МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Омский государственный технический университет»

В. Ф. Нестерук

МОДЕЛИРОВАНИЕ ПЕРИФЕРИЙНОГО ОБОРУДОВАНИЯ В ИНТЕГРИРОВАННОЙ СРЕДЕ РАЗРАБОТКИ PROTEUS

Учебное текстовое электронное издание
локального распространения

Омск
Издательство ОмГТУ
2014

Сведения об издании: 1, 2

ISBN 978-5-8149-1743-0

УДК 004.382.7(075)
ББК 32.973.26-04я73
Н56

Рецензенты:

В. Д. Червенчук, канд. техн. наук, доцент СибАДИ;

А. П. Загородников, канд. техн. наук, директор ООО «АИСистемс»

Нестерук, В. Ф.

Н56 Моделирование периферийного оборудования в интегрированной среде разработки Proteus : учеб. пособие / В. Ф. Нестерук ; Минобрнауки России, ОмГТУ. – Омск : Изд-во ОмГТУ, 2014.

ISBN 978-5-8149-1743-0

Рассматриваются принципы организации и применения интегрированной среды разработки Proteus при отладке программ для встраиваемых микроконтроллеров и моделировании функционирования подключаемого периферийного оборудования.

Предназначено для использования в учебном процессе по направлению «Информатика и вычислительная техника».

УДК 004.382.7(075)
ББК 32.973.26-04я73

Рекомендовано редакционно-издательским советом
Омского государственного технического университета

ISBN 978-5-8149-1743-0

1 электронный оптический диск

Оригинал-макет издания выполнен в Microsoft Office Word 2007
с использованием возможностей Adobe Acrobat X.

Минимальные системные требования:

- процессор Intel Pentium 1,3 ГГц и выше;
- оперативная память 256 Мб;
- свободное место на жестком диске 260 Мб;
- операционная система Microsoft Windows XP/Vista/7;
- разрешение экрана 1024×576 и выше;
- акустическая система не требуется;
- дополнительные программные средства Adobe Acrobat Reader v 5.0 и выше.

№ госрегистрации 0321400990

Редактор *К. В. Муковоз*
Компьютерная верстка *А. Н. Кошелапова*

Сводный темплан 2014 г.
Подписано к использованию 15.05.14.
Объем 4,59 Мб.

Издательство ОмГТУ.
644050, г. Омск, пр. Мира, 11; т. 23-02-12
Эл. почта: info@omgtu.ru

ВВЕДЕНИЕ

Отличием среды ISIS Proteus от широко известных интегрированных сред разработки AVRStudio, MPLAB, E8031 и других является возможность моделирования периферийной обвязки центрального микроконтроллера на уровне принципиальных схем, измерения параметров сигналов во внешних цепях, анимации состояния периферийных устройств. Для моделирования как ядра микропроцессорной системы, так и периферийной обвязки привлекаются библиотеки с широкой номенклатурой моделей цифровых и аналоговых изделий электронной техники ведущих фирм-производителей.

1. РАБОЧЕЕ ОКНО ISIS PROTEUS

Рабочее окно среды ISIS (рис. 1) содержит масштабируемое поле для создания моделируемой схемы проекта, в рамках которого отображается текущий лист чертежа проекта в заданном стандарте через меню **Система → Размеры листа** (A1, A2 и пр.).

Над масштабируемым полем размещаются строка меню и строка инструментов. Слева по вертикали выделяется три поля: поле выбора компонентов схемы, поле перемещения, поле списка выбранных устройств, в верхней части которого размещается окно отображения текущего выбранного компонента или проектируемая схема в уменьшенном масштабе.

Слева в нижней строке рабочего окна показаны кнопки управления анимацией, к ним примыкают поле сообщений, поле комментариев и поле указателя смещения курсора в ортогональной системе координат относительно центра листа чертежа проекта, помеченного крестообразным маркером.

1.1. Поле выбора компонентов

В поле размещаются иконки, за каждой из которых закреплено соответствующее множество компонентов определённого функционального типа. Активизация поля осуществляется щелчком левой кнопки мыши по верхней иконке с символом стрелки курсора. После чего щелчком по любой иконке активизируется соответствующий раздел библиотеки моделей.

Активный раздел библиотеки отображается в выпадающем окне **Pick Devices** (рис. 2) с помощью щелчка по клавише **P** в шапке поля списка выбранных устройств. Ниже приведен перечень закреплённых типов компонентов за иконками поля выбора в порядке просмотра сверху вниз после иконки курсора:

- электронные, электрические, электромеханические компоненты;
- точка соединения;
- метка соединения;
- текстовый фрагмент;
- шина или шинная сборка;
- щуп напряжения;
- щуп тока;
- виртуальные инструменты;
- 2D-графика (линия);
- 2D-графика (прямоугольник);
- 2D-графика (круг);
- субсхема;
- терминальные подключения;
- типы выводов;
- графики диаграмм;
- магнитофон;
- генератор;
- 2D-графика (дуга);
- 2D-графика (полилинейный контур);
- 2D-графика (текст);
- 2D-графика (символ);
- маркер.

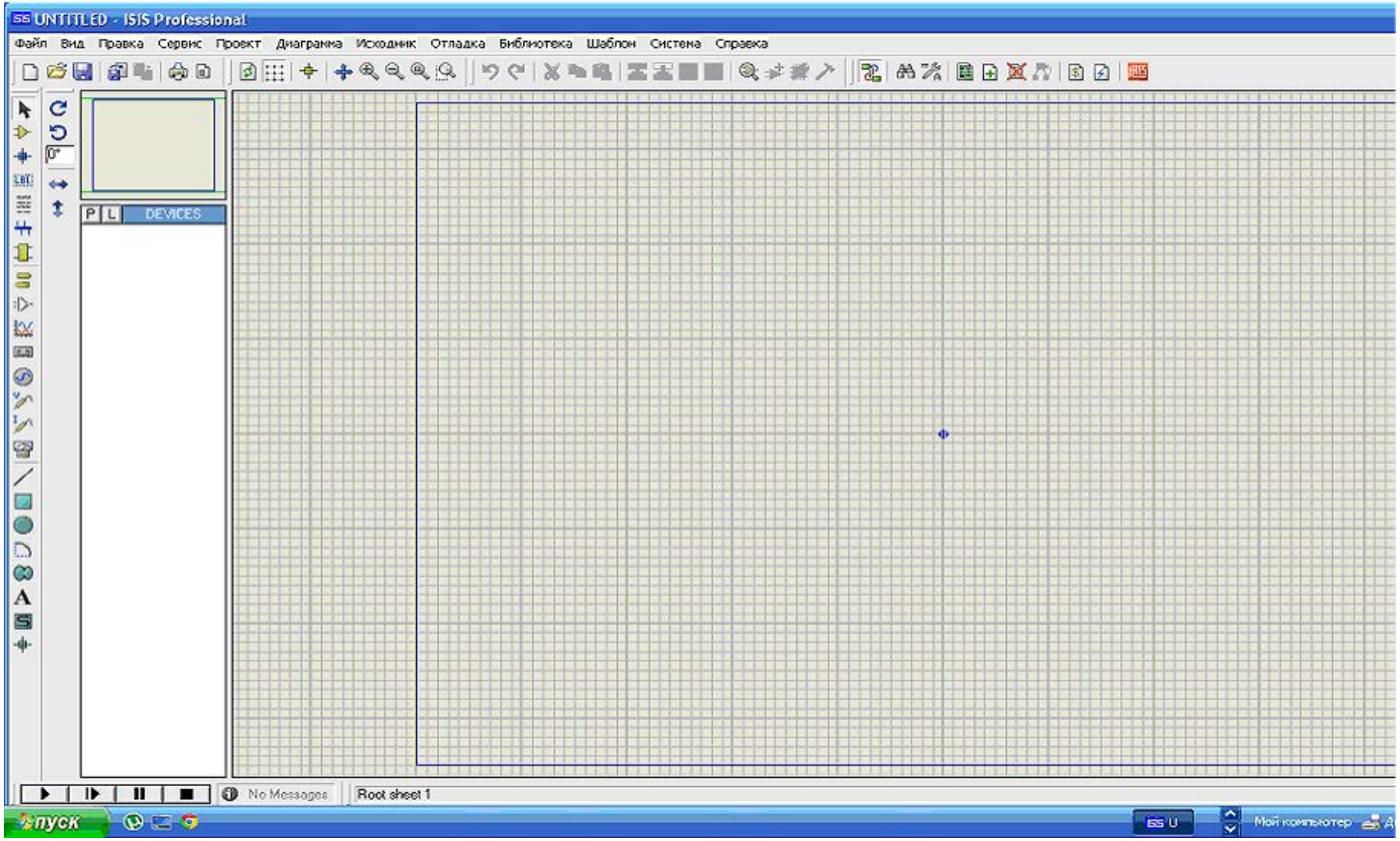

Рис. 1. Рабочее окно

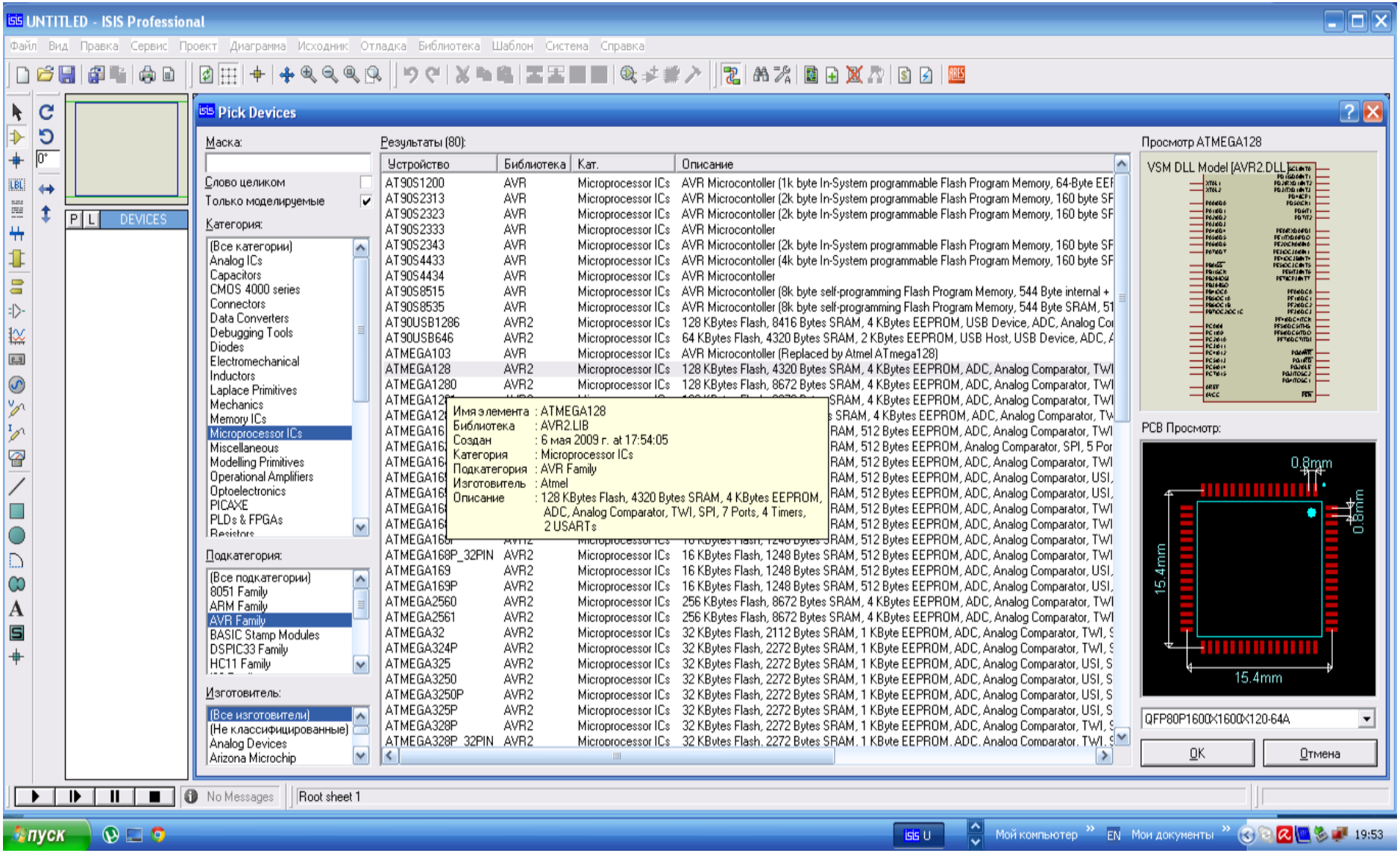

Рис. 2. Вызов библиотеки

При создании моделируемой схемы основные функциональные компоненты выбираются в выпадающем окне **Pick Devices** (рис. 2). Окно содержит центральное поле перечня предлагаемых моделей устройств с линейкой скроллинга, левую вертикальную группу полей: поле маски поиска и три поля с линейками скроллинга – **Категория**, **Подкатегория** и **Изготовитель**, а также правую вертикальную группу полей: просмотра условного графического изображения выбранного устройства, просмотра конструктивного исполнения устройства, выбора типа корпуса устройства – и две кнопки выбора **ОК** и **Отмена**.

В качестве примера на рис. 2 выбран микроконтроллер ATmega 128 фирмы Atmel.

Щелчком мыши по кнопке **ОК** включается данное устройство в поле списка выбранных устройств основного окна (рис. 3) и закрывается выпадающее окно, а щелчком по кнопке **Отмена** закрывается выпадающее окно без изменения списка выбранных устройств.

Имя выбранного устройства, в данном случае **ATmega 128**, отобразилось в колонке **DEVICES** поля списка выбранных устройств. Для размещения условного графического изображения этого устройства в масштабируемом поле рабочего окна необходимо:

– отметить щелчком левой кнопкой мыши в колонке DEVICES требуемое устройство;

– щелчком левой кнопкой мыши в пределах масштабируемого поля вызвать перемещаемый контур устройства;

– путём перемещения курсора задать координаты размещения устройства;

– повторным щелчком зафиксировать положение устройства.

Щелчком по клавише **L** в шапке поля списка выбранных устройств вызывается выпадающее окно менеджера библиотеки **Devices Libraries Manager**. В этом окне возможны манипуляции с содержимым библиотек в соответствии с функциональным назначением кнопок.

В поле перемещения рабочего окна размещаются иконки, управляющие поворотом изображения отмеченного компонента схемы по часовой стрелке или против часовой стрелки на угол, кратный 90°, либо переворотом в горизонтальной или вертикальной плоскости. Кратность поворота определяется количеством щелчков по соответствующей иконке. При этом в окне между иконками поворота и переворота отображается установленный угол поворота.

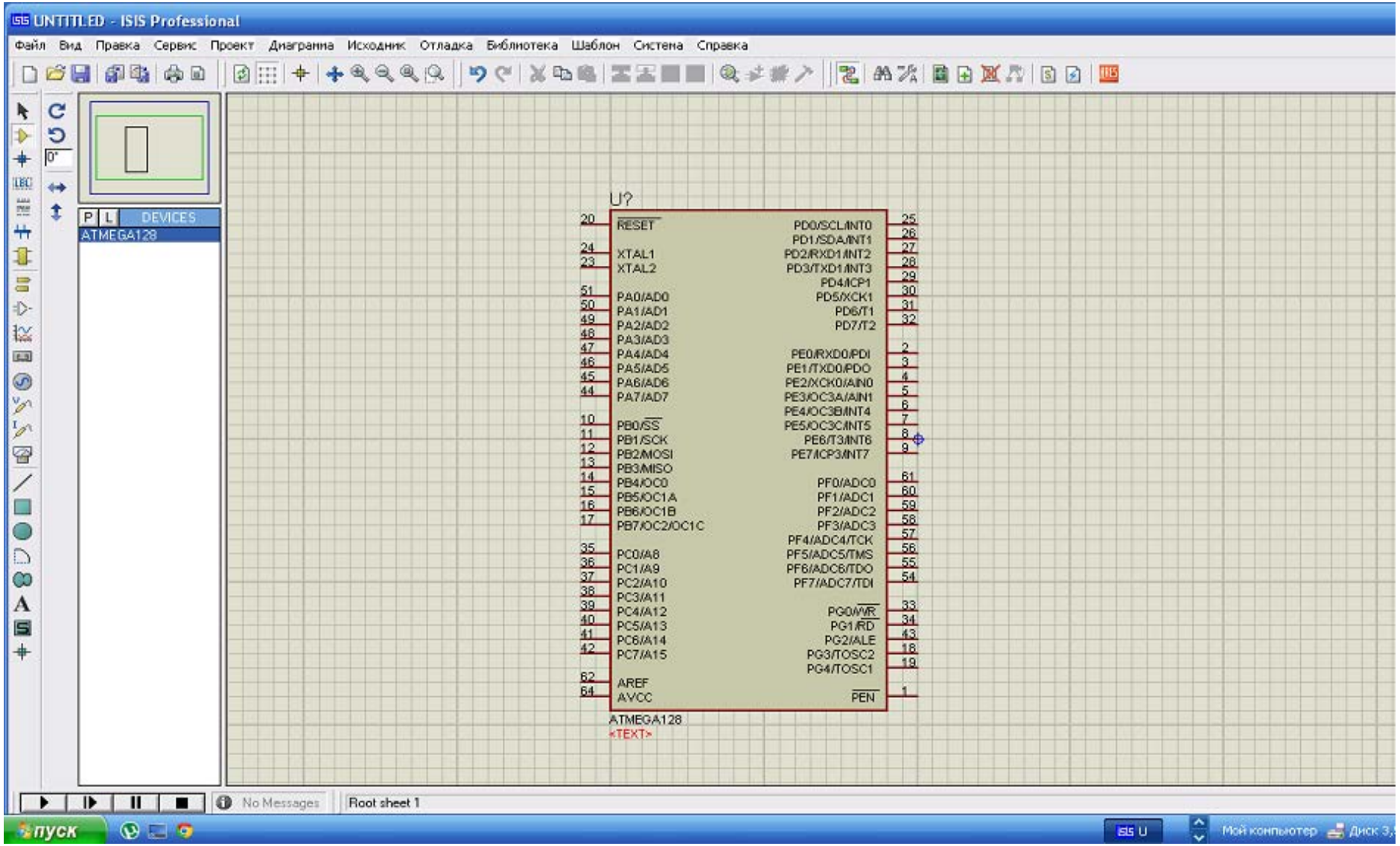

Рис. 3. Размещение устройства

1.2. Типы компонентов

Тип компонентов «Электронные, электрические, электромеханические компоненты» содержит модели активных и пассивных элементов, узлов и устройств от уровня дискретных элементов до уровня больших и сверхбольших интегральных схем, применяемых для построения дискретных и аналоговых электронных, электрических и электромеханических систем:

- аналоговые микросхемы (Analog ICs);
- КМОП 4000 серии (CMOS 4000 series);
- преобразователи (Data Converters);
- средства отладки (Debugging Tools);
- электромеханика (Electromechanical);
- модели примитивов Лапласа (Laplace Primitives);
- микросхемы памяти (Memory ICs);
- микропроцессоры (Microprocessor Ics);
- модели примитивов (Modelling Primitives);
- операционные усилители (Operational Amplifiers);
- оптоэлектроника (Optoelectronics);
- микросхемы PICAXE (PICAXE);
- программируемая логика (PLDs & FPGAs);
- симуляторы примитивов (Simulator Primitives);
- динамики и излучатели (Speakers & Sounders);
- переключатели и реле (Switches & Relays);
- электронные переключатели (Switching Devices);
- электровакуумные приборы (Thermionic Valves);
- датчики-преобразователи (Transducers);
- ТТЛ микросхемы 74 серии (TTL 74 series);
- ТТЛ микросхемы 74ALS серии (TTL 74ALS series);
- ТТЛ микросхемы 74AS серии (TTL 74AS series);
- ТТЛ микросхемы 74F серии (TTL 74F series);
- ТТЛ микросхемы 74HC серии (TTL 74HC series);
- ТТЛ микросхемы 74HCT серии (TTL 74HCT series);
- ТТЛ микросхемы 74LS серии (TTL 74LS series);
- ТТЛ микросхемы 74S серии (TTL 74S series);
- конденсаторы (Capacitors);
- разъёмы (Connectors);
- диоды (Diodes);
- дроссели (Inductors);
- механизмы (Mechanics);
- разное (Miscellaneous);
- резисторы (Resistors);
- транзисторы (Transistors).

Тип компонентов «Точка соединения» позволяет организовать в проектируемой схеме соединение пересекающихся проводников. Тип компонентов «Метка соединения» даёт возможность присвоить имя требуемому соединению схемы. Тип компонентов «Текстовый фрагмент» предназначен для размещения надписей в поле чертежа в соответствии с используемым в проекте стандартом

шрифта. Тип компонентов «Шина или шинная сборка» позволяет с помощью одноразрядной шины либо через шинные сборки отображать на схеме требуемые электрические соединения между элементами схемы. Тип компонентов «Субсхема» дает дополнительные назначения элементам моделируемого устройства. Тип компонентов «Терминальные подключения» даёт возможность разместить на чертеже входные или выходные точки подключения следующих типов:

– клемма общего назначения (DEFAULT);
– входная шина (INPUT);
– выходная шина (OUTPUT);
– двунаправленная шина (BIDIR);
– шина питания (POWER);
– общая шина (GROUND);
– шинная сборка (BUS).

Тип компонентов «Типы выводов» позволяет определить свойства выводов элементов схемы:

– по умолчанию (DEFAULT);
– инверсный (INVERT);
– шинная сборка (BUS);
– инверсный синхровход (NEGCLK);
– короткий (SHORT);
– положительный синхровход (POSCLK).

