Использование асинхронного режима таймера 2. "Точные часы"

Цель: Изучить работу Таймера2 в асинхронном режиме для микроконтроллера ATmega8.

Все таймеры контроллера тактируются от основного тактового генератора, исходя из битов конфигурации контроллера это может быть внутренний RC генератор, внешний кварцевый резонатор или сигнал внешнего генератора. Также регистром **TCNTn** может управлять сигнал идущий со счетного входа **Tn**. В микроконтроллере **Atmega8** есть один таймер, который имеет свой собственный генератор - это Таймер/счетчик 2, т.е. таймер может работать асинхронно от всей остальной периферии. Чтобы задействовать генератор T2 необходимо подключить к выводам **TOSC2** и **TOSC1** кварцевый резонатор на рекомендуемую частоту **32768 Гц**. Корпус этого кварца необходимо заземлить. Обычно асинхронный режим применяют для постройки часов реального времени, т.к. T2 будет отдельно работать от основного ядра контроллера это значительно снижает энергопотребление процессора. Обслуживанием асинхронного режима работы T2 занимается регистр **ASSR**, ниже показано назначение его битов:

Bit	7	6	5	4	3	2	1	0
	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB
Read/Write	R	R	R	R	R/W	R	R	R
Initial Value	0	0	0	0	0	0	0	0

AS2: Разрешение асинхронного тактирования T2
Если AS2 = 0, то T2 тактируется сигналом синхронизации ввода-вывода - clkI/O. Если AS2 = 1, то T2 тактируется низкочастотным кварцевым генератором, связанного с задающим кварцем через выводы TOSC1 и TOSC2. При изменении значения AS2 содержимое регистров TCNT2, OCR2 и TCCR2 может быть нарушено.

TCN2UB: Флаг занятости T2 при обновлении
Если T2 работает асинхронно и выполнена запись в TCNT2, то данный флаг устанавливается. После того, как содержимое TCNT2 обновляется из временного регистра, данный флаг сбрасывается аппаратно. Следовательно, когда TCN2UB=0, в TCNT2 может быть выполнена запись нового значения.

OCR2UB: Флаг занятости регистра порога сравнения при обновлении
Если T2 работает асинхронно и выполнена запись в регистр OCR2, то данный флаг устанавливается. По завершении обновления OCR2 из временного

регистра данный флаг сбрасывается аппаратно. Если OCR2UB=0, то это означает готовность регистра OCR2 к записи нового значения.

TCR2UB: Флаг занятости регистра управления T2 при обновлении Если T2 работает асинхронно и выполнена запись в регистр TCCR2, то данный флаг устанавливается. По завершении обновления TCCR2 из временного регистра данный флаг сбрасывается аппаратно. Если TCR2UB=0, то это означает готовность регистра TCCR2 к записи нового значения.

Из описания на контроллер Atmega8 можно узнать некоторые особенности по работе с данным режимом.

Если выполнить запись в любой из трех регистров T2, когда соответствующий флаг занятости установлен, то обновляемое значение может быть нарушено и может стать причиной несанкционированного возникновения прерывания.

При переключении между асинхронным и синхронным тактовыми источниками T2 содержимое регистров TCNT2, OCR2 и TCCR2 может быть нарушено. Во избежание этого необходимо придерживаться следующей безопасной последовательности переключения:

1. Отключить прерывания T2 путем сброса бит OCIE2 и TOIE2.
2. Выбрать необходимый тактовый источник с помощью бита AS2
3. Выполнить запись новых значений в TCNT2, OCR2 и TCCR2.
4. При переходе в асинхронный режим тактирования дождаться сброса флагов TCN2UB, OCR2UB и TCR2UB.
5. Сбросить флаги прерывания T2
6. При необходимости разрешить прерывания

