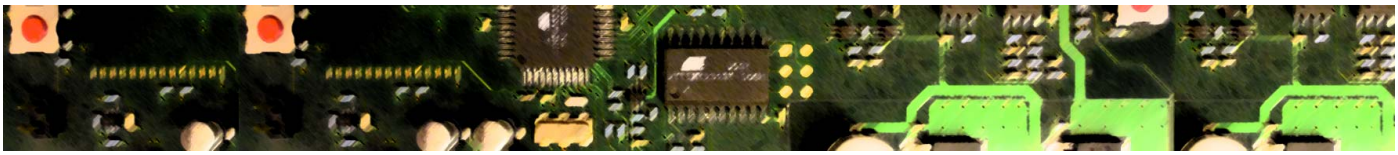

(/)

Разделы

Новости (/index.php/novosti.html)
Встраиваемые системы (/index.php/embedded-programming.html)
Программирование AVR (/index.php/programming-avr.html)
Программирование ARM (/index.php/programmirovanie-arm.html)
Инструменты/технологии (/index.php/instruments-technologies.html)
Как подключить (/index.php/how-connection.html)
Компоненты (/index.php/electronic-components.html)
RTOS (/index.php/rtos.html)
Софт (/index.php/iar-embedded-workbench.html)
Проекты (/index.php/projects-avr.html)
Ссылки (/index.php/links.html)

1wire (/index.php/programming-avr/itemlist/tag/1wire.html) arm (/index.php/programming-avr/itemlist/tag/arm.html)

avr (/index.php/programming-avr/itemlist/tag/avr.html)

avr программатор (/index.php/programming-avr/itemlist/tag/avr программатор.html) ds18b20 (/index.php/programming-avr/itemlist/tag/ds18b20.html)

eeprom (/index.php/programming-avr/itemlist/tag/eeprom.html) i2c (/index.php/programming-avr/itemlist/tag/i2c.html)

IAR (/index.php/programming-avr/itemlist/tag/IAR.html) lcd (/index.php/programming-avr/itemlist/tag/lcd.html)

tsop (/index.php/programming-avr/itemlist/tag/tsop.html) twi (/index.php/programming-avr/itemlist/tag/twi.html)

алгоритмы (/index.php/programming-avr/itemlist/tag/алгоритмы.html)

библиотеки (/index.php/programming-avr/itemlist/tag/библиотеки.html) датчик (/index.php/programming-avr/itemlist/tag/датчик.html)

драйвер (/index.php/programming-avr/itemlist/tag/драйвер.html) интерфейс (/index.php/programming-avr/itemlist/tag/интерфейс.html)

компоненты (/index.php/programming-avr/itemlist/tag/компоненты.html) макросы (/index.php/programming-avr/itemlist/tag/макросы.html)

оу (/index.php/programming-avr/itemlist/tag/оу.html)

программирование микроконтроллеров (/index.php/programming-avr/itemlist/tag/программирование микроконтроллеров.html)

расчет (/index.php/programming-avr/itemlist/tag/расчет.html)

семисегментный индикатор (/index.php/programming-avr/itemlist/tag/семисегментный индикатор.html) си (/index.php/programming-avr/itemlist/tag/си.html)

событийная система (/index.php/programming-avr/itemlist/tag/событийная система.html)

схемотехника (/index.php/programming-avr/itemlist/tag/схемотехника.html) таймер (/index.php/programming-avr/itemlist/tag/таймер.html)

управление (/index.php/programming-avr/itemlist/tag/управление.html) устройства (/index.php/programming-avr/itemlist/tag/устройства.html)

учебный курс avr (/index.php/programming-avr/itemlist/tag/учебный курс avr.html)

шим (/index.php/programming-avr/itemlist/tag/шим.html)

Учебный курс AVR. Таймер - счетчик T0. Режим Normal. Ч2

03/10/2013 - 15:39 | Павел Бобков

Введение

По сути, таймер микроконтроллера - это цифровой счетчик, только "навороченый". На вход счетчика подается тактовый сигнал, по перепадам которого счетчик увеличивает свое значение. При возникновении событий - переполнение счетчика или совпадение его значения с заданным - генерируется запрос на прерывание.

