МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КРАСНОДАРСКОГО КРАЯ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ КРАСНОДАРСКОГО КРАЯ

«КРАСНОДАРСКИЙ КОЛЛЕДЖ ЭЛЕКТРОННОГО ПРИБОРОСТРОЕНИЯ»

Методические указания

**ДЛЯ ПРОВЕДЕНИЯ ЛАБОРАТОРНЫХ РАБОТ**

по дисциплине (МДК 02.02): «Микропроцессорные системы»

для специальностей: 230113 «Компьютерные системы и комплексы»

|  |  |
| --- | --- |
|  | Разработал преподаватель:Радкевич В.В.рассмотрено и одобренона заседании цикловой комиссии\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(наименование ЦК)протокол № \_\_\_\_\_\_\_\_\_\_\_\_\_ от \_\_\_\_\_\_\_\_\_\_\_председатель ЦК \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (ФИО) |

Краснодар

2015г.

***Теоретические сведения***

***Тема 1 Генерация управляющих импульсных сигналов***

Основная задача микроконтроллеров в составе информационно-управляющих систем, как правило, сводится к формированию сигналов управления и обработке информационных сигналов на их выводах. Наиболее просто происходит процесс формирования цифровых импульсных последовательностей. Рассмотрим некоторые возможные способы решения данной проблемы.

Микроконтроллер ATmega8 имеет 23 программируемые линии ввода/вывода, для определения электрического состояния которых могут быть применены модули портов параллельного ввода/вывода, входящие в состав встроенных периферийных устройств. В зависимости от их конфигурации, на выводах микросхемы могут быть сформированы электрические сигналы высокого электрического уровня (близкие к напряжению питания), либо низкого электрического уровня (близкие к потенциалу общего провода).

Так как импульсный сигнал представляет собой чередование мгновенно изменяющихся электрических уровней от некоторого максимального значения к некоторому минимальному значению потенциалов и наоборот, то рассмотренное свойство параллельных портов ввода/вывода возможно применить для формирования цифровых последовательностей на выводах микроконтроллера.

Цифровой импульс обладает следующими основными характеристиками:

1. Длительность фронта импульса (tфр) – это время, в течение которого происходит переход от низкого электрического уровня к высокому;

2. Длительность импульса (tи) – характеризует время удержания высокого электрического уровня в некотором узком диапазоне значений;

3. Длительность среза импульса (tср) – в течение этого времени происходит обратный переход от высокого электрического уровня у низкому;

4. Длительность паузы (tп) – определяет время, в течение которого сигнал удерживает низкий электрический уровень, т.е. фактически отсутствует;

5. Период следования сигнала (T) – время, прошедшее между двумя ближайшими фронтами импульсной последовательности. Определяется как совокупность длительности импульса и паузы:

T= tи+ tп (1.1)

6. Частота следования сигнала (F) – показывает количество проходящих импульсов за интервал времени в одну секунду. Данная величина обратно пропорциональна периоду следования сигнала:

F=1/T (1.2)

7. Скважность (Q) – величина, показывающая отношение длительности импульса к длительности паузы импульсного сигнала:

Q= tи/ tп (1.3)

8. Амплитуда (размах) импульса (A) – максимальное значение напряжение, которое достигается сигналом за продолжительность одного периода следования.

Рассмотренные характеристики импульсного сигнала поясняются рисунком 1.1 и 1.2.



Рисунок 1.1 – Характеристики последовательности импульсов



Рисунок 1.2 – Характеристики импульса

Таким образом, для формирования импульсов на выводах микроконтроллера управляющая программа должна периодически повторять выполнение следующих функций:

1. Сформировать фронт импульса;

2. Удерживать высокий электрический уровень в течение требуемого промежутка времени, соответствующего длительности импульса;

3. Сформировать срез импульса;

4. Удерживать низкий электрический уровень в течение заданного временного интервала, который соответствует длительности паузы.

***Вопросы для самоконтроля***

1. Что представляет собой импульсный сигнал?

2. Какое свойство параллельных портов ввода/вывода можно использовать для формирования импульсного сигнала?

3. Перечислите основные характеристики импульсного сигнала.

4. Какую последовательность действий должна описывать управляющая программа для формирования импульсного сигнала?

***Тема 1.1 Генерация импульсных последовательностей***

Рассмотрим пример проекта генератора импульсной последовательности с заданными параметрами на выводах микроконтроллера.

Для удобства визуального контроля за процессом формирования импульсов на выводах микроконтроллера воспользуемся светодиодами, установленными на плате AVRmegaboardX8. Компиляция исходного текста программ приведенных примеров будет производится при помощи интегрированной среды разработки CodeVisionAVR.

