Министерство образования Республики Беларусь
ГРОДНЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. ЯНКИ КУПАЛЫ

В.М. Рамазанов, Ю.Р. Бейтюк, Г.П. Себровская, Воробьев М.Г.

СХЕМОТЕХНИЧЕСКОЕ МОДЕЛИРОВАНИЕ ЦИФРОВЫХ УСТРОЙСТВ В СРЕДЕ PROTEUS (ISIS)

Методические указания к выполнению лабораторных работ по курсам «Схемотехника аналоговых и цифровых устройств», «Программируемые цифровые устройства» для студентов 3, 4 курсов физико-технического факультета, специальности 1 - 38.02.01 «Информационно-измерительная техника»

Гродно 2012

СОДЕРЖАНИЕ

1. Моделирование цифровых устройств в среде PROTEUS (ISIS)
1.1 Моделирование работы комбинационных и последовательностных цифровых автоматов

Моделирование работы цифровых автоматов является заключительной операцией их синтеза, позволяющей убедиться в правильности всех предшествующих операций по получению уравнений, минимизации и окончательному синтезу электрической принципиальной схемы. Ниже приводится **методика моделирования комбинационных и последовательностных цифровых автоматов**. Она включает следующие шаги:

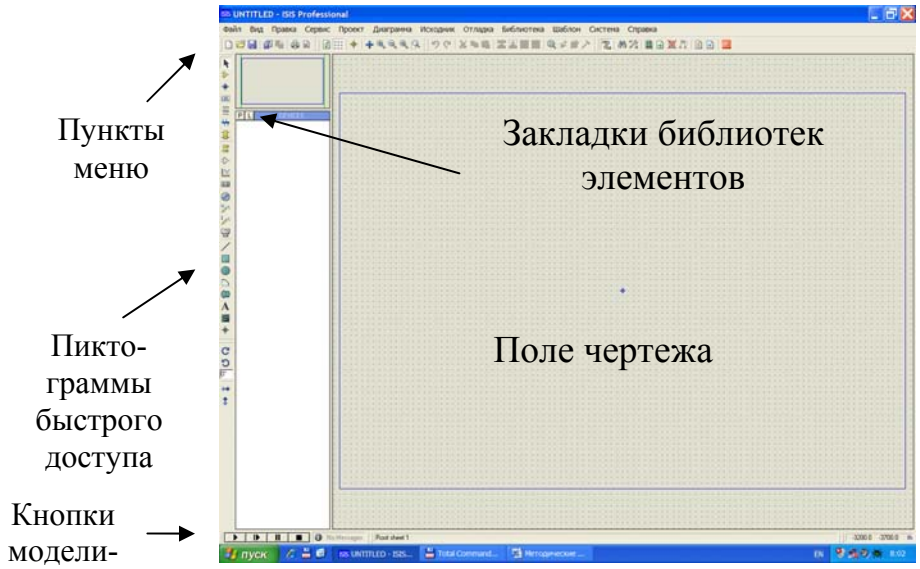


Рис. 1 Назначение элементов интерфейса в Proteus

1. **Создание проекта** в среде ISIS. Вначале **создайте на диске папку проекта** с именем из не более 8 латинских символов. Далее **откройте среду** ISIS Proteus (рис. 1), запустив пиктограмму ISIS с рабочего стола. Интерфейс среды содержит горизонтальное меню с вертикально расположенными слева пиктограммами быстрого доступа к основным инструментам моделирования, внизу слева расположены кнопки управления процессом моделирования. После появления основного окна среды с именем «UNTITLED-ISIS Profes-

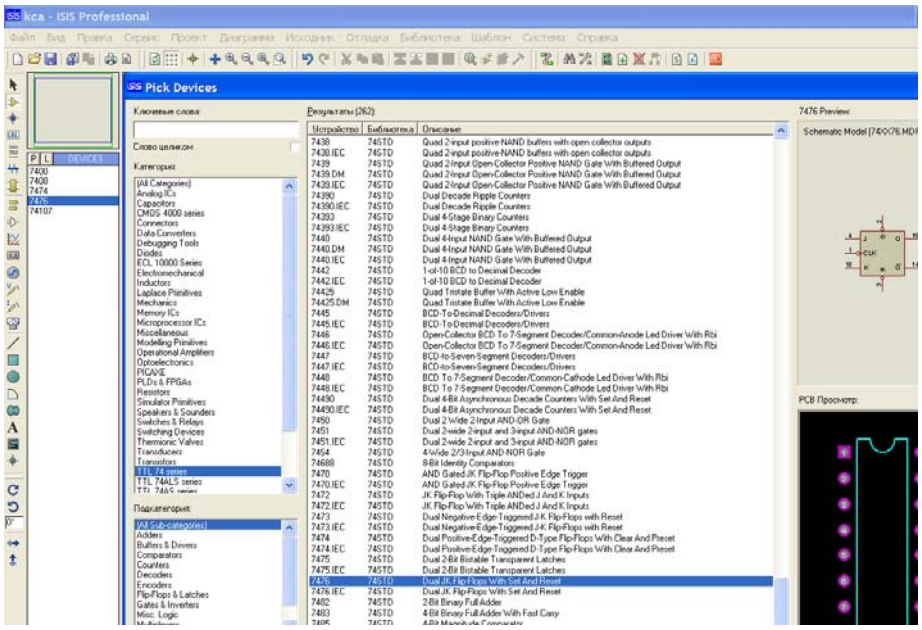
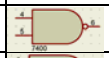


Рис. 2 Выбор требуемых элементов схемы

sional», через пункт основного меню: «**Файл - Сохранить проект**» в появившемся стандартном окне сохранения файла в созданной ранее папке **создайте файл проекта с именем совпадающем с именем папки**. Расширение dsn среда присвоит автоматически, а его **название появится в верхней горизонтальной строке** окна среды. В нашем примере имя файла – KCA.

2. **Выберите требующиеся** Вам для синтеза схемы логические **элементы** (микросхемы «2И», «2И-НЕ», триггера и т.д.) из библиотек. Для этого щелкните левой кнопкой мыши по закладке «P» и в появившемся окне «Pick Devices» (рис. 2) в окне «Категория» выберите строку «TTL 74 Series». После этого двойным щелчком левой кнопки мыши «наберите» требующиеся Вам микросхемы из списка, чтобы они появились в левой колонке окна проекта. Таблица соответствия микросхем 74-ой серии и логических элементов приведена ниже (Табл. №1):

Таблица №1

Логический элемент	**Микросхема 74-ой серии**	**Условное обозначение**
«2И-НЕ»	7400	
«2И»	7408	
«2ИЛИ-НЕ»	7402	
«2ИЛИ»	7432	
JK- триггер	7476	
D-триггер	7474	

При необходимости выберите иные требующиеся Вам элементы. Выбрав элементы, закройте окно «Pick Devices». Удаление ненужных элементов производится их выделение правой кнопкой мыши с последующим удалением через меню, либо клавишей «DEL».

3. **Создайте** электрическую принципиальную **схему** моделируемого устройства. Для этого, выбирая, щелчком левой кнопки мыши, **элементы из списка**, **расположите их на «поле чертежа»**, закрепляя их местопо-

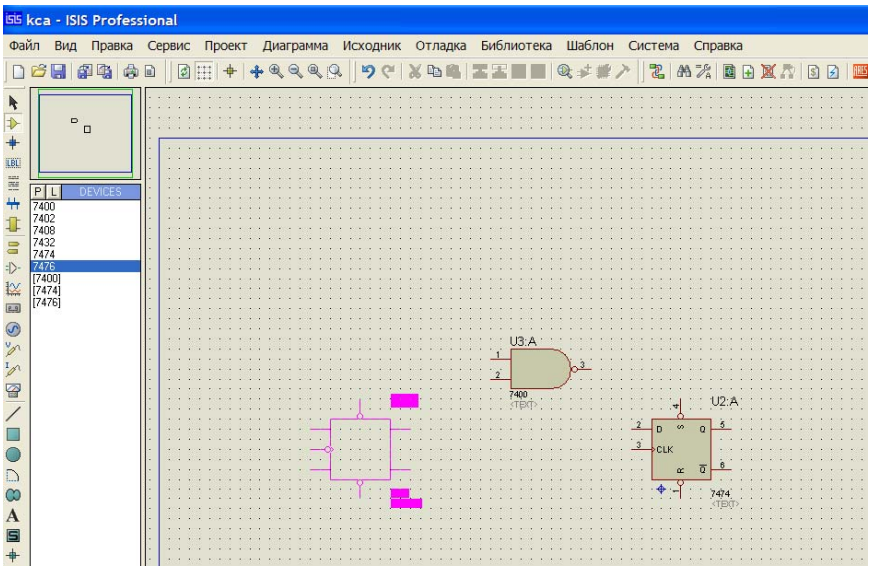


Рис. 3 Расположение элементов схемы на «поле чертежа»

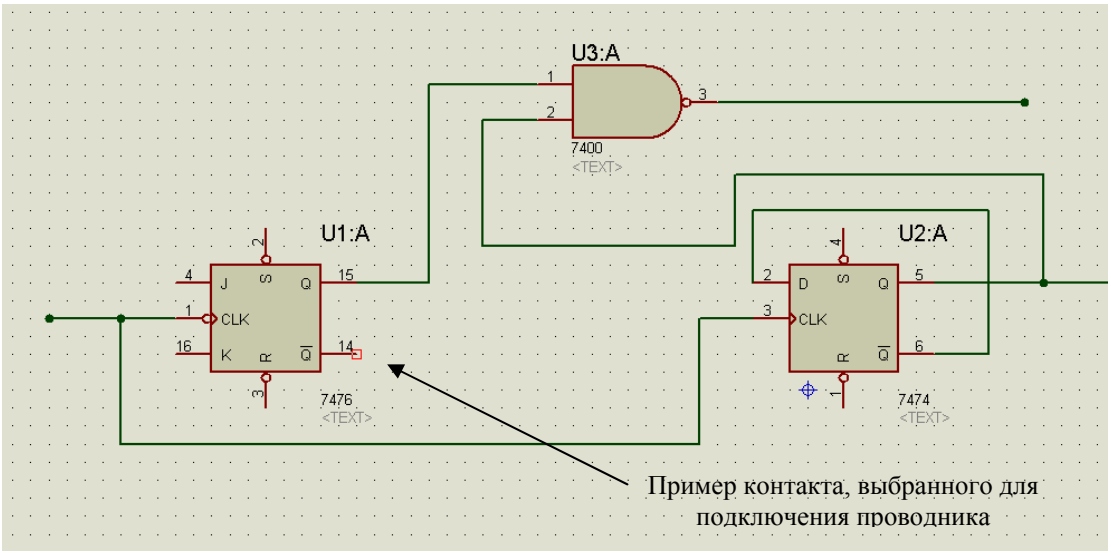


Рис. 4 Соединение элементов схемы на «поле чертежа»

ложение – двойным щелчком левой кнопки мыши (рис. 3). **Соедините контакты** элементов в соответствие с синтезированной Вами схемой. Для этого подведите курсор под требуемый контакт (при этом появится красный квадратик около контакта на рис. 4), щелкните левой кнопкой мыши для фиксации начала проводника, и ведите проводник в нужную точку схемы, завершив операцию двойным щелчком (при необходимости изменить направление проводника, щелкните в нужной точке схемы один раз левой кнопкой).

