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.

ЦИФРОВЫЕ ЭЛЕМЕНТЫ И МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ МЕДИЦИНСКОЙ ТЕХНИКИ

Методические рекомендации по выполнению лабораторных работ для студентов направления подготовки 12.03.04 – «Биотехнические системы и технологии» (бакалавр)

Курск 2023

УДК 621.(076.1)

Составители: А.А.Кузьмин

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

Цифровые элементы и микропроцессорные системы медицинской техники: методические рекомендации по выполнению лабораторных работ для студентов направления подготовки 12.03.04 – «Биотехнические системы и технологии» (бакалавр) / Юго-Зап. гос. ун-т; сост.: А.А.Кузьмин. - Курск, 2023. - 70 с.

Содержат методические рекомендации к проведению лабораторных работ по дисциплине «Цифровые элементы и микропроцессорные системы медицинской техники». 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 12.03.04 – «Биотехнические системы и технологии» (бакалавр)

Текст печатается в авторской редакции

Подписано в печать 25.09.23 Формат 60х84 1/16
Усо.печ.л. 4,1 . Уч.-изд.л. 3,7 .Тираж 30 экз. Заказ: 1077 .Бесплатно.
Юго-Западный государственный университет.
305040. г. Курск, ул. 50 лет Октября, 94.

Лабораторная работа №1
Система схемотехнического моделирования 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 7.1 показан на рисунке 1.1.

Подробнее опишем элементы интерфейса (рисунок 1.1):

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;

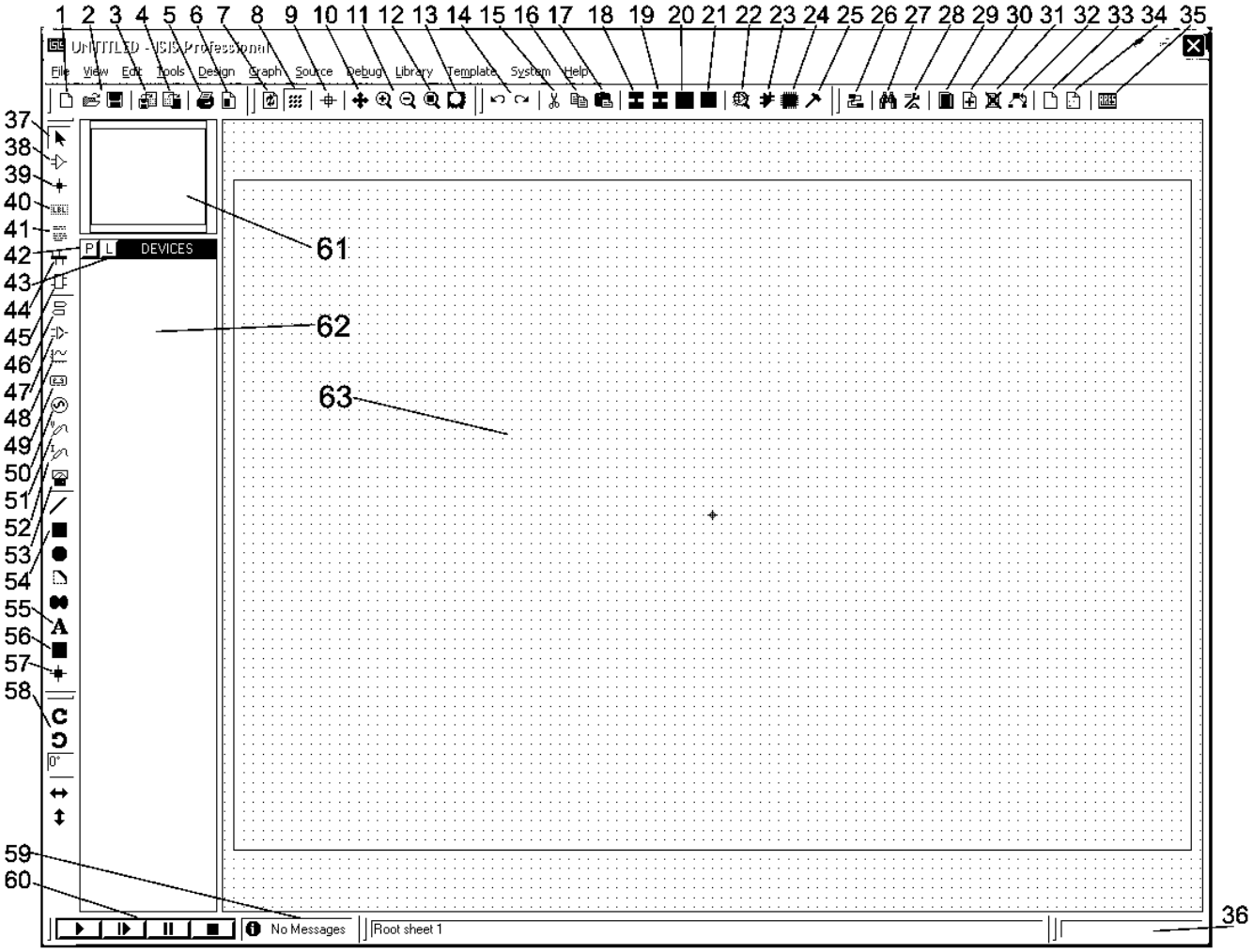


Рисунок 1.1 - Вид окна ISIS при запуске

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

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 1.2).

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 о ее отсутствии, PCB Preview – вид корпуса и контактных площадок.

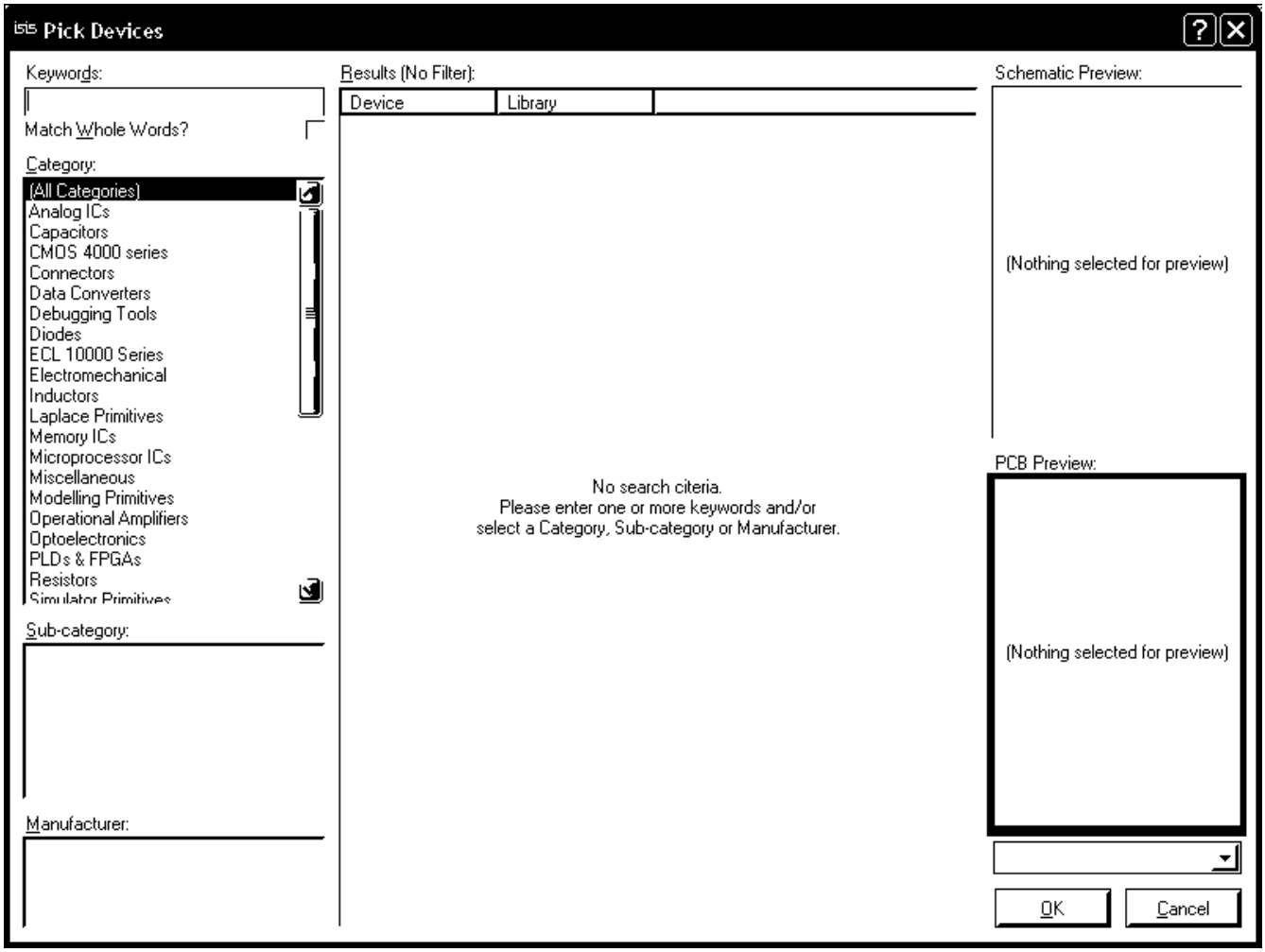


Рисунок 1.2 – Диалог поиска элемента

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 1.3. 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). 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). Затем заменяем 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.

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.

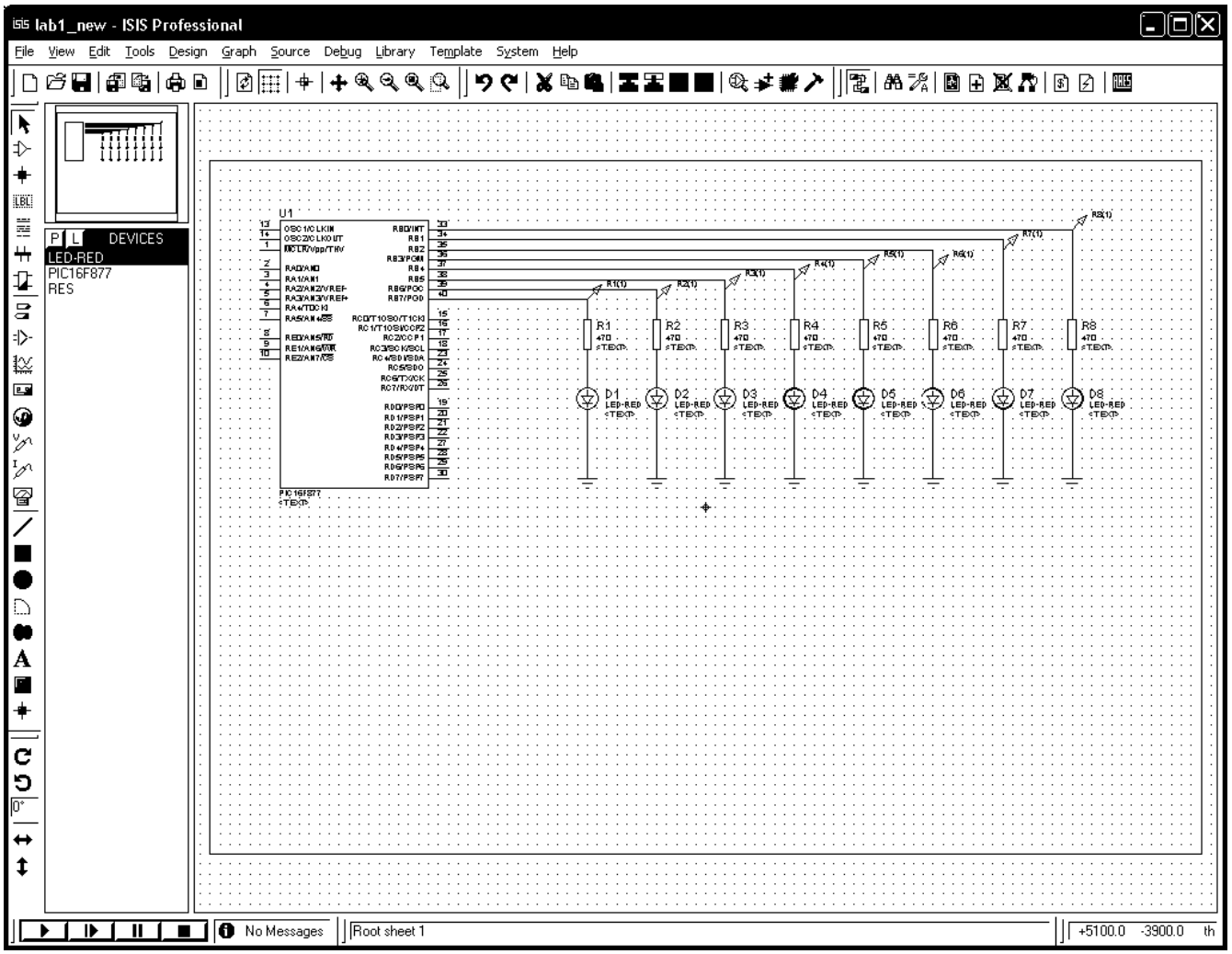


Рисунок 1.3 – Схема гирлянды в Proteus

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.
listp=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.
;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.
; Примечательно, что в каждом цикле одно
;и тоже количество выполняемых команд.
;Название: delay
;Входные данные: число,
;пропорциональное задержке в регистре cnt1.
;Выходные: задержка.
;Используемые регистры: cnt1, cnt2, cnt3.
```

