ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА» (Самарский университет)

РАЗРАБОТКА И ОТЛАДКА МИКРОПРОЦЕССОРНЫХ УСТРОЙСТВ В ВИРТУАЛЬНОЙ СРЕДЕ МОДЕЛИРОВАНИЯ PROTEUS

С А М А Р А
2017

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА»
(Самарский университет)

РАЗРАБОТКА И ОТЛАДКА МИКРОПРОЦЕССОРНЫХ УСТРОЙСТВ В ВИРТУАЛЬНОЙ СРЕДЕ МОДЕЛИРОВАНИЯ PROTEUS

Составитель *В.Г. Иоффе*

С А М А Р А
Издательство Самарского университета
2017

УДК 004.312
ББК 32.973

Составитель ***В.Г. Иоффе***

Рецензент: к.т.н., доцент А .В. П о л у л е х

Разработка и отладка микропроцессорных устройств в виртуальной среде моделирования Proteus [Электронный ресурс]: метод. указания / сост. *В. Г. Иоффе*. – Самара.: Изд-во Самарского университета, 2017. - Электрон. текстовые и граф. дан. (2,42 Мбайт).- 93 с.:ил. 1 эл. опт. диск (CD-ROM).

Методические указания содержат описание виртуальной среде моделирования Proteus 8.x и технологию её использования при разработке и отладке микропроцессорных устройств на базе однокристальных микроконтроллеров семейств 8051, AVR, ARM Cortex-M3.

Предназначены для бакалавров, обучающихся по направлению 09.03.01 – ” Информатика и вычислительная техника“ в качестве методических указаний к лабораторным работам и курсовому проектированию по курсу «Микропроцессорные средства и системы»

Электронные методические указания разработаны на кафедре информационных систем и технологий.

УДК 004.312
ББК 32.973

ОГЛАВЛЕНИЕ

Введение

Одним из основных элементов современной электроники является однокристальный микроконтроллер ОМК. По оценкам компании Semico Research, в 2010 году каждый житель Земли ежедневно должен был иметь дело с 350 микроконтроллерами, установленными в домашнем и офисном оборудовании, автомобильных системах, а также в устройствах личного пользования.

При проектировании систем на базе ОМК используются различные инструментальные средства, отличающиеся функциональными возможностями и стоимостью: программный симулятор, внутрисхемный эмулятор, оценочный модуль, интегрированная среда разработки. [1,2]

Базовым компонентом инструментальных средств является виртуальная среда моделирования ВСМ.

С помощью ВСМ можно решать следующие задачи:

- выбор элементной базы (ОМК, аналоговых и цифровых микросхем);
- разработка программного обеспечения, а в некоторых случаях - принципиальной электрической схемы и печатной платы;
- отладка программно-аппаратных средств с возможностью контроля как внутренних ресурсов ОМК, так и внешних устройств. При отладке могут использоваться различные виртуальные средства: генераторы, осциллографы, логические анализаторы, терминалы последовательных интерфейсов, средства ввода и отображения информации, аналоговые и цифровые микросхемы и так далее;
- формирование файла, загружаемого в программатор или оценочный модуль;
- документирование проекта.

Применение ВСМ повышает эффективность, качество и гибкость проектирования технических средств на основе ОМК:

- сокращаются сроки и упрощается процедура комплексной отладки разрабатываемых устройств;
- проекты легко адаптируются к изменению аппаратных и программных средств;
- появляется возможность исследовать технические характеристики новых моделей ОМК **до момента их приобретения** и про-

водить сравнительный анализ различных типов микроконтроллеров, что приводит к более обоснованному выбору ОМК для проектируемых устройств;

- отладка в виртуальной среде позволяет избежать целый ряд проблем, которые при работе **на реальном оборудовании** могли бы иметь фатальные последствия;
- снижаются затраты на оборудование, так как в процессе проектирования можно более рационально выбрать элементную базу, не используя макетирование, и не требуются контрольно-испытательные средства;
- некоторые ВСМ содержат средства для проектирования печатных плат;
- в целом, снижается стоимость проектирования.

Особенно полезно использование ВСМ в процессе обучения, так как студенты не обладают опытом проектной работы и навыками практической отладки устройств на основе ОМК.

Однако **полное представление** о процессе проектирования устройств с ОМК может обеспечить только **совместное использование ВСМ и оценочных модулей**.