Тип компонентов «Графики диаграмм» предназначен для графического отображения диаграмм следующих видов:

– аналоговая (ANALOGUE);
– цифровая (DIGITAL);
– смешанная (MIXED);
– частотная (FREQUENCY);
– передача (TRANSFER);
– шум (NOISE);
– искажение (DISTORTION);
– Фурье (FOURIER);
– звуковая (AUDIO);
– интерактивная (INTERACTIVE);
– соответствия (CONFORMANCE);
– цифровая развёртка (DC SWEEP);
– аналоговая развёртка (AC SWEEP).

Тип компонентов «Магнитофон…» позволяет моделировать функции записи и воспроизведения на магнитной ленте.

Тип компонентов «Генератор» позволяет формировать входные воздействия следующих видов:

– цифровое (DC);
– синусоидальное (SINE);
– импульсное (PULSE);
– экспоненциальное (EXP);
– гармоническое с синусоидальной частотной модуляцией (SFFM);
– кусочно-линейное (PWLIN);
– прямоугольное импульсное (DPULSE);
– импульсная синхронизация (DCLOCK);
– символьная таблица (SCRIPTABLE);
– выборка из файла (FILE);
– звуковое (AUDIO);
– постоянное цифровое (DSTATE);
– релейное (DEDGE);
– цифровой шаблон (DPATTERN).

Тип компонентов «Щуп напряжения» используется для отображения в текущем времени уровня напряжения в точке подключения щупа. Тип компонентов «Щуп тока» позволяет индицировать текущее значение тока в цепи в точке подключения щупа. Тип компонентов «Виртуальные инструменты» даёт возможность регистрации сигналов в требуемой точке схемы либо генерации сигналов на внешних входах моделируемой схемы с помощью следующих средств:

- осциллограф (OSCILLOSCOPE);
- логический анализатор (LOGIC ANALYSER);
- счётчик времени (COUNTER TIMER);
- виртуальный терминал (VIRTUAL TERMINAL);
- отладчик SPI интерфейса (SPI DEBUGGER);
- отладчик I2C интерфейса (I2C DEBUGGER);
- генератор сигналов (SIGNAL GENERATOR);
- генератор кодов (PATTERN GENERATOR);
- цифровой вольтметр (DC VOLTMETER);
- цифровой амперметр (DC AMMETER);
- аналоговый вольтметр (AC VOLTMETER);
- аналоговый амперметр (AC AMMETER).

Типы компонентов «2D-графика» используются для создания двумерных графических изображений в плоскости чертежа. Тип компонентов «Маркер» позволяет размещать в поле чертежа следующие метки:

- источник (ORIGIN);
- узел (NODE);
- узел шины (BUSNODE);
- метка (LABEL);
- имя вывода (PINNAME);
- номер вывода (PINNUM);
- увеличение (INCREMENT);
- уменьшение (DECREMENT);
- переключение (TOGGLE).

Выбирая требуемые устройства из библиотек для соответствующих типов компонентов, разработчик может создать в масштабируемом поле схему проектируемой системы, разработать и отладить программное обеспечение для программируемых цифровых устройств этой системы, отследить в процессе симуляции уровни токов и напряжений в шинах, получить временные графики сигналов, отследить форму аналоговых сигналов и пр.

В учебном пособии на конкретных примерах будет показано моделирование процессов ввода-вывода информации в системах, выполненных на базе микроконтроллеров фирм Intel, Microchip Technology и Atmel.

2. МОДЕЛИРОВАНИЕ СИСТЕМ НА БАЗЕ МИКРОКОНТРОЛЛЕРОВ ФИРМЫ ATMEL

В рассматриваемых моделях будут использоваться байтовые микроконтроллеры семейства ATmega, занимающие существенную нишу при разработке встраиваемых систем управления. В качестве периферийных средств выберем такие типовые устройства ввода-вывода, как матричная клавиатура и блок семисегментных индикаторов.

2.1. Моделирование опроса матричной клавиатуры в режиме прерывания

Возьмём в качестве базового устройства проектируемой схемы микроконтроллер ATmega 128 фирмы Atmel (см. рис. 3). К выводам порта PE.0–3 через защитные диоды D1–D4 подключим входные шины, а к выводам PE.4–7 – выходные шины матричной клавиатуры 4 × 4. В качестве клавиш клавиатуры будем использовать джамперы, что удобно при пошаговой отладке, которые находим в окне **Pick Devices** в категории **Switches & Relays** в подкатегории **Switches**, а диоды типа S1A – в категории **Diodes** в подкатегории **Rectifiers.**

Подключение компонентов схемы осуществляется через шинную сборку, которая выбирается через иконку **Шина или шинная сборка** поля выбора. После нажатия на иконку перемещаем курсор в начальную точку размещения шинной сборки в масштабируемом поле, делаем однократный щелчок левой кнопкой мыши, перемещаем курсор до конечной точки, в которой выполняем двойной щелчок. Траектория передвижения курсора может изгибаться под прямыми углами. При необходимости точка изгиба фиксируется однократным щелчком.

Подключение выводов устройств к шинной сборке производится одноразрядными шинами, которые начинаются со щелчка левой кнопкой мыши на требуемом выводе и заканчиваются щелчком в нужной точке входа шины в шинную сборку. Для простановки меток шин в шинной сборке можно использовать иконку LBL (метка связи) в поле выбора, после нажатия на которую требуется переместить курсор на нужную шину, выполнить щелчок левой кнопкой мыши и в выпадающем окне ввести текст метки.

В схему устройства (рис. 4) введены также резисторы R1–R4 для задания высоких потенциалов на выходных шинах клавиатуры при отсутствии нажатия и два конъюнктора DD2 и DD3 для формирования запроса прерывания на входе INT0 (вывод PD.0) при нажатии. Циклическое сканирование входных шин клавиатуры унитарным нулём со стороны микроконтроллера будет вызывать его прерывание по входу INT0 при любом нажатии.

Для подключения резисторов R1–R4 к шине питания в поле выбора компонентов используем иконку TERMINALS, а в поле списка выбранных устройств отмечаем строку POWER. Далее по щелчку левой кнопки мыши в требуемом месте масштабируемого поля размещаем стрелку подключения и проставляем метку +E. Поле выбора компонентов (рис. 5) можно вызвать также щелчком правой кнопкой мыши в пределах масштабируемого поля через выпадающее меню **Разместить.**

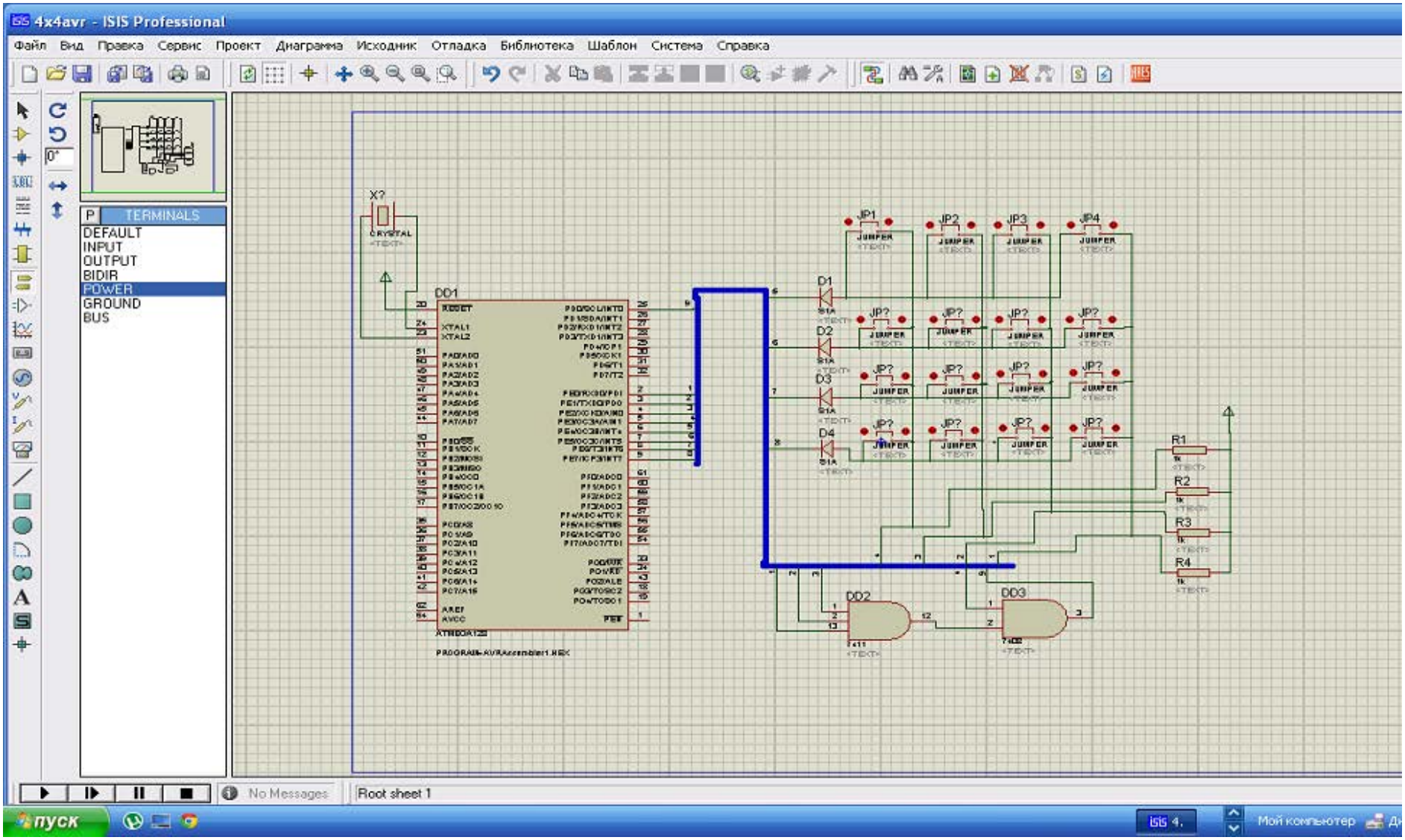

Рис. 4. Подключение компонентов

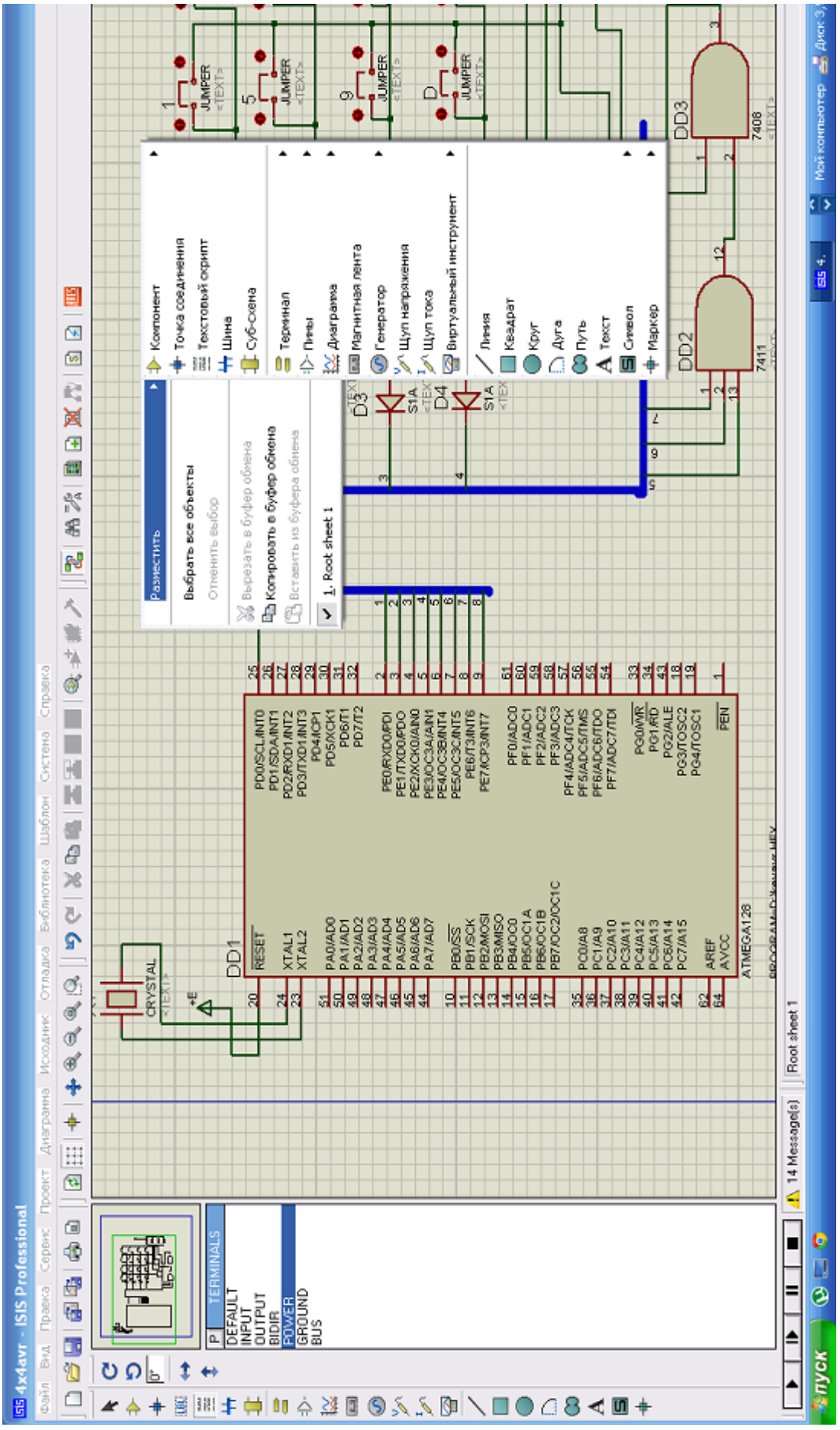

Рис. 5. Выбор компонентов

Однотипные компоненты для правильной компиляции при отладке должны иметь отличные друг от друга имена, так как при начальном размещении в масштабируемом поле они получают одинаковые имена. Так, например, все джамперы первоначально имели метки JP?. В рассматриваемом примере (см. рис. 4) их метки были изменены на номера клавиш от 0 до F. Изменение метки осуществляется щелчком правой кнопкой мыши на метке через выбор строки выпадающего меню **Изменить метку** (рис. 6). Кроме этого, в список выпадающего меню входят команды **Удалить метку**, **Перетащить объект**, **Правка свойств, Удалить объект**, **Повернуть по часовой стрелке**, **Повернуть против часовой стрелки**, **Повернуть на 180 градусов**, **Зеркально по X**, **Зеркально по Y**, **Разложить**, **Переход на дочерний лист**, **Показать справку модели**, **Показать Datasheet**, **Показать в проводнике проекта**, **Показать распределение корпуса**, **Информация оперативной точки**, **Настройка диагностики**, **Создать устройство**, **Корпус**, **Инкремент**, **Декремент**, **Переключить**.

2.2. Разработка алгоритма опроса клавиатуры

Возможный вариант алгоритма взаимодействия микроконтроллера с матричной клавиатурой размером 4 × 4 представлен на рис. 7. Сканирование клавиатуры будет осуществляться по горизонтальным шинам матрицы, подключённым к выходным шинам порта PE.4–7 (см. рис. 4). Период модификации кода сканирования будет определяться переполнением аппаратного таймера микроконтроллера. Выходные сигналы клавиатуры снимаются с вертикальных шин матрицы, соединённых с входными шинами порта PE.0–3.

Запрос прерывания при нажатии на любую клавишу формируется по отрицательному фронту выходного сигнала конъюнктора DD3, поступающему на входную шину порта PD.0, настроенную на альтернативную функцию INT0.

В соответствии с рассмотренными подключениями в вершине 1 алгоритма осуществляется инициализация и запуск аппаратного таймера на формирование периода сканирования, шин порта PE.4–7 на вывод данных, шин портов PD.0 и PE.0–3 на ввод данных, встроенного контроллера прерываний на запросы прерывания от таймеров и внешнего запроса INT0. Также необходимо определить начальное состояние регистра сканирования, регистра счётчика позиции нуля в регистре сканирования, базы буфера данных, регистра счётчика смещения в буфере данных, регистра кода команды и флагового регистра режима разделения времени. Таймер выдержки времени дребезга контактов клавиатуры и таймер выдержки времени реакции оператора будут инициализироваться по ходу реализации алгоритма.

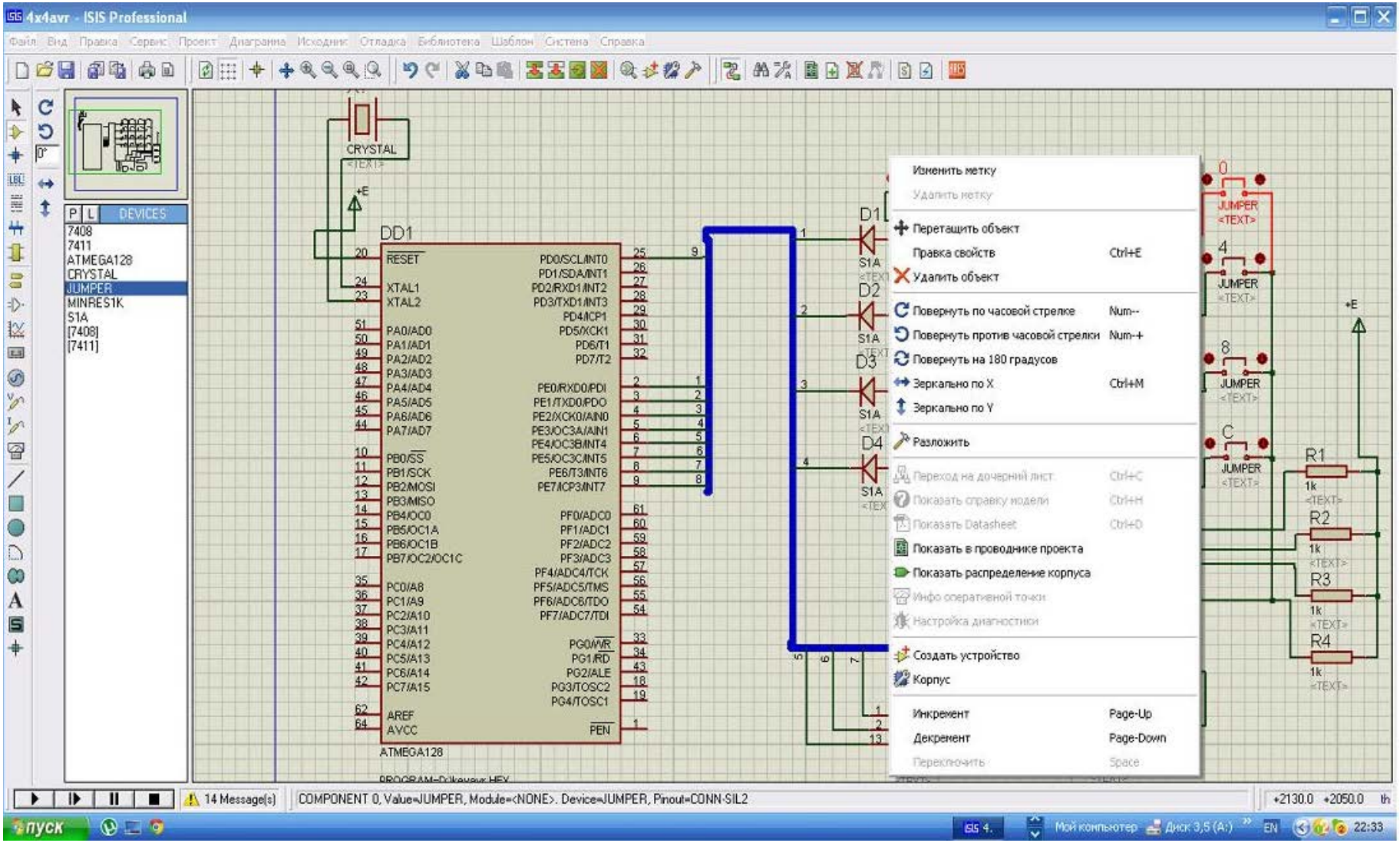

Рис. 6. Меню редактирования

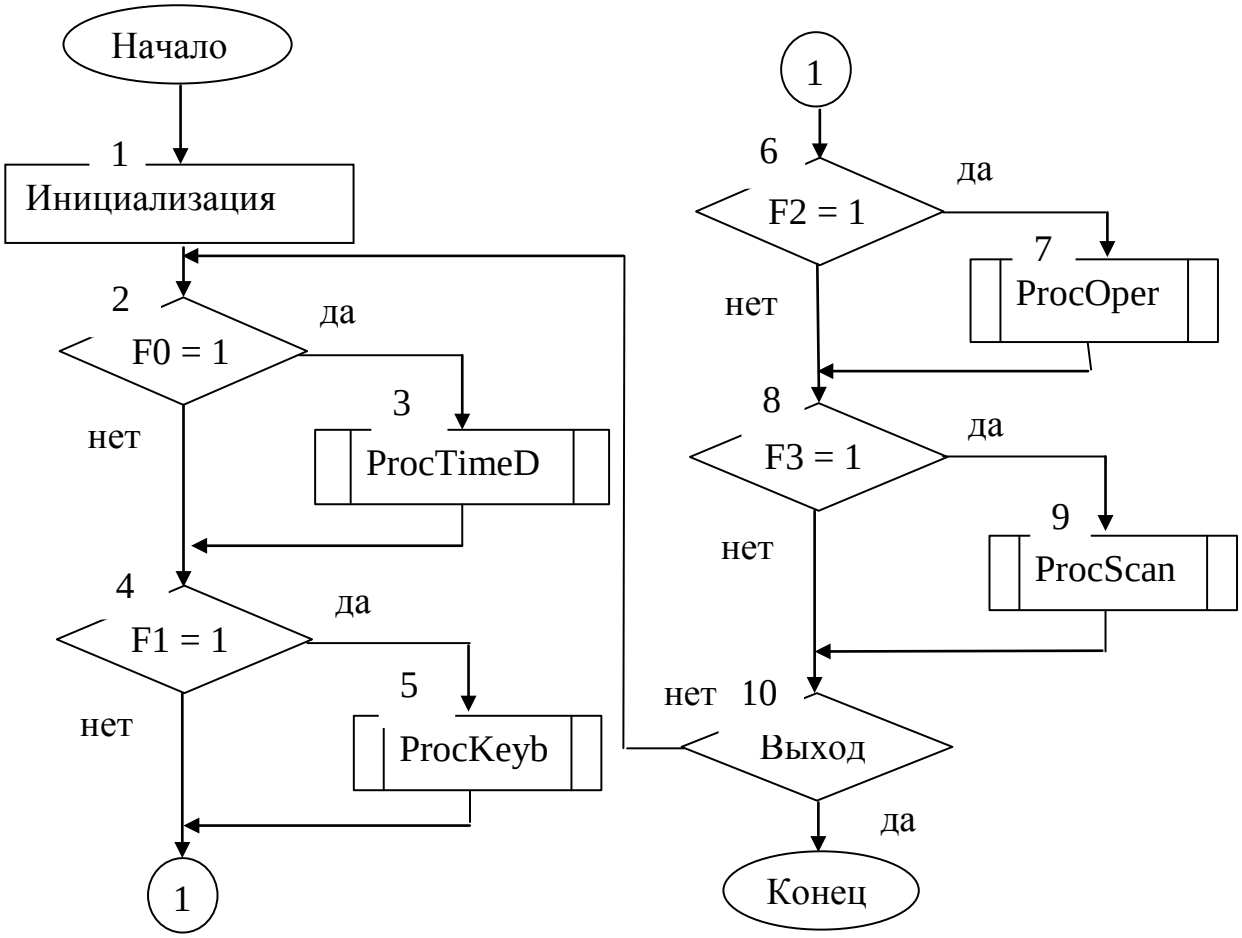

Рис. 7. Фоновый цикл

В вершинах 2, 4, 6, 8 алгоритма выполняется анализ соответствующих флагов F0–F3, по которым разделяется время работы системы. Установка флагов осуществляется по тем либо иным событиям в системе. В данном случае установка F0 производится по прерыванию INT0, F1 – по прерыванию таймера выдержки времени дребезга контактов клавиатуры, F2 – по прерыванию таймера выдержки времени реакции оператора, F3 – по прерыванию таймера формирования периода сканирования.