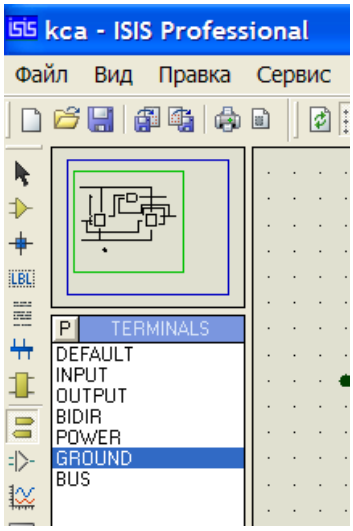


Рис. 5 Выбор терминалов «0» и «1»

4. **Подайте** требующиеся **статические** (постоянные) **сигналы** на требуемые контакты микросхем. На рис. 4 это контакты J K триггера U1:A, и входы S установки исходного состояния обоих триггеров. Это делается с помощью соединения этих контактов со специальным элементом, называемым терминалом (рис. 5). Элементы типа терминал находятся в библиотеке, которая выбирается пиктограммой «Терминалы», в левом вертикальном графическом меню (рис. 5). Уровень логической «1» обеспечивает терминал «POWER», а уровень «0» - терминал «GROUND». Расположение их на чертеже и соединение с проводниками и контактами – аналогично.

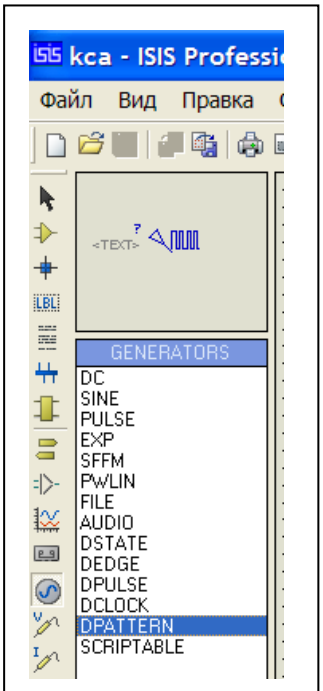
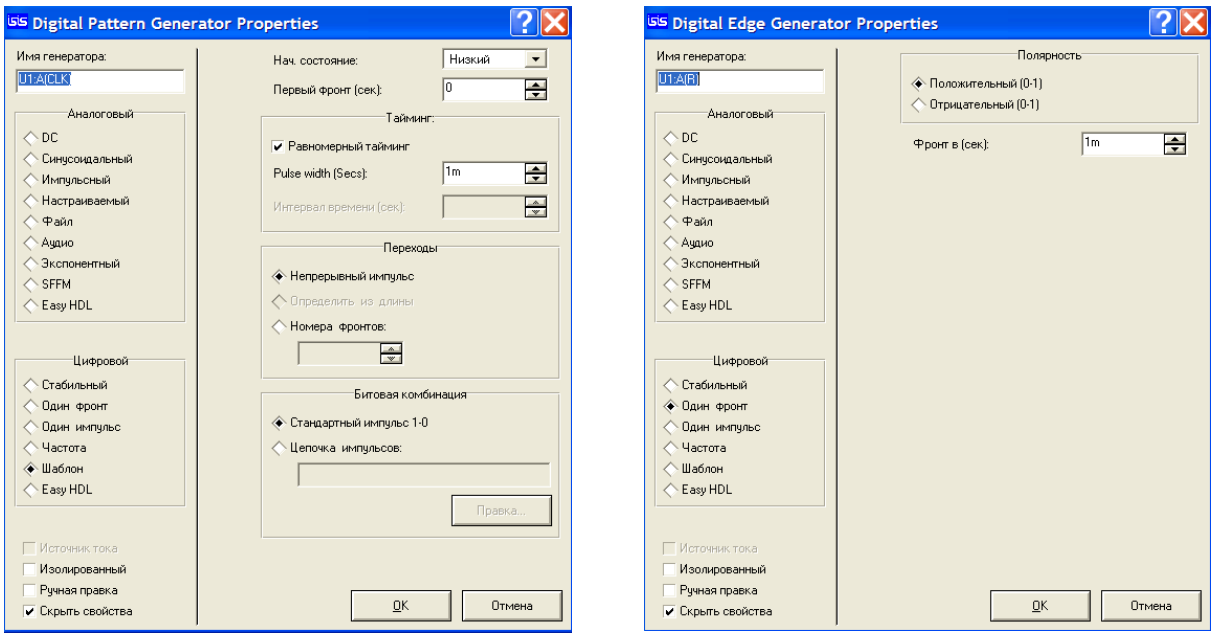


Рис. 6 Выбор генераторов

5. **Подайте** требующиеся **динамические** (переменные) **сигналы** на требуемые контакты микросхем, как пра-

вило, это входные сигналы Вашего устройства. На рис. 4 это сигналы на входах CLK триггеров, и сигналы установок на их входе R. Сигнал на входе CLK – это импульсный тактовый сигнал, сигнал на входах R – дискретный, меняющийся лишь несколько раз в процессе работы устройства. Это можно делать различным образом, но для унификации мы воспользуемся **одним генератором, с соответствующим изменением параметров его выходного сигнала**. Выбор генератора как элемента схемы производится из библиотеки генераторов в левой вертикальной части основного окна программы (рис. 6). **Тип генератора «DPATTERN»**. Расположение на чертеже и соединение с проводниками и контактами – по обычным правилам. Подключаем **два (или несколько) таких генераторов, но**



импульсов CLK перепада напряжения R

Рис. 7 Конфигурирование генераторов входных сигналов

свойства их изменим следующим образом. Щелкнув два раза по изображению генератора, получаем доступ к окну его свойств (рис. 7). **Для генератора CLK** выбираем вид сигнала «Цифровой - Шаблон» и устанавливаем галочку в окне «Равномерный тайминг» окна «Тайминг» и длительность импульса (Pulse width) 50m (50 миллисекунд). Для генератора R – параметры соответственно также указаны на рис. 7 (длительность низкого уровня установлена 1m). Остальные параметры импульсов приведены на рис. 7. Для удобства можно присвоить названия генераторов по смыслу сигналов в окне «Имя генератора». **Для генератора перепада** соответственно выбираем вид сигнала – Цифровой/Один фронт; Полярность – положительная с длительностью фронта 1 m (как пока-

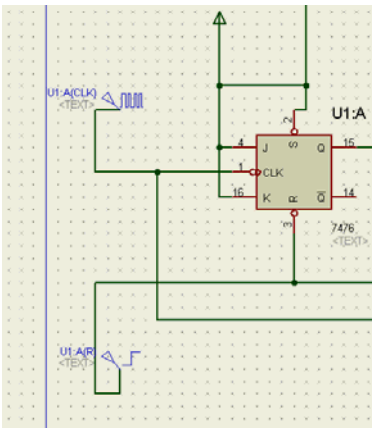


Рис. 8 Подключение генераторов

зано на рис. 7). Фрагмент схемы с подключенными генераторами представлен на рис. 8.

Для генераторов входных сигналов КЦА, имеющих 5 входов, частоты 5–ти генераторов отличаются каждый в 2 раза, в соответствие с их расположением в таблице функционирования (длительность импульса соответственно равны 50m – 100m – 200m – 400m – 800m и т.д.).

6. **Подключаем логические пробники** к интересующим цепям, как правило, это **входные и выходные цепи устройства**, хотя при поиске «неисправностей» может понадобиться их подключение и к «предполагаемо неисправным» участкам цепей схемы. **Для их выбора** из библиотек щелкните левой кнопкой мыши по закладке «P» и в появившемся окне «Pick Devices» (рис. 2) в окне «Ключевые слова» наберите **LOGICPROBE (BIG)**. После этого двойным щелчком левой кнопки мыши «выберите» указанный компонент из списка, чтобы он появился в левой колонке окна проекта. **Разместите их на поле чертежа и подключите к требуемым цепям** стандартным образом. Результирующая схема с подключенными приборами показана на рис. 9 (в ней добавлен ряд логических элементов для демонстрации подачи входных сигналов, периоды которых разли-

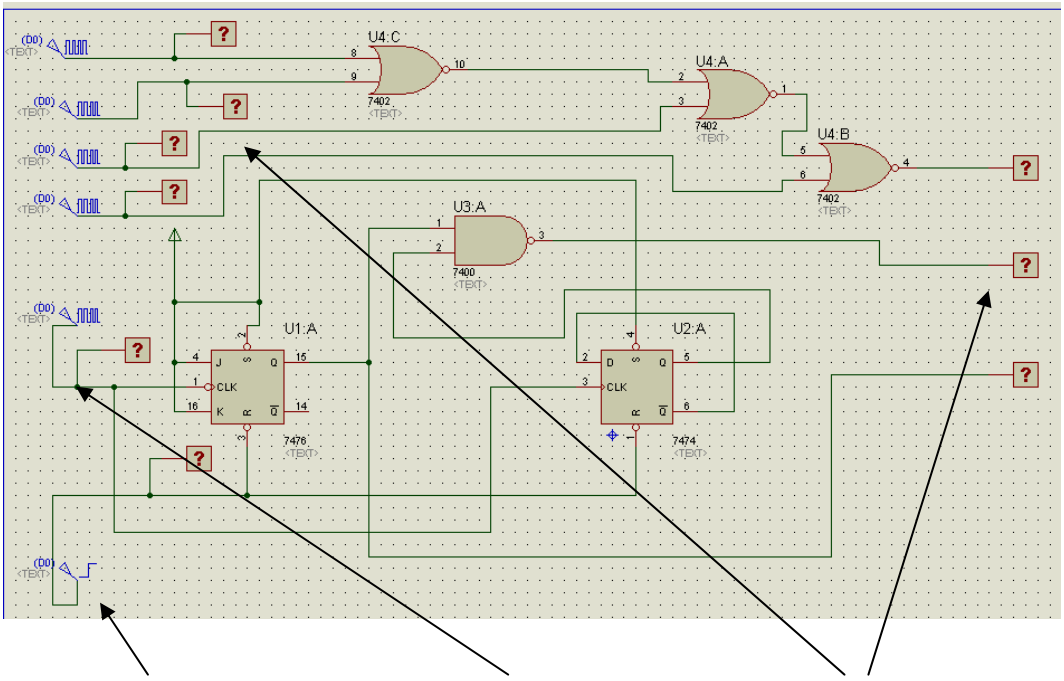


Рис. 9 Схема устройства с подключенными генераторами и пробниками, подготовленная к моделированию

чаются в соответствие с кодом «8-4-2-1»). При необходимости нажатием клавиши F8 расположите всю схему на экранею

7. **Сохраните проект** через основное меню программы.

8. **Перед началом моделирования** установите длительность шага моделирования: в главном меню System/Set Animation Options в окне параметров Single

Step Time = 51m. **Моделирование работы схемы** производится **в пошаговом режиме** путем однократного нажатия на пиктограммы в левом нижнем углу среды (рис. 10). При этом пробники будут отображать уровни логических сигналов на месте значков «?». При этом **одно нажатие кнопки «Шаг» соответствует 1 тактовому импульсу** CLK или сигналу с минимальным периодом. Следует отметить, что в процессе моделирования аналогичные «маленькие пробники» появятся около контактов микросхем автоматически, однако их размеры неудобны для анализа уровней сигналов.

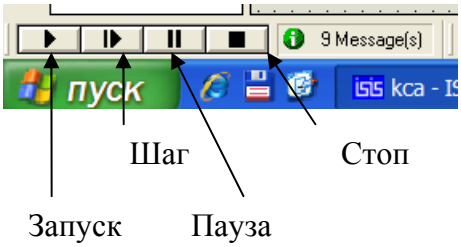


Рис. 10 Пиктограммы управления моделированием

При совпадении осциллограмм с законами изменения выходных сигналов из таблицы функционирования моделирование считается законченным. В противном случае, остановив моделирование, перейдите к редактированию схемы.

