**МИНОБРНАУКИ РОССИИ**

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра биомедицинской инженерии

Утверждаю
Проректор по учебной работе
О.Г. Локтионова
«25» 09 2023 г.

# СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ

Методические рекомендации по выполнению лабораторных работ для студентов направления подготовки 12.04.04 – «Биотехнические системы и технологии» (магистр)

Курск 2023

УДК 621.(076.1)

Составители: А.А.Кузьмин

Рецензент:
Кандидат технических наук, доцент *Т.Н. Конаныхина*

Системы автоматизированного проектирования: методические рекомендации по выполнению лабораторных работ для студентов направления подготовки 12.04.04 – «Биотехнические системы и технологии» (магистр) / Юго-Зап. гос. ун-т; сост.: А.А.Кузьмин. - Курск, 2023. - 52 с.

Содержат методические рекомендации к проведению лабораторных работ по дисциплине «Системы автоматизированного проектирования». Методические указания по структуре, содержанию и стилю изложения материала соответствуют методическим и научным требования, предъявляемым к учебным и методическим пособиям.

Предназначены для студентов направления подготовки 12.04.04 – «Биотехнические системы и технологии» (магистр)

Текст печатается в авторской редакции

Подписано в печать 25.09.23 Формат 60х84 1/16
Усо.печ.л. 3,0 . Уч.-изд.л. 2,7 .Тираж 30 экз. Заказ: 1083 .Бесплатно.
Юго-Западный государственный университет.
305040. г. Курск, ул. 50 лет Октября, 94.

# Лабораторная работа №1

## Автоматизированное проектирование приборов и систем с сипользованием САПР PROTEUS

### 1 Краткие теоретические сведения

Система Proteus предназначена для моделирования аналоговых и цифровых электронных схем, в том числе широкой номенклатуры микроконтроллеров. Среди прочих САПР, которые тоже позволяют моделировать электронные схемы, Proteus выгодно отличают мощные возможности отладки программ для микроконтроллеров, а также интерактивного симулирования схем в реальном времени с вводом-выводом информации на реальные физические порты компьютера (COM, USB).

Proteus состоит из двух основных частей:

ISIS – средство для построения принципиальных схем, схемного моделирования и отладки программ микроконтроллеров.

ARES (Advanced Routing and Editing Software) – модуль для проектирования печатных плат.

Рассмотрим основные приемы работы с модулем ISIS. Запустите ISIS. Вид ISIS версии 7 показан на рисунке 1.

Подробнее опишем элементы интерфейса (рисунок 1):

1 - создается новый проект с установками по умолчанию;

2 – загрузка и запись проекта;

3, 4 – загрузка из файла и запись в файл выделенного блока (секции). Операция по действию аналогична кнопке 18;

5 – вывод проекта на печатающее устройство (принтер);

6 – выделяет область, выводимую на принтер;

7 – перерисовка экрана;

8 – включает/выключает сетку;

9 – установка относительного центра координат. Позволяет установить новый центр координат, после чего в панели 36 будут выводиться новые относительные смещения курсора;

10 – установка центра дисплея над позицией курсора;

11 – кнопки изменения масштаба. Аналогичная функция присвоена колесу мыши.

12 – масштаб изображения – весь лист проекта;

13 - масштаб изображения – выделенная область;

14 – кнопки отмены/возврата изменений;

15 – вырезать выделенные элементы в буфер обмена;

16 – скопировать выделенные элементы в буфер обмена;

17 – вставить элементы из буфера обмена;

18 – скопировать выделенный блок;

19 - перенести выделенный блок (доступно также из контекстного меню при выделении объекта – команда Drag Object);

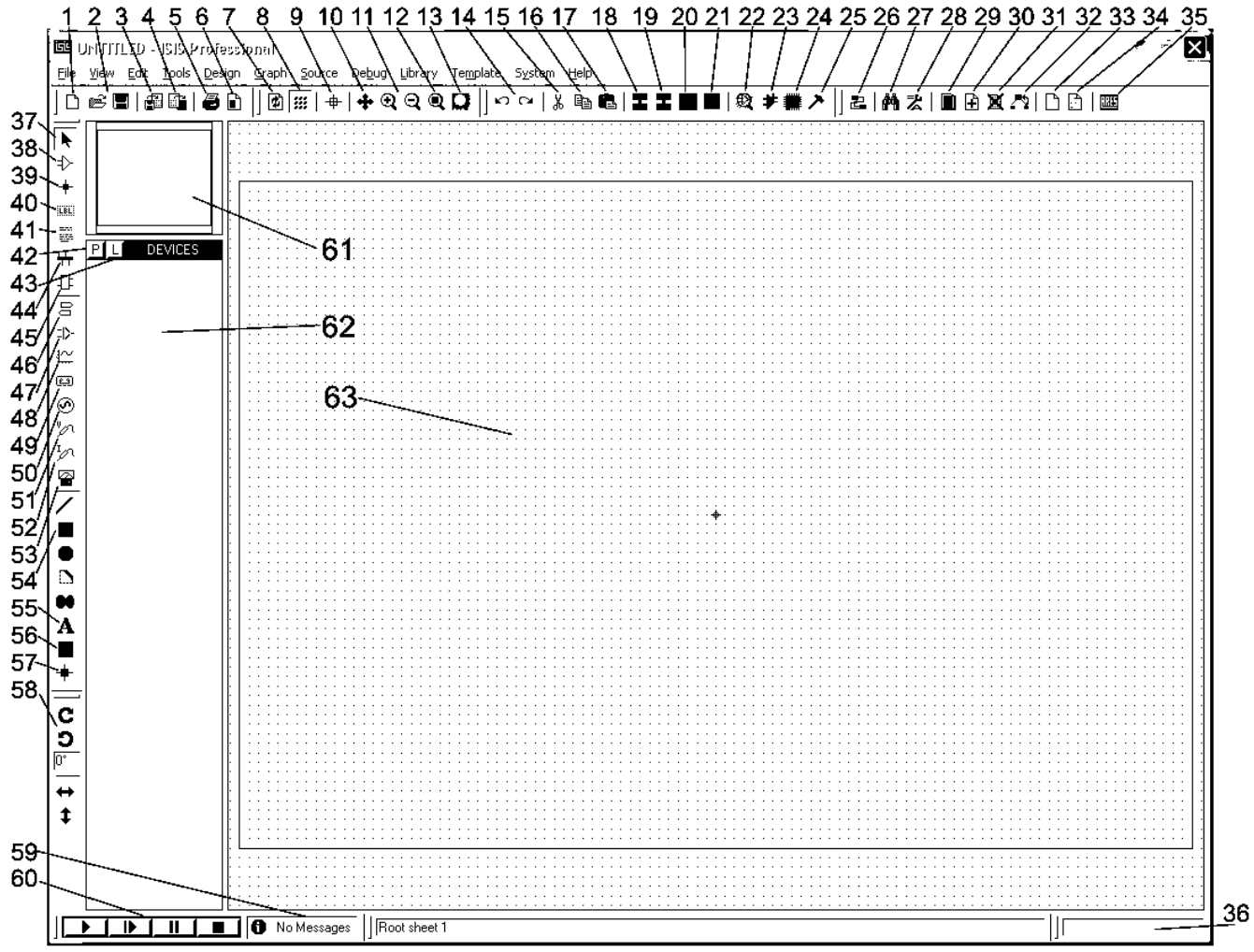


Рисунок 1 - Вид окна ISIS при запуске

20 – повернуть выделенный блок (некоторые варианты поворота доступны также из контекстного меню при выделении объекта);

21 - удалить выделенный блок (доступно также из контекстного меню при выделении объекта – команда Delete Object, или повторное нажатие правой клавиши мышки над выделенным объектом);

22 – выбор схемотехнического элемента, компонента, устройства. Аналог кнопке 42;

23 – вызывает диалог (мастер) объединения выделенной схемы в отдельный компонент и сохранения компонента в библиотеку.

24 - вызывает диалог установки корпуса компонента;

25 – разбивает выделенный компонент на примитивы;

26 – включает/выключает автороутер для проводников (wire autorouter). При включенном автороутере проводники прокладываются автоматически под прямыми углами. Для кратковременного отключения автороутера при прокладывании проводника (например, для диагонального соединения двух точек) нажмите клавишу CTRL;

27 – поиск компонентов, свойства которых удовлетворяют заданному критерию;

28 – вызов диалога присваивания значений свойствам компонентов;

29 – представление проекта в табличном виде для исследования списка элементов, соединений и навигации по ним.

30 – создание нового листа проекта. Перемещение между листами осуществляется клавишами PageUp и PageDown.

31 - удаление листа проекта;

32 – возврат к родительскому листу проекта;

33 – создание отчета-списка используемых материалов (Bill Of Materials);

34 – проверка правил электрических соединений;

35 – создание списка соединений и запуск ARES для дальнейшего проектирования печатной платы;

36 – панель вывода относительных координат;

37 – режим выбора объектов;

38 – режим размещения компонентов и проводников;

39 – установка соединений для х-образных пересечений проводов. По умолчанию соединяющимися считаются только т-образные пересечения проводов.

40 – установка метки на провод. Провода с одинаковыми метками считаются соединенными.

41 – ввод текстовых скриптов;

42 – выбор схемотехнического элемента, компонента, устройства. Аналог кнопке 22;

43 – менеджер подключаемых библиотек;

44 – ввод шин;

45 – размещение линий ввода-вывода составных схем (subcircuit);

46 – размещение внутрисхемных соединителей (terminals), в том числе и таких как земля, питание и др.

47 – размещение выводов компонента;

48 – размещение аналитических графиков (переходных процессов, частотного анализа, цифровых диаграмм и т.п.);

49 – установка «магнитофона» (tape recorder);

50 – размещение генераторов сигналов (постоянного тока, переменного, импульсного, из звукового файла и т.п.);

51 – установка пробника напряжения;

52 – установка пробника тока;

53 – установка интерактивных инструментов (осциллографа, генератора сигналов, логического анализатора, виртуального терминала и т.п.);

54 – размещение двумерных графических примитивов (линии, прямоугольника, окружности, дуги, многоугольника);

55 – размещение произвольного текста;

56 – размещение графического символа из библиотеки;

57 – установка различных маркеров при проектировании элементов;

58 – элементы вращения и отражения объекта, подлежащего размещению;

59 – информационная панель предупреждений и ошибок;

60 – панель кнопок запуска/отладки/остановки интерактивной симуляции.

61 – предварительный просмотр макета страницы или компонента;

62 – список используемых компонентов;

63 – рабочая область.

Пусть в системе Proteus требуется смоделировать гирлянду из восьми светодиодов.

Для управления светодиодами будем применять микроконтроллер PIC16F877. Это достаточно функциональный микроконтроллер среднего семейства фирмы Microchip. Все последующие устройства и примеры программ, а также лабораторные макеты будут использовать именно этот микроконтроллер.

Составим необходимый список элементов. Для этого щелкаем на кнопку 42. Получаем диалог поиска элемента (рисунок 2).

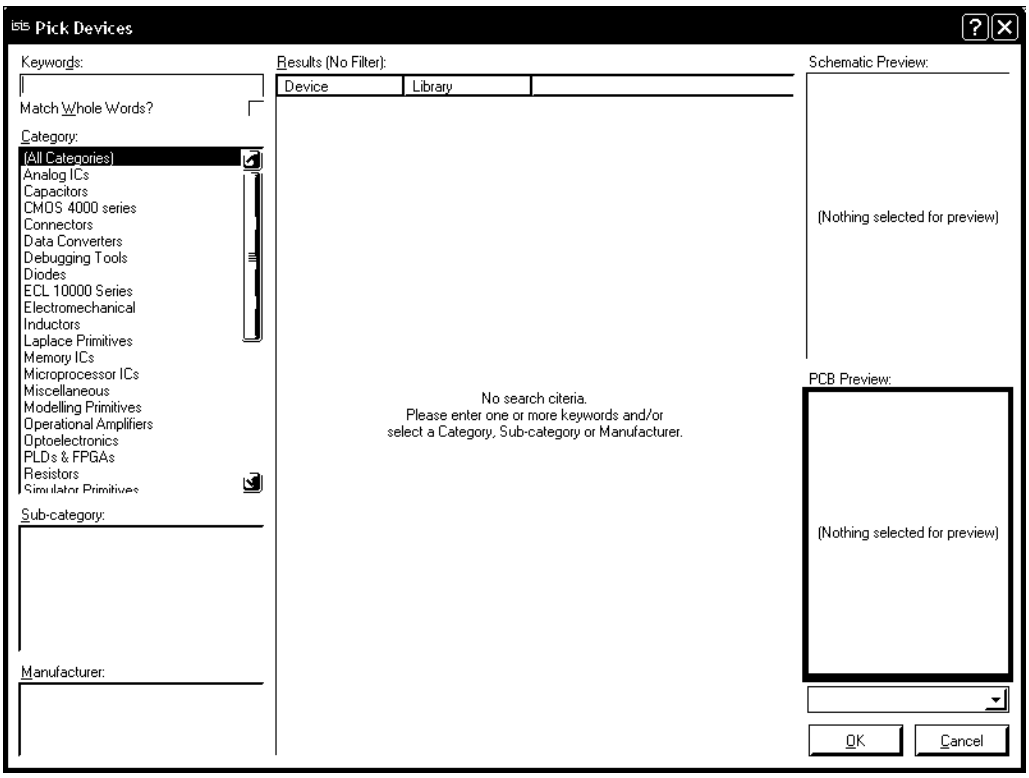


Рисунок 2 – Диалог поиска элемента

В этом диалоге поле «Keywords» необходимо для ввода ключа поиска, список Category позволяет выбрать определенную группу (категорию) компонентов, Manufacturer – производителя. В списке Results отображаются результаты поиска, в картинке Schematic Preview – условное графическое отображение компонента, а также информация о модели компонента или об ее отсутствии, PCB Preview – вид корпуса и контактных площадок.

В поле Keywords вводим ключевые слова для поиска компонента. Для микроконтроллера ключом будет текст «PIC16F877», для светодиода «led-red», для ограничительных резисторов «res». Двойным щелчком по необходимому элементу списка Results добавляем компоненты в список используемых элементов, который отображается в поле 62.

Закрываем диалог поиска элемента и, выбирая необходимые компоненты из сформированного списка используемых элементов, перемещаем их на рабочую область, примерно как показано на рисунке 3. При этом желательно включить режим автоматического назначения позиционных обозначений (пункт меню Tools\Real Time Annotation должен быть включен), иначе позиционные обозначения придется расставлять вручную. Символ схемной земли (GROUND) берется из списка Terminals (кнопка 46).

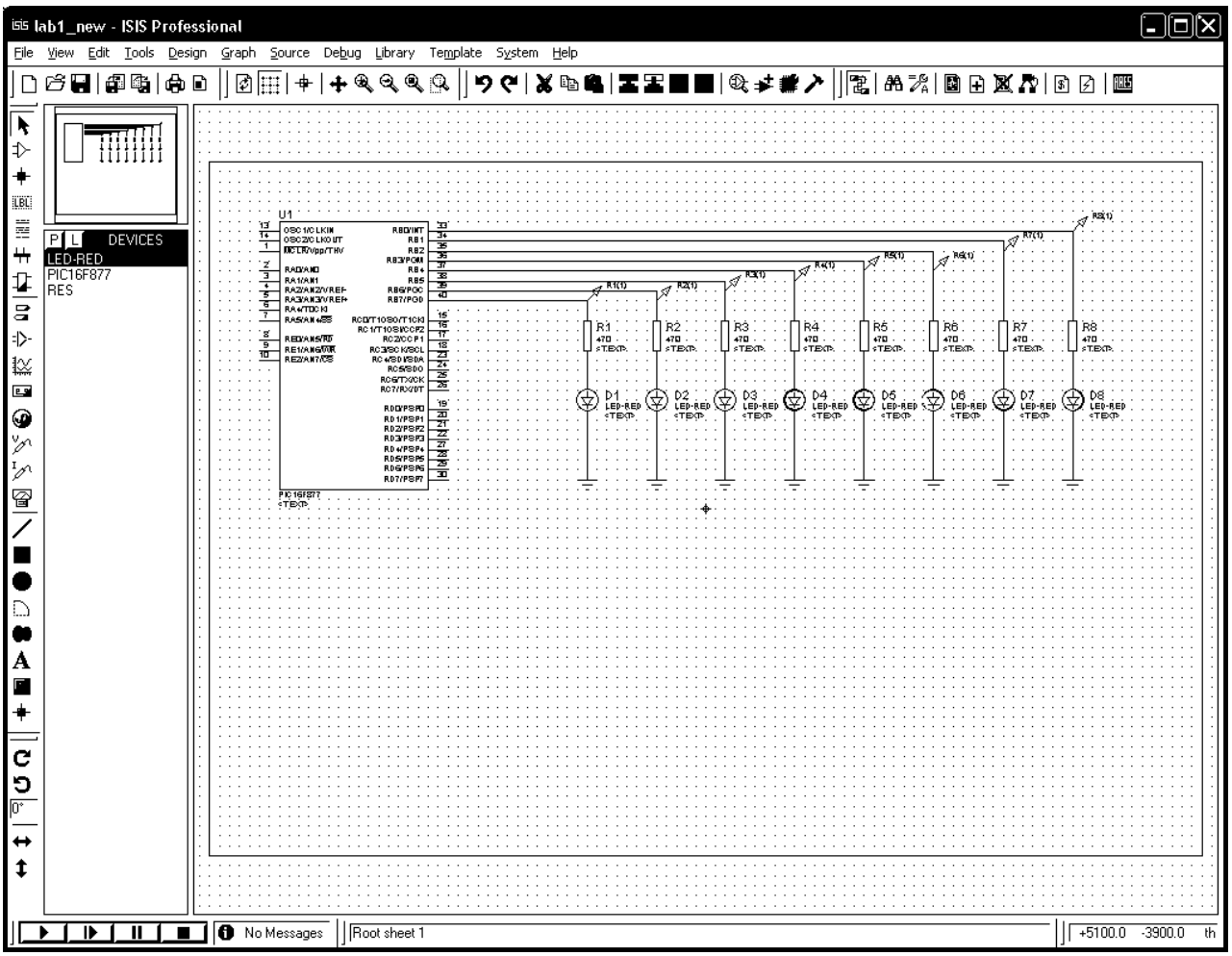


Рисунок 3 – Схема гирлянды в Proteus

Соединяем между собой элементы проводниками (режим проведения проводника включается по умолчанию – курсор в форме карандаша, а при наведении курсора мыши на вывод элемента карандаш окрашивается в зеленый цвет). При этом если имеются два вывода элементов, то между ними обязательно нужно проложить проводник, чтобы они считались соединенными. То есть нельзя соединить, допустим, светодиод и резистор простым перемещением одного из элементов и визуальным совмещением выводов этих элементов. Необходимо отдельно в рабочую область поместить светодиод, отдельно резистор и соединить их выводы проводом.

Затем можно заменить номиналы по умолчанию всех резисторов (10к) на значение 470, или даже еще меньше, иначе яркость свечения светодиодов будет неудовлетворительной. Для этого двойным щелчком по надписи «10k» рядом с резистором открываем диалог редактирования свойства и заменяем номинал резистора на 470 (Ом). Указание единиц без суффиксов устанавливает значение параметра, выраженное в единицах С, т.е. в вольтах, амперах, омах, фарадах. Если необходимо установить значение в кило- или мега- единицах, то после цифрового значения указыва-

ют суффикс «k» или “M”. Если необходимо установить значение в мили-, или микро-, или нано-, или пико-единицах, то после цифрового значения указывают соответственно суффикс «m», «u», «n», «p». Обратите внимание на различие в значении строчного или заглавного написания суффикса в мили- и мега-единицах!

Отредактировать все свойства элемента можно путем выбора из контекстного меню пункта Edit Properties. Или выделить элемент щелчком мыши и нажать Ctrl+E.

