единиц.

По окончании процесса преобразования кода имеется три разряда:

– ddd – количество сотен;

– dd – количество десятков;

– d – количество единиц.

Эти числа выводятся на динамическую индикацию с использованием алгоритма, описанного в лабораторной работе №10.

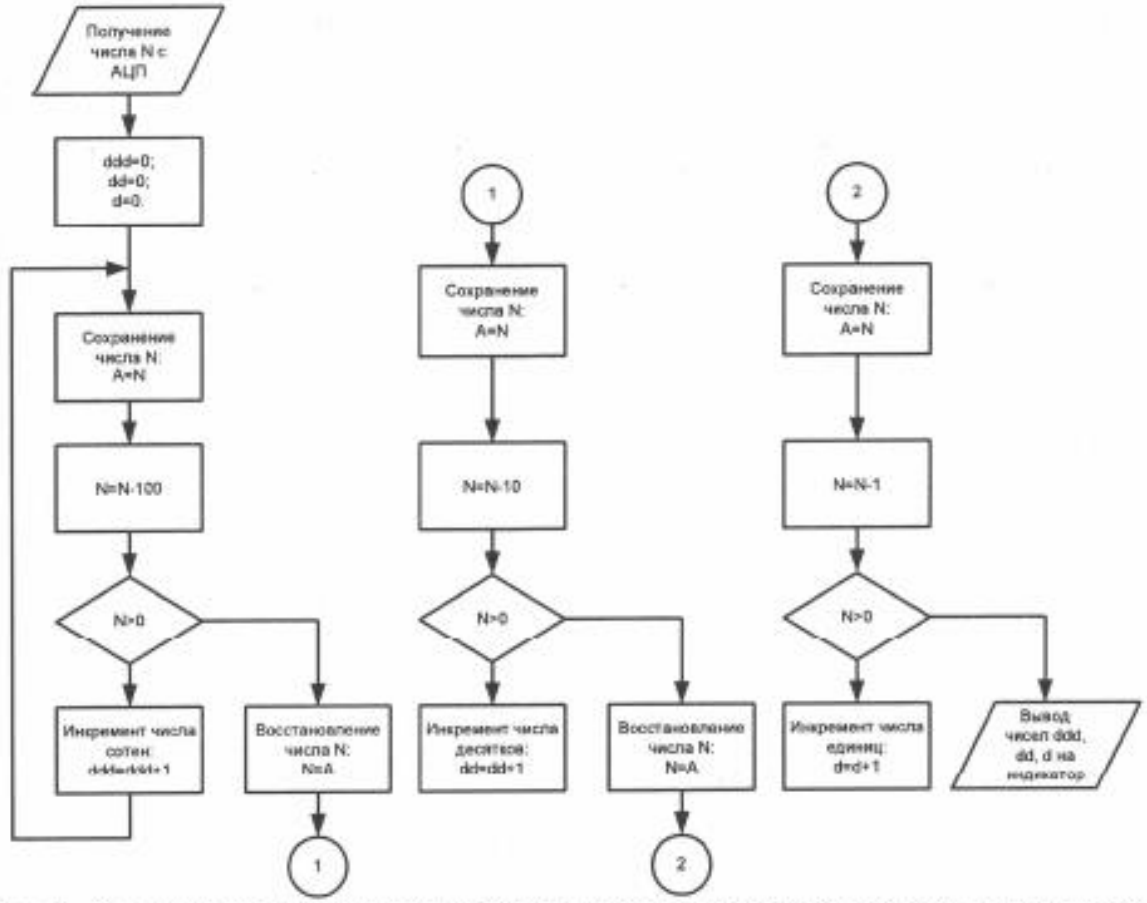


Рис. 1. Алгоритм преобразования кодов для вывода информации на семисегментный индикатор

В примере, рассмотренном в лабораторной работе, осуществляется вывод результата преобразования АЦП на четырехразрядный семисегментный индикатор. Поскольку результат преобразования – 10-битное число, в примере рассматривается работа и двухбайтными числами, которые также называются «словами» (word).