```
; Delay 1sec@20MHz:
; 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
```

Сохраним проект (кнопка 2 Save Design). 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 1.4).

В поле Code Generation Tool выбираем ассемблер фирмы Microchip – MPASM, 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 1.5). 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.

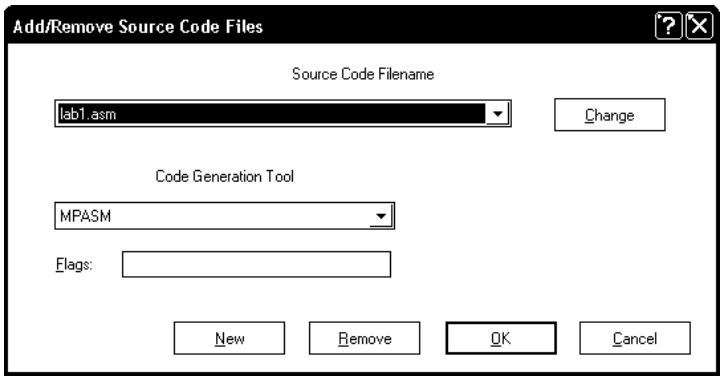


Рисунок 1.4 - Диалог редактирования свойств исходных программных файлов

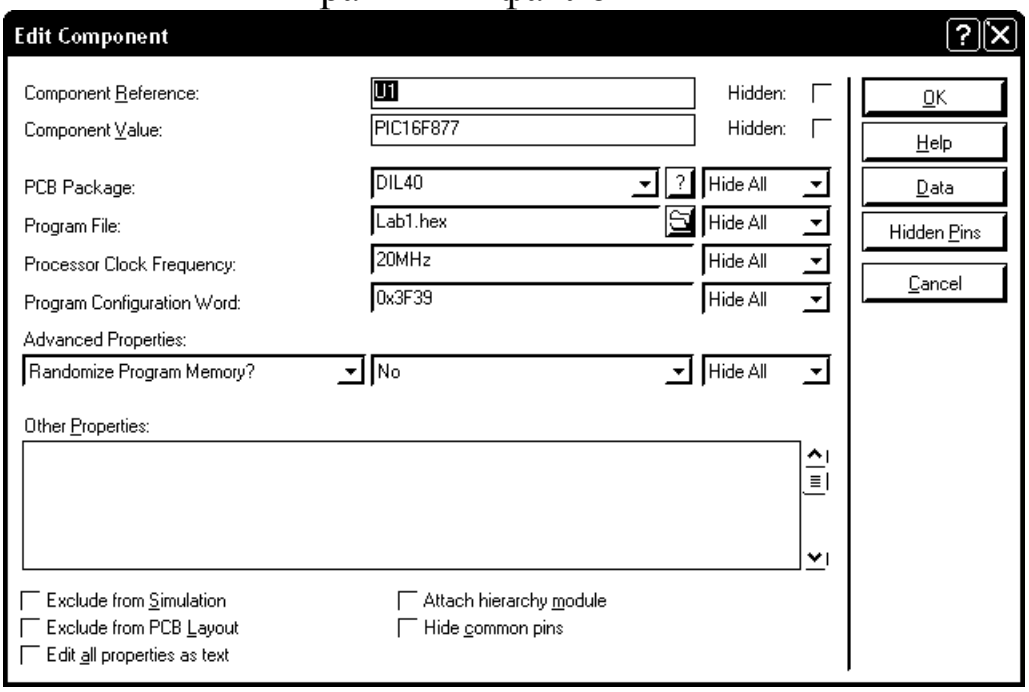


Рисунок 1.5 – Окно редактирования свойств микроконтроллера

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 конфликт, 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 1.6). 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;

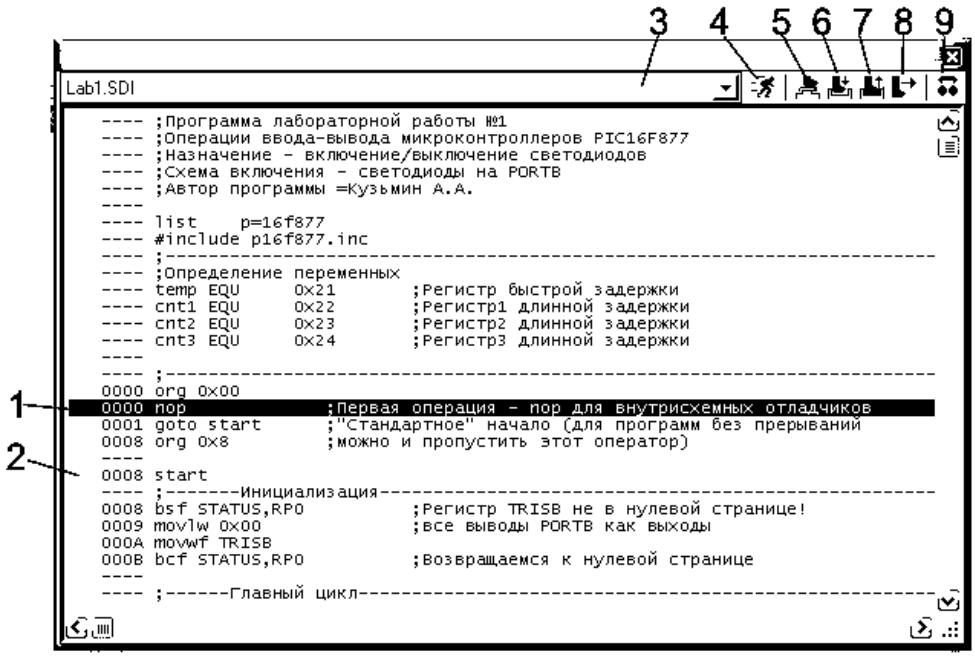


Рисунок 1.6 - Окно исходного кода программы PIC CPU Source Code

7 – кнопка выполнения программы до выхода из текущей процедуры;

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 1.7).

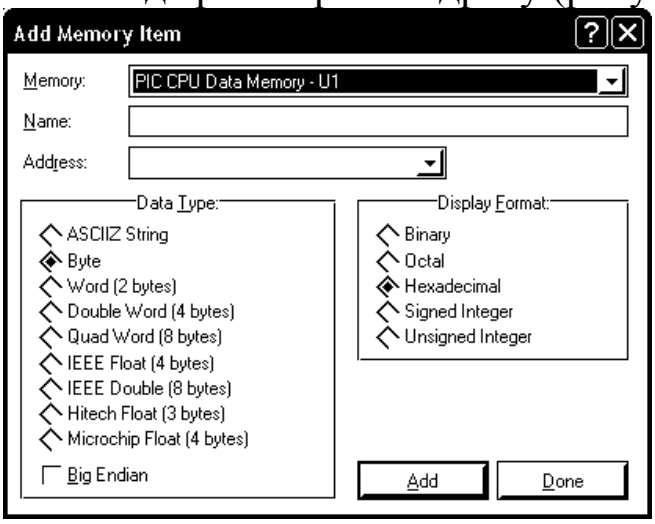


Рисунок 1.7 - Окно ввода регистра по адресу

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 1.8).

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.

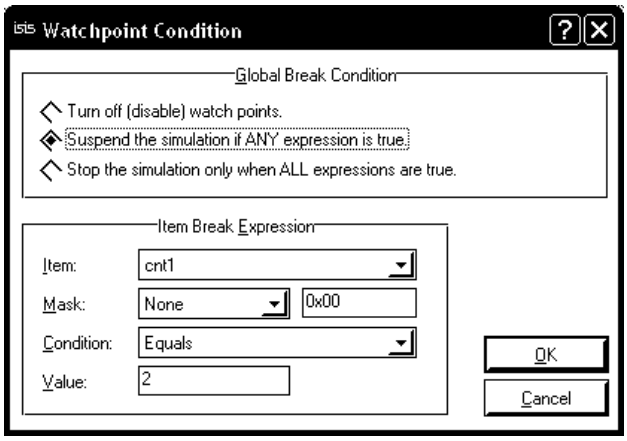


Рисунок 1.8 – Установка условной точки остановки

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 1.9).

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 рассчитана 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 1.10. Аналогично происходит и расчет аналоговых переходных процессов (график ANALOGUE).

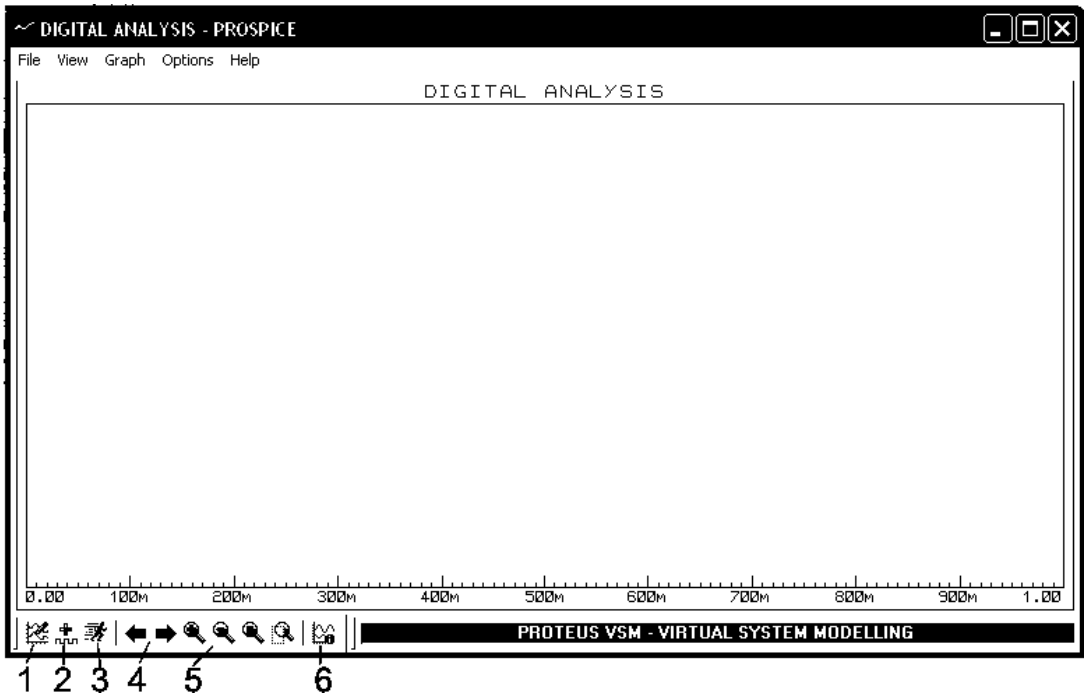


Рисунок 1.9 – Диалог расчета цифровых диаграмм DIGITAL ANALYSIS

4 – Кнопки навигации по графикам по оси времени.

5 – Кнопки выбора масштаба изображения.

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

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.

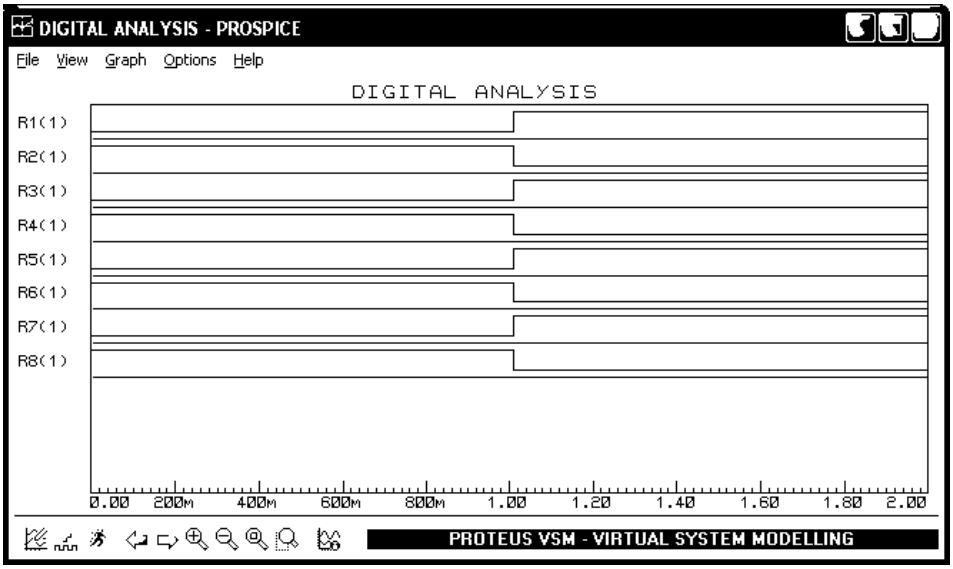


Рисунок 1.10 - Результат расчета цифровых диаграмм гирлянды

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 Почему при изменении номинала ограничивающего резистора изменяется яркость свечения светодиода?

5.8 Как меняются свойства компонентов в Proteus?

5.9 Как микроконтроллерам задаются программы, по которым они работают?

5.10 Как задается частота, на которой работает микроконтроллер?

5.11 Какие ошибки могут возникнуть при запуске симуляции схемы?

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

5.13 Какими элементами интерфейса управляется процесс отладки программ?

5.14 Как при пошаговой отладке отрабатывается выполнение процедур?

5.15 Как установить и снять точку останова?

5.16 Какие дополнительные отладочные окна поддерживает система Proteus?

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

5.18 Зачем нужны пробники напряжения в Proteus?

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

5.20 Какое различие между цифровыми и аналоговыми графиками переходных процессов?

Лабораторная работа №2
Организация ввода-вывода в микроконтроллерах Microchip. Проектирование шин.

1 Краткие теоретические сведения

Шина – это набор проводников, соединяющих выводы модулей в микропроцессорных системах. Модули в таких системах конструируются так, чтобы в один момент времени к шине был подключен только один активный выход, а все остальные выходы должны находиться в третьем (высокоимпедансном) состоянии. Модулями управляет центральный процессор (микроконтроллер) по отдельной шине управления. Разберем пример задания на проектирование шин и операций ввода-вывода по этим шинам.

Задание. Разработать на базе микроконтроллера PIC16 систему ввода трех байтовых переменных и поочередной индикации значений этих переменных через паузу на 8-ми светодиодах. Значения переменных задавать перемычками.

Решение. Итак, согласно заданию требуется 3*8=24 линии ввода для трех байтовых переменных и 8 линий вывода для светодиодов. Итого 32 линии. В микроконтроллере PIC16F877 как раз 32 линии ввода-вывода, однако, использование всех линий для решения этой задачи малоэффективно, так как такое решение оставляет мало путей для модернизации подобной системы. Например, подключение четвертой байтовой переменной потребует перестройки всей системы. Поэтому принимаем решение спроектировать мультиплексируемую шину данных, подключаемую на вход порта B. Т.е. в систему введем блок мультиплексоров, которые в зависимости от значений управляющих сигналов поочередно будут коммутировать на вход порта B значения трех байтовых переменных. Для задания значений переменных будем использовать комбинацию ключей с «подтягивающими» к напряжению питания (Pull up) резисторами (рисунок 2.1).

Блок резисторов, соединенных в одной точке, зададим компонентом Proteus «Respack-8». Общий вывод 1 резисторов соединим с напряжением питания. Перемычки зададим компонентом «Dipsw_8». Соединим выводы резисторов с ключами, как показано на рисунке

2.1. Тогда значение на втором выводе резистора (логический 0 или 1) будет зависеть от положения переключателя.

В качестве мультиплексоров выбираем счетверенные мультиплексоры «два к одному» с выходами с тремя состояниями «74LS257» (рисунок 2.2).

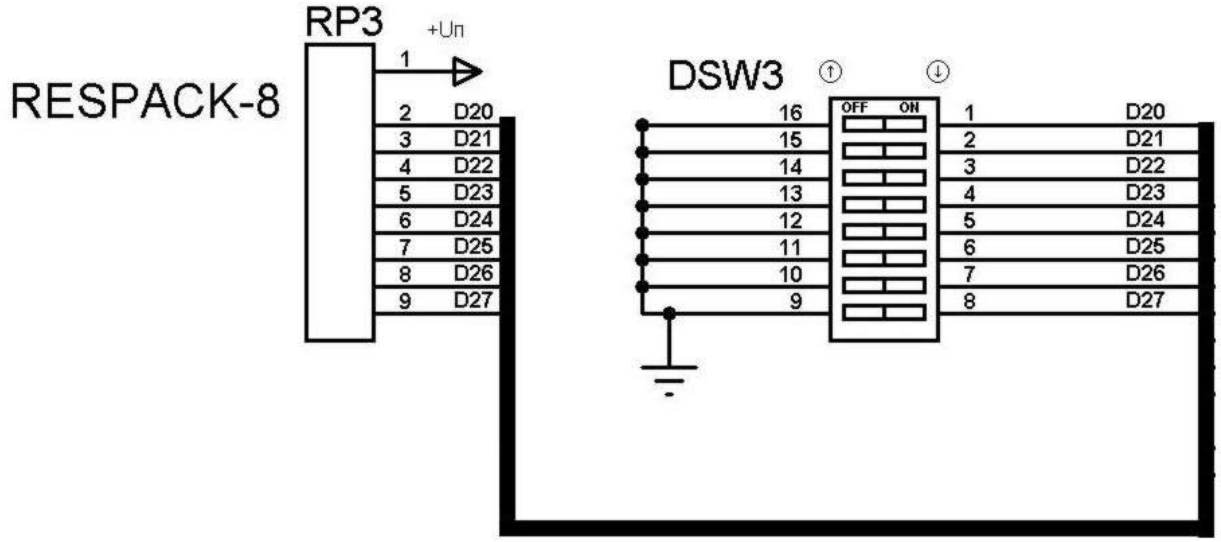


Рисунок 2.1 – Задание значений переменной перемычками с подтягивающими резисторами.

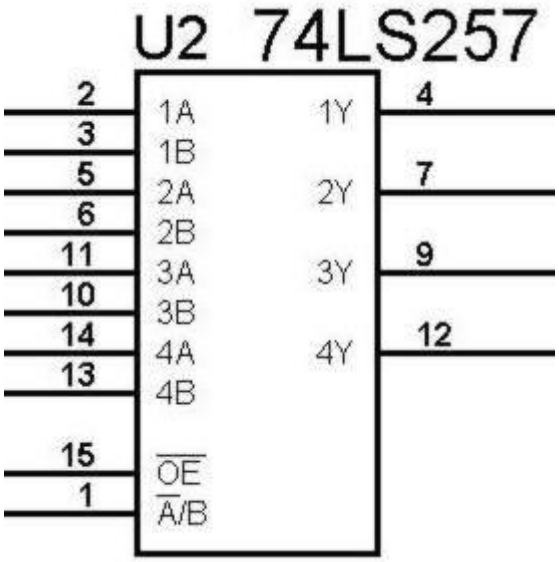


Рисунок 2.2 –Микросхема 74LS257

Эта микросхема имеет восемь входов для данных, четыре выхода и два управляющих входа. Вход 1 (AB) определяет какая группа входов 1A, 2A, 3A, 4A (при AB=0) или 1B, 2B, 3B, 4B (при AB=1)

будет коммутироваться на выходы 1Y, 2Y, 3Y, 4Y. Вход 15 (OE) определяет, находятся ли выходы микросхемы в третьем (высокоомном) состоянии при OE=1 (т.е. выходы отключены от общей шины), или же нет при OE=0 (т.е. выходы подключены к общей шине).

Для коммутирования двух байтовых значений необходимы две микросхемы 74LS257 с объединенными линиями управления. Для коммутирования третьей переменной необходимы еще две микросхемы 74LS257, которые в данном случае играют роль буферов-повторителей с третьим состоянием.

Управляющие линии мультиплексоров присоединим к выводам порта C микроконтроллера. Светодиоды подключим к выводам порта D. Общая схема электрическая принципиальная проектируемой системы показана на рисунке 2.3.

Листинг программы микроконтроллера приведен ниже.

