

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

УТВЕРЖДАЮ

Директор ИШИТР

_____ Д.М. Сонькин

« __ » _____ 2018 г.

В.А. Дорофеев

МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ

Методические указания по выполнению лабораторных работ по курсу
«Микропроцессорные системы»
для студентов, обучающихся по направлению
09.03.01 «Информатика и вычислительная техника»

Издательство
Томского политехнического университета
2018

УДК 004.007
ББК 32.937.26-04 я73
М597

Микропроцессорные системы: методические указания по выполнению лабораторных работ по курсу «Микропроцессорные системы» для студентов, обучающихся по направлению 09.03.01 «Информатика и вычислительная техника» / В.А. Дорофеев. – Томск: Изд-во Томского политехнического университета, 2018. – 26 с.

УДК 681.325.5-181.48 (07)
ББК 32.937.26-04 я73

Методические указания рассмотрены и рекомендованы
к изданию методическим семинаром
кафедры Информационных систем и технологий ИК
«___» _____ 20__ г.

Зав. кафедрой информационных систем и технологий
кандидат технических наук

_____ *А.В. Погребной*

Председатель учебно-методической
комиссии

_____ *В.И. Рейзлин*

Рецензент

Доцент кафедры ИПС Томского политехнического университета
кандидат технических наук
В.И. Рейзлин

© ФГАОУ ВО НИ ТПУ, 2018
© Дорофеев В.А, 2018

Введение

Данные учебно-методические указания предназначены для выполнения лабораторных работ по курсу «Микропроцессорные системы» для бакалавров направления 09.03.01 «Информатика и вычислительная техника».

В первой части описывается настройка используемого оборудования и среды разработки, а также порядок создания нового программного проекта.

Вторая часть содержит шесть лабораторных работ, каждая из которых посвящена исследованию одной из возможностей микропроцессора AVR. Для каждой работы приведены необходимые теоретические сведения и рекомендации по практической реализации.

Темы для лабораторных работ подобраны таким образом, чтобы в максимальной степени осветить возможности микропроцессоров AVR и дать практическое применение знаниям, полученным в теоретической части курса «Микропроцессорные системы».

Первоначальная настройка оборудования

Для проведения лабораторных работ используется микроконтроллер STK600 компании Atmel. Данное устройство принадлежит семейству Starter Kit и служит для демонстрации возможностей микропроцессоров семейства AVR, а также для образовательных целей. Преимуществом данного устройства является простота и наглядность.

Подключение к компьютеру осуществляется через интерфейс USB. Для корректной работы платы STK600 нужно соединить контакты ISP6PIN и SPROG3 6-жильным кабелем, который входит в комплект с устройством.

Для упрощения работы с различными выводами микропроцессора на плату выведены контакты, непосредственно подключенные к ножкам процессора или устройств, размещенных на плате. Так, например, если порт C будет использоваться для управления индикаторами, то группу выводов, помеченных на плате как «PORTC», нужно соединить с группой выводов, помеченных словом «LEDS» 10-жильным кабелем из комплекта. Аналогичные действия требуются для подключения какого-либо порта к кнопкам – соответствующая им группа выводов помечена словом «SWITCHES».

Работа со средой разработки Atmel Studio

Для выполнения лабораторных работ по курсу «Микропроцессорные системы» используется среда разработки *Atmel Studio 7.0* от компании Atmel. На лабораторных компьютерах этот пакет уже установлен, а для домашних экспериментов его можно бесплатно скачать с сайта компании Atmel.

Для создания нового проекта нужно запустить среду Atmel Studio, выбрав соответствующий ярлык в меню Пуск. После запуска среды нужно создать новый проект: для этого можно выбрать команду *File* → *New* → *Project*. Тип проекта следует указать «GCC C Executable Project», в поле *Name* ввести желаемое имя проекта. Рекомендуется установить галочку *Create directory for solution*, в этом случае все файлы проекта будут помещены в отдельную папку.