При выборе ВСМ необходимо учитывать:

- наличие необходимых моделей ОМК и полная имитация их режимов работы;
- возможность модификации содержимого внутренних регистров и памяти в процессе отладки;
- состав библиотек виртуальных компонентов, соответствующих реальным микросхемам, и возможность её расширения;
- количество и возможности эмуляторов внешних устройств;
- поддержку требуемого языка программирования;
- обеспечение совместимости с различными типами компиляторов;
- режимы работы (шаговый, непрерывный, с точками останова, реальное время и так далее);
- наличие средств визуализации основных этапов проектирования;
- (достаточное количество окон отладки, контроль временных диаграмм, построение графиков и так далее);

- возможность разработки электрических принципиальных схем и печатных плат;
- удобство интерфейса пользователя (создание проекта, редактирование, отладка, диагностические сообщения, «всплывающие» окна, документирование результатов проектирования и так далее);
- возможность работы в многопроцессорном режиме;
- наличие средств для проверки качества проектирования (профайлеров);
- полноту документации сопровождения, в которой описана методика работы с симулятором и приведены примеры программирования (желательно на русском языке);
- доступность (наличие бесплатных версий, стоимость).

Большинство доступных ВСМ (симуляторов) предназначены для работы с одним определенным семейством ОМК и поддерживают разработку **только программного обеспечения**: MCS -51/52 (Pinnacle, MCU 8051 IDE, Keil), Atmel (AVRStudio, CodeVisionAVR, VMLab), Motorola (HI-WAVE), MicroChip (MPLAB), MSP 430 (IAR Kickstart) и так далее. В подобных симуляторах виртуальные средства ввода- вывода отсутствуют или очень незначительны.

Исключением является VMLab for AVR, который удовлетворяет большинству требований, сформулированных выше. К его недостаткам следует отнести: ограниченное количеством виртуальных моделей устройств ввода-вывода (в основном, цифровые), не отражающих характеристики реальных компонентов, среда проектирования не позволяет разрабатывать принципиальные электрические схемы.

К ВСМ, работающими с ОМК различных семейств, можно отнести IAR Embedded Workbench (IAR for AVR, IAR for ARM, IAR for 8051 и так далее), графическую среду FlouCode [3,4].

При проектирования радиотехнических устройств **на базе аналоговых и цифровых микросхем наиболее** полно соответствуют перечисленным выше требованиям ВСМ Electronics Workbench, MultiSim, EWB MultiMCU, TINA, которые позволяют разрабатывать электрические принципиальные схемы, печатные платы, отлаживать устройства в режиме реального времени, имеют доста-

точную библиотеку моделей компонентов и виртуальных средств ввода -вывода, хороший интерфейс пользователя. Их общим недостатком является ограниченное число моделей ОМК и терминалов для работы с портами микроконтроллеров (I2C, SPI, USB, паттерн-генераторов и так далее).

Особо хотелось бы отметить систему TINA (фирма Design Soft), которая содержит модели PIC, AVR, 8051, ARM 7, ARM 9 и HCS08 и поддерживает работу с языком описания аппаратных средств VHDL, используемым при программировании микросхем ASIC и FPGA. [5]

Однако при проектировании устройств **на базе ОМК** лучшими характеристиками обладает BCM Proteus Design Suite 8.x фирмы Labcenter Electronics. Это коммерческий пакет программ класса САПР, объединяющий в себе две основных программы: ISIS – средство разработки и отладки в режиме реального времени электронных схем и ARES – средство разработки печатных плат. Proteus позволяет работать с семействами микроконтроллеров MCS-51/52(Atmel, Philips), AVR (Atmel), PIC 10,12,16,18,24 (MicroChip), HC11(Motorola), MSP-430, TMS320 Picolo (TI), ARM7 (Philips), ARM Cortex -M3 (Luminary Micro, NXP Semiconductors), Basic Stamp (Parallax), электронным конструктором Arduino. В процессе развития Proteus число моделей ОМК постоянно увеличивается. [6]

Особенностью Proteus Design Suite 8.x является:

- единая среда для разработки электронных схем, программного обеспечения и печатной платы (включая и 3D моделирование);
- удобство разработки схем на основе ОМК. Разработка проекта начинается с диалога, при котором выбирается микроконтроллер, компилятор и предоставляются шаблоны для разработки электрической схемы и программы;
- единая база данных компонентов обеспечивает обмен данными между модулями Proteus в текущем проекте;
- расширенные средства для работы с документацией к проекту. Программа может быть загружена в виде HEX файла, ASM файла, COF файла, ELF/DWARF2 файла, UBROF файла и т.д., а результаты проектирования сохраняются в популярных форматах PDF, HTML и Excel;

- большое количество различных моделей (десятки тысяч) : аналоговые и цифровые электронные компоненты ведущих фирм, периферийные устройства (светодиодные и ЖК индикаторы, температурные датчики, часы реального времени – RTC), интерактивные элементы ввода-вывода: кнопки, переключатели, клавиатуры, двигатели различных типов, нагреватели, реле, виртуальные терминалы (UART, SPI, I2C, USB), виртуальные измерительные приборы (13 видов) и генераторы (15 видов), возможность построения различных интерактивных графиков и так далее. Такой набор моделей отсутствует в других подобных программах;
- виртуальные модели разработаны на основе программы моделировании ProSPICE и позволяют в процессе проектирования изменять свойства существующих моделей и разрабатывать новые;
- возможность работы в многопроцессорном режиме;
- большое количество примеров проектирования устройств на основе ОМК различных типов.