```
;Программа лабораторной работы №2
;Операции ввода-вывода микроконтроллеров PIC16F877
;Назначение - включение/выключение светодиодов
;Схема включения - светодиоды на PORTD,
;мультиплексоры на порт B
;управление мультиплексорами на PORTC
;Автор программы =Кузьмин А.А.
listp=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 0xFF        ;все выводы PORTB как входы
movwf TRISB
bcf OPTION_REG,7 ;RBPU   разрешим "подтягивающие"
                 ;(PullUp) резисторы в порту B
movlw 0x00        ;все выводы PORTC как выходы
movwf TRISC
movlw 0x00        ;все выводы PORTD как выходы
movwf TRISD
bcf STATUS,RP0        ;Возвращаемся к нулевой странице
;------Главный цикл-------------------------------
loop
             ;Прочитаем значение x1 (DSW1)
bsf PORTC,2       ;отключим мультиплексоры U4, U5
                  ;от шины - установим OE1=1
bcf PORTC,1       ;Выберем нулевой регистр
                  ;мультиплексоров U2,U3 - AB0=0
bcf PORTC,0       ;подключим мультиплексоры U2, U3
                  ;к шине - установим OE0=0
nop     ;задержка для установления сигналов на линиях
movf PORTB,W      ;Читаем шину (порт B)
movwf PORTD       ;Зажигаем диоды
call delay        ;задержка

             ;Прочитаем значение x2 (DSW2)
bsf PORTC,2       ;отключим мультиплексоры U4, U5
                  ;от шины - установим OE1=1
bsf PORTC,1       ;Выберем первый регистр
                  ;мультиплексоров U2,U3 - AB0=1
bcf PORTC,0       ;подключим мультиплексоры U2, U3
                  ;к шине - установим OE0=0
nop     ;задержка для установления сигналов на линиях
movf PORTB,W      ;Читаем шину (порт B)
movwf PORTD       ;Зажигаем диоды
call delay        ;задержка

             ;Прочитаем значение x3 (DSW3)
bsf PORTC,0       ;отключим мультиплексоры U2, U3
                  ; от шины - установим OE0=1
bcf PORTC,2       ;Выберем нулевой регистр
                  ;мультиплексоров U4,U5 - AB1=0
bcf PORTC,2       ;подключим мультиплексоры U4, U5
                  ;к шине - установим OE1=0
```

```
nop          ;задержка для установления сигналов на лини-
ях
movf PORTB,W     ;Читаем шину (порт B)
movwf PORTD      ;Зажигаем диоды
call delay       ;задержка
goto loop
;-----------------------------------------
;-----Подпрограммы-------------------------
;-----------------------------------------
;Подпрограмма задержки.
;Описание – см. листинг лабораторной работы №1

delay           movlw .5
                movwf cnt1
                clrf cnt2
                clrf cnt3
dloop           decfsz cnt3,f
                goto $+2
                decfsz cnt2,f
                goto $+2
                decfsz cnt1,f
                goto dloop
                return
End
```

2. Цель работы

Целью работы является приобретение навыков в организации шин в микроконтроллерных устройствах, а также приобретение навыков в программировании операций ввода-вывода в микроконтроллерах Microchip.

3. Порядок выполнения работы

3.1 Разработать на базе микроконтроллера PIC16 систему ввода трех байтовых переменных, вычисления функции, согласно выданному варианту, и индикации значения этой функции на 8-ми светодиодах. Значения переменных задавать перемычками. Разрешается схемотехнически организовать систему, как показано на рисунке 2.3. Приветствуется разработка собственной системы.

3.2 Промоделировать работу микроконтроллера с разработанной программой в системе Proteus.

3.3 Составить контрольные примеры и вычислить при помощи ЭВМ контрольные результаты. По результатам проверить правильность выполнения разработанной программы.

4. Содержание отчета:

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

Цель работы

Задание на лабораторную работу

Схема спроектированной системы

Контрольные примеры, результат расчета контрольных примеров на ЭВМ и в системе Proteus.

Листинг программы.

Выводы.

5 Контрольные вопросы

5.1 Что произойдет, если к одному и тому же проводнику шины подключить два активных выхода?

5.2 Что означает термин «третье (Z) состояние»?

5.3 Что такое шина данных, шина адреса, шина управления?

5.4 Зачем используются подтягивающие к напряжению питания (pullup) и подтягивающие к земле (pulldown) резисторы?

5.5 Как работает микросхема 74LS257?

5.6 Какие команды в листинге программы примера манипулируют мультиплексорами?

5.7 Какие команды в листинге программы примера манипулируют светодиодами?

5.8 Какие проводники примера можно объединить в шину данных, а какие в шину управления?

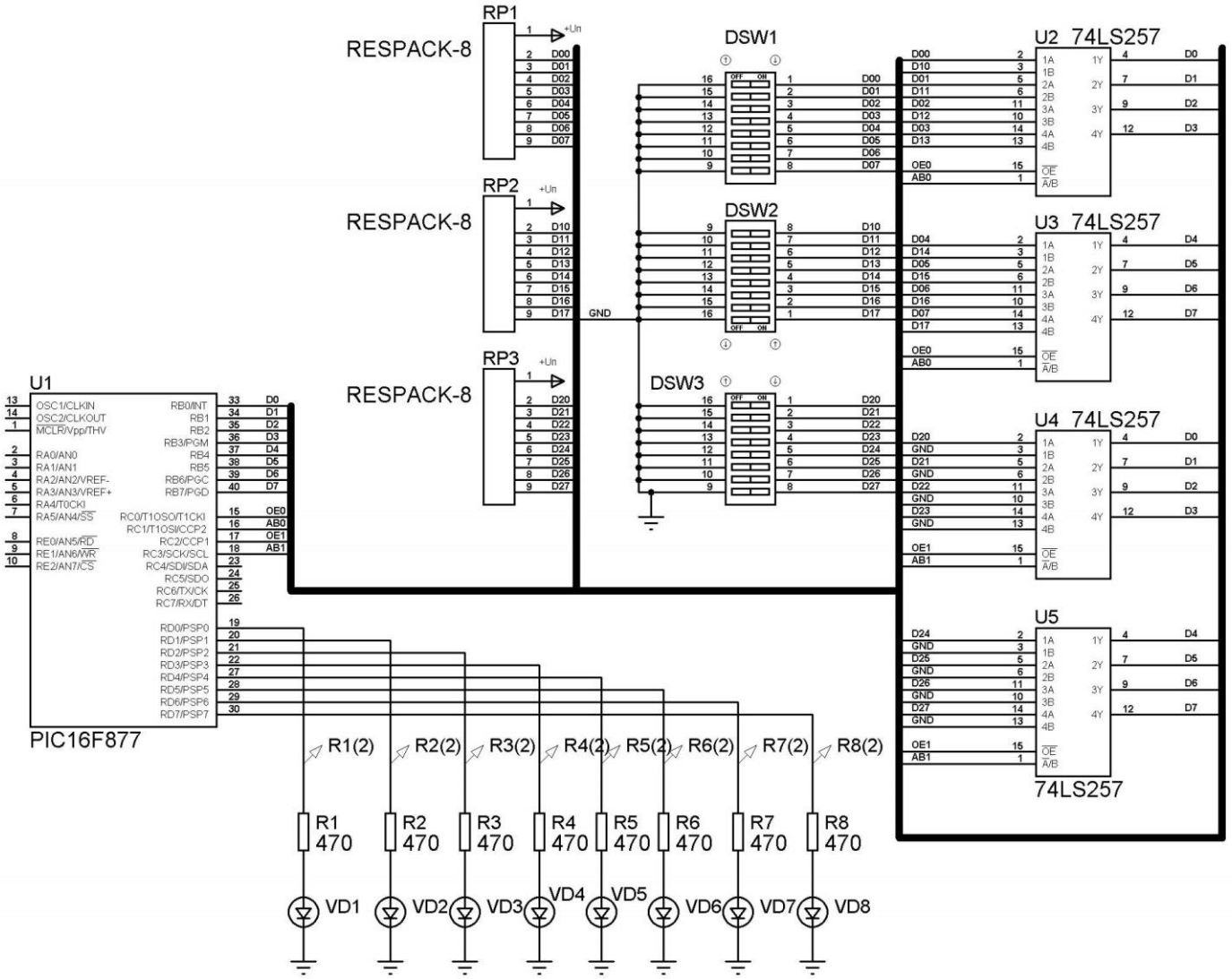


Рисунок 2.3 – Схема проектируемой системы

Лабораторная работа №3
Подключение ЖКИ к микроконтроллеру. Реализация протоколов обмена и сервисных процедур.

1 Краткие теоретические сведения

Одним из распространенных контроллеров по управлению графическими жидкокристаллическими индикаторами (ЖКИ) является контроллер SED1520 фирмы SEIKO EPSON (российский аналог КБ145ВГ4 производства ОАО «Ангстрем» www.angstrem.ru). На базе этого контроллера в России фирма МЭЛТ (www.melt.com.ru) производит жидкокристаллические индикаторы, одним из которых является индикатор МТ-6116, позволяющий отображать графическое поле из 61x16 точек. Если учесть, что потребляемый ток модуля не более 100 мкА, то таких возможностей вполне хватает для производства портативной медицинской аппаратуры.

Жидкокристаллический модуль МТ-6116 состоит из БИС контроллера управления и ЖК панели. Модуль позволяет:

- принимать команды с шины DB7-DB0 (перечень команд приведен в таблице 3.2);
- записывать данные в ОЗУ по 8-ми разрядной шине данных DB7-DB0;
- читать данные из ОЗУ на шину DB7-DB0;
- читать статус состояния на шину DB7-DB0 (таблица 3.2).
- управлять контрастностью и подсветкой.

Назначение внешних выводов модуля приведено в таблице 3.1. Временные диаграммы протокола обмена с модулем показаны на рисунке 3.1. Динамические характеристики модуля приведены в таблице 3.3.

Начальная установка модуля.

Для начальной установки модуля необходимо выполнить следующие действия:

1. после подачи напряжения питания удерживать вывод RES в состоянии логического "0" еще не менее 10 мкс;
2. подать перепад на вывод RES с логического "0" в логическую "1", длительность фронта не более 10 мкс;
3. ожидать сброса бита RESET в байте состояния или выждать не менее 2 мс;

4. подать команду снятия флага RMW(END);
5. подать команду включения обычного режима работы (Static Drive ON/OFF);
6. подать команду выбора мультиплекса (Duty Select);
7. подать команду включения дисплея (Display ON/OFF).

Таблица 3.1 - Назначение внешних выводов МТ-6116.

	Обозначе-	**Назначение**
1	**DB4**	Шина данных 4-й разряд
2	**DB5**	Шина данных 5-й разряд
3	**DB6**	Шина данных 6-й разряд
4	**DB7**	Шина данных 7-й разряд
5	**AO**	Выбор регистра данных / команд
6	**RD/WR**	Чтение/Запись
7	**E**	Строб разрешения чтения / записи
8	**DB3**	Шина данных 3-й разряд
9	**DB2**	Шина данных 2-й разряд
10	**DB1**	Шина данных 1 -й разряд
11	**DB0**	Шина данных 0-й разряд
12	**GND**	Общий контакт
13	**NC**	Не использовать
14	**Vcc**	Питание контроллера
15	**—**	— Питание подсветки
16	**+**	+ Питание подсветки
17	**RES**	Начальная установка

Таблица 3.2 – Команды модуля МТ-6116.

Команда	RD/WR	A0	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Функция
Display ON/OFF	0	0	1	0	1	0	1	1	1	0/1	Включает или выключает ЖКИ, независимо от данных в экранном ОЗУ и внутреннего состояния "1"-включить дисплей "0"-выключить дисплей
Display START Line	0	0	1	1	0	Display START Line (0...31)					Определяет строку ОЗУ , которая будет отображаться в вверхней строке ЖКИ (Стартовая строка ЖКИ).
Set Page	0	0	1	0	1	1	1	0	Page (0...3)		Устанавливает страницу ОЗУ в режиме адреса страницы (стр. 0...3)
Set Address	0	0	0	Column address (0...79)							Устанавливает столбец ОЗУ в режиме адреса столбца
Status Read	1	0	BUSY	ADC	ON/OFF	RESET	0	0	0	0	Чтение байта состояния индикатора BUSY: 1 – модуль занят внутренней обработкой; 0 – модуль готов к работе с внешним МП ADC: 1 – вывод прямых данных; 0 – вывод обратных данных ON/OFF: 1 – ЖКИ выключен; 0 – ЖКИ включен RESET: 1 – состояние сброса; 0 – нормальное состояние
Write Display Data	0	1	Write Data								Запись данных в ОЗУ модуля Эти команды выбирают ОЗУ по ранее заданному адресу, после чего адрес столбца инкрементируется
Read Display Data	1	1	Read Data								Чтение данных из ОЗУ модуля
ADC Select	0	0	1	0	1	0	0	0	0	0/1	Используется для изменения в обратном направлении соответствия между адресом столбца и позиции на индикаторе: 0 – прямое соответствие 1 – обратное соответствие
Static Drive ON/OFF	0	0	1	0	1	0	0	1	0	0/1	Выбор статического или нормального режима управления: 1 – статическое управление (малого потребления) 0 – обычное управление
Duty Select	0	0	1	0	1	0	1	0	0	0/1	Выбор мультиплекса: 0 – Для модуля МТ-6116
Read Modify Write	0	0	1	1	1	0	0	0	0	0	По этой команде устанавливается флаг RMW, после чего инкрементируется адрес счетчика столбца при записи данных в ОЗУ (и не инкрементируется при чтении)
END	0	0	1	1	1	0	1	1	1	0	Снятие флага RMW
RESET	0	0	1	1	1	0	0	0	1	0	Стартовая строка ЖКИ (Display Start Line) сбрасывается в 0, адрес страницы устанавливается равным 0, содержимое ОЗУ не изменяется

Таблица 3.3 - Динамические характеристики модуля

	Параметр		Обозн.	Min	Max
1	Время цикла, нс		TCYC	2000	-
2	Время установки адреса, нс		TAW	40	-
3	Время удержания адреса, нс		TAH	20	-
4	Время установки данных, нс		TDS	160	-
5	Время удержания данных, нс		TDH	20	-
6	Время задержки данных, нс		TDOH	20	120
7	Время доступа, нс		TACC	-	180
8	Длительность импульса разрешения, нс	Режим чтения	TEW	200	-
		Режим записи		160	-

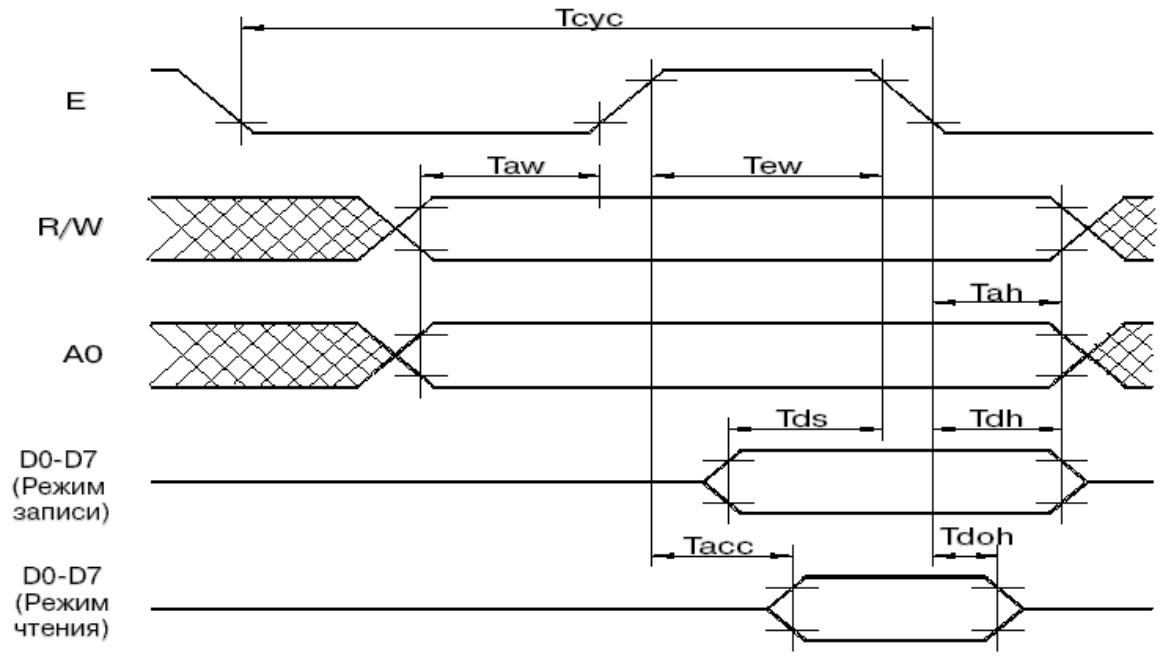


Рисунок 3.1 - Временные диаграммы протокола обмена ЖКИ

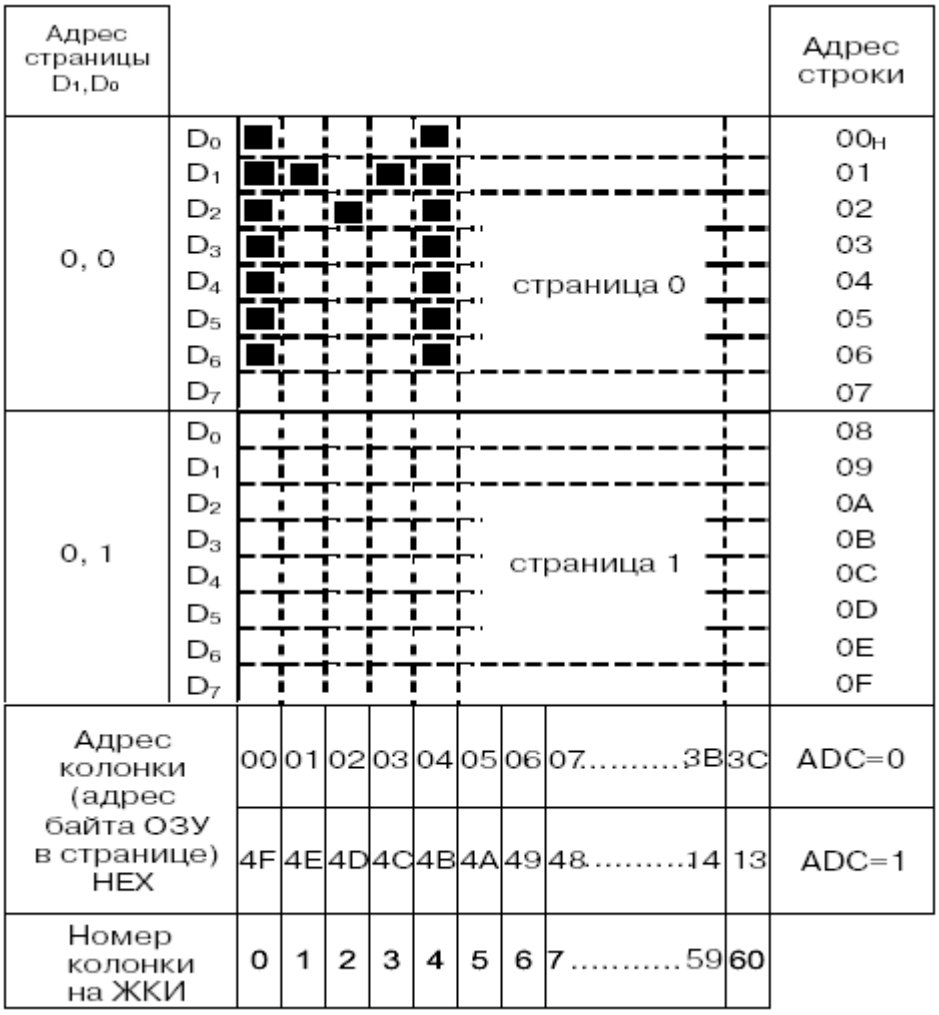


Рисунок 3.2 – Соответствие между ячейками ОЗУ модуля и отображаемыми точками на ЖКИ

Распределение ОЗУ. Модуль содержит ОЗУ для хранения данных, выводимых на ЖКИ, размером 80x32 бит. Все ОЗУ разбито на 4 страницы размером по 80x8 бит каждая. Каждая страница ОЗУ имеет организацию 80x8 бит. Каждой светящейся точке на ЖКИ соответствует логическая "1" в ячейке ОЗУ модуля. Соответствие между ячейками ОЗУ модуля и отображаемыми точками на ЖКИ показано на рисунке 3.2. На ЖКИ отображаются только 61 байт из 80 из каждой страницы. Одновременно отображается две страницы: верхние 8 точек по вертикали соответствуют нулевой странице, нижние 8 - первой (если при начальной установке была выбрана нулевая начальная строка отображения). Это можно изменить командой "Display START Line".

Режимы отображения. Модуль имеет два режима отображения информации из внутреннего ОЗУ: прямой и обратный. Он различается местоположением на ЖКИ первого отображаемого байта и направлением увеличения адреса во внутреннем ОЗУ при смещении отображаемой позиции на ЖКИ. В прямом режиме отображения адрес во внутреннем ОЗУ увеличивается при перемещении отображаемой позиции на ЖКИ вправо. В обратном режиме он наоборот уменьшается. Режим работы выбирается командой "ADC Select".

Чтение и запись данных. Чтение (запись) информации из (в) модуль осуществляется по страницам (80x8 бит или 80x1 байт). Каждая страница представлена как 80 байт. Страницы не пересекаются. Адреса с 80 по 127 не используются, в них невозможно ничего записать, а при чтении по этим адресам на шине данных может присутствовать любая информация. Для чтения или записи байта данных по произвольному адресу необходимо предварительно установить страницу ОЗУ и выбрать столбец внутри страницы ОЗУ. Это осуществляется командами "Set Page" и "Set Address" соответственно. После этого можно прочитать или записать байт данных. Одной команды "Set Page" недостаточно, так как она не изменяет адрес столбца. Для упрощения программ модуль поддерживает также непрерывную последовательность операций чтения или записи (а также их комбинацию): после чтения (записи) одного байта счетчик столбца автоматически увеличивается на 1 и модуль готов к новой операции чтения (записи) по следующему адресу без предварительной установки страницы ОЗУ и адреса столбца. Счетчик столбца считает только внутри одной страницы! При достижении адреса 79 следующим значением счетчика будет 80 и т.д., то есть не происходит ни перехода на следующую страницу, ни сброса счетчика в 0. Таким образом, после чтения (записи) последнего байта данных по адресу 79 модуль прекратит прием (выдачу) информации.

В режиме чтения информации после команд "Set Page" и "Set Address", необходимо однократно выполнить "пустую" операцию чтения, результат которой не использовать.

Модуль поддерживает специальный режим увеличения счетчика адреса столбца только при записи. Это удобно для изменения информации в ОЗУ модуля: можно сначала прочитать данные, изменить их и записать в модуль по тому же адресу (без повторной уста-

новки адреса столбца для операции записи). После операции записи будет выполнен переход к следующему байту данных. Этот режим включается командой "Read Modify Write" и выключается командой "END".

Вертикальное смещение отображаемой информации. Модуль поддерживает команду "Display START Line", устанавливающую номер самой верхней отображаемой строки. Это позволяет реализовать плавный сдвиг информации на ЖКИ по вертикали изменением номера первой отображаемой строки. Номер может быть в интервале от 0 до 31, что соответствует интервалу от первой строки нулевой страницы ОЗУ до последней строки третьей страницы ОЗУ. После отображения последней строки (31) будет отображаться снова нулевая строка.

Лабораторная установка включает в себя микроконтроллер PIC16F877-20I/QFP, ЖКИ MT-6116, четыре кнопки. Для построения модели лабораторной установки в системе Proteus версии 7.2 используется неполный аналог индикатора HDM32GS12-B, так как модели используемого индикатора MT-6116 в системе Proteus нет. ЖКИ HDM32GS12-B использует два контроллера SED1520 и отображает 122x32 точки. Выбор одного из двух контроллеров осуществляется стробами E1 и E2, т.е. второй контроллер можно игнорировать, если подавать на E2 логическую единицу. При этом ЖКИ HDM32GS12-B будет отображать 61x32 точки. Поэтому будем принимать во внимание для моделирования MT-6116 только первые 16 точек по вертикали.

Для модели HDM32GS12-B нет необходимости прибавлять смещение к адресу столбца 0x13 при режиме обратного отображения при ADC=1 в команде "ADC Select". Еще счетчик столбца у MT-6116 считает только внутри одной страницы и при переполнении счетчика переход на новую страницу не производится, а у HDM32GS12-B при переполнении счетчика столбца происходит переход на новую страницу. Модель HDM32GS12-B в системе Proteus версии 7.2 также некорректно отрабатывает (а точнее просто игнорирует) команду SetAdress с ненулевым аргументом. Для чтения из памяти данных в модели HDM32GS12-B приходится организовывать два цикла чтения – результат первого цикла чтения необходимо игнорировать, и один цикл чтения почему-то не сдвигает текущий столбец даже при вы-

ключенном режиме RMW. При использовании реального дисплея МТ-6116 «пустую» операцию чтения следует выполнять лишь однократно и только после применения команд SetPage или SetAdress.

Модель HDM32GS12-B позволяет, также как и МТ-6116, реализовывать команды включения-выключения дисплея (Display ON/OFF), записи данных в видеопамять (Write Display Data) и отображение их (за исключениями, обозначенных выше) на дисплее, изменение текущей страницы памяти (команда SetPage), сброс в ноль значения текущего столбца (SetAdress с нулевым аргументом). Также совпадают назначения всех выводов модулей (таблица 3.1), за исключением вывода E2.

Схема модели лабораторной установки показана на рисунке 3.3.

Шина данных контроллера SED1520 подключена к порту D микроконтроллера. Управляющие выводы ЖКИ подсоединены к следующим линиям микроконтроллера:

A0 к RC0;
RD/WR к RE1;
E к RC3;
RES к RE0.

Диаграммы ввода-вывода ЖКИ МТ-6116 показаны на рисунке 3.2. С учетом схемы подключения ЖКИ и диаграмм протокола ввода-вывода напишем процедуру выдачи сервисных команд (т.е. тех, которые не работают напрямую с видеопамятью):