Пример 1. Вывести результат преобразования АЦП на семисегментный индикатор. Использовать канал АЦП ADC0, результат преобразования – 10 битное число без знака, для управления индикатором использовать порт ввода/вывода C, для управления разрядами индикатора использовать порт ввода-вывода D.

```
//-----------------------------------------------------------------
;Программа динамической индикации результата преобразования АЦП
;Входы:
;     PD0 - аналоговый сигнал задания на АЦП
;Выходы:
;     PC0…PC7 - управление сегментами индикатора
;     PD0…PD3 - управление разрядами индикатора

.include"m8535def.inc"    ;Подключение стандартной библиотеки Atmega8535
.cseg                     ;Начало сегмента кода
```

```
.org$0                      ;По адресу 0
rjmp reset                  ;переход на метку reset
.org$E                      ;По адресу 0E
rjmp ADC_ready              ;переход на метку ADC ready
.org$100                    ;По адресу 100
.db 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F, 0x77,
0x7C, 0x39, 0x5E, 0x79, 0x71 ;располагается таблица кодов символов

reset:                      ;По метке reset
cli                         ;происходит глобальный запрет прерываний
ldi r16,low(RAMEND)         ;Инициализируется указатель стека
ldi r17,high(RAMEND)
out spl,r16
out sph,r17
clr r16                     ;Инициализируются порты ввода/вывода
out PORTA,r16
out PORTC,r16
out PORTD,r16
out DDRA,r16                ;Порт A - на ввод
ser r16
out DDRC,r16                ;Порт C, порт D - на вывод
out DDRD,r16
clr r16
out ADMUX,r16               ;Инициализация мультиплексора АЦП
ldi r16,0xCF
out ADCSRA,r16              ;Инициализация АЦП
ldi r17,0x01
out PORTD,r17               ;Включение разряда HG0 индикатора
sei                         ;Глобальное разрешение прерываний

main:                       ;По метке main опрос разрядов индикатора
cpi r17,0x01                ;Если работает HG1,
breq HG1                    ;то переход на метку HG1
cpi r17,0x02                ;Если работает HG2,
breq HG2                    ;то переход на метку HG2
cpi r17,0x04                ;Если работает HG3,
breq HG3                    ;то переход на метку HG3
cpi r17,0x08                ;Если работает HG4,
breq HG4                    ;то переход на метку HG4
HG1:                        ;По метке HG1
ldi r31,0x02                ;В Z-регистр записывается адрес числа,
mov r30,r29                 ;соответствующего единицам.
rjmp met1                   ;Переход на метку met1
HG2:                        ;По метке HG2
ldi r31,0x02                ;В Z-регистр записывается адрес числа,
mov r30,r28                 ;соответствующего десяткам.
rjmp met1                   ;Переход на метку met1
HG3:                        ;По метке HG3
ldi r31,0x02                ;В Z-регистр записывается адрес числа,
mov r30,r27                 ;соответствующего сотням.
rjmp met1                   ;Переход на метку met1
HG4:                        ;По метке HG4
ldi r31,0x02                ;В Z-регистр записывается адрес числа,
mov r30,r26                 ;соответствующего тысячам.
```

```
rjmp met1                ;Переход на метку met1
met1:                    ;По метке met1
lpm r16,Z                ;Из flash извлекается код числа
out PORTC,r16            ;И выводится на индикацию
clr r16                  ;Производится программная задержка
met2:
inc r16
cpi r16,0x7F
brne met2
lsl r17                  ;После задержки - смена разряда индикатора
cpi r17,0x10
brne met3
ldi r17,0x01
met3:
clr r16
out PORTC,r16
out PORTD,r17
rjmp main                ;Возврат на main

ADC_ready:               ;Прерывание готовности АЦП
cli                      ;глобальный запрет прерываний
in r25,SREG              ;Сохранение регистра SREG
PUSH r16                 ;Сохранение значений регистра r16
PUSH r17                 ;и регистра r17 в стеке
clr r26                  ;Очистка содержимого r26…r29
clr r27
clr r28
clr r29
in r16,ADCL              ;Считывание результата преобразования АЦП
in r17,ADCH              ;в r17:r16

dddd_1:                  ;По метке dddd_1
mov r14,r16              ;результат работы АЦП сохраняется в r15:r14
mov r15,r17
ldi r18,0xE8             ;В r19:r18 записывается код числа 1000 (0x3E8)
ldi r19,0x03
sub r16,r18              ;и производится вычитание r17:r16-r19:r18
sbc r17,r19
brcc dddd_2              ;Если при этом флаг C=0, то переход на dddd_2
mov r16,r14              ;иначе восстановление r17:r16 из r15:r14
mov r17,r15
rjmp ddd_1               ;и переход к подсчету числа сотен.
dddd_2:                  ;По метке dddd_2
inc r26                  ;производится инкремент числа тысяч
rjmp dddd_1              ;и возврат на метку dddd_1.

ddd_1:                   ;По метке ddd_1
mov r14,r16              ;остаток исходного числа сохраняется в r15:r14
mov r15,r17
ldi r18,0x64             ;В r19:r18 записывается код числа 100 (0x64)
ldi r19,0x00
sub r16,r18              ;и производится вычитание r17:r16-r19:r18
sbc r17,r19
brcc ddd_2               ;Если при этом флаг C=0, то переход на ddd_2
```

```
mov r16,r14             ;иначе восстановление r17:r16 из r15:r14
mov r17,r15
rjmp dd_1               ;и переход к подсчету числа десятков.
ddd_2:                  ;По метке ddd_2
inc r27                 ;производится инкремент числа сотен
rjmp ddd_1              ;и возврат на метку ddd_1.

dd_1:                   ;По метке dd_1
mov r14,r16             ;остаток исходного числа сохраняется в r15:r14
mov r15,r17
ldi r18,0x0A            ;В r19:r18 записывается код числа 10 (0x0A)
ldi r19,0x00
sub r16,r18             ;и производится вычитание r17:r16-r19:r18
sbc r17,r19
brcc dd_2               ;Если при этом флаг C=0, то переход на dd_2
mov r16,r14             ;иначе восстановление r17:r16 из r15:r14
mov r17,r15
rjmp d_1                ;и переход к подсчету числа единиц.
dd_2:                   ;По метке dd_2
inc r28                 ;производится инкремент числа десятков
rjmp dd_1               ;и возврат на метку dd_1.

d_1:                    ;По метке d_1
mov r14,r16             ;остаток исходного числа сохраняется в r15:r14
mov r15,r17
ldi r18,0x01            ;В r19:r18 записывается код числа 1 (0x01)
ldi r19,0x00
sub r16,r18             ;и производится вычитание r17:r16-r19:r18
sbc r17,r19
brcc d_2                ;Если при этом флаг C=0, то переход на d_2
mov r16,r14             ;иначе восстановление r17:r16 из r15:r14
mov r17,r15
rjmp ADC_1              ;и переход на метку ADC_1
d_2:                    ;По метке d_2
inc r29                 ;производится инкремент числа единиц
rjmp d_1                ;и возврат на метку d_1.

ADC_1:                  ;По метке ADC_1
ldi r16,0xCF            ;Производится перезапуск АЦП
out ADCSRA,r16
POP r17                 ;Восстанавливаются значения r16,r17
POP r16                 ;в обратном порядке.
out SREG,r25            ;Восстанавливается содержимое регистра SREG
sei                     ;глобальное разрешение прерываний
reti                    ;и выход из п/п обработки прерывания.
//---------------------------------------------------------------------
```

Рассмотрим программу более подробно.