На следующей странице мастера создания нового проекта (*Device Selection*) нужно указать тип устройства: Atmega2560. После нажатия кнопки ОК среда Atmel Studio сформирует все необходимые файлы и откроет файл с основным текстом программы для редактирования.

Компиляция программы осуществляется выбором команды *Compile* в меню *Build*. Если в процессе компиляции были обнаружены ошибки, то в окне *Errors List* будет выведен их список с указанием номеров строк и пояснениями.

Если в процессе компиляции ошибок не обнаружено, то можно запустить откомпилированную программу на устройстве. Для этого нужно указать среде Atmel Studio на каком устройстве будет запускаться программа. В меню *Project* следует открыть свойства проекта (*Properties*) и в разделе *Tool* указать устройство STK600. Устройство должно быть подключено к компьютеру и быть включенным, в противном случае Atmel Studio не отобразит его в списке доступных вариантов.

Структура программы

Программы для микропроцессора ATmega2560 можно писать на языках Ассемблер и Си. В данных методических указаниях приведены примеры для языка Си.

Программа состоит из нескольких основных частей. Во-первых, в начале программы присутствуют директивы `#include`, которые подключают к программе дополнительные заголовочные файлы с определениями констант и макросов для обеспечения работы с нужными функциями микропроцессора, например:

- `avr/io.h` – ввод-вывод
- `avr/interrupt.h` – прерывания
- `avr/wdt.h` – сторожевой таймер

Другая важная часть программы – это функция `main`, которая выполняется после включения устройства. Как правило, в этой функции размещаются команды для инициализации портов ввода-вывода, служебных регистров и переменных, а также главный цикл программы, который будет выполняться всё время, пока устройство включено.

Настройка портов ввода-вывода заключается в определении направления работы каждого бита порта. Например, если нужно настроить порт А на вход (для чтения состояния кнопок), а порт С на выход (для управления светодиодами), то следует записать в регистры настройки портов соответствующие значения:

```
DDRA = 0b00000000;  
DDRC = 0b11111111;
```

Также следует помнить, что в комплекте STK600 у кнопок и индикаторов используется инверсная логика: нажатая кнопка и горящий светодиод соответствуют нулевому уровню сигнала, отпущенная кнопка и погашенный светодиод – единичному.

Другим важным отличием программ для микроконтроллеров от программ для настольных компьютеров является то, что программы для микроконтроллеров никогда не заканчивают своё выполнение и, следовательно, должны быть зациклены в бесконечном цикле:

```
// Бесконечный цикл  
while (1)  
{  
    // ...  
}
```

Операции с битами

В ряде случаев может потребоваться устанавливать или считывать отдельные биты в регистрах или портах. Для этого можно использовать логические операции И, ИЛИ, НЕ. Рассмотрим примеры.

Установка бита

Чтобы установить бит 5 в регистре EIMSK нужно выполнить операцию «логическое ИЛИ» следующим образом:

```
EIMSK = EIMSK | 0b00100000;
```

Не всегда удобно задавать биты в двоичном виде, поэтому можно использовать операцию сдвига: она сдвигает биты в указанном стрелками направлении, а освободившиеся места заполняет нулями. Поэтому предыдущий пример можно записать следующим образом:

```
EIMSK = EIMSK | (1 << 5);
```

В этом случае единица из первого разряда будет сдвинута на пять бит влево, а затем будет произведена операция «логическое ИЛИ». Можно устанавливать сразу несколько битов, например, третий и пятый:

```
EIMSK = EIMSK | (1 << 5) | (1 << 3);
```

Сброс бита

Чтобы сбросить бит 5 в регистре EIMSK нужно выполнить операцию «логическое И» с инвертированным значением маски следующим образом:

```
EIMSK = EIMSK & ~(1 << 5);
```

В этом примере после сдвига получится значение 0b00100000, затем оно будет инвертировано в 0b11011111, а операция «логическое И» оставит в регистре EIMSK неизменными все биты, которые в маске равны единице, а биты, которые в маске равны нулю, будут сброшены.