Давайте разберем, как пользоваться таймером T0 в режиме Normal. В этом режиме таймер считает от какого-то начального значения счетного регистра до максимально возможного (до 255 или 0xFF). Когда таймер T0 досчитывает до максимума, то в следующий такт таймера возникает переполнение счетного регистра TCNT0 - он обнуляется и устанавливается флаг TOV0. Если в программе разрешены прерывания глобально (флаг I регистра SREG) и прерывание таймера T0 по переполнению (флаг TOIE0 регистра TIMSK), то микроконтроллер вызовет соответствующий обработчик. Если значение счетного регистра совпадет с регистром сравнения OCR0, то установится флаг OCF0 и при разрешенном прерывании по событию совпадение, запустится его обработчик.

Таймер T0 в режиме Normal

Рассмотрим практическую задачу - нам нужно каждые 20 мс опрашивать кнопку. Частота микроконтроллера 8 МГц, микроконтроллер ATmega16.

Первое, что нужно сделать - это определиться с выбором коэффициента предделителя таймера и рассчитать начальное значение для счетного регистра TCNT0.

Таймер T0 может тактироваться от внутреннего тактового сигнала микроконтроллера или от внешнего, который подается на вывод T0. При работе от внутреннего тактового сигнала пользователь может выбирать коэффициенты деления частоты этого сигнала. У таймера T0 есть пять возможных вариантов коэффициента предделителя - 1, 8, 64, 256, 1024.

Для решения поставленной задачи, я рассуждаю следующим образом. Если бы один такт таймера T0 имел период 1 мс, то мне бы это подошло. 20 тактов дают 20 мс. Какой коэффициент предделителя таймера позволит получить близкий к 1 мс период тактовой частоты? Можно посчитать.

Тактовая частота микроконтроллера Fcpu = 8000000 Гц
Период тактового сигнала микроконтроллера Tcpu = 1/Fcpu
Период тактового сигнала таймера T0 равен Tt0 = (1/Fcpu)/k = k/Fcpu

При k = 1024 период тактовой частоты таймера T0 будет равен Tt0 = 1024/8000000 = 0.128 мс

Это максимальный период тактового сигнала таймера, который мы можем получить при наших условиях (Fcpu = 8 МГц). При меньших коэффициентах - период получится еще меньше.

Ну хорошо, пусть один такт таймера это 0.128 мс, хватит ли разрядности счетного регистра, чтобы отсчитать этот временной интервал и сколько для этого понадобится тактов? Делим требуемый интервал времени (20 мс) на длительность одного такта таймера и получаем ответ.

n = t/Tto = 20 мс/ 0.128 мс = 156.25

Округлив до целого, получаем 156 тактов. Это меньше 255 (максимального значения счетного регистра), значит разрядности счетного регистра TCNT0 хватит.

Начальное значение для счетного регистра TCNT0 вычисляем как разницу между максимальным числом тактов таймера T0 и требуемым, то есть 256 - 156 = 100. (256 - это максимальное количество временных интервалов, которые может отсчитать любой 8-и разрядный таймер.)

Думаю, теперь понятно, как рассчитывать начальное значение TCNT0 для режима Normal:

- вычисляем период одного такта таймера Tt0 = k/Fcpu,
- вычисляем требуемое количество тактов для заданного интервала n = t/Tto,
- вычисляем начальное значение для счетного регистра TCNT0 = 256 - n.

Можно автоматизировать эту процедуру с помощью макросов. Например, так:

```
#define F_CPU 8000000UL
#define TIME_MS(time, k) (256L - ((time)*(F_CPU))/(1000L*(k)))
```

Но с таким макросом нужно быть начеку, при определенных значениях time и k могут возникать ошибки.