Недостатком Proteus Design Suite 8.x является относительно высокая стоимость.

В лабораторных работах и при курсовом проектировании используется лицензионная версия Proteus Design Suite 8.x с усеченными возможностями – VSM for AVR, 8051/8052, ARM Cortex-M3.

Пользователю доступны средства создания и отладки проектов, библиотека компонентов, ОМК AVR, 8051/8052, ARM Cortex-M3, виртуальные приборы. К сожалению, используемая версия не поддерживает работу с графиками.

Методические указания разработаны для студентов специальности 230100.62 «Информатика и вычислительная техника», изучающих курс «Микропроцессорные средства и системы».

Студенты этой специальностей достаточно хорошо владеют ресурсами компьютера и имеют опыт работы с программными системами.

Целью методических указаний является не описание возможностей Протеуса, а использование этой системы для изучения принципов организации ОМК, приобретения навыков разработки и отладки микропроцессорных устройств, выполнения лабораторных работ и курсового проектирования.

В Интернете достаточно много материалов, содержащих описание Proteus. Наиболее полная версия приведена в [6]. При написании методических указаний использовались так же материалы, опубликованные на сайте фирмы Labcenter Electronics

Поэтому основное внимание в методических указаниях уделяется ресурсам Proteus, необходимым для создания и отладки устройств на основе ОМК.

Основные этапы работы с Proteus состоят в следующем:

1. Выбрать тип ОМК.
2. Определить язык программирования и соответствующий компилятор.
3. Разработать принципиальную электрическую схему и разместить на ней виртуальные приборы, необходимые при отладке.
4. Разработать программное обеспечение.
5. Скомпилировать проект.
6. Выполнить отладку.
7. Оформить отчетную документацию.

1. Особенности разработки микропроцессорных устройств

Работа с Proteus начинается с окна, представленного на рис. 1. В окне Start имеется возможность открыть проект, создать новый проект, открыть проект, выполненный в более ранних версиях, открыть примеры проектирования, открыть последний проект. Назначение других окон следует из их названия.

Начало работы с новым проектом выполняется на основе следующего диалога:

1. Выбрать «New Project».
2. Присвоить имя проекту и создать папку для работы с новым проектом.
3. Выбрать размер поля для электрической схемы.
4. Выбрать семейство ОМК, модель и компилятор.
5. Проверить введенные данные и завершить настройку проекта.
6. После этого будут созданы заготовки для разработки электрической схемы- «Schematic Capture» и программы - «Source Code».

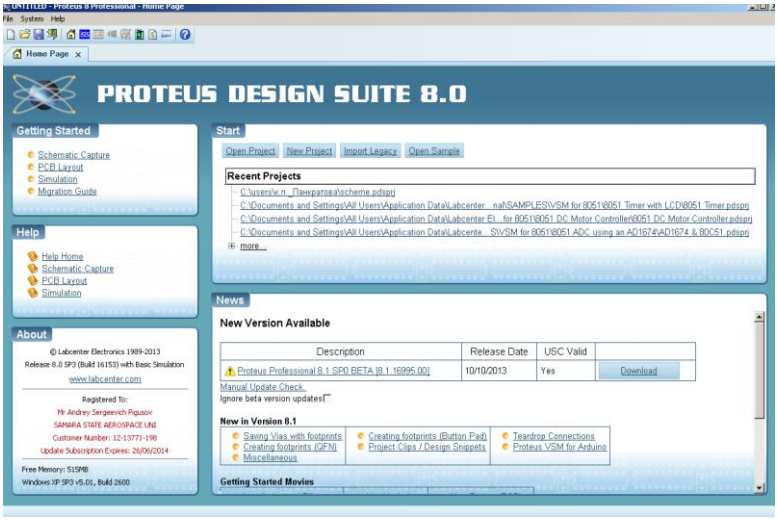


Рис. 1. Начальное окно Proteus

1.1. Средства разработки электрической схемы

Разработку проекта следует начинать с электрической схемы (Рис. 2). При открытии «Schematic Capture» на ней размещается выбранный микроконтроллер. Затем следует разместить в этом окне остальные необходимые компоненты. Пример размещения проекта и назначение отдельных полей показан на рисунке 3. **[ ]**

Верхний горизонтальный набор кнопок **при включении настраивается на режим, при котором можно создавать и отлаживать проект.** Особенности настройки этих окон будут изложены позднее.