Далее необходимо задать программу микроконтроллеру. Обычно исходный текст программы для микроконтроллера пишется или на языке ассемблера, или на языке С. Затем необходимый компилятор транслирует программу в машинные коды и записывает результат в файл с расширением hex.

Создадим в текстовом файле Lab1.asm следующую программу на языке ассемблера:

```
    ;Программа лабораторной работы №1
    ;Операции      ввода-вывода      микроконтроллеров
PIC16F877
    ;Назначение - включение/выключение светодиодов
    ;Схема включения - светодиоды на PORTB
    ;Автор программы =Кузьмин А.А.

    list   p=16f877
    #include p16f877.inc
    ;-----------------------------------------------
-
    ;Определение переменных
    temp EQU     0x21  ;Регистр быстрой задержки
    cnt1 EQU     0x22  ;Регистр1 длинной задержки
    cnt2 EQU     0x23  ;Регистр2 длинной задержки
    cnt3 EQU     0x24  ;Регистр3 длинной задержки

    ;-----------------------------------------------
-
    org 0x00
    nop            ;Первая операция – nop
                ; для внутрисхемных отладчиков
    goto start     ;"Стандартное" начало
```

```
            ;(для программ без прерываний
    org 0x8     ;можно и пропустить этот оператор)

    start
    ;-------Инициализация-----------------------------
-
    bsf STATUS,RP0     ;Регистр TRISB не в
                  ;нулевой странице!
    movlw 0x00     ;все выводы PORTB как выходы
    movwf TRISB
    bcf STATUS,RP0 ;Возвращаемся к нулевой странице

    ;------Главный цикл-----------------------------

    loop
    movlw 0x55     ;01010101 в PORTB
    movwf PORTB
    call delay     ;задержка

    movlw 0xaa     ;10101010 в PORTB
    movwf PORTB
    call delay     ;задержка

    goto loop
    ;-----------------------------------------------
-
    ;-----Подпрограммы-------------------------------
-
    ;-----------------------------------------------
    ;Подпрограмма задержки
    ;Организуется программная задержка
    ;в один цикл loopdelay2
    ;Используемые регистры: temp

    delay2
     movlw 0xff
     movwf temp
    loopdelay2
     decfsz temp,f
     goto loopdelay2
```

```
     return
    ;-----------------------------------------------
-
    ;Подпрограмма задержки. Версия 2
    ;Задержка для этой подпрограммы задается
    ;перед обращением
    ; к ней как загрузка числа в регистр cnt1.
    ;Общее число циклов определяется по формуле:
    ; 256*256*cnt1*7.
    ;Для примера, приблизительно 1 секунда
    ;при частоте 20 МГц задается загрузкой 11 в cnt1.

    ;Название: delay
    ;Входные данные: число, пропорциональное задержке
    ;в регистре cnt1.
    ;Выходные: задержка.
    ;Используемые регистры: cnt1, cnt2, cnt3.

    ; cnt1 = Cycles/256/256/7 =
    ;(20000000/4)/256/256/7 = 10.899 = 11

    delay           movlw .11
                    movwf cnt1
                    clrf cnt2
                    clrf cnt3
    dloop           decfsz cnt3,f
                    goto $+2
                    decfsz cnt2,f
                    goto $+2
                    decfsz cnt1,f
                    goto dloop
                    return
    End
```

Несколько заметок к этому листингу. Строки, начиная от символа «;» и до конца строки являются комментариями. На работоспособность контроллера они не влияют, но комментарии очень рекомендуются для описания ключевых элементов программ, таких как название, назначение, авторство и т.д., как целой программы, так и отдельных подпрограмм или модулей программы.

Необходимо внимательно вводить пробелы в командах, например в директиве «list p=16f877» между символами t и p пробел обязателен. Также необходимо отличать буквы «O» от символа нуля «0», например, в обозначении бита «RP0» в конце стоит именно символ нуля.

Сохраним проект (кнопка 2 Save Design). В пути проекта не рекомендуется использовать нелатинские символы и пробелы. Не рекомендуется сохранять проект на рабочий стол-это может повлиять на работоспособность симулятора проекта. Поместим файл Lab1.asm в тот же каталог, куда мы сохранили проект. Далее для формирования hex-файла кликнем на пункт меню Source\Add Remove Source files. Вызывается диалог редактирования свойств исходных программных файлов (рисунок 4).

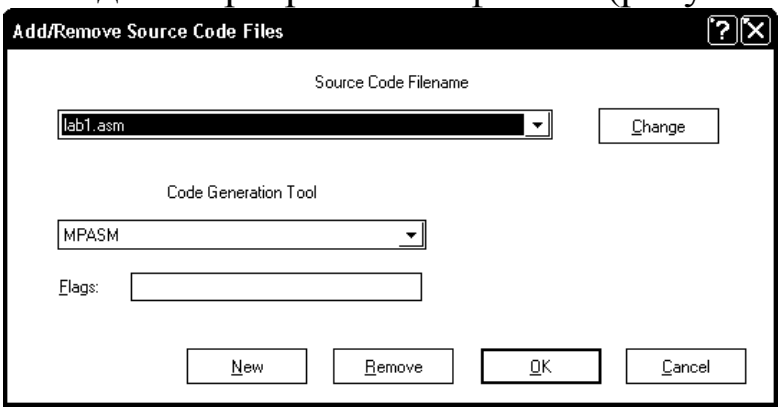


Рисунок 4 - Диалог редактирования свойств исходных программных файлов

В поле Code Generation Tool выбираем ассемблер фирмы Microchip – MPASMWIN, нажимаем кнопку New. В открывшемся диалоге выбора файлов выбираем файл Lab1.asm. Теперь при каждом запуске симуляции исходная программа на ассемблере будет компилироваться выбранным компилятором MPASM в файл Lab1.hex.

Полученный hex файл можно записывать непосредственно в микросхему микроконтроллера с помощью программатора или внутрисхемного отладчика. Для прошивки микроконтроллера нашего проекта необходимо в свойствах компонента PIC16F877 (диалог редактирования свойств, как отмечалось выше, вызывается выбором мышкой компонента и нажатием на Ctrl+E) в поле Program File задать Lab1.hex (рисунок 5). Тут же задаем частоту работы микроконтроллера Processor Clock Frequency в 20 МГц (именно на такую частоту рассчитана секундная задержка в подпрограмме delay). К микроконтроллеру можно присоединить частотозадающее цепи, такие как кварц, RC-цепочку, генераторы и т.д., однако модель по соображениям эффективности работы будет ориентироваться только на частоту, веденную в этом диалоге.

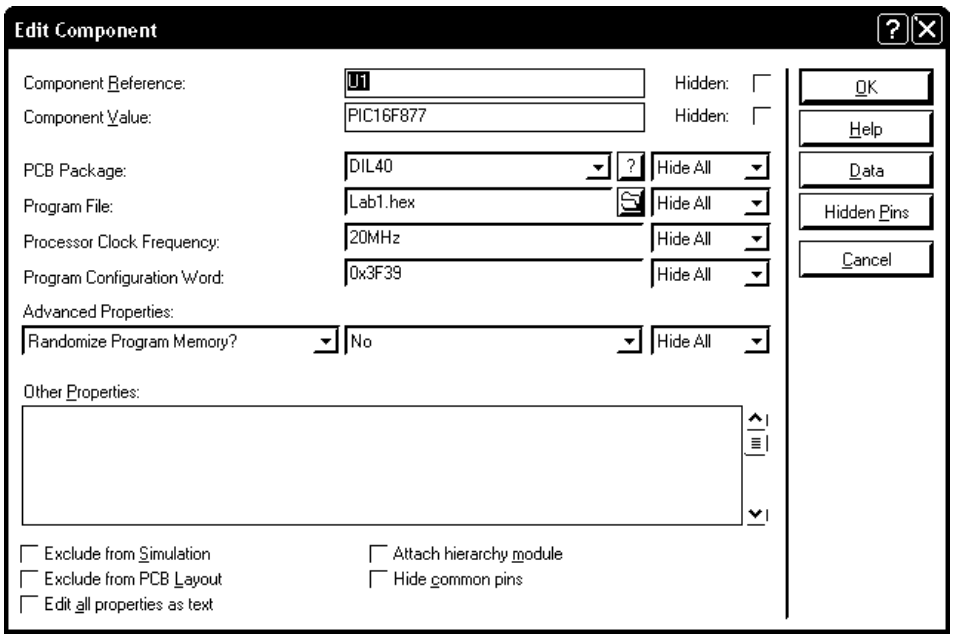


Рисунок 5 – Окно редактирования свойств микроконтроллера

Теперь можно запускать симуляцию кнопкой Play на панели 60. Если все сделано правильно, и нет никаких ошибок, то гирлянда с частотой примерно раз в секунду будет переключать группы зажженных светодиодов.

Изменим эти группы. Для этого в исходном файле с помощью любого текстового редактора заменим в строках

```
.....
movlw 0x55      ;01010101 в PORTB
.....
movlw 0xaa      ;10101010 в PORTB
.....
```

числа на другие значения (например, 0x0 и 0xff). В Proteus тоже есть редактор кода, однако пользоваться им очень неудобно, так как он неправильно отражает русскую кодировку.

Остановим симуляцию (кнопка Stop the simulation панели 60). Сохраним программу. Запускаем симуляцию. Теперь группы светодиодов, которые зажигаются в тот или иной момент времени, поменялись!

При симуляции логические уровни на проводниках отмечаются цветными маркерами в виде квадратов. Если маркер красного цвета, то в данной цепи уровень напряжения, соответствующий логической единице, если маркер синего цвета – то логическому нулю, если маркер серого цвета – то вывод находится в высокоимпедансном третьем (Z) состоянии, если маркер желтого цвета – то на данном проводнике наблюдается кон-

фликт, т.е. одновременно выводится и уровень логической единицы, и уровень логического нуля.

Одним из важнейших режимов симуляции является отладка программ. Управление отладкой программ производится в группе меню Debug. Выберем пункт меню Debug\Start Restart Debugging или нажмем Ctrl+F12. Должно появиться окно исходного кода программы PIC CPU Source Code (рисунок 6). Если исходная программа на ассемблере задана, а окно не появляется автоматически, то необходимо щелкнуть по пункту меню Debug\ PIC CPU Source Code.

В окне исходного кода программы следующие элементы управления:

1 – текущая команда, которую выполняет микроконтроллер – отмечается красным треугольником, и выделенная команда – отмечается синей полосой;

2 – поле для установки точек прерывания (точек останова, breakpoints). Точки прерывания устанавливаются нажатием на кнопку 9 или двойным щелчком мыши на этом поле и отмечаются красными кружками;

3 – файл исходной программы со служебной информацией;

4 – кнопка запуска симуляции (до ближайшей точки останова);

5 – кнопка пошагового выполнения программы без захода в процедуры (процедуры вызываются командой call);

6 - кнопка пошагового выполнения программы с заходом в процедуры;

7 – кнопка выполнения программы до выхода из текущей процедуры;

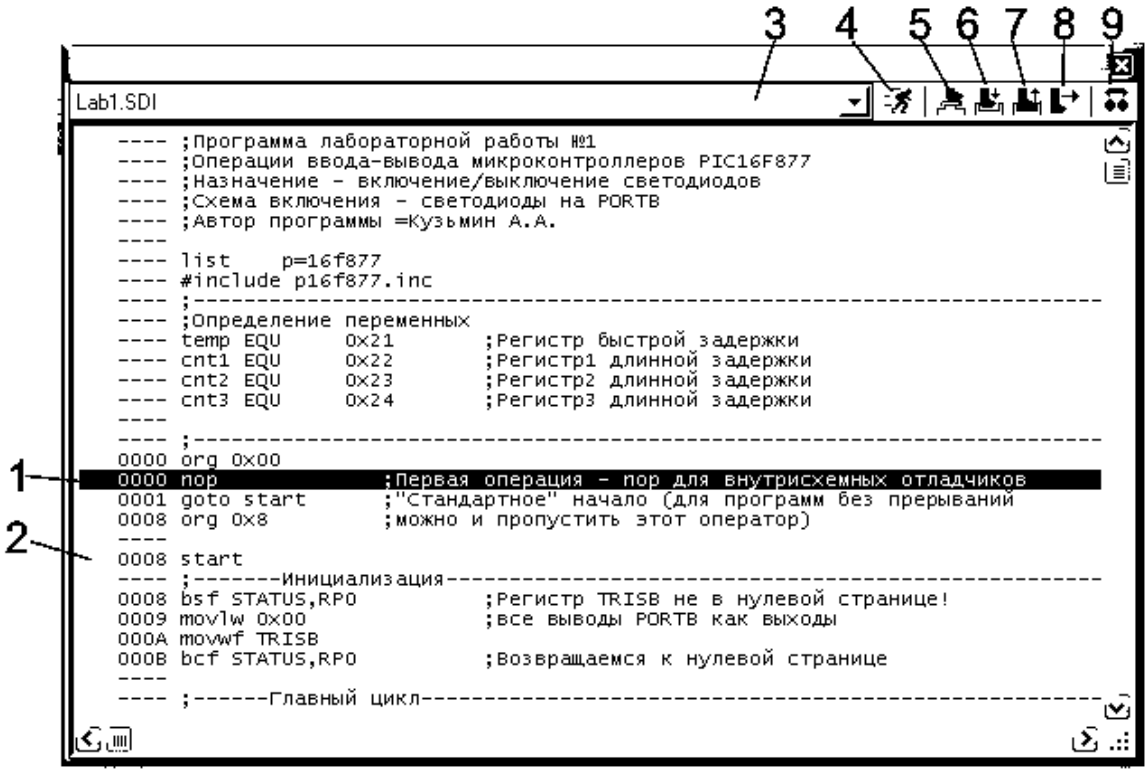


Рисунок 6 - Окно исходного кода программы PIC CPU Source Code

8 – кнопка выполнения программы до выделенной синей полосой команды;

9 – установка/снятие точки прерывания.

Очень важная информация для отладки программ также содержится в окнах PIC CPU Registers – в этом окне содержится информация о наиболее важных регистрах микроконтроллера, PIC CPU Data Memory – в этом окне содержится информация о всех регистрах данных микроконтроллера, PIC CPU EPROM Memory - в этом окне содержится информация о состоянии EEPROM памяти микроконтроллера, PIC CPU Program Memory - в этом окне содержится информация о состоянии памяти программ микроконтроллера, PIC CPU Stack – информация о стеке. Все эти окна можно открыть из группы меню Debug.

Для удобства наблюдения за состоянием определенного регистра существует окно Watch Window. Добавим, например, в окно Watch Window ссылку на регистр1 длинной задержки из нашей программы, который определяется в программе как

cnt1 EQU 0x22 ;Регистр1 длинной задержки.

Для этого открываем окно из пункта меню Debug\ Watch Window. В контекстном меню этого окна выбираем пункт Add Items (By Adress). Получаем окно ввода регистра по адресу (рисунок 7).

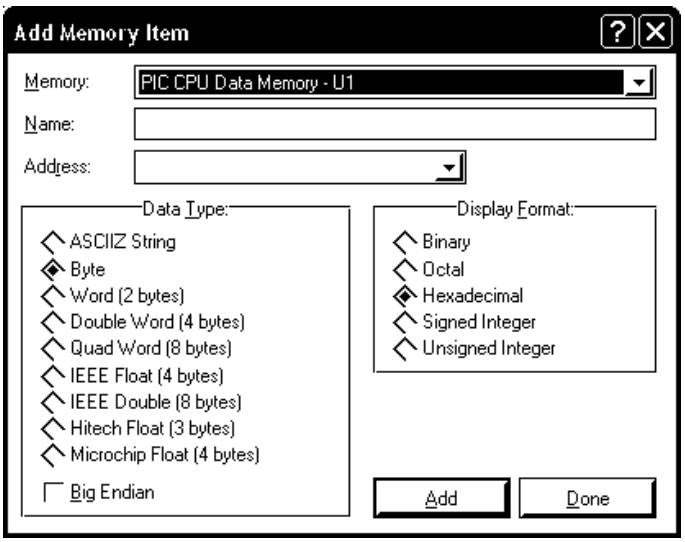

Рисунок 7 - Окно ввода регистра по адресу

В этом диалоге можем выбрать формат данных, формат представления, вид памяти, откуда будет выбираться значение, символическое имя переменной и адрес переменной. Оставим настройки вывода и вид памяти по умолчанию, в имени переменной введем cnt1, а в адресе 0x22. Нажимаем кнопку Add и Done. В результате в окне Watch Window появится информация о состоянии этой переменной.

Важнейшим инструментом отладки является условная точка остановки, т.е. остановка при выполнении определенного условия. Условную точку остановки при, например, равенстве нашего регистра cnt1 допустим двойке, можно поставить в окне Watch Window путем выбора пункта контекстного меню Watchpoint Condition (рисунок 8).

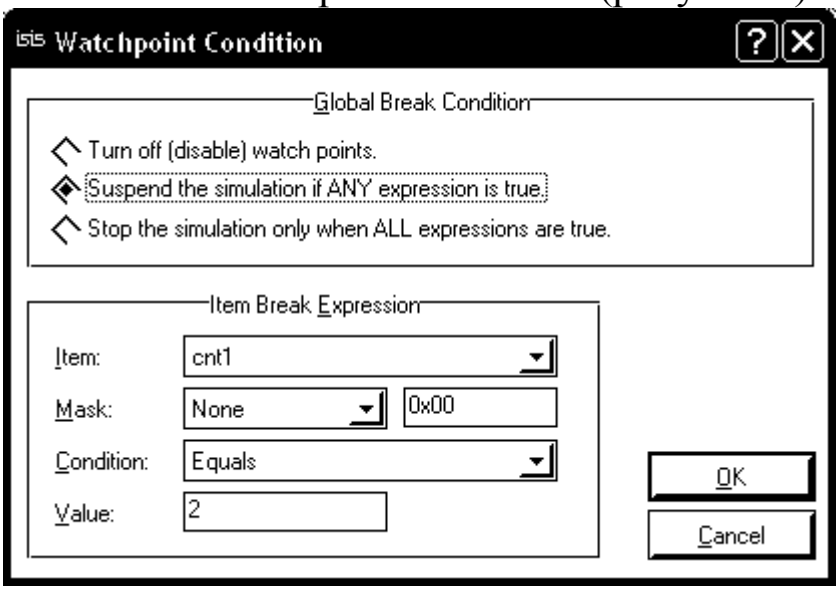

Рисунок 8 – Установка условной точки остановки

Выбираем в этом диалоге следующие настройки: приостановить симуляцию, если условие истинно (Suspend the simulation if ANY expression

is true), проверяемый регистр (Item) – cnt1, условие остановки (Condition) – равенство (Equals), значение остановки (Value) – 2.

Запускаем симуляцию. При равенстве содержимого регистра cnt1 двум, программа остановится.

Продемонстрируем, как с помощью Proteus можно исследовать переходные процессы в схемах и цифровые диаграммы. Для демонстрации исследования цифровых процессов вполне хватит созданной нами гирлянды. Добавим на выводы микроконтроллера, которые соединены с резисторами, пробники (кнопка 51). Если пробники помещать ближе к резисторам, то их названия автоматически присваиваются следующим значениям: R1(1), R2(1)… R8(1). Если пробники помещать ближе к выводам микроконтроллера, то названия у них будут другие: U1(RB0/INT), U1(RB1) и т.д., но смысл пробника останется тем же – он будет показывать напряжение на этом проводе во время симуляции. Кроме того, на цепь, помеченную пробником, легче ссылаться во время анализа переходных процессов. Поместим в рабочую область цифровую диаграмму путем выбора графика DIGITAL из набора графиков для анализа (кнопка 48). Распахнем на весь экран график (пункт Maximize контекстного меню). Получим диалог расчета цифровых диаграмм DIGITAL ANALYSIS (рисунок 9).