9. **Редактирование схемы** производится в том случае, если в результате моделирования были обнаружены ошибки функционирования, т.е. работа схемы отличалась от ее работы по таблице функционирования. После внесения изменений в схему, повторите моделирование.

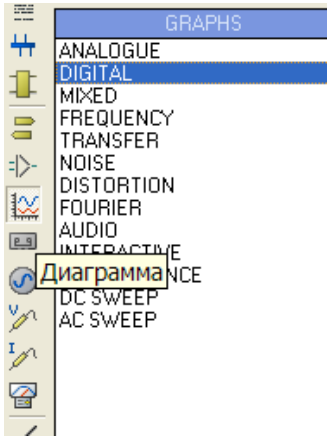


Рис. 11 Выбор графического анализатора

10. **ПРИ НЕОБХОДИМОСТИ НАБЛЮДЕНИЯ** за работой цифровой схемы **в непрерывном режиме** работы, можно воспользоваться **графическим анализатором**. Подключаем графический анализатор к интересующим нас точкам схемы. Как правило, это все входные и выходные сигналы схемы. Это делается в три этапа.

А. Выбор анализатора осуществляется из **боковых пиктограмм**: «Диаграмма-GRAPHS-DIGITAL» (рис. 11). После выбора анализатора, на поле чертежа мышью с нажатой левой кнопкой «рисуется» в подходящем месте «контур прибора», завершаясь повторным щелчком левой кнопки мыши.

В. Подключаем щупы напряжения к требуемым выходным (или иным) проводам схемы. Выбор щупов осуществляется из **боковой пиктограммы** однократным щелчком левой кнопкой мыши: «Щуп напряжения» с фиксацией его на требуемом проводе таким же щелчком.

С. Настраиваем параметры анализатора:

- щелкнув левой кнопкой по изображению анализатора, в появившемся окне «Диаграмма переходного процесса» устанавливаем параметр «Кон. время» равным 2s (либо иное, так чтобы на диаграмме умещалось требуемое число тактовых импульсов для анализа работы схемы).

- щелкнув правой кнопкой по изображению анализатора, выбираем пункт «Добавить трассу» и в появившемся окне «Add Transient Trace» в строке «Щуп

P1» из списка выбираем требуемый провод (точнее имя генератора или щупа).

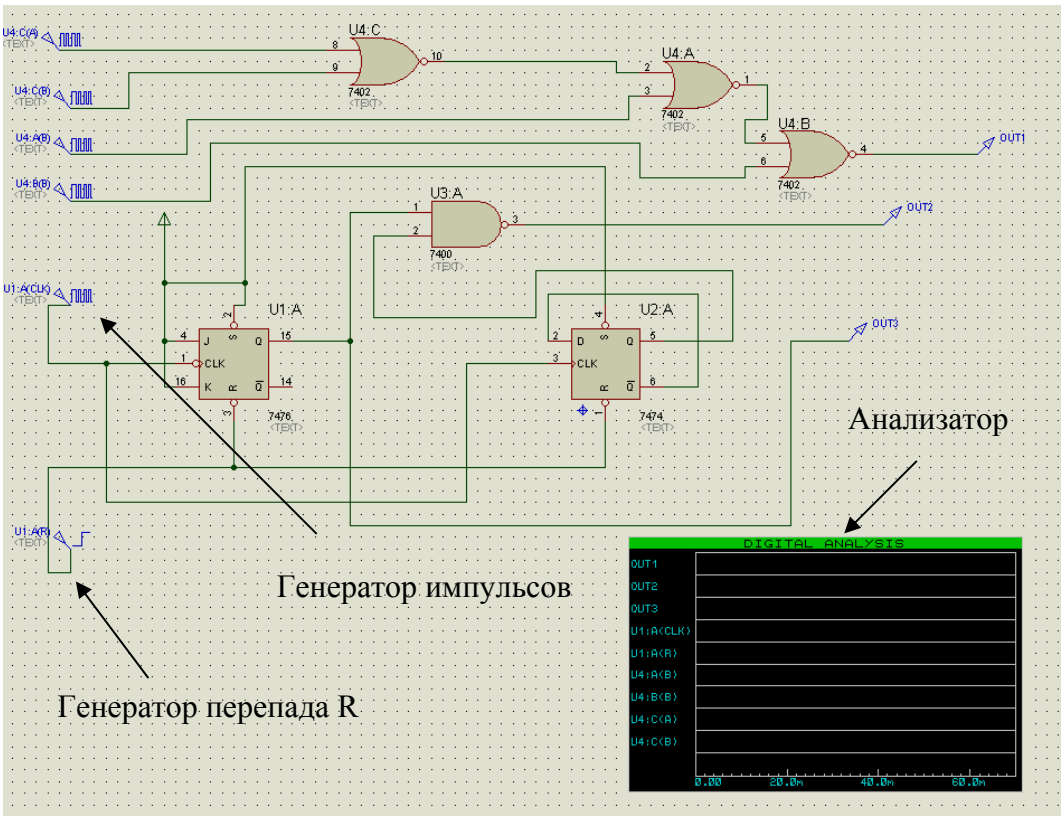


Рис. 12 Схема устройства с подключенными генераторами и анализатором, подготовленная к моделированию

Аналогично подключаем все требуемые провода схемы. Для удобства анализа расположите сигналы в следующем порядке:

- сигнал CLK;
- входные сигналы в порядке УВЕЛИЧЕНИЯ их периода;
- выходные сигналы.

Результирующая схема с подключенными приборами показана на рис. 12.

При работе с анализатором необходимо учитывать:

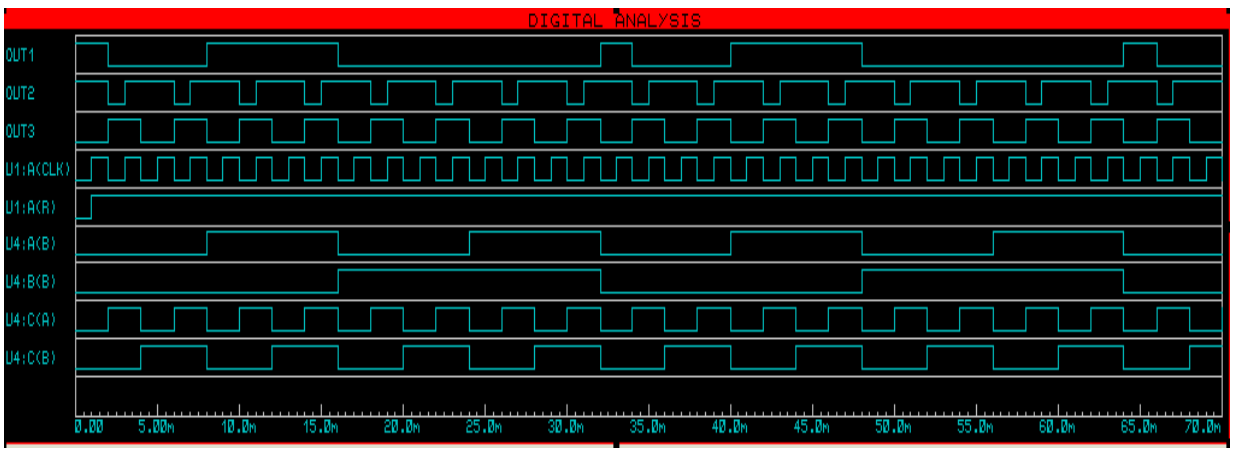


Рис. 13 Результаты моделирования на графическом анализаторе

- названия трасс (проводов) могут быть только английские;
- удалить трассу нельзя.

Моделирование работы схемы производится нажатием клавиши «Пробел». При этом на диаграмме отображаются временные диаграммы в выбранных точках схемы (рис. 13).

Если развернуть окно диаграммы, двойным щелчком левой кнопки мыши по зеленой полоске диаграммы, то удерживая нажатой левую кнопку мыши, появляется **вертикальный перемещаемый маркер**, а в левой части диаграммы отображаются уровни сигналов («1» = H, «0» = L) на временных диаграммах.

1.2 Моделирование работы программируемых цифровых устройств на микроконтроллерах PIC 16C52 с использованием среды MPLAB

При разработке ПО для микроконтроллеров семейства PIC наиболее целесообразно пользоваться специализированным программным продуктом, рекомендуемым производителем контроллеров – средой MPLAB, которая позволяет создавать и отлаживать исходный код. Однако на практике часто пользуются более удобными в плане эмуляции входных сигналов и визуализации результатов продуктами, к которым можно отнести и Proteus. В этом случае MPLAB используется для разработки, редактирования, отладки исходного кода в asm-файле и наблюдения за состоянием внутренних регистров контроллера. Proteus – для имитации входных сигналов и наблюдения за работой выходных портов контроллера к которым подключены внешние исполнительные устройства.

При использовании среды Proteus для моделирования работы PIC контроллеров исходный код по-прежнему создается в среде MPLAB. Кроме того, из-за отсутствия модели PIC16C52 в среде Proteus его приходится заменять другим контроллером (PIC16F877). При этом нужно соблюдать некоторые правила, т.к. контроллер PIC16F877 значительно функциональнее [1].

Рассмотрим **методику моделирования PIC 16C52 с использованием сред Proteus и MPLAB.** Методика включает 4 ЭТАПА.

ЭТАП I. Создание проекта и разработка исходного кода на языке ассемблер **в среде MPLAB** включает в себя следующие шаги:

1. Для написания исходного кода (asm-файла) рекомендуется использовать Microchip MPLAB IDE версии старше 7.50 (на момент написания пособия последняя версия 8.53).

2. После запуска MPLAB в меню выберите «Project»→»Project Wizard». Нажмите «Далее». Выполните предлагаемые шаги:

- шаг 1: создайте папку в которой будут храниться файлы проекта. Для нашего примера, рассматриваемого далее, это будет папка example на диске C.

- шаг 2: из списка предлагаемых контроллеров «Devices» **выберите**

PIC16F877. Нажмите «Далее».

- шаг 3: **выберите язык программирования** - ассемблер, для этого убедитесь, что в окне «Active Toolsuite» **выбран «Microchip MPASM Toolsuite»** и нажмите «Далее».

- шаг 4: **введите название проекта и выберите место его расположения**. Нажмите «Browse…», выберите папку для сохранения проекта, созданную ранее в шаге 1. Введите его имя и нажмите «Сохранить». В нашем примере это будет “c:\example\ex1” – название проекта будет “ex1”, папка “c:\example”. Нажмите «Далее».

- шаг 5: **необязательный шаг** добавления ранее созданных файлов в проект. Ничего не изменяя, нажмите «Далее» затем, «Готово». В главном окне среды и окне со списком файлов проекта заголовок изменится на название_проекта.mcw

3. **Создание asm-файла**. Для этого выберите пункт «Project»→«Add New File to Project…». В появившемся диалоговом окне **введите имя файла с расширением “.asm”** и сохраните. После этого добавленный файл отобразится в окне файлов проекта в папке “Source Files” и появится новое окно с моргающим курсором для ввода исходного кода ПО с именем asm-файла в заголовке.

4. **Разработка исходного кода ПО** производится в соответствие с алгоритмом работы контроллера, синтаксисом языка, системой команд и директив, изложенных в учебном пособии по курсу ПЦУ. Дальнейшее изложение приведено для ПО, реализующего следующий алгоритм:

Пример: разработать ПО для контроллера 16C52, осуществляющее прием байта данных с порта B, операцию арифметического сложения этого байта с константой 0x4D и записывающее результат в РОН с адресом 0x2C. При работе контроллера на порте A должны быть постоянно выставлены сигналы 1011.

