Лабораторная работа № 7

# Использование таймеров-счетчиков в режимах Normal и СТС

Цель работы:

1. Изучение принципа работы внутренних таймеров-счетчиков микро- контроллера ATmega8535.
2. Программирование таймеров-счетчиков для режимов Normal и СТС.

Оборудование и программное обеспечение:

1. Лабораторный макет MK8535 в составе: модуль базовый МБ8535 +

модуль сменный МС01.

1. Кабель соединительный для программирования МК.
2. Персональный компьютер.
3. Интегрированная среда программирования CodeVisionAVR.

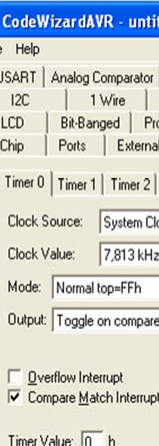
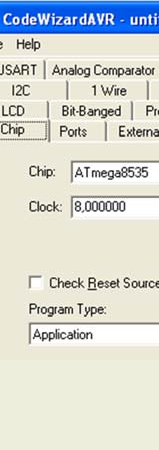
# Часть 1. Исследование режима Normal

**Подготовка к работе**

Рассмотрим и выполним пример, в котором будет использован таймер-счетчик T0 в режиме Normal. Выход таймера-счетчика соединён с выводом OC0 МК ATmega8535 (альтернативная функция вывода 3 порта B), к которому подключен динамик модуля МС01 (см. рис. 50).

В регистр сравнения OCR таймера-счетчика записано значение 0xFF. В моменты достижения счетчиком этого значения будет генериро- ваться прерывание TIMER0 COMP (совпадение таймера/счетчика T0, вектор 20), а на выводе OC0 будет переключаться сигнал и в динамике будут слышны щелчки. Процедура обработки прерывания увеличивает на 1 значение в регистре PORTA, используемого в качестве двоичного

счетчика, и выводит его на 8-разрядный светодиодный индикатор, под- ключенный к порту A. Таким образом, показания индикатора изменяются синхронно со щелчками динамика.



а) б)

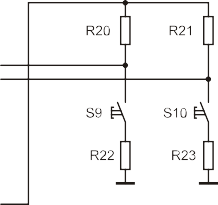
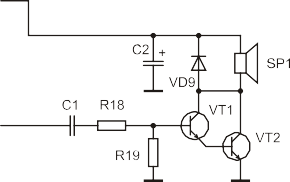
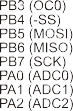
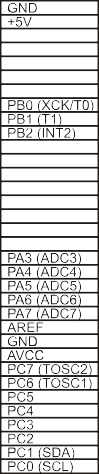
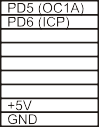


Рис. 51. Выбор тактовой частоты (а) и настройка таймера-счетчика T0 (б)

Оценим частоту сигнала на выходе OC0 при таких настройках:

*f*   *fclk* \_ *I* /*O* 

8000000 Гц

 7813 Гц  15,3 Гц.

Рис. 50. Структурная схема лабораторного макета. Для модуля МС01 показана только та часть схемы, которая используется в данной работе

Создадим проект в CodeVisionAVR и зададим нужный режим ра- боты МК. Порта A настроить на вывод. Линию 3 порта B настроить на вывод, чтобы разрешить прохождение сигнала от таймера на вывод OC0.

На вкладке **Timers** окна **CodeWizardAVR** (рис. 51) выберем **Timer0**, источник тактовых сигналов – тактовые импульсы МК, такто- вая частота таймера-счетчика – 7,813 кГц (это 8 Мгц / 1024), режим ра- боты таймера-счетчика – Normal, режим выхода OC0 – изменение на противоположное значение по каждому совпадению с OCR, разрешить прерывание по совпадению, значение регистра сравнения – 0xFF.

*OC* 2  *N*  (1 *OCR*) 2 1024  (1 255) 2  256

Дополним текст программы, сформированный **CodeWizardAVR**,

кодом обработки прерывания от таймера:

# #include <mega8535.h>

**// Timer 0 output compare interrupt service routine interrupt [TIM0\_COMP] void timer0\_comp\_isr(void)**

**{**

# ++PORTA;

**}**

# void

**{**

**main(void)**

**PORTA = 0x00;**

**// Настроить**

**все линии порта A**

**DDRA = 0xFF; // на вывод**

**PORTB = 0x00; // К выходу OC0**

**DDRB = 0x08; // подключен динамик**

а на выводе OC0 будет переключаться сигнал и в динамике будут слышны щелчки. Процедура обработки прерывания увеличивает на 1 значение в регистре PORTA и выводит его на 8-разрядный двоичный светодиод- ный индикатор, подключенный к порту A.