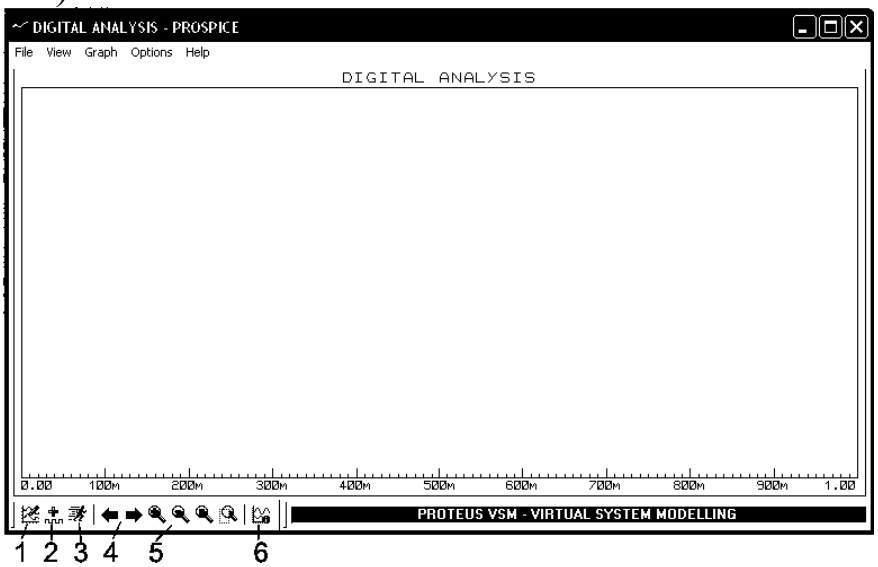


Рисунок 9 – Диалог расчета цифровых диаграмм DIGITAL ANALYSIS

Основные элементы управления этого диалога:

1 – Редактирование общих свойств графика, таких как заголовок, названия осей, начального и конечного времени, а также свойств программы-вычислителя диаграмм Spice. По умолчанию расчет ведется до одной секунды. Так как у нас задержка в переключении диодов рассчита-

на тоже примерно на одну секунду, то для наблюдения переходного процесса поставим значение Stop Time в 2 секунды.

2 – Добавляет пробники для расчета – в диалоге выбирается напряжение на каких пробниках будет добавляться в рассчитанные цифровые диаграммы. Выберем здесь созданные ранее нами пробники R1(1), R2(1)… R8(1) (или U1(RB0/INT), U1(RB1) и т.д.).

3 – Запуск симуляции. Рассчитываются цифровые диаграммы выбранных пробников. Результат нашей симуляции показан на рисунке 10.

4 – Кнопки навигации по графикам по оси времени.

5 – Кнопки выбора масштаба изображения.

6 – Просмотр системных сообщений, выдаваемых во время расчета графиков.

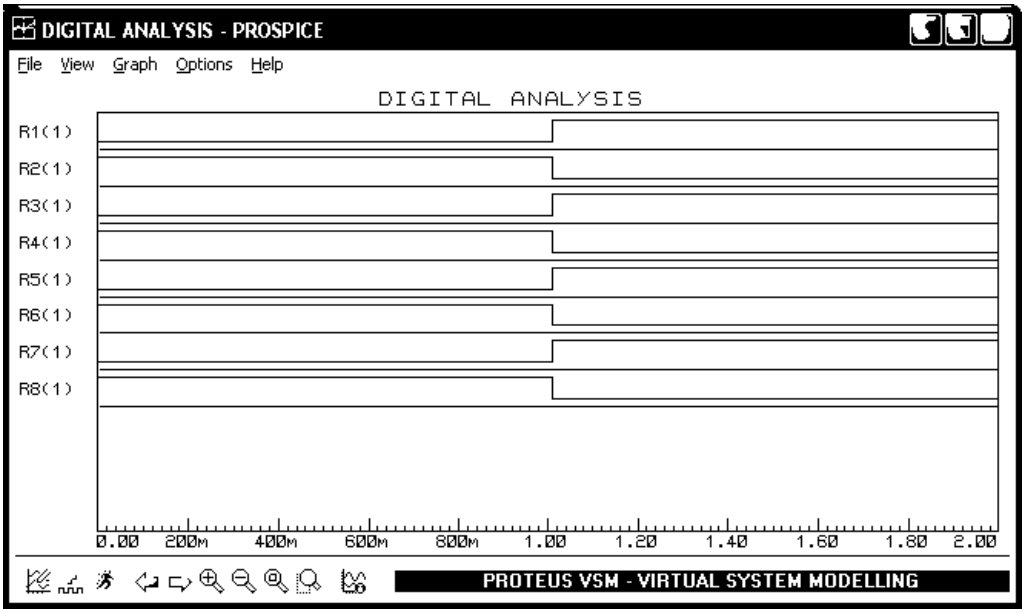


Рисунок 10 - Результат расчета цифровых диаграмм гирлянды

Аналогично происходит и расчет аналоговых переходных процессов (график ANALOGUE).

2. Цель работы

Целью работы является приобретение базовых навыков в моделировании электронных схем с использованием программы Proteus.

3. Порядок выполнения работы

3.1 Собрать схему, согласно выданному варианту.

3.2 Задать соответствующую программу микроконтроллеру.

3.3 Промоделировать собранную схему с использованием отладчика, точек останова, условных точек останова.

3.4 Снять осциллограммы с контрольных точек.

4. Содержание отчета:

Титульный лист с названием и номером работы, а также с фамилиями исполнителей;

Цель работы;

Задание на лабораторную работу;

Снимок экрана (скриншот) системы Proteus с программой, остановленной на точке останова;

Осциллограммы сигналов в контрольных точках;

Выводы.

5 Контрольные вопросы

5.1 Основные элементы интерфейса оболочки ISIS.

5.2 Приемы поиска необходимых элементов в оболочке ISIS.

5.3 Какие электронные компоненты Proteus вы знаете?

5.4 Как поместить на разрабатываемую схему символ земли GROUND?

5.5 Как вводятся соединяющиеся и пересекающиеся проводники?

5.6 Как изменяются номиналы простейших аналоговых компонентов?

5.7 Какие применяются суффиксы для модификации цифровых значений параметров компонентов?

5.8 Почему при изменении номинала ограничивающего резистора изменяется яркость свечения светодиода?

5.9 Как меняются свойства компонентов в Proteus?

5.10 Как микроконтроллерам задаются программы, по которым они работают?

5.11 Как задается частота, на которой работает микроконтроллер?

5.12 Какие ошибки могут возникнуть при запуске симуляции схемы?

5.13 Что обозначают цветные квадраты рядом с проводниками во время симуляции?

5.14 Какими элементами интерфейса управляется процесс отладки программ?

5.15 Как при пошаговой отладке отрабатывается выполнение процедур?

5.16 Как установить и снять точку останова?

5.17 Какие дополнительные отладочные окна поддерживает система Proteus?

5.18 Что такое условная точка остановки и как ее установить в Proteus?

5.19 Зачем нужны пробники напряжения в Proteus?

5.20 Как происходит расчет графиков переходных процессов (цифровых диаграмм) ?

5.21 Какое различие между цифровыми и аналоговыми графиками переходных процессов?

## Лабораторная работа №2
## Автоматизированное проектирование приборов и систем с использованием САПР SPICE

Целью данной лабораторной работы является изучение особенностей построения моделей элементов в системе Proteus, имитационная часть которых основана на SPICE-моделях.

1 Краткие сведения о содержании работы

Пусть необходимо построить модель микросхемы AD633 фирмы Analog Devices. Из документации, которую можно скачать с сайта производителя (www.analog.com), узнаем, что AD633 – это законченный четырехквадрантный аналоговый умножитель (рисунок 2.1).

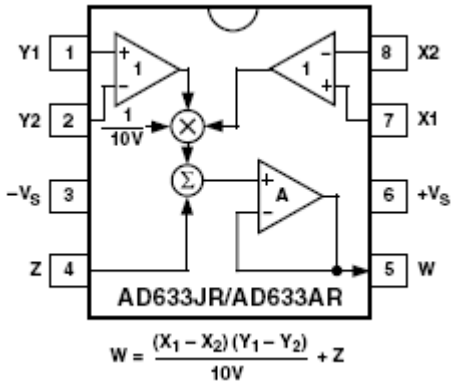


Рисунок 2.1 – Структура умножителя AD633

На сайте производителя также можно найти SPICE модель микросхемы «AD633.cir». SPICE-модель представляет собой текстовый файл, в котором на языке SPICE описаны параметры и соединения лежащих в основе модели цепей. Например, модель AD633 определяется следующим образом (приведен только фрагмент модели):

```
* Copyright 1993 by Analog Devices, Inc.
*
* Refer to "README.DOC" file for License Statement.  Use of this model
* indicates your acceptance with the terms and provisions in the License Statement.
*
* Node assignments
*              X1
*              |  X2
*              |  |  Y1
*              |  |  |  Y2
*              |  |  |  |  VNEG
*              |  |  |  |  |  Z
*              |  |  |  |  |  |  W
*              |  |  |  |  |  |  |  VPOS
*              |  |  |  |  |  |  |  |
.SUBCKT AD633 1  2  3  4  5  6  7  8
*
EREF 100 0 POLY(2) 8 0 5 0 (0,0.5,0.5)
*
* X-INPUT STAGE & POLE AT 15 MHz
*
IBX1 1 0 DC 8E-7
IBX2 2 0 DC 8E-7
EOSX 10 1 POLY(1) (16,100) (5E-3,1)
RX1A 10 11 5E6
RX1B 11 2 5E6
……
GO2 5 72 69 5 11.623E-3
RO2 72 5 86
LO 72 7 1E-7
*
* MODELS USED
*
.MODEL QX NPN(BF=1E4)
.MODEL DX D(IS=1E-15)
.ENDS AD633
```

В этом текстовом файле в строке «.SUBCKT AD633 1 2 3 4 5 6 7 8» указывается после ключевого слова SUBCKT (от «Sub-circuit» - вложенная цепь) наименование модели и порядок расположения выводов. Интерпретация выводов, как правило, указывается в комментарии. Например, первый вывод интерпретируется как вход X1, второй вывод – как вход X2 и т.д. Порядок расположения выводов очень важен для построения будущей модели Proteus.

Далее как в лабораторной работе 1 строится условное графическое отображение модели, причем выводы назовем следующим образом: четыре основных входа – как X1, X2, Y1, Y2, вход коррекции выходного уровня нуля – Z, выход – W, напряжение питания – V+ и V-. Компоновка модели происходит аналогично тому, как это было сделано в лабораторной работе 1. Однако при компоновке необходимо дополнительно определить три свойства модели (рисунок 2.2).

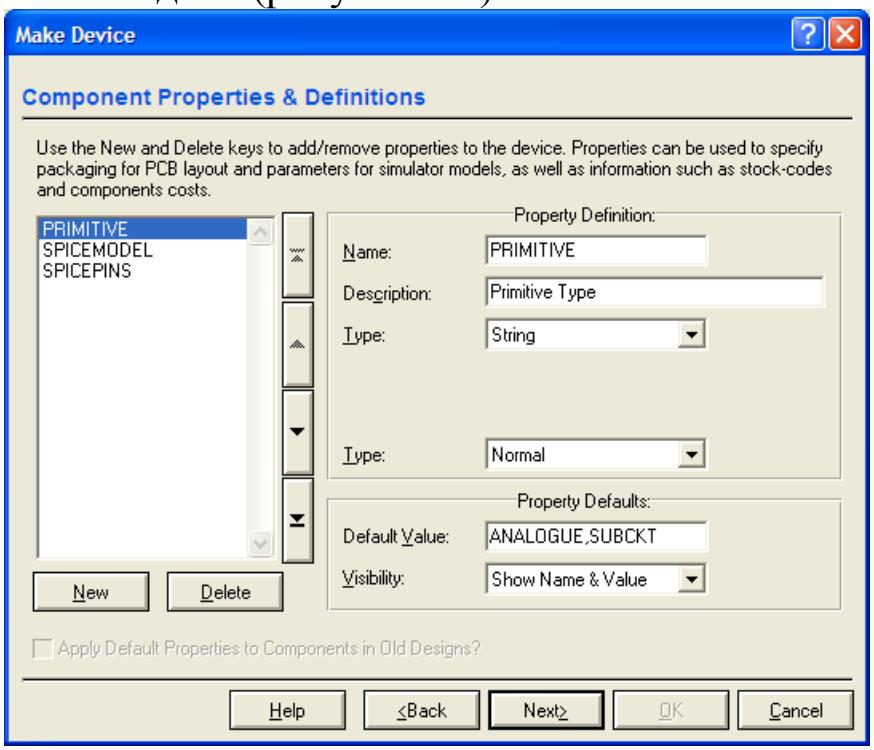


Рисунок 2.2 – Определение свойства PRIMITIVE модели

В диалоге задания свойств по умолчанию модели, кнопка New создает новое свойство; в поле Name определяется имя свойства; в поле Description задается краткое описание свойства; в поле Type задается тип свойства (в нашем случае это строка – String); во втором поле Type задаются разрешения на изменение свойства (нормальное или, например, только для чтения и т.п.); в поле Default Value указывается значение по умолчанию свойства; в поле Visibility определяются параметры вывода на экран свойства.

Первое свойство «PRIMITIVE» должно указывать Proteus, что модель будет аналоговой и задаваться вложенной цепью. Для этой цели значение по умолчанию свойства (Default Value) должно быть равным «ANALOGUE,SUBCKT». Второе свойство «SPICEMODEL» должно указывать Proteus на имя файла модели, и на наименование модели внутри этого файла (например, «AD633,AD633.CIR»). Третье свойство «SPICEPINS» ставит в соответствие список выводов модели Proteus списку выводов SPICE-модели. Список выводов SPICE-модели AD633 приведен выше – это «X1,X2,Y1,Y2,VNEG,Z,W,VPOS». В наименованиях наших выводов этот список выглядит следующим образом: «x1,x2,y1,y2,V-,z,w,V+». Именно эта строка должна содержаться в значении свойства «SPICEPINS» (рисунок 3).

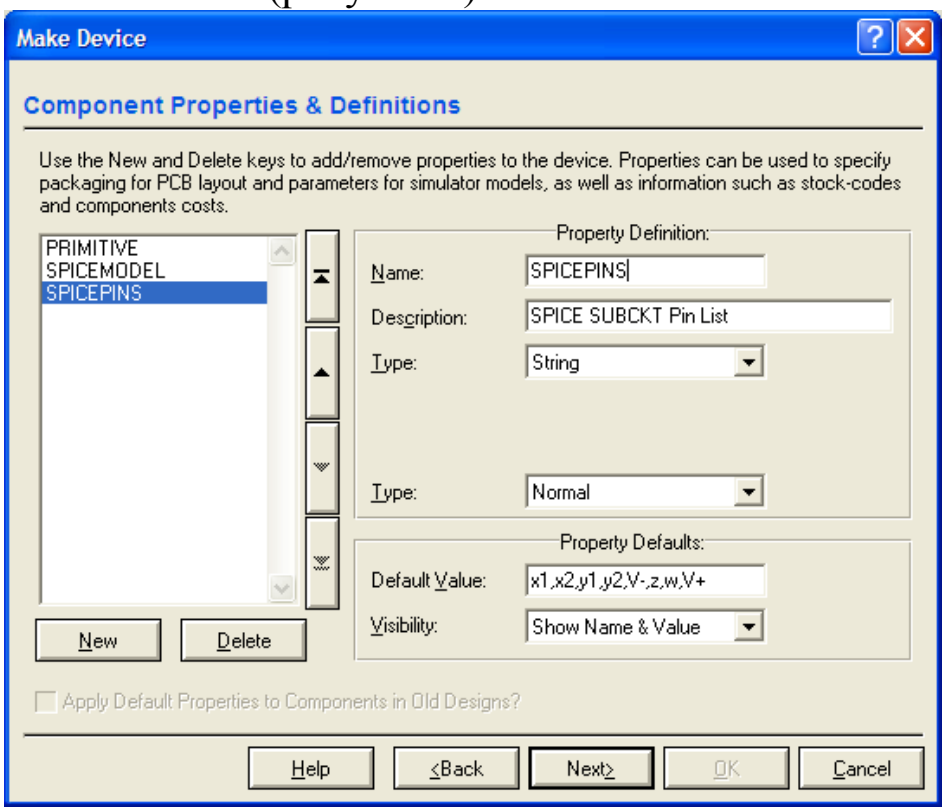


Рисунок 2.3 – Определение свойства SPICEPINS модели

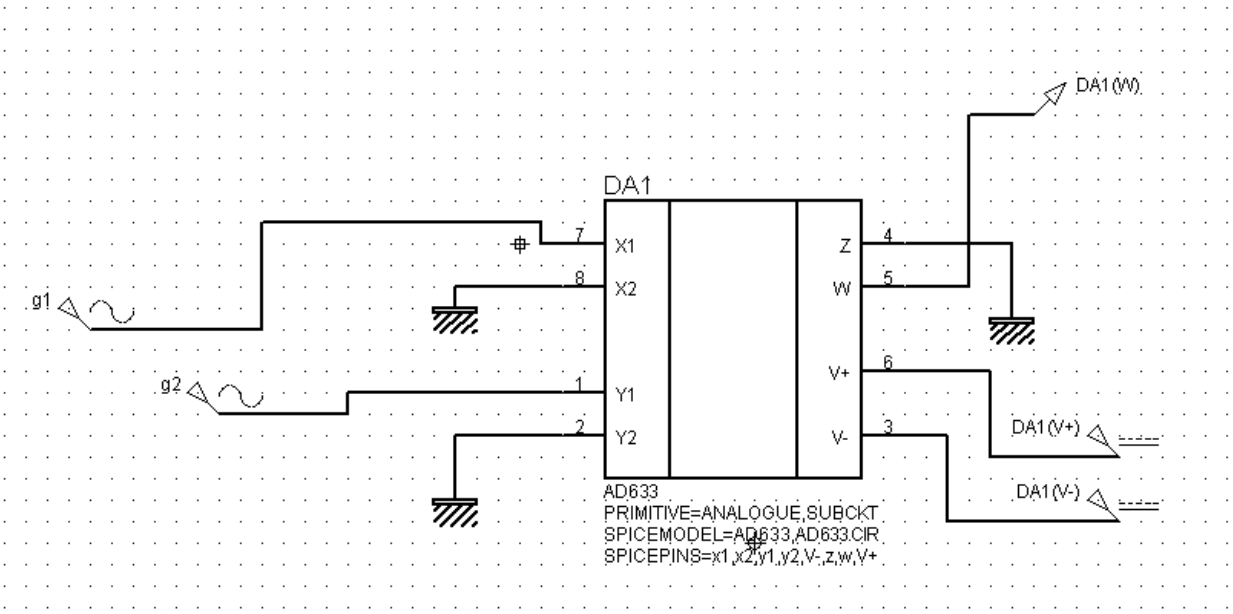


Рисунок 2.4 – Тестовая схема для умножителя AD633

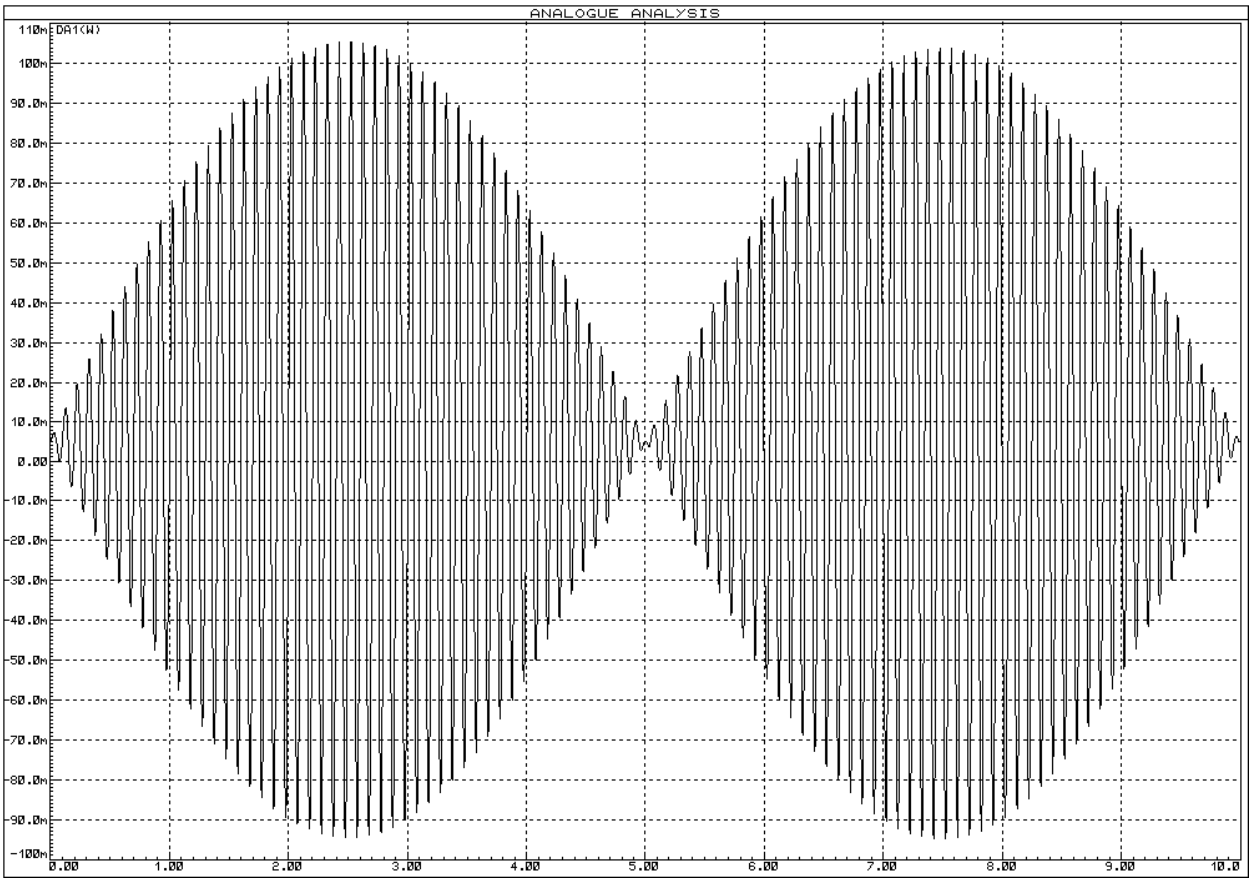


Рисунок 2.5 – Результат перемножения двух синусоид

После окончания процесса конструирования модели, ее необходимо испытать. Для этого подадим на выводы питания соответствующие документации питающие напряжения, входы X2, X3, Z соединим с землей, а на входы X1 и Y1 подадим напряжения с двух синусоидальных генераторов с разной частотой (рисунок 2.4). На вход X1 подадим синусоиду с

амплитудой 1 В и частотой 0.1 Гц, а на вход Y1 подадим синусоиду с амплитудой 1 В и частотой 10 Гц. Полученный результат умножения показан на рисунке 2.5. Нетрудно убедиться, что результат соответствует формуле умножителя, приведенной на рисунке 2.1. Следовательно, новый элемент работает правильно.