Чтение бита

Чтобы проверить значение бита 5 в регистре EIMSK, нужно выполнить операцию «логическое И» с соответствующей маской, и проверить равно ли значение нулю или нет:

```
if (EIMSK & (1 << 5))  
{
```

```

        // Бит установлен
    }
else
{
    // Бит сброшен
}

```

Краткая запись

Язык C позволяет использовать сокращённую запись логических операций, помещая знак операции до знака присваивания. Так, например, операцию

```
EIMSK = EIMSK | (1 << 5);
```

можно записать следующим образом:

```
EIMSK |= (1 << 5);
```

Аналогичные действия возможны и для других операций, например, сложения. Следующие варианты записи будут эквивалентны:

```
N = N + 5;
```

```
N += 5;
```

Именованные биты

Для удобства многие биты в библиотеке AVR имеют свои аббревиатуры. Так, например, бит, соответствующий прерыванию 5 в регистре EIMSK имеет аббревиатуру INT5. Рекомендуется всегда использовать при работе с битами аббревиатуры, а не числовые значения, потому что в разных микроконтроллерах эти биты могут находиться на других позициях в регистре, или даже быть в другом регистре. Поэтому вместо операции

```
EIMSK = EIMSK | (1 << 5);
```

лучше использовать такую форму записи:

```
EIMSK = EIMSK | (1 << INT5);
```


Лабораторная работа 1. Знакомство с комплектом STK600

Цель: научиться практической работе с комплектом STK600, написанию программ на языке Си.

Задание: написать программу, по каждому нажатию кнопки выполняющую очередной шаг последовательности:

- бегущий огонь слева направо или справа налево;
- уменьшающийся или увеличивающийся двоичный счётчик;
- бегущие навстречу друг другу огни;
- свой вариант, по согласованию с преподавателем.

Упрощенный пример программы с пояснениями:

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

int main(void)
{
    // Настраиваем порты
    DDRA = 0b00000000; // Порт А - на вход
    DDRC = 0b11111111; // Порт С - на выход

    // Бесконечный цикл
    while (1)
    {
        // Считываем состояние кнопок
        int T = PINA;
        // Зажигаем светодиоды, соответствующие
        // нажатым кнопкам
        PORTC = T;
    }
}
```

После запуска данная программа ожидает нажатия кнопок и зажигает соответствующие нажатым кнопкам светодиоды.

Для данного примера не важна длительность нажатия на кнопку, но при выполнении основного задания нужно реализовать в программе ожидание нажатия и отпускания кнопок, иначе даже самое короткое нажатие кнопки вызовет очень быстрое переключение состояний светодиодов. Примерный алгоритм может быть таким:

```
// Ожидаем пока не будет нажата какая-либо кнопка
while (PINA == 0b11111111);
// Обработка нажатия кнопок
```

```
...  
// Ожидаем пока все кнопки не будут отпущены  
while (PINA == 0b11111111);
```

Однако, иногда при нажатии кнопки программа всё же может проскакать сразу несколько шагов. Это явление вызвано дребезгом контактов, и может быть исправлено либо на аппаратном уровне, либо программным путём. Однако, такие исправления усложняют логику программы, поэтому в данных лабораторных работах допускается проявление эффекта дребезга контактов.

Лабораторная работа 2. Прерывания

Цель: научиться использовать прерывания микропроцессора AVR.

Задание: аналогично заданию из лабораторной работы 1, но действия с индикаторами выполнять по какому-либо прерыванию (INT0...INT7).

Для выполнения данной лабораторной работы нужно подключить блок кнопок к порту, который содержит отвечающий за нужное прерывание бит:

Таблица 1

Соответствие битов портов прерываниям

Прерывание	Порт	Бит
INT0	D	0
INT1	D	1
INT2	D	2
INT3	D	3
INT4	E	4
INT5	E	5
INT6	E	6
INT7	E	7

Например, для срабатывания прерывания INT4 нужно подключить блок кнопок к порту E и нажимать кнопку SW4, так как именно она подключена к 4 биту порта E.

В данной лабораторной работе функция main должна настроить работу прерываний и войти в пустой бесконечный цикл – переключение индикации в следующую фазу осуществляется в обработчике прерывания.

Для организации работы прерываний необходимо подключить к программе включаемый файл, который содержит объявления используемых констант и макросов:

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

Обработчик прерывания описывается с помощью макроса ISR, параметром которого служит вектор нужного прерывания:

```
ISR(INT0_vect)
{
    // Здесь размещается обработка прерывания INT0
}
```

Для разрешения внешнего прерывания нужно установить соответствующий этому прерыванию бит регистра EIMSK (*External Interrupt Mask Register*) в единицу. Для прерываний INT0 и INT1 с помощью регистров EICRA или EICRB (*External Interrupt Control Register*) можно выбрать тип события, которое вызывает прерывание, например, прерывание по низкому уровню сигнала, по фронту, по спаду или по изменению уровня:

Таблица 2

Используемые биты регистра EIMSK

7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

Таблица 3

Используемые биты регистра EICRA

7	6	5	4	3	2	1	0
ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00

Таблица 4

Используемые биты регистра EICRB

7	6	5	4	3	2	1	0
ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40

Значения битов *Interrupt Sense Control* для задания режима прерывания (только для прерываний INT0 и INT1), вместо x подставляется номер используемого прерывания:

Таблица 5

Режимы прерывания

ISCx1	ISCx0	Описание режима
0	0	Прерывание по низкому уровню INTx
0	1	Прерывание по изменению уровня INTx
1	0	Прерывание по изменению 1→0 уровня INTx

1	1	Прерывание по изменению 0→1 уровня INTx
---	---	---

После настройки регистров нужно разрешить прерывания функцией `sei()`, иначе процессор не будет обрабатывать сигналы прерывания. Для запрета прерываний используется функция `cli()`.

Лабораторная работа 3. Таймер-счётчики

Цель: научиться использовать таймер-счётчики микропроцессора AVR в режиме прерывания по переполнению.

Задание: аналогично заданию из лабораторной работы 1, но действия с индикаторами должны выполняться автоматически с интервалом в 1 секунду.

В микропроцессоре ATmega2560 есть два 8-разрядных таймер-счётчика (0 и 2), и четыре 16-разрядных таймер-счётчика (1, 3, 4 и 5). Каждый из таймер-счётчиков имеет также свои особенности в задании режимов работы.

Рассмотрим более подробно таймер-счётчик 0. Описание остальных таймер-счётчиков можно найти в документации по микропроцессору ATmega2560.

Для управления таймер-счётчиком 0 используется два регистра – TCCR0B (*Timer/Counter Control Register*) и TCNT0 (*Timer/Counter Register*).

Таблица 6

Используемые биты регистра TCCR0B

7	6	5	4	3	2	1	0
					CS02	CS01	CS00

Биты 3-7 в режиме работы таймер-счётчика по переполнению не используются и должны быть установлены в ноль. Биты 0-2 (*Clock Select*) задают делитель тактовой частоты:

Таблица 7

Делители тактовой частоты

CS02	CS01	CS00	Делитель
0	0	0	Таймер-счётчик остановлен
0	0	1	T
0	1	0	T/8
0	1	1	T/64
1	0	0	T/256
1	0	1	T/1024
1	1	0	Внешний источник тактовой частоты (по спаду)
1	1	1	Внешний источник тактовой частоты (по фронту)

Тактовая частота может отличаться в различных устройствах; для комплекта STK600 и процессора *Atmega2560* тактовая частота по умолчанию равна 8 МГц, однако это значение может быть изменено.

Регистр TCNT0 содержит значение счётчика. Это значение увеличивается на единицу с тактовой частотой с учетом делителя частоты. Например, если выбран делитель тактовой частоты T/1024, то значение регистра TCNT0 будет увеличиваться с частотой 7.8 КГц (8 МГц / 1024). Прерывание происходит в тот момент, когда значение счётчика из максимально возможного сбрасывается в ноль. Поскольку нулевой таймер-счётчик восьмиразрядный, то максимальное значение счётчика равно 255.

Таким образом, частота прерываний рассчитывается по следующей формуле 1.

$$F = \frac{F_{CPU}}{C \cdot V} \quad (1)$$

где F – частота прерываний, F_{CPU} – частота процессора, C – значение делителя, V – разница между максимальным и начальным значением счётчика.

Из этой формулы можно вычислить, что максимальный интервал срабатывания нулевого таймер-счётчика равен примерно 30 Гц, или 33 мс. Если требуется выполнять какие-то действия с большим интервалом, то можно либо использовать 16-разрядный таймер-счётчик, либо с помощью глобальной переменной каждый раз пропускать несколько прерываний.

Для разрешения прерываний по переполнению нулевого таймер-счётчика нужно установить в единицу нулевой бит регистра TIMSK0, (*Timer/Counter Interrupt Mask Register*) который представляет собой маску прерываний таймер-счётчика 0. Каждый из его битов разрешает или запрещает один из режимов прерывания таймер-счётчиков:

Таблица 8

Биты регистра TIMSK0

7	6	5	4	3	2	1	0
					OSIE0B	OSIE0A	TOIE0

Бит TOIE0 (*Timer/Counter Overflow Interrupt Enable*) включает режим прерывания по переполнению таймер-счетчика 0, а биты OSIE0A и OSIE0B – (*Timer/Counter Output Compare Interrupt Enable*) режим прерывания по сравнению, когда значение счетчика достигает

эталонного.

Для организации обработчика прерывания по переполнению используется макрос ISR():