Генератор оптимизирован под использование часового кварцевого резонатора на частоту 32768 Гц. Подключение внешнего тактового сигнала к выводу TOSC1 может сказаться на некорректности работы таймера. Тактовая частота ЦПУ должна быть минимум в четыре раза выше частоты данного генератора. Запись в любой из регистров TCNT2, OCR2 или TCCR2 происходит за два положительных фронта TOSC1, т.к. данные предварительно помещаются во временный регистр, а затем передаются по назначению. Необходимо предусмотреть, чтобы до окончания передачи содержимого временного регистра к назначенному регистру не выполнилась еще одна запись в этот регистр. Каждый из трех упомянутых регистров имеют свои индивидуальные временные регистры. Это означает, что, например, запись в TCNT2 не влияет на процесс записи в регистр OCR2. Чтобы определить в какой регистр была выполнена запись, реализован регистр асинхронного состояния ASSR.

Если экономичный режим или расширенный дежурный режим вводится после записи в TCNT2, OCR2 или TCCR2, то необходимо дождаться завершения обновления записанного регистра, в случае если T2 используется для пробуждения из этих режимов. Иначе микроконтроллер перейдет в режим сна прежде чем вступят в силу желаемые изменения. Это особенно важно, если прерывание по результату сравнения T2 используется для пробуждения микроконтроллера, т.к. функция отработки условия совпадения блокируется после записи в OCR2 или TCNT2. Если цикл записи не заканчивается и микроконтроллер переводится в режим сна прежде чем OCR2UB станет равным нулю, то микроконтроллер больше не будет прерываться при выполнении условия сравнения и, следовательно, не сможет пробудиться.

Если T2 используется для пробуждения микроконтроллера из экономичного режима или расширенного дежурного режима, то, если требуется перевести данный микроконтроллер снова в один из этих режимов, необходимо учесть несколько особенностей. Для сброса логики прерываний требуется один такт TOSC1. Если интервал времени между пробуждением микроконтроллера и повторным вводом режима сна меньше чем один период TOSC1, то прерывание в дальнейшем не возникнет и микроконтроллер не сможет пробудиться. Если нет уверенности в прохождении достаточного времени перед повторным вводом в экономичный режим или расширенный дежурный режим, то необходимо придерживаться следующей последовательности действий, которая гарантирует прохождение одного периода TOSC1:

1.Запись значения в TCCR2, TCNT2 или OCR2.

2.Ожидание сброса соответствующего флага занятости при обновлении в регистре ASSR.

3.Ввод экономичного или расширенного дежурного режима.

Если выбрана асинхронная работа, то генератор на 32768 Гц T2 находится постоянно включенным, за исключением режима выключения и дежурного режима микроконтроллера. После сброса при подаче питания или пробуждения из режима выключения и дежурного режима программист должен учитывать, что для возобновления нормальной стабильной работы данного генератора требуется минимум 1 секунда. Таким образом, рекомендуется включить задержку минимум на 1 сек. перед использованием T2 после подачи питания или выхода из режима выключения или дежурного режима. Содержимое всех регистров T2 необходимо рассматривать как потерянное после подачи питания или пробуждения из указанных выше режимов из-за нестабильности тактового сигнала при запуске независимо от

того, какой асинхронный источник используется (генератор или внешний сигнал на выводе TOSC1).

Выход микроконтроллера из экономичного и расширенного дежурного режимов при асинхронном тактирования Т2 происходит в следующей последовательности. Если выполняется условие прерывания, то инициируется процесс пробуждения следующим тактом синхронизации таймера, т.е. таймер минимум на 1 изменит свое состояние, прежде чем процессор получит доступ к его состоянию. После пробуждения микроконтроллер задерживается на 4 такта синхронизации ЦПУ, а затем переходит на вектор обработки прерывания, а по завершении процедуры его обработки возвращается к выполнению инструкции, следующей за SLEEP.