Схема подключения светодиодов к выводам МК показана на рисунке 1.3. Рассмотрим элементы схемы. Светодиод VD1 имеет красное свечение, VD2 – желтое свечение, а VD1 – зеленое свечение. Резисторы R1 – R3 ограничивают ток, протекающий через светодиоды так, что его значение не превышает 10мА. Светодиоды подключены между выводами микроконтроллера и общим проводом питания схемы, следовательно, для того, чтобы один из светодиодов начал излучать свет, на соответствующем ему выводе необходимо сформировать высокий электрический уровень.



Рисунок 1.3 – Схема подключения светодиодов

Для управления состоянием электрического уровня на выводах МК, к которым подключены светодиоды, будем использовать модуль параллельного порта D. Разряды этого порта, управляющие светодиодами следует настроить на вывод. Для этого запишем логические единицы в соответствующие разряды регистра DDRD, а состояние разрядов регистра PORTD будут определять электрические уровни на выводах микросхемы.

Листинг 1.1 содержит программный текст, формирующий одиночный импульс на выводе седьмого разряда порта D. В результате выполнения представленной программы микроконтроллером можно будет наблюдать кратковременное одиночное включение красного светодиода на время в одну секунду. Рассмотрим данный листинг.

Заголовочный файл mega8.h содержит описание памяти используемого микроконтроллера. В нем произведено именование адресов памяти в диапазоне регистров ввода/вывода в соответствии с именами, предложенными к использованию компанией производителем Atmel в фирменной документации.

В заголовочном файле delay.h представлены прототипы функций delay\_ms() и delay\_us(), формирующих интервалы времени программным методом соответственно в мили и микро секундах.

Листинг, описывающий процесс формирования одиночного импульса, содержится в главной функции main() (строки 13 – 16) и весьма прост. Формирование импульса на выводе микроконтроллера происходит в результате выполнения следующих операций:

1. delay\_ms(2000); – вызов функции формирования временного интервала, в качестве параметров которой передано значение длительности паузы между импульсами;

2. PORTD.7=1; – формирование фронта импульсного сигнала, устанавливает высокий электрический уровень на выводе МК;

3. delay\_ms(1000); – вызов функции формирования временного интервала, в качестве параметров которой передано значение длительности импульса;

4. PORTD.7=0; – формирование фронта импульсного сигнала, устанавливает высокий электрический уровень на выводе МК.

Листинг 1.1 – Формирование одиночного импульса

|  |  |
| --- | --- |
| 123456789101112131415161718 | //-----подключение библиотек---------------------------//-----------------------------------------------------#include <mega8.h> //описание памяти микроконтроллера#include <delay.h> //временные задержки//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* //=====ГЛАВНАЯ ФУНКЦИЯ=================================//=====================================================void main (void){//-----настройка портов ввода/вывода-------------------DDRD=0x80; //7 разряд - выход//-----фоновая программа-------------------------------delay\_ms(2000); //длительность паузыPORTD.7=1; //формирование фронта импульсаdelay\_ms(1000); //длительность импульсаPORTD.7=0; //формирование среза импульса}//END MAIN//===================================================== |

Команды модификации отдельных разрядов регистров типа PORTD.7 являются отступлением от стандарта языка программирования Си и специфичны исключительно для данной среды. Подобные команды интерпретируются компилятором CodeVisionAVR как битовые операции и доступны для ограниченного числа регистров. К отдельным разрядам например переменных подобное обращение не возможно.

В следующем листинге 1.2 команды, формирующие одиночный импульс объединены блоком оператора while, в качестве условия выполнения которому передано значение единицы. В данном случае единица будет интерпретирован как истинно, вследствие чего цикл while будет повторяться бесконечное количество раз, в результате чего на выводе микроконтроллера будет сформирована непрерывная импульсная последовательность.

Листинг 1.2 – Формирование непрерывной импульсной последовательности

|  |  |
| --- | --- |
| 1234567891011121314151617181920212223 | //-----подключение библиотек---------------------------//-----------------------------------------------------#include <mega8.h> //описание памяти микроконтроллера#include <delay.h> //временные задержки//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* //=====ГЛАВНАЯ ФУНКЦИЯ=================================//=====================================================void main (void){//-----локальные объекты-------------------------------const unsigned int Tp=2000,//длительность паузы Ti=1000;//длительность импульса//-----настройка портов ввода/вывода-------------------DDRD=0x80; //7 разряд - выход//-----фоновая программа-------------------------------while(1){ //бесконечный циклdelay\_ms(Tp); //длительность паузыPORTD.7=1; //формирование фронта импульсаdelay\_ms(Ti); //длительность импульсаPORTD.7=0; //формирование среза импульса}//while(1)}//END MAIN//===================================================== |

Текст программы листинга 1.2 обладает еще одним отличием от предыдущего примера. Временные параметры импульсного сигнала здесь определены константным типом, вместо непосредственной передачи значений длительности и паузы импульсного сигнала в качестве входных параметров функции delay\_ms(). Этот прием делает программу проще для восприятия, т.к. вместо конкретной величины текст программы содержит имя, которое проще ассоциировать с некоторым физическим параметром. К тому же упростится возможность дальнейшего изменения значений в программе.