```
ISR(TIMER0_OVF_vect)
{
    // Помещаем в TCNT0 новое начальное значение
    TCNT0 = 256 - 30;
    // Затем можно обработать само прерывание
    ...
}
```


Лабораторная работа 4. Генерация сигналов

Цель: научиться использовать таймер-счётчики микропроцессора AVR для генерации сигналов с заданной частотой.

Задание: написать программу-синтезатор, которая по нажатию каждой из кнопок воспроизводит соответствующую ей музыкальную ноту.

Для генерации сигналов лучше всего подходит таймер-счётчик 1. В данной лабораторной работе он используется в режиме СТС – *Clear Timer on Compare*, в котором значение таймера автоматически сбрасывается в ноль при достижении заданной в регистре OCR1A (*Output Compare Register*) величины. При каждом таком сбросе значение бита OC1A (*Output Compare*, соответствующее пятому биту порта В, PB5) изменяется на противоположное. Соответственно, два таких изменения ($0 \rightarrow 1$, $1 \rightarrow 0$) создают требуемый сигнал:

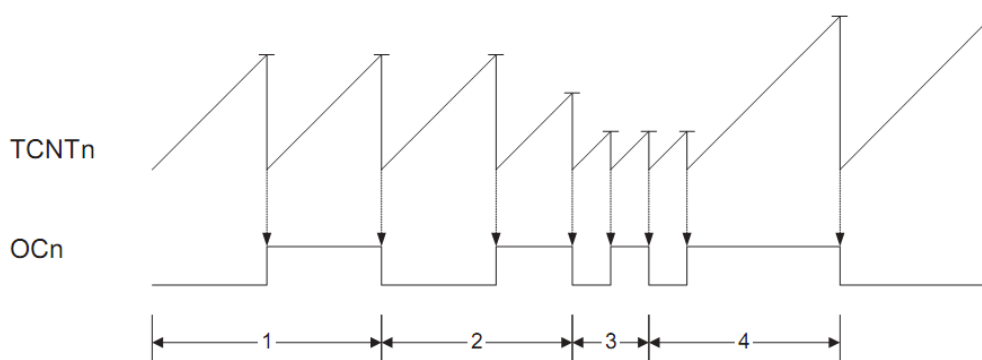


Рис. 1. Генерация частоты на выходе OCn

Значение частоты полученного сигнала для таймер-счётчика 1 можно рассчитать по следующей формуле:

$$f_{OC1} = \frac{f_{CPU}}{2 \cdot N \cdot (1 + OCR1A)} \quad (2)$$

где f_{OC1} – это требуемая частота, f_{CPU} – частота процессора, N – делитель частоты, $OCR1A$ – значение регистра OCR1A, до которого будет производиться счёт. Делитель частоты N лучше брать небольшой (1 или 8), поскольку при большом делителе частоты величины значения OCR1A получаются достаточно близкими и возрастает погрешность частоты.

Таблица 9

Биты регистра TCCR1A

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10
--------	--------	--------	--------	--------	--------	-------	-------

Биты 6-7 (*Compare Output Mode*) задают режим сравнения таймер-счётчика 1:

Таблица 10

Режимы сравнения таймер-счётчика 1

COM1A1	COM1A0	Режим
0	0	Канал OC1A отключен
0	1	Инверсия OC1A при совпадении
1	0	Очистка OC1A при совпадении
1	1	Установка OC1A при совпадении

Для генерации звука нужно выбрать режим инверсии при совпадении.

Таблица 11

Используемые биты регистра TCCR1B

7	6	5	4	3	2	1	0
ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10

Биты WGM10–WGM13 (*Waveform Generation Mode*) задают режим генерации сигналов, например, разновидности широтно-импульсной модуляции. Режиму CTC соответствуют биты WGM равные 0100 (другие режимы описаны в документации). Обратите внимание, что для правильной установки режима генерации сигналов нужно менять значения как в регистре TCCR1A, так и в регистре TCCR1B.