Исходное значение флагов устанавливается нулевое, кроме F6 = 1, вследствие чего система будет находиться в фоновом цикле, пока не установится единичное значение флага F3 или не сформируется признак завершения работы (вершина 10). Единичное значение какого-либо из флагов F0–F3 вызывает программный переход на соответствующую процедуру, в ходе выполнения которой флаг обнуляется. В данном алгоритме – это процедуры ProcTimeD, ProcKeyb, ProcOper, ProcScan. Флаг F6 в фоновом цикле не анализируется, его назначение – разрешать (F6 = 1) или запрещать (F6 = 0) реакцию на нажатие клавиши внутри процедуры ProcScan.

Процедуры обслуживания прерываний, в ходе которых устанавливаются флаги F0–F3, приведены на рис. 8, где используются следующие обозначения: РС – регистр сканирования; БС – буфер кода сканирования; РСП – регистр счётчика позиции; БСП – буфер кода счётчика позиции. Буферы предназначены для сохранения текущего состояния РС и РСП на момент нажатия клавиши.

На рис. 9 представлены граф-схемы процедур ProcTimeD и ProcOper, вызываемых по флагам F0 и F2, устанавливаемым в прерывающих процедурах INT0 и INTTS. В первой процедуре осуществляется загрузка и запуск таймера выдержки времени дребезга контактов, а во второй – разрешение дальнейших прерываний по нажатию клавиши.

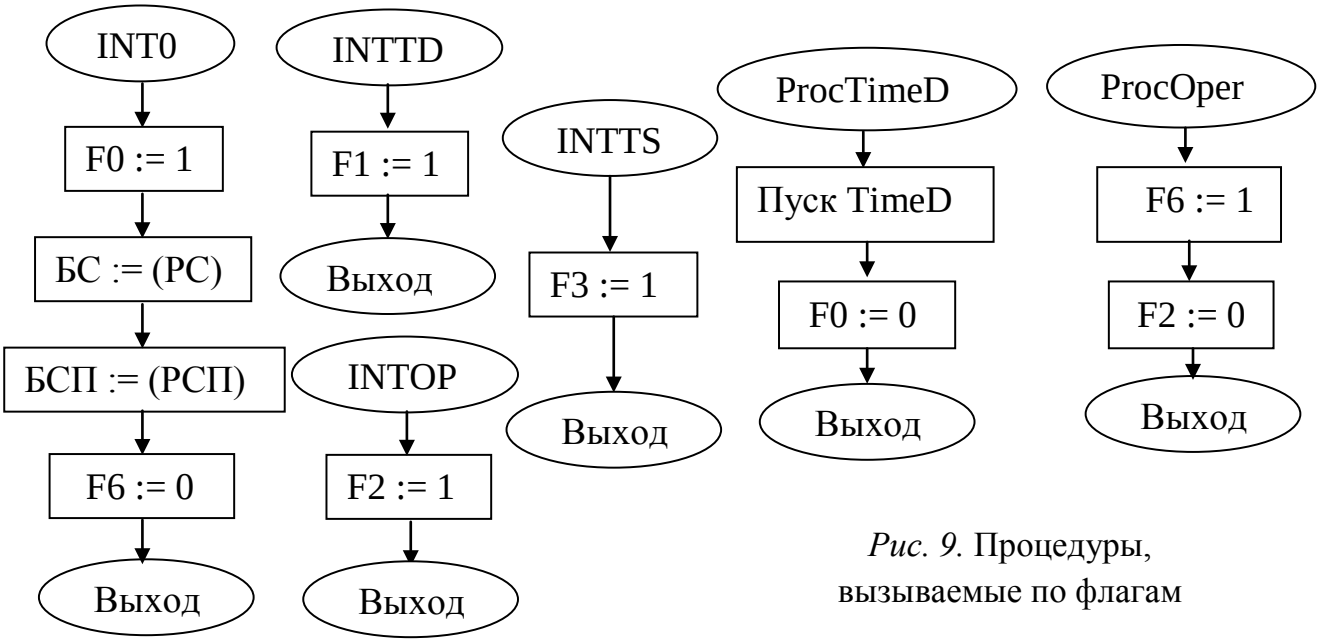

Рис. 8. Процедуры прерывания

Рис. 9. Процедуры, вызываемые по флагам

На рис. 10 приведена граф-схема процедуры ProcKeyb, которая вызывается по флагу F1. Этот флаг устанавливается в прерывающей процедуре INTTD, вызываемой по переполнению таймера выдержки времени дребезга контактов. В граф-схеме дополнительно к ранее рассмотренным используются следующие мнемонические обозначения: СВП – счётчик позиции нуля в ответном коде клавиатуры; АСС – аккумулятор процессорного ядра микроконтроллера; СЛП – мнемокод операции правого логического сдвига; С – выдвигаемый бит; РКК – регистр кода клавиши; ББД – база буфера данных; СБ – текущее смещение при обращении к буферу данных; РК – регистр кода команды; TimeO – таймер выдержки времени реакции оператора.

Вводимые с клавиатуры данные разделяются на числовые (от 0 до 9) и управляющие (от Ah до Fh). Последние классифицируются как коды команд и в отличие от кодов чисел, размещаемых в буфере данных, заносятся в РК. Глубина буфера равна модулю пересчёта СБ и выбирается разработчиком (в примере модуль равен четырём). По результатам ввода чисел от 0 до 9 или команд выставляются соответствующие оповещающие флаги F4 и F5 для других процедур системы.

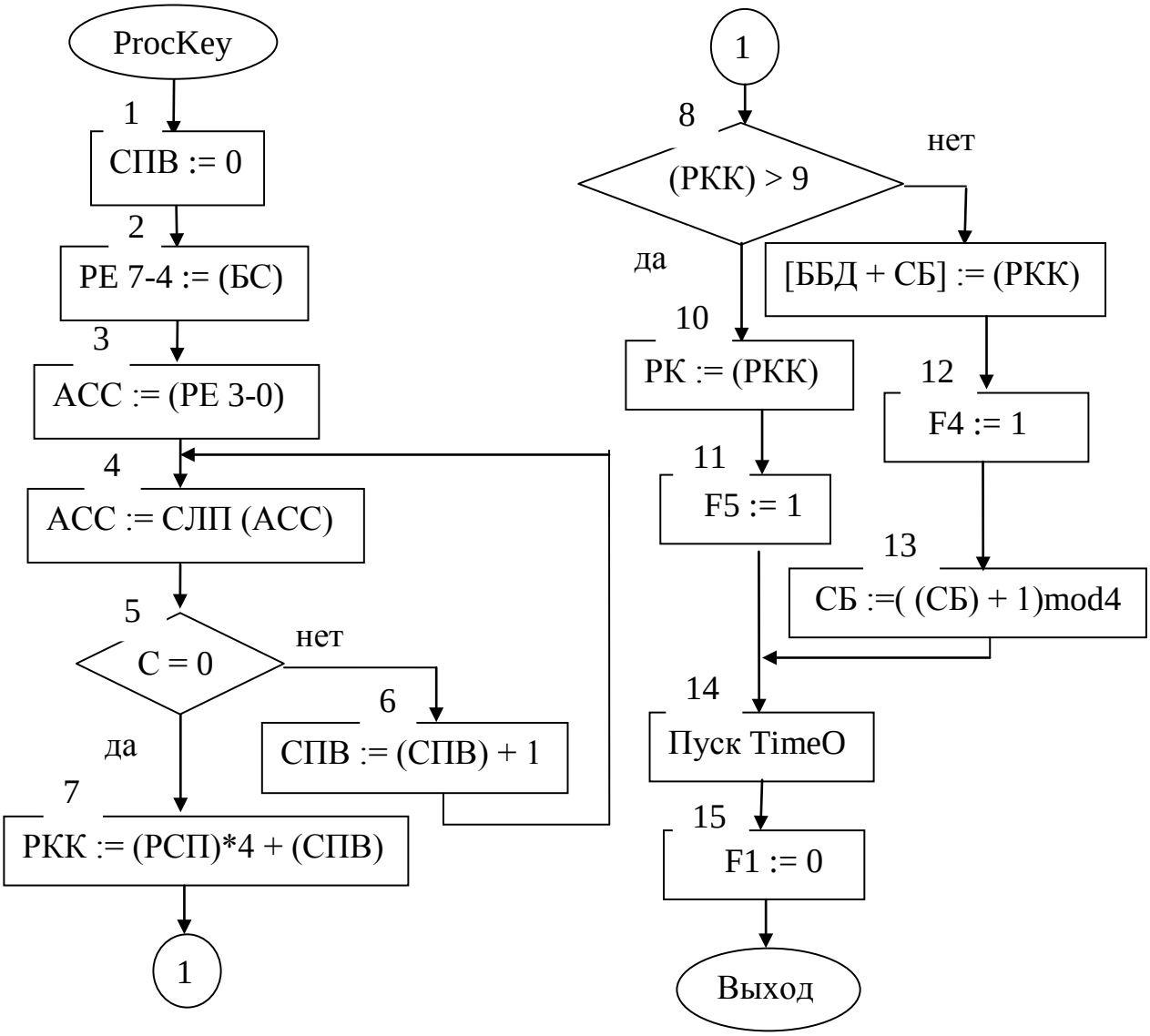

Рис. 10. Опрос клавиатуры

Граф-схема процедуры ProcScan представлена на рис. 11. В процедуре, вызываемой по флагу F3, выполняются модификации содержимого регистров РС и РСП. Содержимое первого из них сдвигается циклически влево, а содержимое РСП инкрементируется на +1 и округляется по mod4 при каждом сдвиге в РС. Реакция на прерывание по INT0 разрешается в узком временном интервале, когда значения РС и РСП неизменны. Дополнительно флаг F6 блокирует повторные прерывания до окончания выдержки времени реакции оператора.

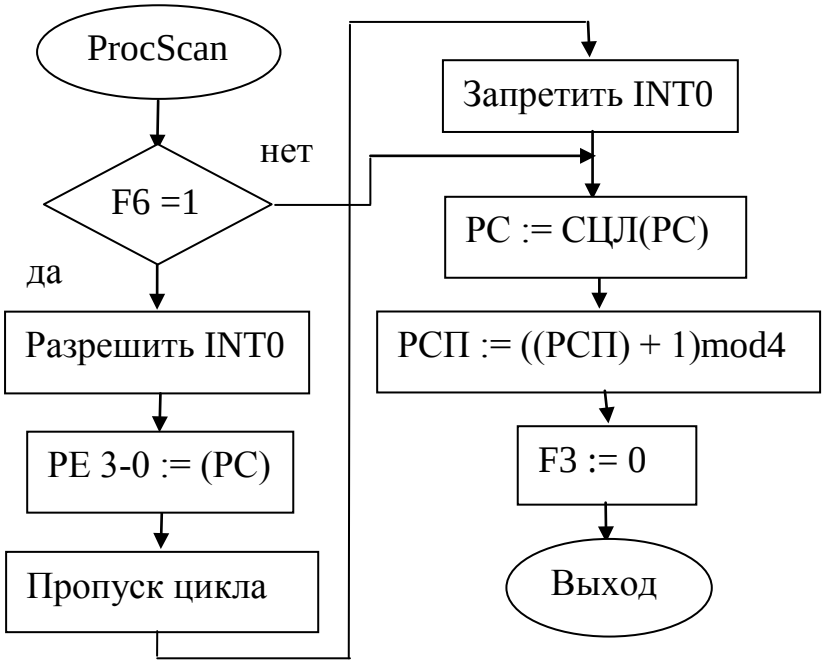

Рис. 11. Сканирование

2.3. Программная реализация алгоритма

Ниже представлена программа, составленная на языке ассемблера в системе команд микроконтроллера ATmega 128. Для компилирования программ микроконтроллеров ATmega в настройках среды ISIS следует указать ассемблер AVRASM2.

```
/* опрос матричной клавиатуры 4×4 в режиме прерывания
 * AVRAssembler1.asm */
.include "m128def.inc"; определение символьных переменных и констант
.def temp = r16; регистр временного хранения
.def scan = r18; регистр кода сканирования
.def pos = r19; счётчик позиции нуля в коде сканирования
.def flags = r20; регистр флагов
.def off = r21; смещение буфера клавиатуры
.def bufs = r22; буферный регистр кода сканирования
.def bufp = r23; буферный регистр позиции нуля в коде сканирования
.def vpos = r24; счётчик позиции нуля в ответном коде клавиатуры
.def rkl = r25; регистр кода клавиши
.def rkk = r17; регистр кода команды
.equ base = 0x60; база буфера клавиатуры
.equ scan0 = 0xee; начальное значение кода сканирования
.cseg
.org 0x00
jmp     start; обход зоны векторов  прерываний
.org 0x02; вектор внешнего прерывания int0
```

```
jmp     inte0; переход по внешнему прерыванию int0
.org 0x12;  вектор прерывания таймера 2 (таймер TimeO)
jmp     tim2; переход по совпадению таймера 2
.org 0x18;  вектор прерывания таймера 1 (таймер  периода сканирования)
jmp     tim1; переход по совпадению таймера 1
.org 0x1e;  вектор прерывания таймера 0 (таймер TimeD)
jmp     tim0; переход по совпадению таймера 0
start:    ; инициализация аппаратной среды
ldi flags, 0x40; установка флага блокировки F6 (разрешение INT0)
ldi scan, scan0; инициализация регистра сканирования
ldi pos, 0x00; инициализация счётчика позиции горизонтали
ldi xl,base; инициализация базы буфера клавиатуры
ldi xh, 0x00
ldi off, 0x00; инициализация смещения буфера клавиатуры
ldi temp, low(ramend); инициализация указателя стека
out spl, temp
ldi temp, high(ramend)
out sph, temp
ldi temp , 0xf0; настройка регистра направления порта PE
out ddre, temp
ldi temp , 0xfe; настройка регистра направления порта PD
out ddrd, temp
ldi temp, 0x92; разрешение локальных прерываний по
out timsk, temp; совпадению в таймерах tcnt2, 0 и tcnt1a
ldi temp, 0x7f; загрузка в регистры сравнения ocr0, 2 и ocr1ah,l
out ocr0, temp; констант сравнения
ldi temp, 0xff;
out ocr2, temp;
ldi temp, 0x2f;
out ocr1al, temp;
ldi temp, 0x0;
out ocr1ah, temp
ldi temp, 0x08; настройка управляющих регистров
out tccr0, temp; таймеров tcnt2, 0 на сброс по совпадению
out tccr2, temp; с константами в ocr2, 0
ldi temp, 0x09; настройка управляющего регистра tcnt1 на пуск и
out tccr1b, temp; сброс по совпадению с константой в ocr1a
sei; глобальное разрешение прерываний
; цикл фоновой программы
```

```
fon:     sbrc flags, 0; анализ флага F0
jmp ProcTimeD; переход на процедуру
m1:      sbrc flags, 1; анализ флага F1
jmp ProcKeyb; переход на процедуру
m2:      sbrc flags, 2; анализ флага F2
jmp ProcOper; переход на процедуру
m3:      sbrc flags, 3; анализ флага F3
jmp ProcScan; переход на процедуру
rjmp fon; возврат на начало фонового цикла
tim0:  ldi temp, 0x00; останов таймера TimeD
        out tccr0, temp
        sbr flags, 0x2; установка флага F1
        reti; возврат из процедуры
tim1:  sbr flags,0x8; установка флага F3
        reti; возврат из процедуры
tim2:  ldi temp, 0x00; останов таймера TimeO
        out tccr2, temp
        sbr flags, 0x4; установка флага F2
        reti; возврат из процедуры
inte0:  ldi temp, 0x0; сброс кода маски
        out eimsk, temp; запрет прерывания по INT0
        cbr flags, 0x40; сброс флага блокировки F6 (запрет реакции на INT0)
        sbr flags,0x1; установка флага F0
        mov bufs, scan; сохранение scan
        mov bufp, pos; сохранение pos
        reti; возврат из процедуры
ProcTimeD:  ldi temp, 0x1; пуск таймера TimeD
        out tccr0, temp
        cbr flags, 0x1; сброс флага F0
        jmp m1
ProcKeyb:  clr vpos; сброс счётчика позиции в ответном коде клавиатуры
      out porte, bufs; вывод хранимого кода сканирования
      in temp, porte; ввод ответного кода клавиатуры
      mov rkl, bufp; пересылка и сдвиг хранимой позиции нуля
      lsl rkl;       в коде сканирования
      lsl rkl
      slr:   ; цикл определения позиции вертикали
      lsr temp;
      brcc sum;
```

```
        inc vpos;
        rjmp slr
sum:  or rkl, vpos; формирование кода клавиши
        cpi rkl, 0x0a; сравнение кода с 0ah
        brsh  codc; переход, если ≥
        add xl, off; формирование адреса для записи в буфер клавиатуры
        st x, rkl; запись кода клавиши в буфер клавиатуры
        ldi xl, base; восстановление значения базы буфера
        ldi xh, 0x00
        inc off; увеличение смещения
        andi off, 0x03; свёртка смещения по mod4
        sbr flags,0x20; установка флага F5
        jmp strtim
codc:  mov rkk, rkl; сохранение кода команды
        sbr flags,0x10; установка флага F4
strtim:  ldi temp, 0x1; пуск таймера TimeO
          out tccr2, temp
          cbr flags, 0x1; сброс флага F1
          jmp m2
ProcOper:  sbr flags,0x40; разрешение INT0 (установка флага F6 )
          cbr flags, 0x4; сброс флага F2
          jmp m3
ProcScan:  sbrs flags, 0x6; анализ флага F6
          jmp modif;
          ldi temp, 0x1; формирование кода маски
          out eimsk, temp; разрешение прерывания по INT0
          out porte, scan;
          ldi temp, 0x0; сброс кода маски
          out eimsk, temp; запрет прерывания по INT0
modif:  lsl scan; модификация кода сканирования
          brcc count
          inc scan
count:  inc pos; наращивание счётчика позиции горизонтали
          andi pos, 0x03; свёртка счётчика позиции горизонтали по mod4
          andi flags, 0xf7; сброс флага F3
          jmp fon
```

3. ОТЛАДКА ПРОГРАММЫ В СРЕДЕ ISIS

Перед загрузкой .asm-файла программы в ISIS желательно предварительно выполнить её отладку в какой-либо иной интегрированной среде разработки, например в одной из версий AVRStudio. Последние имеют более информативный и удобный пользовательский интерфейс. Хотя отладочные средства входят и в состав ISIS, но они более подходят для прогона уже отлаженной программы при моделировании состояния проектируемой системы.

В любом случае первым шагом является запуск среды через вызов файла ISIS.exe. В результате на экран выводится рабочее окно среды (см. рис. 1), в котором необходимо скомпоновать схему моделируемого устройства (см. рис. 3–4), если у вас ранее не был создан и сохранён требуемый проект. В этом случае через меню **Файл → Новый проект** осуществляется открытие нового проекта, а через меню **Система → Размеры листа** задаётся требуемый формат чертежа. Далее через вызов меню **Файл → Сохранить проект как …** присваиваете имя проекту.