В листинге 1.3 приведен пример программы, которая формирует заданное количество серий импульсов.

При реализации поставленной задачи недостаточно просто формировать импульсы с определенными параметрами, необходимо также производить подсчет их количества. Для этого в программе введены две переменные – счетчики i и j. Счетчик j считает количество пакетов импульсов, а i – импульсов в пакете.

Значение счетчика j инициализировано при объявлении и уменьшается на единицу каждый раз, после проверки условия оператора цикла while. Как только содержимое этой переменной примет значение равное нулю, формирование импульсных последовательностей будет прекращено.

С помощью счетчика i производится подсчет количества импульсов в каждом формируемом пакете. В начале выполнения оператора цикла с счетчиком for переменной i присваивается нулевое значение. При каждом повторном выполнении тела цикла ее содержимое сравнивается с предельным значением. После чего содержимое счетчика i увеличивается на единицу.

Для разделения пакетов импульсов следующих друг за другом перед каждым повторным выполнением оператора for формируется дополнительная пауза.

Листинг 1.3 – Формирование пакетов импульсов

|  |  |
| --- | --- |
| 12345678910111213141516171819202122232425262728293031 | //-----подключение библиотек---------------------------//-----------------------------------------------------#include <mega8.h> //описание памяти микроконтроллера#include <delay.h> //временные задержки//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//=====ГЛАВНАЯ ФУНКЦИЯ=================================//=====================================================void main (void){//-----локальные объекты-------------------------------const unsigned int Tp=2000,//длительность паузы Ti=1000,//длительность импульса T=3000; //задержка между пакетами const unsigned char n=4, //количество пакетов m=5; //количество импульсов unsigned char i, //счетчик импульсов j=n; //счетчик пакетов//-----настройка портов ввода/вывода-------------------DDRD=0x80; //7 разряд - выход//-----фоновая программа-------------------------------while(j--){ //формирование пакетов импульсовfor(i=0;i<m;i++){ //формирование импульсов в пакете delay\_ms(Tp); //длительность паузы PORTD.7=1; //формирование фронта импульса delay\_ms(Ti); //длительность импульса PORTD.7=0; //формирование среза импульса }//for(;;) delay\_ms(T); //задержка между пакетами}//while()}//END MAIN//===================================================== |

***Вопросы для самоконтроля***

1. Опишите особенности схемы включения индикационных светодиодов лабораторного макета.

2. Как осуществляется управление светодиодами схемы?

3. Какие периферийные устройства МК, и в каких режимах, задействованы при реализации управляющих сигналов?

4. Поясните назначение подключаемых файлов mega8.h и delay.h, используемых в приведенных примерах.

5. Как в рассмотренных примерах происходит формирование интервалов времени?

6. Опишите процесс формирования одиночного импульса на выводе микроконтроллера.

7. Как происходит формирование одного пакета импульсов?

8. Как происходит формирование пакетов импульсов?

***Тема 1.2 Генерация сложных последовательностей импульсов***

Как правило, при решении задач программирования систем на основе микроконтроллеров необходимость в формировании простых серий импульсных сигналов возникает не так часто, как необходимость в сложных последовательностях, формируемых по определенным правилам. Характеристики таких сигналов могут отличаться от одного периода к другому, как собственно и количество их повторений.

Как и в предыдущем подразделе воспользуемся светодиодами, установленными на плате AVRmegaboardX8. Компиляция исходного текста программ приведенных примеров также будет выполнена при помощи интегрированной среды разработки CodeVisionAVR.

В качестве первого примера генерации относительно сложной последовательности управляющих сигналов рассмотрим листинг 1.4, в котором реализовано некоторое подобие работы полицейской сирены. В результате выполнения программы производится поочередная манипуляция красным и желтым светодиодами, чередующаяся с их синхронным переключением.

Чередующееся включение красного и желтого светодиодов происходит в строках 20 – 25. Количество повторений считает счетчик i, ограниченный значением m. Цикличность переключений организованна оператором цикла со счетчиком for. Формирование синхронных переключений в строках 26- -31 производится аналогично, но ограничением числа повторов является значение n.

Особое внимание стоит уделить командным строкам 21, 23 и 27, где производятся манипуляции с разрядами регистра PORTD. Последовательно разберем эти команды.