Основные действия при создании электрической схемы выполняются с помощью левого вертикального набора кнопок, которые позволяют выбрать необходимые компоненты и соединить их соответствующим образом.

Все кнопки имеют контекстно чувствительную подсказку и при наведении курсора всплывает их название:

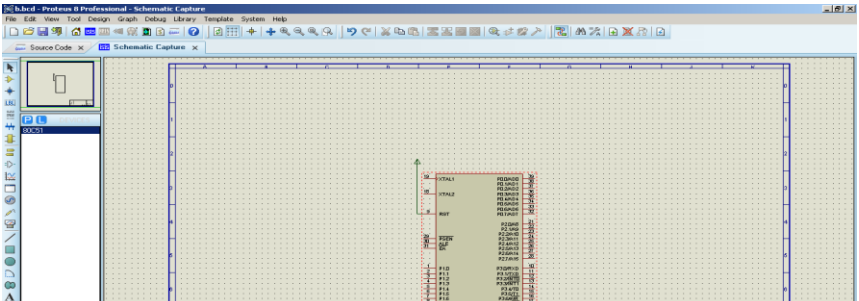


Рис. 2. Окно для разработки электрической схемы

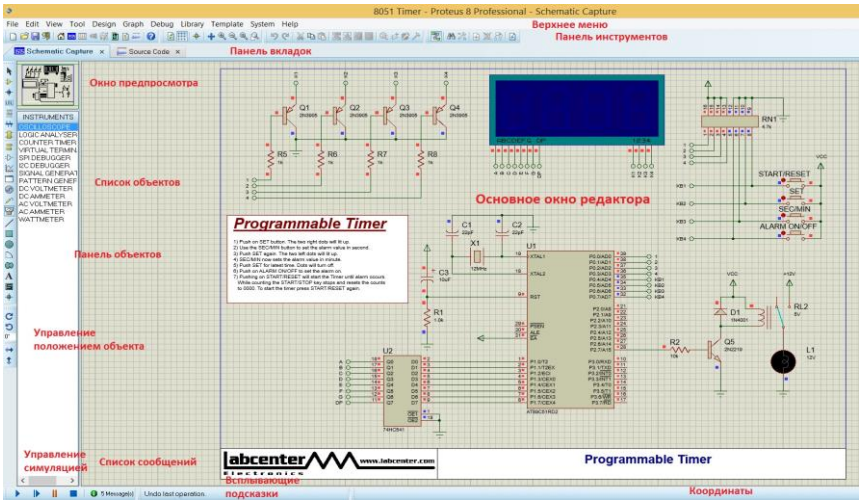


Рис. 3. Структура рабочего поля

Selection Mode – (жирная черная косая стрелка-указатель) – режим выбора. В этом режиме в окне редактирования единичным щелчком левой кнопки мыши по объекту (компоненту, проводу, шине, графическому элементу) можно выделить его – он становиться красным, а удерживая левую кнопку нажатой и обводя группу объектов можно выделить блок. Кроме того, в этом режима возможно проведение соединительных линий между выводами компонентов или шин. В окне предпросмотра при этом виден уменьшенный лист проекта (синяя рамка) и положение текущего окна редактирования (зеленая рамка).

Component Mode – (кнопка с изображением желтой мнемоники операционного усилителя) – режим выбора/размещения компонентов. В этом режиме компоненты из селектора объектов размещаются в окне редактирования. При выборе требуемого компонента в селекторе его вид отображается в окне предпросмотра. С помощью круговых стрелок **(Rotate)** можно выбрать в каком положении будет размещаться объект на поле в окне редактирования. Это положение будет отражено в окне предпросмотра. В режиме **Component Mode** первый щелчок левой кнопкой мыши по полю окна редактирования вызывает подсветку контура размещаемого объекта, а второй щелчок устанавливает его на выбранное место. Также, как и в предыдущем режиме доступно проведение проводов между выводами компонентов.

Junction Dot Mode – (прицел с синим квадратиком) – режим расстановки точек соединения на проводах. Расстановка, как и в предыдущем режиме за два щелчка мыши: подсветка, установка.

Wire Lebel mode – (кнопка LBL) – режим текстовой маркировки проводов и шин в проекте при наведении курсора на маркируемый провод/шину под изображением карандаша появляется косое перекрестие, после чего щелчок мышью вызывает окно редактирования **Edit Wire Lebel**. В окне **String** проводнику присваивается уникальное в рамках проекта имя, либо выбирается из уже имеющихся через раскрывающийся список по стрелке справа от окна **String.**

Маркировка должна быть уникальной и не содержать русских символов, только латинские буквы и цифры.

