Микроконтроллеры AVR семейства Mega

Таймеры

Таймеры. Общие сведения

- В программировании, **таймером** является объект, возбуждающий событие по истечении заданного промежутка времени. Событием является посылка сообщения, вызов функции, установка параметров объекта ядра и т.д.

- Микроконтроллеры AVR содержат в своем составе несколько встроенных таймером, которые по назначению делятся на две группы:
  - таймеры общего назначения;
  - сторожевой таймер (Watchdog Timer).

Таймеры общего назначения

Таймеры общего назначения:

- Эту группу таймеров называют **«таймеры/счетчики»**.
- Используются для формирования различных интервалов времени и прямоугольных импульсов заданной частоты.
- Могут работать в режиме счетчика и подсчитывать тактовые импульсы заданной частоты, измеряя таким образом длительность внешних сигналов.
- В МК AVR применяются 8-ми разрядные и 16-ти разрядные таймеры/счетчики, их количество зависит от конкретной модели контроллера.
- Во всех моделях микроконтроллеров семейства присутствуют, как минимум, два таймера/счетчика – T0 и T1.

Таймеры общего назначения

- **Таймер/счетчик Т0**

  Имеет минимальный набор функций:

- может использоваться только для отсчета и измерения временных интервалов или как счетчик внешних событий;
- (+) возможность генерации сигналов ШИМ фиксированной разрядности;
- (+) работа в асинхронном режиме в качестве часов реального времени.

- **Таймер/счетчик Т1**

- может использоваться для отсчета временных интервалов и как счетчик внешних событий;
- может выполнять запоминание своего состояния по внешнему сигналу;
- может работать в качестве многоканального ШИМ модулятора переменной разрядности.

- **Таймер/счетчик Т2** полностью аналогичен таймеру/счетчику Т0.
- **Таймер/счетчик Т3** идентичен таймеру/счетчику Т1.

Сторожевой таймер

- **Сторожевой таймер (Watchdog timer)** – аппаратно реализованная схема контроля за зависанием системы.
- Представляет собой таймер, который периодически должен сбрасывается контролируемой системой. Если сброса не произошло в течение некоторого интервала времени, происходит принудительная перезагрузка системы.
- В некоторых случаях сторожевой таймер может посылать системе сигнал на перезагрузку («мягкая» перезагрузка), в других же — перезагрузка происходит аппаратно (например, замыканием контактов кнопки Reset).

Таймеры. Общие сведения

Источник тиков таймера

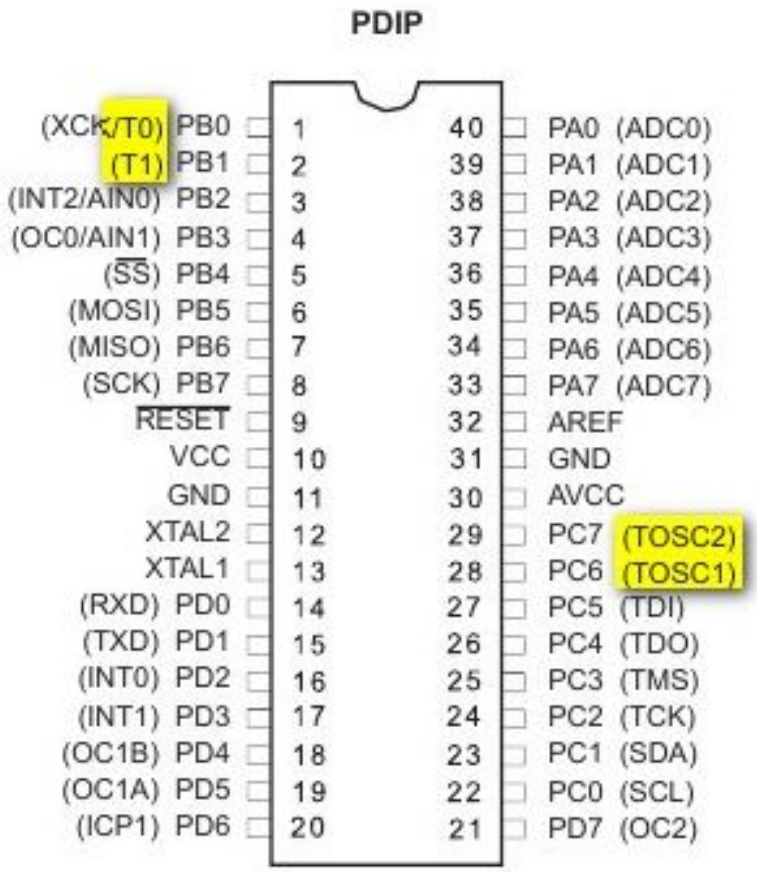
- Таймер/Счетчик (ТС) считает либо тактовые импульсы от встроенного тактового генератора, либо со счетного входа. При соответствующей настройке ТС будет считать либо передний (перепад с 0-1), либо задний (перепад 1-0) фронт импульсов, пришедших на входы.

- **Важно**, чтобы частота входящих импульсов **не превышала** тактовую частоту процессора, иначе он не успеет обработать импульсы.

- Некоторые таймеры способны работать в асинхронном режиме - считать не тактовые импульсы процессора, а импульсы своего собственного генератора, работающего от отдельного кварца.

Расстановка выводов ATmega16

- ножки T1 и T0 - счетные входы Timer 0 и Timer 1.
- TOSC1 и TOSC2 - импульсы своего собственного генератора, работающего от отдельного кварца