Для открытия ранее созданного проекта используется вызов **Файл → Открыть проект**, в выпадающем списке надо отметить имя требуемого проекта и нажать кнопку **Открыть**.

После компоновки схемы моделируемого устройства и формирования .asm-файла программы для используемого в схеме типа микроконтроллера осуществляется загрузка .asm-файла через вызов меню **Исходник → Добавить/Удалить файлы исходника…** . В выпадающем одноимённом окне (рис. 12) в разделе **Целевой процессор** задаётся тип моделируемого микроконтроллера, в разделе **Инструмент генерации кода** – тип компилятора, в разделе **Имя файла исходника** – имя загружаемого .asm-файла. В последнем разделе имеются кнопки **Новый, Удалить, Изменить.** Кнопки **Новый** и **Изменить** позволяют через системный браузер выбрать загружаемый файл источника. Кнопка **Удалить** производит очистку строки с именем файла, ранее используемого в проекте. Загрузка завершается нажатием на кнопку **Ок**. Открыть после загрузки файл исходника для просмотра или редактирования можно с помощью меню **Исходник → <имя файла>.asm.**

Режимы моделирования процесса функционирования системы определяются командами выпадающего меню **Отладка** (рис. 13):

- Запуск/Перезапуск отладки;
- Приостановить анимацию;
- Остановить анимацию;
- Выполнить;
- Выполнить без контрольных точек;
- Шаг с выходом;
- Шаг к;
- Анимация;
- Сброс всплывающих окон;
- Переступить;

– Выполнить до указанного времени;
– Сброс постоянных данных модели;
– Настройка диагностики…;
– Использовать удалённый монитор отладки;
– Шаг с заходом;
– По горизонтали;
– По вертикали.

Команда **Запуск / Перезапуск отладки** активизирует системы отладки, симуляции и анимации. Система отладки выполняет компиляцию загруженного .asm-файла и передаёт исполняемый файл программы симулятору при отсутствии ошибок в тексте программы. В противном случае в поле сообщений выводится сообщение о наличии ошибок и их количестве, а в масштабируемом поле отображается выпадающее окно с перечнем операций компилятора и обнаруженных ошибок (рис. 14).

Для устранения ошибок необходимо открыть текст файла-источника (см. ранее), устранить замеченные ошибки, сохранить отредактированный файл и повторно запустить отладку. При отсутствии ошибок компиляции активизируется симулятор и в масштабируемом поле поверх схемы моделируемого устройства отображаются выпадающие окна системы отладки (рис. 15), выбранные разработчиком из списка окон, появившихся в выпадающем меню **Отладка**:

– AVR Sourse Code;
– AVR Data Memory;
– AVR EPROM Memory;
– AVR CPU Registers;
– AVR I/O Registers;
– Simulation Log;
– Watch Window;
– AVR Program Memory.

Отмеченные в этом списке окна можно захватить левой кнопкой мыши и переместить в удобное для разработчика место поля экрана либо разместить их автоматически выбором меню **Отладка → По горизонтали** или **Отладка → По вертикали**. В каждом выпадающем окне отладки отображается текущее состояние соответствующих аппаратных средств или текстовое сопровождение процесса отладки, что позволяет разработчику оперативно реагировать на состояние моделируемой системы.

Параллельно с процессом отладки осуществляется анимация, отображающая состояние сигналов в шинах устройства, состояние устройств системы, в частности периферийного оборудования системы, позволяющая визуально наблюдать процессы ввода и вывода информации.

Команда **Выполнить** запускает процессы симуляции и анимации в автоматическом режиме прогона программы, если не установлены промежуточные точки останова. В этом режиме разработчик взаимодействует с моделью фактически в режиме реального времени с поправкой на быстродействие инструментального процессора. При низкоскоростном процессоре персонального компьютера задержки моделирования будут существенны, но независимо от характеристик процессора реальное время работы моделируемой системы отображается в поле комментариев в нижней строке рабочего окна.

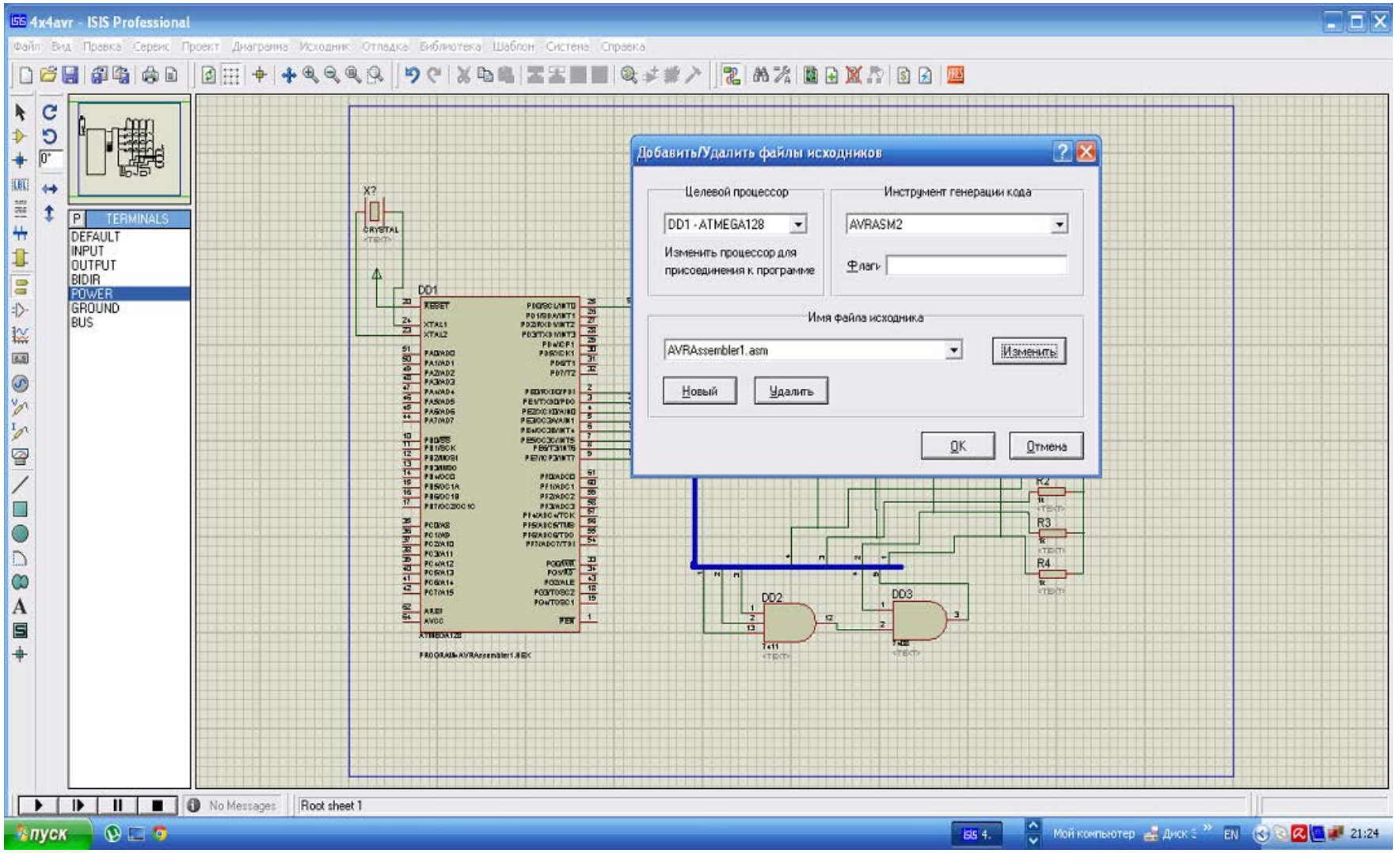

Рис. 12. Подключение файла-исходника

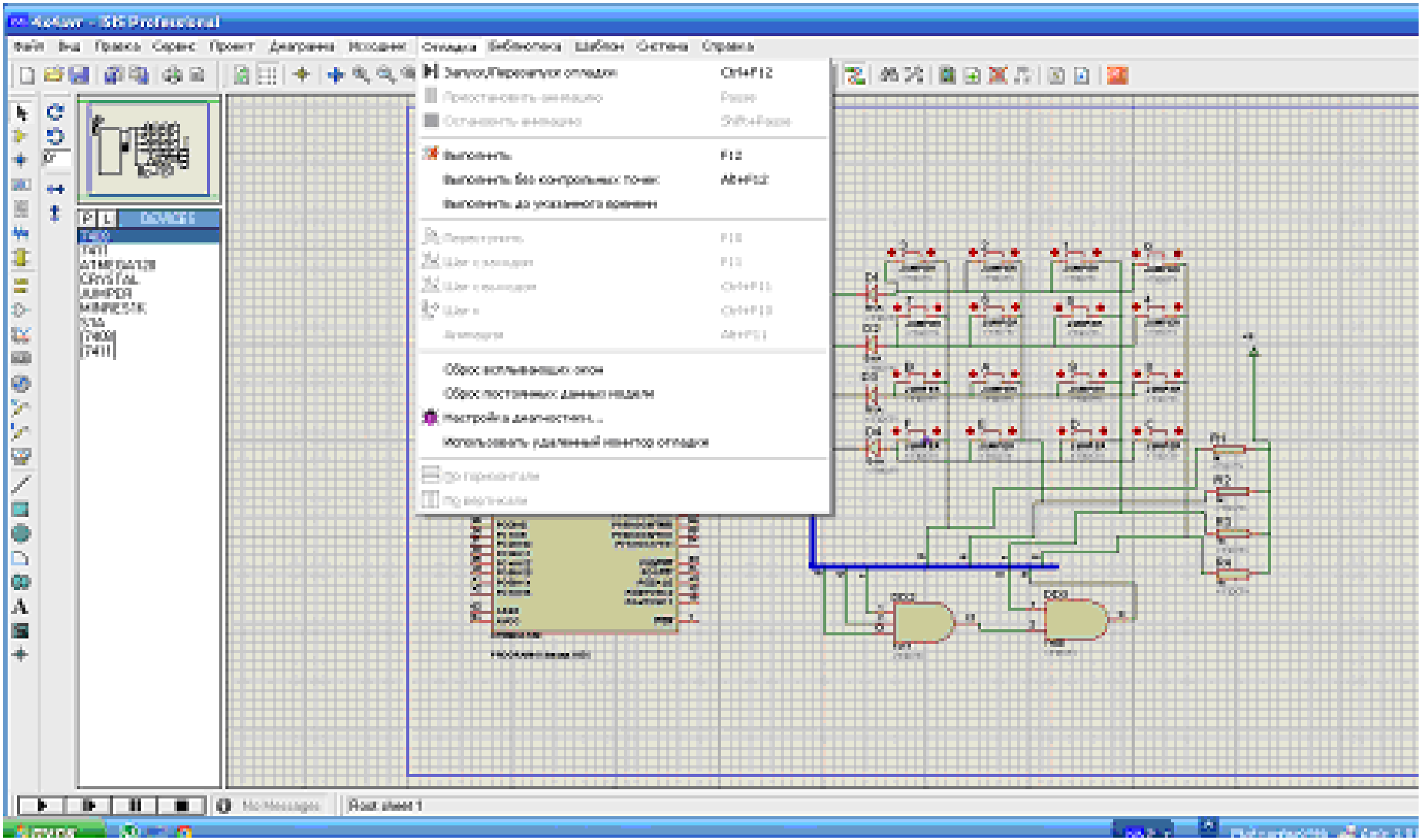

Рис. 13. Меню **Отладка**

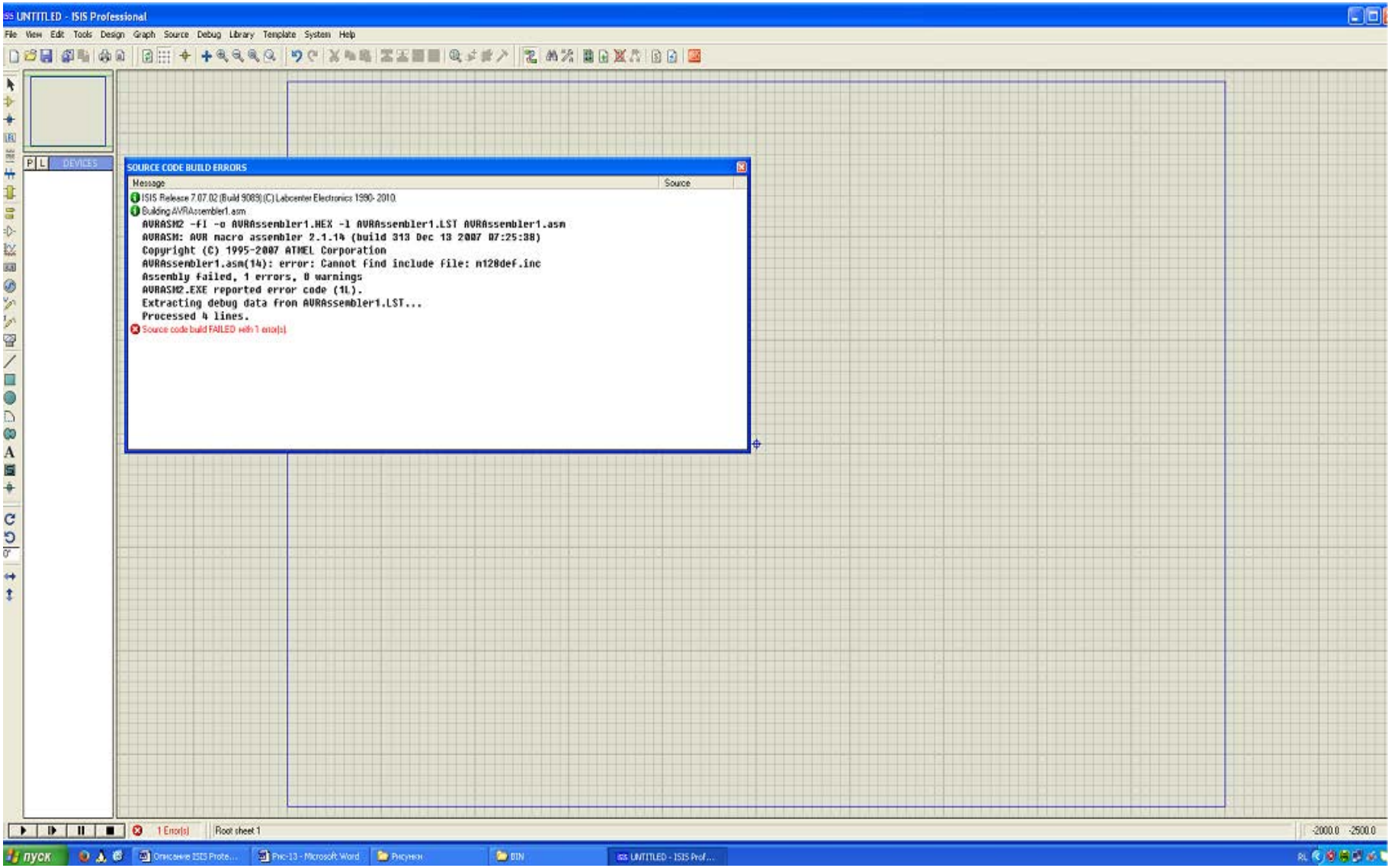

Рис. 14. Сообщение об ошибках

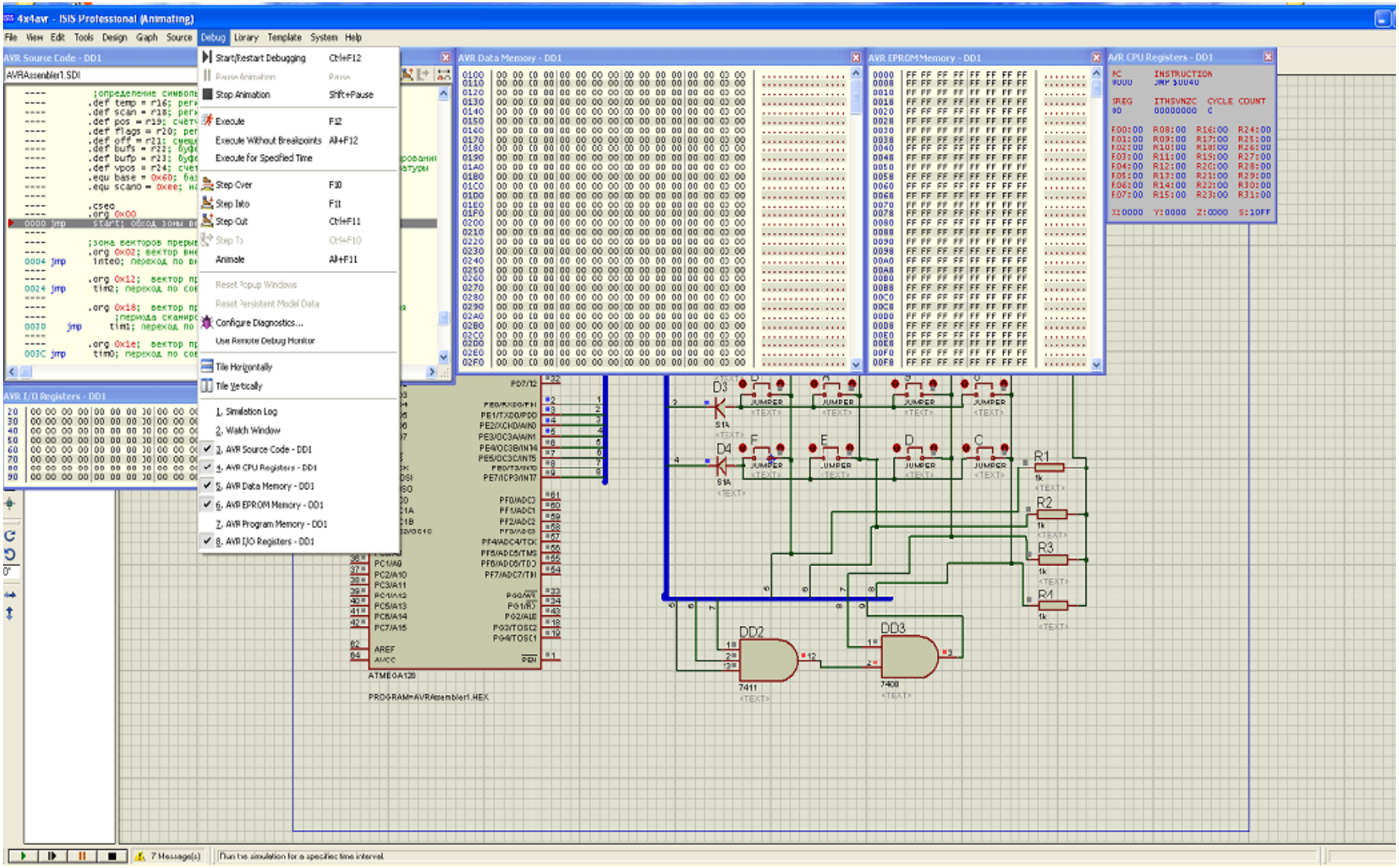

Рис. 15. Выпадающие окна системы отладки

После запуска симуляции и анимации в автоматическом режиме становится активной кнопка вызова команды **Приостановить анимацию**, нажатие на которую вызывает приостанов автоматического прогона с отображением текущего состояния аппаратных средств в выпадающих окнах системы отладки.

Выход из режима симуляции и анимации осуществляется по команде **Остановить анимацию**.

Если разработчика интересует состояние системы в определённый момент времени выполнения программы, то перед вызовом команды **Выполнить** необходимо поставить контрольную точку останова слева от мнемокода соответствующей команды в окне системы отладки AVR Sourse Code. Контрольная точка останова устанавливается двойным щелчком левой кнопкой мыши либо по командам выпадающего меню после щелчка правой кнопкой мыши. В этом случае команда **Выполнить** вызовет автоматический прогон программы до контрольной точки останова. Дальнейшее моделирование (как и с самого начала) может осуществляться с применением команд полуавтоматической отладки. Команда **Переступить** позволяет в покомандном (пошаговом) режиме отладки при попадании на начало процедуры выполнить процедуру в автоматическом прогоне и продолжать пошаговую отладку после возврата в основную программу.

Команда **Шаг с заходом** в отличие от команды **Переступить** продолжает моделирование в пошаговом режиме и при входе в процедуру. Команда **Шаг с выходом** при запуске её внутри тела процедуры определяет автоматическое завершение и выход из процедуры с остановом на очередной команде основной программы работы моделируемой системы. Команда **Шаг к** активизируется, если одна из последующих команд программы в окне отладки AVR Sourse Code выделена курсором, который устанавливается в виде фиолетовой полосы по щелчку левой кнопкой мыши на строке данной команды. Команда **Шаг к** запускает автоматический прогон от текущей команды программы до курсора. Команда **Анимация** позволяет продолжить процесс анимации после приостанова.

Интерфейс оператора при отладке допускает также имитацию механического воздействия на клавиатуру, выполнение измерения значений параметров электрических сигналов в цепях схемы устройства, фиксацию формы и временного соотношения импульсных и аналоговых сигналов и пр.

Так, например, на рис. 16 использованы типы компонентов **Щуп напряжения** и **Щуп тока**. Первый из них подключён к шине 7 и к шине питания, второй – к шине 5. Щуп напряжения с меткой 7 показывает уровень напряжения 4,99991 В на соответствующей вертикальной шине клавиатуры при разомкнутых джамперах (клавиша не нажата). Другой шуп напряжения показывает уровень напряжения питания +5 В. Щуп тока с меткой 5 показывает уровень тока в цепи источника логической единицы R1. Замыкание джампера с меткой 5, подключённого к шине 7, приводит (рис. 17) к изменению уровня напряжения на шине до 0,740969 В при появлении на входе этого джампера сканирующего нуля с выхода порта PE.1 (шина 2). Уровень напряжения питания не изменился.