1. Сначала производится подключение библиотеки используемого контроллера Atmega 8535. После этого объявляется адрес начала сегмента кода, определяются вектора прерываний и адрес Flash-памяти, по которому записывается таблица кодов символов:

```
.include"m8535def.inc"
.cseg
```

– светодиоды VD1 – VD8 с клеммами для их подключения к источнику напряжения (например, к микроконтроллеру);

– электродвигатель постоянного тока М с усилителем мощности и клеммой для подачи на него управляющего напряжения;

– семисегментный четырехсимвольный светодиодный индикатор с клеммами подачи напряжения на сегменты A, B, C, D, E, F, G, H, а также на общую точку каждого сегмента индикатора;

– два фильтра низкой частоты для фильтрации ШИМ-сигналов на выходе микроконтроллера.

С тыльной стороны модуля располагается разъем для подачи напряжения ~220В 50 Гц на модуль, а также разъем для подключения модуля к персональному компьютеру по интерфейсу USB.

Табл. 1. Краткая характеристика микроконтроллера ATmega8535

Параметр	Значение
Частота установленного кварцевого резонатора	8 МГц
Напряжение электропитания	2,7 – 5,5 В
Объем внутренней Flash – памяти	8 кБайт
Объем энергонезависимой памяти	512 Байт
Объем внутренней ОЗУ	512 Байт
32 программируемых входа/выхода	32 на 4 портах
JTAG – интерфейс	нет
8-битные таймеры/счетчики с ШИМ	2 шт.
16-битный таймер/счетчик с ШИМ	1 шт.
10-разрядный аналогово-цифровой преобразователь	есть
Количество каналов АЦП	8
Аналоговый компаратор	есть
Источники внешних прерываний	3 шт.
Универсальный приемопередатчик USART	есть
SPI – интерфейс	есть
TWI – интерфейс	есть

```
.org$0
rjmp reset
.org$E
rjmp ADC_ready
.org$100
.db 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F, 0x77,
0x7C, 0x39, 0x5E, 0x79, 0x71
```

2. По метке reset осуществляется конфигурирование периферийных устройств контроллера (портов ввода/вывода и АЦП) и инициализация указателя стека:

```
reset:
cli
ldi r16,low(RAMEND)
ldi r17,high(RAMEND)
out spl,r16
out sph,r17
clr r16
out PORTA,r16
out PORTC,r16
out PORTD,r16
out DDRA,r16
ser r16
out DDRC,r16
out DDRD,r16
clr r16
out ADMUX,r16
ldi r16,0xCF
out ADCSRA,r16
ldi r17,0x01
out PORTD,r17
sei
```

Порт ввода/вывода A инициализируется на ввод информации для работы с АЦП, порты C и D инициализируются на вывод для работы с индикатором. Мультиплексов АЦП инициализируется на работу с каналом ADC0 (`clr r16; out ADMUX,r16`), АЦП запускается в одиночном режиме преобразования с предделителем 128 (`ldi r16,0xCF; out ADCSRA,r16`).

3. По метке main сначала производится опрос включенного состояния PORTD путем опроса регистра r17:

```
main:
cpi r17,0x01
breq HG1
cpi r17,0x02
breq HG2
cpi r17,0x04
breq HG3
cpi r17,0x08
breq HG4
```

В зависимости от состояния r17 производится переход на метки HG1...HG4. Так, если в данный момент работает разряд HG1 (`cpi r17,0x01`), то осуществляется переход на метку HG1 (`breq HG1`).

4. Далее, в зависимости от метки, производится запись в Z-регистр значения адреса Flash-памяти, по которому находится код нужного символа, соотвествующего числу тысяч, сотен, десятков или единиц:

```
HG1:
ldi r31,0x02
mov r30,r29
rjmp met1
HG2:
ldi r31,0x02
mov r30,r28
rjmp met1
HG3:
ldi r31,0x02
mov r30,r27
rjmp met1
HG4:
ldi r31,0x02
mov r30,r26
rjmp met1
met1:
lpm r16,Z
out PORTC,r16
```

Принимается, что число тысяч находится в РОН r26, число сотен – в РОН r27, число десятков – в РОН r28, число единиц – в РОН r29. После записи адреса символа осуществляется переход на метку met1, по которой происходит считывание данных из Flash-памяти в РОН r16 (`lpm r16,Z`) и вывод их на PORTC (`out PORTC,r16`).

5. В момент передачи данных на PORTC на одном из разрядов индикатора зажигается нужный символ, после чего необходимо сделать небольшую задержку времени:

```
clr r16
met2:
inc r16
cpi r16,0x7F
brne met2
```

Для реализации задержки сначала содержимое РОН r16 обнуляется (`clr r16`), после чего происходит его инкремент (`inc r16`) с последующим сравнением с уставкой (`cpi r16,0x7F`). При значении r16, меньшем уставки, происходит возврат на метку met2 (`brne met2`). При достижении уставки программа следует дальше.

6. По прошествии выдержки времени необходимо выключить индикатор и переключиться на индикацию другого разряда:

```
lsl r17
cpi r17,0x10
brne met3
ldi r17,0x01
met3:
clr r16
out PORTC,r16
out PORTD,r17
rjmp main
```

С этой целью содержимое r17 последовательно сдвигается на один разряд влево (`lsl r17`), после чего производится выдача его содержимого на PORTD

(`out PORTD,r17`). Однако при достижении r17 значения 0x10 (`cpi r17,0x10`) в него необходимо записать значение 0x01 (`ldi r17,0x01`) для того, чтобы разряды PD0…PD3 включались циклично, а не происходило включение разрядов PD4…PD7 порта D. Во-избежание ложного отображения информации перед сменой включенного разряда индикатора происходит выключение сегментов индикатора (`clr r16; out PORTC,r16`). После этого основной цикл программы замыкается (`rjmp main`).