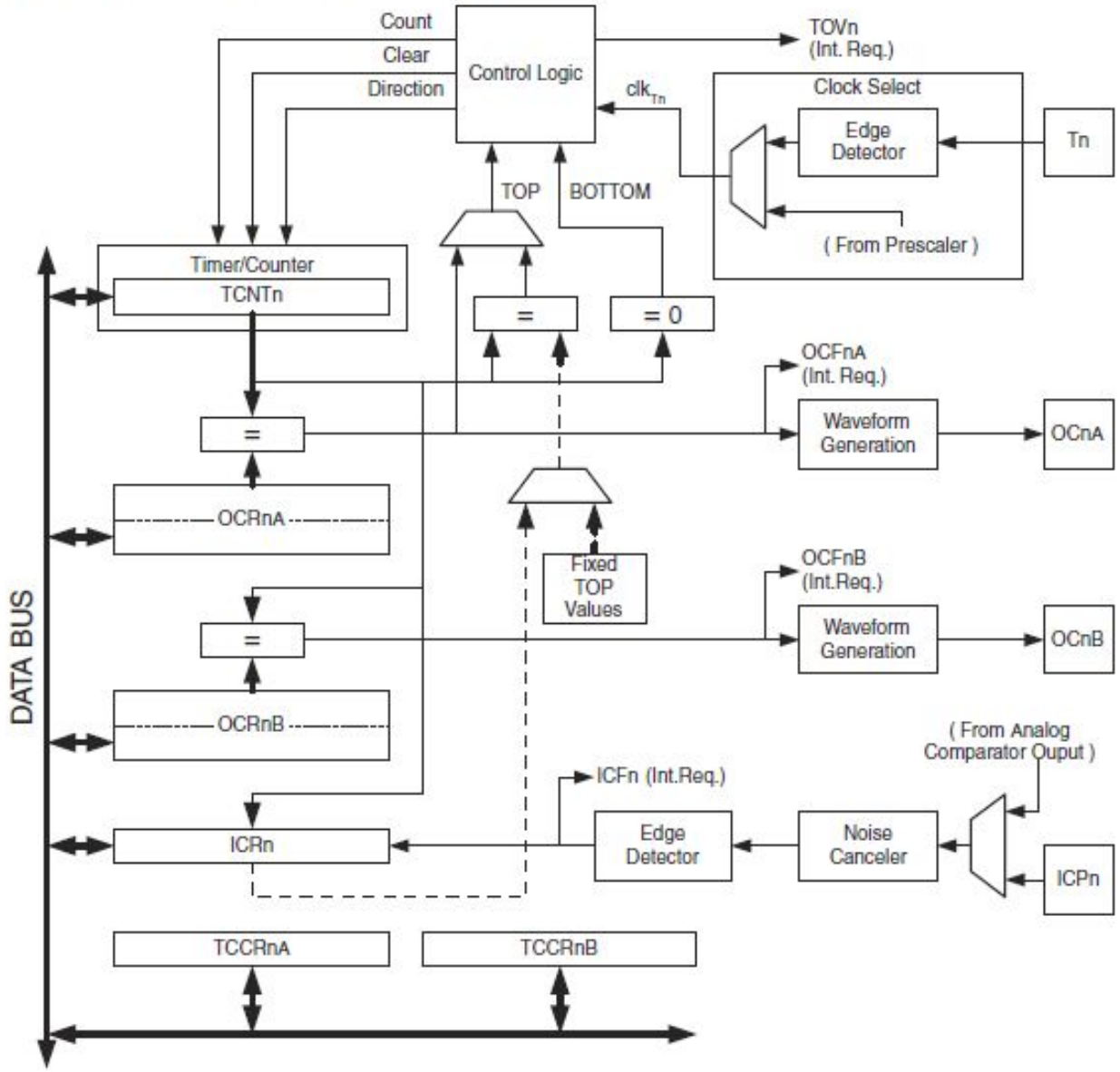
Таймеры. Общие сведения

Предделитель

- Если таймер считает импульсы от тактового генератора или от своего внутреннего, то их еще можно пропустить через предделитель. Делить можно на 8, 32, 64, 128, 256, 1024.

- Предделитель работает все время, вне зависимости от того включен таймер или нет. Поэтому предделители нужно сбрасывать.
- Также надо учитывать и то, что предделитель един для всех счетчиков, поэтому, сбрасывая его, надо учитывать то, что у другого таймера собьется задержка до следующего тика.

Figure 32. 16-bit Timer/Counter Block Diagram[1]



Таймеры. Общие сведения

Счетный регистр

- Весь счет накапливается в счетном регистре TCNTx
- Он может быть как 8-, так и 16-разрядным (состоит из 2-х частей TCNTxH и TCNTxL — старший и младший байты соответственно).

- Запись в старший регистр (TCNTxH) ведется вначале в регистр TEMP.

- Считывается все в обратном порядке. Сначала младший байт (при этом старший – в TEMP), потом старший.

Timer/Counter 1 – TCNT1H and TCNT1L

Bit	7	6	5	4	3	2	1	0	
	TCNT1[15:8]								TCNT1H
	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Регистр TIMSK

- Регистр TIMSK отвечает за прерывания, генерируемые при работе таймеров микроконтроллера.
- За разрешение прерывания переполнения счетного регистра TCNT1 таймера 1 отвечает бит TOIE1(Timer1 Overflow Interrupt Enable).

Timer/Counter Interrupt Mask Register – TIMSK[1]

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Таймеры. Контрольные регистры

Контрольные регистры

Регистр TCCRx.

	7	6	5	4	3	2	1	0	
	–	–	–	–	–	CS02	CS01	CS00	ATmega8x ATmega163x
Чтение (R)/Запись (W)	R	R	R	R	R	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	ATmega8515x ATmega16x ATmega162x ATmega32x ATmega64x/128x
Чтение (R)/Запись (W)	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

	7	6	5	4	3	2	1	0	
	FOC0	PWM0	COM01	COM00	CTC0	CS02	CS01	CS00	ATmega161x ATmega323x
Чтение (R)/Запись (W)	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

а)

	7	6	5	4	3	2	1	0	
	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	ATmega8x ATmega16x ATmega162x ATmega32x ATmega64x/128x
Чтение (R)/Запись (W)	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

	7	6	5	4	3	2	1	0	
	FOC2	PWM2	COM21	COM20	CTC2	CS22	CS21	CS20	ATmega161x ATmega163x ATmega323x
Чтение (R)/Запись (W)	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

б)

<table>
<tr><th>Раз-ряд</th><th>Название</th><th>Описание</th></tr>
<tr><td>7</td><td>FOCn</td><td>Принудительное изменение состояния вывода OCn (режимы Normal и CTC). При записи лог. 1 в этот разряд состояние вывода OCn изменяется в соответствии с установкам разрядов COMn1:COMn0. Прерывание при этом не генерируется и сброс таймера (в режиме CTC) не производится. В режимах Fast PWM и Phase Correct PWM этот разряд должен быть сброшен в «0». При чтении разряда всегда возвращается «0».</td></tr>
<tr><td>6, 3</td><td>WGMn1:WGMn0 (CTCn, PWMn в ATmega161x)</td><td>Режим работы таймера/счетчика. Эти разряды определяют режим работы таймера/счетчика следующим образом:
<table>
<tr><th>Номер режима</th><th>WGMn1 (CTCn)</th><th>WGMn0 (PWMn)</th><th>Режим работы таймера/счетчика Tn</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Normal</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Phase correct PWM</td></tr>
<tr><td>2</td><td>1</td><td>0</td><td>CTC (сброс при совпадении)</td></tr>
<tr><td>3</td><td>1</td><td>1</td><td>Fast PWM</td></tr>
</table></td></tr>
<tr><td>5, 4</td><td>COMn1:COMn0</td><td>Режим работы блока сравнения. Эти разряды определяют поведение вывода OCn при наступлении события «Совпадение». Влияние содержимого этих разрядов на состояние вывода зависит от режима работы таймера/счетчика</td></tr>
<tr><td>2...0</td><td>CSn2...CSn0</td><td>Управление тактовым сигналом. Эти разряды определяют источник тактового сигнала микроконтроллера. Действие этих разрядов зависит от исполнения таймера/счетчика и будет описано ниже</td></tr>
</table>

Таймеры. Контрольные регистры

Контрольные регистры

Регистр TCCRx.