Листинг 1.4 – “Полицейская сирена”

|  |  |
| --- | --- |
| 12345678910111213141516171819202122232425262728293031 | //-----подключение библиотек---------------------------//-----------------------------------------------------#include <mega8.h> //описание памяти микроконтроллера#include <delay.h> //временные задержки//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* //=====ГЛАВНАЯ ФУНКЦИЯ=================================//=====================================================void main (void){//-----локальные объекты-------------------------------const unsigned char n=5, //кол-во поочередных и m=10; //синхронных включениях const unsigned int Tp=500,//задержка при поочередном Ts=200;//и синхронном включениях unsigned char i; //счетчик повторений//-----настройка портов ввода/вывода-------------------DDRD=0xC0; //7,6 разряды - на вывод//-----фоновая программа-------------------------------while(1){ //бесконечный циклfor(i=0;i<n;i++){ //поочередное переключение PORTD=1<<7; //вкл. красного, выкл. желтого delay\_ms(Tp); //задержка между переключениями PORTD=1<<6; //вкл. желтого, выкл. красного delay\_ms(Tp); //задержка между переключениями }for(i=0;i<m;i++){ //синхронное переключение PORTD=1<<7|1<<6;//включение красного и желтого delay\_ms(Ts); //задержка между переключениями PORTD=0; //выключение красного и желтого delay\_ms(Ts); //задержка между переключениями } |

Продолжение листинга 1.4

|  |  |
| --- | --- |
| 323334 | }//while(1)}//END MAIN//===================================================== |

В строке 21 происходит присвоение регистру PORTD значения, полученного в результате операции сдвига влево десятичной единицы на семь двоичных разрядов. Вычисление данного выражения произойдет на стадии компиляции текста программы, в результате чего, в регистр PORTD будет записано вычисленное значение, содержащее единицу в седьмом двоичном разряде, все остальные разряды двоичного числа будут хранить нулевые значения, при этом красный светодиод будет включен, а желтый – выключен. Аналогичным образом происходит включение желтого светодиода и выключение красного в строке 23.

Запись вида “PORTD=1<<7” аналогична записи “PORTD=0x80”, но результат ее более очевиден. Прочесть эту команду можно было бы как: “установить седьмой разряд регистра PORTD в единицу, а всем остальным разрядам присвоить нулевые значения”.

В строке 27 происходит одновременное включение красного и желтого светодиодов. Значение, присваиваемое регистру PORTD, в команде “PORTD=1<<7|1<<6” как и в рассмотренных ранее командах, вычисляется на стадии компиляции. В начале будут вычислены операции сдвига влево десятичных единиц на 6 и 7 двоичных разрядов соответственно. Далее между результатами выполнения этих операций будет произведено логическое сложение (“ИЛИ”). В итоге будет получена бинарная маска с единицами в шестом и седьмом разрядах, все остальные разряды будут хранить нулевой результат. Прочесть данную команду можно как: “установить седьмой и шестой разряды регистра PORTD в единицу, а всем остальным разрядам присвоить нулевые значения”.

В следующем примере разберем текст программы, формирующей последовательности импульсных сигналов на выводах микроконтроллера для управления подключенными к ним светодиодами в соответствии с алгоритмом работы светофора.

Для упрощения определения граничных значений времени переключения сигналов “светофора”, последовательность управляющих сигналов, которую необходимо сформировать, представим в виде графика на рисунке 1.4. Чтобы абстрагироваться от конкретных значений, всем интервалам времени на графике дадим имена, которыми будем пользоваться в программе.



Рисунок 1.4 – График переключения сигналов “светофора”

Рассмотрим значение интервалов времени приведенных на рисунке:

1. Tr – длительность работы красного сигнала светофора;

2. Tg – длительность работы зеленого сигнала светофора;

3. Ty – длительность работы желтого сигнала светофора;

4. Tp – длительность интервалов при переключении зеленого сигнала светофора. Так как длительность переключений для всех шести интервалов имеет равное значение, то на графике время Tp умножено на 6.

На графике переключений видно, что некоторые интервалы времени накладываются друг на друга по шкале времени. Так в завершении цикла работы красного сигнала светофора на некоторое время включается желтый сигнал. А во время работы зеленый сигнал светофора не все время включен непрерывно, цикл его работы завершается тремя переключениями. Следовательно, для организации правильной работы “светофора” ко времени, необходимо вычислить значения интервалов между переключениями его сигналов.