```
;----------------- -------------------------
;Команды дисплея MT6116
;код команды - в регистре w
;(то, что выводится в PORTD)
commanddisp
  bcf PORTC,3      ;E=0 - как строб
  bcf PORTC,0      ;a0=0 - выбор регистра команд
  bcf PORTE,1      ;rd/wr =0 (запись)
  movwf PORTD      ;выводим команду
                   ;на шину данных
  call delay2      ;задержка не менее Tds
  bsf PORTC,3      ;E - как строб
  call delay2      ;задержка не менее Tew
  bcf PORTC,3      ;E - как строб
  call delay2      ;задержка не менее
```

```
                    ;Tdh
return

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

Пример использования этой подпрограммы: Выдадим команду на дисплей Display On. Согласно таблице команд при этом на шину данных необходимо подать комбинацию B’10101111’ (или 0xaf). Поэтому код будет следующим:

```
;команда display on
 movlw 0xaf
 call commanddisp
```

Процедура чтения из видеопамяти модели тогда будет выглядеть следующим образом:

```
;------------------------ --------------------
;Команды чтения данных дисплея HDM32GS12-B
;читается текущий столбец,
;затем текущим становится следующий столбец
;  (при выключенном режиме дисплея RMW)
;Данные - в регистре videobyte
;Только командами чтения удается
;установить нужный столбец
;в модели HDM32GS12-B!!!
readdata
  bsf STATUS,RP0   ;Переключаем порт D для чтения
  movlw 0xff
  movwf TRISD
  bcf STATUS,RP0
  ;-*-*-*-*-*-*-*
  bcf PORTC,3      ;E - как строб
  bsf PORTC,0      ;a0 - выбор регистра команд
  bsf PORTE,1      ;rd/wr =1 (чтение)

  call delay2
  bsf PORTC,3      ;E - как строб
```

```
  call delay2
  movf PORTD,w
 ; movwf videobyte ;Этот цикл - пустой!
  bcf PORTC,3      ;E - как строб
  call delay2
  bcf PORTC,3      ;E - как строб
  bsf PORTC,0      ;a0 - выбор регистра команд
  bsf PORTE,1      ;rd/wr =1 (чтение)
  ;-*-*-*-*-*-*-*-*
       ;И еще раз все читаем!!! -
      ;только так все работает
  call delay2
  bsf PORTC,3      ;E - как строб
  call delay2
  movf PORTD,w
  movwf videobyte  ;записываем ответ
  bcf PORTC,3      ;E - как строб
  call delay2
  ;-*-*-*-*-*-*-*-*
  bsf STATUS,RP0   ;Переключаем порт D для записи
  movlw 0x00
  movwf TRISD
  bcf STATUS,RP0
return
```

При использовании реального устройства «пустой» цикл не используют.

Суммируя все различия между моделью и лабораторным стендом, можно сформулировать несколько простых правил для написания программ:

1 Для нормального отображения информации на лабораторном стенде и в модели необходимо включать режим ADC=1 в команде "ADC Select". Для обратного отображения необходимо выключать режим ADC=0 в команде "ADC Select".

2 К адресу столбца на лабораторном стенде необходимо прибавлять константу 0x13 (например, в командах Set Address). В модели ничего прибавлять не надо.

Смещение и значение команды "ADC Select" рекомендуется определять вначале листинга с помощью директивы #define и менять их в зависимости от того, на какую платформу будет устанавливаться программа:

```
#define correct 0x0  ;0x13-смещение адреса для MT6116.
                     ; =0 для моделирования в протеусе
#define adcbyte 0xa1 ;Значение ADC. 0xa1 для прямого
                     ; отображения. 0xa0 для обратного
```

3 Программно следить за переполнением значения текущего столбца.

4 При моделировании в Proteus заменять команду SetAdress=N на команду SetAdress =0 и команду сдвига на N позиций, которую можно реализовать, например, так:

```
;-----------------------------------------
;Команды изменения (приращения)
;номера текущего столбца в HDM32GS12-B
;Работает также и в MT6116, но сдвиг
;получается в два раза больше!
;основана на readdata (readdata заключена в цикл)
;Приращение - в регистре sdvig
sdvigstolb
  bsf STATUS,RP0   ;Переключаем порт D для чтения
  movlw 0xff
  movwf TRISD
  bcf STATUS,RP0
loopsdvig
  bcf PORTC,3      ;E - как строб
  bsf PORTC,0      ;a0 - выбор регистра команд
  bsf PORTE,1      ;rd/wr =1 (чтение)
  call delay2
  bsf PORTC,3      ;E - как строб
  call delay2
  movf PORTD,w
 ; movwf videobyte ;Этот цикл - пустой!
  bcf PORTC,3      ;E - как строб
  call delay2
  bcf PORTC,3      ;E - как строб
  bsf PORTC,0      ;a0 - выбор регистра команд
  bsf PORTE,1      ;rd/wr =1 (чтение)
       ;И еще раз все читаем!!! -
      ;только так все работает
  call delay2
  bsf PORTC,3      ;E - как строб
  call delay2
  movf PORTD,w
  movwf videobyte  ;записываем ответ
```

```
    bcf PORTC,3      ;E - как строб
    call delay2
    bsf STATUS,RP0   ;Переключаем порт D для записи
    movlw 0x00
    movwf TRISD
    bcf STATUS,RP0
    decfsz sdvig
    goto loopsdvig
  return
```

2. Цель работы

Целью работы является приобретение навыков в сопряжении различных контроллеров по заданному физическому и логическому интерфейсу, а также в закреплении навыков в программировании операций ввода-вывода в микроконтроллерах Microchip.

3. Порядок выполнения работы

3.1 Решить практическую задачу по сопряжению и программированию внешнего контроллера, обеспечивающего графический вывод на LCD дисплей, согласно своему варианту.

3.2 Проверить работоспособность разработанной программы в системе Proteus, обращая внимание на неполное соответствие компьютерной модели ее реальному аналогу.

3.3 Проверить работоспособность разработанной программы 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

Контрольные примеры, результат выполнения микроконтроллером контрольных примеров.

Листинг программы.

Выводы.