Text Script Mode – (горизонтальные пунктиры, изображающие текст) – режим размещения текстовых скриптов (простых многострочных текстов). Щелчок по свободному полю в проекте вызывает всплывающее окно встроенного редактора текста **Edit Script Block** (рис. 4). В окне **Text** набираем текстовый блок. Допустим импорт текста из текстовых файлов или наоборот экспорт через соответствующие кнопки внизу справа. Переключателями **Rotation, Justification** выбирается расположение/ориентация текста в проекте. Особенности шрифта редактируются во вкладке **Style.**

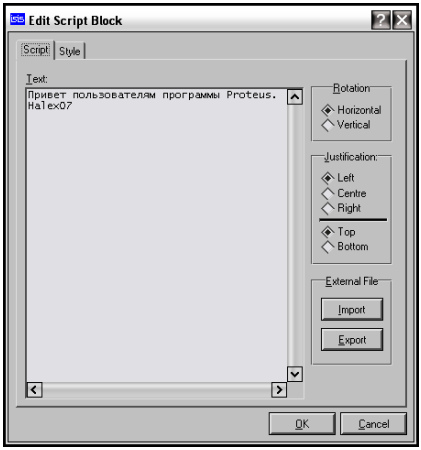


Рис. 4 Окно Edit Script Block

Buses Mode – (горизонтальная синяя шина с отводами вверх/вниз) – режим рисования соединительных шин. Первым щелчком левой кнопкой мыши в нужной точке проекта задают начало шины, последующими одиночными ставим точки поворотов, двойным щелчком завершают рисование. Если **Wire Autorouter** в верхнем меню выключен, шины можно протягивать не только под прямым углом, но и наискось.

Subcircuit Mode – (желтый прямоугольник с выводами справа и слева) – режим размещения субмодулей. Модули – прямоугольники с толстой синей окантовкой и заливкой цветом компонентов– позволяют в Протеусе вынести функционально законченные узлы на отдельные листы (**Child Sheet** - дочерний лист модуля). Модуль рисуется удержанием нажатой левой кнопки мыши в окне проекта по диагонали с угла на угол. После чего через изменение свойств доступно присвоение ему индивидуального имени (по умолчанию подставляется **SUB?**). При установке **Subcircuit Mode** в окне селектора становятся доступными для выбора терминалы (порты ввода/вывода и питания) субмодуля. Расстановка выбранных терминалов возможна по левой и правой вертикальной синей окантовке модуля. По неписаным канонам принято входы **(Input)** ставить слева, а выходы **(Output)** справа. Изображение терминала появляется в окне предпросмотра при выборе его в селекторе.

Terminal Mode – (два желтых горизонтальных указателя вправо/влево) – режим расстановки терминалов. **Терминалы позволяют связать две или несколько точек схемы, расположенных как на одном листе, так и на разных листах, не проводя между ними соединительной линии.** Для этого в свойствах **(Properties)** связанных между собой терминалов им указывают одинаковые имена. Имена указываются в окне **String** вручную или выбираются из уже назначенных через выпадающее меню при щелчке по стрелке справа в окне **String.** Окно свойств открывается двойным щелчком по установленному в схему терминалу, либо через правую кнопку мыши, выбрав опцию **Edit Properties (Ctrl+E)**. Выбранный в селекторе терминал перед установкой отображается в окне предпросмотра. Его положение можно изменять кнопками поворота изображения. Еще одно важное замечание: выбор **Default, Output, Input и Bidir** влияет только на изображение терминала на схеме. Симулятору ISIS абсолютно безразлично какой из этих терминалов установлен – одноименные терминалы способны пропускать сигнал в любом направлении.

Если терминалы Power и Ground после установки не поименованы особо, то они считаются подключенными к глобальным для проекта питанию и земле.

Наименование **терминала BUS** формируется следующим образом: **ИМЯ[НАЧ_РАЗРЯД..КОН_РАЗРЯД]. ИМЯ** – наименование шины латиницей без пробелов и спецсимволов, **НАЧ_РАЗРЯД и КОН_РАЗРЯД** - два числа, из непрерывного возрастающего ряда, определяющие разрядность данного терминала. С**кобки обязательно должны быть квадратные и между начальным и конечным разрядом ставится ДВЕ, а не три точки.**

Допустим, есть шина адреса с именем (лэйблом) **A[0..15].** Поместив на ее конце, или отводе терминал с именем **A[0..15]** получим на одноименном терминале шины в другом месте схемы все 16 разрядов сигнала, которые через отходящие от шины провода с именами **A0, A1** и т.д. можно развести по компонентам. Если же разместить на отводе **A[0..15]** терминал **A[8..11],** то поместив терминал с таким же именем в другом месте, получим на нем только четыре выделенных разряда **A8, A9, A10 и A11**, которые могут использоваться для соединения с компонентами.