- Первые 3 бита этого регистра: CSx2..CSx0 отвечают за установку предделителя и источник тактового сигнала:

- 000 - таймер остановлен
- 001 - предделитель равен 1, то есть выключен. Таймер считает тактовые импульсы
- 010 - предделитель равен 8, тактовая частота делится на 8
- 011 - предделитель равен 64, тактовая частота делится на 64
- 100 - предделитель равен 256, тактфовая частота делится на 256
- 101 - предделитель равен 1024, тактовая частота делится на 1024
- 110 - тактовые импульсы идут от ножки T0 на переходе с 1 на 0
- 111 - тактовые импульсы идут от ножки T0 на переходе с 0 на 1

Таймеры. Прерывания

Прерывания

- За прерывания от таймеров отвечают регистры TIMSK, TIFR.
- У мощных AVR, таких как ATMega128, есть еще ETIFR и ETIMSK - своего рода продолжение, так как таймеров там больше.
- TIMSK – это регистр масок. То есть биты, находящиеся в нем, локально разрешают прерывания.
- За прерывание по переполнению отвечают биты:
  - TOIE - разрешение на прерывание по переполнению таймера 0
  - TOIE1 - разрешение на прерывание по переполнению таймера 1
  - TOIE2 - разрешение на прерывание по переполнению таймера 2
- Регистр TIFR это непосредственно флаговый регистр. Когда какое-то прерывание срабатывает, то флаг прерывания устанавливается.

Пример, код на С:

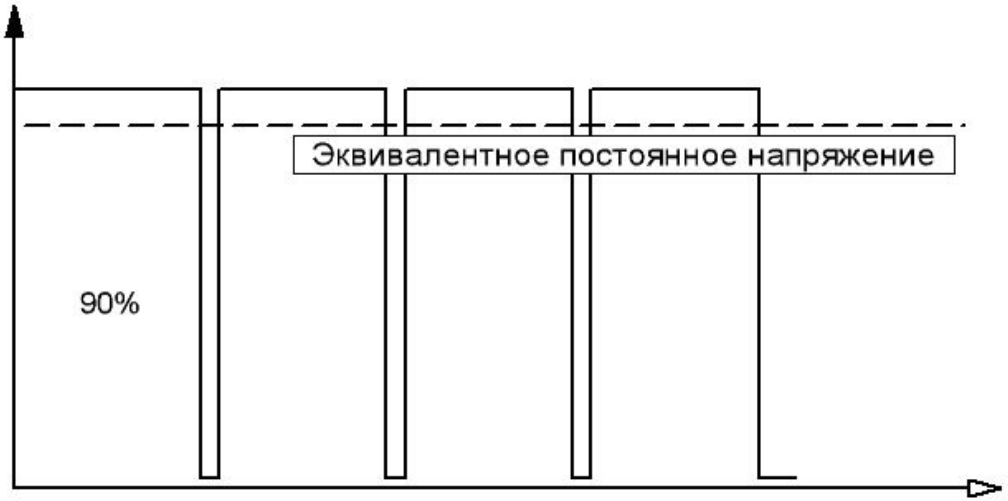
```
#define F_CPU 16000000UL

#include <avr/io.h>
#include <avr/interrupt.h>

uint8_t num=;

ISR(TIMER1_OVF_vect)
{
PORTD=(1<<num);
num++;

if(num>2)
{
num=;
}

TCNT1=61630;//Начальное значение таймера
}

int main(void)
{
DDRD|=(1<<PD0)|(1<<PD1)|(1<<PD2);

TCCR1B|=(1<<CS12)|(1<<CS10);//Предделитель = 1024
TIMSK|=(1<<TOIE1);//Разрешить прерывание по переполнению таймера 1
TCNT1=61630;//Начальное значение таймера

sei();//Разрешить прерывания

while(1)
{
//Основной цикл программы, он пуст, так как вся работа в прерывании
}
}
```

Широтно Импульсная Модуляция

Широтно Импульсная Модуляция

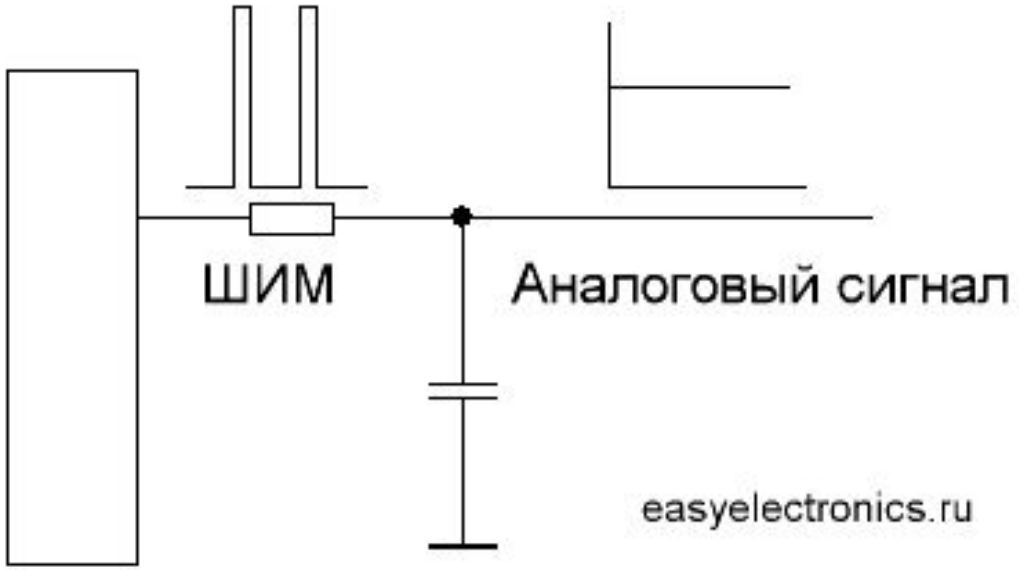
- Широтно-Импульсная Модуляция **(**PWM - Pulse Width Modulation**)** это способ задания аналогового сигнала цифровым методом.

- Меняя **скважность** (отношение длительности периода к длительности импульса) можно плавно менять эту площадь, а значит и напряжение на выходе.



Широтно Импульсная Модуляция

- В качестве сглаживающей интегрирующей цепи в ШИМ может быть применена RC цепочка:

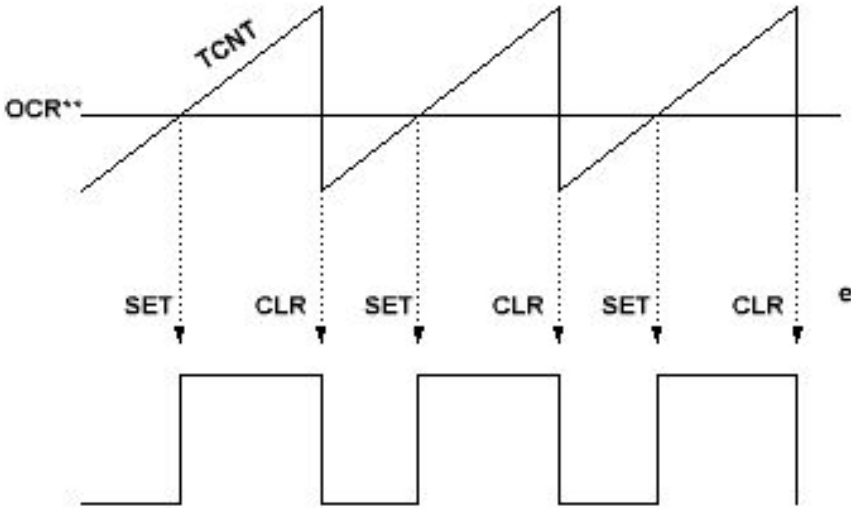


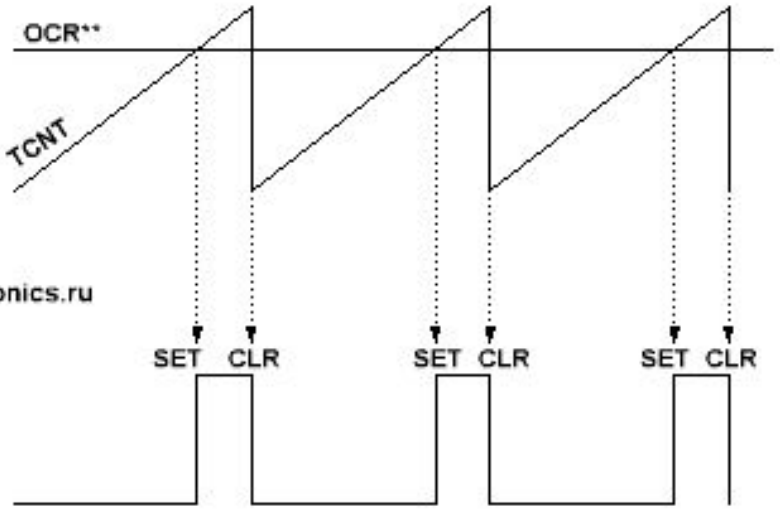
Аппаратная реализация ШИМ

Аппаратная реализация ШИМ

- В случае **ATMega** проще всего сделать на его ШИМ-генераторе, который встроен в таймеры
-
- У таймера есть регистр сравнения **OCR**. Когда значение в счётном регистре таймера достигает значения находящегося в регистре сравнения, то могут возникнуть следующие аппаратные события:
  - Прерывание по совпадению
  - Изменение состояния внешнего выхода сравнения **OC**.
- Выходы сравнения выведены наружу, на выводы микроконтроллера.

Аппаратная реализация ШИМ

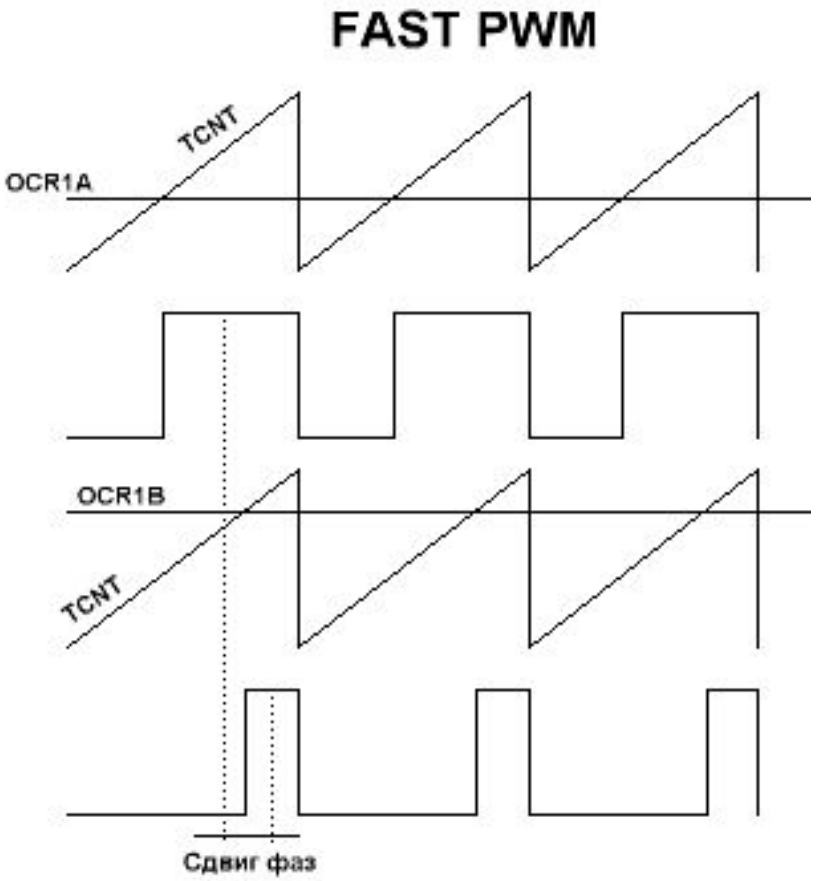




Режимы ШИМ.

Быстрая ШИМ (Fast PWM)

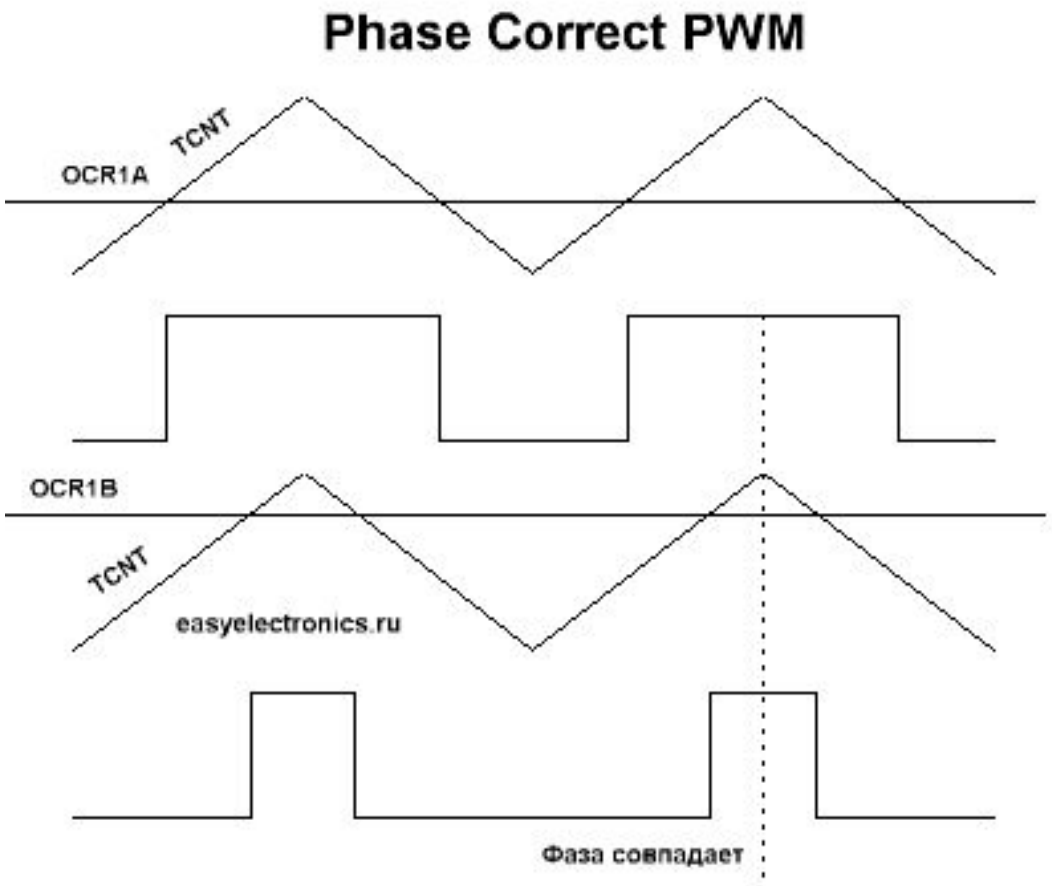
- В этом режиме счетчик считает от нуля до 255, после достижения переполнения сбрасывается в нуль и счет начинается снова. Когда значение в счетчике достигает значения регистра сравнения, то соответствующий ему вывод **OCxx** сбрасывается в ноль. При обнулении счетчика этот вывод устанавливается в 1.