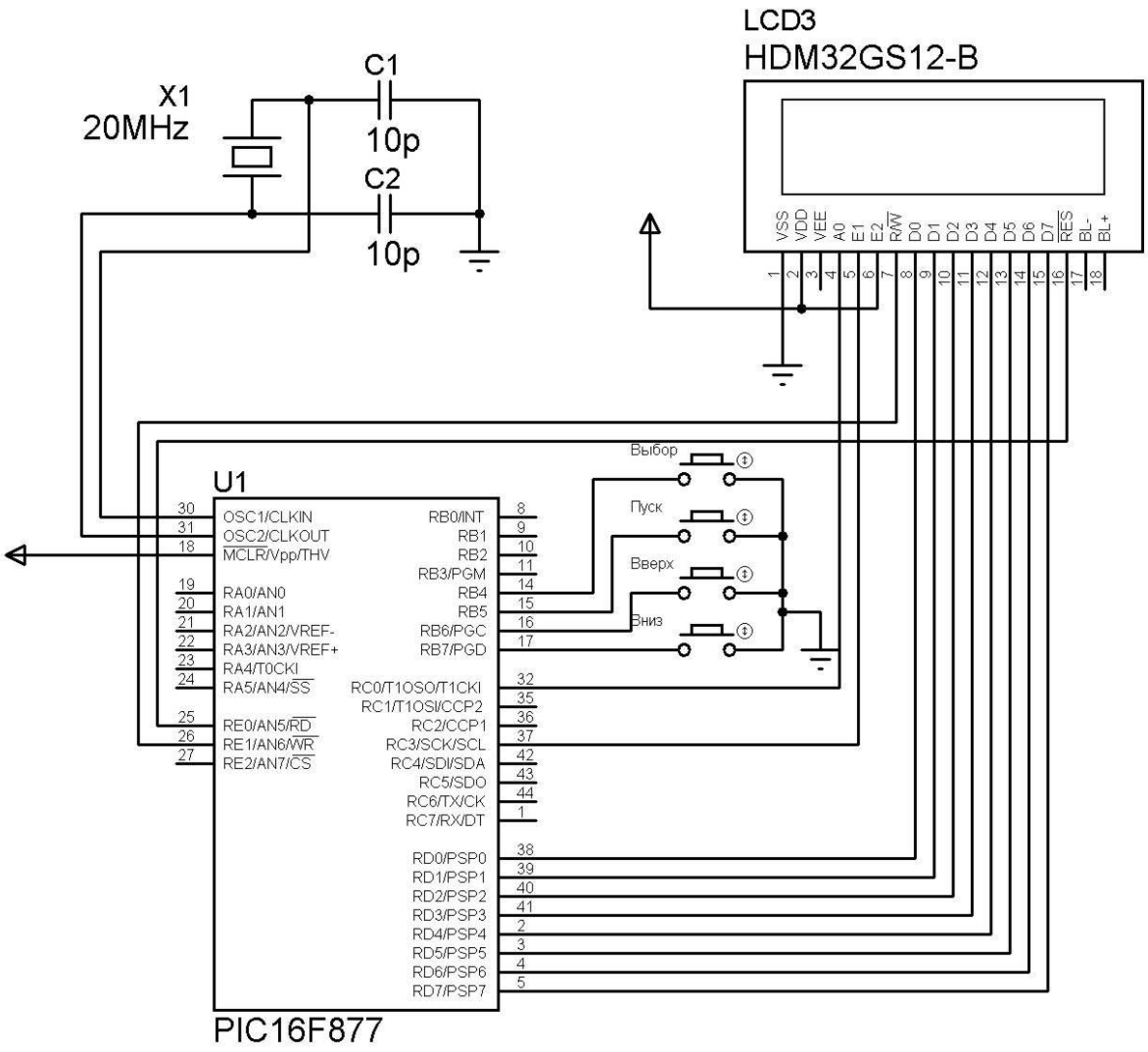


Рисунок 3.3 - Схема модели лабораторной установки

5 Контрольные вопросы

5.1 От чего зависит направление передачи информации в линиях DB0..DB7 модуля MT6116?

5.2 Какие события происходят после изменения уровня с нулевого на единичный строба разрешения чтения/записи E? Какие события происходят после изменения уровня с единичного на нулевой этого строба?

5.3 Напишите подпрограммы необходимых для инициализации модуля MT-6116 задержек для частоты работы микроконтроллера 20 МГц.

5.4 Рассчитайте с какой максимальной частотой может обновляться все содержимое индикатора MT-6116.

5.5 Поясните смысл команды ADC Select.

5.6 Напишите последовательность команд контроллера ЖКИ для вывода на индикатор в заданной позиции горизонтальной линии на весь экран.

5.7 Напишите последовательность команд контроллера ЖКИ для вывода на индикатор в заданной позиции вертикальной линии на весь экран.

5.8 Сколько страниц ОЗУ поддерживает контроллер модуля MT-6116?

5.9 Содержимое каких страниц выводится на индикатор модуля MT-6116?

Лабораторная работа №4
Подключение микросхем ЦАП к микроконтроллеру.

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 4.1 приведена функциональная схема и обозначение выводов микросхемы MAX503.

Назначение выводов микросхемы MAX503 приведено в таблице 4.1. На рисунке 4.2 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 4.3.

На рисунке 4.4 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).

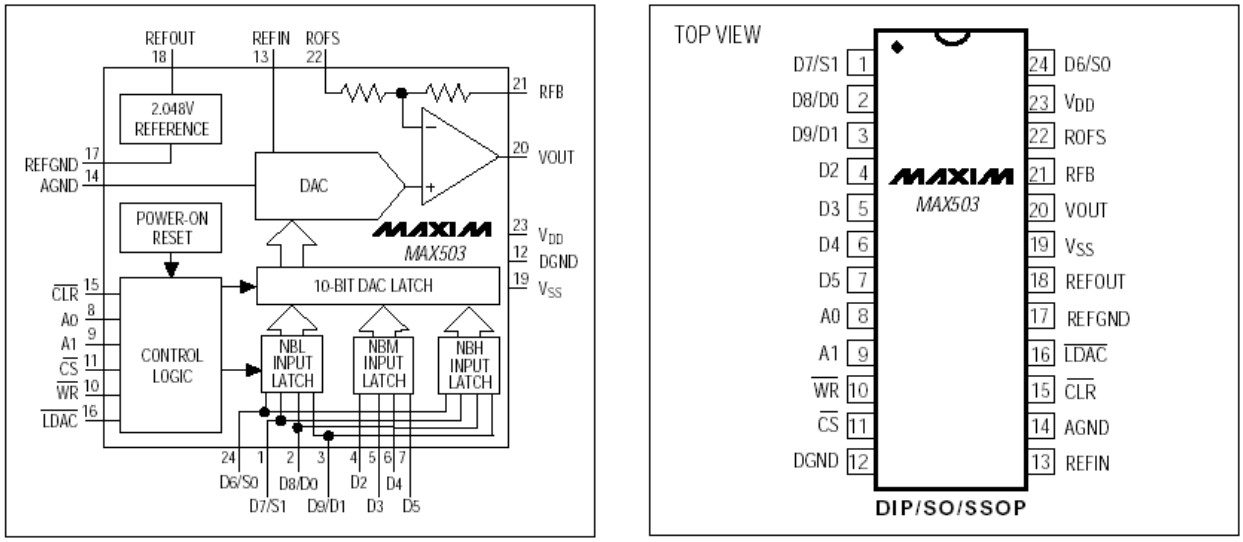


Рисунок 4.1 - Функциональная схема и обозначение выводов микросхемы MAX503

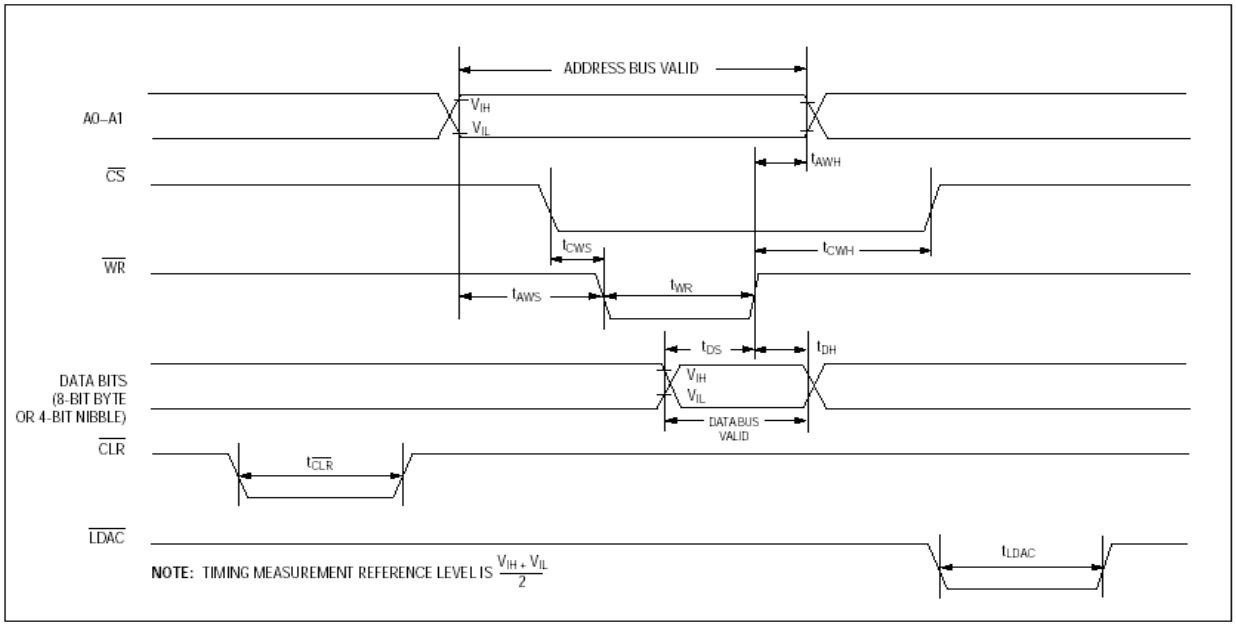


Рисунок 4.2 - Временная диаграмма циклов записи в MAX503

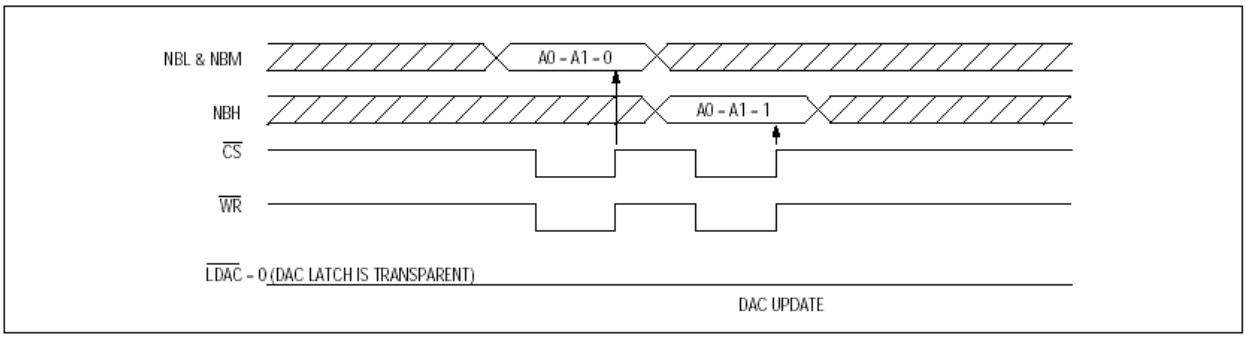


Рисунок 4.3 – Диаграмма обмена информацией при «прозрачных» защелках регистра ЦАП (LDAC=0). При режиме записи NBL и NBM во входной регистр записываются 8 младших бит информации, при режиме записи NBH записываются два старших бита.