2. Порядок выполнения работы

2.1 Найти документацию, SPICE-модель микросхемы, согласно выданному варианту.

2.2 Сконструировать новую модель микросхемы в системе Proteus (условное графическое изображение микросхемы выполнять согласно ЕСКД, имитационная часть модели должна обеспечиваться SPICE-моделью).

2.3 Разработать в Proteus испытательный стенд микросхемы.

2.4 Провести испытания работоспособности новой модели микросхемы.

3. Содержание отчета:

Титульный лист с названием и номером работы, а также с фамилией исполнителя;

Цель работы;

Задание на лабораторную работу;

Структура, схема включения микросхемы согласно документации;

Схема испытательного стенда микросхемы;

Результаты испытания работоспособности новой модели микросхемы.

Выводы.

# Лабораторная работа №3
# Аналого-цифровое и цифро-аналоговое преобразование, создание цифровых приборов и систем на базе современных микроконтроллеров и микропроцессоров

## 1 Краткие теоретические сведения

Модуль аналого-цифрового преобразования (АЦП) в микроконтроллере PIC16F877 имеет восемь входных каналов. Входной аналоговый сигнал через коммутатор каналов заряжает внутренний конденсатор АЦП $C_{HOLD}$. Модуль АЦП преобразует напряжение, удерживаемое на конденсаторе $C_{HOLD,}$ в соответствующий 10-разрядный цифровой код методом последовательного приближения. Источник верхнего и нижнего опорного напряжения может быть программно - выбран с выводов VDD, VSS, AN3/VREF+ или AN2/VREF-.

Допускается работа модуля АЦП в SLEEP режиме микроконтроллера, при этом в качестве источника тактовых импульсов для АЦП должен быть выбран RC генератор.

Для управления АЦП в микроконтроллере используется 4 регистра:

· Регистр результата ADRESH (старший байт);
· Регистр результата ADRESL (младший байт);
· Регистр управления ADCON0;
· Регистр управления ADCON1.

Регистр ADCON0 используется для настройки работы модуля АЦП, а с помощью регистра ADCON1 устанавливается, какие входы микроконтроллера будут использоваться модулем АЦП и в каком режиме (аналоговый вход или цифровой порт ввода/вывода). При сбросе микроконтроллера все выводы, мультиплицированные с модулем АЦП (ANx), настраиваются как аналоговые входы.

Структурная схема модуля АЦП показана на рисунке 1. Структура регистра ADCON0 приведена на рисунке 2. Структура регистра ADCON1 приведена на рисунке 3.

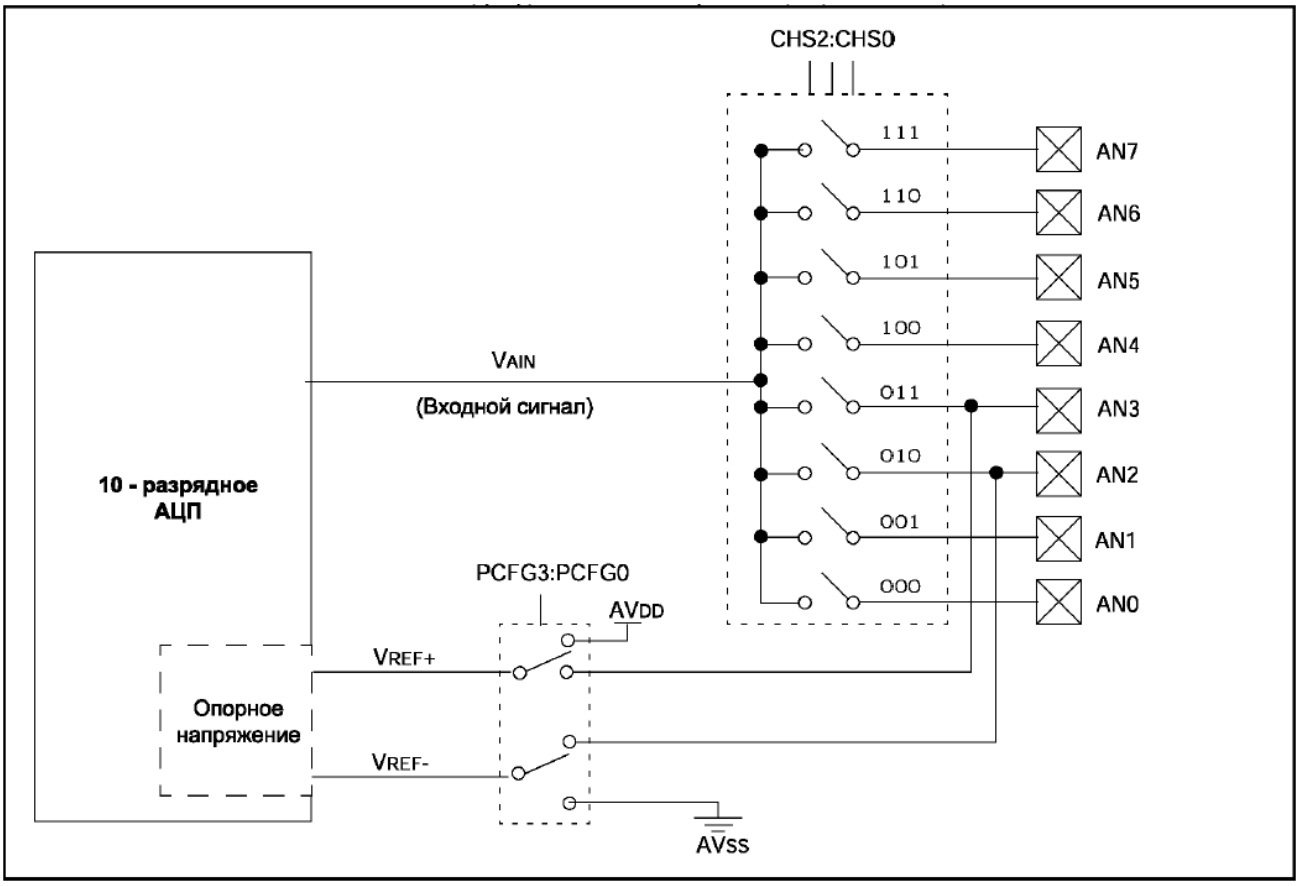


Рисунок 1 - Структурная схема модуля 10 - разрядного АЦП

В регистрах ADRESH:ADRESL сохраняется 10 - разрядный результат аналого-цифрового преобразования. Когда преобразование завершено, результат преобразования записывается в регистры ADRESH:ADRESL, после чего сбрасывается бит GO/-DONE (ADCON0<2>) и устанавливается флаг прерывания ADIF.

После включения и настройки АЦП необходимо выбрать рабочий аналоговый канал. Соответствующие биты TRIS аналоговых каналов должны настраивать канал порта ввода/вывода на вход. Перед началом преобразования необходимо выдержать временную паузу для обеспечения необходимой точности преобразования (конденсатор $C_{HOLD}$ должен успевать полностью заряжаться до уровня входного напряжения). Точные формулы расчета временной паузы приведены в документации. Отметим, что при сопротивлении источника сигнала 10 кОм эта пауза составляет примерно 20 мкс, а при сопротивлении источника сигнала 50 Ом эта пауза составляет примерно 11 мкс.

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| **ADCS1** | **ADCS0** | **CHS2** | **CHS1** | **CHS0** | **GO/-DONE** | **-** | **ADON** |
| Бит 7 | | | | | | | Бит 0 |

R – чтение бита
W – запись бита
U – не реализовано, читается как '0'
–n – значение после POR
–x – неизвестное значение после POR

биты 7-6: **ADCS1:ADCS0:** Выбор источника тактового сигнала
00 = $F_{OSC}/2$
01 = $F_{OSC}/8$
10 = $F_{OSC}/32$
11 = $F_{RC}$ (внутренний RC генератор модуля АЦП)

биты 5-3: **CHS2:CHS0:** Выбор аналогового канала
000 = канал 0, (AN0)
001 = канал 1, (AN1)
010 = канал 2, (AN2)
011 = канал 3, (AN3)
100 = канал 4, (AN4)
101 = канал 5, (AN5)
110 = канал 6, (AN6)
111 = канал 7, (AN7)

**Примечание.** Для микроконтроллеров, в которых не реализованы все 8 каналов АЦП. Модуль АЦП аппаратно позволяет выполнять выборку нереализованных каналов.

бит 2: **GO/-DONE:** Бит статуса модуля АЦП
Если ADON=1
1 = модуль АЦП выполняет преобразование (установка бита вызывает начало преобразования)
0 = состояние ожидания (аппаратно сбрасывается по завершению преобразования)

бит 1: **Не используется:** читается как '0'

бит 0: **ADON:** Бит включения модуля АЦП
1 = модуль АЦП включен
0 = модуль АЦП выключен и не потребляет тока

Рисунок 2 – Структура регистра ADCON0

| R/W-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| **ADFM** | - | - | - | **PCFG3** | **PCFG2** | **PCFG1** | **PCFG0** |
| Бит 7 | | | | | | | Бит 0 |

R – чтение бита
W – запись бита
U – не реализовано, читается как 0
–n – значение после POR
–x – неизвестное значение после POR

бит 7: **ADFM:** Формат сохранения 10-разрядного результата (см. рисунок 23-6)
1 = правое выравнивание, 6 старших бит ADRESH читаются как '0'
0 = левое выравнивание, 6 младших бит ADRESL читаются как '0'

биты 6-4: **Не используются:** читаются как '0'

биты 3-0: **PCFG3:PCFG0:** Управляющие биты настройки каналов АЦП

| PCFG3: PCFG0 | AN7 | AN6 | AN5 | AN4 | AN3 | AN2 | AN1 | AN0 | $V_{REF}+$ | $V_{REF}-$ | Кан./ $V_{REF}$ (1) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000 | A | A | A | A | A | A | A | A | $AV_{DD}$ | $AV_{SS}$ | 8/0 |
| 0001 | A | A | A | A | $V_{REF}+$ | A | A | A | AN3 | $AV_{SS}$ | 7/1 |
| 0010 | D | D | D | A | A | A | A | A | $AV_{DD}$ | $AV_{SS}$ | 5/0 |
| 0011 | D | D | D | A | $V_{REF}+$ | A | A | A | AN3 | $AV_{SS}$ | 4/1 |
| 0100 | D | D | D | D | A | D | A | A | $AV_{DD}$ | $AV_{SS}$ | 3/0 |
| 0101 | D | D | D | D | $V_{REF}+$ | D | A | A | AN3 | $AV_{SS}$ | 2/1 |
| 011x | D | D | D | D | D | D | D | D | $AV_{DD}$ | $AV_{SS}$ | 0/0 |
| 1000 | A | A | A | A | $V_{REF}+$ | $V_{REF}-$ | A | A | AN3 | AN2 | 6/2 |
| 1001 | D | D | A | A | A | A | A | A | $AV_{DD}$ | $AV_{SS}$ | 6/0 |
| 1010 | D | D | A | A | $V_{REF}+$ | A | A | A | AN3 | $AV_{SS}$ | 5/1 |
| 1011 | D | D | A | A | $V_{REF}+$ | $V_{REF}-$ | A | A | AN3 | AN2 | 4/2 |
| 1100 | D | D | D | A | $V_{REF}+$ | $V_{REF}-$ | A | A | AN3 | AN2 | 3/2 |
| 1101 | D | D | D | D | $V_{REF}+$ | $V_{REF}-$ | A | A | AN3 | AN2 | 2/2 |
| 1110 | D | D | D | D | D | D | D | A | $AV_{DD}$ | $AV_{SS}$ | 1/0 |
| 1111 | D | D | D | D | $V_{REF}+$ | $V_{REF}-$ | D | A | AN3 | AN2 | 1/2 |

A = аналоговый вход D = цифровой канал ввода/вывода

**Примечание 1.** В этом столбце указывается число аналоговых каналов, доступных для выполнения преобразования, и число входов источника опорного напряжения.

Рисунок 3 – Структура регистра ADCON1

Рекомендованная последовательность действий для работы с АЦП:

1. Настроить модуль АЦП:

· Настроить выводы как аналоговые входы, входы VREF или цифровые каналы ввода/вывода (ADCON1);

· Выбрать входной канал АЦП (ADCON0);

· Выбрать источник тактовых импульсов для АЦП (ADCON0);

· Включить модуль АЦП (ADCON0).

2. Настроить прерывание от модуля АЦП (если необходимо):

· Сбросить бит ADIF в '0';

· Установить бит ADIE в '1';

· Установить бит PEIE в '1';

· Установить бит GIE в '1'.

3. Выдержать паузу, необходимую для зарядки конденсатора $C_{HOLD}$.

4. Начать аналого-цифровое преобразование:

· Установить GO/-DONE бит в '1' (ADCON0).

5. Ожидать, окончания преобразования:

· Ждать, пока бит GO/-DONE не будет сброшен в '0'; ИЛИ

· Ожидать прерывание по окончанию преобразования.

6. Считать результат преобразования из регистров ADRESH:ADRESL, сбросить бит ADIF в '0', если это необходимо.

7. Для следующего преобразования необходимо выполнить шаги, начиная с пункта 1 или 2. Время преобразования одного бита определяется как время $T_{AD}$. Минимальное время ожидания перед следующим преобразованием должно составлять $2T_{AD}$.

Время получения одного бита результата равно $T_{AD}$. Для 10-разрядного результата требуется как минимум 11.5 $T_{AD}$. Параметры тактового сигнала для АЦП определяются программно, $T_{AD}$ может принимать следующие значения:

· $2T_{OSC}$;

· $8T_{OSC}$;

· $32T_{OSC}$;

· Внутренний RC генератор модуля АЦП (2-6 мкс).

Для получения корректного результата преобразования необходимо выбрать источник тактового сигнала АЦП, обеспечивающий время TAD не менее 1.6 мкс. Таблицы выбора источника тактового сигнала приведены в документации. Отметим, что для частоты 20 МГц необходимо выбирать источником TAD или RC-генератор, или 32TOSC=1.6 мкс (значения конфигурационных битов ADCS1:ADCS0 = 10). Когда тактовая частота микроконтроллера больше 1МГц, рекомендуется использовать RC генератор АЦП только для работы в SLEEP режиме.

В следующем примере показана последовательность действий для работы с АЦП. Выводы настроены как аналоговые входы. Источник опорного напряжения – $AV_{DD}$, $AV_{SS}$. Разрешены прерывания от модуля АЦП. Источником импульсов преобразования является RC генератор АЦП. Аналоговое цифровое преобразование выполняется с вывода AN0:

```
    BSF STATUS, RP0   ; Выбрать банк 1
    CLRF ADCON1   ; Настроить входы АЦП
    BSF PIE1, ADIE    ; Разрешить прерывания от АЦП
    BCF STATUS, RP0 ; Выбрать банк 0
    MOVLW 0xC1    ; Тактовые импульсы от RC генера-
тора АЦП,
```

```
    MOVWF ADCON0  ; включить АЦП, выбрать канал 0
    BCF PIR1, ADIF   ; Сбросить флаг прерываний от
АЦП
    BSF INTCON, PEIE ; Разрешить периферийные преры-
вания
    BSF INTCON, GIE ; Разрешить прерывания в системе
    ;
    ; Выдержать паузу, необходимую для заряда внут-
реннего конденсатора CHOLD.
    ; Затем начинать преобразование АЦП.
    ;
    BSF ADCON0, GO   ; Старт преобразования
    : ; Ожидать установку флага ADIF или сброс
    : ; бита GO/-DONE по завершению преобразования
```

10-разрядный результат преобразования сохраняется в спаренном 16-разрядном регистре ADRESH:ADRESL. Запись результата преобразования может выполняться с правым или левым выравниванием, в зависимости от значения бита ADFM (см. рисунок 4). Не задействованные биты регистра ADRESH:ADRESL читаются как ‘0’. Если модуль АЦП выключен, то 8-разрядные регистры ADRESH и ADREL могут использоваться как регистры общего назначения.

Модуль АЦП может работать в SLEEP режиме микроконтроллера при условии, что источником импульсов преобразования АЦП будет внутренний RC генератор (ADCS1:ADCS0=11). При выборе RC генератора импульсов модуль АЦП сделает задержку в один машинный цикл перед началом преобразования. Это позволяет программе пользователя выполнить команду SLEEP, тем самым уменьшить “цифровой шум” во время преобразования. После завершения преобразования аппаратно сбрасывается бит GO/-DONE в '0', результат преобразования сохраняется в регистрах ADRESH:ADRESL. Если разрешено прерывание от АЦП, то микроконтроллер выйдет из режима SLEEP. Если же прерывание было запрещено, то после преобразования модуль АЦП будет выключен, хотя бит ADON останется установленным.

Если был выбран другой источник тактовых импульсов АЦП (не внутренний RC генератор), то выполнение программой инструкции SLEEP прервет процесс преобразования и выключит модуль АЦП, оставив установленным бит ADON. Выключение модуля АЦП уменьшит ток

потребления микроконтроллера. Инструкция SLEEP должна быть выполнена сразу после команды, устанавливающей бит GO/-DONE в '1'.