При разработке ПО необходимо **обязательно учитывать отличия контроллера 16C52 от 16F877**:

- адресация РОН-ов начинается с адреса 0x20, т.е. первый свободный регистр имеет адрес 0x20, а не 0x07;

- вход счетчика T0CKI является контактом порта A RA4, т.е. фактически 5-ым контактом порта, поэтому обязательное конфигурирование порта A должно быть:

movlw 0x3*

tris PORTA, где * – требуемая Вам комбинация вход/выход 4-х контактов порта A;

- регистр FSR 8-ми разрядный, а не 5-ти;

- ПО **должно** начинаться с блока команд, приведенного на рис. 14.

Для написания кода активизируйте окно asm-файла, вставьте в него обязательную часть, после метки begin расположите Ваш код (рис. 15), закончив его командой end.

5. **Компиляция исходного кода** осуществляется путем нажатия клавиши F10. При первой компиляции среда выдаст сообщение (рис. 16), выберите пункт «Relocatable». В случае отсутствия синтаксических ошибок среда выдаст в окне

```
        PROCESSOR   PIC16F877
        errorlevel -302
        __config    _WDT_OFF
#include <p16f877.inc>
        org   0x000
        movlw   0x20
        movwf   STATUS
        clrw
        movwf   PCLATH
        movlw    0x07
        movwf    ADCON1
        banksel   ADCON0
        clrf    PORTA
        clrf    PORTB
        goto   begin
begin
```

Рис. 14 Обязательная начальная часть ПО для 16C52

```
PORTA       equ 0x05
PORTB       equ 0x06
        movlw 0xFF
        tris PORTB
        movlw 0x30
        tris PORTA
        movlw 0x4D
        addwf PORTB,0
        movwf 0x2C
        bsf PORTA,0x03
        bcf PORTA,0x02
        bsf PORTA,0x01
        bsf PORTA,0x00
        goto begin
        end
```

Рис. 15 Код ПО для 16C52

компиляции «Output» сообщение «BUILD SUCCEEDED», при их наличии - сообщение «BUILD FAILED». В этом же окне появятся сообщения о характере ошибок и номерах строк ПО, в которых они обнаружены. Проанализируйте ошибки, после чего закройте окно «Output». **Устраните синтаксические ошибки** и вновь откомпилируйте код. Завершив работу, **сохраните результат** (пункт меню «Project»→«Save Project») и выйдите из среды MPLAB.

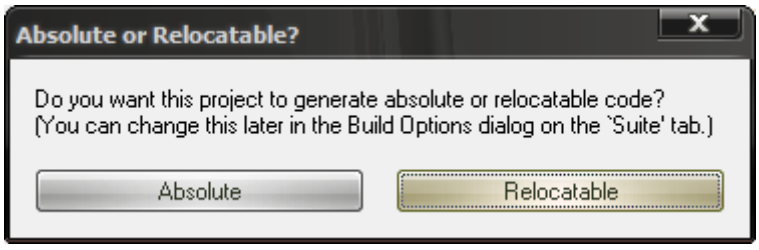


Рис. 16 Сообщение среды перед первой компиляцией

ЭТАП II. Создание схемы соединения контроллера с внешними устройствами **в среде Proteus**. Это делается для организации взаимодействия между средой MPLAB и Proteus, т.к. Proteus позволяет наглядно формировать входные и визуализировать выходные сигналы контроллера в процессе его работы вместе с подключенными к нему устройствами. Этап включает в себя следующие шаги:

1. Используя п.п. 1-7 методики из раздела 1.1 настоящего пособия **создайте схему**, содержащую контроллер с подключенными к нему источниками **входных сигналов** и **выходными** исполнительными **устройствами**:

- файл проекта Proteus **сохраните в папке проекта MPLAB**;
- при выборе элементов схемы используйте контроллер **PIC 16F877** (введя его название в строке «Keywords» окна «Pick Devices»);

- к **выходным** контактах контроллера (в нашем примере это 4 младших разряда порта A, RA3 – RA0, RA3 – старший разряд) подключите индикаторы : **LOGICPROBE (BIG)** (найдя их аналогично);

- на **входные** контакты портов контроллера (в нашем примере это все 8 контактов порта B, RB7 – RB0, RB7 – старший разряд) подайте входные сигналы. Для этого используйте либо фиксируемую в нажатом положении кнопку **LOGICSTATE**, либо аналогичную кнопку, но без фиксации - LOGICTOGGLE. В результате у Вас должна получиться схема (рис. 17).

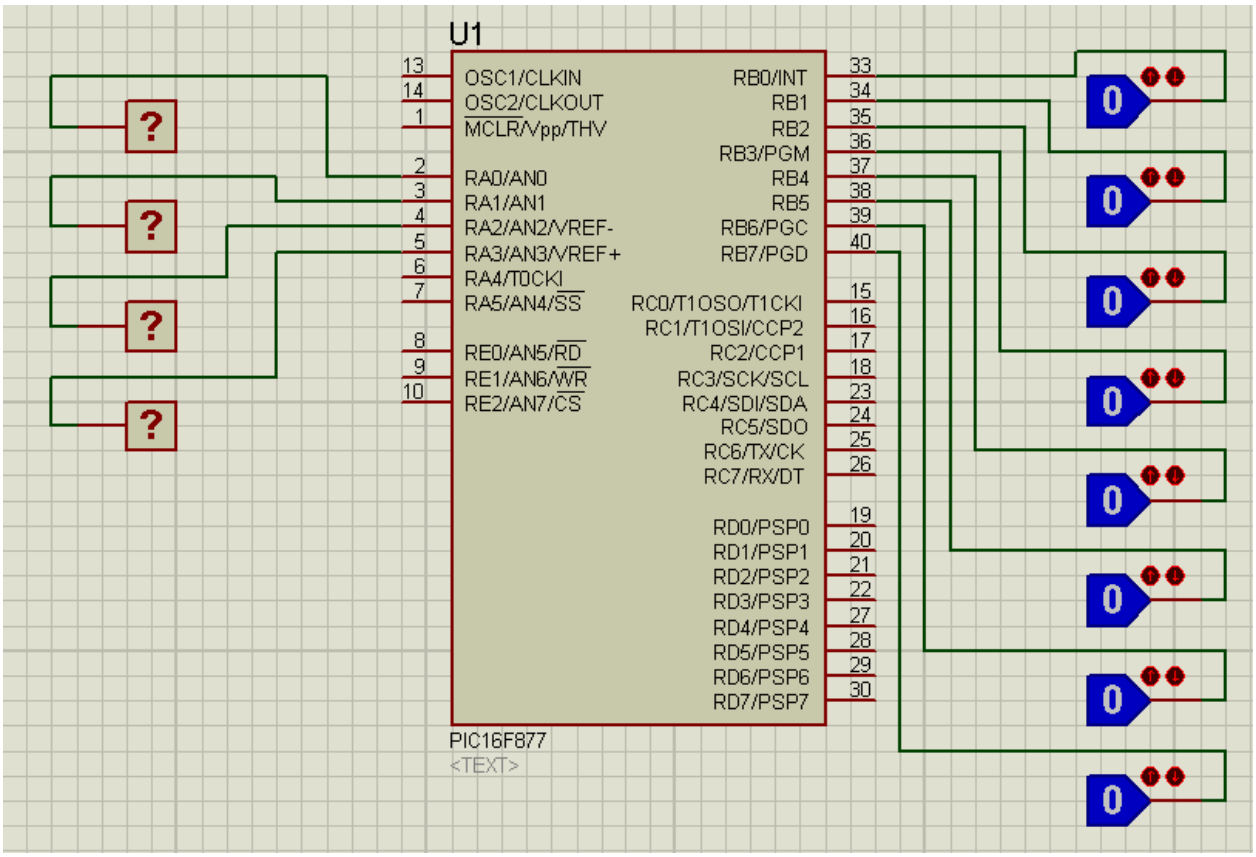


Рис. 17 Схема устройства в среде Proteus

2. В свойствах контроллера (двойной щелчок по нему левой клавишей мыши) необходимо выставить его частоту равной 4MHz. Измените поле ‘Processor Clock Frequency’ и нажмите «ОК».

3. Сохраните проект и закройте Proteus.

ЭТАП III. Подключение схемной модели контроллера **из среды Proteus к исходному коду в среде MPLAB** для этого:

1. **Откройте Ваш проект** (созданный в п. I) в среде **MPLAB**, через пункт меню «Project» → «Open…». Закройте окна «Output», «ex1.mcw».

2. **Выберите средство отладки** ПО в MPLAB через пункт меню ‘Debugger’→’Select Tool’→’Proteus VSM’. Должно появиться окно «Proteus MPLAB VSM Viewer». Если оно не появилось, активизируйте его через пункт меню

«View»→« Proteus VSM Viewer».

3. В появившемся окне «Proteus MPLAB VSM Viewer» через стандартную пиктограмму «Open Design» **откройте Вашу схему**, созданную в разделе II методики в среде Proteus, и **масштабируйте ее** для удобства наблюдения.

4. **Откройте дополнительные окна для просмотра содержимого регистров** контроллера. Для этого выберите пункт «View»→«File Registers» для регистров общего назначения и пункт «View»→«Special Functions Registers» для специальных регистров. Упорядочьте их на экране, выбрав пункт «Window»→ «Tile Vertically» (Рис. 18), закрыв лишние окна.

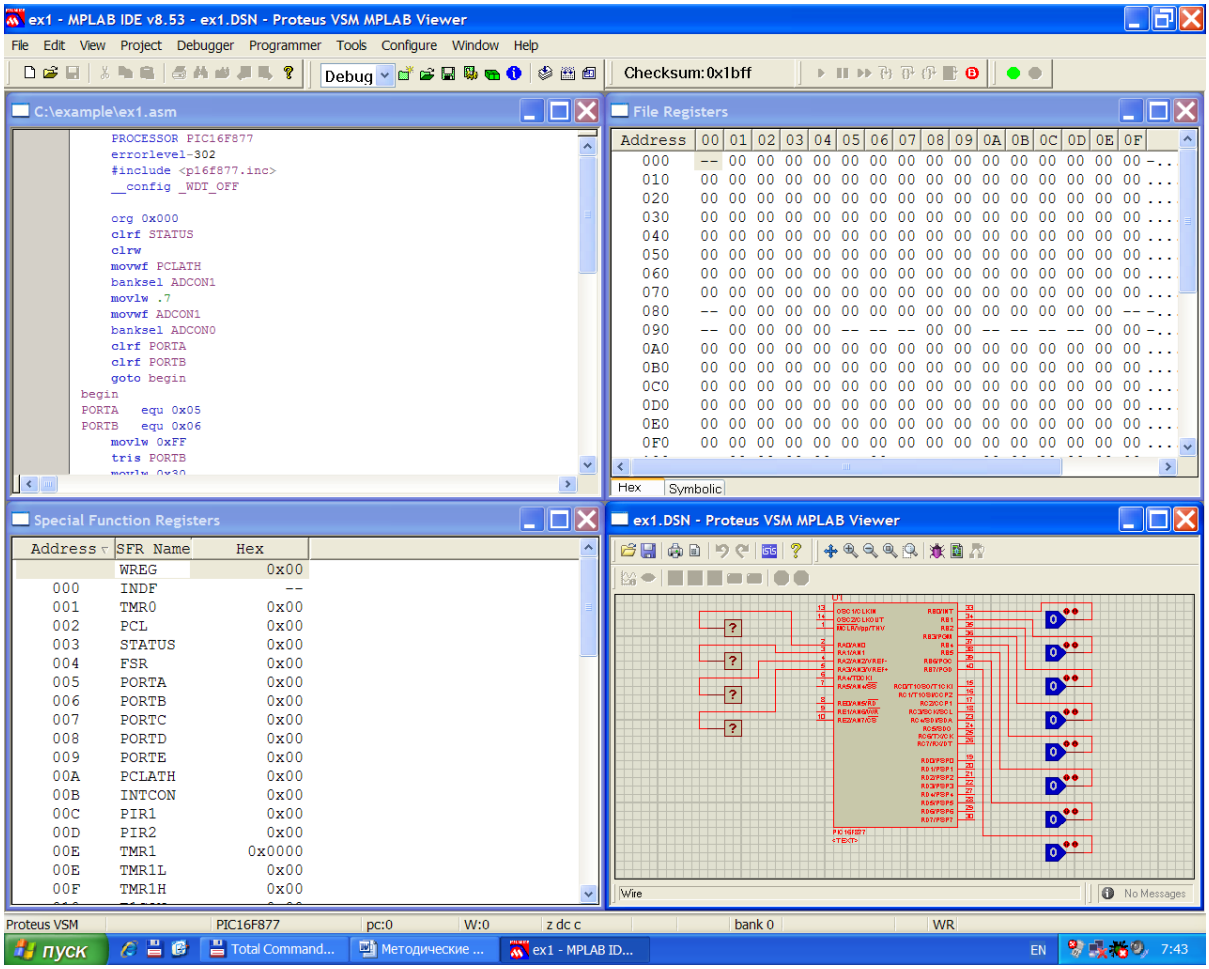


Рис. 18 Вид среды MPLAB, подготовленной для отладки ПО контроллера 16C52, с подключенным окном среды Proteus