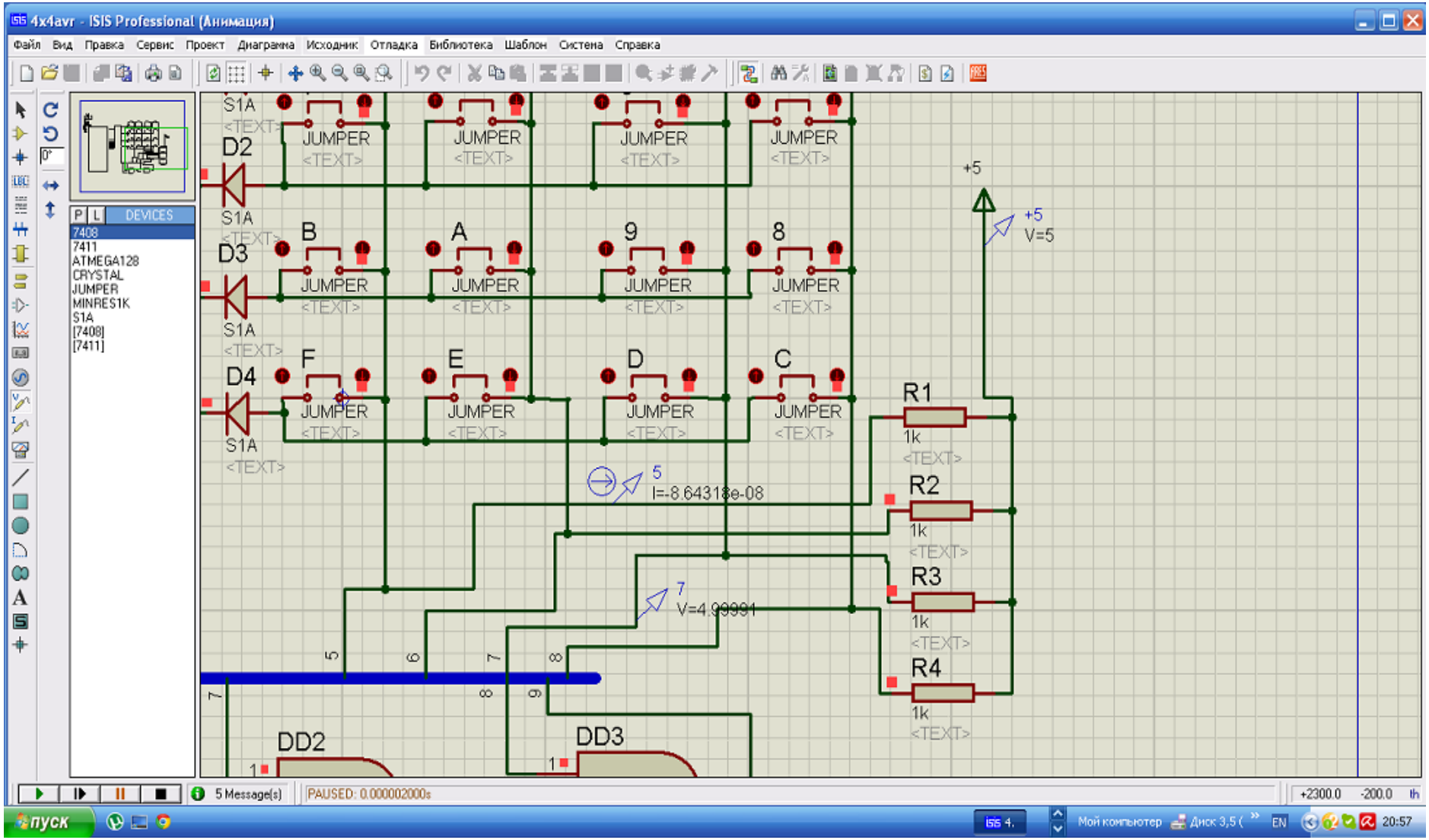

Рис. 16. Подключение измерителей

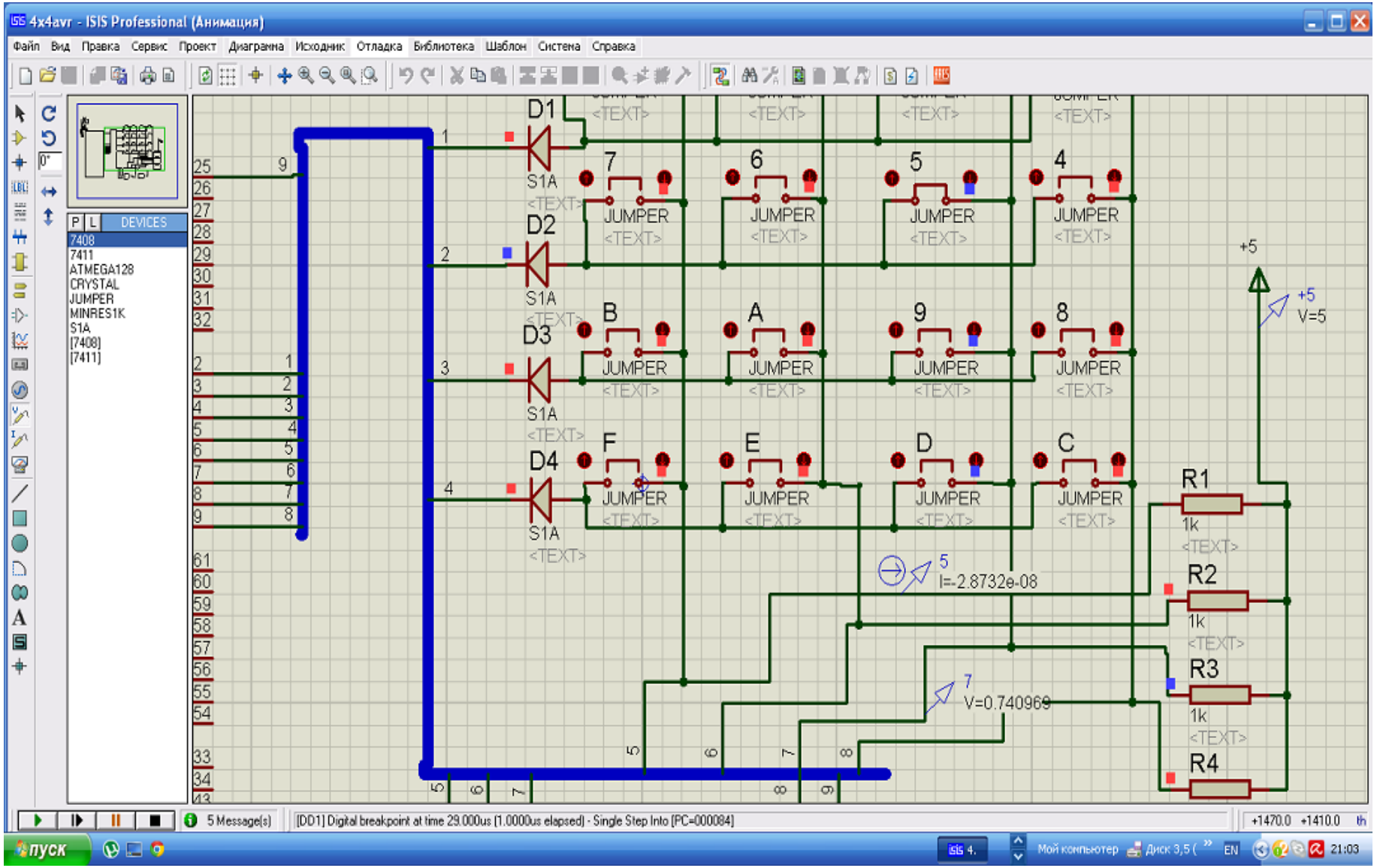

Рис. 17. Регистрация параметров

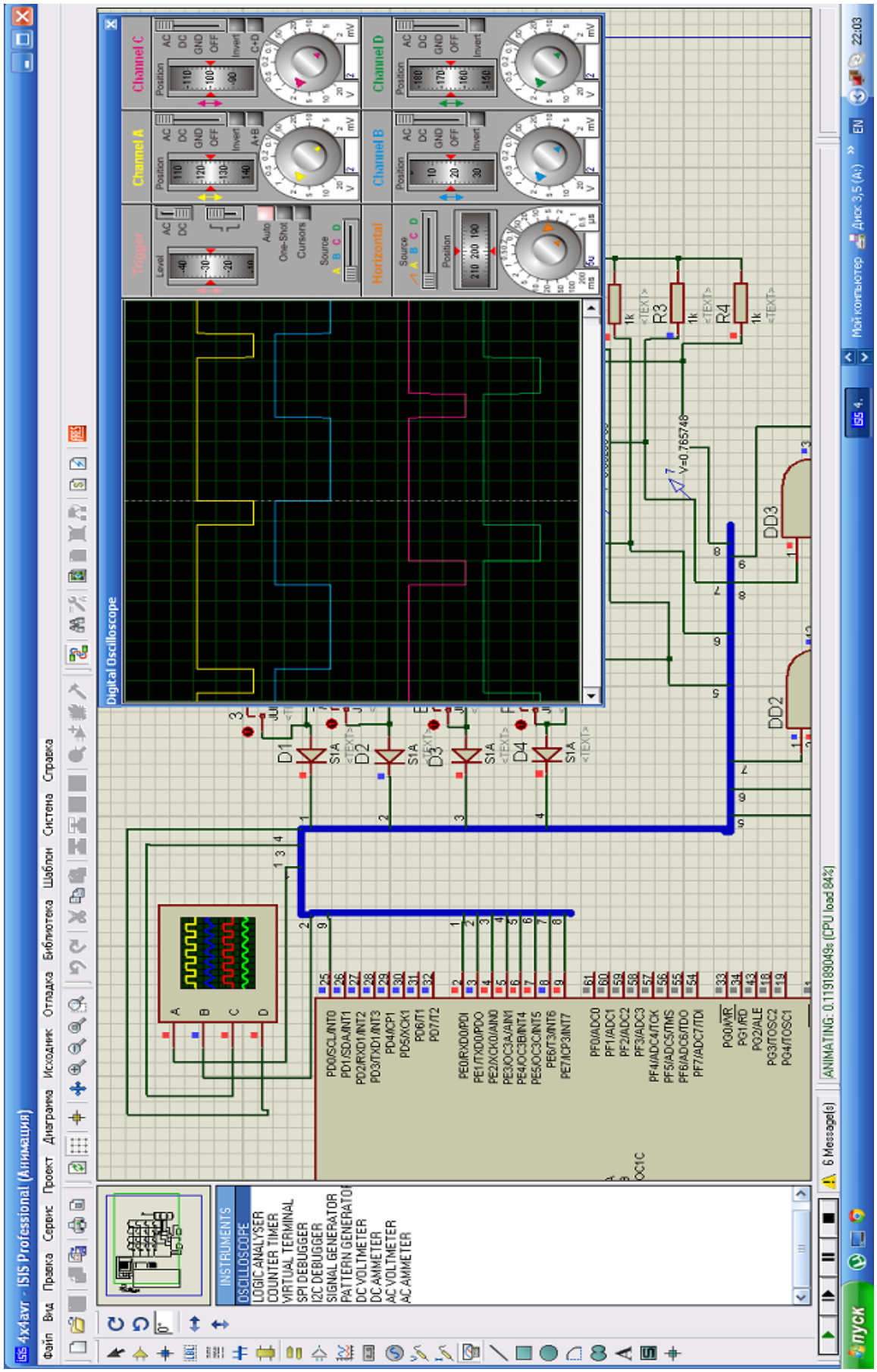

Рис. 18. Подключение осциллографа

На рис. 18 к выходам порта PE.0–PE.3 подключен тип компонента **Осциллограф**, каналы A, B, C и D которого соединены соответственно с шинами 1–4. При запуске отладки и анимации в режиме автоматического прогона программы на экране отображаются соответствующие временные диаграммы, позволяющие наблюдать перемещение сканирующего нуля по входным шинам клавиатуры. Используя верньеры позиционирования диаграмм по разным каналам, амплитудной и временной настройки изображения на экране осциллографа, можно установить требуемый масштаб отображения диаграмм, их взаимное размещение и синхронизировать с частотой развёртки осциллографа, что позволяет определить временные, амплитудные параметры импульсов и их взаимное смещение.

Подобным образом можно регистрировать интересующие разработчика параметры системы в требуемых точках схемы устройства и при необходимости изменять их, модифицируя как аппаратную часть системы, так и её программное обеспечение. Процесс отладки модели завершается при достижении удовлетворительных результатов.

4. МОДЕЛИРОВАНИЕ ВЫВОДА ИНФОРМАЦИИ НА БЛОК СЕМИСЕГМЕНТНЫХ ИНДИКАТОРОВ

В качестве примера рассмотрим схему управления блоком семисегментных индикаторов (рис. 19), выполненную на основе микроконтроллера ATmega103 фирмы Atmel. В качестве блока индикации выбран четырёхпозиционный блок семисегментных светодиодных индикаторов. Индикаторы всех позиций имеют общие входы семисегментного кода (A, B, C, D, E, F, G, H) и раздельные входы подключения общих катодов сегментов в каждой позиции (1, 2, 3, 4).

В качестве источника семисегментного кода задействован порт PB микроконтроллера, а по порту PE.0–3 выводится код сканирования. Шины порта PE.0–3 подключены к раздельным входам подключения общих катодов сегментов соответствующих позиций блока индикации.

Буфер данных предполагается разместить в оперативной памяти микроконтроллера, а таблицу перекодировки – в программной памяти. При побайтной выборке семисегментных кодов из программной памяти следует учитывать тот факт, что её ячейки имеют двухбайтовый формат. В этой связи следует удваивать словарный адрес обращения к таблице при побайтной выборке.

Для загрузки данных в оперативную память используется входной порт PF, к шинам PF.0–3 которого подключены источники логических единиц на резисторах R4–R1 и источники логических нулей, выполненные на соответствующих джамперах JP1–JP4. Выбор джамперов в качестве средств коммутации обусловлен тем, что при пошаговом моделировании и анимации джампер фиксирует своё состояние в отличие от кнопки – симулятор не может в одном шаге отрабатывать управление кнопкой и ввод данных с её выхода. Следует учитывать также задержки симуляции при замыкании и размыкании джамперов – замыкать и размыкать джамперы следует с упреждением минимум на один шаг работы симулятора.

4.1. Алгоритм вывода информации на блок семисегментных индикаторов

Рассмотрим в качестве примера вывод информации в динамическом режиме. В соответствии с требованиями стандартов частота смены изображения на блоке индикации должна быть не менее 40 Гц (период смены изображения равен 25 мс). При такой частоте заметна пульсация изображения. Поэтому желательно работать на более высоких частотах, например на частоте 100 Гц (период 10 мс).

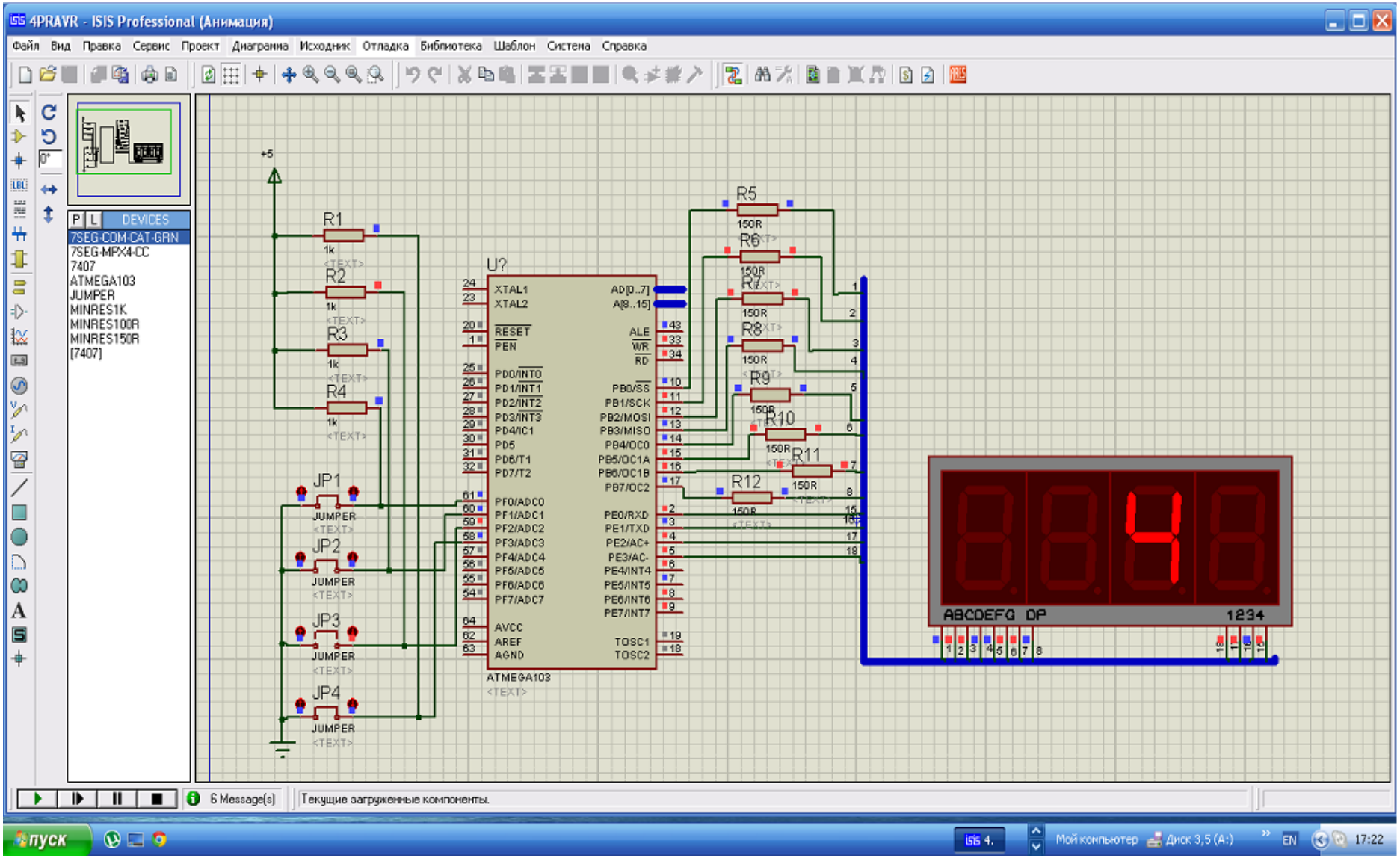

Рис. 19. Подключение блока семисегментных индикаторов

Если учесть, что в динамическом режиме вывода в каждый момент времени зажигается только одна позиция в блоке, то период горения одной позиции для четырёхпозиционного блока составит 2,5 мс.

Для формирования соответствующего вре-менного интервала можно использовать аппаратный или программный таймер периода (ТП), подобный тому, что использовался ранее для подсчета периода смены кода сканирования при управлении клавиатурой.

Процедуры алгоритма вывода информации приведены на рис. 20–23. В фоновом цикле (рис. 20) выполняется анализ флагов F8 и F9. Флаг F8 выставляется в процедуре INTTS по прерыванию таймера ТП (рис. 21), а флаг F9 – в процедуре ProcScan, которая вызывается по F8 = 1.

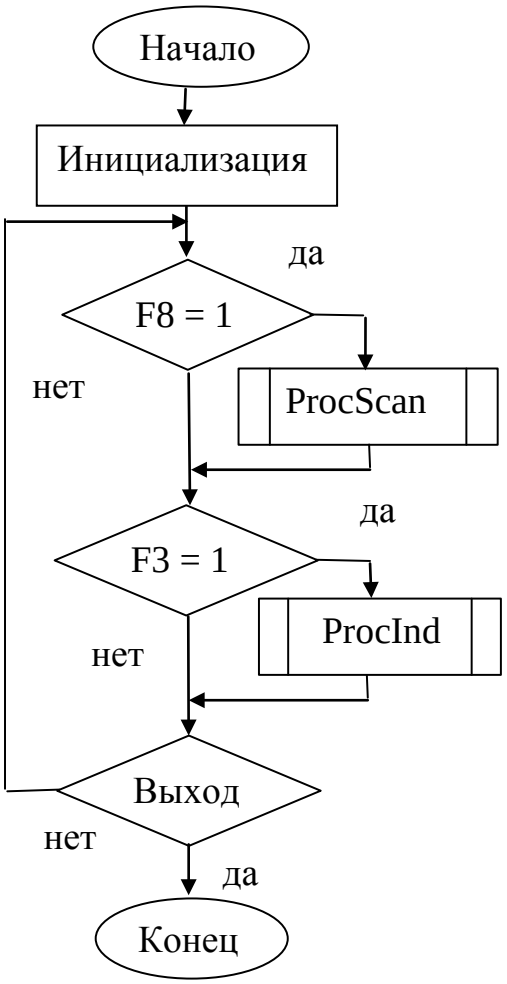

Рис. 20. Фоновый цикл

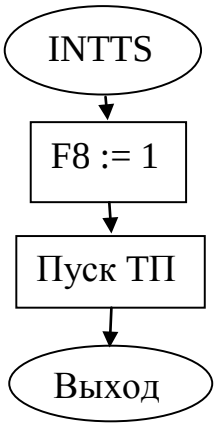

Рис. 21. Пуск ТП

Процедура ProcScan (рис. 22) осуществляет модификацию путём циклического сдвига содержимого регистра сканирования (РС), определяющего активным нулём в коде сканирования позицию зажигаемого индикатора. Начальное значение содержимого РС – 77h, что соответствует старшей позиции индикато-

ра. Позиция нуля в коде сканирования должна соответствовать значению смещения в буфере данных (СМБД) относительно базового адреса буфера данных (БАБД), отображаемых в блоке индикаторов. Начальное значение СМБД должно равняться нулю. Завершение модификации РС и СМБД сопровождается установкой флага F9, по которому из фонового цикла вызывается процедура ProcInd вывода информации в блок индикации (рис. 23).