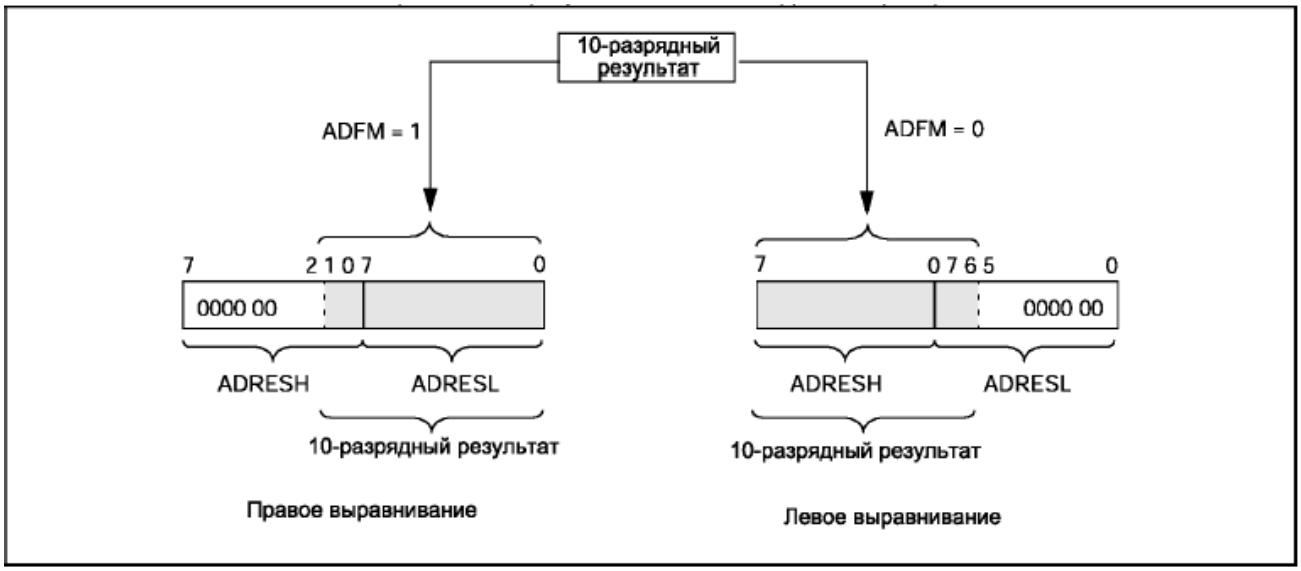


Рисунок 4 - Выравнивание результата аналого-цифрового преобразования

Абсолютная точность АЦП определяется суммарной ошибкой, исходя из ошибки дискретизации, интегральной ошибки, ошибки шкалы, ошибки смещения и монотонности. Суммарная ошибка определяется как максимальный разброс между текущим и идеальным результатом для любого значения. Абсолютная ошибка АЦП меньше ± 1 значащего бита при VDD=VREF, но она возрастает при отклонении VREF от VDD.

В некотором диапазоне напряжений на аналоговом входе цифровой результат будет один и тот же. Это возникает из-за дискретизации, которая неизбежна при преобразовании аналоговой величины в цифровую форму. Ошибка дискретизации составляет ± от значащего бита, и единственный способ уменьшить ее - увеличить разрядность АЦП.

Ошибку смещения составляет разность между результатом первого преобразования и идеальным значением. Эта ошибка сдвигает всю передаточную функцию, и может быть учтена при помощи калибровки. Ошибка вносится в результате наложения токов утечки и выходного сопротивления источника сигнала. Ошибка усиления измеряется как максимальное отклонение результата, скорректированного с учетом ошибки смещения. Эта ошибка проявляется в виде изменения наклона передаточной функции. Ошибка усиления может быть откалибрована и учтена. Ошибка линейности определяется как разница в приращении входного напряжения для получения одинакового приращения выходного кода и не

поддается калибровке. Интегральная ошибка вычисляется как отклонение результата, скорректированного с учетом ошибки усиления. Дифференциальная ошибка вычисляется как отклонение максимальной длины кода результата от идеальной длины кода без учета других ошибок.

| № пар. | Обоз. | Описание | | Мин. | Тип** | Макс. | Ед. | Примечание |
|---|---|---|---|---|---|---|---|---|
| A01 | $N_R$ | Разрядность | | - | - | 10 | бит | $V_{REF} = V_{DD} = 5.12В$, $V_{SS} \le V_{AIN} \le V_{REF}$ |
| A03 | $E_{IL}$ | Интегральная погрешность | | - | - | < ± 1 | LSb | $V_{REF} = V_{DD} = 5.12В$, $V_{SS} \le V_{AIN} \le V_{REF}$ |
| A04 | $E_{DL}$ | Дифференциальная погрешность | | - | - | < ± 1 | LSb | $V_{REF} = V_{DD} = 5.12В$, $V_{SS} \le V_{AIN} \le V_{REF}$ |
| A06 | $E_{OFF}$ | Ошибка смещения | | - | - | < ± 2 | LSb | $V_{REF} = V_{DD} = 5.12В$, $V_{SS} \le V_{AIN} \le V_{REF}$ |
| A07 | $E_{GN}$ | Ошибка усиления | | - | - | < ± 1 | LSb | $V_{REF} = V_{DD} = 5.12В$, $V_{SS} \le V_{AIN} \le V_{REF}$ |
| A10 | - | Монотонность[3] | | Гарантируется | | | - | $V_{SS} \le V_{AIN} \le V_{REF}$ |
| A20 | $V_{REF}$ | Опорное напряжение ($V_{REF+}$-$V_{REF-}$) | | 2.0 | - | $V_{DD}$ + 0.3 | В | Минимальное значение для 10-разрядного АЦП |
| A21 | $V_{REF+}$ | Положительное опорное напр. | | $AV_{DD}$ – 2.5 | | $AV_{DD}$ + 0.3 | В | |
| A22 | $V_{REF-}$ | Отрицательное опорное напр. | | $AV_{SS}$ – 0.3 | | $V_{REF+}$ - 2.0 | В | |
| A25 | $V_{AIN}$ | Аналоговый вход | | $V_{SS}$ – 0.3 | - | $V_{REF}$ + 0.3 | В | |
| A30 | $Z_{AIN}$ | Сопротивление источника сигн. | | - | - | 10.0 | кОм | |
| A40 | $I_{AD}$ | Потребляемый ток АЦП | F | - | 220 | - | мкА | Среднее потребление при включенном АЦП[1] |
| | | | LF | - | 90 | - | мкА | |
| A50 | $I_{REF}$ | Потребляемый ток от источника опорного напряжения[2] | | 10 | - | 1000 | мкА | Во время выборки $V_{AIN}$. Основано на дифференц. значении заряда $C_{HOLD}$ до $V_{AIN}$. |
| | | | | - | - | 10 | мкА | Во время преобразования. |

** - В столбце "Тип." приведены параметры при VDD=5.0В, 25ºС, если не указано иное. Эти параметры являются ориентировочными, используются при разработке устройств и не измеряются.

Примечания:

1. Выключенный модуль АЦП не потребляет тока, кроме токов утечки.

2. Ток со входа RA3 или VDD в зависимости от выбранного источника опорного напряжения.

3. Результат АЦП никогда не уменьшается с увеличением напряжения на входе и не имеет кодов отсутствия напряжения.

2. Цель работы: уметь использовать модуль АЦП в микроконтроллерах Microchip.

3. Порядок выполнения работы

3.1 Решить практическую задачу согласно своему варианту с использованием модуля АЦП.

3.2 Проверить работоспособность разработанной программы в системе Proteus.

4. Содержание отчета:

Титульный лист с названием и номером работы, а также с фамилией исполнителя

Цель работы

Задание на лабораторную работу

Схема спроектированной системы

Листинг программы.

Выводы

Лабораторная работа №4
Автоматизированное проектирование узлов приборов и систем обработки сигнала во временном домене

## 1 Краткие теоретические сведения

Микросхемы ЦАП позволяют микроконтроллерам производить генерацию аналоговых сигналов. Интерфейсы микросхем ЦАП делятся на параллельные и последовательные интерфейсы. Параллельные интерфейсы занимают больше линий ввода-вывода у микроконтроллера, но они и более быстродействующие. Последовательные интерфейсы ЦАП занимают значительно меньше линий ввода-вывода, так как данные в таких интерфейсах, как правило, передаются по двум основным линиям – линии данных и линии синхронизации. Существуют микросхемы, в которые информация передается даже по одному проводу – шине «One Wire». Однако такая передача данных требует, как правило, больше тактов работы микроконтроллера. Разберем пример подключения ЦАП к микроконтроллеру на примере микросхемы MAX503.

Микросхема MAX503 (см. документацию на www.maxim-ic.com) – это экономичный, 10-битный преобразователь цифрового кода в напряжение, который использует или однополярное напряжение питания +5В, или двухполярное напряжение питания плюс минус 5В. Величина потребляемого тока (250 мкА) от напряжения питания +5В позволяет использовать эту микросхему для производства портативной медицинской аппаратуры.

На рисунке 1 приведена функциональная схема и обозначение выводов микросхемы MAX503.

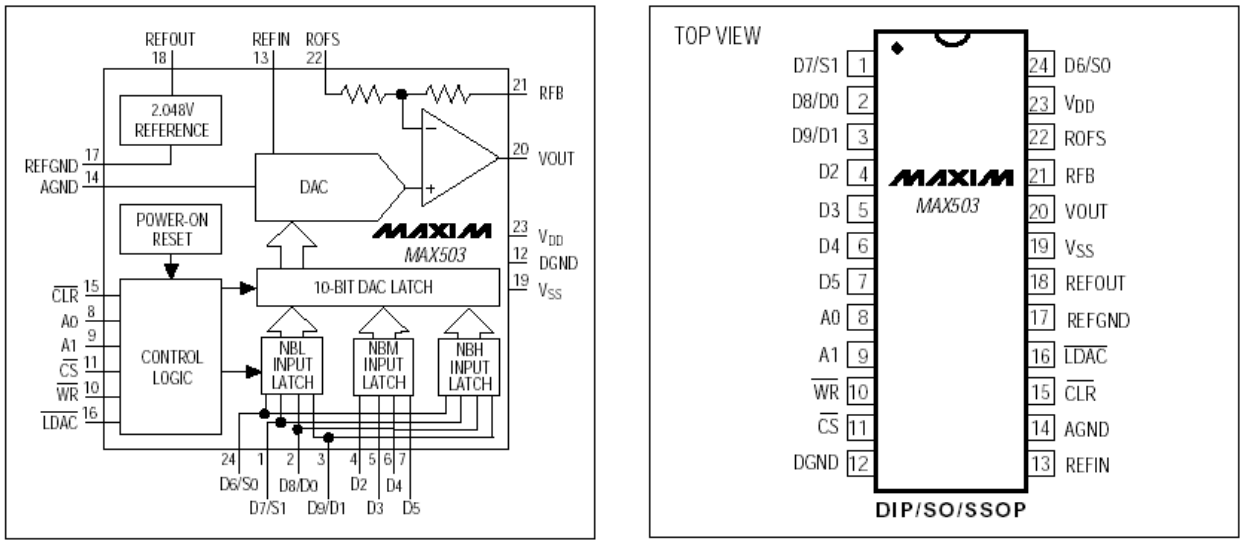


Рисунок 1 - Функциональная схема и обозначение выводов микросхемы MAX503

Назначение выводов микросхемы MAX503 приведено в таблице 1. На рисунке 2 приведена временная диаграмма циклов записи в MAX503.

MAX503 предоставляет несколько путей записи информации в свои внутренние регистры. Используем способ, при котором изменение цифровой комбинации во входном регистре сразу производит изменение аналогового напряжения на выходе, т.е. режим без дополнительной отдельной пересылки 10 бит данных из входного регистра в регистр ЦАП, при таком способе не используется управляющий вывод LDAC (LDAC=0). Диаграмма обмена информацией при таком способе показана на рисунке 3.

На рисунке 4 приведена схема включения MAX503 в лабораторном макете.

С учетом всей вышеприведенной информации (схемы включения, режима обмена, цикла записи и т.д.) напишем программу обмена информацией между микроконтроллером и ЦАП MAX503 (листинг 1).

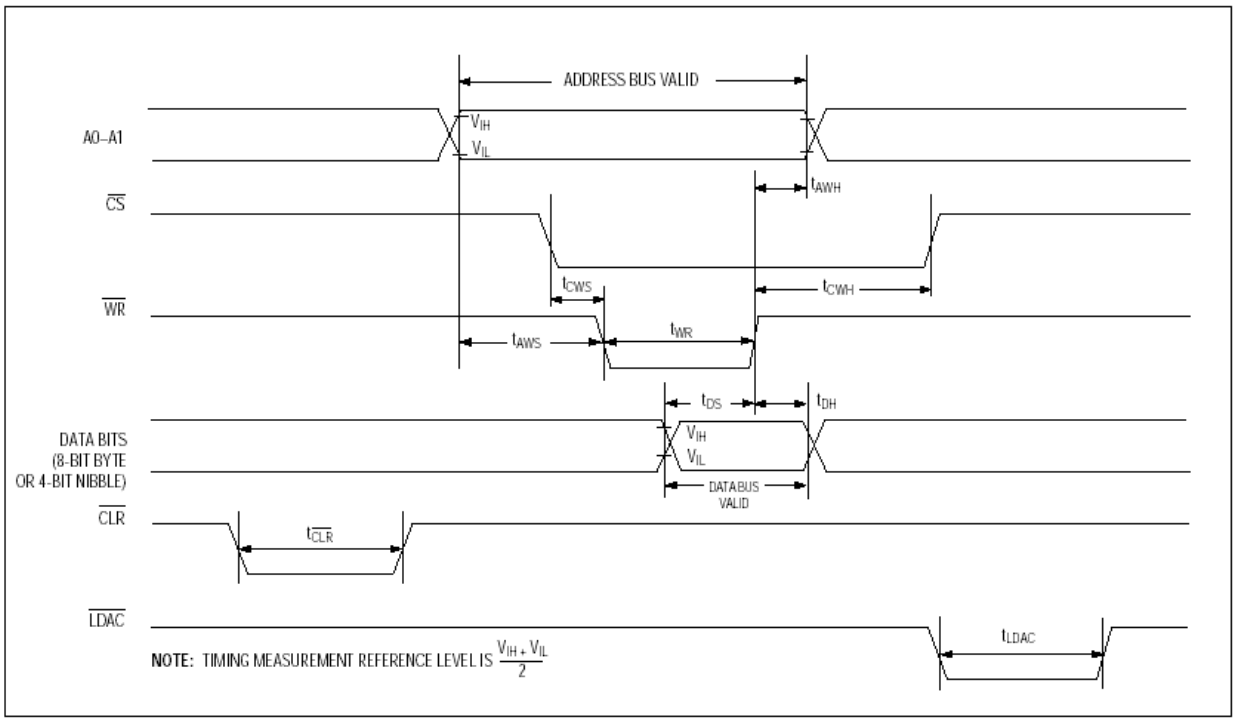


Рисунок 2 - Временная диаграмма циклов записи в MAX503

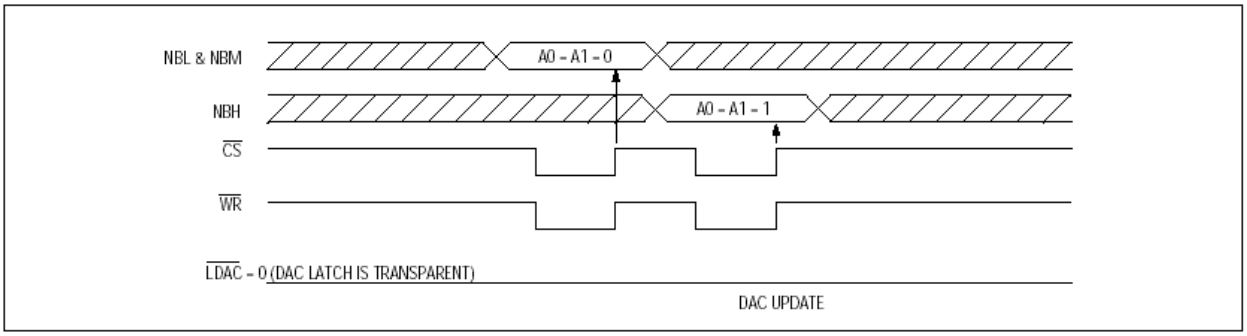


Рисунок 3 – Диаграмма обмена информацией при «прозрачных» защелках регистра ЦАП (LDAC=0). При режиме записи NBL и NBM во входной регистр записываются 8 младших бит информации, при режиме записи NBH записываются два старших бита.

Таблица 1 - Назначение выводов микросхемы MAX503

| **Номер вывода** | **Имя вывода** | **Назначение вывода** |
|---|---|---|
| 1 | D7/ S1 | Вход D7 когда A0 = A1 = 1 или вход S1 когда A0 = 0 и A1 = 1. |
| 2 | D8/ D0 | Вход D8 когда A0 = A1 = 1 или вход D0 когда A0 = 0 и A1 = 1. |
| 3 | D9/ D1 | Вход D9 когда A0 = A1 = 1 или вход D1 когда A0 = 0 и A1 = 1. |
| 4 | D2 | Вход D2. |
| 5 | D3 | Вход D3. |
| 6 | D4 | Вход D4. |
| 7 | D5 | Вход D5. |
| 8 | A0 | Адресная линия A0. |
| 9 | A1 | Адресная линия A1. |
| 10 | WR | Вход записи (активный уровень – низкий). Используется совместно с CS для записи данных во входные защелки, которые выбраны адресными линиями A0 и A1. |
| 11 | CS | Выбор кристалла (активный уровень – низкий). Позволяет включить запись в микросхему данных из общих шин |
| 12 | DGND | Цифровая земля. |
| 13 | REFIN | Вход источников опорного напряжения (ИОН). Подключите внешний источник опорного напряжения к этому выводу или подключите вывод REFOUT (вывод 18) для использования внутреннего источника на 2.048 В. |
| 14 | AGND | Аналоговая земля. |
| 15 | CLR | Очистка (активный уровень низкий). Очищает защелки ЦАП (регистр ЦАП) в нули. |
| 16 | LDAC | Загрузка защелок в ЦАП (активный уровень низкий). Перемещает содержимое входных защелок в регистр ЦАП и изменяет выходное напряжение. |
| 17 | REFGND | Земля ИОН. Должна быть присоединена к AGND когда используется внутренний ИОН. Подсоединяется к VDD для отключения внутреннего ИОН и экономии энергии. |
| 18 | REFOUT | Выход внутреннего ИОН. Внутренний ИОН на 2.048 |

| | | |
|---|---|---|
| | | В. |
| 19 | VSS | Отрицательный вывод напряжения питания. Часто подключается к земле при однополярном питании или к -5В при двухполярном питании. |
| 20 | VOUT | Выход ЦАП(напряжение). |
| 21 | RFB | Вывод резистора обратной связи выходного усилителя. Рекомендуется подключать напрямую к VOUT. |
| 22 | ROFS | Вывод резистора смещения выходного усилителя. Подключается к VOUT для коэффициента усиления = 1, к AGND для коэффициента усиления = 2 или к REFIN для биполярного выхода. |
| 23 | VDD | Положительный вывод напряжения питания. (+5V) |
| 24 | D6/S0 | Вход D6 когда A0 = A1 = 1 или вход S0 когда A0 = 0 и A1 = 1. |

Листинг 1 – Программа обмена информацией между микроконтроллером и ЦАП MAX503

```
; При инициализации:
clrf  PORTC    ;сигнал ldac должен быть равен 0 !!!
(rc2)
;……
;-----------------------------------------------
;Функции работы с цап MAX503CWG
;-----------------------------------------------
;8 бит информации - в регистре dacreg
;2 старших бита - в регистре dacreg2
;сигнал ldac должен быть равен 0 !!!
sendtodac

 bcf PORTC,0   ;a0,a1=low
 bcf PORTC,1
 bcf PORTE,2   ;cs=0
 call delay2   ;выдержка tcws
 bcf PORTE,1   ;wr=0
       ;сначала передаются 6 бит d0-d5, а d6=0, d7=0
 movf dacreg,w
 andlw 0x3f
 movwf PORTD
 call delay2
 bsf PORTE,1   ;wr=1
 call delay2
 bsf PORTC,0   ;a0,a1=high
 bsf PORTC,1
 call delay2   ;выдержка tcws
 bcf PORTE,1   ;wr=0
        ;потом передается d6,d7, а в d0 и d1 переда-
ются d8 и d9
 movf dacreg,w
 andlw 0xc0
 iorwf dacreg2,w
 movwf PORTD
 call delay2
 bsf PORTE,1   ;wr=1
 call delay2
```

```
 bsf PORTE,2   ;cs=1
return
```

2 Цель работы: уметь находить и пользоваться документацией на микросхемы; реализовывать аналоговый вывод из микроконтроллерных устройств.