В результате получим следующий алгоритм работы “светофора”:

1. Включение красного сигнала светофора на время Tr-Ty;

2. Включение желтого сигнала светофора на время Ty;

3. Выключение красного и желтого сигналов светофора;

4. Включение зеленого сигнала светофора на время Tg-6\*Tp;

5. Формирование одной серии, состоящей их трех импульсов длительность импульса и паузы которых равны Tp, для управлением переключений зеленого сигнала светофора;

6. Выключение зеленого сигнала светофора;

7. Переход к выполнению пункта 1.

Данный алгоритм обработки реализован в листинге 1.5.

В этом примере также стоит уделить внимание командам манипуляции разрядами регистров использованных в строках 20, 22, 27 и 29. Команды, подобные приведенным в строках 24 и 32, уже были рассмотрены в предыдущих примерах.

Рассмотрим команду “PORTD|=1<<6”, расположенную в строке 22. Операция сдвига влево “1<<6” в результате дает бинарную маску, в которой шестой разряд установлен в единицу, а остальные – сброшены. Операция “|=” обозначает, что производится логическое сложение значений в правой и левой частях выражения, после чего результат передается объекту с лева. Таким образом, с помощью данной команды можно установить один разряд регистра или переменной, не влияя на содержимое остальных разрядов.

Команда “PORTD&=~(1<<5)”, расположенная в строке 27, позволяет сбросить один заданный разряд регистра или переменной, не влияя при этом на прочее содержимое. Инверсия значения, полученного в результате операции сдвига влево десятичной единицы, дает бинарную маску, в которой все двоичные разряды установлены, а разряд, позиция которого совпадает с количеством сдвигов влево, сброшен. Операция ‘&=’, аналогично ‘|=’, объекту слева передает результат операции логического умножения над значениями левой и правой частей, в результате чего происходит сброс заданного разряда.

Листинг 1.5 – “Светофор”

|  |  |
| --- | --- |
| 1234567891011121314151617181920212223 | //-----подключение библиотек---------------------------//-----------------------------------------------------#include <mega8.h> //описание памяти микроконтроллера#include <delay.h> //временные задержки//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* //=====ГЛАВНАЯ ФУНКЦИЯ=================================//=====================================================void main (void){//-----локальные объекты-------------------------------const unsigned int Tr=10000,//время вкл. красного Tg=12000,//время вкл. зеленого Ty=2000, //время вкл. желтого Tp=1000; //период перекл. зеленого unsigned char i; //счетчик переключений зеленого//-----настройка портов ввода/вывода-------------------DDRD=0xE0; //7,6 и 5 разряды - на вывод//-----фоновая программа-------------------------------while(1){ //бесконечный циклPORTD|=1<<7; //вкл. красныйdelay\_ms(Tr-Ty); //пауза вкл. красногоPORTD|=1<<6; //вкл. желтыйdelay\_ms(Ty); //пауза вкл. желтого+красного |

Продолжение листинга 1.5

|  |  |
| --- | --- |
| 2425262728293031323334353637 | PORTD=1<<5; //выкл. желтый+красный, вкл. зеленыйdelay\_ms(Tg-6\*Tg); //пауза вкл. зеленогоfor(i=0;i<3;i++){ //цикл переключения зеленого сигнала PORTD&=~(1<<5);//выкл. зеленый delay\_ms(Tp); //задержка выключения PORTD|=1<<5; //вкл. зеленый delay\_ms(Tp); //задержка включения } PORTD=1<<6; //выкл. зеленый, вкл. желтыйdelay\_ms(Ty); //пауза вкл. желтогоPORTD=0; //выключение "светофора"}//while(1)}//END MAIN//===================================================== |

В некоторых случаях рассмотренное решение может быть не приемлемым, т.к. выполнение программы занимает все свободное процессорное время.

Алгоритм, описанный в листинге 1.5, представляет собой некоторое подобие сценария, на основе которого производится заданная последовательность действий. При этом большую часть времени микроконтроллер занимается формированием интервалов времени, а управление сигналами светофора выполняется сравнительно не долго.

Конечно, можно было бы заменить команды вызова функций, реализующих задержки времени, на эквивалентные по времени обработчики некоторых параллельных задач, но это было бы крайне не удобно с точки зрения реализации. При этом возможность дальнейшего корректирования графика управления светофором значительно затруднится.

Применив парадигму автоматного программирования для решения данной задачи, получим программу, описанную в листинге 1.6, которая моделирует конечный автомата, описывающий состояния сигналов светофора.

Для удобства написания текста программы, график на рисунке 1.4 был дополнен метками времени, характеризующими состояния сигналов светофора, т.е. фазы конечного автомата. В результате получен рисунок 1.5, на котором отмечены интервалы времени между включением сигналов и началом цикла работы светофора.