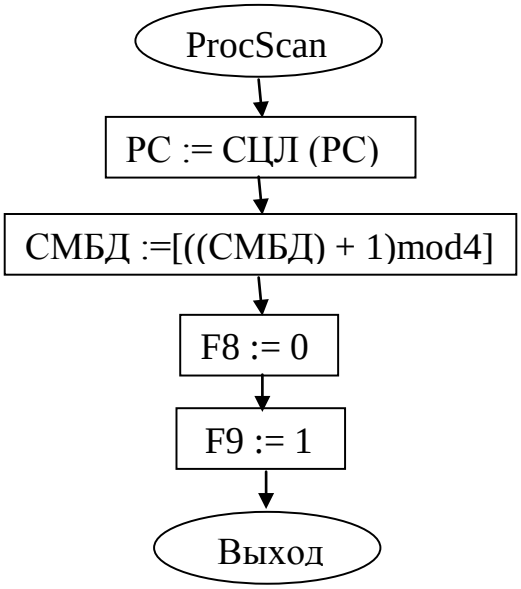

Рис. 22. Сдвиг РС

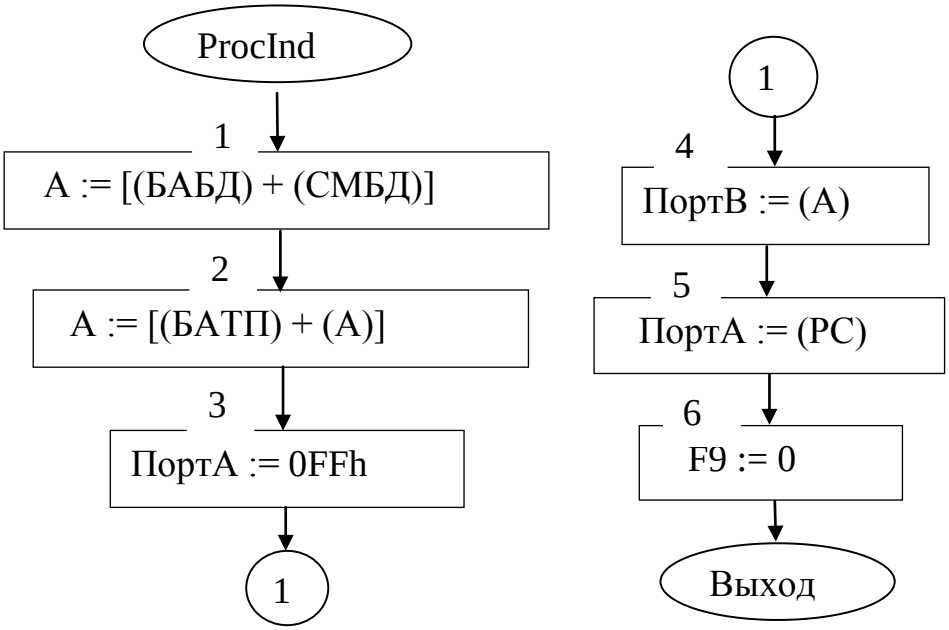

Рис. 23. Вывод символа

В первой вершине ProcInd в аккумулятор (А) из буфера данных загружается текущий символ. Во второй вершине код символа прибавляется в качестве смещения к базовому адресу таблицы перекодировки (БАТП) и по полученному ад-

ресу из таблицы выбирается семисегментный код символа. В третьей вершине осуществляется гашение индикаторов путём вывода в порт сканирования ПортА кода 0FFh, после чего в четвёртой вершине осуществляется вывод семисегментного кода в порт ПортВ, но зажигание текущего индикатора произойдёт только после выдачи в ПортА кода сканирования из PC. Гашение в третьей вершине используется для устранения наложения символов при смене кода сканирования.

4.2. Программная реализация алгоритма индикации

В соответствии с рассмотренным алгоритмом ниже представлена программа, составленная на языке ассемблера в системе команд микроконтроллера ATmega 103.

```
.include "m103def.inc"
; определение символьных переменных и констант
.def temp = r16; регистр временного хранения
.def scan = r18; регистр кода сканирования
.def pos = r19; счётчик позиции нуля в коде сканирования
.def flags = r20; регистр флагов
.def off = r21; смещение буфера данных
.equ base = 0x60; база буфера данных
.equ scan0 = 0x77; начальное значение кода сканирования
.cseg
.org 0x00
 jmp     start; обход зоны векторов
; зона векторов прерываний
.org 0x12;  вектор прерывания таймера 2
jmp     tim2; переход по совпадению таймера 2
.org 0x30 ; таблица перекодировки
.dw 0x063f, 0x4f5b, 0x6d66, 0x077d, 0x6f7f, 0x7c77, 0x5e39, 0x7179
; инициализация аппаратной среды
start: ldi zl,0x60; инициализация указателя Z на базу таблицы перекодировки
ldi zh, 0x00
ldi xl,base; инициализация указателя X на базу буфера данных
ldi xh, 0x00
in temp, pinf;  ввод с PortF кодов символов в буфер данных
st x+,temp
in temp, pinf
st x+,temp
in temp, pinf
```

```
st x+,temp
in temp, pinf
st x+,temp
ldi xl,base; инициализация базы буфера данных
ldi xh, 0x00
ldi temp, 0xff; настройка регистров направления портов PE и PB
out ddre, temp
out ddrb, temp
ldi scan, scan0; инициализация регистра сканирования
ldi pos, 0x00; инициализация счётчика позиции и смещения
ldi temp, low(ramend); инициализация указателя стека
out spl, temp
ldi temp, high(ramend)
out sph, temp
ldi temp, 0x80; разрешение локального прерывания по
out timsk, temp; совпадению в таймере tcnt2
ldi temp, 0x3f; загрузка в регистр сравнения ocr2 константы сравнения
out ocr2, temp;
ldi temp, 0x09; настройка управляющего регистра таймера tcnt2 на сброс
out tccr2, temp; по совпадению с константой в ocr2 и пуск таймера
sei; глобальное разрешение прерываний
fon:     ; цикл фоновой программы
sbrc flags, 2; анализ флага F8
jmp ProcScan; переход на процедуру
m3:  sbrc flags, 3; анализ флага F9
jmp ProcInd; переход на процедуру
rjmp fon; возврат на начало фонового цикла
tim2:   sbr flags, 0x4; установка флага F2
reti; возврат из процедуры
ProcScan:  lsl scan; модификация кода сканирования
brcc count
inc scan
count:  inc pos; наращивание счётчика позиции
andi pos, 0x03; свёртка счётчика позиции по mod4
andi flags, 0xfb; сброс флага F8
sbr flags, 0x8; установка флага F9
jmp m3
ProcInd:  ld temp, x+; выборка текущего символа
cpi xl,0x64
```

```
brne sum
ldi xl,base; инициализация указателя X на базу буфера данных
sum:  add zl, temp; формирование адреса перекодировки
lpm; выборка семисегментного кода в R0
ldi temp, 0xff
out porte, temp; гашение индикаторов
out portb, r0; вывод семисегментного кода
out porte, scan; зажигание индикатора
andi flags, 0xf7; сброс флага F9
ldi zl,0x60; инициализация указателя Z на базу таблицы перекодировки
ldi zh, 0x00
jmp fon
```

Симуляция и анимация работы устройства по данной программе может осуществляться как в шаговом, так и в автоматическом режимах прогона. Но в последнем случае перед запуском автоматического прогона ввод данных с джамперов порта PF требуется выполнить в шаговом режиме, если в оперативную память микроконтроллера необходимо ввести конкретные коды отображаемых символов. Изменять состояние джамперов необходимо с упреждением минимум на один командный цикл относительно момента их опроса, что обусловлено задержками работы симулятора и средств анимации.

5. МОДЕЛИРОВАНИЕ СИСТЕМ НА БАЗЕ МИКРОКОНТРОЛЛЕРОВ ФИРМЫ MICROCHIP TECHNOLOGY

Периферийные интерфейсные микроконтроллеры фирмы Microchip Technology семейства PIC характеризуются широкой номенклатурой, покрывающей многие области использования встраиваемых управляющих микропроцессорных систем. Микроконтроллеры PIC обладают всеми достоинствами RISC-процессоров и при близких характеристиках к микроконтроллерам фирмы Atmel имеют меньшую стоимость.

5.1. Программная реализация алгоритма вывода информации на семисегментный блок индикации

Схемотехническое исполнение моделируемого устройства (рис. 24) внешне мало отличается от ранее рассмотренного (см. рис. 19). Изменился только тип микроконтроллера – теперь это микроконтроллер PIC 16F874 фирмы Microchip Technology. Соответственно изменилась и внутренняя архитектура процессора, что получит отображение в программной реализации ранее разработанного алгоритма (см. рис. 20–23).

В отличие от микроконтроллеров ATmega, имеющих ортогональный доступ к регистровому файлу, микроконтроллеры семейства PIC16 относятся к аккумуляторному типу. Вследствие этого большинство операций выполняется с участием рабочего регистра W. Это приводит к существенному проценту команд предварительной загрузки W в общем объёме программы.

Для компилирования программ микроконтроллеров PIC в настройках среды ISIS следует указать ассемблер MPASM. Текст программы имеет следующий вид:

```
List P=16f874
#include p16f874.inc
org 0
goto start
org 0x4; вектор прерывания таймера
goto tim
org 0x20
start: movlw 0x30
movwf 0x4; формирование косвенного адреса базы буфера данных
movfw 0xd0
movwf TMR0; начальная установка таймера
bsf INTCON,GIE; разрешение глобального прерывания
bsfINTCON, T0IE; разрешение локального прерывания
movlw 0x0
bsf STATUS, RP0; настройка регистра статуса на первый банк
```

```
movwf 0x30
movwf OPTION_REG; настройка синхронизации таймера
movwf TRISB; настройка портов на вывод
movlw 0xf0
movwf TRISC
bcf STATUS,RP0; настройка регистра статуса на нулевой банк
movf  PORTC,w; ввод кода символа
andlw 0xf0; выделение кода символа
swapf 0x30; обмен тетрад
movf  PORTC,w; ввод кода символа
andlw 0xf0; выделение кода символа
movwf 0x31
swapf 0x31; обмен тетрад
movf  PORTC,w; ввод кода символа
andlw 0xf0; выделение кода символа
movwf 0x32
swapf 0x32; обмен тетрад
movf  PORTC,w; ввод кода символа
andlw 0xf0; выделение кода символа
movwf 0x33
swapf 0x33; обмен тетрад
movlw 0x0
movwf PORTB; обнуление шин порта
movlw 0xff
movwf PORTC; гашение индикаторов
movlw 0x77; начальная установка регистра сканирования
movwf 0x22
movlw 0xff
movwf 0x21; обнуление счётчика позиции нуля
movwf 0x20; сброс флагов
fon:  bcf STATUS,RP0; настройка регистра статуса на нулевой банк
btfsc 0x20,0; проверка флага F8
goto scan; переход на модификацию кода сканирования
f1:  btfsc 0x20,1; проверка флага F9
goto indic; переход на вывод информации в блок индикации
goto fon
tim:  bcf INTCON,T0IF; сброс флага переполнения
movlw 0xd0; перезагрузка таймера
movwf TMR0
bsf 0x20, 0; установка флага F8
retfie
scan: bsf STATUS,C; установка бита C в единицу
rlf 0x22; сдвиг регистра сканирования
```

```
incf 0x21; инкремент счётчика позиции
movlw 0x3
andwf 0x21; округление счётчика по mod4
movlw 0xee
btfss STATUS,C; пропустить следующую команду, если C = 1
movwf 0x22; перезагрузка регистра сканирования
bsf STATUS,C; установка бита C в единицу
bsf 0x20, 1; установка флага F9 вызова процедуры индикации
bcf 0x20,0; сброс флага F8 модификации регистра сканирования
goto f1; возврат в фон
indic:   movlw 0xff
movwf PORTC; гашение индикаторов
bcf 0x20,1; сброс флага индикации F9
movf 0x21,w; загрузка в W косвенного адреса базы буфера данных
addwf 0x4; формирование исполнительного косвенного адреса символа
movf 0x0,w; выборка символа в W
call dc; вызов процедуры перекодировки в семисегментный код
movwf PORTB; вывод семисегментного кода
movf 0x22,w
movwf PORTC; вывод кода сканирования (зажигание индикатора)
movlw 0x30; восстановление косвенного базового адреса буфера данных
movwf 0x4
goto fon
dc:          ; процедура перекодировки
addwf PCL
retlw 0x3f
retlw 0x06
retlw 0x5b
retlw 0x4f
retlw 0x66
retlw 0x6d
retlw 0x7d
retlw 0x07
retlw 0x7f
retlw 0x6f
retlw 0x77
retlw 0x7c
retlw 0x39
retlw 0x5e
retlw 0x79
retlw 0x71
end
```

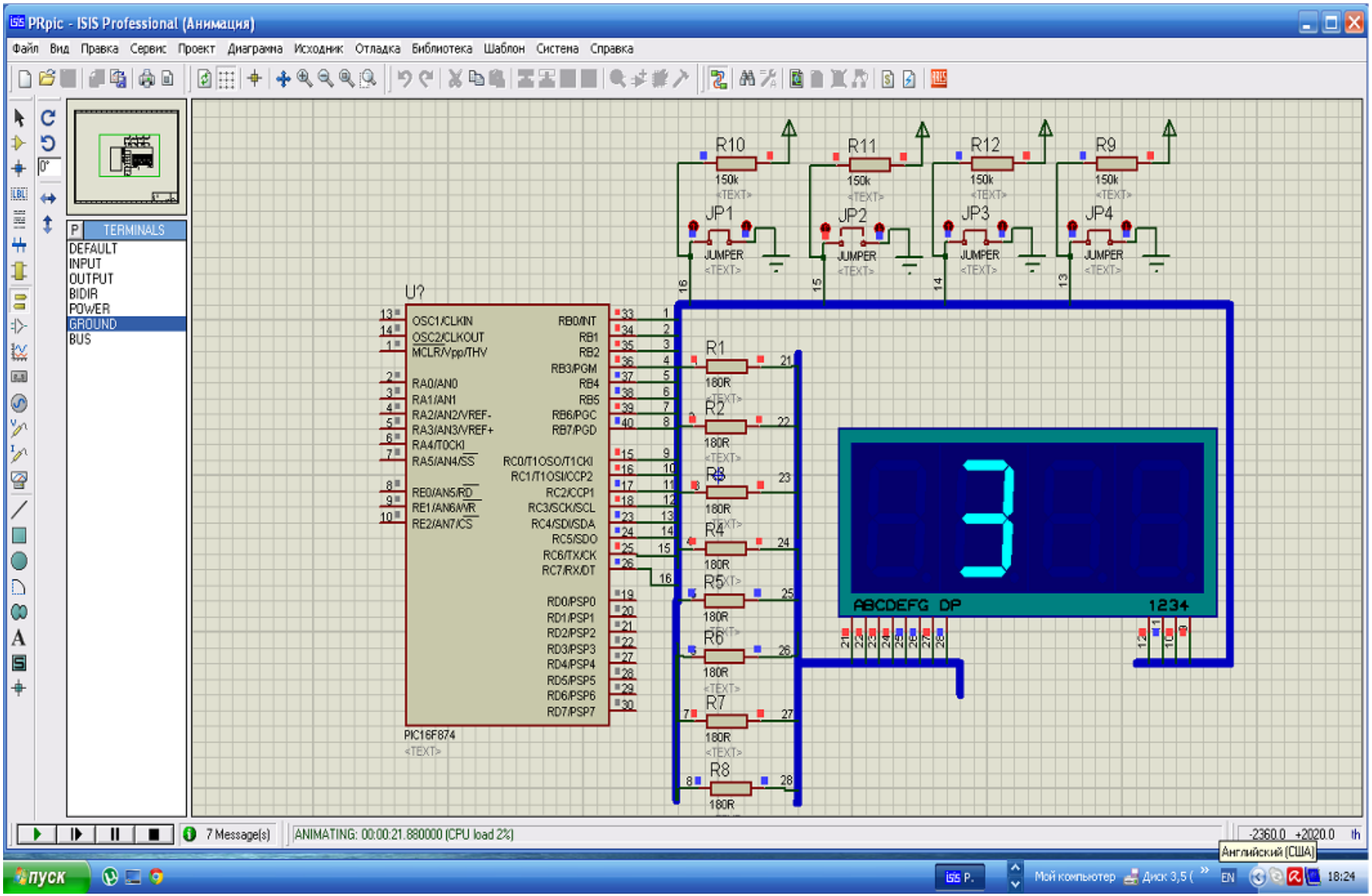

Рис. 24. Сопряжение микроконтроллера PIC с индикатором

Оригинальным в архитектуре PIC16 является организация таблицы перекодировки (см. выше). Базовый адрес таблицы заносится в программный счётчик при вызове процедуры, а значение кода символа, являющееся смещением в таблице, заносится в W перед вызовом процедуры. Суммирование базы и смещения с занесением в программный счётчик результата осуществляется по первой команде процедуры, что вызывает переход на соответствующую строку таблицы, состоящей из команд возврата из процедуры **RETLW K** (возврат из процедуры с занесением константы K в W), где K – семисегментный код, соответствующий смещению.

Кроме этого, вектор прерывания является общим для всех источников запросов прерывания, и перед переходом по вектору необходимо программно идентифицировать источник запроса. В данном примере используется только один источник запроса – флаг переполнения таймера, поэтому идентификации источника запроса не проводилось.

При прогоне программы в автоматическом режиме желательно поставить точки останова на командах ввода кода символа с джамперов по порту PC.4–7. Это позволит заносить в буфер данных требуемые сочетания символов.

5.2. Схема подключения клавиатуры к микроконтроллеру PIC16874

При составлении схемы моделируемого устройства была учтена особенность системы прерывания микроконтроллеров PIC. В отличие от других архитектур в микроконтроллерах данной архитектуры в системе прерывания имеется внешнее прерывание по изменению сигналов на шинах порта PB (в PIC16F874 – это шины RB.4–7).

Указанная особенность позволяет подключить матричную клавиатуру 4×4 к порту PB без дополнительных логических схем формирования запроса прерывания по нажатию на клавиатуру. Схема подключения приведена на рис. 25. Побитовая настройка шин порта PB также даёт возможность использовать только один порт для взаимодействия с клавиатурой. В представленной схеме шины RB.0–3 настроены на вывод кода сканирования, а шины RB.4–7 – на ввод ответного кода клавиатуры.

5.3. Программная реализация алгоритма опроса клавиатуры

В отличие от предшествующей программы вывода информации на семисегментный блок индикации, где использовался лишь один источник запроса прерывания, в приводимой ниже программе опроса клавиатуры задействовано четыре источника прерываний: таймер реального времени, ведущий подсчёт периода модификации регистра сканирования; прерывание по изменению сигналов на шинах порта PB; прерывание от таймера подсчёта времени, дребезга контактов и прерывание от таймера выдержки времени реакции оператора.

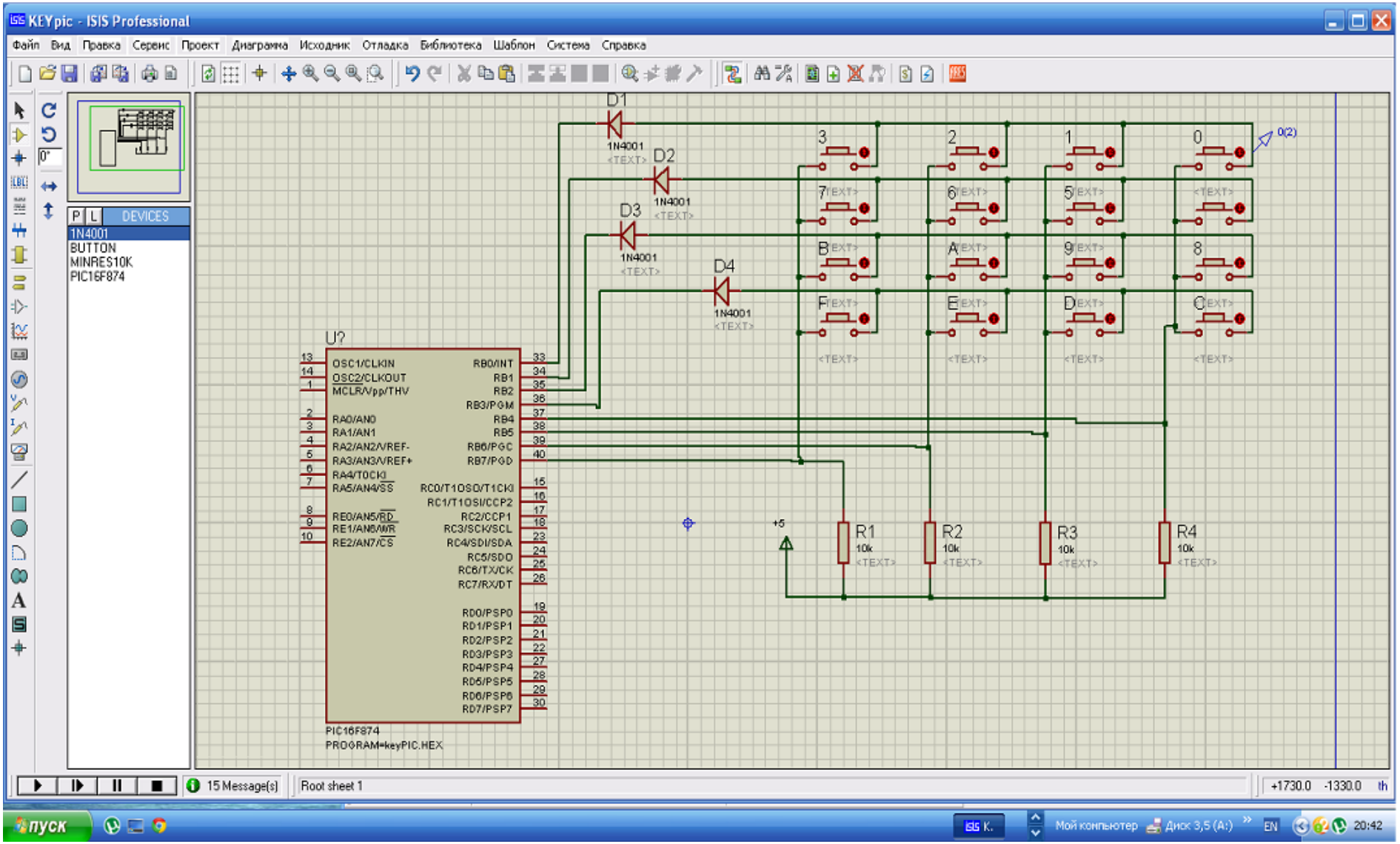

Рис. 25. Подключение матричной клавиатуры

Как упоминалось ранее, в архитектуре PIC16F874 имеется только общий вектор прерывания 0×04. В связи с этим после перехода по вектору необходимо организовать последовательный опрос флагов запросов от возможных источников с учётом их приоритетной цепочки. В рассматриваемой программе анализ флагов запросов выполняется по приоритетной цепочке меток tim:, bint:, tim1:, tim2:. Соответственно, высший приоритет имеет запрос от таймера реального времени (метка tim:), а низший приоритет у запроса от таймера выдержки времени реакции оператора (метка tim2:).