Листинг 4.1 – 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 передаются 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
```

Таблица 4.1 - 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.

2 Цель работы: уметь находить и пользоваться документацией на иностранные 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?

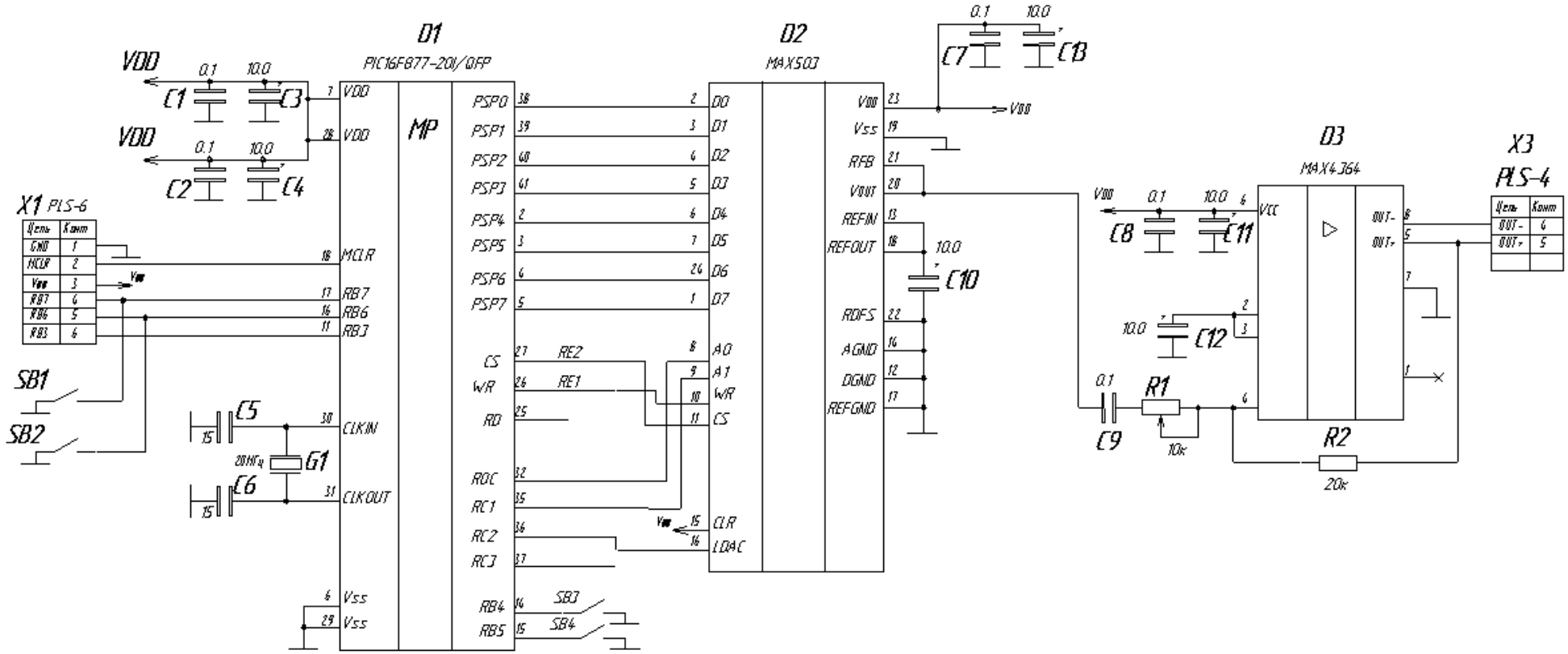


Рисунок 4.4 – Схема включения MAX503 в лабораторном макете.

Лабораторная работа №5
Подключение клавиатуры к микроконтроллеру.

1 Краткие теоретические сведения

Считывать состояние кнопок, выключателей довольно просто. Достаточно подсоединить их между входом и землей, как показано на рисунке 5.1. Вход подтянут к высокому логическому уровню ограничительным резистором, величина сопротивления которого может доходить до 100 кОм, чем обеспечивается малое потребление тока.

В том случае, если ваше устройство работает вблизи мощного источника помех (например, двигателя), желательно использовать резистор с небольшим сопротивлением (обычно 4,7 кОм или 10 кОм). Наводки на высокоомный вход будут значительно большими, чем на низкоомный. Однако включать в схему резисторы с еще меньшими значениями сопротивления целесообразно только в особых случаях.

Когда контакты выключателя разомкнуты, на входе будет высокий логический уровень, при замыкании контактов - низкий.

Все механические выключатели имеют одно негативное свойство, известное как «дребезг контактов», которое обусловлено колебаниями упругих контактов при их замыкании и размыкании. Длительность колебаний составляет всего несколько миллисекунд. При этом вместо «чистого» прямоугольного импульса (рисунок 5.2а) получается искаженный импульс или пачка импульсов (рисунок 5.2б).

Обычно такой недостаток устраняют с помощью RS-триггеров, одновибраторов или интегрирующих R-C цепочек, устанавливаемых перед триггерами Шмидта. В устройствах на базе микроконтроллеров борьбу с «дребезгом контактов» возлагают на программу, которая осуществляет многократное считывание состояния входа, подключенного к переключателю, определяя момент устойчивого изменения его состояния.

Если клавиатура состоит из нескольких клавиш, то они могут быть подключены к микроконтроллеру как отдельные кнопки, то есть каждая через свой порт. Если клавиатура большая, необходимо искать другое решение, поскольку портов у микроконтроллера не слишком много. Здесь возможны два варианта.

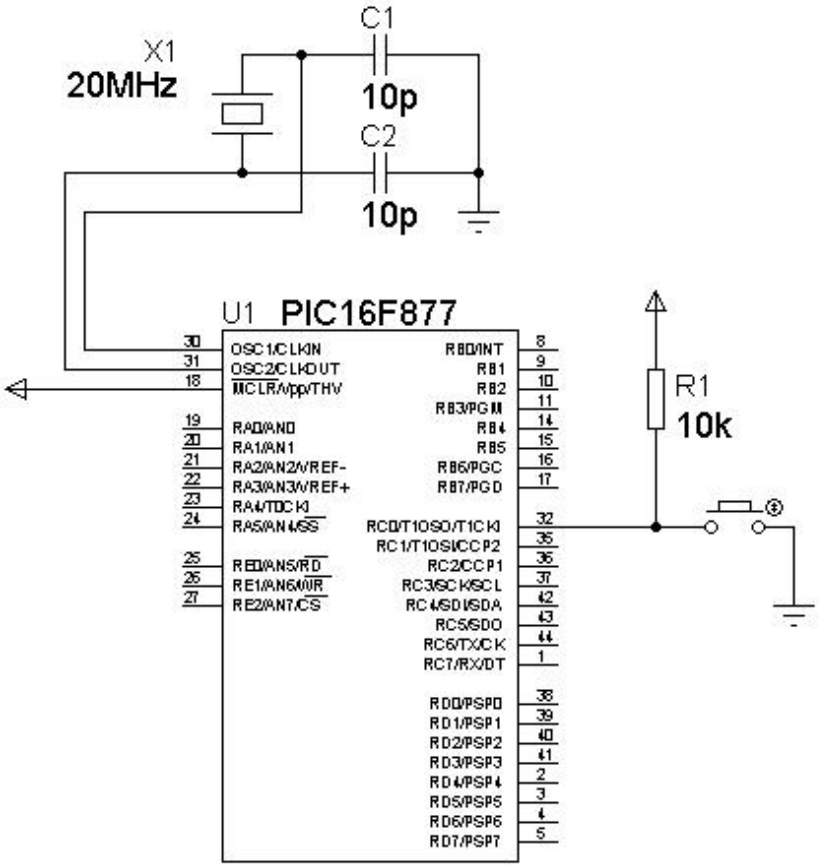


Рисунок 5.1 – Подключение кнопки к линии порта ввода-вывода

а) б)

Рисунок 5.2 – Явление дребезга контактов. а) – теоретический сигнал; б) – реальный сигнал

Первый состоит в том, чтобы применить внешний клавиатурный кодер, который подключается к N клавишам, а на выходе формирует M-разрядный код, причем $N = 2^M$. Таким образом, шестнадцать клавиш могут быть закодированы четырьмя битами.

На рисунке 5.3 показан пример использования подобной микросхемы, а именно ИС типа MM74C922 фирмы National Semiconductor. К ее входу подключается клавиатура из шестнадцати клавиш. Данная ИС осуществляет постоянное сканирование клавиш, устраняет влияние «дребезга контактов» и кодирует состояние клавиатуры. Выходной код снимается с выходов D, C, B и A. Нажатие одной из клавиш и наличие кодированных данных на выходе индицируется высоким уровнем сигнала на

выходе DA. Принцип взаимодействия кодера с микроконтроллером очень прост. Функция микроконтроллера сводится в данном случае к проверке состояния выхода DA и, при обнаружении уровня логической единицы на нем, считыванию кода данных с выходов D, C, B и A.

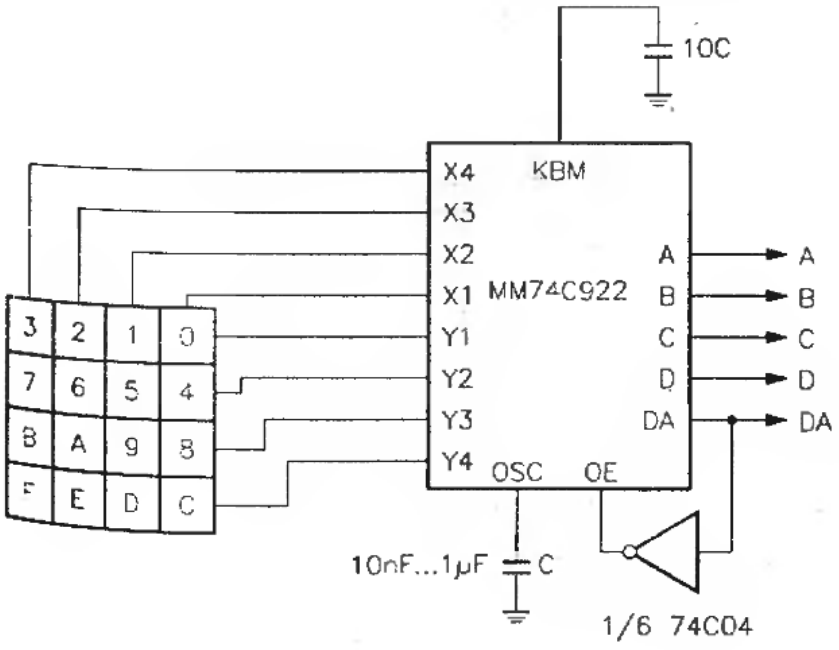


Рисунок 5.3 – Схема включения микросхемы MM74C922

Алгоритм опроса линии DA может быть произвольным, в том числе может быть увязан с алгоритмом выполнения микроконтроллером системных функций. В этом случае управляющая программа, проходя большой цикл, периодически проводит тестирование линии DA.

Второй вариант - работа в режиме прерывания. В этом случае выход DA соединяется с линией порта B микроконтроллера PIC. При нажатии на клавишу состояние линии DA изменяется, что вызывает прерывание текущей программы и переход к соответствующей подпрограмме обработки клавиатурного прерывания. Данное решение гарантирует, что любое нажатие на клавишу не будет пропущено (естественно при условии достаточной скорости обработки нажатия клавиш), как это иногда происходит при программном опросе линии DA.

Микроконтроллер может с успехом выполнять и функции клавиатурного кодера. Использование микроконтроллеров в матричных клавиатурах - самое универсальное решение. На рисунке 5.4 представлен вариант подключения к микроконтроллеру матричной клавиатуры с шестнадцатью клавишами, причем их число может быть без труда увеличено.

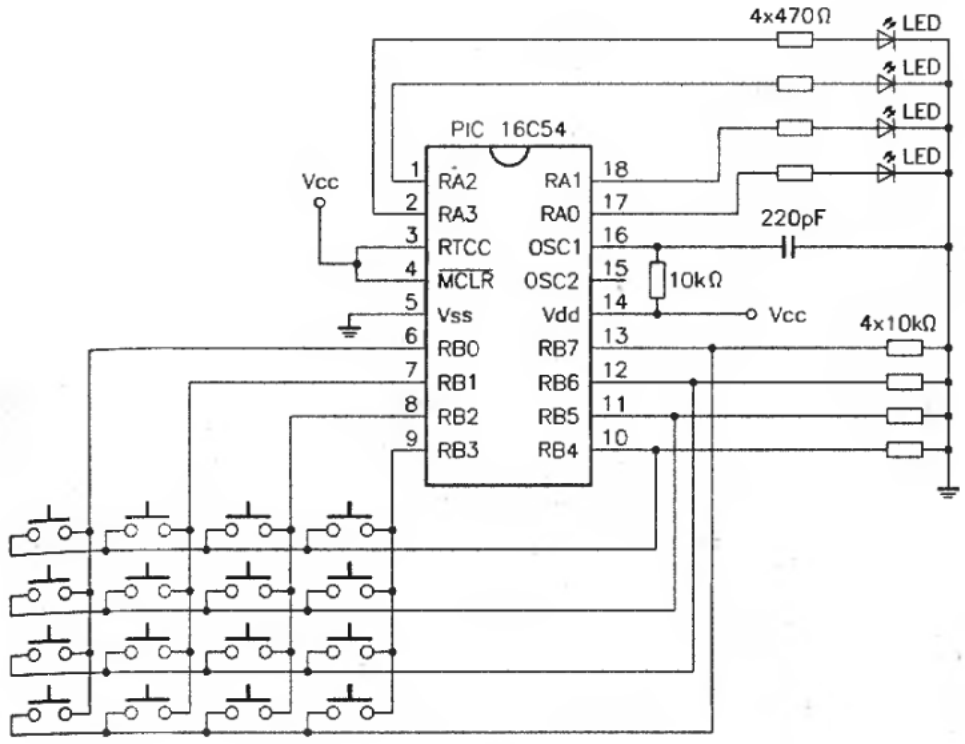


Рисунок 5.4 – Использование микроконтроллера PIC16C54 в качестве контроллера клавиатуры

Клавиши находятся на пересечении строк и столбцов матрицы. При нажатии на клавишу происходит замыкание соответствующей строки и столбца. Программа по номерам строки и столбца может определить, какая клавиша была нажата. Программа работает следующим образом. Линии столбцов соединены с портами RB0 - RB3 микроконтроллера, являющимися выходами, линии строк, напротив, подключены к входным портам RB4 - RB7. Программа осуществляет сканирование клавиатуры по строкам и столбцам, определяя момент появления логической единицы на одном из входных портов. В процессе сканирования непрерывно инкрементируется переменная key. При обнаружении логической единицы значение переменной key определяет номер нажатой клавиши. Сканирование столбцов осуществляется путем изменения позиции единицы в четырехразрядном позиционном коде, то есть путем последовательной выдачи на линии столбцов кодов 0001,0010,0100 и 1000. При каждом коде производится сканирование строк. Если какая-либо клавиша была нажата, то в одной из строк будет обнаружена единица. Сканирование в этот момент завершается, а значение переменной key идентифицирует нажатую клавишу.

До начала сканирования значение переменной key устанавливается равным нулю. Если ни одна клавиша не была нажата, программа возвращает в качестве значения переменной key число 16(10h). Программа осу-

ществляет индикацию номера клавиши в двоичном коде с помощью четырех светодиодов, подсоединенных к порту A.

В некоторых устройствах микроконтроллер активизируется только в ответ на воздействие пользователя на клавиатуру, а все остальное время остается в «спящем» режиме (sleep) для экономии энергии. Такой возможностью обладают практически все PIC16-микроконтроллеры. Выход из sleep-режима может осуществляться по любому внешнему прерыванию, в том числе по прерыванию, вызванному изменением состояния линий порта B.

2 Цель работы: уметь подключать клавиатуру к микроконтроллеру и реагировать на события нажатия клавиш.

3. Порядок выполнения работы

3.1 Согласно своему варианту подключить к микроконтроллеру клавиатуру.

3.2 Разработать программу обработки событий клавиатуры.

3.3 Проверить работоспособность разработанной системы в 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 Зачем при подключении кнопок используются подтягивающие (pullup) резисторы?

5.2 На каких входах микроконтроллера PIC16F877 есть внутренние подтягивающие резисторы?

5.3 Опишите суть явления дребезга контактов.

5.4 Какие методы борьбы применяются для ликвидации явления дребезга контактов?

5.5 Как подключают сравнительно большую клавиатуру к микроконтроллеру?

5.6 Напишите алгоритм определения двойного нажатия клавиши.

5.7 Как определить момент нажатия и момент отпускания клавиши?

Лабораторная работа №6
Подключение микроконтроллера к COM порту ЭВМ

1 Краткие теоретические сведения

Последовательный интерфейс для передачи данных использует одну сигнальную линию, по которой информационные биты передаются друг за другом последовательно. Последовательная передача позволяет сократить количество сигнальных линий и увеличить дальность связи. Характерной особенностью является применение не-ТТЛ сигналов.

Последовательная передача данных может осуществляться в асинхронном или синхронном режимах. При асинхронной передаче каждому байту предшествует старт-бит, сигнализирующий приемнику о начале посылки, за которым следуют биты данных и, возможно, бит паритета (четности). Завершает посылку стоп-бит, гарантирующий паузу между посылками. Старт-бит следующего байта посылается в любой момент после стоп-бита, то есть между передачами возможны паузы произвольной длительности. Старт-бит, имеющий всегда строго определенное значение (логический 0), обеспечивает простой механизм синхронизации приемника по сигналу от передатчика. Подразумевается, что приемник и передатчик работают на одной скорости обмена. Внутренний генератор синхронизации приемника использует счетчик-делитель опорной частоты, обнуляемый в момент приема начала старт-бита. Этот счетчик генерирует внутренние стробы, по которым приемник фиксирует последующие принимаемые биты. В идеале стробы располагаются в середине битовых интервалов, что позволяет принимать данные и при незначительном рассогласовании скоростей приемника и передатчика. Очевидно, что при передаче 8 бит данных, одного контрольного и одного стоп-бита предельно допустимое рассогласование скоростей, при котором данные будут распознаны верно, не может превышать 5%. С учетом фазовых искажений и дискретности работы внутреннего счетчика синхронизации реально допустимо меньшее отклонение частот. Чем меньше коэффициент деления опорной частоты внутреннего генератора (чем выше частота передачи), тем больше погрешность привязки стробов к середине битового интервала, и требования к согласованности частот становятся более строгими. Чем выше частота передачи, тем больше влияние искажений фронтов на фазу принимаемого сигнала. Взаимодействие этих факторов приводит к повышению требований к согласованности частот приемника и передатчика с ростом частоты обмена.

1)

Рисунок 6.1 - Формат асинхронной передачи

Формат асинхронной посылки позволяет выявлять возможные ошибки передачи. Если принят перепад, сигнализирующий о начале посылки, а по стробу старт-бита зафиксирован уровень логической единицы, старт-бит считается ложным и приемник снова переходит в состояние ожидания. Об этой ошибке приемник может и не сообщать.

Если во время, отведенное под стоп-бит, обнаружен уровень логического нуля, фиксируется ошибка стоп-бита. Если применяется контроль четности, то после посылки бит данных передается контрольный бит. Этот бит дополняет количество единичных бит данных до четного или нечетного в зависимости от принятого соглашения. Прием байта с неверным значением контрольного бита приводит к фиксации ошибки. Контроль формата позволяет обнаруживать обрыв линии: при этом принимаются логический нуль, который сначала трактуется как старт-бит, и нулевые биты данных, потом срабатывает контроль стоп-бита.

Для асинхронного режима принят ряд стандартных скоростей обмена: 50, 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600,19 200, 38 400, 57 600 и 115 200 бит/с. Иногда вместо единицы измерения "бит/с" используют "бод" (baud), но при рассмотрении двоичных передаваемых сигналов это некорректно. В бодах принято измерять частоту изменения состояния линии, а при недвоичном способе кодирования (широко применяемом в современных модемах) в канале связи скорости передачи бит (бит/с) и изменения сигнала (бод) могут отличаться в несколько раз.

Количество бит данных может составлять 5, 6, 7 или 8 (5-ти битные форматы распространены незначительно). Количество стоп-бит может быть 1, 1,5 или 2 ("полтора бита" означает только длительность стопового интервала).

Асинхронный обмен в PC реализуется с помощью COM-порта с использованием протокола RS232C. Интерфейс RS232C предназначен для подключения аппаратуры, передающей или принимающей данные. Стандарт описывает управляющие сигналы интерфейса, пересылку данных, электрический интерфейс и типы разъемов. В стандарте предусмотрены

асинхронный и синхронный режимы обмена, но COM-порты поддерживают только асинхронный режим.

Электрический интерфейс стандарта RS232C использует несимметричные передатчики и приемники - сигнал передается относительно общего провода схемной земли. Интерфейс НЕ ОБЕСПЕЧИВАЕТ ГАЛЬВАНИЧЕСКОЙ РАЗВЯЗКИ устройств. Логической единице соответствует напряжение на входе приемника в диапазоне от минус 12 до минус 3 В. Логическому нулю соответствует диапазон от +3 до +12 В. Диапазон от минус 3 до +3 В - зона нечувствительности, обусловливающая гистерезис приемника: состояние линии будет считаться измененным только после пересечения порога. Уровни сигналов на выходах передатчиков должны быть в диапазонах от минус 12 до минус 5 В и от +5 до +12 В для представления единицы и нуля соответственно.

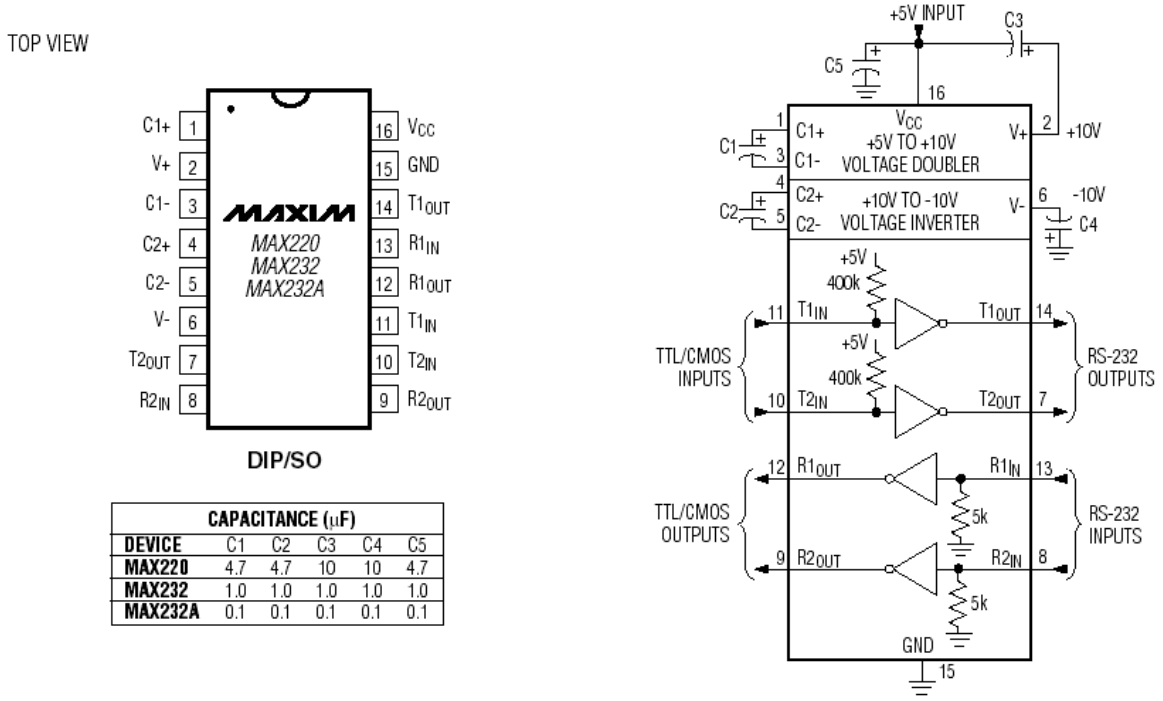
Интерфейс предполагает наличие ЗАЩИТНОГО ЗАЗЕМЛЕНИЯ для соединяемых устройств, если они оба питаются от сети переменного тока и имеют сетевые фильтры.

Подключение и отключение интерфейсных кабелей устройств с автономным питанием должно производиться при отключенном питании. Иначе разность невыровненных потенциалов устройств в момент коммутации может оказаться приложенной к выходным или входным (что опаснее) цепям интерфейса и вывести из строя микросхемы.

Для интерфейса RS232C специально выпускаются буферные микросхемы приемников (с гистерезисом и передатчиком двуполярного сигнала). Очень удобны в использовании микросхемы фирмы Maxim типа MAX232, MAX202 и т.п. Цоколевка и схема включения этих микросхем приведена на рисунке 6.2. На рисунке 6.3 приведена практическая схема преобразователя сигналов RS232C – ТТЛ.

Стандарт RS232C также регламентирует типы применяемых разъемов. На COM-портах принято устанавливать вилки (male "папа") DB25P или более компактный вариант DB9P. Девятиштырьковые разъемы не имеют контактов для дополнительных сигналов, необходимых для синхронного режима (в большинстве 25-ти штырьковых разъемов эти контакты не используются). На аппаратуре (модемах) устанавливают розетки (female "мама") DB25S или DB9S.

Если аппаратура соединяется друг с другом без модемов, то разъемы устройств (вилки) соединяются между собой нуль-модемным кабелем (Zeromodem или Zmodem), имеющим на обоих концах розетки, контакты которых соединяются перекрестно по одной из схем, приведенных на рисунке 6.4.



CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Рисунок 6.2 - Цоколевка и схема включения буферных микросхем фирмы Maxim

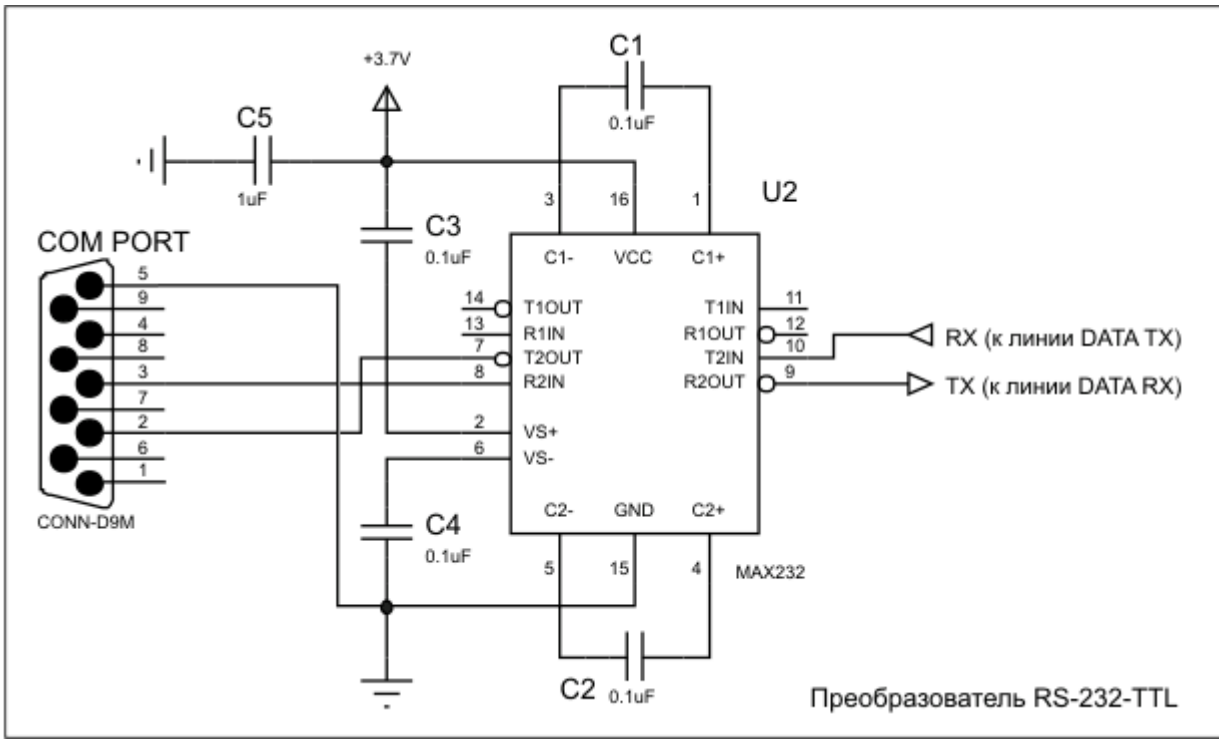


Рисунок 6.3 - Практическая схема преобразователя сигналов RS232C – ТТЛ

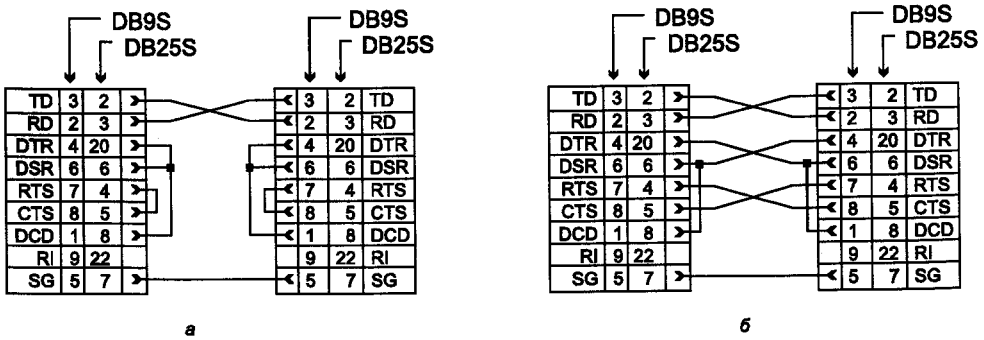


Рисунок 6.4 - Нуль-модемный кабель: а минимальный, б полный

Таблица 6.1 – Разъемы и сигналы интерфейса RS-232C

Обозначение цепи	Контакт разъема		Направление	Название цепи
RS232	DB25S	DB9S	I/O	
PG	1	—	—	Protect Ground — Защитная земля
TD	2	3	O	Transmit Data — Передаваемые данные
RD	3	2	I	Receive Data -Принимаемые данные
RTS	4	7	O	Request To Send — Запрос на передачу
CTS	5	8	I	Clear To Send — Готовность модема к приему данных для передачи
DSR	6	6	I	Data Set Ready — Готовность модема к работе
SG	7	5	-	Signal Ground — Схемная земля
DCD	8	1	I	Data Carrier Detected — Несущая обнаружена
DTR	20	4	O	Data Terminal Ready — Готовность терминала (PC) к работе
RI	22	9	I	Ring Indicator — Индикатор вызова

Таблица 6.2 – Наименование и назначение сигналов интерфейса RS-232C

Сигнал	Назначение
PG	Защитная земля, соединяется с корпусом устройства и экраном кабеля
SG	Сигнальная (схемная) земля, относительно которой действуют уровни сигналов
TD	Последовательные данные — выход передатчика
RD	Последовательные данные — вход приемника
RTS	Выход запроса передачи данных: состояние - «включено» уведомляет модем о наличии у терминала данных для передачи. В полудуплексном режиме используется для управления направлением — состояние «включено» является сигналом модему на переключение в режим передачи
CTS	Вход разрешения терминалу передавать данные. Состояние «выключено» аппаратно запрещает передачу данных. Сигнал используется для аппаратного управления потоками данных
DTR	Выход сигнала готовности терминала к обмену данными. Состояние «включено» поддерживает коммутируемый канал в состоянии соединения
OSR	Вход сигнала готовности от аппаратуры передачи данных (модем в рабочем режиме подключен к каналу и закончил действия по согласованию с аппаратурой на противоположном конце канала)
DCD	Вход сигнала обнаружения несущей удаленного модема
RI	Вход индикатора вызова (звонка). В коммутируемом канале этим сигналом модем сигнализирует о принятии вызова

Для связи по последовательному интерфейсу в микроконтроллерах Microchip используется модуль USART (УАПП) – универсального последовательного асинхронного приемопередатчика.

USART – это один из модулей последовательного порта ввода/вывода (имеет существенные отличия от модуля SSP), который может работать в полнодуплексном асинхронном режиме для связи с терминалами, персональными компьютерами или синхронном полудуплексном

режиме для связи с микросхемами ЦАП, АЦП, последовательными EEPROM и т.д.

USART может работать в одном из трех режимов:

• Асинхронный, полный дуплекс (для связи, например, с COM-портом ЭВМ);

• Ведущий синхронный, полудуплекс;

• Ведомый синхронный, полудуплекс.

Далее в основном мы будем рассматривать асинхронный режим работы USART.

Биты SPEN (RCSTA<7>) и TRIS должны быть установлены в ‘1’ для использования выводов TX/CK и RX/DT в качестве портов универсального синхронно-асинхронного приемопередатчика.

На рисунке 6.5 приведена структура управляющего регистра передатчика TXSTA.

На рисунке 6.6 приведена структура управляющего регистра приемника RXSTA.

Генератор частоты обмена USART BRG используется для работы USART в синхронном ведущем и асинхронном режимах. BRG представляет собой отдельный 8-разрядный генератор скорости обмена в бодах, период которого определяется значением в регистре SPBRG. В асинхронном режиме бит BRGH (TXSTA<2>) тоже влияет на скорость обмена (в синхронном режиме бит BRGH игнорируется). Приведем формулы для вычисления скорости обмена в бодах при асинхронном режиме работы модуля USART (SYNC = 0).

Учитывая требуемую скорость и FOSC, выбирается самое близкое целое значение для записи в регистр SPBRG (от 0 до 255), рассчитанное по формулам приведенным ниже. Затем рассчитывается ошибка скорости обмена.

Формулы расчета скорости обмена данными:

при значении бита BRGH = 0

Скорость обмена = FOSC / (64 (X + 1)) (1)

при значении бита BRGH = 1

Скорость обмена = FOSC / (16 (X + 1)) (2)

где X = значение регистра SPBRG (от 0 до 255).

В следующем примере показан расчет значения для регистра SPBRG и погрешность скорости обмена для условий:

FOSC = 16 МГц;

Скорость приема/передачи данных = 9600 бит/с;

BRGH = 0;

Желаемое значение скорости = FOSC / (64 (X + 1))

9600 = 16 000 000 / (64 (X + 1))

X = [25.042] = 25

Вычисленное значение скорости = 16 000 000 / (64 (25 + 1)) = 9615

Ошибка = 100 x (Вычисленное – Желаемое) / Желаемое значение скорости

Ошибка = 100 x (9615 – 9600) / 9600 = 0.16%

В некоторых случаях может быть выгодно использовать высокоскоростной режим работы USART (BRGH=1), поскольку уравнение FOSC / (16 (X + 1)) позволяет уменьшить погрешность скорости.

Запись нового значения в регистр SPBRG сбрасывает таймер BRG, гарантируя немедленный переход на новую скорость.

В асинхронном режиме USART использует стандартный формат NRZ (так же как и COM-порт ЭВМ): один стартовый бит, восемь или девять битов данных и один стоповый бит. Наиболее часто встречается 8-разрядный формат передачи данных. Интегрированный 8-разрядный генератор BRG позволяет получить стандартные скорости передачи данных. Генератор скорости обмена, как уже отмечалось, может работать в одном из двух режимов: высокоскоростной (x16 BRGH=1 TXSTA<2>), низкоскоростной (x64 BRGH=0 TXSTA<2>). Приемник и передатчик последовательного порта работают независимо друг от друга, но используют один и тот же формат данных и одинаковую скорость обмена. Бит четности аппаратно не поддерживается, но может быть реализован программно, применяя 9-разрядный формат данных. Все данные передаются младшим битом вперед.

TXSTA: Регистр управления и статуса передатчика

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	**TX9**	**TXEN**	**SYNC**	**-**	**BRGH**	**TRMT**	**TX9D**
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: **CSRC:** Выбор источника тактового сигнала
Синхронный режим
1 = ведущий, внутренний тактовый сигнал от BRG
0 = ведомый, внешний тактовый сигнал с входа CK

Асинхронный режим
Не имеет значения

бит 6: **TX9:** Разрешение 9-разрядной передачи
1 = 9-разрядная передача
0 = 8-разрядная передача

бит 5: **TXEN:** Разрешение передачи
1 = разрешена
0 = запрещена
Примечание. В синхронном режиме биты SREN/CREN отменяют действие бита TXEN.

бит 4: **SYNC:** Режим работы USART
1 = синхронный
0 = асинхронный

бит 3: **Не используется:** читается как '0'

бит 2: **BRGH:** Выбор высокоскоростного режима
Синхронный режим
Не имеет значения

Асинхронный режим
1 = высокоскоростной режим
0 = низкоскоростной режим

бит 1: **TRMT:** Флаг очистки сдвигового регистра передатчика TSR
1 = TSR пуст
0 = TSR полон

бит 0: **TX9D:** 9-й бит передаваемых данных (может использоваться для программной проверки четности)

Рисунок 6.5 - Структура управляющего регистра передатчика TXSTA.

RCSTA: Регистр управления и статуса приемника

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	**RX9**	**SREN**	**CREN**	**ADDEN**	**FERR**	**OERR**	**RX9D**
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: **SPEN:** Разрешение работы последовательного порта
1 = модуль USART включен (выводы RX/DT, TX/CK подключены к USART)
0 = модуль USART выключен

18....1.1

бит 6: **RX9:** Разрешение 9-разрядного приема
1 = 9-разрядный прием
0 = 8-разрядный прием

бит 5: **SREN:** Разрешение одиночного приема
Синхронный режим
1 = разрешен одиночный прием
0 = запрещен одиночный прием
Сбрасывается в ‘0’ по завершению приема.
Примечание. В режиме ведомого не имеет значения

Асинхронный режим
Не имеет значения

бит 4: **CREN:** Разрешение приема
Синхронный режим
1 = прием разрешен (при установке бита CREN автоматически сбрасывается бит SREN)
0 = прием запрещен

Асинхронный режим
1 = прием разрешен
0 = прием запрещен

бит 3: **ADDEN:** Разрешение детектирования адреса(1)
Асинхронный 9-разрядный прием (RX9=1)
1 = детектирование адреса разрешено. Если бит RSR<8>=1, то генерируется прерывание и загружается приемный буфер.
0 = детектирование адреса запрещено. Принимаются все байты, девятый бит может использоваться для проверки четности.

Асинхронный 8-разрядный прием (RX9=0)
Не имеет значения

Синхронный режим
Не имеет значения

бит 2: **FERR:** Ошибка кадра, сбрасывается при чтении регистра RCREG
1 = произошла ошибка кадра
0 = ошибки кадра не было

бит 1: **OERR:** Ошибка переполнения внутреннего буфера, устанавливается в ‘0’ при сбросе бита CREN
1 = произошла ошибка переполнения
0 = ошибки переполнения не было

бит 0: **RX9D:** 9-й бит принятых данных (может использоваться для программной проверки четности)

Рисунок 6.6 - Структура управляющего регистра приемника RXSTA.

В SLEEP режиме микроконтроллера модуль USART(асинхронный режим) выключен.

Выбор асинхронного режима USART выполняется сбросом бита SYNC в ‘0’ (TXSTA<4>).

Модуль USART в асинхронном режиме состоит из следующих элементов:

- Генератор скорости обмена;
- Цепь опроса;
- Асинхронный передатчик;
- Асинхронный приемник.

Асинхронный передатчик USART показан на рисунке 6.7.

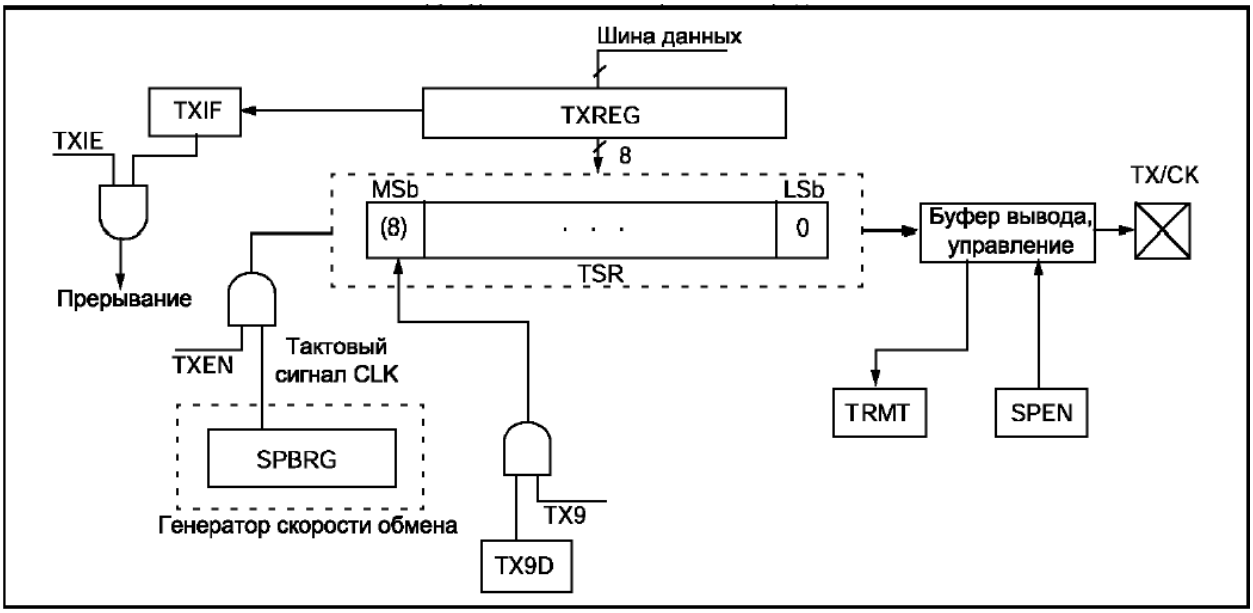


Рисунок 6.7 – Структура асинхронного передатчика USART

Главным в передатчике является сдвиговый регистр TSR, который получает данные из буфера передатчика TXREG. Данные в регистр TXREG загружаются программно. После передачи стопового бита предыдущего байта, в последнем машинном такте цикла BRG, TSR загружается новым значением из TXREG (если оно присутствует), после чего устанавливается флаг прерывания TXIF. Прерывание может быть разрешено или запрещено битом TXIE. Флаг TXIF устанавливается независимо от состояния бита TXIE и не может быть сброшен в ‘0’ программно. Очистка флага TXIF происходит только после загрузки новых данных в регистр TXREG. Аналогичным образом бит TRMT (TXSTA<1>) отображает состояние регистра TSR. Бит TRMT доступен только на чтение и не может вызвать генерацию прерывания. Регистр TSR не отображается на память и не доступен для чтения. Флаг TXIF устанавливается в ‘1’ только, когда бит TXEN=1 и сбрасывается автоматически в '0' после загрузки новых данных в регистр TXREG.

Для разрешения передачи необходимо установить бит TXEN (TXSTA<5>) в ‘1’. Передача данных не начнется до тех пор, пока в

TXREG не будут загружены новые данные и не придет очередной тактовый импульс от генератора BRG.

Можно сначала загрузить данные в TXREG, а затем установить бит TXEN. Как правило, после разрешения передачи регистр TSR пуст, таким образом, данные записываемые в TXREG сразу передаются в TSR, а TXREG остается пустым. Это позволяет реализовать слитную передачу данных. Сброс бита TXEN в ‘0‘ вызовет немедленное прекращение передачи, сброс передатчика и перевод вывода TX/CK в третье состояние.

Для разрешения 9-разрядной передачи необходимо установить бит TX9 (TXSTA<6>) в ‘1’. Девятый бит данных записывается в бит TX9D (TXSTA<0>). Девятый бит данных должен быть указан до записи в регистр TXREG, потому что данные, записанные в регистр TXREG, могут быть сразу загружены в сдвиговый регистр TSR (если он пуст).

Рекомендованная последовательность действий для передачи данных в асинхронном режиме:

1. Установить требуемую скорость передачи с помощью регистра SPBRG и бита BRGH.

2. Выбрать асинхронный режим сбросом бита SYNC в ‘0’ и установкой бита SPEN в ‘1’.

3. Если необходимо, разрешить прерывания установкой битов TXIE, PEIE, GIE в ‘1’.

4. Если передача 9-разрядная, установить бит TX9 в ‘1’.

5. Разрешить передачу установкой бита TXEN в ‘1’, автоматически устанавливается флаг TXIF.

6. Если передача 9-разрядная, записать 9-й бит данных в TX9D.

7. Записать данные в регистр TXREG (начало передачи данных).

Структурная схема асинхронного приемника USART показана на рисунке 6.8.

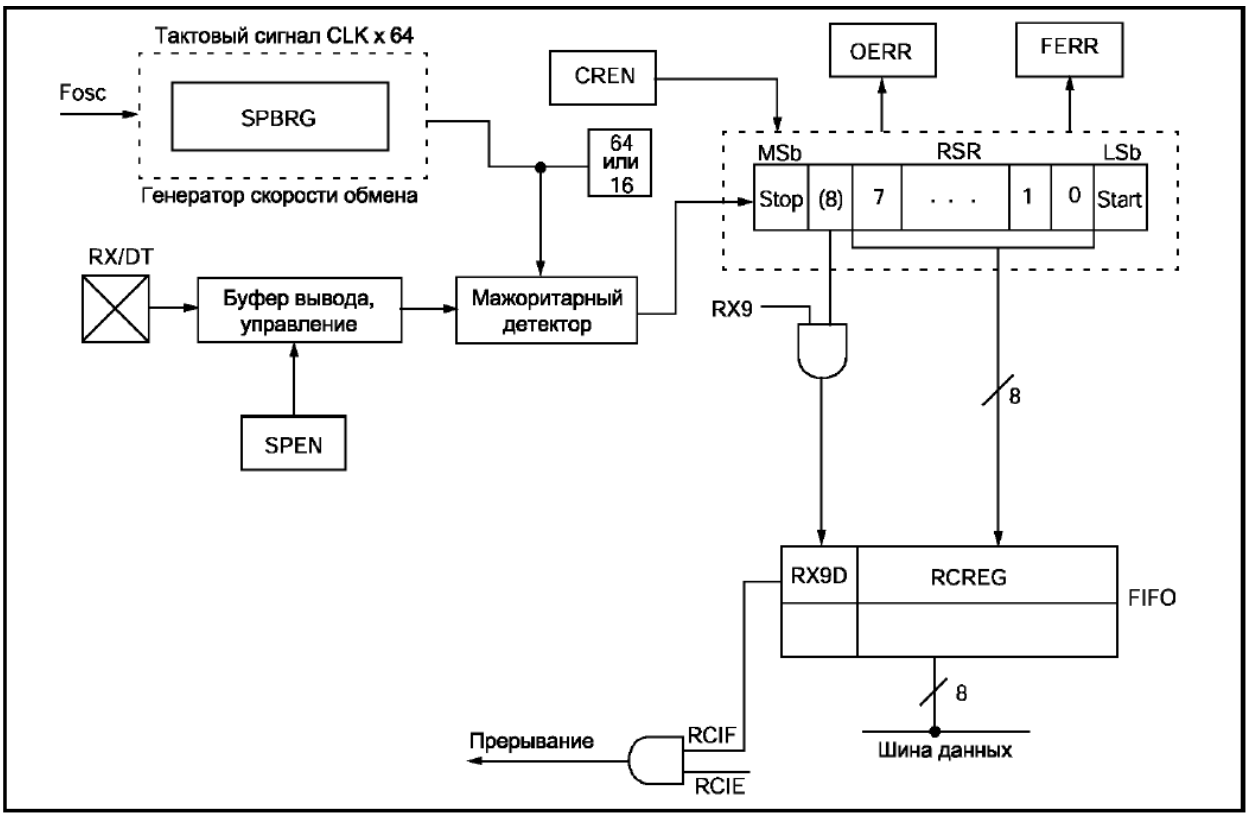


Рисунок 6.8 – Структура асинхронного приемника USART

Данные подаются на вход RX/DT в блок восстановления данных, представляющий собой скоростной сдвиговый регистр, работающий на частоте в 16 раз превышающей скорость передачи или FOSC.

Включение приемника производится установкой бита CREN (RCSTA<4>) в ‘1’.

Главным в приемнике является сдвиговый регистр RSR. После получения стопового бита данные переписываются в регистр RCREG, если он пуст. После записи в регистр RCREG выставляется флаг прерывания RCIF.

Прерывание можно разрешить/запретить установкой/сбросом бита RCIE. Флаг RCIF доступен только на чтение, сбрасывается аппаратно при чтении из регистра RCREG. Регистр RCREG имеет двойную буферизацию, т.е. представляет собой двухуровневый буфер FIFO. Поэтому можно принять 2 байта данных в FIFO RCREG и третий в регистр RSR. Если FIFO заполнен и обнаружен стоповый бит третьего байта, устанавливается бит переполнения приемника OERR (RCSTA<1>). Байт, принятый в RSR, будет потерян. Для извлечения двух байт из FIFO необходимо дважды прочитать регистр RCREG. Бит OERR нужно программно очистить сбросом бита CREN, т.е. запрещением приема. В любом случае, если бит OERR установлен, логика приемника выключена.

Бит ошибки кадра FERR (RCSTA<2>) устанавливается в ‘1’, если не обнаружен стоповый бит. FERR и девятый бит принятых данных буферизируются так же, как принятые данные. Рекомендуется сначала прочитать регистр RCSTA, затем RCREG, чтобы не потерять информацию RX9D и FERR.

Рекомендованные действия при приеме данных в асинхронном режиме:

1. Установить требуемую скорость передачи с помощью регистра SPBRG и бита BRGH.

2. Выбрать асинхронный режим сбросом бита SYNC в ‘0’ и установкой бита SPEN в ‘1’.

3. Если необходимо, разрешить прерывания установкой битов RCIE, PEIE и GIE в ‘1’.

4. Если прием 9-разрядный, установить бит RX9 в ‘1’.

5. Разрешить прием установкой бита CREN в ‘1’.

6. Ожидать установку бита RCIF, или прерывание, если оно разрешено битом RCIE.

7. Считать 9-й бит данных (если разрешен 9-разрядный прием) из регистра RCSTA и проверить возникновение ошибки.

8. Считать 8 бит данных из регистра RCREG.

9. При возникновении ошибки переполнения сбросить бит CREN в ‘0’.

Приведем пример инициализации приемника/передатчика USART в асинхронном режиме с 8-ми разрядными данными с разрешением прерываний:

```
BSF STATUS,RP0      ; Банк 1
BSF TRISC,7         ;rx/rc7 - как вход!
MOVLW <baudrate>    ; Установить скорость
                    ;обмена данными
MOVWF SPBRG
MOVLW 0x20      ;Передача 8-разрядных данных,
                ; вкл передатчик,
MOVWF TXSTA         ; низкоскоростной асинхронный
                    ; режим
BSF PIE1,TXIE       ; Разрешить прерывания
                    ;от передатчика USART
BSF PIE1,RCIE       ; Разрешить прерывания
                    ;от приемника USART
BCF STATUS,RP0      ; Банк 0
```

```
MOVLW 0x90        ; Прием 8 - разрядных
                  ;данных, включить приемник,
MOVWF RCSTA           ; включить модуль USART
```

Для моделирования работы USART микроконтроллера с COM-портом ЭВМ в системе Proteus будем использовать уникальный компонент COMPIM. Этот компонент является связующим звеном между эмулируемой схемой и реальным COM-портом ЭВМ, т.е. все сигналы, которые подаются на этот компонент транслируются в реальные сигналы на заданном COM порту. Входы и выходы компонента COMPIM используют логику ТТЛ, что позволяет подключать эмулируемый микроконтроллер напрямую к COM-порту без использования преобразователей уровня.

Схема лабораторной установки показана на рисунке 6.9.

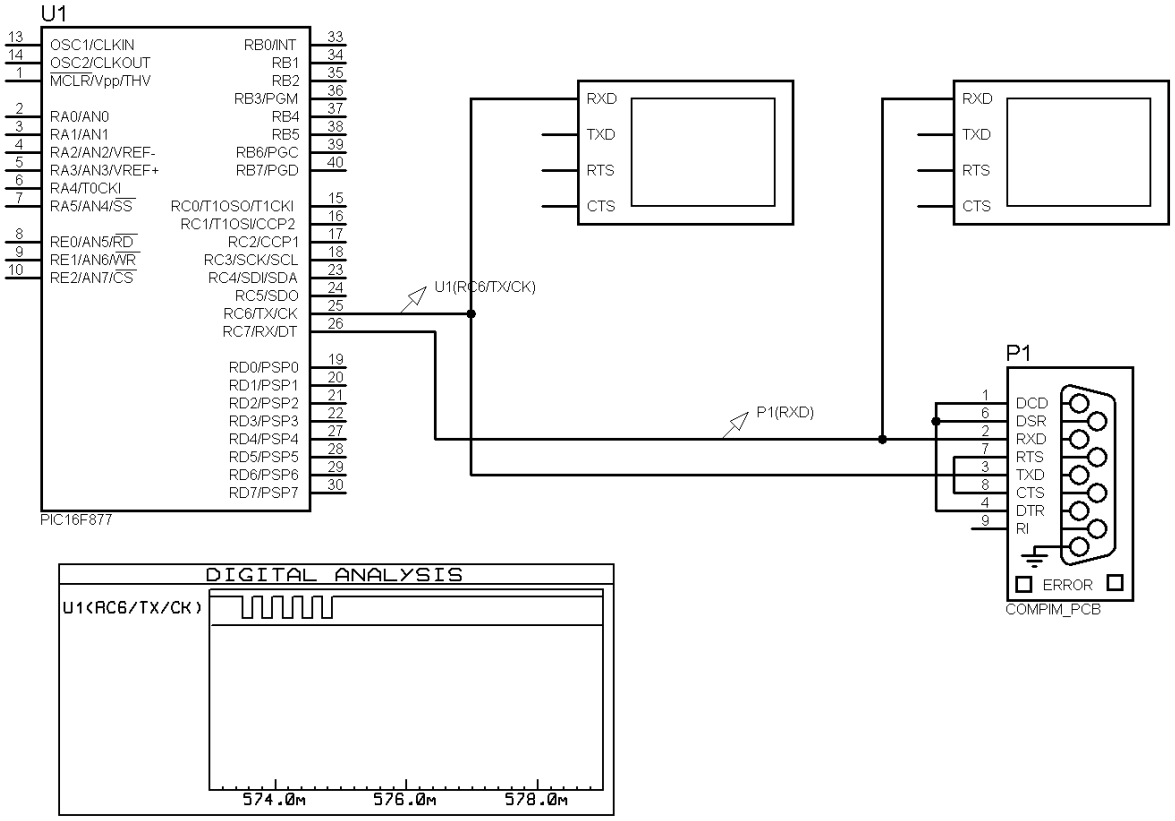


Рисунок 6.9 - Схема лабораторной установки для исследования работы микроконтроллера с COM-портом ЭВМ

Кроме компонента COMPIM и микроконтроллера на схеме представлены еще два виртуальных терминала для исследования средствами Proteus принимаемых и передаваемых данных, а также график изменения цифровой линии TX для возможности измерения реальной скорости передачи информации.

2 Цель работы: уметь подключать микроконтроллер к COM-порту ЭВМ.

3. Порядок выполнения работы

3.1 Согласно своему варианту подключить микроконтроллер к COM-порту другой ЭВМ или к USART другого микроконтроллера.

3.2 Разработать программу ввода-вывода информации.

3.3 Проверить работоспособность разработанной системы в 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 происходит синхронизация источника асинхронного последовательного сигнала и приемника?

5.2 В каких случаях предпочтительнее использовать асинхронную, а в каких синхронную последовательную передачу данных?

5.3 Какое значение необходимо записать в регистр SPBRG, чтобы микроконтроллер PIC16F877 асинхронно передавал данные со скоростью 19200 бит/с при подключенном к микроконтроллеру кварце на 20 МГц?

5.4 Что может быть причиной несоответствия передаваемых и принимаемых данных при асинхронной последовательной передаче?

5.5 Какие флаги сигнализируют о детектировании ошибок асинхронного приемника USART в микроконтроллере PIC16F877?

5.6 Каким способом можно измерить реальную скорость передачи информации при асинхронной последовательной передаче?