3. Порядок выполнения работы

3.1 Найти документацию на заданную согласно своему варианту микросхему ЦАП.

3.2 Разобраться в режимах работы микросхемы ЦАП, ее протоколах информационного обмена.

3.3 Решить практическую задачу по сопряжению микросхемы ЦАП и микроконтроллера. Разработать подпрограмму информационного обмена микроконтроллера с микросхемой ЦАП.

3.4 Проверить работоспособность разработанной системы в Proteus.

4. Содержание отчета:

Титульный лист с названием и номером работы, а также с фамилией исполнителя

Цель работы

Задание на лабораторную работу

Схема спроектированной системы

Листинг программы.

Основные выдержки из фирменной документации на микросхему ЦАП (схема включения, назначение выводов, диаграммы записи и т.д.)

Выводы.

5 Контрольные вопросы

5.1 От чего зависит быстродействие ЦАП?

5.2 На какой частоте работают современные микросхемы ЦАП?

5.3 Зачем для ЦАП необходим источник опорного напряжения (ИОН)?

5.4 Какие выводы у ЦАП MAX503 образуют шину данных, какие шину адреса, а какие шину управления?

5.5 Нарисуйте типичную характеристику преобразования ЦАП.

5.6 От чего зависит максимальное выходное напряжение у ЦАП MAX503?

6 Задания.

Требуется заданную микросхему ЦАП подключить к микроконтроллеру, написать процедуры обмена. Документацию на микросхемы, при необходимости, закачать с сайта производителя.

| № Варианта | Тип ЦАП | Производитель ЦАП |
|---|---|---|
| 1 | MAX504 | Maxim-ic |
| 2 | MAX515 | Maxim-ic |
| 3 | MCP4921 | Microchip |
| 4 | LTC1450 | Linear Technology |
| 5 | LTC1451 | Linear Technology |
| 6 | LTC1456 | Linear Technology |
| 7 | LTC1655 | Linear Technology |
| 8 | MCP4922 | Microchip |
| 9 | AD5305 | Analog Devices |
| 10 | AD5310 | Analog Devices |
| 11 | AD5315 | Analog Devices |
| 12 | AD5320 | Analog Devices |
| 13 | AD5611 | Analog Devices |
| 14 | AD5325 | Analog Devices |
| 15 | MCP4921 | Microchip |
| 16 | MCP4821 | Microchip |
| 17 | MCP4725 | Microchip |
| 18 | MAX5822 | Maxim-ic |
| 19 | MAX5821 | Maxim-ic |
| 20 | MAX5820 | Maxim-ic |
| 21 | MAX5541 | Maxim-ic |

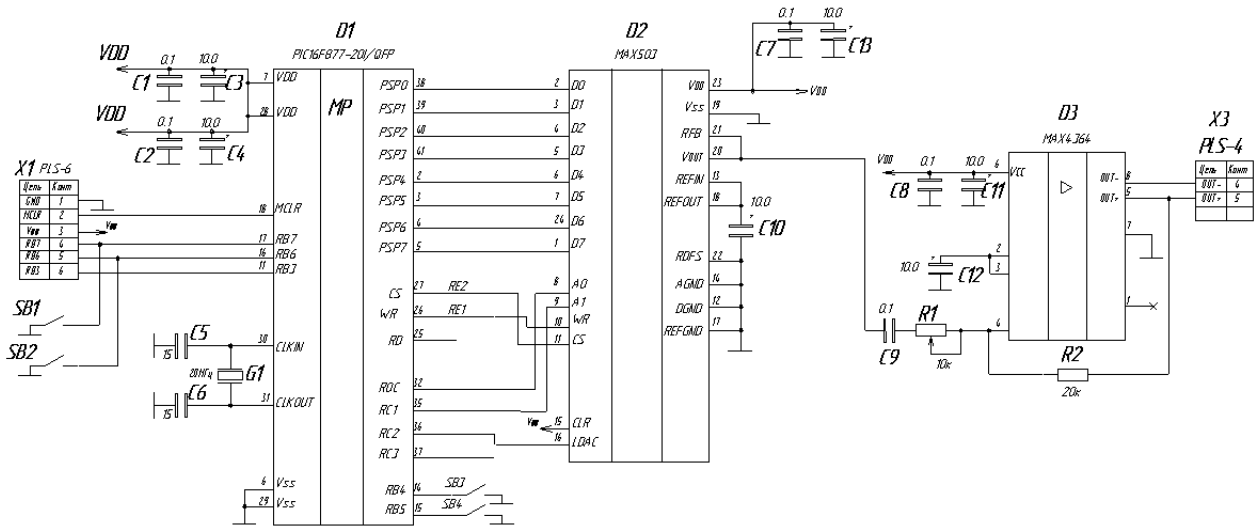


Рисунок 4 – Схема включения MAX503 в лабораторном макете.

## Лабораторная работа №5
## Основы разработки моделей элементов электрических цепей в Proteus

### 1 Краткие теоретические сведения

Любая модель элемента электрической цепи в системе схемотехнического моделирования Proteus состоит из нескольких частей:

- визуальная часть - условное графическое обозначение (УГО) элемента в совокупности с описанием свойств элемента,

- корпусная часть – корпус элемента на печатной плате (необязательная часть),

-имитационная часть – описывает поведение элемента при симуляции (необязательная часть).

В зависимости от типа имитационной части подразделяют модели PSpice, аналоговые модели, цифровые модели, смешанные модели и модели VSM, написанные на языке C++.

Целью данной лабораторной работы является изучение особенностей построения УГО элементов.

САПР Proteus предоставляет мощный инструмент для построения электрических принципиальных схем. Однако большинство УГО элементов не соответствуют требованиям ЕСКД. Поэтому возникает необходимость или переделывать существующие УГО элементов, или создавать собственную новую библиотеку элементов со стандартными обозначениями.

Для примера построим УГО операционного усилителя. Как известно из ГОСТ 2.759-82 «ЕСКД. Обозначения условные графические в схемах. Элементы аналоговой техники» УГО операционного усилителя состоит из прямоугольника, у которого есть основное поле, и дополнительно могут быть выделены одно или два дополнительных поля. Размер прямоугольника определяется числом выводов. Входы операционного усилителя обозначаются слева, а выходы справа. Инверсные входы обозначаются кружком. Над выводами рисуется метка - порядковый номер вывода на микросхеме. В основном поле рисуется треугольник - символ усилителя. В дополнительных полях может присутствовать обозначение вывода.

Создавать новый элемент можно на любой части рабочего листа Proteus. С помощью графических примитивов сформируем прямоугольник-корпус элемента.

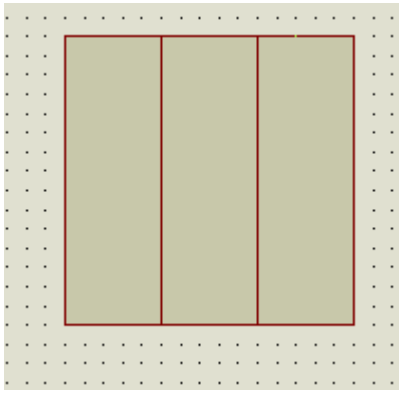

Рисунок 1.1 – Формирование прямоугольника-корпуса элемента с полями

Для этого надо щелкнуть на иконке «2-D Graphics Box Mode». Желательно проверить, что задан графический стиль «Component». С помощью примитива «2-D Graphics Line Mode» создадим два дополнительных поля (рисунок 1.1). Далее с помощью этого же примитива нарисуем треугольник-символ операционного усилителя, а также добавим кружок – символ инверсного входа. Для более точного рисования можно уменьшить шаг сетки (пункт меню View\Snap 50th). Однако для создания выводов всегда рекомендуется возвращаться к крупному шагу сетки, например, 0.1 дюйма (рисунок 1.2).

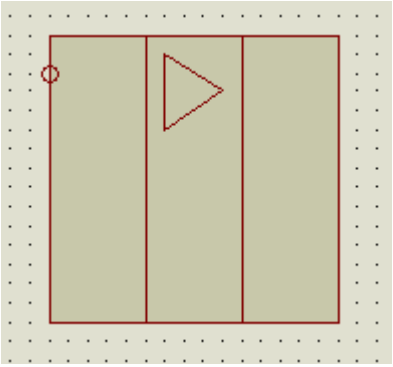

Рисунок 1.2 – Формирование символа усилителя и символа инверсного входа

Следующий важный этап – это добавление выводов. Для этого необходимо щелкнуть иконку «Device Pins Mode» (желательно при этом не изменять стиль «Default») и добавить необходимое количество выводов. При щелчке левой кнопкой мыши на выводе появляется окно диалога – редактора свойств выводов (рисунок 1.3).

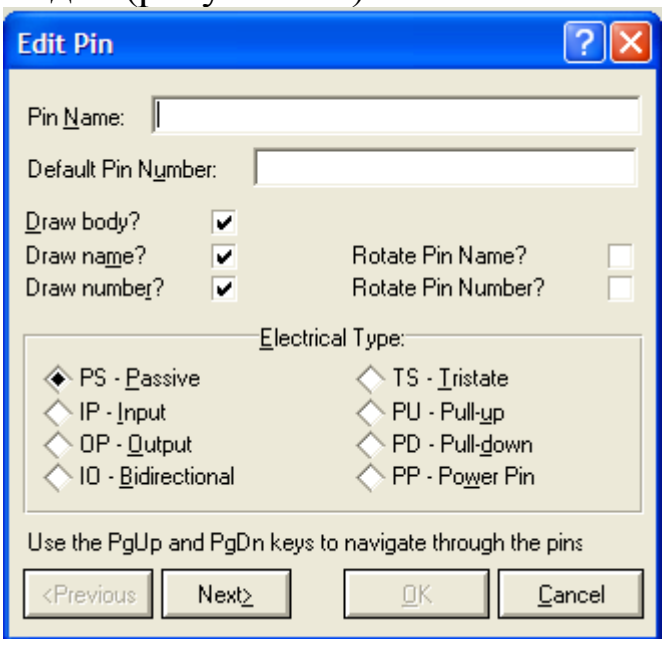


Рисунок 1.3 – Редактор свойств выводов.

Необходимо указать имя вывода (Pin Name), номер вывода (Pin Number), выводить ли на экран сам вывод (Draw body), отдельно его имя (name) и номер (number). Также необходимо определить тип вывода – пассивный (Passive), вход (Input), выход (Output), двунаправленный (Bidirectional), с третьим Z состоянием (Tristate), с подтягивающим к питанию резистором (Pull-up), с шунтирующим к земле резистором (Pull-down), вывод питания (Power pin). Назовем инвертирующий вход нашего операционного усилителя как NEG IP, неинвертирущий вход POS IP, выход OUT, плюс питания V+, минус питания V-. Обычно номер инвертирую-

щего входа 2, неинвертирущего 3, выхода 6, плюса питания 7, минуса питания 4. УГО усилителя с определенными выводами приведено на рисунке 1.4.

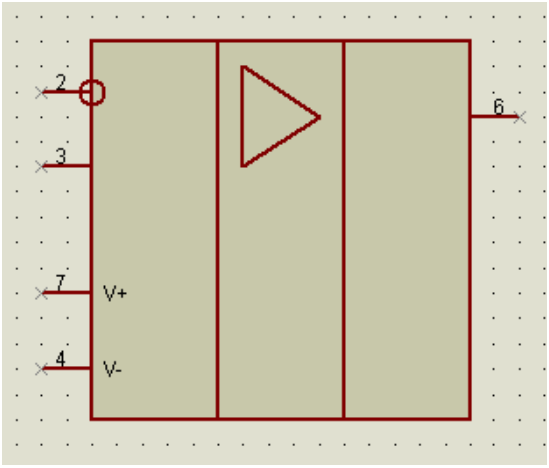


Рисунок 1.4 – УГО операционного усилителя

Чтобы поместить созданный элемент в библиотеку, необходимо выделить весь вновь созданный компонент и нажать на кнопку «Make Device». В результате запустится помощник, первый диалог которого показан на рисунке 1.5.

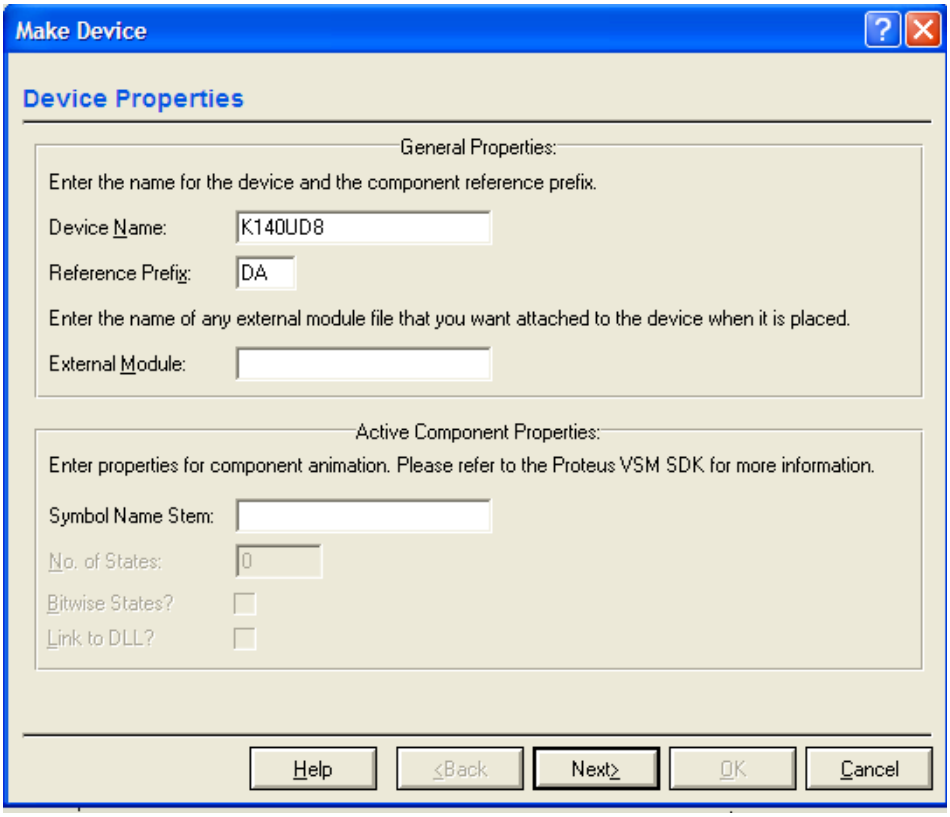


Рисунок 1.5 – Определение основных свойств элементов.

В основных свойствах необходимо указать имя компонента (Device Name), буквенный префикс позиционного обозначения (Reference Prefix),

если есть, то имя внешнего файла – модуля, а также свойства имитационной части активных компонентов, если это необходимо.

На следующем шаге мастера можно задать один или несколько отпечатков корпуса (footprint) компонента на печатной плате. Для выбора типа корпуса из библиотеки существующих корпусов необходимо нажать на кнопку Add/Edit. Открывается форма простановки соответствия между номерами выводов корпуса и УГО. Для установки типа корпуса необходимо в этой форме нажать кнопку «Add» для поиска необходимого корпуса. Например, для микросхемы КР140УД8 ближе всего подходит тип корпуса DIL08 (рисунок 1.6).

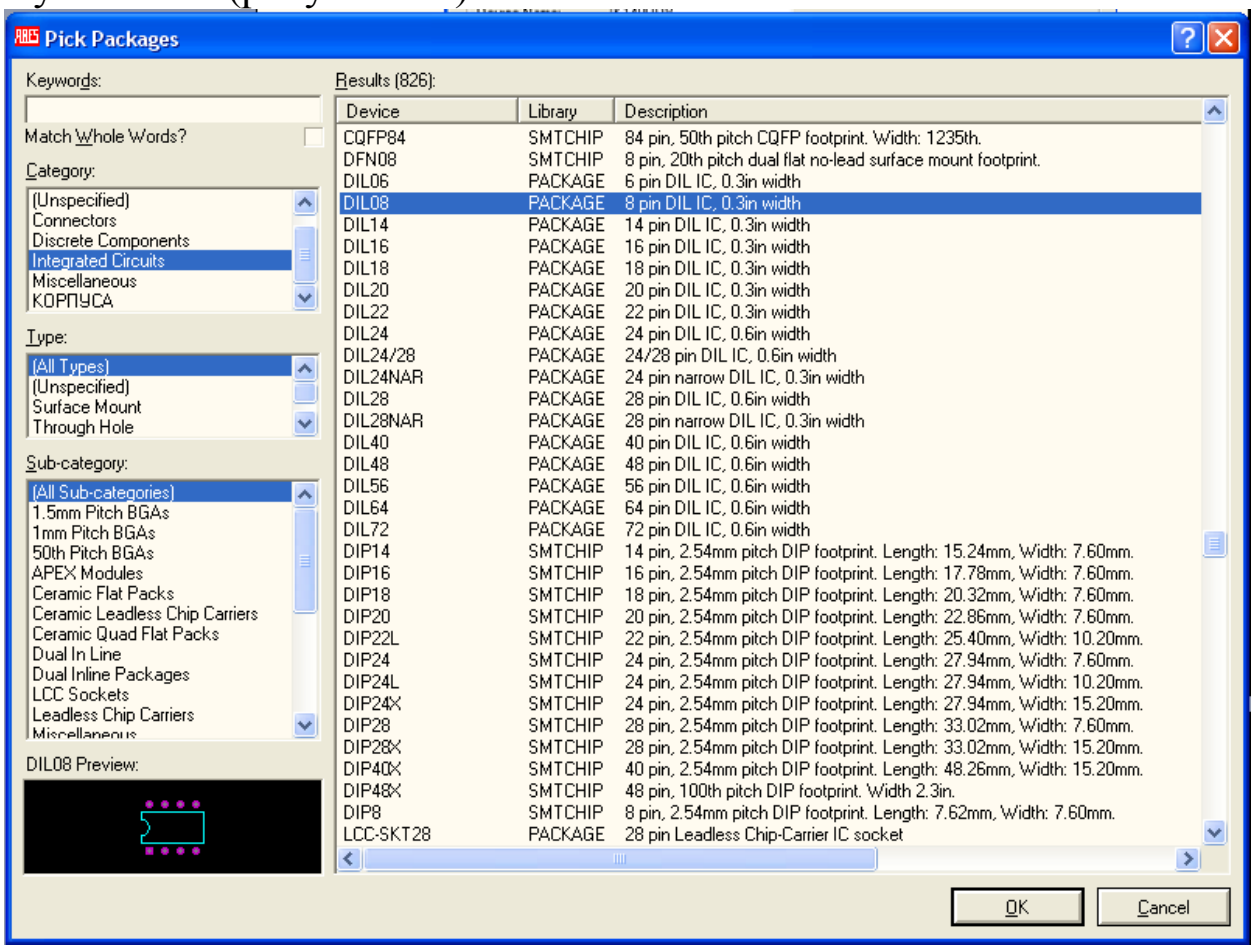


Рисунок 1.6 – Поиск типа корпуса

В результате установки типа корпуса, мы возвращаемся к форме простановки соответствия выводов (рисунок 7). На этом этапе можно переделать связь между номерами выводов УГО и номерами выводов корпуса, путем простановки других номеров выводов в поле «А» таблицы соответствия, если это необходимо. То есть определенному названию вывода (пина) можно присвоить другой номер корпуса, отличный от номера в УГО.