Рисунок 1.5 – График переключения сигналов “светофора”

Листинг 1.6 – “Светофор”

|  |  |
| --- | --- |
| 12345 | //-----подключение библиотек---------------------------//-----------------------------------------------------#include <mega8.h> //описание памяти микроконтроллера#include <delay.h> //временные задержки//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

Продолжение листинга 1.6

|  |  |
| --- | --- |
| 67891011121314151617181920212223242526272829303132333435 | //-----замена имен типов-------------------------------//-----------------------------------------------------typedef unsigned char byte; //1 байтtypedef unsigned int uWord; //2 байта//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* //=====ГЛАВНАЯ ФУНКЦИЯ=================================//=====================================================void main (void){//-----локальные объекты-------------------------------byte counter=0; //счетчик базовых интерваловconst uWord Tbas=1000;//базовый интервалconst byte Tr=10, //время вкл. красного Tg=12, //время вкл. зеленого Ty=2, //время вкл. желтого Tp=1; //период перекл. зеленого//-----график переключения сигналов светофора----------const byte t1=Tr-Ty, //вкл. желтый  t2=Tr, //выкл. кр. и ж.,вкл. зел. t3=Tr+Tg-6\*Tp, //выкл. зеленый t4=Tr+Tg-5\*Tp, //вкл. зеленый t5=Tr+Tg-4\*Tp, //выкл. зеленый t6=Tr+Tg-3\*Tp, //вкл. зеленый t7=Tr+Tg-2\*Tp, //выкл. зеленый t8=Tr+Tg-Tp, //вкл. зеленый t9=Tr+Tg, //выкл. зел., вкл. желтый t10=Tr+Tg+Ty; //вык. желтый, вкл. кр.//-----настройка портов ввода/вывода-------------------DDRD=0xE0; //7,6 и 5 разряды - на выводPORTD.7=1; //вкл. красный сигнал светофора |

Продолжение листинга 1.6

|  |  |
| --- | --- |
| 3637383940414243444546474849505152535455565758596061626364 | //-----фоновая программа-------------------------------while(1){ //бесконечный циклdelay\_ms(Tbas); //формирование базового интервалаif(++counter==t1) //увеличение счетчика и проверка  PORTD.6=1; //вкл. желтыйelse if(counter==t2) PORTD=1<<5; //выкл. кр. и желтый, вкл. зел.else if(counter==t3) PORTD.5=0; //выкл. зеленыйelse if(counter==t4) PORTD.5=1; //вкл. зеленыйelse if(counter==t5) PORTD.5=0; //выкл. зеленыйelse if(counter==t6) PORTD.5=1; //вкл. зеленыйelse if(counter==t7) PORTD.5=0; //выкл. зеленыйelse if(counter==t8) PORTD.5=1; //вкл. зеленыйelse if(counter==t9) PORTD=1<<6; //выкл. зеленый, вкл. желтыйelse if(counter==t10){ PORTD=1<<7; //выкл. желтый, вкл. красный counter=0; //возобновление цикла работы  }}//while(1)}//END MAIN//===================================================== |

В данном случае, процесс формирование сигналов управления разделяется на десять фаз, названных t1 – t10. Каждая фаза характеризует конечное состояние сигналов светофора ко времени, а шаг автомата определяется значением переменной – счетчика counter, который увеличивается на единицу, после каждого сформированного базового временного интервала. Определение конкретного состояния конечного автомата производится конструкцией множественного выбора, состоящей из последовательности операторов if – else.

Длительность базового временного интервала выбирается как минимальная кратная величина для каждого значения длительности сигнала светофора. Следовательно, значение промежутков времени между переключениями сигналов светофора указывают количество базовых временных интервалов, а не время.

Также в представленном листинге вызывает интерес способ определения типов. В строке 8 при помощи оператора typedef для типа unsigned char объявляется новое имя byte. В строке 9 аналогично для типа unsigned int объявлено имя uWord. Подобное решение не только позволяет сократить количество символов при определении типов объектов, но и позволяет легко корректировать объемы памяти, выделяемые для объектов. Это может быть важно, например, при переносе фрагмента кода между различными компиляторами, т.к. в зависимости от реализации для одних и тех же типов возможно выделение разных объемов памяти. Так для объектов типа int может быть выделено 2, 4 или более байт. Подобный прием позволит внести корректировку путем изменения всего одной команды.

Не смотря на значительное увеличение количества командных строк листинга 1.6 в сравнении с листингом 1.5, последний пример имеет явное преимущество: вызов функции формирования интервалов времени за один цикл обработки состояния светофора выполняется однократно и остается неизменным по продолжительности. Все это в значительной мере упрощает возможность реализации дополнительных параллельно выполняемых задач.

Если внимательно рассмотреть обработчики, выделенные элементами конструкции множественного выбора, то можно выделить некоторые условия, при истинности которых состояния автомата будут совпадать.