# // Timer/Counter 0 initialization

Поскольку при

изменении значения OCR0 будет изменяться

**// Clock source: System Clock**

**// Clock value: 7,813 kHz**

**// Mode: Normal top = FFh**

**// OC0 output: Toggle on compare match TCCR0 = 0x15;**

**TCNT0 = 0x00; OCR0 = 0xFF;**

**TIMSK = 0x02; // Timer Interrupt initialization #asm(«sei») // Global enable interrupts**

**while (1);**

**}**

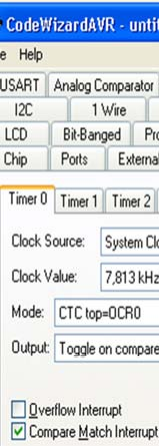
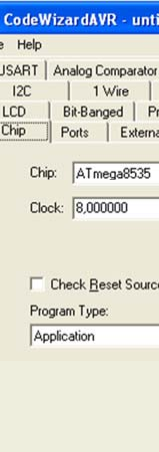
# Задание

и время, необходимое для достижения равенства TCNT0 и OCR0, то мы будем наблюдать изменение периода импульсов на выходе OC0 микро- контроллера. Одновременно будет изменяться частота звука в динамике с скорость счета на светодиодном индикаторе.

Создадим проект в CodeVisionAVR и зададим нужный режим ра- боты МК. Порта A настроить на вывод. Линию 3 порта B настроить на вывод, чтобы разрешить прохождение сигнала от таймера на вывод OC0.

На вкладке **Timers** окна **CodeWizardAVR** (рис. 52) выберем **Timer0**, источник тактовых сигналов – тактовые импульсы МК, тактовая частота таймера-счетчика – 7,813 кГц (это 8 Мгц/1024), режим работы таймера-счетчика – CTC, режим выхода OC0 – изменение на противопо- ложное значение по каждому совпадению с OCR0, разрешить прерывание по совпадению, значение регистра сравнения – 0x7F.

1. Создать проект и реализовать рассмотренный выше пример программы. При помощи осциллографа наблюдать сигнал на выходе OC0 микро- контроллера. Каковы частота и скважность импульсов на выходе OC0 и почему?



1. Выбрать другие тактовые частоты для таймера T0 и проследить изме- нения частоты звука и приращения счетчика.

# Часть 2. Исследование режима CTC

**Подготовка к работе**

Рассмотрим и выполним пример, в котором таймер-счетчик T0 будет работать в режиме CTC. В этом режиме мы будем изменять зна- чение в регистре сравнения OCR0 с помощью кнопок S9 и S10. Началь- ное значение OCR0 установим равным 127 (0x7F), кнопкой S9 можно будет увеличивать его до предельного значения 255 (0xFF), а кнопкой S10 – уменьшать до нуля. Выход таймера-счетчика соединён с выводом OC0 МК ATmega8535 (альтернативная функция вывода 3 порта B), к которому подключен динамик модуля МС01 (см. рис. 50). В моменты достижения счетчиком значения, равного OCR0, будет генерироваться прерывание TIMER0 COMP (совпадение таймера/счетчика T0, вектор 20),

а) б)

Рис. 52. Выбор тактовой частоты (а) и настройка таймера-счетчика T0 (б)

Оценим изменение частоты сигнала на выходе OC0 при таких настройках:

# // Mode: CTC top=OCR0

**// OC0 output: Toggle on compare match TCCR0 = 0x1D;**

*fOC* \_ *OCR*255

 7813 Гц  15, 3 Гц,

2  256

*fOC* \_ *OCR*0

 7813 Гц  3906, 5 Гц. .

2 1

# TCNT0 = 0x00; OCR0 = 0x7F;

Итак, ожидается изменение частоты от 15,3 Гц до 3906,5 Гц.

Ниже приведен текст программы для МК ATmega8535. В нем опущены те строки, автоматически сформированные CodeWizardAVR, которые не существенны для данного примера.