Теперь переходим к коду. **Чтобы использовать таймер T0 (да и любой другой тоже), его нужно настроить (инициализировать) и описать обработчик прерывания (если они используются).**

Инициализация таймера состоит из следующих шагов:

- остановка таймера,
- задание режима Normal в TCCR0 без старта,
- установка начального значения TCNT0,
- сброс флагов в регистре TIFR,
- разрешение прерывания по переполнению в TIMSK,
- установка предделителя в TCCR0, то есть старт таймера

В данной последовательности возможны вариации.

Для нашей задачи код инициализации будет выглядеть так:

```
/*значение для счетного регистра*/
#define T_POLL 100

....

 TCCR0 = 0;
 TCCR0 = (0<<WGM01)|(0<<WGM00);
 TCNT0 = T_POLL;
 TIFR = (1<<TOV0);
 TIMSK |= (1<<TOIE0);
 TCCR0 |= (1<<CS02)|(0<<CS01)|(1<<CS00);
```

Вторая строчка инициализации, по сути, бесполезна, она добавлена для наглядности. Чтобы четко видеть, какой режим таймера устанавливается.

Сброс флагов прерываний в регистре TIFR выполняется записью 1 в соответствующий разряд. Эту операцию нужно выполнять именно перезаписью регистра, а не с помощью побитового ИЛИ. И вот почему.

Допустим, в регистре TIFR устанавлены два флага прерывания - TOV1 и TOV0. TOV0 нам нужно сбросить. При установке требуемого разряда с помощью ИЛИ происходит примерно следующая вещь.

```
//TIFR имеет значение 0b00000101
//установлены флаги TOV1 и TOV0
//выполняется код TIFR |= (1<<TOV0);

//TIFR копируется в R16
IN R16, 0x38

//в R16 устанавливается разряд TOV0
//хотя он и так уже установлен
ORI R16, 0x02

//R16, равный 0b00000101, записывается в регистр TIFR
OUT 0x38, R16
```

В результате сброшены оба флага, а мы хотели сбросить один.

Продолжаем.

Синтаксис описания обработчиков прерывания у разных компиляторов немного отличается. Для IAR`а обработчик прерывания таймера T0 по событию переполнение будет выглядеть так:

```
#pragma vector = TIMER0_OVF_vect
__interrupt void TimerT0Ovf(void)
{
   /*перезапись счетного регистра*/
   TCNT0 = T_POLL;

   /*здесь должен быть опрос кнопки*/

}
```

TIMER0_OVF_vect - это адрес вектора прерывания по событию переполнение. Он берется из заголовочных файлов на микроконтроллер. В данном случае я взял его из файла iom16.h.

Первая строка обработчика (TCNT0 = T_POLL;) выполняет перезапись счетного регистра, то устанавливает его начальное значение. Если этого не сделать, таймер продолжит счет с 0. Перезапись счетного регистра нужно выполнять в начале обработчика прерывания.

Весь код для нашей задачи будет выглядеть примерно так. (Код приведен для IAR`а. Для других компиляторов нужно изменить заголовочные файлы и обработчик прерывания.)

```
#include <ioavr.h>
#include <stdint.h>
#include <intrinsics.h>

#define T_POLL 100

int main( void )
{
   /*инициализация таймера*/

   TCCR0 = 0;
   TCCR0 = (0<<WGM01)|(0<<WGM00);
   TCNT0 = T_POLL;
   TIFR |= (1<<TOV0);
   TIMSK |= (1<<TOIE0);
   TCCR0 |= (1<<CS02)|(0<<CS01)|(1<<CS00);

   /*инициализация остальной периферии*/
   DDRB |= (1<<PB0);

   __enable_interrupt();
   while(1);

   return 0;
}

/*обработчик прерывания T0
по событию переполнение*/
#pragma vector = TIMER0_OVF_vect
__interrupt void TimerT0Ovf(void)
{
   /*перезапись счетного регистра*/
   TCNT0 = T_POLL;

   /*опрос кнопки*/

   /*инверсия PB0 для отладки*/
   PORTB ^= (1<<PB0);

}
```

Управление выводом OC0

В режиме Normal таймер T0 может изменять состояние вывода OC0 при совпадении счетного регистра и регистра сравнения. Причем даже без прерываний. Варианты управления определяются разрядами COM01 и COM00 регистра TCCR0.