Например, можно выделить состояния установившегося зеленого сигнала светофора, приведенные в строках 46 – 47, 50 – 51, 54 – 55, и состояния его отключения, приведенные в строках 44 – 45, 48 – 49, 52 – 53. Получается, что эти состояния могут складываться по совокупности нескольких условий.

Состояния, описываемые в строках 42 – 43 и 56 – 57 нельзя отнести к перечисленным общим случаям, так как в них кроме манипуляций с зеленым сигналом светофора производятся дополнительные действия. И не смотря на то, что в результате выполнения данных обработчиков устанавливаются аналогичные перечисленным случаям состояния светофора (включен зеленый сигнал, желтый и красный – выключены), их объединение было бы идеологически не верным, так как при необходимости расширения функциональности программного кода эти состояния могут быть дополнены различными командами.

Таким образом, можно группировать условия выполнения однотипных состояний при помощи операций логического сложения, что поможет сэкономить рассмотренный код. В листинге 1.7 приводится описание операции множественного выбора с произведенными на основе изложенных мыслей корректировками.

Листинг 1.7 – Обработчик отдельных состояний “светофора”

|  |  |
| --- | --- |
| 40414243 | if(++counter==t1) //увеличение счетчика и проверка  PORTD.6=1; //вкл. желтыйelse if(counter==t2) PORTD=1<<5; //выкл. кр. и желтый, вкл. зел. |

Продолжение листинга 1.7

|  |  |
| --- | --- |
| 4445464748495051525354555657 | else if((counter==t3)|| (counter==t5)|| (counter==t7)) PORTD.5=0; //выкл. зеленый else if((counter==t4)|| (counter==t6)|| (counter==t8)) PORTD.5=1; //вкл. Зеленыйelse if(counter==t9) PORTD=1<<6; //выкл. зеленый, вкл. желтыйelse if(counter==t10){ PORTD=1<<7; //выкл. желтый, вкл. красный counter=0; //возобновление цикла работы  } |

***Вопросы для самоконтроля***

1. Как в листинге 1.4 производится поочередное переключение красного и желтого светодиодов?

2. Как в этом примере производится подсчет переключений сигналов?

3. Какие действия выполняет программа, представленная в листинге 1.5?

4. Как в листинге 1.6 определяется состояние управляющих светодиодами сигналов?

5. Как определяется длительность базового временного интервала?

6. Какими недостатками обладают программы листингов 1.5 и 1.6?

7. При помощи каких команд можно осуществлять манипуляции с отдельными разрядами регистров управления портами микроконтроллера?

***Лабораторная работа 1-4***

***Тема: Генерация импульсного сигнала***

***Цель:*** приобретение практических навыков в составлении и отладке программ генерации управляющих сигналов на языке СИ микроконтроллеров семейства AVR, использовании инструментальных средств разработки аппаратно-программных систем; закрепление знаний теории.

***Требования к содержанию отчета:***

1. Наименование практической работы;

2. Цели и задачи;

3. Ответы на контрольные вопросы;

4. Листинг программы генерации непрерывной последовательности импульсов;

5. Листинг программы генерации пакетов импульсов;

6. Листинг программы “полицейская сирена”;

7. Листинг программы “Светофор”;

8. Вывод о проделанной работе.

Приводимые в отчете листинги программ должны содержать подробные комментарии и соответствовать синтаксису языка программирования.

***Задание:***

1. Ознакомиться с теоретическими сведениями первого раздела методических указаний;

2. Ответить на контрольные вопросы подразделов 1.1, 1.2 и 1.3;

3. Разработать программу генерации непрерывной импульсной последовательности в соответствии с таблицей вариантов 1.1;

4. Разработать программу генерации пакетов импульсов в соответствии с таблицей вариантов 1.2;

5. Разработать листинг программы “полицейская сирена” в соответствии с таблицей вариантов 1.3;

6. Разработать листинг программы “световор” в соответствии с таблицей вариантов 1.4;

7. Произвести отладку разработанных программ с помощью симулятора AVR Studio.

8. Произвести моделирование схемы лабораторного макета под управлением разработанных программ;

9. Продемонстрировать результаты моделирования преподавателю;

10. Загрузить файл прошивки в память целевого микроконтроллера.

Пункты 5 – 10 последовательно повторять для каждой из разработанных программ.