Чтение регистра TCNT2 сразу после пробуждения из экономичного режима (Power-save) может дать некорректный результат. Поскольку TCNT2 тактируется от асинхронного источника TOSC, то чтение TCNT2 должно быть выполнено через регистр, синхронизированный с внутренней синхронизацией ввода-вывода. Синхронизация выполняется каждый нарастающий фронт на TOSC1. При пробуждении из экономичного режима активизируется снова синхронизация ввода-вывода (clkI/O) и при чтении TCNT2 фактически будет считываться предыдущее значение (которое записано перед вводом в режим сна) до следующего нарастающего фронта на TOSC1. Фаза тактового сигнала TOSC после выхода из экономичного режима непредсказуема, т.к. зависит от момента пробуждения микроконтроллера. С учетом этого, при чтении содержимого TCNT2, рекомендуется соблюдать следующую последовательность действий:

1.По усмотрению записать произвольное значение или в регистр OCR2 или в TCCR2.

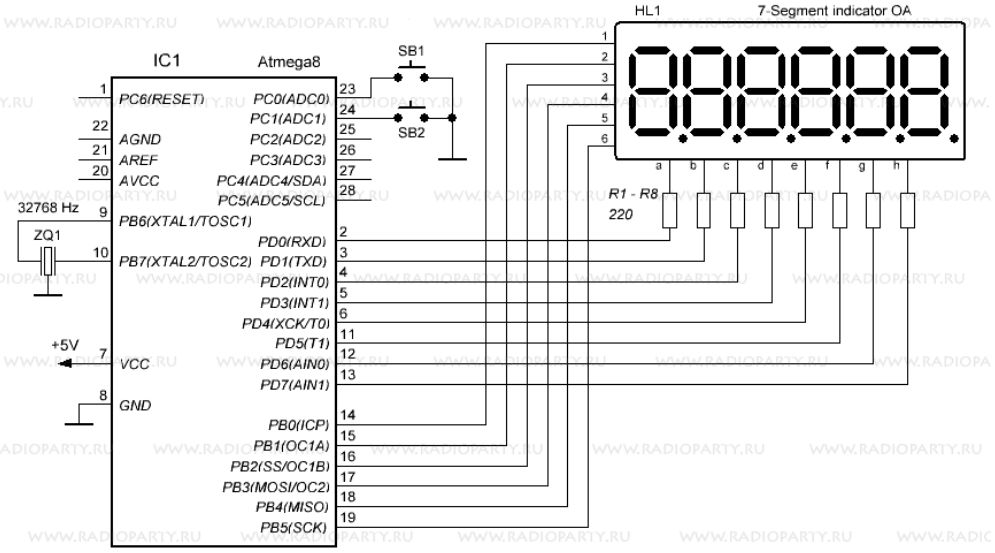
2.Дождаться сброса флага занятости при обновлении, соответствующего выбранному в п.1 регистру.

3.Выполнить чтение TCNT2.

Во время асинхронной работы синхронизация флагов прерываний асинхронного таймера требует три такта синхронизации ЦПУ плюс один такт синхронизации таймера. Таким образом, как минимум таймер должен изменить свое состояние на 1 прежде чем процессор сможет считать его содержимое, вызывающее установку флагов прерываний. Выход генератора прямоугольных импульсов OC2 привязан к синхронизации таймера и не синхронизирован с тактированием ЦПУ.

В пример практической реализации данного режима таймера доработаем часы, программа для которых была написана на одном из предыдущих занятий. Индикатор будет шестиразрядным, т.к. добавится отображение

секунд. Для этого я доработал библиотеку для работы с шестиразрядным индикатором. Микроконтроллер работает от внутреннего генератора частотой 8МГц. Биты TCN2UB, OCR2UB, TCR2UB не используем т.к. нет



необходимости. Кнопками SB1 и SB2 прибавляются значения часов и минут соответственно. При нажатии на SB1, SB2 одновременно меняется яркость индикатора.