ЭТАП IV. Проведение отладки программы производится в следующей последовательности:

1. **Откомпилируйте ПО**, нажав клавишу F10. Если п.5 этапа I был выполнен правильно, то ошибок быть не должно, в противном случае устраните их в тексте asm-файла. При отсутствии ошибок закройте окно «Output».

2. **Начните пошаговую отладку** ПО:

- **установите в** окне **Proteus** требуемые по условиям задачи исходные значения **входных сигналов**;

- **начните отладку**, нажав на панели инструментов кнопку с зеленым кругом (либо **клавишу F12**). При этом напротив первой выполняемой команды asm-файла появится зеленая стрелка;

- проведите пошаговую отладку, нажимая клавишу F7 и подавая в окне Proteus требуемые входные сигналы. **ПРИ ИЗМЕНЕНИИ ВХОДНЫХ СИГНАЛОВ СЛЕДИТЕ, ЧТОБЫ ЭТИ ИЗМЕНЕНИЯ «УВИДЕЛ» КОНТРОЛЛЕР, Т.Е. ЧТОБЫ ОНИ ПОЯВИЛИСЬ В РЕГИСТРАХ ПОРТОВ В ОКНЕ СПЕЦИАЛЬНЫХ РЕГИСТРОВ MPLAB (А НЕ ТОЛЬКО В ОКНЕ PROTEUS)**. Пока этого не произошло, контроллер будет обрабатывать «прошлые» значения входных сигналов. Каждое нажатие клавиши F7 приводит к выполнению одной команды контроллером. Необходимо чтобы ПЕРЕД выполнением команды обработки входных сигналов (по крайней мере за 1 шаг перед ней) их новое значение уже было в регистре порта (в нашем примере это порт B). Иногда может понадобиться повторное нажатие кнопки, чтобы изменение сигнала появилось в регистре порта, либо повторная эмуляция (с выходом через нажатие кнопки с красным кружком и повторным ее запуском через нажатие кнопки с зеленым кружком). В нашем примере такой командой является команда **addwf PORTB,** которая складывает прочитанный из порта B сигнал с байтом константы, находящимся в регистре W. Поэтому, например, если в окне Proteus изменить сигнал на контактах порта B непосредственно перед выполнением этой команд (по нажатию клавиши F7), то MPLAB этого изменения «не увидит» и команда выполнится со «старым» содержимым порта B.

- в процессе работы наблюдайте за изменением контактов портов контроллера и его внутренних регистров. Отладка ПО заключается в том, чтобы проверить правильность функционирования (соответствие алгоритму из задания) контроллера при ВСЕХ возможных комбинациях входных сигналов во ВСЕХ заданных режимах работы;

- на практике для наблюдения за состоянием специальных регистров контроллера иногда бывает удобно воспользоваться специальным окном, в котором будет отображаться состояние только выбранных Вами регистров. Это делается через пункт меню «View» →«Watch»;

- для проверки работоспособности контроллера при большом числе возможных комбинаций входных сигналов, либо большом числе команд в коде, можно воспользоваться режимом анимации (пункт Debugger → Animate). В этом режиме ПО начнет автоматически выполняться с некоторой фиксированной скоростью и останется лишь изменять входные сигналы;

- для завершения отладки нажмите на кнопку с красным кругом (либо комбинацию клавиш CTRL-F12).

3. При совпадении результатов работы контроллера с заданным алгоритмом, предъявите работу преподавателю, при наличии расхождений внесите изменения в исходный код и повторите отладку. **После внесения любых изменений в asm-файл повторно откомпилируйте код**.

1.3 Моделирование программируемых цифровых устройств на основе PIC16F877 в среде Proteus.

1.3.1 Назначение, схемы и описания виртуальных стендов.

Виртуальный стенд представляет собой модель реального лабораторного стенда на промышленном контроллере TTF 5.0 (см. п. 4 Методических указаний к выполнению лабораторных работ по курсу «Программируемые цифровые устройства», далее - МУЛР), содержащую электрическую схему соединения модели микроконтроллера PIC 16F877 с моделями внешних устройств. Схема создана в среде Proteus по правилам, изложенным в п. 1.1 данного пособия. Виртуальный стенд предназначен для записи в модель контроллера, его «прошивки», полученной в среде MPLAB, и наблюдения за работой контроллера с сигналами датчиков и исполнительных устройств стенда в среде Proteus, не подключаясь к реальному оборудованию. Число виртуальных стендов – 4, по числу реальных стендов, все они индивидуальны, не взаимозаменяемы и различаются не только видом периферийного оборудования, но и уровнями сигналов. Методика использования такого «стенда» приведена ниже в п. 1.3.2, а сейчас рассмотрим более подробно его состав и особенности использования его компонентов по сравнению с реальными устройствами «этих же» стендов.

ВНИМАНИЕ: при работе с виртуальными стендами запрещено менять внутренние параметры моделей. Это может привести к неработоспособности стенда в целом.

Схема стенда № 1 (нумерация совпадает с реальными стендами, **уровни сигналов представлены в Таблице № 1** в п. 4.2 МУЛР) приведена на рис. 19 и включает в себя:

- **мотор с датчиком числа его оборотов**. Модель по сигналам управления полностью совпадает с реальным устройством, описанным в п. 4.2 МУЛР. Управляется двигатель цифровым сигналом.

- датчик температуры DS1820. Модель по сигналам управления и системе команд полностью совпадает с реальным устройством, описанным в п. 11.3 учебного пособия по курсу «Программируемые цифровые устройства». Модель позволяет имитировать изменение текущей температуры датчика, путем нажатия на символы «+», «-». При этом на его выходе по предварительному запросу контроллера формируется последовательный код в соответствие с п. 11.3 учебного пособия по курсу «ПЦУ», а значение температуры отображается во внутреннем окне модели.

- **1.5 разрядный семисегментный индикатор**. Модель по сигналам управления полностью совпадает с реальным устройством, описанным в п. 4.2 МУЛР.

- энергонезависимая память 25C040 объемом 512 байт с интерфейсом SPI. Модель по сигналам управления, режимам работы и системе команд полностью совпадает с реальным устройством, описанным в п. 14.3 учебного пособия по курсу «Программируемые цифровые устройства». Дополнительно модель позволяет наблюдать за состоянием всех своих внутренних регистров. Для этого,

нажмите кнопку «Пауза», а затем, щелкнув правой кнопкой мыши по изо-

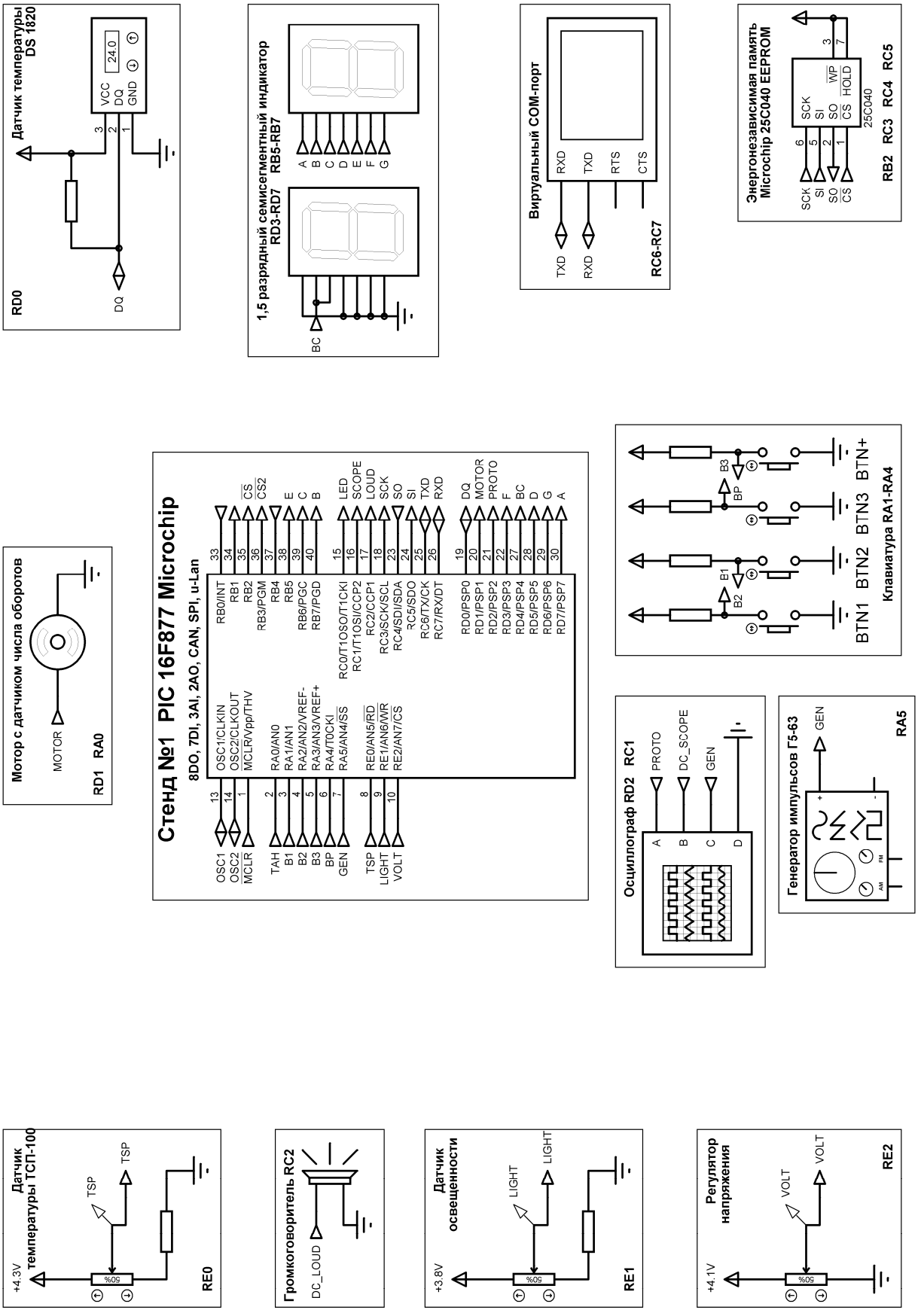


Рис. 19 Схема виртуального стенда №1

бражению модели микросхемы памяти и выбрав пункт «Internal Memory», в

появившемся окне «SPI Memory Internal Memory» Вы получите доступ к текущему состоянию внутренних регистров РПЗУ. Обратите внимание, что состояние регистров будет отображено на момент времени непосредственно ПЕРЕД остановкой моделирования. В режиме моделирования это окно не отображается, т.е. состояния регистров можно наблюдать только в момент остановки моделирования, а не в процессе их изменения.