Система прерываний PIC16F874 имеет также два уровня группового маскирования (запрета) запросов прерывания: глобальное маскирование и периферийное маскирование. На каждом из этих уровней дополнительно возможно локальное маскирование любого отдельного запроса. Так, для запросов от TMR0 и RB.4–7 необходимо снимать маску глобального маскирования установкой бита GIE и соответствующие локальные маски установкой битов TOIE и RBIE, а для запросов от TMR1 и TMR2 требуется снять маску глобального маскирования установкой бита GIE, маску периферийного маскирования установкой бита PEIE и локальные маски установкой битов TMR1IE и TMR2IE.

В ходе программирования необходимо также учитывать распределение регистров специальных функций по страницам карты памяти данных и своевременно переключать банки памяти данных. Так, например, регистр направления TRISB находится в первом банке, а соответствующий ему PORTB – в нулевом банке.

```
List P=16f874
#include p16f874.inc
; присвоение символьных имён
scan equ 0x22; регистр сканирования
pos equ 0x21; счётчик позиции нуля в коде сканирования
flags equ 0x20; регистр флагов
bufscan equ 0x23; буфер кода сканирования
bufpos  equ 0x24; буфер позиции нуля в коде сканирования
temp equ 0x25; временной регистр
count equ 0x26; счётчик кода клавиши
comreg equ 0x27; регистр команды
offbuf equ 0x28; смещение при записи в буфер данных
org 0; вектор сброса
goto start
org 0x4; общий вектор прерывания
goto tim
org 0x20
```

```
start:  clrf flags
clrf offbuf; обнуление смещения
movlw 0x77
movwf scan; загрузка регистра сканирования
movlw 0x03
movwf pos; установка счётчика позиции нуля
movlw 0x30
movwf 0x4; установка косвенного адреса базы буфера клавиатуры
movlw 0xd0
movwf TMR0; начальная установка таймера
bsf INTCON,GIE; разрешение глобального прерывания
bsf INTCON, T0IE; разрешение локального прерывания по TMR0
bsf INTCON, RBIE; разрешение локального прерывания по RB.4–7
bsf INTCON, PEIE; разрешение периферийных прерываний 1
movlw 0x0
bsf STATUS, RP0; настройка регистра статуса на первый банк
bsf PIE1, TMR1IE; разрешение периферийного прерывания от TMR1
bsf PIE1, TMR2IE; разрешение периферийного прерывания от TMR2
movwf OPTION_REG; настройка синхронизации таймера TMR0
movlw 0xf0
movwf TRISB; настройка порта RB.4–7 на ввод и порта RB.0–3 на вывод
fon:            ; фоновый цикл
bcf STATUS,RP0; настройка регистра статуса на нулевой банк
btfsc flags,0; анализ флага модификации кода сканирования
goto modif
f1:  bcf STATUS,RP0; настройка регистра статуса на нулевой банк
btfsc flags,1; анализ флага прерывания по нажатию клавиатуры
goto dreb
f2:  bcf STATUS,RP0; настройка регистра статуса на нулевой банк
btfsc flags,2; анализ флага по выдержке времени дребезга
goto opros
f3:  bcf STATUS,RP0; настройка регистра статуса на нулевой банк
btfsc flags,3; анализ флага по выдержке времени реакции оператора
goto oper
goto fon
tim:  btfss INTCON,T0IF; проверка прерывания по переполнению TMR0
goto bint
bcf INTCON,T0IF; сброс флага переполнения TMR0
movlw 0xd0; перезагрузка таймера
```

```
movwf TMR0
bsf flags, 0; установка флага модификации кода сканирования
retfie
bint:  btfss INTCON,RBIF; проверка прерывания по RB.4–7
goto tim1
movf scan,w
movwf bufscan; сохранение значения кода сканирования
movf pos,w
movwf bufpos; сохранение значения кода позиции нуля
bcf INTCON, RBIE; запрет локального прерывания по RB.4–7
bcf INTCON,RBIF; сброс флага запроса прерывания по RB.4–7
bsf flags, 1; установка флага прерывания по нажатию клавиатуры
retfie
tim1:  btfss PIR1, TMR1IF; проверка прерывания по переполнению TMR1
goto tim2
bcf PIR1,TMR1IF; сброс флага запроса прерывания по TMR1
bsf flags, 2; установка флага по выдержке времени дребезга
bcf T1CON,TMR1ON; останов таймера TMR1
retfie
tim2: bcf PIR1,TMR2IF; сброс флага запроса прерывания по TMR2
bsf flags,3; установка флага по выдержке времени реакции оператора
bcf T2CON,TMR2ON; останов таймера TMR2
retfie
modif:  bsf STATUS,C; установка бита C в единицу
rlf scan; сдвиг регистра сканирования
incf pos; инкремент счётчика позиции
movlw 0x3
andwf pos; округление счётчика по mod4
movlw 0xee
btfss STATUS,C; пропустить следующую команду, если C = 1
movwf scan; перезагрузка регистра сканирования
bsf STATUS,C; установка бита C в единицу
movf scan,w;
movwf PORTB; вывод текущего кода сканирования
bcf flags,0; сброс флага модификации регистра сканирования
goto f1; возврат в фон
dreb:  bcf INTCON, RBIE; запрет прерывания по RB.4–7
movlw 0xff; загрузка и запуск таймера выдержки времени дребезга
movwf TMR1H
```

```
movlw 0xf0
movwf TMR1L
bsf T1CON,TMR1ON
bcf flags,1; сброс флага прерывания по нажатию клавиатуры
goto f2; возврат в фон
opros:  movf bufscan,w
movwf PORTB; вывод кода сканирования на момент нажатия
movf PORTB,w; ввод ответного кода клавиатуры
movwf temp
swapf temp; обмен тетрад в ответном коде
clrf count; очистка счётчика позиции нуля в ответном коде
cycle:  incf count    ; определение позиции нуля в ответном коде
rrf temp
btfsc STATUS,C; пропустить следующую команду, если C = 0
goto cycle
decf count
bcf STATUS,C; установка бита C в ноль
rlf bufpos; сдвиг хранимого кода позиции нуля
rlf bufpos
movf bufpos,w
iorwf count; формирование кода нажатой клавиши
movlw 0x6; сравнение на ≥ 0Ah
addwf count, w
andlw 0x10
btfss STATUS,C; пропустить следующую команду, если C = 1
goto buf; переход на запись данных в буфер
movf count,w
movwf comreg; сохранение кода команды
bsf flags,4; установка флага команды
optim: movlw 0xf0; запуск таймера выдержки времени реакции оператора
movwf TMR2
bsf T2CON,TMR2ON
bcf flags,2; сброс флага по выдержке времени дребезга
goto f3; возврат в фон
buf:  movf 0x4,w; формирование исполнительного адреса записи в буфер
addwf offbuf,w
movwf 0x4
movf count,w
movwf 0x0; косвенная запись в буфер
```

```
bsf flags,5; установка флага данных
incf offbuf,w; инкремент смещения
andlw 0x3; свертка смещения по mod4
movwf offbuf; сохранение смещения
movlw 0x30
movwf 0x4; восстановление косвенного адреса базы буфера; клавиатуры
goto optim
oper:  bsf INTCON, RBIE; разрешение прерывания по RB.4–7
bcf flags,3; сброс флага по выдержке времени реакции оператора
goto fon; возврат в фон
end
```

Сравнительный анализ плотности упаковки программного текста и быстродействия для рассмотренных архитектур показывает, что программная реализация одних и тех же алгоритмов более эффективна на микроконтроллерах ATmega фирмы Atmel. Это объясняется существенной усечённостью системы команд микроконтроллеров семейства PIC16x в рамках RISC-архитектуры и её аккумуляторным типом, требующим промежуточных пересылок операндов в рабочий регистр W.

6. МОДЕЛИРОВАНИЕ СИСТЕМ НА БАЗЕ МИКРОКОНТРОЛЛЕРОВ СЕМЕЙСТВА MCS-51 ФИРМЫ INTEL

Классическая архитектура базового подсемейства MCS-51 ближе к CISC-архитектурам аккумуляторного типа из-за достаточно функциональной системы команд, хотя имеет разделённые средства памяти данных и памяти команд, характерные для Гарвардской архитектуры и RISC-систем.

Микроконтроллеры MCS-51 получили широкое распространение, качественно модифицировались и адаптировались многими фирмами, в частности такими, как Atmel, Infineon (бывший Siemens) и др. В расширениях MCS-151 и MCS-251 наблюдается отход от аккумуляторной архитектуры, наращивание системы команд операциями с ортогональным доступом к средствам памяти, конвейеризация вычислительного процесса, введение новых протоколов обмена по системной шине, что делает их конкурентоспособными по отношению к другим производителям.

6.1. Построение схемы моделируемого устройства

Внешнее подключение периферийных устройств к базовому микроконтроллеру подобно ранее рассмотренному (рис. 26). В качестве базового выбран микроконтроллер 80C51, к которому подключены и матричная клавиатура, и блок семисегментных индикаторов. В результате совмещения в одном устройстве средств ввода и вывода информации изменилась процедура модификации кода сканирования, в которую составной частью вошла процедура вывода данных на блок индикаторов. Кроме этого, в микроконтроллере 80C51 имеется только два аппаратных таймера вместо ранее используемых в ATmega и PIC трёх аппаратных таймеров. В этой связи функции одного из аппаратных таймеров необходимо воспроизвести в программном таймере.

На рис. 27 представлен совмещённый вариант алгоритма модификации кода сканирования, управления программным таймером и вывода содержимого буфера данных на блок индикаторов. На программный таймер были возложены функции таймера выдержки времени дребезга контактов клавиатуры. Соответственно, два аппаратных таймера микроконтроллера реализуют функции таймера реального времени (таймер T/C0) и таймера выдержки времени реакции оператора (таймер T/C1).

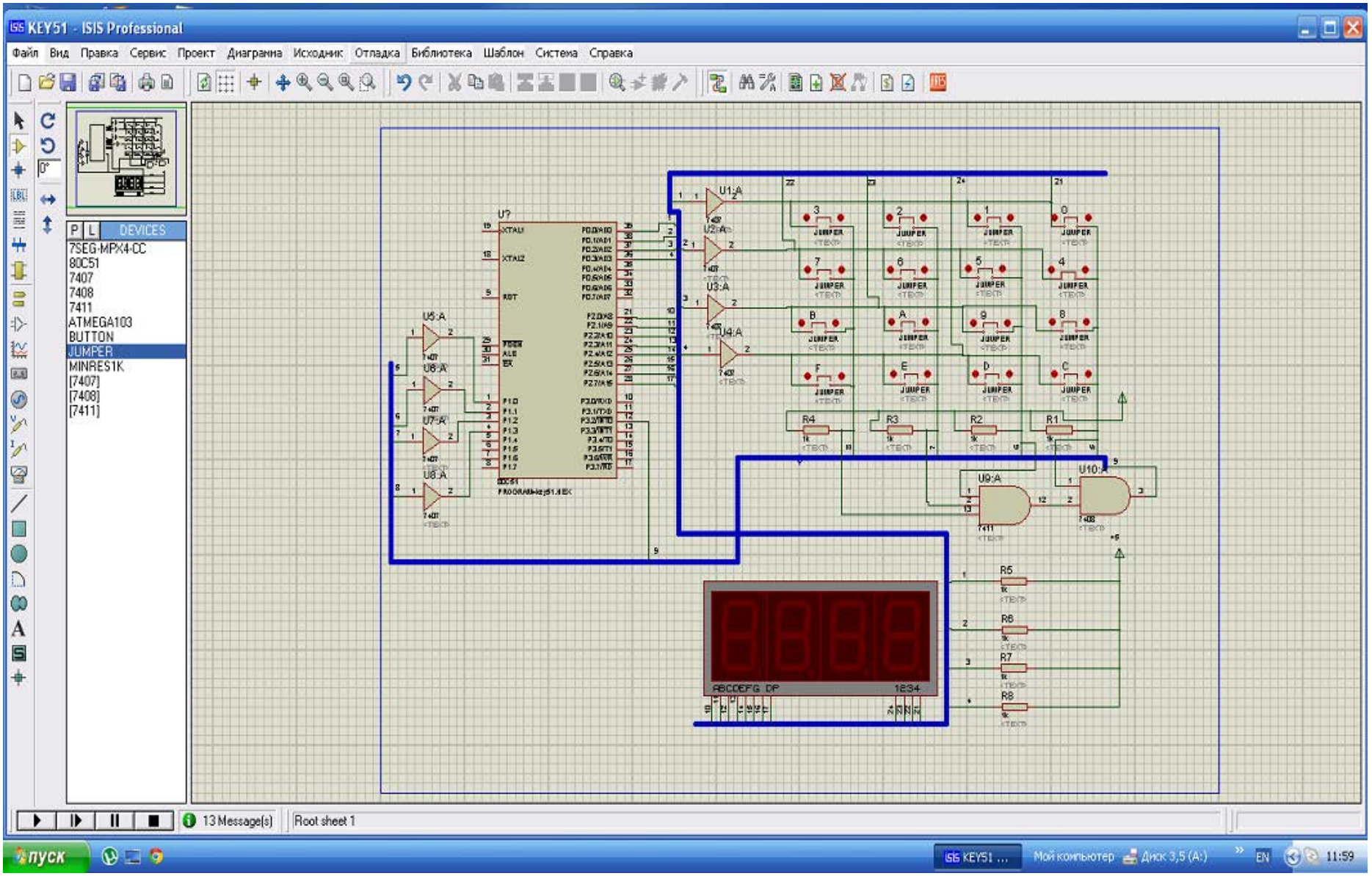

Рис. 26. Совместное подключение клавиатуры и индикатора

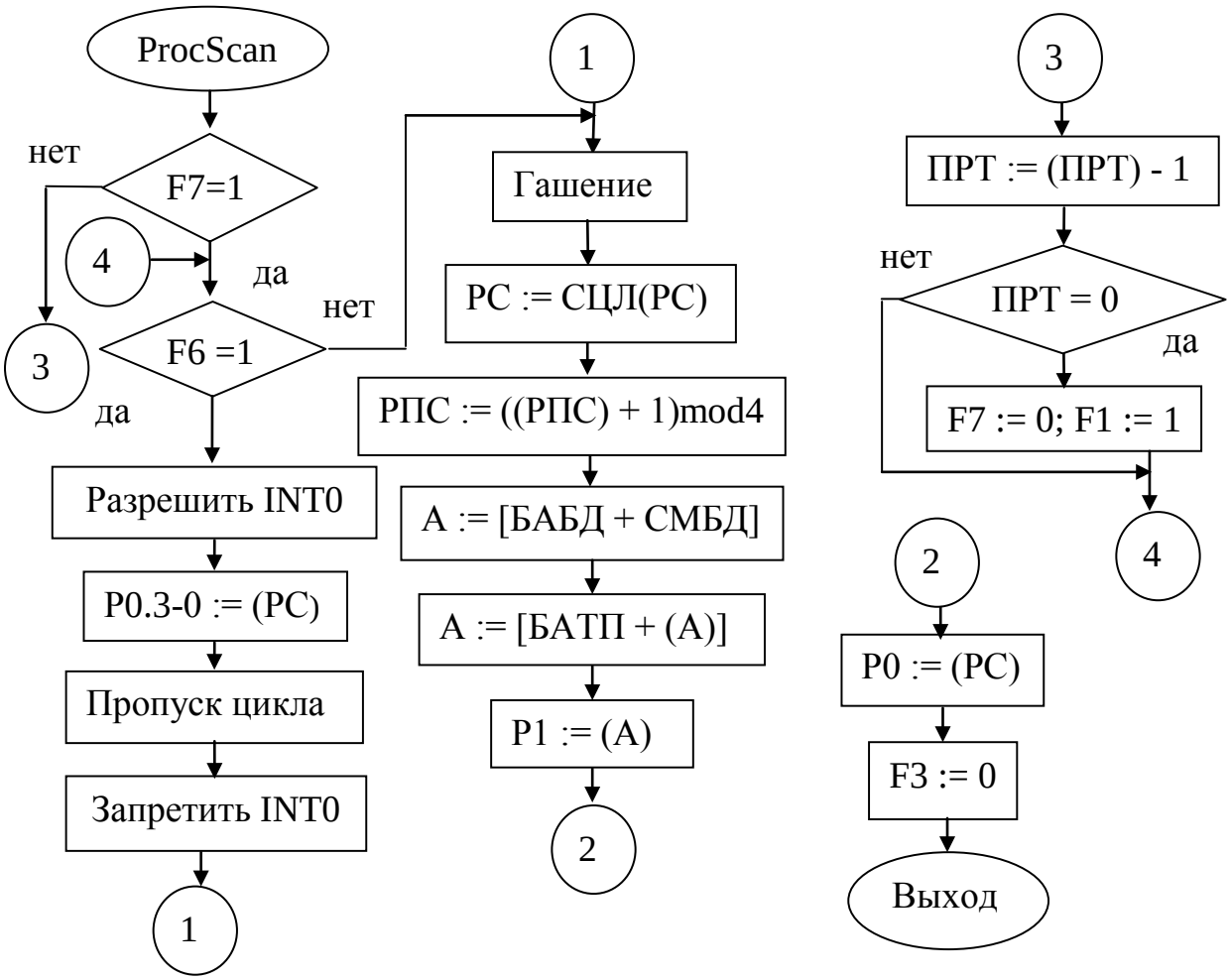

Рис. 27. Модификация кода

Аббревиатуры, используемые на рис. 27, соответствуют ранее приведённым на рис. 23 с тем отличием, что имена портов изменены на реальные для 80C51. Также введён новый флаг F7, единичное значение которого означает активное состояние программного таймера (ПРТ). Обнуление ПРТ сообщает о завершении выдержки времени дребезга контактов клавиатуры и вызывает установку флага F1, по которому из фонового цикла производится вызов процедуры опроса клавиатуры.