Режимы ШИМ

ШИМ с фазовой коррекцией (Phase Correct PWM)

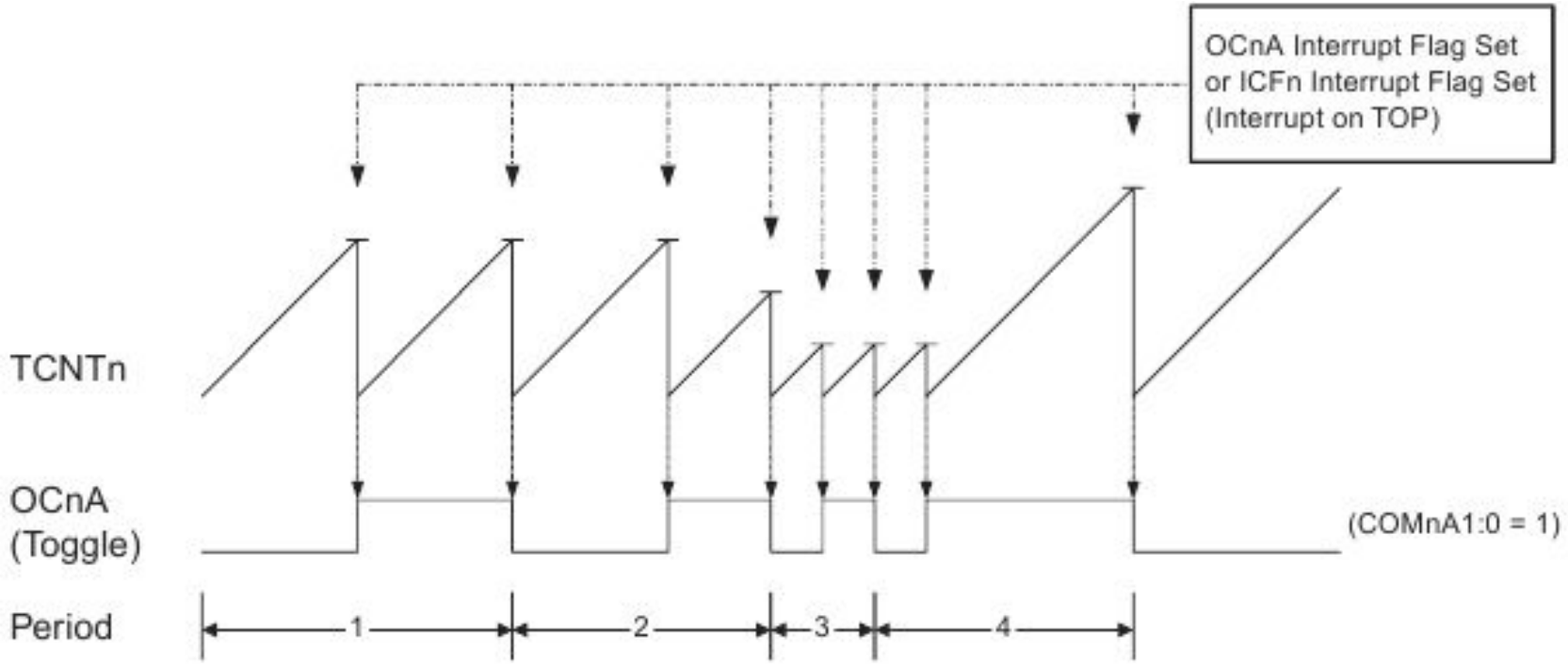
- ШИМ с точной фазой. Работает похоже, но тут счетчик считает несколько по-другому. Сначала от 0 до 255, потом от 255 до 0. Вывод OCxx при первом совпадении сбрасывается, при втором устанавливается. Частота ШИМ при этом падает вдвое, из-за большего периода.



Режимы ШИМ.

Сброс по совпадению (Clear Timer On Compare)

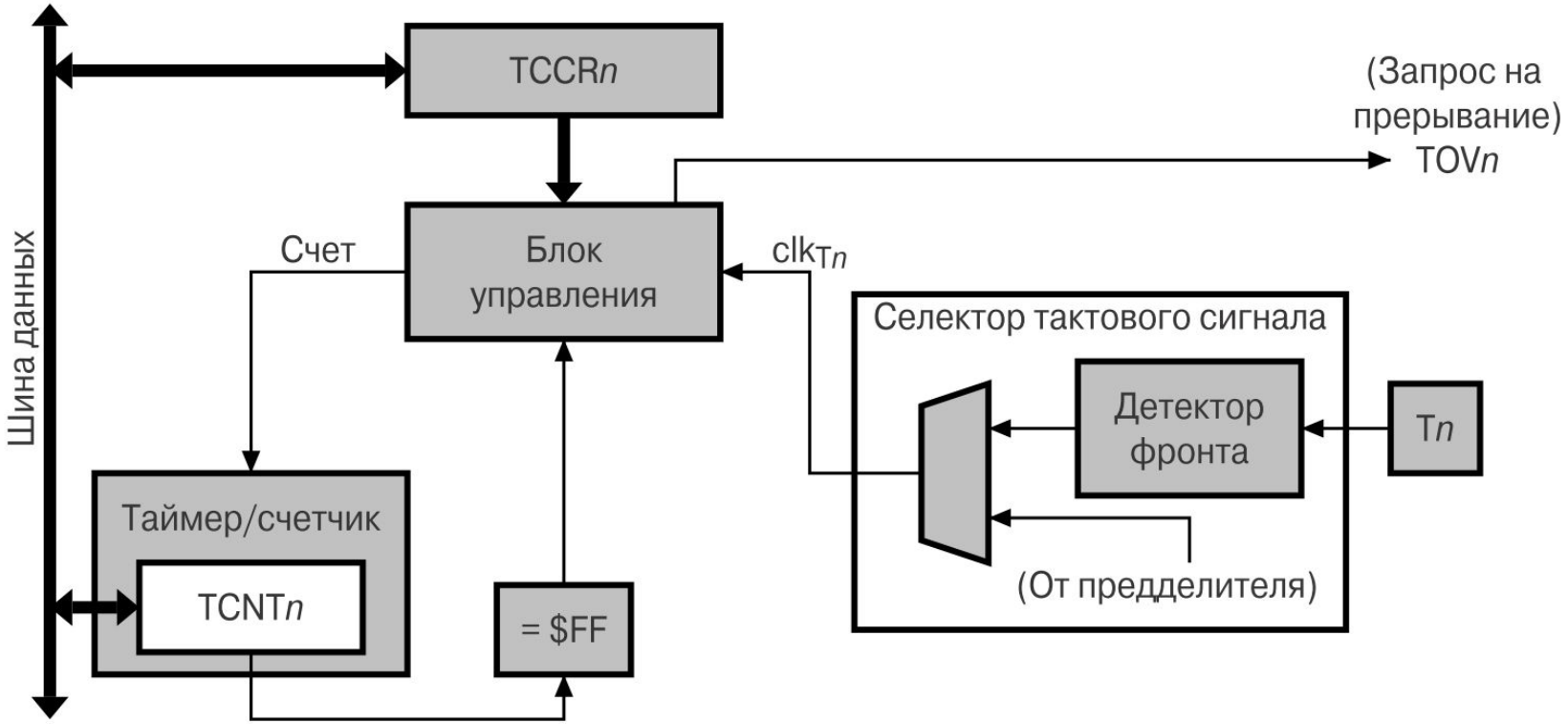
- Счетный таймер тикает не от 0 до предела, а от 0 до регистра сравнения! А после чего сбрасывается.
- В результате, на выходе получаются импульсы всегда одинаковой скважности, но разной частоты.