- **клавиатура** с 4-мя не фиксируемыми клавишами. Модель по сигналам управления полностью совпадает с реальным устройством, описанным в п. 4.2 МУЛР. Эмуляция удерживаемой в нажатом состоянии клавиши производится щелчком левой кнопки мыши по красному кружочку рядом с моделью кнопки.

- **аналоговый осциллограф**. Стандартный компонент Proteus, по органам управления соответствует 4-х канальному реальному осциллографу. Используется 3 входа – A, B, C. На вход A подключен цифровой сигнал с контакта RD2 микроконтроллера, на вход B – аналоговый сигнал с выхода интегратора, подключенного к контакту RC1 модуля ШИМ контроллера (в соответствие с Таблицей №1 в п. 4.2 МУЛР). На вход C подключен выход генератора прямоугольных импульсов, рассматриваемый ниже. Это сделано для удобства визуальной настройки длительности импульсов этого генератора. Активизация компонента происходит только в режиме «Play».

- **генератор импульсов**. Стандартный компонент Proteus, имеет органы плавного и дискретного управления амплитудой (регулятор Level – плавно, регулятор Range - множитель) и частотой (регулятор Centre – плавно, регулятор Range - множитель), и видом генерируемого сигнала (в нашем случае используется как генератор прямоугольных импульсов). Соответствует реальному генератору Г5-63 (см. ПРИЛОЖЕНИЕ №1 к МУЛР), сигнал с которого подается на вход RA5 микроконтроллера PIC 16F877 (и параллельно, для удобства визуальной настройки, на вход C осциллографа). Режим генерации прямоугольных импульсов возможен только со скважностью 2. Активизация компонента происходит только в режиме «Play».

- **регулятор напряжения**. Модель реализована в виде переменного резистора и по сигналам управления полностью совпадает с реальным устройством, описанным в п. 4.2 МУЛР.

- **датчик освещенности**. Модель реализована в виде переменного резистора, выходное напряжение с которого, при изменении положения движка, имитирует изменение выходного напряжения с датчика при изменении его освещенности. По сигналам управления полностью совпадает с реальным устройством, описанным в п. 4.2 МУЛР.

- **датчик температуры ТСП-100**. Модель реализована в виде переменного резистора, выходное напряжение с которого, при изменении положения движка, имитирует изменение выходного напряжения с датчика при изменении температуры окружающей среды. По сигналам управления полностью совпадает с реальным устройством, описанным в п. 4.2 МУЛР.

- **громкоговоритель**. Модель реализована в виде встроенного компонента Proteus со встроенным (на схеме не показан) преобразователем «частота-напряжение». Преобразователь выполняет роль интегратора, выделяющего по-

стоянную составляющую из импульсной последовательности с ШИМ на контакте RC2 микроконтроллера. При моделировании формирования звуковых сигналов с помощью модели громкоговорителя необходимо учитывать следующие особенности:

1. Для формирования **паузы** в звучании громкоговорителя (фактически его выключения) контроллер ОБЯЗАТЕЛЬНО должен формировать на контакте RC2 **импульсный сигнал со скважностью 2** и произвольным периодом а не просто уровень логического «0».

2. У модели предусмотрена связь со звуковой картой Вашего ПК, поэтому все звуковые сигналы будут слышны в колонках или наушниках.

- **интерфейс связи с ПК**. Модель реализована в виде виртуального COM порта ПК, с помощью которого на контакт RX модели микроконтроллера можно отправлять **отдельные байты** (или группы байт, но по одному) и принимать его ответы (с выхода TX модели микроконтроллера). Таким образом, эмулируется работа ПК (например, через утилиту S-Test см. Гл. 3 МУЛР) с модулем UART реального микроконтроллера. Визуально модель появляется (только в режиме «Play») в виде окна (терминала) ввода-вывода в котором отображаются символы или их коды в канале связи. При работе с терминалом нужно учитывать следующие особенности:

1. **Для передачи байта** на контроллер, на клавиатуре ПК нажимается клавиша (кроме некоторых специальных клавиш, таких как Esc, Enter, F1 – F12 и некоторых других), при этом происходит автоматическая отправка этого байта на микроконтроллер, но в окне терминала переданный символ (или его код) НЕ отображается. Чтобы включить отображение вводимых символов нажмите правую кнопку мыши в окне “Virtual Terminal” и выберите пункт меню "Echo Typed Characters". Если Вы хотите видеть не символы, а их коды, аналогично выберите "Hex Display Mode". При этом отображение вводимых (и отправляемых) символов происходит без пробелов, а их кодов – с пробелами. Отправлять можно только буквенно-цифровые символы с клавиатуры ПК. Отправка произвольных байт, например, 0xDF - невозможна.

2. **Прием** осуществляется автоматически при включении режима «Play» (или пошагового исполнения ПО). Отображение принимаемых символов происходит БЕЗ разделения их с отправляемыми.

Схема стенда № 2 (нумерация совпадает с реальными стендами, **уровни сигналов представлены в Таблице № 2** в п. 4.2 МУЛР) приведена на рис. 20 и включает в себя:

- **мотор с датчиком числа его оборотов**. Модель построена с использованием встроенного двигателя постоянного тока, к которому специальным образом добавлены (НЕ показанные на рис. 20) интегратор и датчик числа оборотов. Интегратор преобразует ШИМ сигнал с контакта RC2 микроконтроллера в постоянное напряжение, управляющее двигателем в соответствие с Табл. № 2 п. 4.2 МУЛР. Рядом с двигателем расположен индикатор уровня постоянного напряжения, поступающего на двигатель с модуля ШИМ контроллера (точнее с выхода интегратора). Модель датчика числа оборотов по сигналам управления пол-

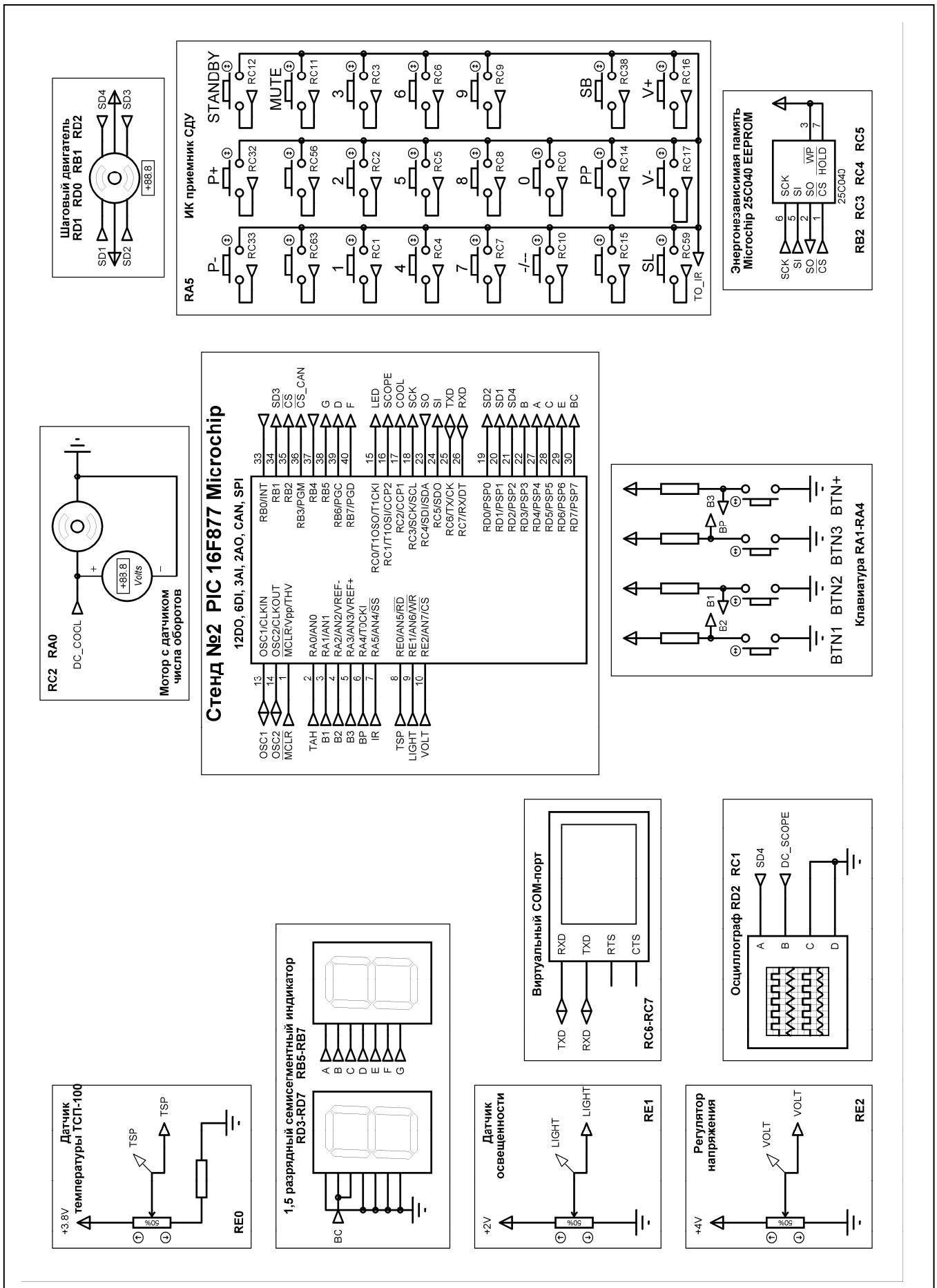


Рис. 20 Схема виртуального стенда №2

ностью совпадает с реальным устройством, описанным в п. 4.2 МУЛР.

- **шаговый двигатель**. Модель по сигналам управления полностью совпадает с реальным устройством, описанным в п. 4.2 МУЛР.

- **ИК-приемник СДУ**. В качестве модели фотоприемника использована группа генераторов прямоугольных импульсов, с регулируемой скважностью (расстановкой) импульсов. Включение генератора производится нажатием соответствующей кнопки, эмулирующей кнопку пульта СДУ. При однократном нажатии кнопки происходит генерация пачек импульсов, форма которых соответствует рис. 17 п. 4.4 МУЛР. В режиме «Play» кнопку надо держать нажатой не менее 2-х секунд, в режиме «F11» - ее надо зафиксировать в нажатом положении. Одновременное нажатие 2-х и более кнопок одновременно – ЗАПРЕЩЕНО, т.к. приводит к непредсказуемому поведению модели. Об этом событии Вам дополнительно скажет уровни напряжения в виде желтых квадратиков рядом с кнопками во время моделирования. При использовании модели необходимо учитывать, что формирование пачки занимает время около 25 000 мкс, что делает невозможным использование отладки по F11 для отладки приема. В этих случаях рекомендуется использовать готовую процедуру приема пакетов СДУ, описанную в Гл. 16 учебного пособия по курсу «Программируемые цифровые устройства».

- **1.5 разрядный семисегментный индикатор**. Модель по сигналам управления полностью совпадает с реальным устройством, описанным в п. 4.2 МУЛР.

- **интерфейс связи с ПК**. См. описание стенда №1.

- **энергонезависимая память 25C040 объемом 512 байт с интерфейсом SPI.** См. описание стенда №1.