Таблица 1.1 – Генератор импульсов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вариант № | Длительность импульса (мс) | Длительность паузы (мс) | Оператор организации цикла | Цвет светодиода |
| 1 | 200 | 800,5 | for | Красный |
| 2 | 300 | 700,5 | while | Желтый |
| 3 | 400 | 600,5 | do while | Зеленый |
| 4 | 500 | 500,5 | for | Красный |
| 5 | 600 | 400,5 | while | Желтый |
| 6 | 200,5 | 800 | do while | Зеленый |
| 7 | 300,5 | 700 | for | Красный |
| 8 | 400,5 | 600 | while | Желтый |
| 9 | 500,5 | 500 | do while | Зеленый |
| 10 | 600,5 | 400 | for | Красный |
| 11 | 700 | 300,5 | while | Желтый |
| 12 | 800 | 200,5 | do while | Зеленый |
| 13 | 100 | 500,5 | for | Красный |
| 14 | 800,5 | 200 | while | Желтый |
| 15 | 500,5 | 500 | do while | Зеленый |

Таблица 1.2 – Генератор пакетов импульсов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Импульс (мс) | Пауза (мс) | Кол-во импульсов | Кол-во пакетов | Оператор цикла(пакеты, импульсы в пакете) | Цвет светодиода |
| 1 | 200 | 800,5 | 2 | 5 | for, while | Красный |
| 2 | 300 | 700,5 | 3 | 4 | while, for | Желтый |
| 3 | 400 | 600,5 | 4 | 3 | for, while | Зеленый |
| 4 | 500 | 500,5 | 5 | 3 | while, for | Красный |
| 5 | 600 | 400,5 | 2 | 5 | for, while | Желтый |
| 6 | 200,5 | 800 | 3 | 5 | while, for | Зеленый |
| 7 | 300,5 | 700 | 4 | 3 | for, while | Красный |
| 8 | 400,5 | 600 | 5 | 3 | while, for | Желтый |
| 9 | 500,5 | 500 | 2 | 6 | for, while | Зеленый |
| 10 | 600,5 | 400 | 3 | 6 | while, for | Красный |
| 11 | 700 | 300,5 | 4 | 5 | for, while | Желтый |
| 12 | 800 | 200,5 | 5 | 3 | while, for | Зеленый |
| 13 | 100 | 500,5 | 2 | 4 | for, while | Красный |
| 14 | 800,5 | 200 | 3 | 7 | while, for | Желтый |
| 15 | 500,5 | 500 | 4 | 4 | for, while | Зеленый |

Таблица 1.3 – “Полицейская сирена”

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Импульс (мс) | Пауза (мс) | поочеред-ные вкл. | одновре-менные вкл. | Оператор цикла(вместе, по очереди) | Цвета |
| 1 | 200 | 800,5 | 2 | 5 | for, while | Кр. желтый |
| 2 | 300 | 700,5 | 3 | 4 | while, for | Кр. зеленый |
| 3 | 400 | 600,5 | 4 | 3 | for, while | жел. зеленый |
| 4 | 500 | 500,5 | 5 | 3 | while, for | Кр. желтый |
| 5 | 600 | 400,5 | 2 | 5 | for, while | Кр. зеленый |
| 6 | 200,5 | 800 | 3 | 5 | while, for | жел. зеленый |
| 7 | 300,5 | 700 | 4 | 3 | for, while | Кр. желтый |
| 8 | 400,5 | 600 | 5 | 3 | while, for | Кр. зеленый |
| 9 | 500,5 | 500 | 2 | 6 | for, while | жел. зеленый |
| 10 | 600,5 | 400 | 3 | 6 | while, for | Кр. желтый |
| 11 | 700 | 300,5 | 4 | 5 | for, while | Кр. зеленый |
| 12 | 800 | 200,5 | 5 | 3 | while, for | жел. зеленый |
| 13 | 100 | 500,5 | 2 | 4 | for, while | Кр. желтый |
| 14 | 800,5 | 200 | 3 | 7 | while, for | Кр. зеленый |
| 15 | 500,5 | 500 | 4 | 4 | for, while | жел. зеленый |

Таблица 1.4 – “Светофор”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вариант № | Красный (с) | Желтый (с) | Зеленый (с) | Импульс (с) |
| 1 | 10 | 40 | 30 | 1 |
| 2 | 15 | 45 | 3 | 1 |
| 3 | 20 | 50 | 4 | 1 |
| 4 | 25 | 55 | 5 | 1 |
| 5 | 30 | 60 | 2 | 1 |
| 6 | 35 | 10 | 3 | 1 |
| 7 | 40 | 15 | 4 | 1 |
| 8 | 45 | 20 | 5 | 1 |
| 9 | 50 | 25 | 2 | 1 |
| 10 | 55 | 30 | 3 | 1 |
| 11 | 60 | 35 | 4 | 1 |
| 12 | 55 | 40 | 5 | 1 |
| 13 | 50 | 35 | 2 | 1 |
| 14 | 45 | 30 | 3 | 1 |
| 15 | 40 | 50 | 4 | 1 |