## Свежие комментарии

- Narod Stream к записи STM Урок 111. FreeRTOS. Очереди. Часть 2
- imperror к записи STM Урок 111. FreeRTOS. Очереди. Часть 2
- ok_195 к записи PIC Урок 5. Таймеры
- ok_195 к записи PIC Урок 5. Таймеры
- ok_195 к записи PIC Урок 5. Таймеры

## Форум. Последние ответы

- Leonid в Программирование МК AVR
  2 дн., 19 час. назад
- nikolay в Программирование МК STM32
  5 дн., 13 час. назад
- den2313 в Программирование МК STM32
  1 неделя, 1 день назад
- Ortos в Программирование МК AVR
  2 нед., 5 дн. назад
- arkonen в Программирование МК AVR
  2 нед., 6 дн. назад

Март 2018

| Пн | Вт | Ср | Чт | Пт | Сб | Вс |
|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 26 | 27 | 28 | 29 | 30 | 31 | |
| « Фев | | | | | | |

## Архивы

- Март 2018
- Февраль 2018
- Январь 2018
- Декабрь 2017
- Ноябрь 2017
- Октябрь 2017
- Сентябрь 2017
- Август 2017
- Июль 2017
- Июнь 2017
- Май 2017

Главная › Программирование PIC › PIC Урок 9. TIMER2

# PIC Урок 9. TIMER2

Posted on Март 3, 2018 by Narod Stream

Опубликовано в Программирование PIC — Нет комментариев ↓



В уроке 5 и в уроке 8 мы познакомились с двумя таймерами микроконтроллера PIC — TIMER0 и TIMER1 .Последний TIMER1 нам интересен был тем, что он, считая также только вперёд и без посторонних модулей сбрасываясь тоже по переполнению, является уже 16-битным, поэтому считает он уже не до 255, а до 65535, что более удобно и позволяет без лишних плясок с переменными выжидать большие промежутки времени до следующего прерывания.

Вообще у линейки контроллеров PIC, которую мы изучаем, есть три таймера, поэтому последний из таймеров — **TIMER2**, мы также не можем пройти стороной и обязаны его изучить и попробовать в работе.

Таймер TIMER2 является 8-битным, но зато у него есть два делителя, что позволяет также получать немалые периоды между прерываниями. Также в одном из регистров данного таймера — **PR2** — можно явным образом задавать значение периода, что также вносит огромное удобство в программирование значения точного периода, позволяя, в отличие от таймера 1 один раз при инициализации задать период, а не заносить значение в регистр счётчика при каждом прерывании.

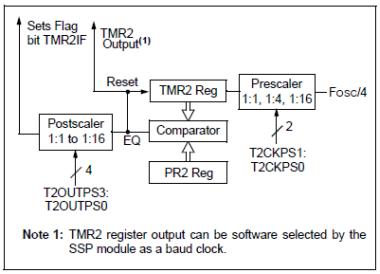
Давайте посмотрим блок-схему таймера 2

## Мета

- Регистрация
- Войти
- RSS записей
- RSS комментариев
- WordPress.org

- Март 2017
- Февраль 2017
- Январь 2017
- Декабрь 2016
- Ноябрь 2016


**FIGURE 7-1: TIMER2 BLOCK DIAGRAM**




**Note 1:** TMR2 register output can be software selected by the SSP module as a baud clock.

Мы видим, что у таймера есть два делителя. Первый — входной делитель или предделитель, который срабатывает сразу после поступления сигнала от внутреннего тактового генератора, который является единственным тактовым генератором, с которым может работать TIMER2. Затем сигнал после предделителя попадает в регистр **TMR2**, являющийся регистром счёта. Как только значение данного регистра сравняется со значением регистра PR2, сигнал выходит уже с компаратора и идёт на выходную ножку и может быть использован для тактирования и управления какими-либо внешними устройствами. Также данный сигнал попадает в выходной делитель или постделитель и после него уже происходит управление флагом прерывания **TMR2IF**. То есть на частоту сигнала, идущего на внешний выход, постделитель не влияет.
Теперь рассмотрим следующий регистр — регистр управления таймером 2 — **T2CON**

**T2CON: TIMER2 CONTROL REGISTER (ADDRESS 12h)**

| U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | TOUTPS3 | TOUTPS2 | TOUTPS1 | TOUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 |
| bit 7 | | | | | | | bit 0 |

Бит **7** данного регистра не используется.
Биты **6-3** — **TOUTPS3:TOUTPS0 (Timer2 Output Postscale Select bits)** — выбор коэффициента деления выходного делителя (постделителя). Значения данных битов следующим образом плавно управляют коэффициентом деления
0000 = 1:1
0001 = 1:2
0010 = 1:3
•••
1111 = 1:16
Следующий бит **2** — **TMR2ON (Timer2 On bit)** — бит включения таймера
**1** — TIMER2 включен
**0** — TIMER2 выключен

Биты **1-0** — **T2CKPS1:T2CKPS0 (Timer2 Clock Prescale Select bits)** — биты управления коэффициентом входного делителя (предделителя). Коэффициент следующим образом зависит от значений битов:

00 = Предделитель 1:1
01 = Предделитель 1:4
1x = Предделитель 1:16

Регистр **TMR2**, являющийся регистром счёта, очищается при сбросе POR, MCLR Reset, WDT Reset или BOR.