- **клавиатура**. См. описание стенда №1.

- **аналоговый осциллограф**. См. описание стенда №1.

- **регулятор напряжения**. См. описание стенда №1, с учетом отличий уровней сигналов с датчика в соответствие с Табл. №2 в п. 4.2 МУЛР.

- **датчик освещенности**. См. описание стенда №1, с учетом отличий уровней сигналов с датчика в соответствие с Табл. №2 в п. 4.2 МУЛР.

- **датчик температуры ТСП-100**. См. описание стенда №1, с учетом отличий уровней сигналов с датчика в соответствие с Табл. №2 в п. 4.2 МУЛР.

Схема стенда № 3 (нумерация совпадает с реальными стендами, **уровни сигналов представлены в Таблице № 3** в п. 4.3 МУЛР) представлена на рис. 21 и включает в себя:

- **правый вентилятор**. Модель реализована в виде двигателя постоянного тока и по сигналам управления соответствует вентилятору2 в п. 4.3 МУЛР, управляемому модулем ШИМ контроллера. Рядом с двигателем расположен индикатор уровня постоянного напряжения, поступающего на двигатель с модуля ШИМ контроллера (точнее с выхода интегратора). Интегратор, включенный между контактом RC2 микроконтроллера и двигателем, преобразующий последовательность прямоугольных импульсов в постоянное напряжение, на рис. 21 не показан.

- **фотодиодные датчики положения D1 (левый) и D2 (правый)**. Модели реализованы в виде переменных резисторов, выходное напряжение с которых, при изменении положения движка, имитирует изменение выходного напряжения

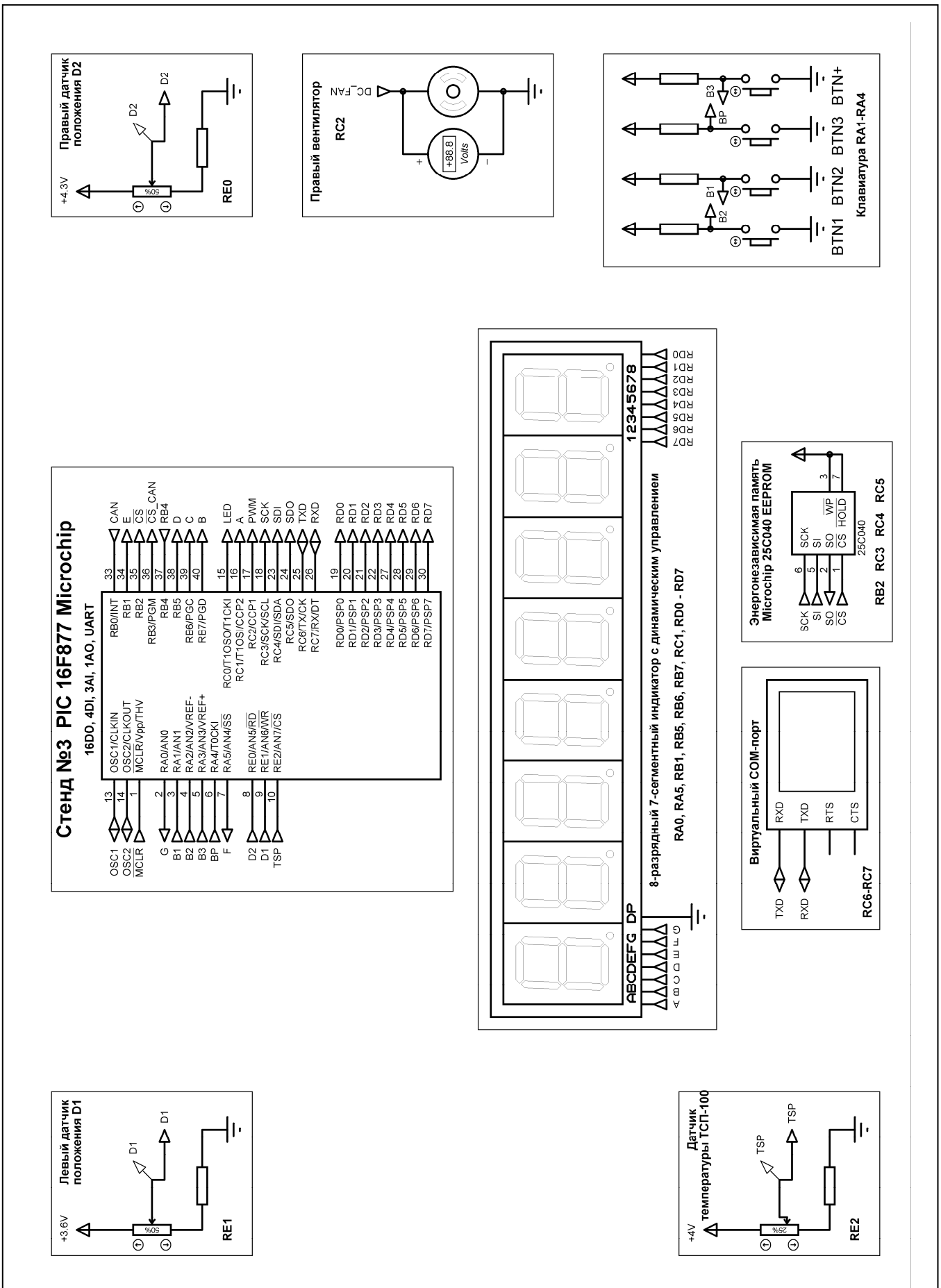


Рис. 21 Схема виртуального стенда №3

с датчиков при изменении текущего положения шарика в трубе. По сигналам

управления полностью совпадает с реальным устройством, описанным в п. 4.2 МУЛР. При использовании этих моделей **необходимо учитывать** следующее:

1. Изменять напряжения на обоих датчиках можно только в ПАУЗЕ моделирования или при ПОШАГОВОМ ИСПОЛНЕНИИ программы. Это связано с тем, что модели в виртуальном стенде не зависимы и позволяют выставить любую пару значений со своих выходов, а в реальном стенде их значения зависят от положения шарика в трубе и могут принимать только фиксированные пары значений.

2. Пары значений напряжений с датчиков должны соответствовать ОДИНАКОВОЙ координате шарика в трубе и могут принимать значения, соответствующие этой координате на рис. 19, 20 в МУЛР.

3. В противном случае при моделировании могут возникать ситуации, соответствующие таким сигналам с датчиков, которых в реальном стенде не может быть при любом положении шарика.

- **интерфейс связи с ПК**. См. описание стенда №1.

- **энергонезависимая память 25C040 объемом 512 байт с интерфейсом SPI.** См. описание стенда №1.

- **клавиатура** . См. описание стенда №1.

- **8 разрядный 7-ми сегментный индикатор с динамическим управлением**. Модель по сигналам управления полностью совпадает с реальным устройством, описанным в п. 4.3 МУЛР.

- **датчик температуры ТСП-100**. См. описание стенда №1, с учетом отличий уровней сигналов с датчика в соответствие с Табл. №3 в п. 4.3 МУЛР.

Схема стенда № 4 (нумерация совпадает с реальными стендами, **уровни сигналов представлены в Таблице № 4** в п. 4.4 МУЛР) приведена на рис. 22 и включает в себя:

- **8 разрядный 7-ми сегментный индикатор с динамическим управлением**. Модель по сигналам управления полностью совпадает с реальным устройством, описанным в п. 4.4 МУЛР.

- **интерфейс связи с ПК**. См. описание стенда №1.

- **регулятор напряжения**. См. описание стенда №1, с учетом отличий уровней сигналов с датчика в соответствие с Табл. №4 в п. 4.4 МУЛР.

- **датчик освещенности**. См. описание стенда №1, с учетом отличий уровней сигналов с датчика в соответствие с Табл. №4 в п. 4.4 МУЛР.

- **датчик температуры ТСП-100**. См. описание стенда №1, с учетом отличий уровней сигналов с датчика в соответствие с Табл. №4 в п. 4.4 МУЛР.

- **алфавитно-цифровой LCD индикатор 2 x 16 символов с интерфейсом HD44780 Hitachi**. Модель по сигналам управления полностью совпадает с реальным устройством, описанным в п. 4.4 МУЛР.

- **матричный индикатор 4 символа 5x4**. Модель по сигналам управления совпадает с реальным устройством, описанным в п. 4.4 МУЛР. Небольшое отличие заключается в наличие 3-х «лишних нижних» строк в модели индикатора, связанных с использованием в качестве модели встроенных 8x8 индикаторов Proteus. Они не подключены к контроллеру, всегда потушены и на работу стенда не влияют.

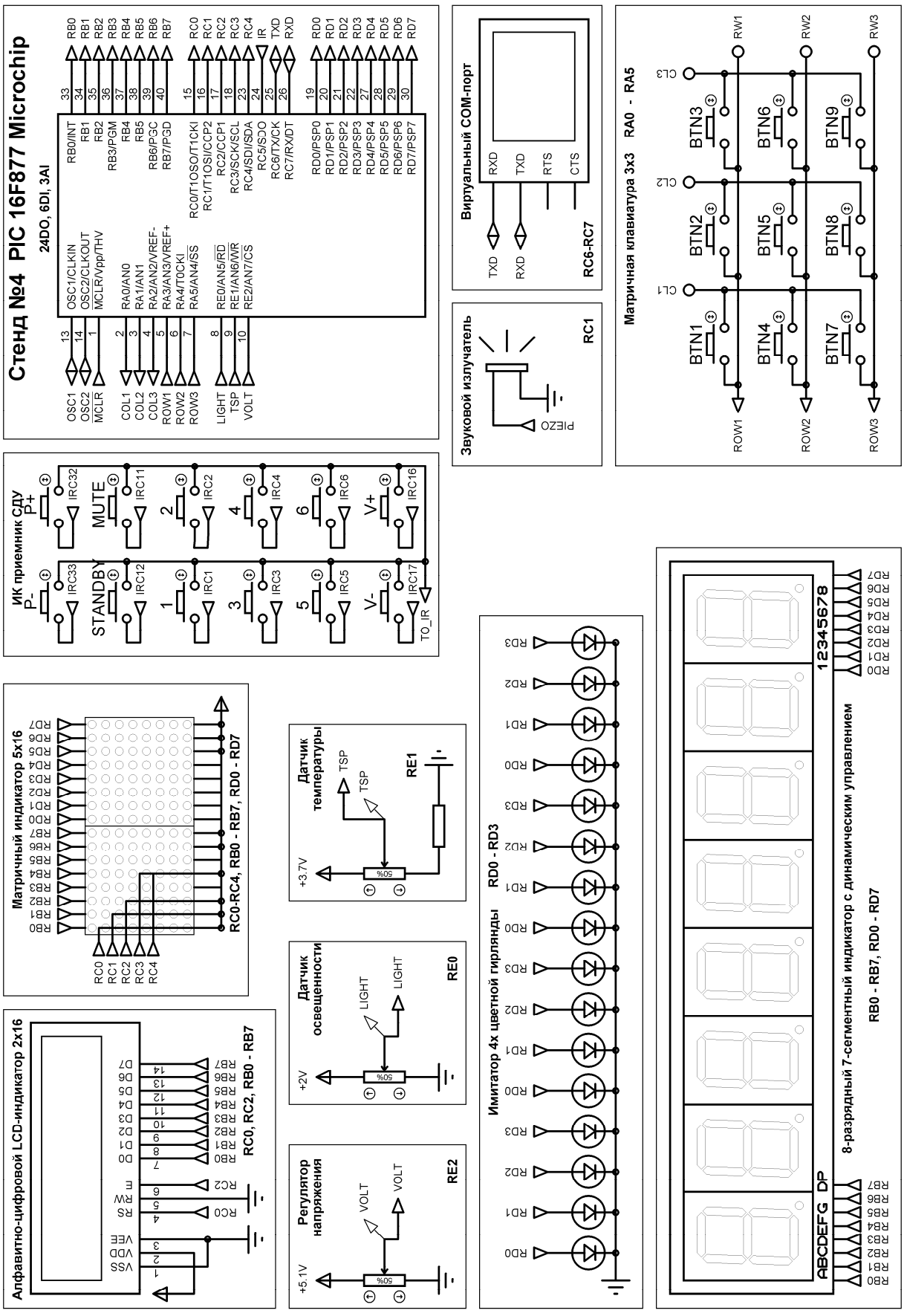


Рис. 22 Схема виртуального стенда №4

- имитатор 4-х цветной гирлянды. Модель по сигналам управления полностью совпадает с реальным устройством, описанным в п. 4.4 МУЛР.