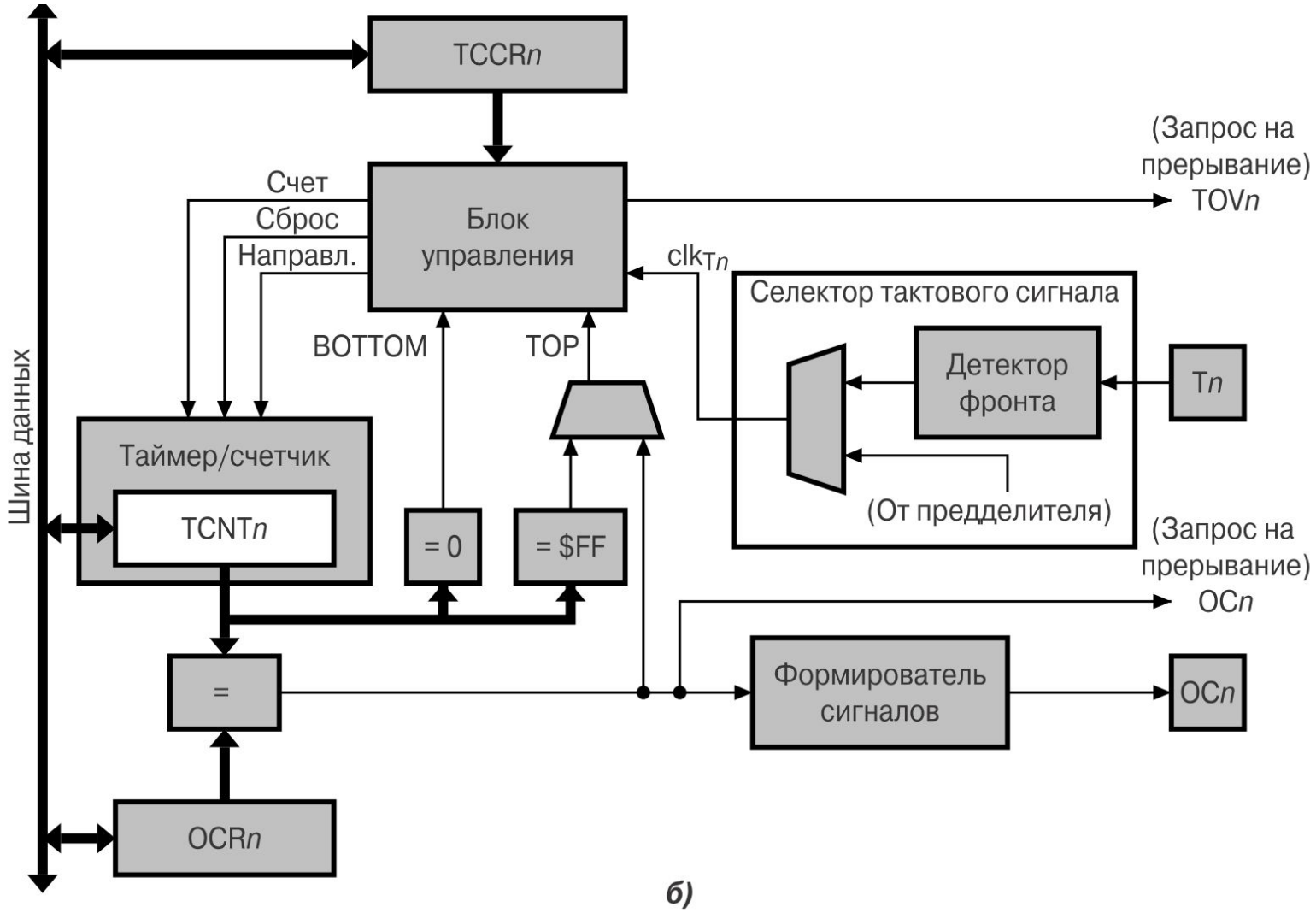
Таймер/счетчик TC0

- Восьмиразрядный таймер/счетчик T0 присутствует во всех моделях микроконтроллеров семейства Mega.
- TC0 может тактироваться внутренне синхронно или внешне асинхронно.
- Реализовано три исполнения восьмиразрядных таймеров/счетчиков:

Таймер/счетчик T0

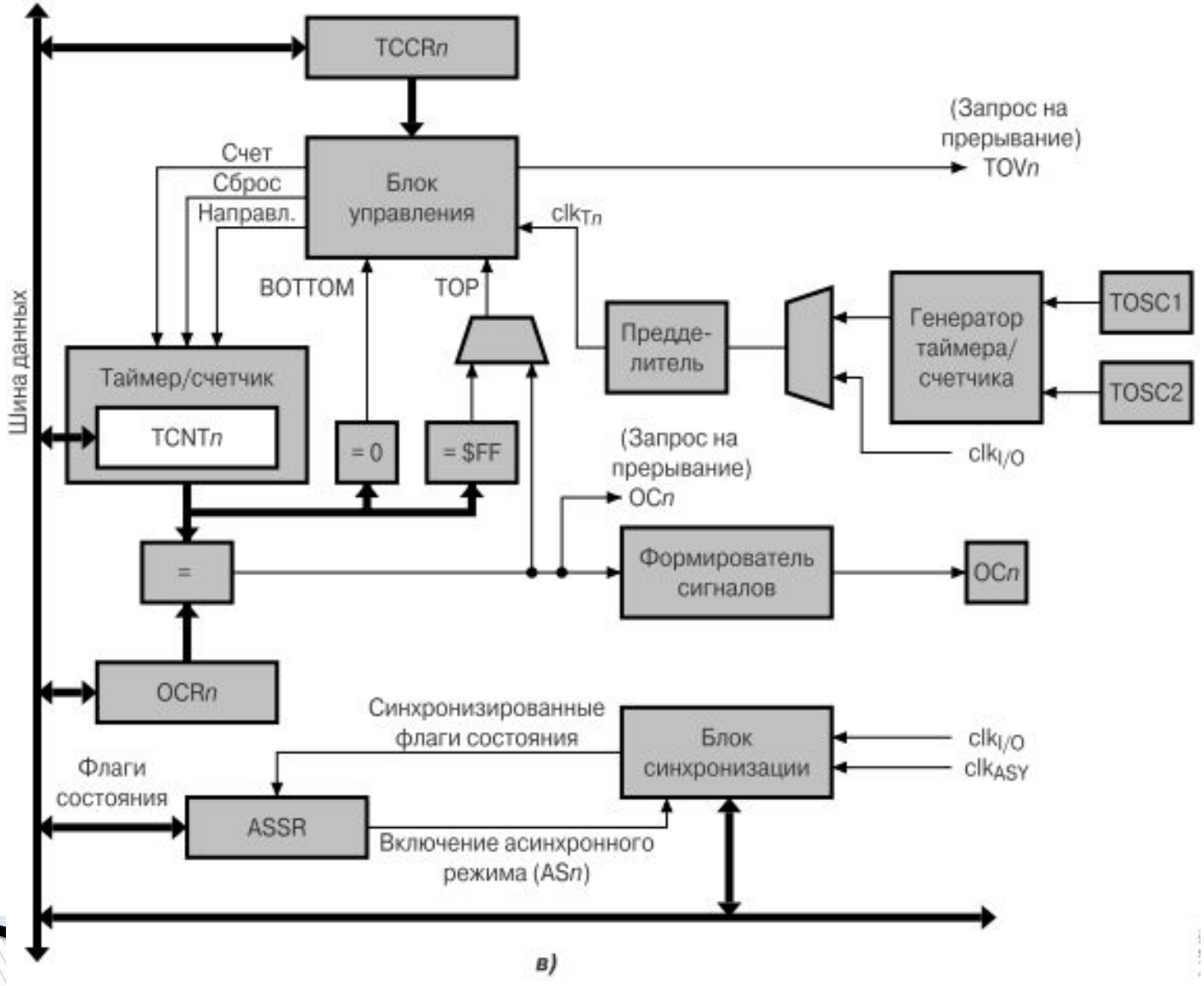


Таймер/счетчик T0



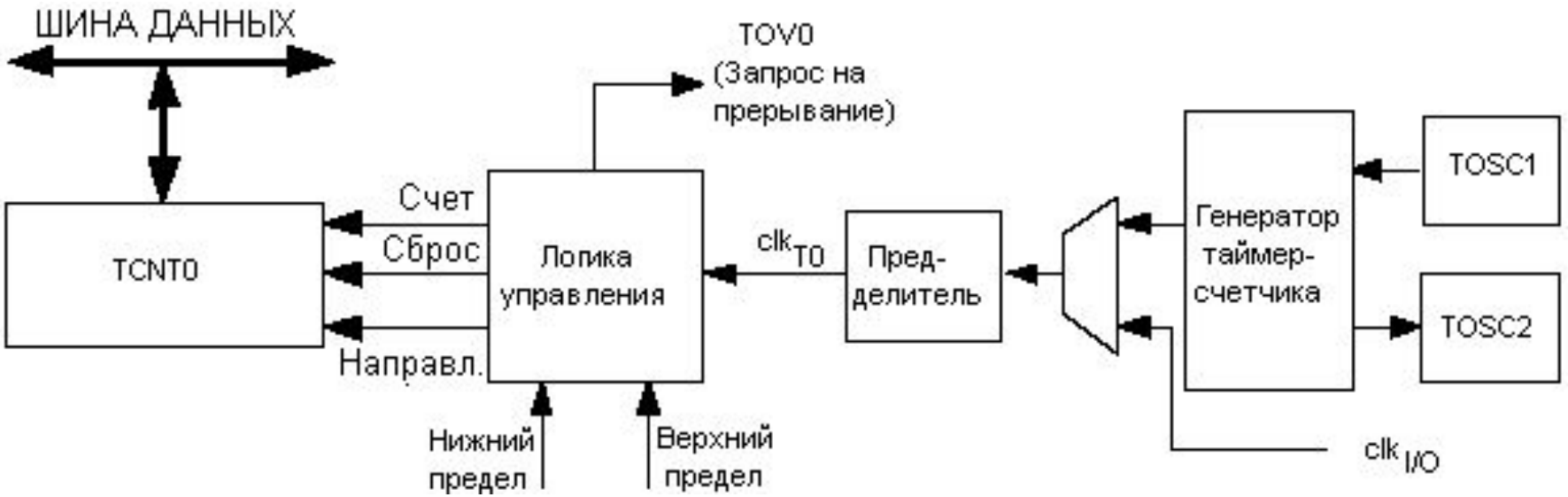
б)