7. При окончании преобразования АЦП происходит приостановка основного цикла программы, прыжок на вектор прерывания АЦП по адресу 0x0E и переход на метку ADC_ready. При переходе на эту метку сначала производится запрет всех прерываний и сохранение регистра SREG в РОН r25. Также значения РОН r16, r17 сохраняются в стеке для освобождения этих регистров. При выходе из подпрограммы обработки прерывания эти регистры должны быть восстановлены:

```
ADC_ready:
cli
in r25,SREG
PUSH r16
PUSH r17
clr r26
clr r27
clr r28
clr r29
in r16,ADCL
in r17,ADCH
```

Дополнительно перед началом подсчета количества тысяч, сотен, десятков и единиц результата преобразования АЦП производится сброс счетчиков этих величин (`clr r26, clr r27, clr r28, clr r29`), затем – считывание результата преобразования АЦП в регистр r17(h):r16(l) (`in r16,ADCL; in r17,ADCH`).

8. Далее производится подсчет количества тысяч по алгоритму, описанному выше (рис. 1):

```
dddd_1:
mov r14,r16
mov r15,r17
ldi r18,0xE8
ldi r19,0x03
sub r16,r18
sbc r17,r19
brcc dddd_2
mov r16,r14
mov r17,r15
rjmp ddd_1
dddd_2:
inc r26
rjmp dddd_1
```

Сначала результат преобразования АЦП сохраняется в регистре r15:r14 (`mov r14,r16; mov r15,r17`). После этого в двойной регистр r19:r18 записывается число 1000 (0x03E8) и производится вычитание r17:r16-r19:r18 (`sub r16,r18; sbc r17,r19`). Если в результате вычитания не произошло переноса, то есть результат больше нуля, то идет переход на метку dddd_2 (`brcc dddd_2`), по которой

производится инкремент количества тысяч и возврат на метку dddd_1. Если при вычитании устанавливается флаг переноса, то содержимое регистра r17:r16 восстанавливается из r15:r14 (`mov r16,r14; mov r17,r15`) и идет переход на подсчет количества сотен, далее – десятков и единиц (аналогично).

9. При окончании процесса подсчета количества разрядов результата преобразования АЦП производится переход на метку ADC_1:

```
ADC_1:
ldi r16,0xCF
out ADCSRA,r16
POP r17
POP r16
out SREG,r25
sei
reti
```

По этой метке производится перезапуск АЦП (`ldi r16,0xCF; out ADCSRA,r16`), восстановление r16, r17 из стека в обратном порядке (`POP r17, POP r16`), восстановление регистра SREG (`out SREG,r25`), прерывания разрешаются (`sei`). После этого осуществляется выход из подпрограммы обработки прерывания и возврат на прерванное место основного цикла (`reti`).

Работа № 6. Внешние прерывания

Цель работы

Освоить работу с системой внешних прерываний микроконтроллера Atmega8535.

Программа работы

1. Изучить необходимый теоретический материал о принципах работы внешних прерываний микроконтроллера Atmega 8535.
2. Разобраться в программе, представленной в лабораторной работе.
3. Написать и отладить собственную программу по заданию преподавателя.

Пояснения к работе

При работе высокоскоростных цифровых устройств, синхронизации автономных систем с электрической сетью, подсчете длительности импульсов микроконтроллеру требуется контролировать с высокой точностью момент появления того или иного сигнала.

Производить контроль состояния цифровых входов можно, например, простой директивой `in r16, PINA`, производя это в основном цикле программы:

```
;-----------------------------------------------------------------
main:
….
….
in r16,PINA
….
….
rjmp main
;-----------------------------------------------------------------
```

Однако при использовании этого способа опроса состояния портов точность фиксации момента перехода вывода порта из одного состояния в другое будет определяться величиной программы.

Для того, чтобы в момент изменения состояния цифровых входов происходило прерывание основного цикла программы и осуществлялся переход на обработку прерывания, в микроконтроллере Atmega 8535 предусмотрена система внешних прерываний.

Внешние прерывания в микроконтроллере обозначены как INT0, INT1, INT2. Соответственно с каждым прерыванием связан определенный вывод микросхемы:

– с прерыванием INT0 связан вывод PORTD2;

– с прерыванием INT1 связан вывод PORTD3;

– с прерыванием INT2 связан вывод PORTB2.

Каждому прерыванию соответствует свой адрес в таблице векторов контроллера:

– адрес вектора прерывания INT0 0x01;

– адрес вектора прерывания INT1 – 0x02;

– адрес вектора прерывания INT2 – 0x12.

В статическом режиме, когда состояние сигнала на выводе микросхемы не изменяется, выполняется основной цикл программы. Однако когда состояние сигнала меняется, возникает запрос на внешнее прерывание, которое вызывает переход из основного цикла программы на адрес вектора прерывания, по которому происходит переход на подпрограмму обработки этого прерывания. После окончания обработки прерывания происходит возврат на прерванное место основного цикла программы.

Для инициализации внешних прерываний предназначены специальные регистры: главный регистр управления внешних прерываний **GICR** (General Interrupt Control Register), регистр управления прерываниями INT0, INT1 **MCUCR**, регистр управления прерыванием INT2 **MCUCSR** и регистр флагов прерываний **GIFR**.

Регистр **GICR** представляет собой 8-разрядный регистр управления, состав которого приведен в табл. 1.