COM01	COM00	Поведение вывода OC0
0	0	Отключен от счетчика T0
0	1	Состояние вывода меняется на противоположное
1	0	Вывод сбрасывается в 0
1	1	Вывод устанавливается в 1

ChipEnable.ru

Вот пример программы, генерирующей прямоугольный сигнала на выводе OC0.

```
#include <ioavr.h>
#include <intrinsics.h>

int main( void )
{
   /*инициализация таймера T0*/

   TCCR0 = 0;
   TCCR0 = (0<<COM01)|(1<<COM00)|(0<<WGM01)|(0<<WGM00);
   TCNT0 = 0;
   OCR0 = 0;
   TIMSK = 0;
   TCCR0 |= (1<<CS02)|(0<<CS01)|(1<<CS00);

   /*инициализация OC0*/
   DDRB |= (1<<PB3);

   while(1);
   return 0;
}
```

Вывод OC0 будет менять свое состояние на противоположное при нулевом значении счетного регистра.

Несколько моментов относительно использования таймера

Обработчик прерывания таймера (да и любой другой периферии) нужно делать как можно короче.

Если расчетное значение для счетного регистра (или регистра сравнения) округляется, то временной интервал будет отсчитываться таймером с погрешностью.

Если нужно отсчитать большой временной интервал, разрядности таймера может не хватить. В этом случае нужно использовать или 16 разрядный таймер, или программный.

И последнее. Может случится ситуация, что обработка прерывания таймера задержится (например, по вине другого обработчика) и регистр TCNT0 уже посчитает несколько тактов. Если просто перезаписать значение TCNT0, то следующее прерывание вызовется позже, чем нужно. Получится, что предыдущее (задержанное) и новое прерывания не выдержат требуемый интервал.

Эту ситуацию можно сгладить, если выполнять перезапись счетного регистра вот так:

TCNT0 = TCNT0 + startValue;

Сложение текущего значения счетного регистра с инициализируемым, учтет эти лишние такты. Правда есть одно НО! При больших значения startValue операция сложения может вызвать переполнение счетного регистра.

Например, startValue = 250, а таймер успел досчитать до 10. Тогда операция сложения приведет к такому результату:

10 + 250 = 260

Берем 8 разрядов от 260 получаем 4. В TCNT0 запишется 4.

Ссылки

Учебный курс AVR. Таймер - счетчик T0. Регистры. Ч1

(/index.php/programming-avr/item/171-avr-timer-t0-ch1.html)

Твитнуть Нравится 0

Tagged under #avr (/index.php/programming-avr/itemlist/tag/avr.html) #учебный курс avr (/index.php/programming-avr/itemlist/tag/учебный%20курс%20avr.html) #таймер (/index.php/programming-avr/itemlist/tag/таймер.html)

Related items

- Библиотека для опроса кнопок (/index.php/programming-avr/item/218-biblioteka-dlya-oprosa-knopok.html)
- Работа с SD картой. Воспроизведение wav файла. Ч3 (/index.php/programming-avr/item/212-rabota-s-sd-kartoy-vosproizvedenie-wav-fayla-ch3.html)
- Работа с SD картой. Подключение к микроконтроллеру. Ч1 (/index.php/programming-avr/item/209-rabota-s-sd-kartoy-podklyuchenie-k-mikrokontrolleru-ch1.html)
- AVR315: Использование TWI модуля в качестве ведущего I2C устройства (/index.php/programming-avr/item/208-avr315-ispolzovanie-twi-modulya-v-kachestve-veduschego-i2c-ustroystva.html)
- Учебный курс AVR. Использования TWI модуля как ведущего I2C устройства. Работа на прерываниях. Ч5 (/index.php/programming-avr/item/201-uchebnyy-kurs-avr-ispolzovaniya-twi-modulya-kak-veduschego-i2c-ustroystva-rabota-na-preryvaniyah-ch5.html)

Комментарии (/index.php/component/jcomments/feed/com_k2/187.html)

# (/index.php/programming-avr/item/187-uchebnyy-kurs-avr-taymer-schetchik-t0-rezhim-normal-ch2.html#comment-3319) **Peter** 04.10.2013 17:49
TIFR |= (1 < < TOV0); // Danger! RMW!