Также сигнал от данного регистра, кроме внешней ножки, может



## Рубрики


- 1-WIRE (3)
- ADC (6)
- DAC (4)
- FreeRTOS (2)
- GPIO (26)
- I2C (19)
- SPI (13)
- USART (8)
- Программирование AVR (131)
- Программирование PIC (10)
- Программирование STM32 (222)
- Тесты устройств и аксессуаров (1)

управлять скоростью работы модуля SSP.

Также ещё не стоит забывать одну важную вещь: значение обоих делителей сбрасывается при записи в регистры TMR2 и T2CON и учитывать это при составлении программы.

Схема для реализации нашего кода с прошлого занятия не изменилась. С помощью таймера 2 мы будем управлять значением числа, выводимого на четырёхразрядный светодиодный индикатор. То есть контроллер у нас не меняется.

Проект мы будем использовать также с прошлого занятия **TIMER1.X** и назовём его **TIMER2.X**.

Откроем наш проект в **MPLAB.X**, сделаем его главным, откроем сначала файл **led.c** и удалим из него следующую глобальную переменную

```
static unsigned int LED_Count=0;
```

Также из функции **TIM0_Callback** удалим условие, оставив там лишь вызов функции вывода на индикатор числа (почему-то так лучше работает, видимо, на то, чтобы определить неравенство величин, уходит большое количество времени)

```
void TIM0_Callback(void)
{
  ledprint(TIM1_Count);
  if(n_count==0)
```

Перейдём в файл **main.c** и удалим из функции **main()** всё, что связано с таймером 1

```
T1CKPS0=1; //Prescaler 8
(1000000/31250/8 = 4 Hz)
T1CKPS1=1;
TMR1CS=0; //Internal clock
TMR1L=0xEE; // 65536 - 31250 = 34286
= 0x85EE
TMR1H=0x85;
  …
TMR1IE=1;
TMR1ON=1;
```

Мы попробуем сегодня запрограммировать счётчик, который будет считать медленно — приблизительно раз в секунду. Только для этого мы не сможем использовать только один период таймера с делителями. Поэтому придётся в обработчике прерывания прибегнуть к переменной, которую мы будем там до определённой величины инкрементировать, а затем сбрасывать. Таймер мы запрограммируем так, что он будет отсчитывать периоды с частотой 25 герц, чтобы было целое число герц и чтобы нам потом легче было управлять общей частотой нашего счётчика.

Поэтому значения делителей и регистра PR0 я рассчитал следующим образом

```
PEIE=1;
TOUTPS3=1; //Prescaler Out 10
TOUTPS2=0;
```

```
TOUTPS1=0;
TOUTPS0=1;
T2CKPS0=1;//Postcaler In 16
T2CKPS1=1;
PR2=0xF9; // 249 - 1000000/10/16/250
= 25Hz
```

Затем включим прерывания нашего таймера и собственно таймер

```
PR2=0xF9; // 249 - 1000000/10/16/250
= 25Hz
TMR2IE=1;
TMR2ON=1;
```

Добавим глобальную переменную для счётчика прерываний

```
unsigned char ncnt=0;
unsigned int TIM1_Count=0;
```

Далее мы перейдём в функцию-обработчик прерываний **interrupt isr** и подправим там условие прерывания от таймера 2, так как там было условие прерывания от таймера 1

```
else if(TMR2IE&&TMR2IF)
```

В теле данного условия сначала удалим следующий код

```
~~TMR1L=0xEE;~~
~~TMR1H=0x85;~~
```

Исправим сброс прерывания на нужный таймер

```
TMR2IF=0;
```

Имя переменной для счётчика **TIM1_Count** трогать не будем, так как от этого ничего не зависит.

Добавим код инкрементирования счётчика прерываний

```
TMR2IF=0;
ncnt++;
```

А инкрементирование и сброс счётчика **TIM1_Count** теперь будет в теле следующего условия

```
ncnt++;
if(ncnt>25)
{
  TIM1_Count++;
  if(TIM1_Count>9999) TIM1_Count=0;
}
```

Также в данном теле мы сбросим счётчик прерываний

```
if(ncnt>25)
{
  ncnt=0;
  TIM1_Count++;
```

Ну, вроде бы, и всё.

Соберём код, прошьём контроллер и посмотрим результат нашей работы

Наш счётчик прекрасно считает!

Таким образом, сегодня мы освоили ещё один таймер — TIMER2, воспользовались его делителями и регистром сравнения, чтобы запрограммировать счётчик секунд. Данный счётчик очень приблизительный, так как некоторое время уходит на некоторые операции, но тем не менее это позволяет нам выдерживать хоть и приблизительные, но немалые интервалы.

Всем спасибо за внимание!




**Исходный код**

Купить программатор (неоригинальный) можно здесь: **PICKit3**

Купить программатор (оригинальный) можно здесь: **PICKit3 original**

Отладочную плату PIC Open18F4520-16F877A можно приобрести здесь: **PIC Open18F4520-16F877A**

**Семисегментный чертырехразрядный индикатор красный с общим анодом 10 шт**

**Смотреть ВИДЕОУРОК (нажмите на картинку)**




‹ STM Урок 110. FreeRTOS. Приоритеты задач
STM Урок 111. FreeRTOS. Очередь. Часть 1 ›

## Добавить комментарий

Ваш e-mail не будет опубликован.
Обязательные поля помечены *
**Комментарий**

**Имя** *

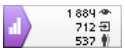
**E-mail** *

**Сайт**

**три + два =**

**Отправить комментарий**








**Наверх**