Device Pins Mode – (расчлененное изображение операционного усилителя на сером фоне) – режим расстановки выводов модели устройства (компонента) при его создании.

Graph Mode – (две оси координат с синусоидами) – режим размещения графиков в проекте.

Возможные варианты анализа с помощью графиков выбираются в окне селектора: **ANALOGUE, DIGITAL** и т. д. Окно соответствующего графика растягивается по полю проекта зажатой левой кнопкой мыши диагонально.

В используемой версии эти функции отсутствуют.

Tape Recorder Mode – (изображение магнитофонной кассеты) – режим установки магнитофона.

Сигналы, генерируемые разработанным устройством, в процессе симуляции можно записать в файл с последующим использованием их в другом проекте.

Generator Mode (синусоида в окружности) – режим расстановки виртуальных генераторов. В этом режиме в окне селектора доступны для выбора и размещения в схеме виртуальные генераторы сигналов.

Условно их можно разделить на цифровые (все начинающиеся с буквы **D, за исключением DC** – постоянный потенциал) и аналоговые (все оставшиеся):

DC — источник постоянного напряжения.

Sine — генератор синусоидального напряжения с управляемыми амплитудой, частотой и фазой.

Pulse — аналоговый импульсный генератор с управляемыми амплитудой, периодом и временем фронта и среза.

Exp — экспоненциальный импульсный генератор; производит импульсы такого же вида, какой имеют RC цепи при заряде/разряде.

SFFM — частотно-модулированный одночастотный генератор — производит сигнал, определяемый частотной модуляцией одного синусоидального сигнала другим.

Pwlin — генератор сигнала, аппроксимируемого кусочно- линейной функцией. Используется для создания сигналов произвольной формы.

File — как в **Pwlin**, но данные берутся из ASCII файла.

Audio — используются **Windows WAV** файлы для входных сигналов. Особый интерес представляет комбинация с графиком **Audio**, так как позволяет прослушать полученный от схемы сигнал на аудио выходе компьютера.

DState — установившийся логический уровень.

DEdge — формирователь **однократного** перехода из 0 в 1 или наоборот.

DPulse —формирователь **однократного** цифрового импульса с заданными параметрами.

DClock — генератор цифрового тактового сигнала.

DPattern — генератор произвольной цифровой последовательности.

Easy HDL - программируемый генератор, для которого предварительно надо написать программу на встроенном в Протеус языке Easy HDL.

Установка выбранного генератора осуществляется на свободное место схемы двумя последовательными щелчками или сразу на провод, после чего в свойствах ему задаются требуемые параметры. В окне предпросмотра видно изображение генератора. При последующем подключении к выводу компонента или проводу генератор автоматически изменит свое имя на имя соответствующего ближайшего вывода. Через окно свойств генератора его можно переименовать по своему усмотрению.

Окно свойств генератора вызывается, если мышь подвести к изображению генератора на схеме до появления кисти с указательным пальцем и нажать правую клавишу. В окне устанавливаются основные параметры выбранного генератора.

Установка параметров генераторов не должна вызвать трудности. Исключением является генератор произвольной последовательности DPattern, параметрами которого являются:

Initial State - значение в нулевой момент времени

First Edge - время старта, при котором начинается формирование требуемой последовательности. До этого момента выходной сигнал будет определяться значением Initial State;

Timing - шаг выходного сигнала **DPattern** может быть не зависящим от уровня сигнала **(Equal mark/space timing)** или устанавливаться различным для логической единицы **(Pulse)** и нуля **(Space Time)**.

Transitions – выходной сигнал может быть настроен на циклическое повторение сформированной последовательности до окончания симуляции, или на фиксированное число повторений, определяемое количеством переходов между логическими состояниями (**Specific Number of Edges)**

Bit Pattern – по умолчанию генератор формирует последовательность чередующихся высоких и низких уровней. Специальный режим **Bit Pattern (Specific pulse train)** позволяет задать строку-шаблон, которая может содержать следующие символы:

– 0, L – выходной сигнал имеет стабильный низкий уровень; («L» в верхнем регистре).

– 1, H – стабильный высокий уровень; («H» в верхнем регистре).

– l – слабый низкий уровень;

– h – слабый высокий уровень;

– F,f – «плавающий» уровень.

Script — генератор будет управляться скриптом DIGITAL BASIC. Генератор получает доступ к скрипту при объявлении переменной PIN с тем же именем, что и ссылка генератора.

Probe Mode (желтый щуп) – режимы расстановки пробников напряжения и тока на провода схемы. **Пробники ставятся на провода, а не на выводы компонентов** и аналогично генераторам автоматически изменяют свои имена. Если установить пробник на пустое место, он вместо имени высветит знак вопроса.