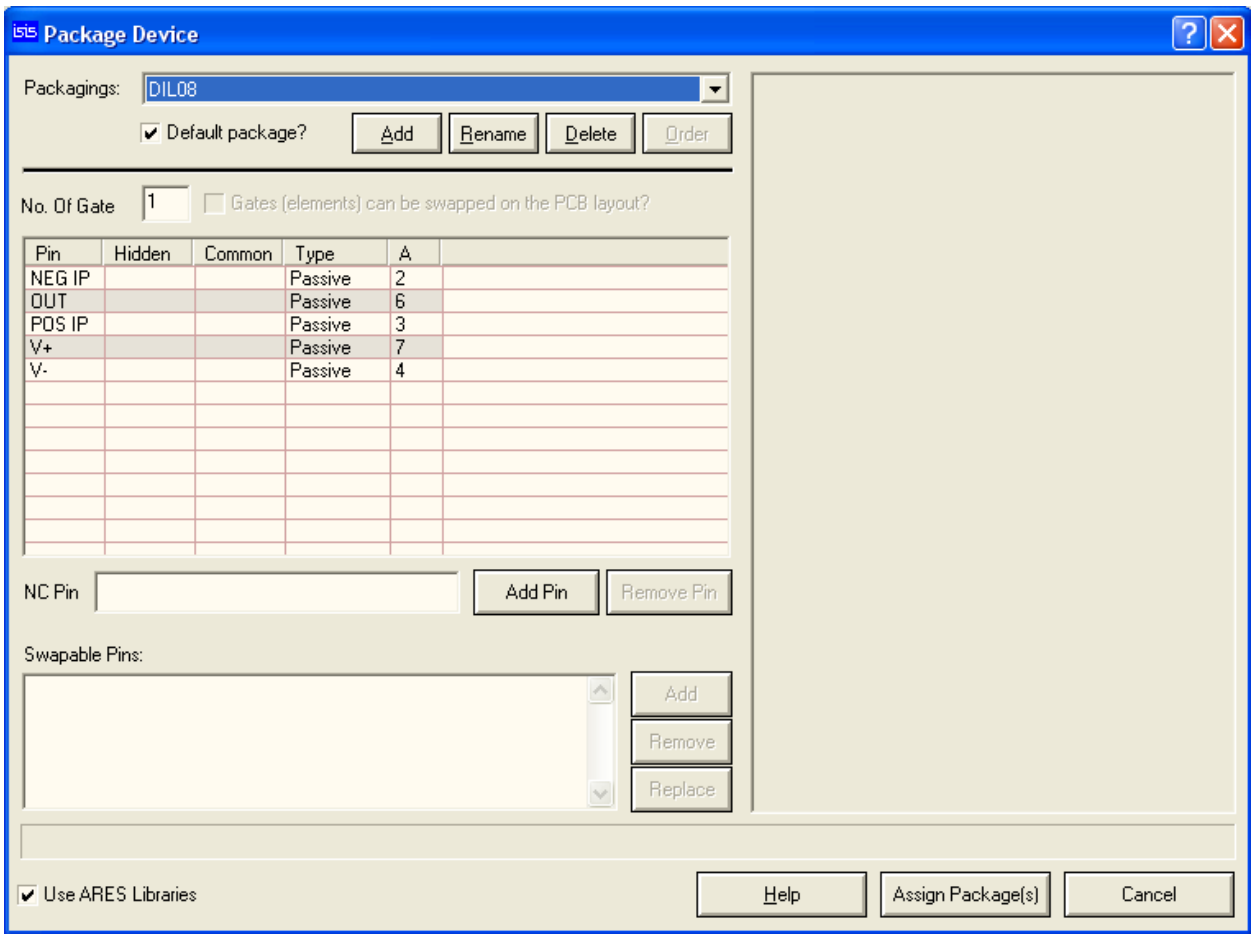


Рисунок 1.7 – Установка в соответствие выводов корпуса и УГО

Второй шаг мастера с присвоенным корпусом показан на рисунке 1.8.

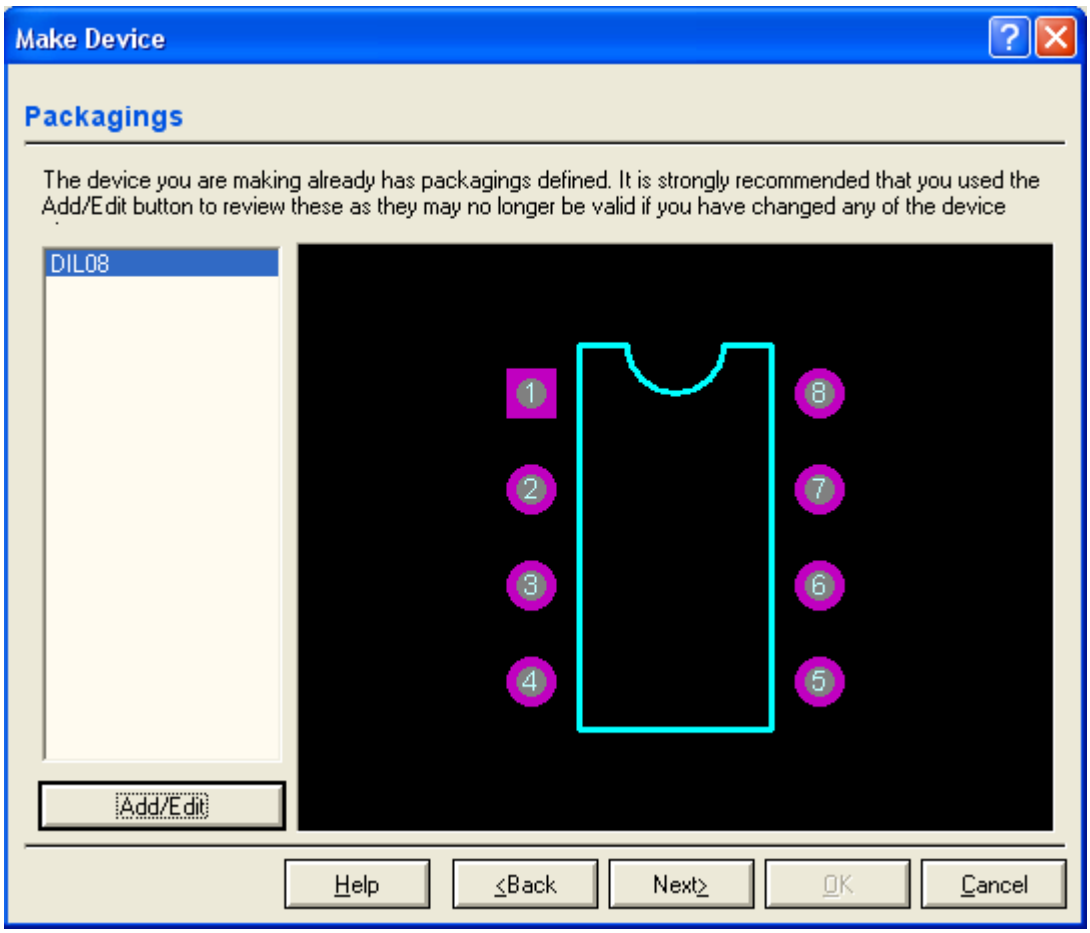


Рисунок 1.8 – Форма присвоения типа корпуса компоненту.

Следующий шаг мастера – задание свойств и определений компонента (рисунок 9). Мы задали только корпус нашему компоненту, поэтому на этой форме будет только свойство PACKAGE. При создании симуляционных моделей в этом диалоге необходимо будет определять тип симуляционной модели, параметры и свойства моделей по умолчанию и т.д.

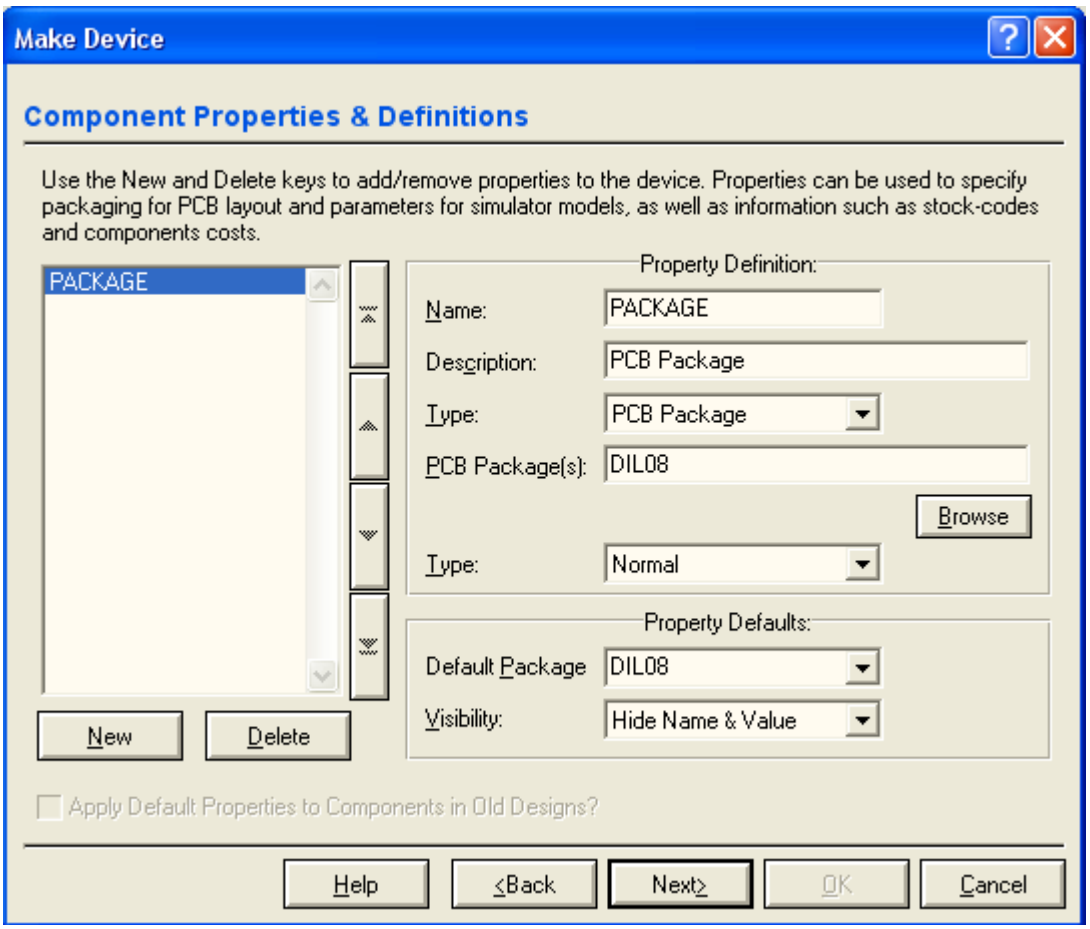


Рисунок 1.9 – Форма определения свойств по умолчанию компонента

Следующий шаг мастера – задание файла документации на компонент, а также связанный с компонентом раздел справки, можно пропустить.

На последнем этапе нужно задать пользовательскую библиотеку, в которую помещается компонент, а также заполнить сведения о категории элемента, его подкатегории, если это необходимо, сведения о производителе и вербальное описание компонента (рисунок 1.10).

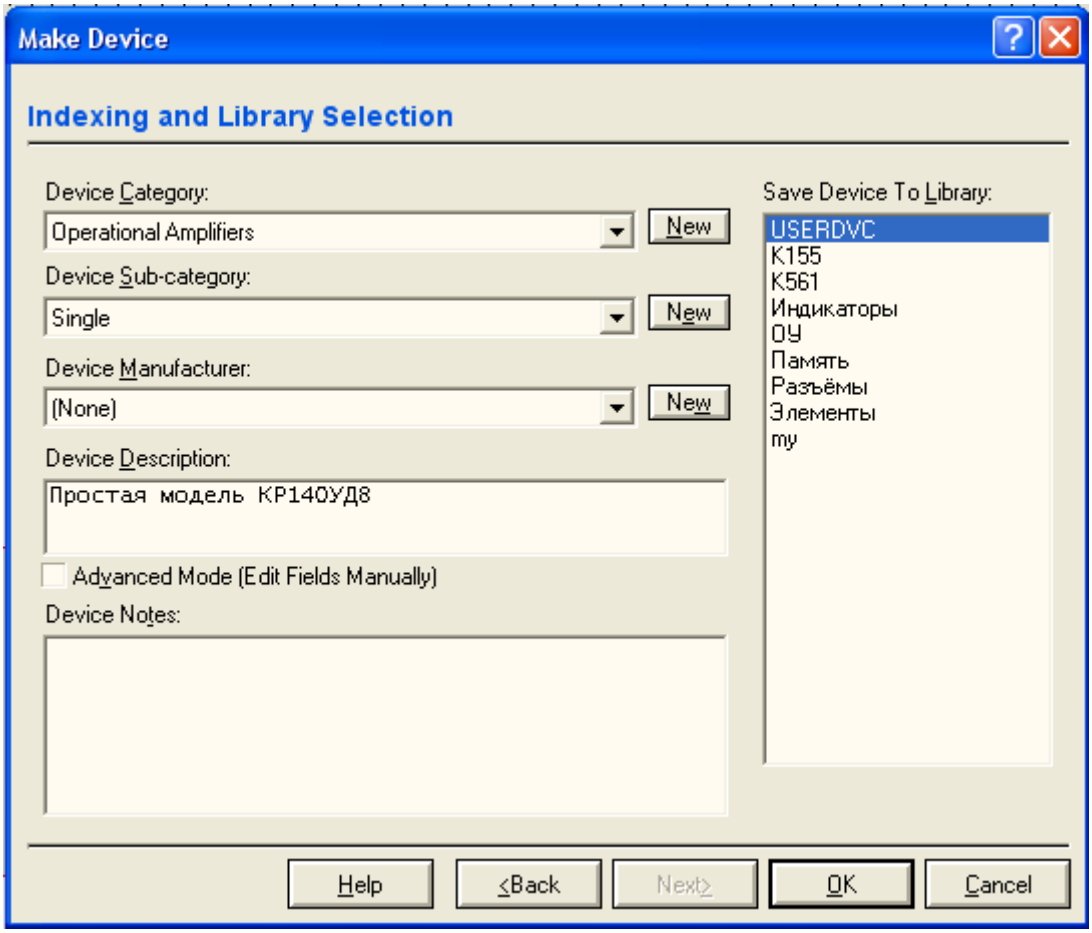


Рисунок 1.10 – Заполнение информации о библиотеке и категории компонента.

После заполнения необходимых полей этой последней формы, работа по оформлению компонента завершается. Компонент готов к употреблению!

2. Цель работы

Целью данной лабораторной работы является изучение особенностей построения УГО элементов в САПР Proteus согласно ЕСКД.

3. Порядок выполнения работы

3.1 Получить вариант компонента.

3.2 Найти техническую документацию на компонент.

3.3 Создать УГО компонента в системе Proteus, согласно ЕСКД.

3.4 Задать компоненту в соответствии с документацией соответствующий тип корпуса.

3.5 Записать результаты работы в пользовательскую библиотеку.

4. Содержание отчета должно соответствовать требованиям на техническую текстовую документацию по ГОСТ 2.105-95 и должно включать:

Титульный лист с названием и номером работы, а также с фамилией исполнителя;

Цель работы;

Задание на лабораторную работу;

Рисунки основных этапов построения компонента, аналогичные рисункам 4-10 данных методических указаний.

Выдержки из документации на компонент, подтверждающие выбранные решения (корпуса, нумерация выводов и т.п.).

Выводы.

## Лабораторная работа №6

## Программирование связи в операционных системах Win32 с микроконтроллерными устройствами по последовательному интерфейсу RS232C.

1 Краткие теоретические сведения

С последовательными и параллельными портами в Win32 работают как с файлами. Следовательно, начинать надо с открытия порта как файла - необходимо воспользоваться функцией CreateFile. Ее прототип выглядит так:

```
HANDLE CreateFile(
   LPCTSTR               lpFileName,
   DWORD                 dwDesiredAccess,
   DWORD                 dwShareMode,
   LPSECURITY_ATTRIBUTES lpSecurityAttributes,
   DWORD                 dwCreationDistribution,
   DWORD                 dwFlagsAndAttributes,
   HANDLE                hTemplateFile
);
```

Функция имеет много параметров, большинство из которых можно не использовать. Краткое описание параметров:

**lpFileName** - Указатель на строку с именем открываемого или создаваемого файла. Формат этой строки может быть очень хитрым. В частности, можно указывать сетевые имена для доступа к файлам на других компьютерах. Можно открывать логические разделы или физические диски и работать в обход файловой системы. Последовательные порты

имеют имена "COM1", "COM2", "COM3", "COM4" и так далее. Параллельные порты называются "LPT1", "LPT2" и так далее.

**dwDesiredAccess -** Задает тип доступа к файлу. Возможно использование следующих значений: 0 - Опрос атрибутов устройства без получения доступа к нему.

GENERIC_READ - Файл будет считываться.

GENERIC_WRITE - Файл будет записываться.

GENERIC_READ|GENERIC_WRITE - Файл будет и считываться и записываться.

**dwShareMode -** Задает параметры совместного доступа к файлу. Коммуникационные порты нельзя делать разделяемыми, поэтому данный параметр должен быть равен 0.

**lpSecurityAttributes -** Задает атрибуты защиты файла. Поддерживается только в Windows NT. Однако при работе с портами должен в любом случае равняться NULL.

**dwCreationDistribution** - Управляет режимами автосоздания, автоусечения файла и им подобными. Для коммуникационных портов всегда должно задаваться OPEN_EXISTING.

**dwFlagsAndAttributes -** Задает атрибуты создаваемого файла. Так же управляет различными режимами обработки. Для наших целей этот параметр должен быть или равным 0, или FILE_FLAG_OVERLAPPED. Нулевое значение используется при синхронной работе с портом, а FILE_FLAG_OVERLAPPED при асинхронной, или другими словами, при фоновой обработке ввода/вывода.

**hTemplateFile -** Задает описатель файла-шаблона. При работе с портами всегда должен быть равен NULL.

При успешном открытии файла, в нашем случае порта, функция возвращает описатель (HANDLE) файла. При ошибке INVALID_HANDLE_VALUE. Код ошибки можно получить путем вызова функции GetLastError.

Открытый порт должен быть закрыт перед завершением работы программы. В Win32 закрытие объекта по его описателю выполняет функция CloseHandle:

```
BOOL CloseHandle(HANDLE hObject);
```

Функция имеет единственный параметр - описатель закрываемого объекта. При успешном завершении функция возвращает не нулевое значение, при ошибке нуль.

Приведем пример открытия порта COM2:

```
#include <windows.h>
```

```
        . . .
        HANDLE port;
        . . .
    port=CreateFile("COM2",GENERIC_READ|GENERIC_WRITE,0,NULL
,OPEN_EXISTING,0,NULL);
        if(port==INVALID_HANDLE_VALUE) {
           MsgBox(NULL,"Не возможно открыть последовательный
порт","Error",MB_OK);
           ExitProcess(1);
        }
        . . .
        CloseHandle(port);
        . . .
```

В данном примере открывается порт COM2 для чтения и записи, используется синхронный режим обмена. Проверяется успешность открытия порта, при ошибке выводится сообщение и программа завершается. Если порт открыт успешно, то он закрывается.

Открыв порт, мы получили его в свое распоряжение. Теперь с портом может работать только наша программа. Однако, прежде чем мы займемся вводом/выводом, мы должны настроить порт. Это касается только последовательных портов, для которых мы должны задать скорость обмена, параметры четности, формат данных и прочее. Кроме того, существует несколько специфичных для Windows параметров. Речь идет о тайм-аутах, которые позволяют контролировать как интервал между принимаемыми байтами, так и общее время приема сообщения. Есть возможность управлять состоянием сигналов управления модемом.