6.2. Программа функционирования моделируемого устройства

```
;Опрос клавиатуры 4×4 в режиме прерывания
indad equ r1; регистр косвенной адресации
scan equ 02; регистр кода сканирования
pos equ 03; регистр позиции нуля в коде сканирования
```

```
keybuf equ 00; регистр базы буфера данных
keyoff equ 04; регистр смещения в буфере данных
com equ 05; регистр кода команды
ptim equ 06; счётчик программного таймера
klav equ 07; регистр кода клавиши
flags equ 20h; ячейка флагов процесса
hflags equ 21h; ячейка флагов данных и команд
bufpos equ 30h; буфер кода позиции
bufscan equ 31h; буфер кода сканирования
temp equ 32h; временной регистр
.CODE
org 0; вектор пуска
 jmp strt
org 3; вектор прерывания по нажатию клавиатуры
jmp klint
org 0bh; вектор прерывания по переполнению T/C0
jmp rtim
org 1bh; вектор прерывания по переполнению T/C1
jmp timeop
org 20h
table:          ; таблица перекодировки
.dw 3F06h,5B4Fh,666Dh,7D07h,7F6Fh,777Ch,395Eh,7971h
org 30h         ; инициализация
strt:  mov DPTR,#table
mov flags,#0;
mov scan, #077h; код сканирования
mov pos, #3; позиция нуля в коде сканирования
mov keybuf, #70h; база буфера данных
mov keyoff, #0
mov tmod,#11h; режим 16 бит T/C0 и T/C1
mov sp, #78h; установка указателя стека
setb et0; разрешение прерывания по T/C0
setb et1; разрешение прерывания по T/C1
setb ea; общее разрешение прерывания
setb flags.6; разрешение реакции на прерывание INT0
mov TL0,#0d0h; начальная установка T/C0
mov TH0,#0ffh
setb TR0; запуск T/C0
fon:            ; фоновый цикл
jb flags.0, dreb  ; проверка нажатия
```

```
mf1:    jb flags.1,opros; проверка переполнения таймера дребезга
mf2:    jb flags.2, oper; проверка переполнения T/C1
mf3:    jb flags.3, procscan; модификация сканирования
jmp fon
dreb:   clr flags.0;
mov ptim,#0d0h; загрузка программного таймера дребезга
setb flags.7; запуск программного таймера дребезга
jmp mf1
opros:  clr flags.1
mov a,bufpos; формирование кода клавиши
rl a
rl a
mov klav,a;
mov P0,bufscan; сканирование клавиатуры
mov P1,#0ffh
mov a,P1;  ввод с клавиатуры
m:  inc temp; подсчёт кода клавиши
rrc a
jc m
dec temp
mov a,temp
orl a,klav;
mov klav, a; сохранение кода клавиши
add a,#06; проверка на > 09
jb a.4,comm
mov a,keybuf; базовый адрес буфера данных
add a,keyoff; исполнительный адрес буфера данных
mov indad,a
mov a,klav
mov @indad, a; занесение кода клавиши в буфер данных
inc keyoff; модификация смещения
mov a,keyoff
anl a,#3
mov keyoff, a
setb hflags.1; установка флага данных
top:  mov tl1,#0d0h; запуск T/C1
mov th1,#0ffh
setb TR1
```

```
jmp  mf2
comm:  mov a, klav
mov com, a; сохранение кода команды
setb hflags.0; установка флага команды
jmp top
oper:  clr flags.2;
clr tr1; останов T/C1
setb flags.6; разрешение реакции на INT0
jmp mf3
procscan:       ; процедура модификации и индикации
jb flags.7,progtim; переход к инкременту программного таймера
test:  jnb flags.6,modif; переход на модификацию кода сканирования
setb ex0; разрешение прерывания по INT0
mov p0, scan; сканирование по P0
nop
clr ex0; запрет прерывания по INT0
modif:
mov p0,#0ffh; гашение индикатора
mov a, scan; модификация scan
rl a
mov scan, a;
inc pos; модификация pos
mov a, pos;
anl a,#03; округление pos по mod4
mov pos, a
; вывод на индикатор
push 01; сохранение регистра косвенной адресации
mov a, keybuf; формирование адреса выборки из буфера данных
add a, pos
mov indad, a; загрузка регистра косвенной адресации
mov a, @indad; выборка символа из буфера данных
movc a,@a+dptr; обращение к таблице перекодировки
mov p2, a; вывод семисегментного кода
mov p0, scan; вывод кода сканирования (зажигание индикатора)
pop 01; восстановление регистра косвенной адресации
clr flags.3
jmp fon
progtim:  djnz ptim, test; декремент программного таймера дребезга
```

```
setb flags.1; установка флага срабатывания программного таймера
clr flags.7; сброс флага запуска программного таймера
jmp test
klint:  setb flags.0; установка флага нажатия на клавиатуру
mov bufscan, P0; сохранение текущего кода сканирования
mov bufpos, pos; сохранение текущего кода позиции нуля
clr flags.6; очистка флага разрешения реакции на нажатие
reti
timeop:  setb flags.2; установка флага переполнения T/C1
reti
rtim: setb flags.3; установка флага переполнения таймера реального времени
mov tl0,#0d0h; перезагрузка таймера реального времени
mov th0,#0ffh
setb tr0; запуск таймера реального времени
reti
end
```

Для компилирования программ микроконтроллеров MCS-51 в настройках среды ISIS следует указать ассемблер ASEM51.

7. МОДЕЛИРОВАНИЕ ПРОЦЕССА УПРАВЛЕНИЯ БИПОЛЯРНЫМ ШАГОВЫМ ДВИГАТЕЛЕМ

Шаговые двигатели позволяют обеспечить наиболее функциональное цифровое управление режимами их работы со стороны процессора. В моделируемом устройстве (рис. 28) используется двухобмоточный биполярный двигатель с тремя парами полюсов ротора. Иллюстрационные схемы двигателя представлены на рис. 29. Индексами 1а и 1б обозначены полюса первой статорной обмотки, индексами 2а и 2б – полюса второй статорной обмотки. Символы N и S на противоположных концах диаметральных сечений соответствуют парам полюсов ротора шагового двигателя и обмоток статора. Пары полюсов ротора сдвинуты относительно друг друга на 60°, а пары полюсов статорных обмоток – на 90°.

На рис. 30 приведена требуемая временная диаграмма состояния намагниченности полюсов статорных обмоток за период одного оборота двигателя. В каждом такте управления символами «+» и «–» обозначена полярность прикладываемого к выводам статорных обмоток импульсного напряжения. Протекающие при этом токи в обмотках создают соответствующую намагниченность S и N полюсов статора. За каждые четыре такта управления ротор двигателя совершает поворот на один квадрант (рис. 29, *а, б, в, г*), а за шестнадцать тактов – один полный оборот.

На схеме (рис. 28) в качестве процессора U1 выбран микроконтроллер ATmega 128. Для вывода управляющих кодов используется младшая тетрада порта PORTD. Для согласования с транзисторными мостовыми комплементарными парами, выполненными на полевых транзисторах Q1–Q8, используются буферы/усилители микросхемы U2. Средние точки пар Q1, Q2 и Q3, Q4 соединены соответственно с выводами обмоток 1а и 1б, а средние точки пар Q5, Q6 и Q7, Q8 – с выводами обмоток 2а и 2б.

Алгоритм управления двигателем представлен на рис. 31. Анализ временной диаграммы (рис. 30) показывает, что для поворота двигателя на один квадрант требуется всего четыре различных кода управления, выставляемых по очереди в четырёх последовательных тактах, и это сочетание повторяется четыре раза за один оборот.

В алгоритме предлагается разместить эти четыре кода управления в одном массиве и выводить очередной код в очередном такте, наращивая смещение (СМ) адреса выборки из массива при каждом обращении и выполняя свёртку полученного смещения по mod4.

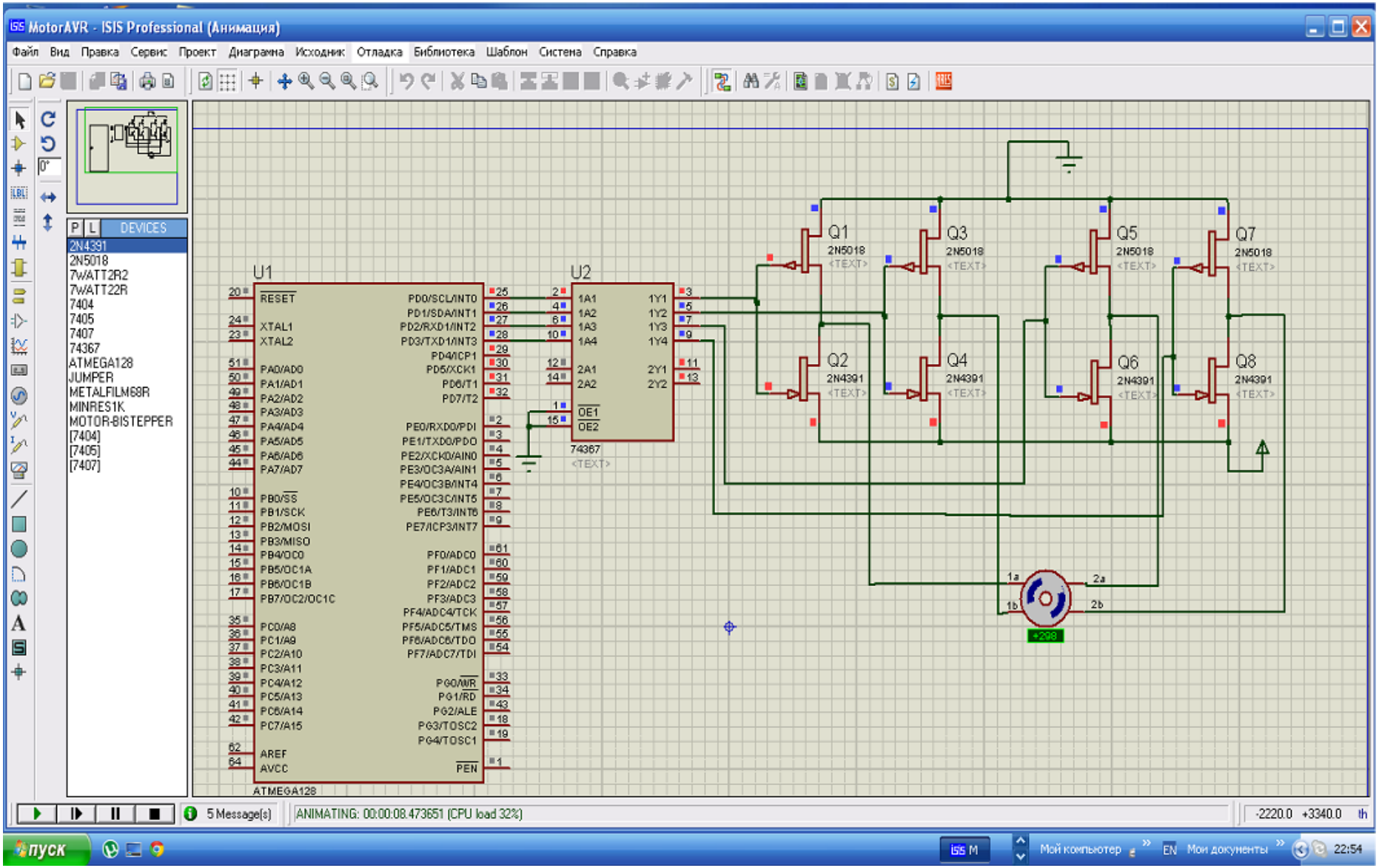

Рис. 28. Подключение шагового двигателя

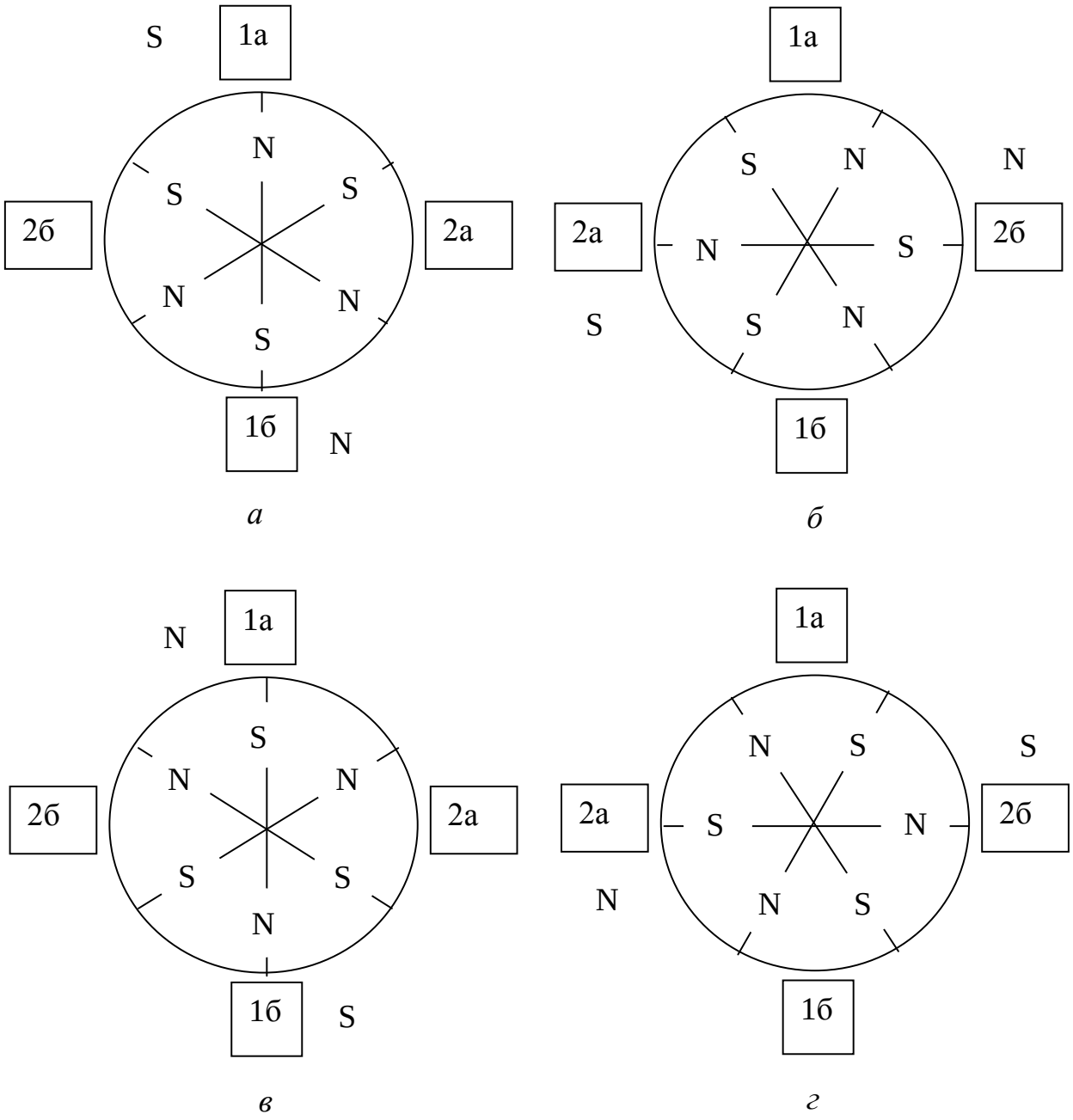

Рис. 29. Состояние намагниченности полюсов

1а	+	–	–	–	+	–	–	–	+	–	–	–	+	–	–	–
1б	–	–	+	–	–	–	+	–	–	–	+	–	–	–	+	–
2а	–	+	–	–	–	+	–	–	–	+	–	–	–	+	–	–
2б	–	–	–	+	–	–	–	+	–	–	–	+	–	–	–	+
Такт	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Рис. 30. Диаграмма смены намагниченности полюсов

Для формирования требуемой временной задержки удержания кода управления на шинах двигателя используется таймер реального времени (ТРВ), реакция на переполнение которого осуществляется по прерыванию.

В прерывающей процедуре (Тайм) осуществляется перезапуск ТРВ и установка флага завершения выдержки времени (Ф), единичное значение которого в основной программе разрешает следующий цикл управления.

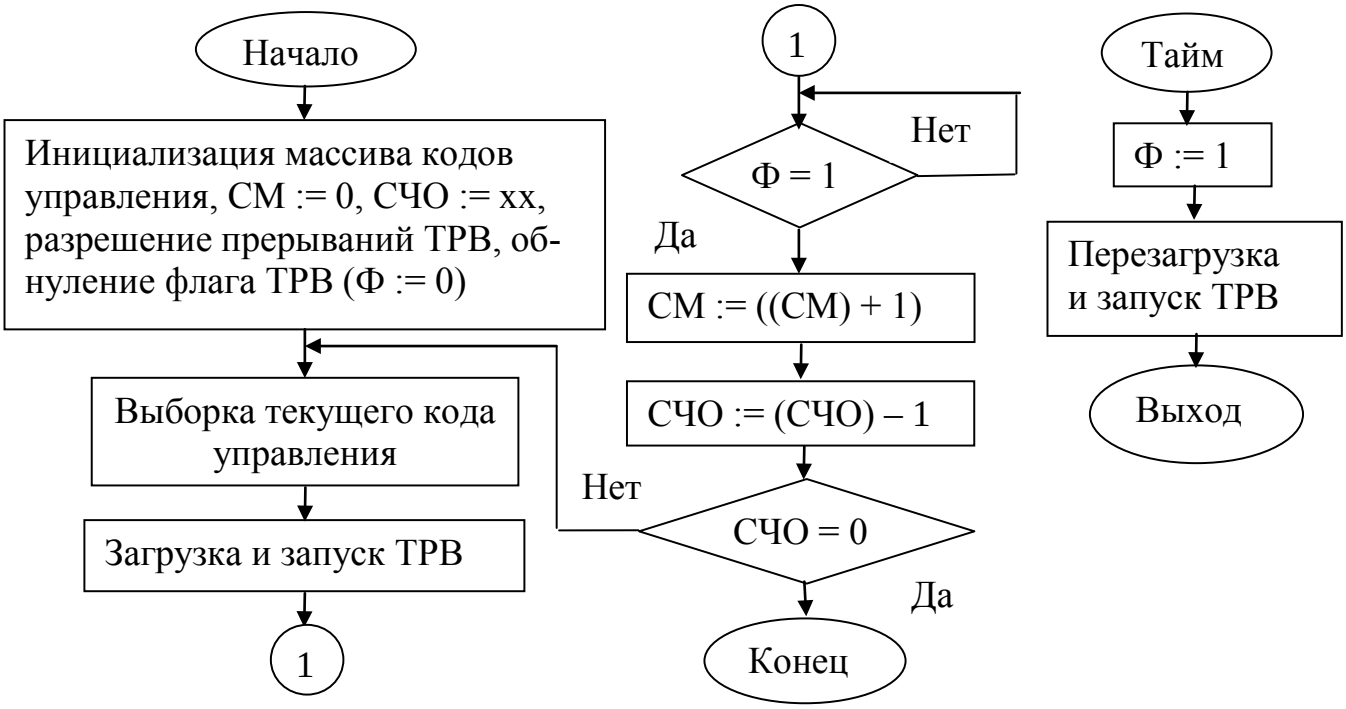

Рис. 31. Алгоритм управления

Количество циклов по заданному числу оборотов двигателя отслеживается в счётчике оборотов (СЧО), при обнулении которого двигатель останавливается. Соответствующая программа управления биполярным шаговым двигателем представлена ниже.

```
.include "m128def.inc"
 ; определение символьных имён
.def temp = r16; регистр временного хранения
.def flags = r17; регистр флага
.def count = r18; счётчик циклов управления
.def off = r19; смещение адреса кода управления
.org 0; вектор сброса
 jmp start
 .org 0x14; вектор прерывания TCNT2
 jmp start
 .org 0x14; вектор прерывания TCNT2
 jmp tim
```

```
.org 0x30
.db 0xf1, 0xf4, 0xf2, 0xf8 ; массив кодов управления
.org 0x40  ; инициализация
start:    ldi temp, 0xff; настройка порта PORTD на вывод
out ddrd, temp
out xdiv, temp; запуск делителя тактовой частоты
ldi temp, low (ramend); установка указателя стека
out spl, temp
ldi temp, high (ramend)
out sph, temp
sei ; разрешение глобального прерывания
ldi count, 0xf0; загрузка счётчика числа циклов
ldi off, 0x0; обнуление смещения адреса кода управления
ldi temp,0x30; загрузка базы массива кодов управления
mov zl, temp
ldi temp,0x0
mov zh, temp
m1: lsl zl
add zl, off; формирование исполнительного адреса выборки
lpm; выборка текущего кода управления в R0
out portd, r0; вывод кода управления
ldi temp,0x30; восстановление базы массива кодов управления
mov zl, temp
inc off; наращивание смещения
andi off, 0x3; округление смещения
ldi temp, 0x40; разрешение локального прерывания от TCNT2
out timsk, temp
ldi temp,0x5; запуск TCNT2
out tccr2, temp
ldi flags, 0; сброс флага выдержки времени
m2:  sbrs flags, 0; цикл ожидания переполнения TCNT2
jmp m2
dec count; декремент счётчика циклов
breq fin; переход на завершение по обнулению счётчика циклов
jmp m1
tim:  ldi temp, 0x0;  перезагрузка и останов TCNT2
out tcnt2, temp
out tccr2, temp
ldi flags, 0x1; установка флага выдержки времени
reti
fin:       jmp fin
```

7.1. Особенности моделирования и отладки в среде ISIS

При отладке программ в среде ISIS наиболее информативным является пошаговый режим, так как в рабочем поле (см. рис. 15) постоянно отображаются заданные разработчиком выпадающие окна среды микроконтроллера и текста программы. Однако при этом возникают проблемы моделирования асинхронных воздействий оператора на элементы моделируемой периферийной среды, например нажатия на кнопки клавиатуры. Как указывалось ранее, для устранения этих невязок рекомендуется в схеме вместо кнопок использовать джамперы. Отлаженные фрагменты программы можно прогонять в режиме останова по контрольным точкам либо в полностью автоматическом режиме. Однако в этом случае окна среды микроконтроллера отображаются только при останове по контрольным точкам либо при вызове команды приостанова анимации.

В процессе моделирования могут наблюдаться сбои при асинхронных воздействиях оператора. Поэтому желательно дублировать отладку программ в интегрированных средах разработки, предоставляемых производителями микропроцессорной техники. Это позволит выяснить причину сбоя, исключив ошибки в программе, и скорректировать процесс моделирования.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Радиоежегодник [Электронный ресурс]. – Электрон. журн. – Вып. 24. PROTEUS по-русски. – 2013. – Режим доступа: www.rlocman.ru.

2. Нестерук, В. Ф. Микропроцессорные системы сбора и первичной обработки сигналов : конспект лекций для дистанционной формы обучения / В. Ф. Нестерук. – Омск : Изд-во ОмГТУ, 2007. – 136 с.

3. Нестерук, В. Ф. Микропроцессорные системы сбора и первичной обработки сигналов : сборник заданий к практическим занятиям для дистанционной формы обучения / В. Ф. Нестерук. – Омск : Изд-во ОмГТУ, 2007. – 96 с.

4. Микропроцессорные системы сбора и первичной обработки сигналов : метод. указания к курсовому проектированию для дистанционной формы обучения / сост. В. Ф. Нестерук. – Омск : Изд-во ОмГТУ, 2007. – 44 с.

5. Нестерук, В. Ф. Цифровые модели для компьютерных технологий обучения основам организации ЭВМ : учеб. пособие / В. Ф. Нестерук. – Омск : Изд-во ОмГТУ, 2001. – 76 с.