Табл. 1. Регистр управления GICR

Бит	7	6	5	4	3	2	1	0
Название	INT1	INT0	INT2	–	–	–	IVSEL	IVCE

Биты 7…5: INT1, INT0, INT2. Эти биты отвечают за разрешение или запрещение работы внешних прерываний. Если соответствующий бит установлен в состояние логической «1», то прерывание, управляемое этим битом, активно. В этом случае режим работы внешних прерываний определятся установками в регистрах **MCUCR** и **MCUCSR**.

Биты 4…2. Эти биты в регистре **GICR** зарезервированы и не задействуются при программировании.

Биты 1,0: IVSEL, IVCE. Эти биты в микроконтроллере не предназначены для изменения пользователем и при программировании не изменяются.

Регистр управления прерываниями INT0, INT1 **MCUCR** предназначен для определения логики срабатывания прерываний (табл. 2).

Табл. 2. Регистр управления **MCUCR**

Бит	7	6	5	4	3	2	1	0
Название	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00

Биты 7…4: SM2, SE, SM1, SM0. Эти биты в микроконтроллере не предназначены для изменения пользователем и при программировании не изменяются.

Биты 3, 2: ISC11, ISC10. Эти биты определяют событие, при котором происходит срабатывание прерывания INT1 (табл. 3).

Табл. 3. Биты управления прерыванием INT1

ISC11	**ISC10**	**Описание**
0	0	Прерывание по низкому уровню сигнала на PORTD3
0	1	Прерывание при любом изменении PORTD3

1	0	Прерывание по спадающему фронту на PORTD3
1	1	Прерывание по нарастающему фронту на PORTD3

Биты 1, 0: ISC01, ISC00. Эти биты определяют событие, при котором происходит срабатывание прерывания INT0 (табл. 4).

Табл. 4. Биты управления прерыванием INT0

ISC01	ISC00	Описание
0	0	Прерывание по низкому уровню сигнала на PORTD3
0	1	Прерывание при любом изменении PORTD3
1	0	Прерывание по спадающему фронту на PORTD3
1	1	Прерывание по нарастающему фронту на PORTD3

Регистр управления прерыванием INT2 **MCUCSR** предназначен для определения логики срабатывания прерывания INT2 (табл. 5).

Табл. 5. Регистр управления **MCUCSR**

Бит	7	6	5	4	3	2	1	0
Название	–	ISC2	–	–	WDRF	BORF	EXTRF	PORF

Биты 7, 5, 4. Эти биты зарезервированы и для программирования не предназначены.

Биты 3, 2, 1, 0: WDRF, BORF, EXTRF, PORF. Эти биты в микроконтроллере не предназначены для изменения пользователем и при программировании не изменяются.

Бит 6: ISC2. Этот бит определяет логику срабатывания прерывания INT2. Если бит установлен в состояние логической «1», то прерывание срабатывает по спадающему фронту сигнала на входе, иначе – по нарастающему. При изменении состояния бита ISC2 возможно ложное срабатывание прерывания. Поэтому рекомендуется сначала остановить прерывание INT2 обнулением бита INT2 в регистре **GICR**, после этого изменить состояние бита ISC2, после чего очистить флаг прерывания INT2 в регистре **GIFR** и активировать прерывание установкой бита INT2 в регистре **GICR**.

Регистр флагов прерываний **GIFR** содержит флаги прерываний при их срабатывании. Состав регистра приведен в табл. 6.

Табл. 6. Регистр управления **GIFR**

Бит	7	6	5	4	3	2	1	0
Название	INTF1	INTF0	INTF2	–	–	–	–	–

Биты 7…5: INTF1, INTF0, INTF2. Эти биты содержат флаги соответствующих внешних прерываний. Если прерывание срабатывает и оно разрешено в регистре GICR, выставляется его флаг. При выходе из подпрограммы обработки прерывания по команде reti осуществляется аппаратный сброс флага прерывания. Если есть необходимость ручного сброса прерывания, необходимо в соответствующий бит регистра GIFR записать значение логической «1».

Биты 4...0. Эти биты зарезервированы и для программирования не предназначены.

Для изучения принципа работы с внешними прерываниями далее в лабораторной работе рассмотрен пример, в котором с помощью внешних прерываний реализуется задержка времени.

Пример. Написать программу, в которой осуществляется инкремент 8-разрядного регистра 1 раз в 500 мс. При достижении кодом числа 0x0F начинается декремент кода до нуля, и далее процесс повторяется. В качестве временной задержки использовать источник сигнала 50 Гц, синхронизированный с электрической сетью и подключенный к внешнему прерыванию INT0.

```
;-----------------------------------------------------------------
;Использование внеших прерываний
;Входы:
;     PD2 - подача прямоугольных импульсов с генератора 50Гц
;Выходы:
;     PC0…PC7 - индикация кода на светодиодах

.include"m8535def.inc"    ;подключение библиотеки Atmega8535
.cseg                     ;начало сегмента кода
.org$0                    ;по адресу 0
rjmp reset                ;происходит переход на метку reset
.org$1                    ;по адресу 1
rjmp INT0_ready           ;происходит переход на метку INT0_ready

reset:                    ;По метке reset
cli                       ;происходит запрет всех прерываний
ldi r16,low(RAMEND)       ;Производится инициализация указателя стека
ldi r17,high(RAMEND)
out spl,r16
out sph,r17
ldi r16,0xFB              ;Инициализация портов ввода/вывода
out PORTD,r16
clr r16
out DDRD,r16
out PORTC,r16
ser r16
out DDRC,r16
ldi r16,0x03              ;Инициализация регистра MCUCR
out MCUCR,r16
ldi r16,0x40              ;инициализация регистра GICR
out GICR,r16
clr r16                   ;Очистка необходимых РОН
clr r17
clr r18
sei                       ;Разрешение всех прерываний

main:                     ;По метке main реализован бесконечный цикл
rjmp main

INT0_ready:               ;При появлении внешнего прерывания INT0
```

```
Cli                     ;Осуществляется запрет всех прерываний
inc r16                 ;Происходит инкремент r16
cpi r16,0x19            ;и его сравнение с уставкой 0x19 (25)
breq m1                 ;Если r16=0x19, то переход на метку m1
rjmp m3                 ;иначе переход на метку m3
m1:                     ;По метке m1
clr r16                 ;РОН r16 очищается
cpi r18,0x01            ;и производится проверка r18
breq rev                ;Если r18=0x01, то переход на метку rev
inc r17                 ;иначе происходит инкремент r17
cpi r17,0x0F            ;и его сравнение с уставкой 0x0F
brne m2                 ;Если r17≠0x0F, то переход на метку m2
ldi r18,0x01            ;Иначе установка r18=0x01
rjmp m2                 ;и переход на m2
rev:                    ;По метке rev
dec r17                 ;РОН r17 декрементируется
cpi r17,0x00            ;и его значение сравнивается с уставкой
brne m2                 ;Если r17≠0, то переход на метку m2
ldi r18,0x00            ;иначе обнуление r18
m2:                     ;По метке m2
out PORTC,r17           ;осуществляется вывод на PORTC содержимого r17
m3:                     ;По метке m3
sei                     ;осуществляется разрешение всех прерываний
reti                    ;и выход из п/п обработки прерывания
;-----------------------------------------------------------------
```