В цифровых цепях установка токовых пробников бессмысленна, а пробники напряжения на этих цепях будут индицировать не значение напряжения, а логический уровень сигнала.

Стрелка в кружке **токовых пробников** –должна быть ориентирована вдоль провода. Если направление тока в проводе совпадает, значение тока при симуляции будет положительным, если нет – отрицательным.

Virtual Instruments Mode – (кнопка с изображением стрелочного прибора*)* – режим выбора и размещения виртуальных инструментов в проекте.

В Протеусе имеется обширный набор виртуальных приборов: вольтметр, амперметр, четырехканальный осциллограф, счетчик/частотомер, генератор сигналов, логический анализатор, тер-

миналы и так далее. В этом режиме осуществляется их выбор и размещение в проекте.

Большинство из них, за исключением вольтметров/амперметров, имеют однополюсное подключение. Это означает, что измерение осуществляется **относительно земляного провода.**

Описание виртуальных приборов приведено в приложении А.

Дополнительную информацию о подключении и работе с виртуальными приборами можно получить из примеров, приведенных в папке **«Sample».**

Необходимо помнить, что виртуальные приборы требуют определенное количество ресурсов компьютера и в сложных проектах могут стать причиной тормоза симуляции в режиме реального времени.

Остальные кнопки левого набора относятся к режиму **2D** (двухмерной) графики. Их мнемоника и назначения соответствуют большинству графических редакторов.

Перечисленный набор кнопок (меню) можно вызвать также, если щёлкнуть правой клавишей мышки на свободном месте окна редактора и выбрать из выпадающего меню Place, а затем нужную функцию.

1.2. Разработка электрической схемы

Выберите кнопку **Component Mode**, а затем щёлкните по букве **«P»** в селекторе объектов. Появится окно **Pick Devices**, с помощью которого осуществляется доступ к библиотеке компонентов (рис. 5).

Поиск необходимых компонентов можно выполнять по ключевому слову(**Keywords**), категории(**Category)**, подкатегории(**Subcategory**), фирме производителя(**Manufacturer**). Для извлечения компонента из библиотеки его нужно выделить из списка и двойным щелчком левой кнопки мыши поместить в селектор. При этом в правой части окна отображается условное изображение, тип корпуса и геометрические размеры, необходимые при разработке печатной платы, наличие модели.

После выбора всех компонентов закрывают окно **Pick Devices** и переходят к размещению на схеме генераторов и виртуальных приборов.

Если необходимо изменить размещение компонентов, поместите курсор мышки на компонент, щёлкните и, когда рядом с курсором появится **перекрестие**, нажмите и, удерживая левую клавишу мышки, перенесите компонент в новое место. Окно ориентации компонента и редактирования его свойств откроется после наведения курсора на компонент и нажатия правой кнопки мыши.

Для соединения компонентов необходимо в разделе **Tool** основного меню установить опцию **Wire Autorouter**. Эта опция выбрана, если «нажата» иконка слева в меню. **При включении этот режим устанавливается по умолчанию**. Когда выбрана опция **Wire Autorouter**, соединение между двумя точками выполняется, обходя препятствия.

Курсор мышки подводится к концу вывода компонента. При появлении красного квадратика на конце вывода, курсор становится похожим на карандаш. Щёлкните левой клавишей мышки, чтобы начать соединение, а затем перемещайте мышку к выводу, к которому требуется провести провод. Когда на этом выводе появится красный квадратик, щёлкните левой клавишей мышки ещё раз, чтобы завершить соединение.

Если этой функцией не пользоваться, то при проведении провода следует привязку самих проводов и поворотов в нужных местах фиксировать одиночными щелчками левой кнопкой.

При большом количестве соединений следует использовать терминалы или шины **(Terminal Mode).**