Таймер/счетчик T0



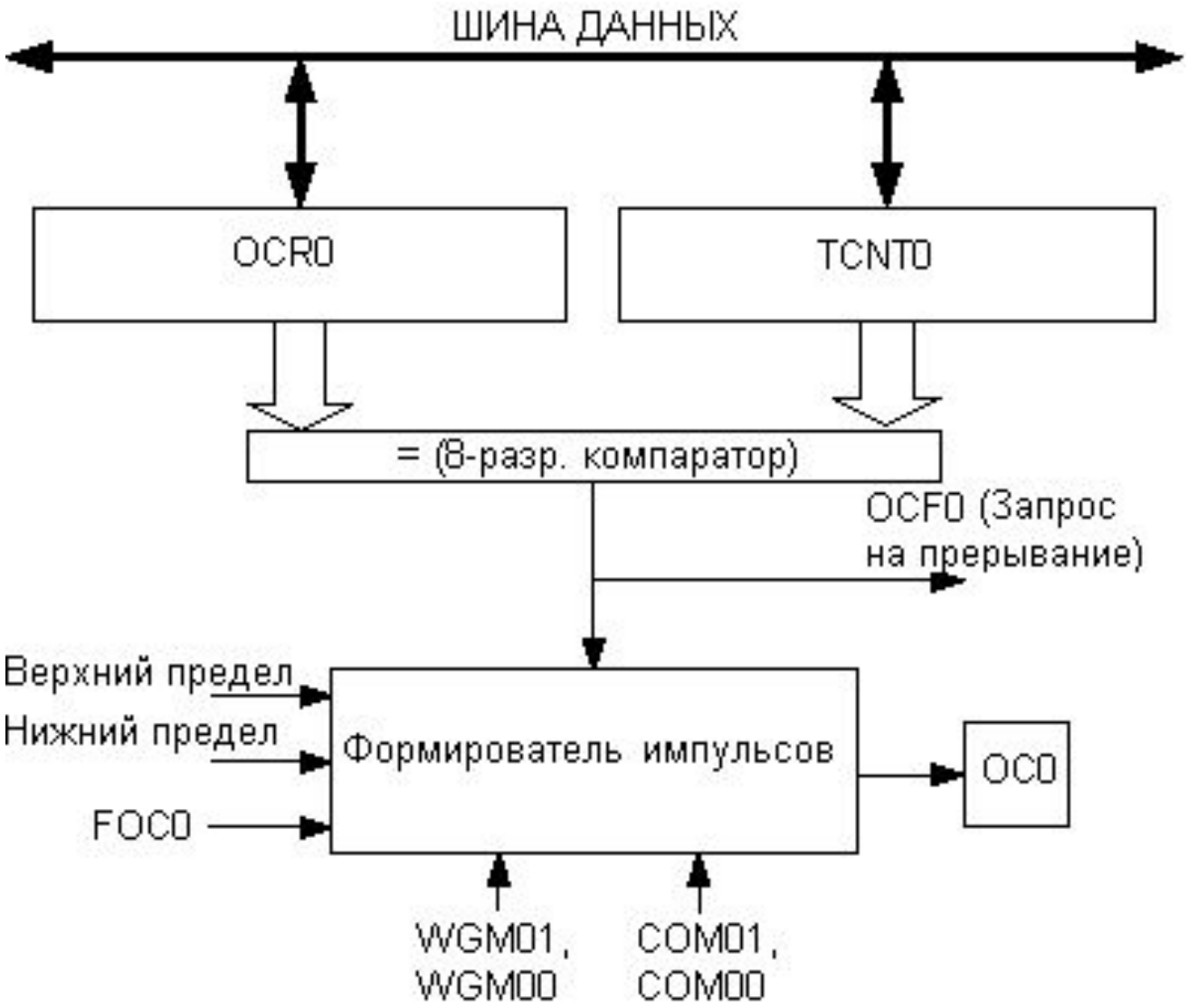
Таймер/счетчик T0

Блок счетчика



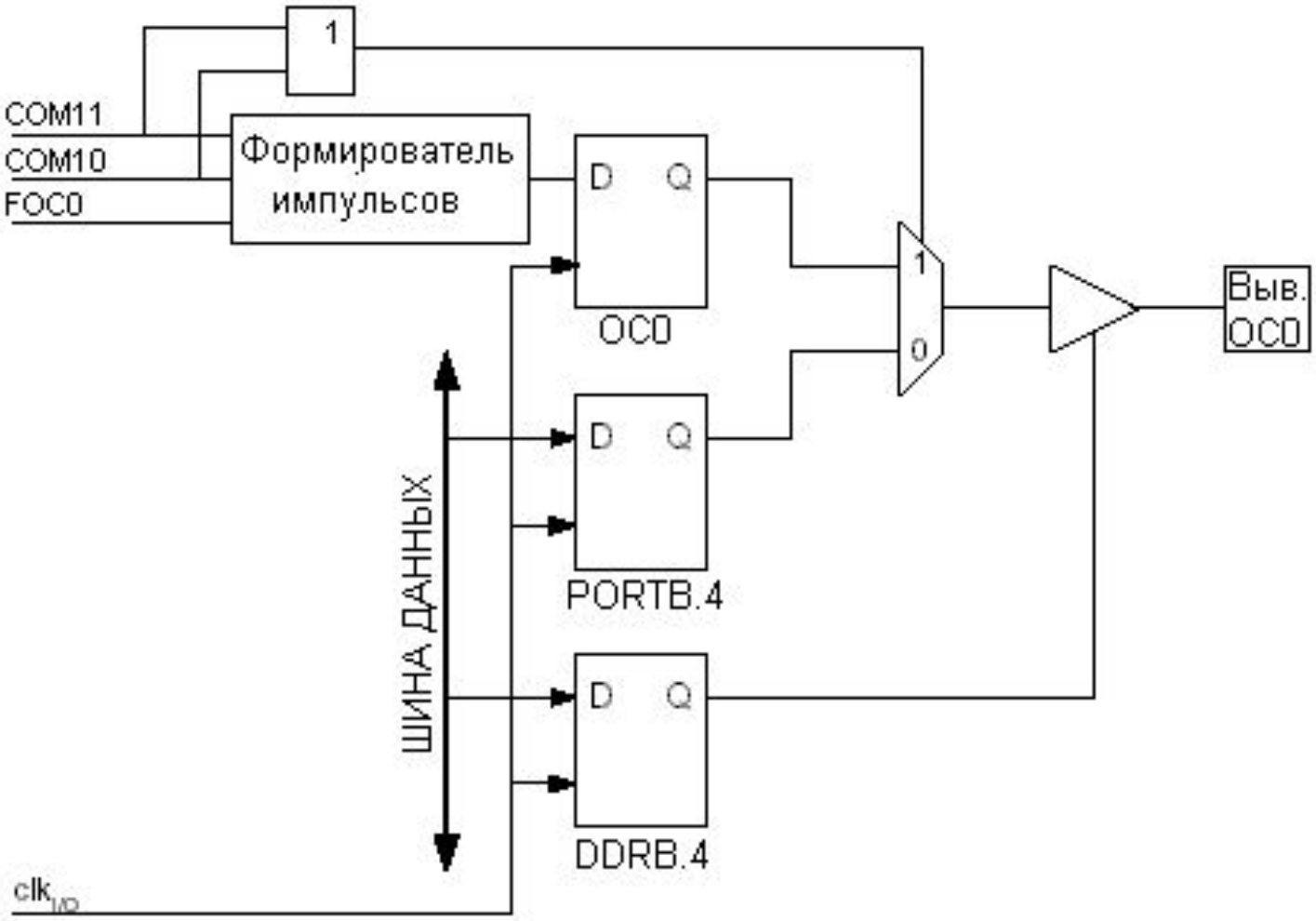
Таймер/счетчик T0

Блок сравнения



Таймер/счетчик T0

Блок формирования выходного сигнала



Пример 1. Таймер T0 в режиме работы формирователя временных интервалов.

- Создадим программу подсчета количества прерываний таймера T0.

```
#include <avr/io.h>
#include <avr/interrupt.h>

int i=0; // объявление глобальной переменной i

void port_init(void)
{  PORTD = 0x00; // порт D работает на выход
   DDRD = 0xFF;
}

void timer0_init(void) // инициализация таймера T0
{  TCCR0 = 0x00; //остановка счетчика
   TCNT0 = 0x64; //задание начального значения в счетный регистр
   TCCR0 = 0x05; //запуск таймера с параметрами TCCR= 0000 0101
}
```

Пример 1. Таймер Т0 в режиме работы формирователя временных интервалов.

```
#pragma interrupt_handler timer0_ovf_isr:10
void timer0_ovf_isr(void) // Работа счетчика при прерываниях
// по переполнению
{ TCNT0 = 0x64; //установка начального значения счетчика
  i++; //инкремент переменной i
}
void init_devices(void)
{ cli();  // запрет действия прерываний
  port_init();
  timer0_init();
  TIMSK = 0x01; //TIMSK=0000 0001->разрешено прерывание
  //по переполнению таймера Т0
  sei(); // разрешение действия прерываний
}
```

Пример 1. Таймер T0 в режиме работы формирователя временных интервалов.

```
int main(void)
{ init_devices();
  while(1)
  PORTD=i; //или, например: PORTD=i/50;
}
```