Рассмотрим программу более подробно.

1. Сначала производится подключение библиотеки используемого контроллера и указываются метки, на которые необходимо переходить программе при появлении прерываний или при сбросе. Для этого указываются используемые вектора прерываний reset (адрес 0) и INT0 (адрес 1) согласно технической документации на контроллер:

```
.include"m8535def.inc"
.cseg
.org$0
rjmp reset
.org$1
rjmp INT0_ready
```

2. По метке reset производится инициализация стека и периферийных устройств микроконтроллера:

```
reset:
cli
ldi r16,low(RAMEND)
ldi r17,high(RAMEND)
out spl,r16
out sph,r17
ldi r16,0xFB
out PORTD,r16
clr r16
out DDRD,r16
out PORTC,r16
ser r16
out DDRC,r16
```

Вершина стека помещается в конце памяти данных (`ldi r16,low(RAMEND);ldi r17,high(RAMEND)`), инициализируются порты ввода/вывода: PORTC инициализируется на вывод информации, а PORTD – на ввод, за исключением бита PD2, используемого внешним прерыванием INT0.

3. Далее производится инициализация внешних прерываний:

```
ldi r16,0x03
out MCUCR,r16
ldi r16,0x40
out GICR,r16
clr r16
clr r17
clr r18
sei
```

Активируется прерывание INT0 с помощью установки бита INT0 в регистре GICR (`ldi r16,0x40; out GICR,r16`), у прерывания INT0 выбирается режим срабатывания триггера по нарастающему фронту сигнала (`ldi r16,0x03; out MCUCR,r16`). Дополнительно очищаются необходимые в работе регистры общего назначения.

4. Основной цикл программы выполняется пустым, так как задержка времени реализуется не программно, а с помощью подсчета прерываний (периодов сетевого напряжения):

```
main:
rjmp main
```

5. При появлении нарастающего фронта импульса на выводе PD2 микроконтроллера срабатывает внешнее прерывание, в результате чего происходит переход на адрес памяти 0x01 (вектор прерывания INT0), по которому осуществляется переход на метку INT0_ready. По этой метке начинается подпрограмма обработки прерывания:

```
INT0_ready:
cli
inc r16
cpi r16,0x19
breq m1
rjmp m3
```

После входа в подпрограмму сначала запрещается срабатывание других прерываний (`cli`). В регистре r16 производится подсчет количества сработавших прерываний (`inc r16`). Когда это число достигает 25, что соответствует временной задержке 50 мс (`cpi r16,0x19`), осуществляется переход на метку m1 (`breq m1`). В противном случае осуществляется переход по метке m3 (`rjmp m3`).

6. По метке m1 счетчик прерываний очищается, после чего программа определяет, какой процесс в данный момент идет – инкремент или декремент счетчика кода, выдаваемого на индикацию:

```
m1:
clr r16
cpi r18,0x01
breq rev
inc r17
cpi r17,0x0F
brne m2
ldi r18,0x01
```

ЛАБОРАТОРНЫЕ РАБОТЫ

Работа № 1. Таймеры T0/T2. Режим широтно-импульсной модуляции

Цель работы

Освоить теоретический и практический материал по работе 8-разрядных таймеров микроконтроллера Atmega 8535 в режиме широтно-импульсной. Применить приобретенные навыки при написании программы.

Программа работы

1. Изучить необходимый теоретический материал о режиме «Быстрый ШИМ» таймеров T0 и T2.
2. Изучить необходимый теоретический материал о режиме «Фазово-корректный ШИМ» таймеров T0 и T2.
3. Разобраться в программах по использованию ШИМ, представленных в лабораторной работе.
4. Написать и отладить собственную программу с использованием таймеров в режиме ШИМ в соответствии с вариантом.

Пояснения к работе

Таймеры/счетчики большинства современных микропроцессоров и микроконтроллеров имеют возможность работать как в режиме подсчета временных интервалов, так и в режиме широтно-импульсной модуляции (ШИМ).

Широтно-импульсной модуляцией называется формирование среднего значения сигнала с помощью сигналов логического «0» и логической «1» за период модуляции T (рис. 1).