# #include <mega8535.h>

**// External Interrupt 0 service routine interrupt [EXT\_INT0] void ext\_int0\_isr(void)**

**{**

# if (OCR0 < 0xFF) ++OCR0;

**}**

# // External Interrupt 1 service routine interrupt [EXT\_INT1] void ext\_int1\_isr(void)

**{**

**if (OCR0 > 0) --OCR0;**

**// External Interrupt(s) initialization**

**// INT0: On**

**// INT0 Mode: Falling Edge**

**// INT1: On**

**// INT1 Mode: Falling Edge**

**// INT2: Off GICR |= 0xC0; MCUCR = 0x0A; MCUCSR = 0x00; GIFR = 0xC0;**

**TIMSK = 0x02; // Timer/Counter Interrupt ini- tialization**

**#asm(«sei») // Global enable interrupts PORTA = counter;**

**while (1);**

**}**

**}**

# //Timer 0 output compare interrupt service routine interrupt [TIM0\_COMP] void timer0\_comp\_isr(void)

**{**

# ++PORTA;

**}**

# void main(void)

**{**

# Задание

1. Создать проект и реализовать рассмотренный выше пример программы. При помощи осциллографа наблюдать сигнал на выходе OC0 микро- контроллера. Как значение OCR0 влияет на частоту импульсов на вы- ходе OC0 ?
2. Выбрать другие тактовые частоты для таймера-счетчика и проследить изменения частоты звука и приращения счетчика.

**PORTA = 0x00; // Настроить все линии порта A DDRA = 0xFF; // на вывод**

**PORTB = 0x00; // К выходу OC0**

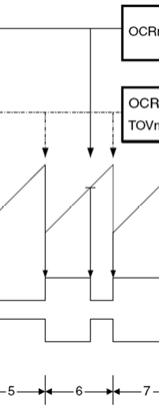
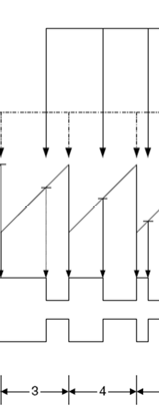
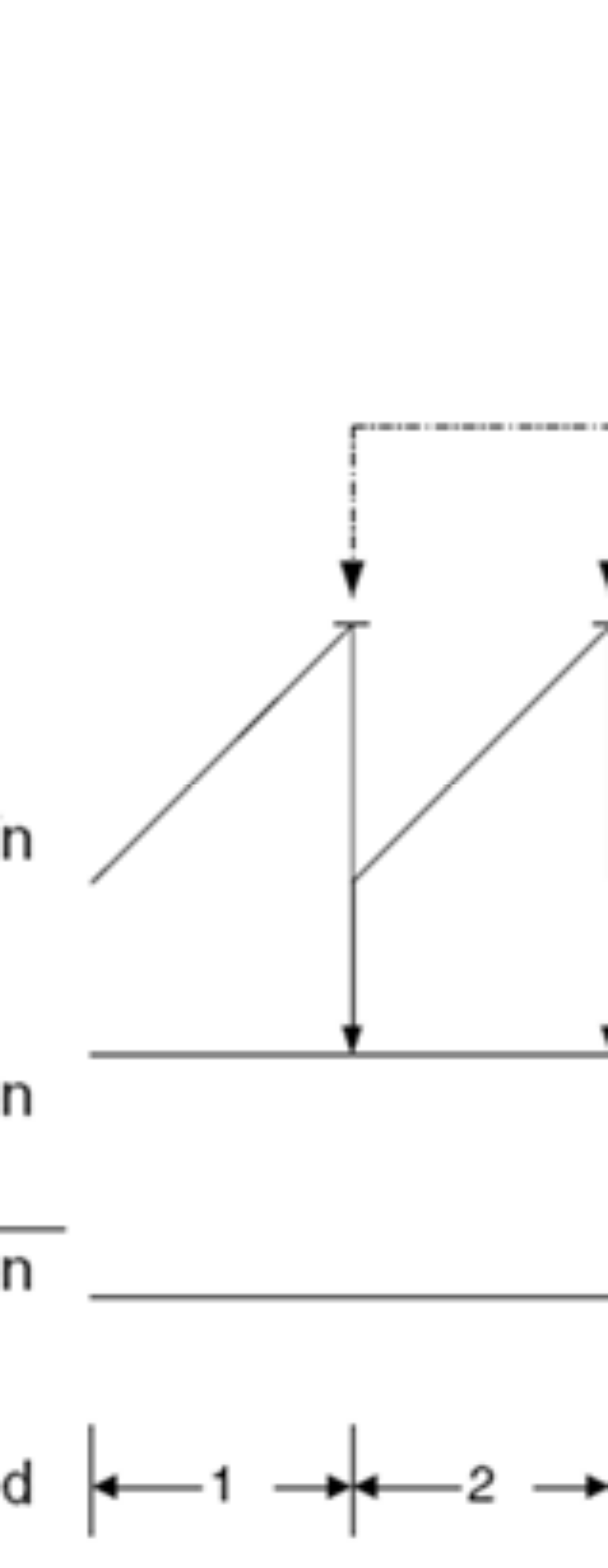
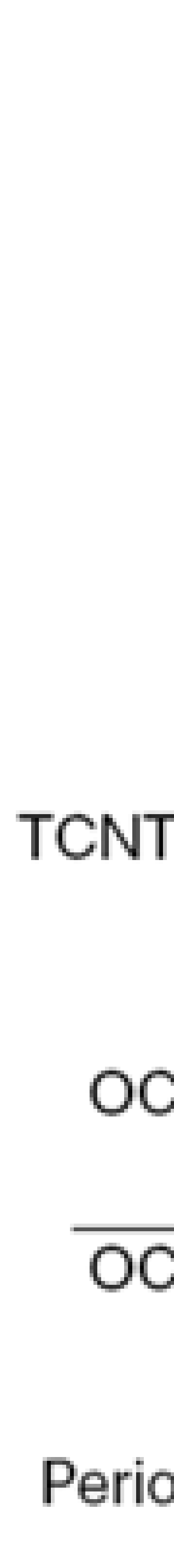
**DDRB = 0x08; // подключен динамик**