Как было написано выше для активации асинхронного режима таймера выставляем бит AS2 в единицу, и подключаем кварц на 32768Гц. В обработчике прерывания по переполнению T2, оно вызывается раз в секунду, инкрементируем значения секунд, потом минут и часов соответственно и выводим эти значения на индикатор. Ниже код программы с подробными комментариями:

```
// Использование асинхронного режима таймера 2. "Точные часы"
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include "display.h" // Подключаем библиотеку для семисегментных индикаторов

volatile unsigned char second, minute, hour, level;

// Прерывание по переполнению T2
ISR(TIMER2_OVF_vect)
{
if(second++ >= 59)
{
```

```
second = 0;
minute++;
}
if(minute > 59)
{
minute = 0;
hour++;
}
if(hour > 23)
hour = 0;

set_time(hour,minute,second); // Выводим данные на дисплей
}

int main(void)
{
DDRC |= (0 << PC1)|(0 << PC0); // Кнопки
PORTC |= (1 << PC1)|(1 << PC0); // Подключаем подтягивающие резисторы

TIMSK &= ~(1 << OCIE2)|(1 << TOIE2); // Запрещаем прерывания по T2
ASSR |= (1 << AS2); // Включаем асинхронный режим T2
TCNT2 = 0; // Сбрасываем регистр счета
TCCR2 |= (1 << CS22)|(1 << CS20); // Предделитель на 128(32768/128 = 256 тиков/с)

sei(); // Глобально разрешаем прерывания

set_brightness(80); // Начальная установка яркости дисплея
display_init(); // Инициализация дисплея

hour = 12;  // Начальная установка времени
minute = 0;
second = 0;

shift_in('t'); // выводим бегущую строку "test display"
_delay_ms(50);
shift_in('E');
_delay_ms(50);
shift_in('s');
_delay_ms(50);
shift_in('t');
_delay_ms(50);
shift_in(' ');
_delay_ms(50);
```

```
shift_in('d');
_delay_ms(50);
shift_in('i');
_delay_ms(50);
shift_in('s');
_delay_ms(50);
shift_in('p');
_delay_ms(50);
shift_in('L');
_delay_ms(50);
shift_in('a');
_delay_ms(50);
shift_in('y');
_delay_ms(50);
shift_in(' ');
_delay_ms(50);
shift_in(' ');
_delay_ms(50);
shift_in(' ');
_delay_ms(50);
shift_in(' ');
_delay_ms(50);
shift_in(' ');
_delay_ms(50);
clear_screen();
_delay_ms(100);

TIMSK |= (1 << TOIE2); // Разрешаем прерывание по переполнению T2

while(1)
{
unsigned char dr;

if((PINC&(1 << PC0)) == 0 && (PINC&(1 << PC1)) == 0) // Если кнопка Часы+ и Минуты+ нажаты
{
dr = 0;
for(unsigned char i = 0; i < 10; i++) // Программный антидребезг контактов
  {
   _delay_ms(5);
   if((PINC&(1 << PC0)) == 0 && (PINC&(1 << PC1)) == 0)
   dr++;
  }
if(dr > 5)
  {
```