Ответить | Ответить с цитатой | Цитировать

(/index.php/programming-avr/item/187-uchebnyy-kurs-avr-taymer-schetchik-t0-rezhim-normal-ch2.html#comment-3320) **Pashgan** 05.10.2013 23:50
Лаконично.

Ответить | Ответить с цитатой | Цитировать

(/index.php/programming-avr/item/187-uchebnyy-kurs-avr-taymer-schetchik-t0-rezhim-normal-ch2.html#comment-3321) **Peter** 06.10.2013 06:43
Да потому что много раз говорилось об опасности применения операции Чтение-Модификация-Запись для регистра TIFR.

Вместо операции |= применяйте =

Ответить | Ответить с цитатой | Цитировать

(/index.php/programming-avr/item/187-uchebnyy-kurs-avr-taymer-schetchik-t0-rezhim-normal-ch2.html#comment-3322) **Pashgan** 06.10.2013 19:16
Исправил. Дополнил.

Ответить | Ответить с цитатой | Цитировать

(/index.php/programming-avr/item/187-uchebnyy-kurs-avr-taymer-schetchik-t0-rezhim-normal-ch2.html#comment-3801) **Рогалик** 08.02.2014 17:20
"256 - это максимальное количество временных интервалов". Подскажите, правильно ли я понял, что 225-ый это максимальный счётный интервал, а 256-ой, это уже интервал времени в течении которого производится сброс счётчика?

Ответить | Ответить с цитатой | Цитировать

(/index.php/programming-avr/item/187-uchebnyy-kurs-avr-taymer-schetchik-t0-rezhim-normal-ch2.html#comment-3807) **Pashgan** 11.02.2014 17:49
256-й интервал - это интервал между 255 отсчетом и 0. Сброс происходит в конце 256 интервала.

Ответить | Ответить с цитатой | Цитировать

(/index.php/programming-avr/item/187-uchebnyy-kurs-avr-taymer-schetchik-t0-rezhim-normal-ch2.html#comment-4797) **Артем Колесников** 06.05.2016 15:17
Можно добавить дополнительное пояснение по поводу сброса флагов, почему нужно записывать регистр целиком, а не операцией |=. Потому что для обнуления битов таких регистров процессор выполняет операцию побитового XOR, поэтому чтобы обнулить нужный бит нужно "записать" в нужный бит 1, если в нем стоит 1, то в итоге получится 0 (сброс). Во всех остальных, в которых стоит 1, в результате XOR с соответствующим нулем получится 1 - сохранение бита.

Ответить | Ответить с цитатой | Цитировать

(/index.php/programming-avr/item/187-uchebnyy-kurs-avr-taymer-schetchik-t0-rezhim-normal-ch2.html#comment-4926) **Антон_Ш** 20.10.2016 11:06
Внимание! кто пользуется Proteus.
Имеется ошибка при симуляции режима Normal.
Заключается в том что после запуска таймера не происходит прерывание по совпадению. Оно произойдет только после первого переполнения таймера.
Запуск ---------OCR0 = 50 (нет прерывания) ---------TCNT0 = 256 (прерывание перепол. есть) --------OCR0 = 50 (прерывание произойдет) ....

Ответить | Ответить с цитатой | Цитировать

Обновить список комментариев

RSS лента комментариев этой записи (/index.php/component/jcomments/feed/com_k2/187.html)

Добавить комментарий

Имя (обязательное)

E-Mail

Обновить

Отправить