**// Timer/Counter 0 initialization**

**// Clock source: System Clock**

**// Clock value: 7,813 kHz**

* 1. **РЕЖИМЫ FAST PWM И РЕЖИМ PHASE CORRECT PWM**



**Режим Fast PWM**

Режим Fast PWM (Fast Pulse Width Modulation – Быстродейству- ющий ШИМ) позволяет генерировать высокочастотный сигнал с ши- ротно-импульсной модуляцией. В связи с высокой частотой генерируе- мого сигнала данный режим с успехом может использоваться в таких приложениях, как регулирование мощности, выпрямление, цифроанало- говое преобразование и др.

Счетный регистр в этом режиме функционирует как суммирую- щий счетчик, инкремент которого осуществляется по каждому импульсу тактового сигнала clkT0. Состояние счетчика изменяется от 0x00 до 0xFF, после чего счетный регистр сбрасывается и цикл повторяется. При достижении счетчиком максимального значения устанавливается флаг прерывания TOV0 регистра TIFR. При равенстве содержимого счетного регистра и регистра сравнения OCR0 устанавливается флаг OCF0 реги- стра TIFR.

Особенностью работы схемы сравнения в этом режиме является двойная буферизация записи в регистр OCR0, заключающаяся в том, что записываемое число на самом деле сохраняется в специальном буфер- ном регистре. А изменение содержимого регистра сравнения происхо-

Рис. 53. Формирование ШИМ-сигналов в режиме Fast PWM

Частота генерируемого сигнала

*f*

дит только в момент достижения счетчиком максимального значения 0xFF. Благодаря такому решению исключается появление несимметрич- ных импульсов сигнала на выходе модулятора (помех), которые были

*fOC* 

*clk* \_ *I* / *O* ,

*N*  256

бы неизбежны при непосредственной записи в регистр сравнения.

Поведение вывода ОС0 микроконтроллера в этом режиме опреде- ляется содержимым разрядов СОМ01:СОМ00 (табл. 17 и рис. 53).

*Таблица 17*

Поведение вывода ОС0 в режиме Fast PWM

где *N* – коэффициент деления предделителя (1, 8, 64, 256 или 1024).

Отдельно следует рассмотреть случаи, когда в регистре сравнения находятся предельно возможные значения (0x00 и 0xFF). В первом слу- чае, если содержимое регистра сравнения OCR0 равно 0x00, на выходе ОС0 при каждом 256-м такте сигнала clkT0 будут наблюдаться короткие выбросы. Если же содержимое регистра сравнения OCR0 равно 0xFF,

то вывод ОС0 переключится в устойчивое состояние, установками разрядов СОМ01:СОМ00.