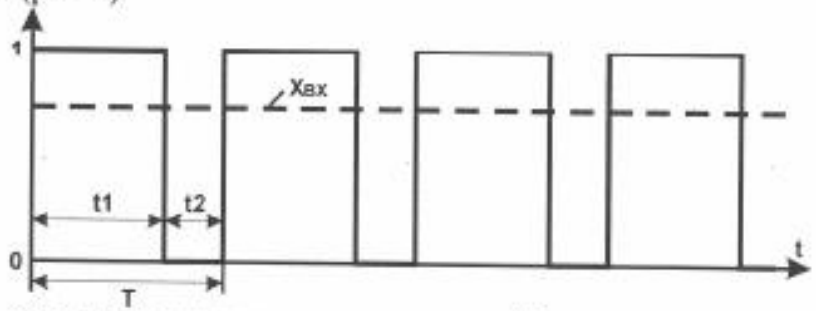


Рис. 1. Принцип широтно-импульсной модуляции сигнала

ШИМ дает возможность с помощью логических сигналов получать на выходе микроконтроллера аналоговый сигнал, который можно изменять от 0 до 1. Значение этого сигнала можно рассчитать по следующей формуле:

$$X_{\text{ВХ}} = \frac{t_1}{T}$$

```
rjmp m2
rev:
dec r17
cpi r17,0x00
brne m2
ldi r18,0x00
```

Если в регистре r18 установлено число 0x01 (`cpi r18,0x01`), то это значит, что происходит декремент кода (`dec r17`), если r18=0, то происходит инкремент кода (`inc r17`). При инкременте производится проверка достижения кодом заданной уставки (`cpi r17,0x0F`). При достижении этой уставки направление счета изменяется на противоположное (`ldi r18,0x01`). Аналогичное действие производится при декременте и достижении кодом нулевого значения (`cpi r17,0x00`).

7. По метке m2 производится вывод значения РОН r17 на индикацию и выход из подпрограммы обработки прерываний:

```
m2:
out PORTC,r17
m3:
sei
reti
```

Перед выходом из подпрограммы обработки прерывания разрешается срабатывание всех остальных прерываний (`sei`), после чего осуществляется выход из подпрограммы (`reti`).

Каждый таймер микроконтроллера Atmega 8535 имеет выводы, которые подключаются к портам ввода/вывода микроконтроллера и используются в качестве их альтернативной функции при работе в режиме ШИМ. Так, к выходу таймера T0 подключен вывод PB3, а к выходу таймера T2 подключен вывод PD7.

В микроконтроллере Atmega 8535 таймеры T0 и T2 работают в двух режимах широтно-импульсной модуляции: быстрый ШИМ и фазово-корректный ШИМ.

В режиме быстрого ШИМ счетный регистр **TCNT0(2)** таймера производит формирование пилообразной развертки (рис. 2, а), инкрементируя свое значение по каждому импульсу с предделителя, который устанавливается в регистре управления **TCCR0(2)** таймера T0 и T2. При достижении счетным регистром значения 255 происходит его автоматическое обнуление.

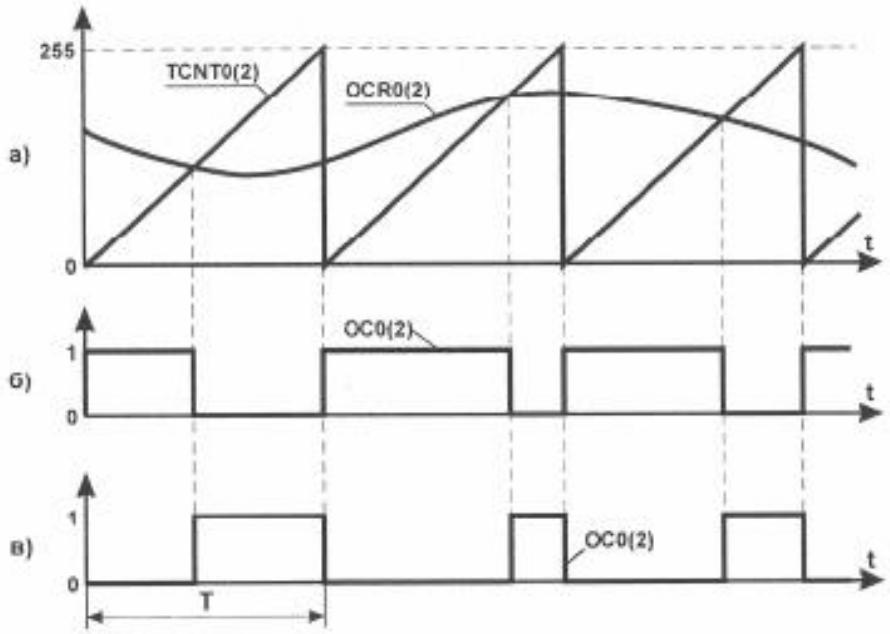


Рис. 2. Принцип работы таймеров T0 и T2 в режиме «Быстрый ШИМ»

В регистрах сравнения **OCR0(2)** таймеров T0 и T2 может быть записано любое число от 0 до 255. С этим числом сравнивается значение счетного регистра **TCNT0(2)** и при их равенстве происходит переключение вывода таймера OC0 или OC2 в соответствии с настройками регистра управления **TCCR0(2)** таймера. В случае, если при совпадении TCNT0(2)=OCR0(2) вывод OC0(2) таймера обнуляется, то ШИМ получается неинвертирующим (рис. 2, б), а если при совпадении вывод устанавливается в состояние логической «1», то ШИМ – инвертирующий (рис. 2, в).

В режиме фазового ШИМ счетный регистр **TCNT0(2)** формирует пилообразный сигнал развертки с нарастающим и спадающим фронтами. Сначала регистр инкрементирует свое значение от 0 до 255, а затем происходит его декремент от 255 до 0 (рис. 3, а).