- матричная клавиатура 3 х 3. Модель по сигналам управления полностью совпадает с реальным устройством, описанным в п. 4.4 МУЛР.

- **ИК-приемник СДУ**. См. описание стенда №2, за исключением отсутствия нескольких кнопок пульта СДУ (см. рис. 22)

- звуковой излучатель (пищалка). Модель по сигналам управления совпадает с реальным устройством, описанным в п. 4.4 МУЛР. У модели предусмотрена связь со звуковой картой Вашего ПК, поэтому все звуковые сигналы будут слышны в колонках или наушниках

1.3.2 Методика моделирования работы PIC16F877 в средах MPLAB и Proteus с использованием виртуальных стендов

Моделирование работы микроконтроллера PIC 16F877 на виртуальных стендах, описанных в предыдущем пункте, является ОБЯЗАТЕЛЬНЫМ предварительным шагом выполнения лабораторных работ с использованием реального стендового оборудования в лаборатории БЛК-204. Без проведения моделирования на виртуальном стенде и сдачи их положительных результатов инженеру лаборатории студенты не допускаются к работе на реальных стендах.

Методика моделирования включает следующие этапы:

1. В соответствие с Вашим заданием (Глава 2 МУЛР), схемой подключения периферийного оборудования к микроконтроллеру (Глава 4 МУЛР), правилами п.1.2 (выполняется только Этап I) данного пособия **разработайте исходный код ПО для микроконтроллера PIC 16F877 в среде MPLAB**. Отладьте его, по крайней мере, на отсутствие синтаксических ошибок. Откомпилируйте проект (клавиша F10) и убедитесь в наличие файла с расширением **cof** в папке проекта MPLAB. Он позволит «видеть» содержимое asm-файла в среде Proteus. После этого **среду MPLAB не закрывайте**.

2. Загрузите среду Proteus и через пункт меню «Open Design» **загрузите схему** требующегося Вам **виртуального стенда** (один из файлов stand1.dsn – stand4.dsn). **Сохраните** схему, через пункт меню «File-Save as» (нажав «ОК» на предупреждающее сообщение среды), **под другим именем в папку проекта MPLAB**. Далее везде уже работаем только с этим файлом схемы.

3. **ОБЯЗАТЕЛЬНО удалите** не используемые в Вашем задании элементы виртуального стенда, т.к. они будут **сильно «тормозить моделирование»** и вызывать лишние предупреждающие сообщения среды. Для этого выделите мышью (с нажатой левой кнопкой) РАМКУ элемента и нажмите клавишу DEL. Отмена ошибочного удаления – Edit/Undo или Ctrl-Z. **Загрузите Вашу прошивку** в модель микроконтроллера PIC 16F877. Для этого, дважды щелкните левой кнопкой мыши по изображению микроконтроллера. В появившемся окне «Edit Component» в поле «Program File» введите имя **cof-файла** (из папки с проектом MPLAB), полученного в пункте 1 методики, и нажмите «ОК». Сохраните проект.

4. **Проведите пошаговую отладку** Вашего ПО в среде Proteus. Для этого

нажмите кнопку «Pause», при этом появится окно с текстом исходного asm-файла и активной первой выполняемой командой. Отладка проводится в несколько этапов (особенно если это первая отладка Вашего ПО), причем на практике в зависимости от сложности ПО отдельные этапы можно не выполнять. Однако, в общем случае отладка может производиться в следующей последовательности:

a. Для наблюдения за состоянием внутренних регистров микроконтроллера (кроме портов, т.к. их состояния отображаются средой моделирования автоматически) **добавьте** специальное **окно «Watch Window»**. Для этого через пункт меню Debug/Watch вызовите это окно. Щелкнув правой кнопкой мыши по нему, в появившемся меню выберите пункт «Add Items (By Name)…» или «Add Items (By Address)…» для добавления имени или адреса регистра, за состоянием которого Вы хотите наблюдать в процессе моделирования. Первый пункт позволяет (двойным щелчком левой кнопки мыши) выбрать специальные регистры контроллера, второй (нажатием кнопки «Add») – произвольный РОН по его абсолютному, без учета банка памяти, адресу. При этом также можно выбрать удобный формат представления содержимого, например, двоичный. Остальные окна моделирования можно закрыть.
b. Установите постоянные значения всех входных сигналов. Для этого требуемые Вам кнопки зафиксируйте в нажатом положении, а движки переменных резисторов установите в положение, при котором имитируется нужное Вам значение напряжений с датчиков. При изменении сигналов обязательно учитывайте особенности моделей из п. 1.3.1.
c. **Нажимая клавишу «F11»** (НЕ кнопку «Step» в среде Proteus) на клавиатуре, проведите отладку, наблюдая за состоянием периферийных устройств и изменяя напряжения с моделей датчиков.
d. При необходимости установите точки принудительной остановки выполнения программы, так называемые «BreakPoints». Это делается в тех случаях, когда пошаговая отладка либо занимает большое время, например, при выполнении процедур задержек, приема контроллером групп битов, либо часть кода не представляет интереса для «медленной» пошаговой отладки, например, уже была отлажена ранее. Для этого в окне исходного кода, на требуемой команде ДВА раза щелкните левой кнопкой мыши, при этом слева рядом с командой появится красный кружок. Удаление точки остановки производится путем ДВУХ двойных щелчков левой кнопкой мыши по этой команде. Выбор команды, на которой устанавливается точка остановки, может производиться в соответствие с рис. 23.

В обоих случаях при нажатии кнопки «Play» будет автоматически (а соответственно и быстро) выполнена часть программы, расположенная ДО точки остановки (команда на точке остановки выполнена НЕ БУДЕТ). После этого выполнение ПО будет приостановлено, и далее может быть продолжено Вами уже в режиме обычной пошаговой отладки по F11, когда за каждое нажатие выполняется одна команда. Т.е. в случае 1 на рис. 23 можно будет отлаживать сомнительный участок кода, а в случае 2 – «длинный или неинтересный» кусок будет

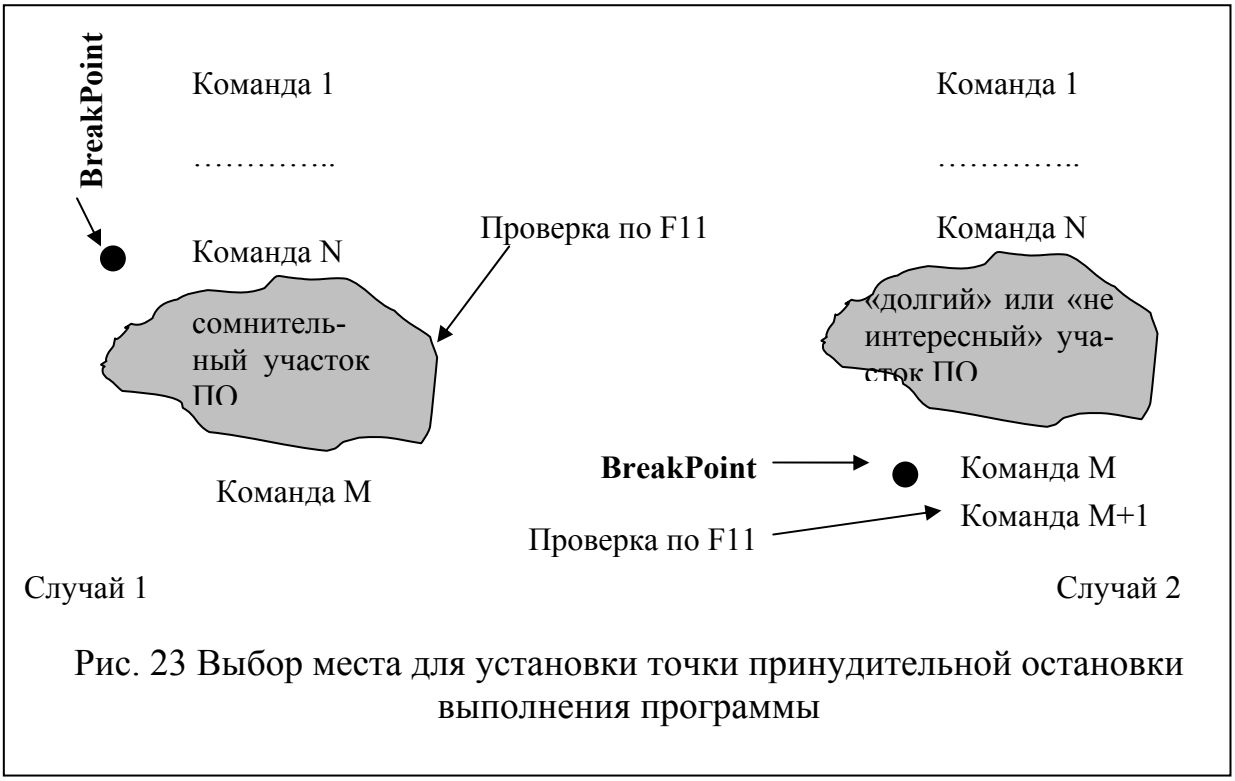


Рис. 23 Выбор места для установки точки принудительной остановки выполнения программы

выполнен автоматически и пошаговая отладка начнется Вами с команды M.

На практике число точек остановки может быть несколько (либо не быть вовсе), а место их включения и способ чередования с режимом F11 вы можете выбирать сами. Основное, что нужно помнить – ПО будет выполнено до точки остановки и быстро, остальную часть кода можно проверять, комбинируя «BreakPoint-ы» и F11.

5. **В случае обнаружения ошибок** алгоритма или несоответствия функционирования ПО условиям задания, **остановите моделирование кнопкой «Stop».** Не закрывая среды Proteus, перейдите к проекту в среде MPLAB, **внесите** требуемые **изменения в asm-файл и повторно перекомпилируйте его**, проведенные Вами изменения будут автоматически учтены средой Proteus. После этого, вновь проведите отладку в среде Proteus. Только при полном соответствии функционирования Вашего ПО и условий задания переходите к следующему пункту.

6. Проведите окончательную проверку функционирования Вашего ПО в режиме **анимации**. Для этого нажмите кнопку «Play» и проверьте реакцию контроллера на все возможные сигналы с датчиков в режиме непрерывного выполнения ПО. Для контроля состояний внутренних регистров микроконтроллера в

этом режиме также можно пользоваться окном «Watch Window» - оно не исчезает при моделировании. Убедившись в правильности функционирования ПО, предъявите результаты работы преподавателю.

7. Сохраните проект в средах MPLAB, Proteus и **убедитесь в наличии файла с расширением hex**, являющегося файлом «прошивки» микроконтроллера PIC 16F877. Этот файл подлежит заливке в РПЗУ контроллера посредством утилиты STest (см. Гл.3 МУЛР).