|  |  |  |
| --- | --- | --- |
| СОМ01 | СОМ00 | Описание |
| 0 | 0 | Таймер/счетчик отключен от вывода ОС |
| 0 | 1 | Зарезервировано |
| 1 | 0 | Сбрасывается в 0 при равенстве регистров TCNT и OCR. Устанавливается в 1 при TCNT = 0x00 (неинвертированный ШИМ-сигнал) |
| I | 1 | Устанавливается в 1 при равенстве регистров TCNT и OCR. Сбрасывается в 0 при TCNT = 0x00 (инвертированный ШИМ-сигнал) |

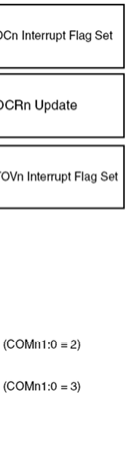
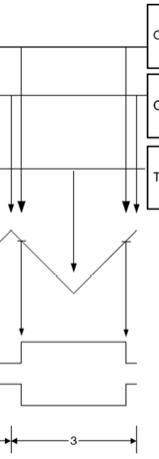
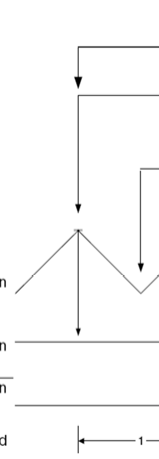
определяемое

# Режим Phase Correct PWM

Режим Phase Correct PWM (ШИМ с точной фазой), как и режим Fast PWM, предназначен для генерации сигналов с широтно-импульсной модуляцией. Однако в этом режиме счетный регистр функционирует как

реверсивный счетчик, изменение состояния которого осуществляется

по каждому импульсу тактового сигнала clkT0. Состояние счетчика сна- чала изменяется от 0x00 до 0xFF, а затем обратно до 0x00. Соответ- ственно максимальная частота сигнала в этом режиме в два раза меньше максимальной частоты сигнала в режиме Fast PWM. Тем не менее, бла- годаря «симметричности» изменения состояния счетчика, режим Phase Correct PWM предпочтительнее использовать для решения задач управ- ления двигателями.



При достижении счетчиком максимального (0xFF) значения про- исходит смена направления счета, но счетчик остается в этом состоянии в течение одного периода сигнала clkT0. При достижении счетчиком ми- нимального значения (0x00) также происходит смена направления счета и одновременно устанавливается флаг прерывания TOV0 регистра TIFR. При равенстве содержимого счетного регистра и регистра сравнения OCR0 устанавливается флаг OCF0 регистра TIFR и изменяется состоя- ние вывода ОС0. Каким образом изменяется состояние вывода ОС0 определяется содержимым разрядов СОМ01:СОМ00 регистра TCCR0 (табл. 18 и рис. 54).

*Таблица 18*

Поведение вывода ОС0 в режиме Phase Correct PWM

Рис. 54. Формирование ШИМ-сигнала в режиме Phase Correct PWM

*Таблица 19*

Устойчивые состояния выхода схемы сравнения

|  |  |  |
| --- | --- | --- |
| СОМ01 | СОМ00 | Описание |
| 0 | 0 | Таймер/счетчик отключен от вывода ОС |
| 0 | 1 | Зарезервировано |
| 1 | 0 | Сбрасывается в 0 при прямом счете и устанавливается в 1  при обратном счете (неинвертированный ШИМ-сигнал) |
| I | 1 | Устанавливается в 1 при прямом счете и сбрасывается в 0  при обратном счете (инвертированный ШИМ-сигнал) |

|  |  |  |  |
| --- | --- | --- | --- |
| СОМ01 | СОМ00 | Регистр OCR | Состояние вывода ОС |
| 1 | 0 | 0x00 | 0 |
| 1 | 0 | 0xFF | 1 |
| 1 | 1 | 0x00 | 1 |
| 1 | 1 | 0xFF | 0 |

Для избежания несимметричных выбросов в этом режиме тоже реализована двойная буферизация записи в регистр OCR0. Благодаря этому действительное изменение содержимого регистра сравнения про- исходит только в момент достижения счетчиком максимального значе- ния 0xFF.

Частота генерируемого в рассматриваемом режиме сигнала

Если в регистр сравнения записать значение 0x00 или 0xFF, то при следующем совпадении состояния счетчика и содержимого регистра сравнения выход схемы сравнения переключится в устойчивое состоя-

*fOC*

  *fclk* \_ *I* / *O* ,

*N*  512

ние согласно табл. 19. где *N* – коэффициент деления предделителя (1, 8, 64, 256 или 1024).