Если при превышении счетным регистром **TCNT0(2)** значения, содержащегося в регистре сравнения **OCR0(2)**, происходит обнуление вывода OC0(2) таймера, то ШИМ является неинвертирующим (рис. 3, б). В противном случае ШИМ – инвертирующий (рис. 3, в).

Несмотря на то, что в режиме фазового ШИМ частота ШИМ-сигнала меньше, чем в режиме быстрого ШИМ, он обладает большей разрешающей способностью и используется для достижения большей точности модуляции.

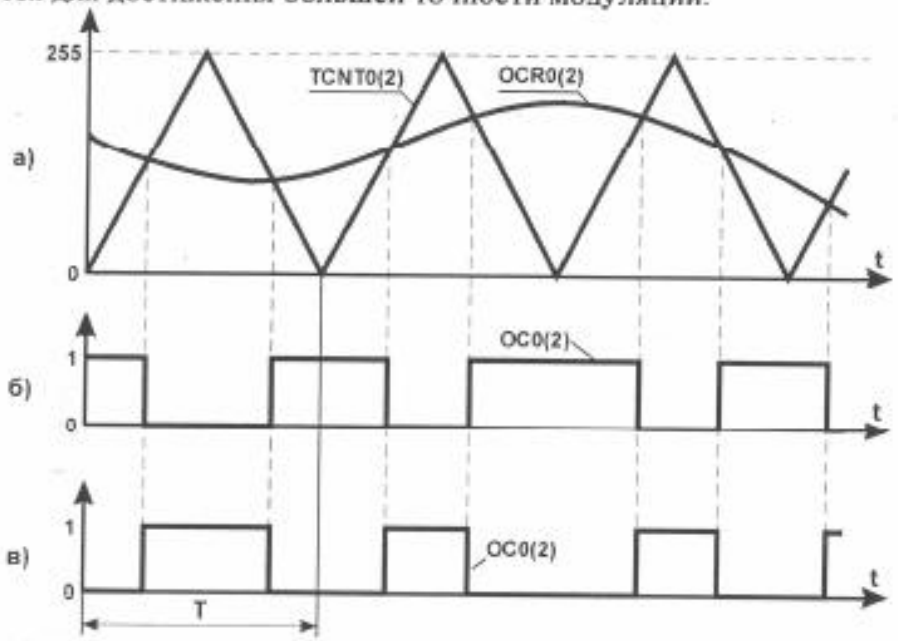


Рис. 3. Принцип работы таймеров T0 и T2 в режиме «Фазовый ШИМ»

Описание регистров таймера T0 в режиме ШИМ

Режим широтно-импульсной модуляции, как и все остальные режимы работы таймера T0, инициализируется в регистре управления TCCR0 (табл. 1).

Табл. 1. Регистр управления таймера T0 TCCR0

Бит	7	6	5	4	3	2	1	0
Название	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

За установку режима работы таймера отвечают биты WGM01:WGM00. При установке WGM01=0, WGM00=1 активируется режим фазового ШИМ, при установке WGM01=1, WGM00=1 активируется режим быстрого ШИМ.

При активации режима широтно-импульсной модуляции биты COM01 и COM00 определяют поведение вывода таймера T0, в качестве которого выступает вывод PB3 микроконтроллера. Поведение вывода в соответствии с этими битами представлено в табл. 2 и табл. 3.

Табл. 2. Поведение вывода PB3 (OC0) таймера T0 в режиме быстрого ШИМ

COM01	**COM00**	**Описание**
0	0	Нормальная функция порта, вывод OC0 отключен
0	1	Комбинация бит зарезервирована
1	0	Очистка OC0 при совпадении. Неинвертирующий ШИМ
1	1	Установка OC0 при совпадении. Инвертирующий ШИМ

Табл. 3. Поведение вывода PB3 (OC0) таймера T0 в режиме фазового ШИМ

COM01	COM00	Описание
0	0	Нормальная функция порта, вывод OC0 отключен
0	1	Комбинация бит зарезервирована
1	0	Очистка OC0 при совпадении при счете вверх. Неинвертирующий ШИМ
1	1	Установка OC0 при совпадении при счете вверх Инвертирующий ШИМ

Частота ШИМ-сигнала устанавливается битами CS02…CS00. Эти биты определяют источник задания частоты и коэффициент делителя (табл. 4).

Табл. 4. Установка источника задания частоты

CS02	CS01	CS00	Описание
0	0	0	Таймер остановлен
0	0	1	Источник – генератор частоты процессора. ($f_{T0} = f_{CLK}/1$)
0	1	0	Источник – генератор частоты процессора. ($f_{T0} = f_{CLK}/8$)
0	1	1	Источник – генератор частоты процессора. ($f_{T0} = f_{CLK}/64$)
1	0	0	Источник – генератор частоты процессора. ($f_{T0} = f_{CLK}/256$)
1	0	1	Источник – генератор частоты процессора. ($f_{T0} = f_{CLK}/1024$)
1	1	0	Внешний сигнал на выводе T0. Спадающий фронт сигнала.
1	1	1	Внешний сигнал на выводе T0. Нарастающий фронт сигнала.

В соответствии с табл. 4, частота ШИМ рассчитывается следующим образом:
– для режима быстрого ШИМ: $f = f_{T0}/256$;
– для режима фазового ШИМ: $f = f_{T0}/512$.

Описание регистров таймера T2 в режиме ШИМ

Режим широтно-импульсной модуляции, как и все остальные режимы работы таймера T2, инициализируется в регистре управления TCCR2 (табл. 5).

Табл. 5. Регистр управления таймера T2 TCCR2

Бит	7	6	5	4	3	2	1	0
Название	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20

Описание и назначение управляющих бит регистра TCCR2 таймера T2 аналогичны описанным битам регистра TCCR0 таймера T0, поэтому отдельно не