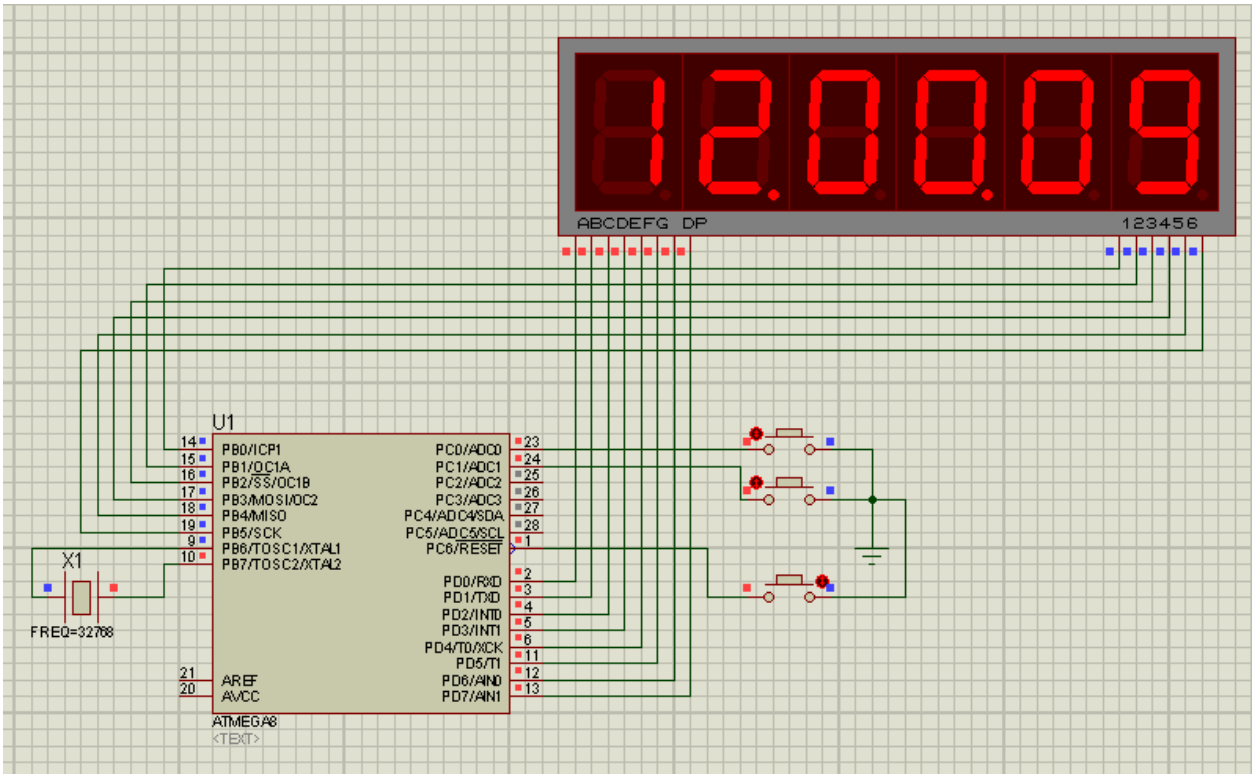
```
    if(level++ != 99) level = 0; // Увеличиваем яркость
    set_brightness(level); // Применяем установленную яркость
   }
  }

  if((PINC&(1 << PC0)) == 0 && PINC&(1 << PC1)) // Если кнопка Часы+ нажата
  {
  dr = 0;
  for(unsigned char i = 0; i < 10; i++) // Программный антидребезг контактов
   {
    _delay_ms(5);
    if((PINC&(1 << PC0)) == 0)
    dr++;
   }
  if(dr > 5)
   {
    if(hour++ >= 23) hour = 0;    // Увеличиваем часы
    set_time(hour,minute,second); // Выводим данные на дисплей
   }
  }

  if((PINC&(1 << PC1)) == 0 && PINC&(1 << PC0)) // Если кнопка Минуты+ нажата
  {
  dr = 0;
  for(unsigned char i = 0; i < 10; i++) // Программный антидребезг контактов
   {
    _delay_ms(5);
    if((PINC&(1 << PC1)) == 0)
    dr++;
   }
  if (dr > 5)
   {
    if(minute++ >= 59) minute = 0; // Увеличиваем минуты
    set_time(hour,minute,second); // Выводим данные на дисплей
   }
  }

  }
  }
```

Работа в Proteus:



Задания для выполнения:

1. Изучить работу Таймера2 в асинхронном режиме на примере микроконтроллера ATmega8.
2. Реализовать приложенную программу в среде Proteus.

Результаты работы отправить на e-mail: rasov@rambler.ru с темой **Таймер2_ФИО**