Биты 5-7 регистра TCCR1B в режиме таймер-счётчика CTC не используются и должны быть установлены в ноль. Биты 0-2 задают делитель тактовой частоты (см. таблицу 7).

Для генерации сигналов, соответствующих музыкальным нотам, нужно использовать следующие частоты:

Таблица 12

Частоты музыкальных нот, Гц

Нота	Октава				
	Большая	Малая	1	2	3
До	65,4	130,8	261,6	523,3	1046,5
До-диез	69,3	138,6	277,2	554,4	1108,7
Ре	73,4	146,8	293,7	587,3	1174,6
Ре-диез	77,8	155,6	311,1	622,3	1244,5
Ми	82,4	164,8	329,6	659,3	1318,5
Фа	87,3	174,6	349,2	698,5	1396,9
Фа-диез	92,5	185,0	370,0	740,0	1480,0
Соль	98,0	196,0	392,0	784,0	1568,0
Соль-диез	103,8	207,7	415,3	830,6	1661,2
Ля	110,0	220,0	440,0	880,0	1760,0
Ля-диез	116,5	233,1	466,2	932,3	1864,6
Си	123,5	246,9	493,9	987,8	1975,5

Лабораторная работа 5. Сторожевой таймер

Цель: научиться использовать сторожевой таймер микропроцессора AVR.

Задание: написать программу, производящую некоторые визуальные действия. Реализовать включение и отключение сторожевого таймера, а также имитацию зависания программы по нажатию выбранных кнопок. При запуске программы реализовать анализ причин запуска – обычный ли это запуск программы или в результате срабатывания сторожевого таймера. Отображать результаты анализа зажиганием светодиодов на три секунды.

Сторожевой таймер (*Watchdog Timer*) представляет собой механизм для слежения за выполнением программы и её перезапуском в случае необходимости. Сторожевой таймер не включен по умолчанию, его можно активировать и деактивировать с помощью регистра *WDTCR* (*Watchdog Timer Control Register*). Сторожевой таймер выполнен в виде независимого модуля, работающего на частоте 128 кГц.

Таблица 13

Используемые биты регистра WDTCR

7	6	5	4	3	2	1	0
			WDCE	WDE	WDP2	WDP1	WDP0

Биты *WDP0...WDP3* (*Watchdog Prescaler*) определяют количество циклов, в течение которых сторожевой таймер ожидает сброса счётчика:

Таблица 14

Интервалы сторожевого таймера

WDP3	WDP2	WDP1	WDP0	Количество циклов	Время срабатывания
0	0	0	0	2 048	16 мс
0	0	0	1	4 096	32 мс
0	0	1	0	8 192	64 мс
0	0	1	1	16 384	125 мс
0	1	0	0	32 768	250 мс
0	1	0	1	65 536	500 мс
0	1	1	0	131 072	1 с
0	1	1	1	262 144	2 с
1	0	0	0	524 288	3 с

1	0	0	1	1 048 576	4 с
---	---	---	---	-----------	-----

Если за данное количество циклов не поступило команды сброса сторожевого таймера, то программа будет перезапущена. Сторожевой таймер сбрасывается командой