Основные параметры последовательного порта описываются структурой DCB. Временные параметры структурой COMMTIMEOUTS. Настройка порта заключается в заполнении управляющих структур и последующем вызове функций настройки.

Основную информацию содержит структура DCB:

```
typedef struct _DCB {
   DWORD DCBlength; // sizeof(DCB)
   DWORD BaudRate;  // current baud rate
   DWORD fBinary:1; // binary mode, no EOF check
   DWORD fParity:1;    // enable parity checking
   DWORD fOutxCtsFlow:1;  // CTS output flow control
   DWORD fOutxDsrFlow:1;  // DSR output flow control
   DWORD fDtrControl:2;   // DTR flow control type
   DWORD fDsrSensitivity:1;  // DSR sensitivity
   DWORD fTXContinueOnXoff:1; // XOFF continues Tx
   DWORD fOutX:1;       // XON/XOFF out flow control
   DWORD fInX:1;         // XON/XOFF in flow control
```

```
    DWORD fErrorChar:1;   // enable error replacement
    DWORD fNull:1;        // enable null stripping
    DWORD fRtsControl:2;  // RTS flow control
    DWORD fAbortOnError:1;// abort reads/writes
                          // on error
    DWORD fDummy2:17;   // reserved
    WORD  wReserved;    // not currently used
    WORD  XonLim;       // transmit XON threshold
    WORD  XoffLim;      // transmit XOFF threshold
    BYTE  ByteSize;     // number of bits/byte, 4-8
    BYTE  Parity;       // 0-4=no,odd,even,mark,space
    BYTE  StopBits;     // 0,1,2 = 1, 1.5, 2
    char  XonChar;      // Tx and Rx XON character
    char  XoffChar;     // Tx and Rx XOFF character
    char  ErrorChar;   // error replacement character
    char  EofChar;      // end of input character
    char  EvtChar;      // received event character
    WORD  wReserved1;   // reserved; do not use
} DCB;
```

Эта структура содержит почти всю управляющую информацию, которая в реальности располагается в различных регистрах последовательного порта. Описание полей структуры:

**DCBlength** - Задает длину, в байтах, структуры DCB. Используется для контроля корректности структуры при передаче ее адреса в функции настройки порта.

**BaudRate** - Скорость передачи данных. Возможно указание следующих констант: CBR_110, CBR_300, CBR_600, CBR_1200, CBR_2400, CBR_4800, CBR_9600, CBR_14400, CBR_19200, CBR_38400, CBR_56000, CBR_57600, CBR_115200, CBR_128000, CBR_256000. Как видно, эти константы соответствуют всем стандартным скоростям обмена. На самом деле, это поле содержит числовое значение скорости передачи, а константы просто являются символическими именами. Поэтому можно указывать, например, и CBR_9600, и просто 9600. Однако рекомендуется указывать символические константы.

**fBinary** - Включает двоичный режим обмена. Win32 не поддерживает недвоичный режим, поэтому данное поле всегда должно быть равно 1, или логической константе TRUE (что предпочтительней). В Windows 3.1, если это поле было равно FALSE, включался текстовый режим обмена. В этом режиме поступивший на вход порта символ заданный полем EofChar свидетельствовал о конце принимаемых данных.

**fParity** - Включает режим контроля четности. Если это поле равно TRUE, то выполняется проверка четности, при ошибке, в вызывающую программу, выдается соответствующий код завершения.

**fOutxCtsFlow** - Включает режим слежения за сигналом CTS. Если это поле равно TRUE и сигнал CTS сброшен, передача данных приостанавливается до установки сигнала CTS. Это позволяет подключенному к компьютеру прибору приостановить поток передаваемой в него информации, если он не успевает ее обрабатывать.

**fOutxDsrFlow** - Включает режим слежения за сигналом DSR. Если это поле равно TRUE и сигнал DSR сброшен, передача данных прекращается до установки сигнала DSR.

**fDtrControl** - Задает режим управления обменом для сигнала DTR. Это поле может принимать следующие значения:

DTR_CONTROL_DISABLE - Запрещает использование линии DTR

DTR_CONTROL_ENABLE - Разрешает использование линии DTR

DTR_CONTROL_HANDSHAKE - Разрешает использование рукопожатия для выхода из ошибочных ситуаций. Этот режим используется, в частности, модемами при восстановлении в ситуации потери связи.

**fDsrSensitivity** - Задает чувствительность коммуникационного драйвера к состоянию линии DSR. Если это поле равно TRUE, то все принимаемые данные игнорируются драйвером (коммуникационный драйвер расположен в операционной системе), за исключением тех, которые принимаются при установленном сигнале DSR.

**fTXContinueOnXoff** - Задает, прекращается ли передача при переполнении приемного буфера и передаче драйвером символа XoffChar. Если это поле равно TRUE, то передача продолжается, несмотря на то, что приемный буфер содержит более XoffLim символов и близок к переполнению, а драйвер передал символ XoffChar для приостановления потока принимаемых данных. Если поле равно FALSE, то передача не будет продолжена до тех пор, пока в приемном буфере не останется меньше XonLim символов и драйвер не передаст символ XonChar для возобновления потока принимаемых данных. Таким образом, это поле вводит некую зависимость между управлением входным и выходным потоками информации.

**fOutX** - Задает использование XON/XOFF управления потоком при передаче. Если это поле равно TRUE, то передача останавливается при приеме символа XoffChar, и возобновляется при приеме символа XonChar.

**fInX** - Задает использование XON/XOFF управления потоком при приеме. Если это поле равно TRUE, то драйвер передает символ XoffChar, когда в приемном буфере находится более XoffLim, и XonChar, когда в приемном буфере остается менее XonLim символов.

**fErrorChar** - Указывает на необходимость замены символов с ошибкой четности на символ задаваемый полем ErrorChar. Если это поле равно TRUE, и поле fParity равно TRUE, то выполняется замена.

**fNull** - Определяет действие, выполняемое при приеме нулевого байта. Если это поле TRUE, то нулевые байты отбрасываются при передаче.

**fRtsControl** - задает режим управления потоком для сигнала RTS. Если это поле равно 0, то по умолчанию подразумевается RTS_CONTROL_HANDSHAKE. Поле может принимать одно из следующих значений:

RTS_CONTROL_DISABLE - Запрещает использование линии RTS

RTS_CONTROL_ENABLE - Разрешает использование линии RTS

RTS_CONTROL_HANDSHAKE - Разрешает использование RTS рукопожатия. Драйвер устанавливает сигнал RTS когда приемный буфер заполнен менее, чем на половину, и сбрасывает, когда буфер заполняется более чем на три четверти.

RTS_CONTROL_TOGGLE - Задает, что сигнал RTS установлен, когда есть данные для передачи. Когда все символы из передающего буфера переданы, сигнал сбрасывается.

**fAbortOnError** - Задает игнорирование всех операций чтения/записи при возникновении ошибки. Если это поле равно TRUE, драйвер прекращает все операции чтения/записи для порта при возникновении ошибки. Продолжать работать с портом можно будет только после устранения причины ошибки и вызова функции ClearCommError.

**fDummy2** - Зарезервировано и не используется.

**wReserved** - Не используется, должно быть установлено в 0.

**XonLim** - Задает минимальное число символов в приемном буфере перед посылкой символа XON.

**XoffLim** - Определяет максимальное количество байт в приемном буфере перед посылкой символа XOFF. Максимально допустимое количество байт в буфере вычисляется вычитанием данного значения из размера приемного буфера в байтах.

**ByteSize** - Определяет число информационных бит в передаваемых и принимаемых байтах.

**Parity** - Определяет выбор схемы контроля четности. Данное поле должно содержать одно из следующих значений:

EVENPARITY - Дополнение до четности

MARKPARITY - Бит четности всегда 1

NOPARITY - Бит четности отсутствует

ODDPARITY - Дополнение до нечетности

SPACEPARITY - Бит четности всегда 0

**StopBits** - Задает количество стоповых бит. Поле может принимать следующие значения:

ONESTOPBIT - Один стоповый бит

ONE5STOPBIT - Полтора стоповых бита

TWOSTOPBIT - Два стоповых бита

**XonChar** - Задает символ XON используемый как для приема, так и для передачи.

**XoffChar** - Задает символ XOFF используемый как для приема, так и для передачи.

**ErrorChar** - Задает символ, использующийся для замены символов с ошибочной четностью.

**EofChar** - Задает символ, использующийся для сигнализации о конце данных.

**EvtChar** - Задает символ, использующийся для сигнализации о событии.

**wReserved1** - Зарезервировано и не используется.

Так как поля структуры DCB используются для конфигурирования микросхем портов, на них накладываются некоторые ограничения. Размер байта должен быть 5, 6, 7 или 8 бит. Комбинация из пяти битного байта и двух стоповых бит является недопустимой. Так же, как и комбинация из шести, семи или восьми битного байта и полутора стоповых бит.

Структура DCB самая большая из всех, использующихся для настройки последовательных портов. Заполнение всех полей этой структуры может вызвать затруднения. Возможно воспользоваться функцией BuildCommDCB, которая позволяет заполнить поля структуры DCB на основе строки, по синтаксису аналогичной строке команды mode. Вот как выглядит прототип этой функции:

```
BOOL BuildCommDCB(LPCTSTR lpDef, LPDCB lpDCB);
```

Как видно, функция очень проста и имеет всего два параметра:

**lpDef** - Указатель на строку с конфигурационной информацией в формате команды mode. Например, следующая строка задает скорость 1200, без четности, 8 бит данных и 1 стоповый бит.

baud=1200 parity=N data=8 stop=1

**lpDCB -** Указатель на заполняемую структуру DCB. При этом структура должна быть уже создана и заполнена нулями, кроме поля DCBlength, которое должно содержать корректное значение. Возможно так же использование уже заполненной структуры DCB, например полученной вызовом одной из функций чтения параметров порта.

В случае успешного завершения, функция BuildCommDCB возвращает ненулевое значение. В случае ошибки возвращается 0.

Обычно функция BuldCommDCB изменяет только явно перечисленные в строке lpDef поля. Однако существуют два исключения из этого правила:

- При задании скорости обмена 110 бит в секунду автоматически устанавливается формат обмена с двумя стоповыми битами. Это сделано для совместимости с командой mode из MS-DOS или Windows NT.
- По умолчанию запрещается программное (XON/XOFF) и аппаратное управление потоком. Вы должны вручную заполнить требуемые поля DCB, если требуется управление потоком.

Функция BuildCommDCB поддерживает как новый, так и старый форматы командной строки mode. Однако Вы не можете смешивать эти форматы в одной строке.

Следует заметить, что функция BuildCommDCB только заполняет поля DCB указанными значениями. Это подготовительный шаг к конфигурированию порта, но не само конфигурирование, которое выполняется рассматриваемыми далее функциями. Поэтому Вы можете вызвать BuildCommDCB для общего заполнения структуры DCB, затем изменить значения не устраивающих Вас полей, и после этого вызывать функцию конфигурирования порта.

Заполнить DCB можно еще одним способом. Вызовом функции GetCommState. Эта функция заполняет DCB информацией о текущем состоянии устройства, точнее о его настройках. Вот как она выглядит:

```
BOOL GetCommState(
   HANDLE hFile,
   LPDCB  lpDCB );
```

Функция очень проста и имеет всего два параметра:

**hFile -** Описатель открытого файла коммуникационного порта. Этот описатель возвращается функцией CreateFile. Следовательно, прежде чем

получить параметры порта, Вы должны его открыть. Для функции BuildCommDCB это не требовалось.

**lpDCB** - Указатель на DCB. Для DCB должен быть выделен блок памяти.

При успешном завершении функция возвращает ненулевое значение. При ошибке нуль. Получить параметры порта можно в любой момент, а не только при начальной настройке.

Заполнив DCB можно приступать к собственно конфигурированию порта. Это делается с помощью функции SetCommState:

```
BOOL SetCommState(
   HANDLE hFile,
   LPDCB  lpDCB   );
```

Эта функция имеет точно такие же параметры, как GetCommState. Различается только направление передачи информации. GetCommState считывает информацию из внутренних управляющих структур и регистров порта, а SetCommState наоборот, заносит ее. Следует быть осторожным при вызове функции SetCommState, поскольку она изменит параметры даже в том случае, если очереди приема/передачи не пусты, что может вызвать искажение потока передаваемых или принимаемых данных.

Еще одна тонкость этой функции заключается в том, что она завершится с ошибкой, если поля XonChar и XoffChar в DCB содержат одинаковые значения.

В случае успешного завершения возвращается отличное от нуля значение, а в случае ошибки - нуль.

Приведем пример инициализации COM-порта:

```
//-------------------------------------
DCB dcb;         //Структуры состояния ком-порта
HANDLE hCom;            //Windows дескриптор
DWORD dwError;          //Переменная для ошибок
BOOL fSuccess;          //Флаг удачной операции

hCom = CreateFile("COM2",
    GENERIC_READ | GENERIC_WRITE,
    0,   /* без разделения ресурса-монопольно */
    NULL, /* без секьюрити атрибутов  */
    OPEN_EXISTING,   /*   порт   обязан   использовать
OPEN_EXISTING */
    0,     /* не многопотоковый (overlapped) ввод-вывод
*/
    NULL  /* hTemplate должен быть  NULL для портов */
    );
```

```
    if (hCom == INVALID_HANDLE_VALUE) {
        dwError = GetLastError();

        /* ошибка открытия порта */
       throw "Не могу открыть порт!"; }

    /*
     * Получаем текущую конфигурацию порта
     */

    fSuccess = GetCommState(hCom, &dcb);

    if (!fSuccess) {
        /* ошибка. */
      throw "Не могу определить состояние порта!";
    }

    /* Введем параметры: скорость=9600, 8 бит данных, нет
четности, 1 стоповый бит */
    dcb.BaudRate = 9600;
    dcb.ByteSize = 8;
    dcb.Parity = NOPARITY;
    dcb.StopBits = ONESTOPBIT;
    dcb.fRtsControl=0;
    dcb.fDtrControl=1;
    fSuccess = SetCommState(hCom, &dcb);
    if (!fSuccess) {
        /* ошибка. */
      throw "Не могу установить требуемые настройки порта!";
    }
    //---------------------------------------------
```

Прием и передача данных выполняется функциями ReadFile и WriteFile, то есть теми же самыми, которые используются для работы с дисковыми файлами. Вот как выглядят прототипы этих функций:

```
BOOL ReadFile(
   HANDLE       hFile,
   LPVOID       lpBuffer,
   DWORD        nNumOfBytesToRead,
   LPDWORD      lpNumOfBytesRead,
   LPOVERLAPPED lpOverlapped
);

BOOL WriteFile(
   HANDLE       hFile,
```

```
    LPVOID       lpBuffer,
    DWORD        nNumOfBytesToWrite,
    LPDWORD      lpNumOfBytesWritten,
    LPOVERLAPPED lpOverlapped
);
```

Описание параметров:

**hFile -** Описатель открытого файла коммуникационного порта.

**lpBuffer -** Адрес буфера. Для операции записи данные из этого буфера будут передаваться в порт. Для операции чтения в этот буфер будут помещаться принятые из линии данные.

**nNumOfBytesToRead, nNumOfBytesToWrite -** Число ожидаемых к приему или предназначенных к передаче байт.

**nNumOfBytesRead, nNumOfBytesWritten -** Число фактически принятых или переданных байт. Если принято или передано меньше данных, чем запрошено, то для дискового файла это свидетельствует об ошибке, а для коммуникационного порта совсем не обязательно. Причина в таймаутах.

**lpOverlapped -** Адрес структуры OVERLAPPED, используемой для асинхронных (многопоточных) операций. Для синхронных операций данный параметр должен быть равным NULL.

Коммуникационный порт не совсем обычный файл. Например, для него нельзя выполнить операцию позиционирования файлового указателя. С другой стороны, порт позволяет управлять потоком, что нельзя делать с обычным файлом.

Как правило, первой операцией, после открытия порта, является его сброс, который реализуется следующей функцией:

```
BOOL PurgeComm(
    HANDLE hFile,
    DWORD  dwFlags );
```

Вызов этой функции позволяет решить две задачи: очистить очереди приема/передачи в драйвере и завершить все находящиеся в ожидании запросы ввода/вывода. Какие именно действия выполнять задается вторым параметром (значения можно комбинировать с помощью побитовой операции OR:

PURGE_TXABORT - Немедленно прекращает все операции записи, даже если они не завершены

PURGE_RXABORT - Немедленно прекращает все операции чтения, даже если они не завершены

PURGE_TXCLEAR - Очищает очередь передачи в драйвере

PURGE_RXCLEAR - Очищает очередь приема в драйвере

Вызов этой функции нужен для отбрасывания мусора, который может находиться в приемном буфере на момент запуска программы, или как результат ошибки в работе устройства. Очистка буфера передачи и завершение операций ввода/вывода так же потребуются при ошибке, как процедура восстановления, и при завершении программы, для красивого выхода.

Следует помнить, что очистка буфера передачи, как и экстренное завершение операции записи, не выполняют передачу данных находящихся в этом буфере. Данные просто отбрасываются. Если же передача остатка данных необходима, то перед вызовом PurgeComm следует вызвать функцию:

```
BOOL FlushFileBuffers( HANDLE hFile );
```

Последовательный канал передачи данных можно перевести в специальное состояние, называемое разрывом связи. При этом передача данных прекращается, а выходная линия переводится в состояние "0". Приемник, обнаружив, что за время необходимое для передачи стартового бита, битов данных, бита четности и стоповых битов, приемная линия ни разу не перешла в состояние "1", так же фиксирует у себя состояние разрыва.

```
BOOL SetCommBreak( HANDLE hFile  );
BOOL ClearCommBreak(HANDLE hFile );
```

Следует заметить, что состояние разрыва линии устанавливается аппаратно. Поэтому нет другого способа возобновить прерванную, с помощью SetCommBreak, передачу данных, кроме вызова ClearCommBreak.

Более тонкое управление потоком данным позволяет осуществить функция:

```
BOOL EscapeCommFunction(  HANDLE hFile,
   DWORD  dwFunc   );
```

Выполняемое действие определяется вторым параметром, который может принимать одно из следующих значений:

CLRDTR – Сбрасывает сигнал DTR

CLRRTS - Сбрасывает сигнал RTS

SETDTR - Устанавливает сигнал DTR

SETRTS - Устанавливает сигнал RTS

SETXOFF - Симулирует прием символа XOFF

SETXON - Симулирует прием символа XON

SETBREAK - Переводит выходную линию передатчика в состояние разрыва.

CLRBREAK - Снимает состояние разрыва для выходной линии передатчика.

2. Цель работы: уметь программировать в операционных системах Win32 связь с микроконтроллерными устройствами по последовательному интерфейсу RS232C.

3. Порядок выполнения работы

3.1 Решить практическую задачу согласно своему варианту с использованием связи по последовательному интерфейсу RS232C.

3.2 Проверить работоспособность разработанной программы в системе Proteus или на лабораторном стенде.

4. Содержание отчета:

Титульный лист с названием и номером работы, а также с фамилией исполнителя

Цель работы

Задание на лабораторную работу

Схема спроектированной системы

Листинг программы.

Выводы.

5 Контрольные вопросы

5.1 Как по COM-порту передавать значения отсчетов 10 битного АЦП микроконтроллера PIC16F877?

5.2 С какой целью необходимо закрывать дескриптор (описатель) открытого порта?

5.3 Какими командами можно изменить скорость обмена COM порта?

5.4 Какими командами можно зажечь светодиод, подключенный к линии CTS?

5.5 Будет ли правильно воспринимать информацию приемник, если скорость COM-порта приемника отличается от скорости COM-порта передатчика на 10%?

5.6 Какие механизмы контроля правильности передаваемых данных есть в COM порту?

5.7 С какой целью используется функция сброса порта PurgeComm ?