```
wdt_reset();
```

Установка бита WDE разрешает работу сторожевого таймера. Однако, изменение состояния сторожевого таймера (включение и отключение) выполняется в два этапа, чтобы исключить возможность случайного срабатывания. Сначала нужно единицы в биты WDCE (*Watchdog Change Enable*) и WDE (*Watchdog System Reset Enable*) а в течение следующих четырёх циклов записать ноль в бит WDE.

Определить причину запуска программы можно путём анализа битов в регистре MCUSR (*MCU Status Register*):

Таблица 15

Используемые биты регистра MCUSR

7	6	5	4	3	2	1	0
			JTAGRF	WDRF	BORF	EXTRF	PORF

Так, если бит WDRF (*Watchdog Reset Flag*) установлен в единицу, значит, произошло срабатывание сторожевого таймера. Значения других битов можно найти в документации. Если программа была перезапущена сторожевым таймером, то данный бит WDRF должен быть сброшен вручную.

Лабораторная работа 6. Интерфейс RS-232

Цель: научиться использовать коммуникационные возможности микропроцессора AVR.

Задание: модифицировать программу из задания 4 таким образом, чтобы управление осуществлялось нажатием компьютерных клавиш, при этом в терминале должно появляться название проигрываемой ноты.

Микропроцессор AVR может использовать интерфейс RS-232 для обмена данными с внешними источниками. На плате STK600 предусмотрено два разъема RS-232: один, отмеченный меткой CAN, предназначен для использования в промышленных контроллерах, второй – помеченный на плате как RS232 – может использоваться для получения и передачи данных.

Для подключения выводов процессора к интерфейсу нужно с помощью двухжильного кабеля подключить контакты PE0 и PE1 к контактам RXD и TXD соответственно, расположенным в блоке контактов RS232 SPARE.

Чтобы принимать и передавать данные на персональном компьютере, нужно запустить и настроить какую-либо программу-терминал (например, *HyperTerminal* или *Putty*) следующим образом:

- Порт: COM-порт, к которому подключен кабель, например, COM1
- Скорость: 9600
- Биты данных: 8
- Четность: нет
- Стоповые биты: 1
- Управление потоком: нет

Микропроцессор ATmega2560 имеет четыре независимых блока USART (*Universal Asynchronous Receiver-Transmitter*, *Универсальный асинхронный приёмопередатчик*), используемых для приема и передачи данных. В данной лабораторной работе будет использоваться USART0.

Перед приемом или передачей данных нужно выполнить ряд настроек в регистрах UCSR_n (*USART Control and Status Register*):

- Регистр UCSR0A используется для синхронной передачи данных, поэтому в данной лабораторной работе его следует обнулить.
- В регистре UCSR0B следует установить в единицу биты RXEN0

(*Receiver Enable*) и *TXEN0 (Transmitter Enable)* включающие приемник и передатчик нулевого блока USART соответственно.

- Регистр *UCSR0C* определяет параметры передачи данных:

Таблица 16

Биты регистра UCSR0C

7	6	5	4	3	2	1	0
	UMSEL	UPM1	UPM0	USBS	UCSZ2	UCSZ1	UCSZ0

Бит *UMSEL (USART Mode Select)* определяет тип передачи (0 – асинхронная, 1 – синхронная). В данной лабораторной работе используется асинхронная передача.

Биты *UPM0* и *UPM1 (USART Parity Mode)* отвечают за режим четности, для данной лабораторной работы они должны быть равны нулю.

Бит *USBS (USART Stop Bit Select)* определяет количество стоповых бит (0 – один бит, 1 – два бита).

Биты *UCSZx (USART Character Size)* регламентируют количество битов данных:

Таблица 17

Количество бит данных

UCSZ2	UCSZ1	UCSZ0	Размер символа
0	0	0	5 бит
0	0	1	6 бит
0	1	0	7 бит
0	1	1	8 бит
1	1	1	9 бит

Наконец, значение в регистре *UBRR0 (USART Baud Rate Register)* определяет скорость передачи данных. Расчёт значений производится по следующей формуле:

$$UBRR = \frac{f_{CPU}}{16 \cdot BAUD} - 1 \quad (3)$$

где *UBRR* – искомое значение, *f_{CPU}* – частота процессора, *BAUD* – желаемая скорость передачи. Если в результате расчётов получается дробное число, его нужно округлить до ближайшего целого.

Для чтения или записи данных используется регистр *UDR (USART Data Register)*. Однако перед выполнением операции записи нужно дождаться обнуления бита *UDRE0 (USART Data Register Empty)* в регистре

UCSR0A, а перед чтением – дождаться обнуления бита RXC0 (*Receive Complete*) в этом же регистре:

```
char UART_getchar()
{
    while (!(UCSR0A & (1 << RXC0)));
    return UDR0;
}

void UART_putchar(unsigned char c)
{
    while (!(UCSR0A & (1 << UDRE0)));
    UDR0 = c;
}
```


Содержание

Введение	3
Первоначальная настройка оборудования	4
Работа со средой разработки Atmel Studio	5
Структура программы.....	6
Операции с битами	7
Установка бита	7
Сброс бита.....	7
Чтение бита	7
Краткая запись.....	8
Именованные биты	8
Лабораторная работа 1. Знакомство с комплектом STK600	9
Лабораторная работа 2. Прерывания	11
Лабораторная работа 3. Таймер-счётчики	14
Лабораторная работа 4. Генерация сигналов.....	17
Лабораторная работа 5. Сторожевой таймер	20
Лабораторная работа 6. Интерфейс RS-232.....	22

Учебное издание

ДОРОФЕЕВ Вадим Анатольевич

МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ

Методические указания к выполнению лабораторных работ по курсу «Микропроцессорные системы» для студентов, обучающихся по направлению 09.03.01 «Информатика и вычислительная техника»


Отпечатано в Издательстве ТПУ в полном соответствии с качеством предоставленного оригинал-макета

Подписано к печати . .2018. Формат 60x84/16. Бумага «Снегурочка».
Печать XEROX. Усл. печ. л. 9,01. Уч.-изд. л. 8,16.
Заказ 000-15. Тираж 100 экз.



Национальный исследовательский Томский политехнический университет
Система менеджмента качества
Издательства Томского политехнического университета
сертифицирована в соответствии с требованиями ISO 9001:2008



ИЗДАТЕЛЬСТВО  **ТПУ** . 634050, г. Томск, пр. Ленина, 30
Тел./факс: 8(3822) 56-35-35, www.tpu.ru