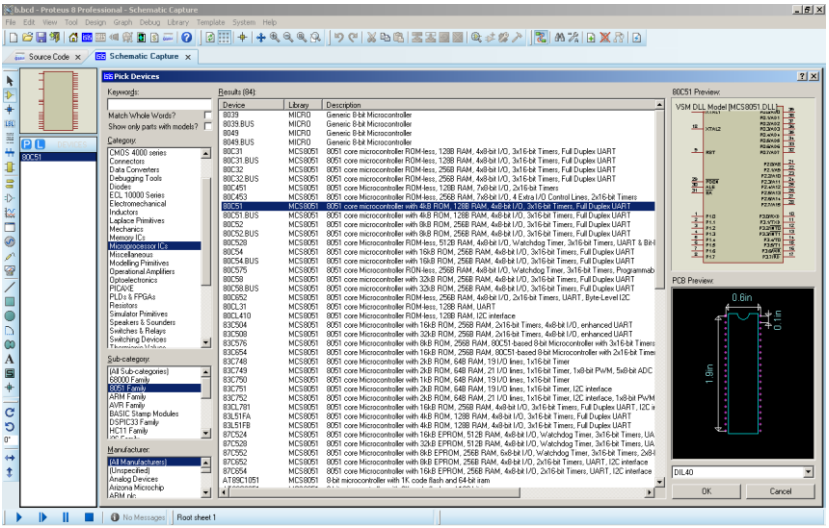


Рис. 5. Окно выбора компонентов

Работа с электрической схемой завершается корректировкой свойств используемых компонентов. Для этого следует курсор мыши навести на компонент и выполнить двойной щелчок левой кнопкой мыши. При открытии окна свойств компонента устанавливаются его режимы работы. Настройка свойств виртуальных приборов приведена в Приложении А.

Настройка ОМК требует выбора следующих параметров:

- **8051**
  - Clock frequency – частота работы ОМК
  - Enable trace logging – включить журнал трассировки
  - Simulate program fetches – симуляция программного считывания, выборка, вызов
  - Code memory map – карта памяти программ
  - Data memory map – карта памяти данных
- **AVR**
  - WDTON (Watchdog timer always on) – Сторожевой таймер всегда включен
  - CKOPT (oscillator options) – Настройки генератора
  - BOOTRST (Select Reset Vector) – Выбор вектора сброса

- CKSEL Fuses – Определение режима работы тактового генератора
- Boot Loader Size – Размер загрузчика
- SUT Fuses – Определение задержки после старта
- Clock Frequency – Частота системного генератора
- Initial Component of data EEPROM – Инициализация компонента памяти данных EEPROM
- Disassemble Binary Code – Дизассемблировать двоичный код

- **ARM Cortex –M3**
  - Crystal frequency - Частота ядра
  - Disassemble Binary Code – Дизассемблировать двоичный код

С помощью кнопки **HELP,** можно получить информацию об особенностях моделей используемого компонента.

Выбор частоты системного генератора определяется задачами, которые решает ОМК, требуемыми быстродействием и энергопотреблением.

Если необходимо выполнять преобразование или формирование частотно-временных сигналов с высокой точностью, то следует использовать **внешний кварцевый резонатор,** так как стабильность внутренних генераторов составляет (2-4) %.

Применение **внутренних генераторов** снижает аппаратные затраты и упрощает конструкцию печатной платы.

Частота генератора должна быть достаточна для реализации технического задания, но при этом необходимо учитывать, что с увеличение частоты растет энергопотребление.

Однако использование внешнего кварцевого резонатора и неоправданное увеличение частоты приводит к повышенной загрузке процессора ПЭВМ в процессе моделирования, что затрудняет реализацию режима реального времени.

Частота ОМК может быть задана в Гц (**Hz**), КГц(**kHz**), МГц(**M**), ГГц(**G.**). При задании частоты в Гц размерность можно не указывать. **При установке дробного значения целая и дробная части разделяются точкой. Например, 4.092kHz.**

Номиналы любых компонентов в Proteus задаются аналогичным образом. Если номинал указывается просто числом, то в зави-

симости от типа компонента и задаваемого параметра оно будет соответствовать основной единице измерения данной величины. Возможно и буквенное обозначение физической величины: **V** – Вольт, **A**- Ампер, **Ohm** (или **R**) – Ом, **F** – Фарада, **H** – Генри, **S** – секунда, – Герцы - **Hz.** Производные величин: нано – **n**, микро – **u**, мили – **m**, кило – **k**, мега – **M**, гига – **G**.

1.3 Разработка программного обеспечения

После завершения электрической схемы необходимо открыть окно **Source Code.** В зависимости от типа компилятора, выбранного при создании проекта, открывается шаблон для написания программы. Вид окна при программировании на ассемблере приведен на рис.6.

Программе присваивается имя, она сохраняется, компилируется (**Build**). После исправления ошибок, сообщение о которых формируется в нижней части окна (**VSM Studio Output**), повторно выполнить компиляцию, сохранить проект и приступить к отладке.

Начальный адрес программы пользователя (метка Start) выбирают за пределами зоны векторов прерываний, которые в 8051/8052 и AVR **по умолчанию** располагаются в младших адресах памяти программ.

Принципы программирования и основные директивы ассемблера изложены для 8051 в [7,8], а для AVR в [9,12,13]

В этом же окне можно запустить процесс отладки (см. нижнюю строку кнопок на рис. 6).

Если программа разрабатывается на СИ, то при выборе компилятора предлагаются бесплатные версии SDCC for 8051, WinAVR, GCC for ARM, коммерческие версии IAR for 8051, Keil for 8051, Code Vision AVR, IAR for ARM, Keil for ARM. В принципе может использоваться любая среда разработки, на выходе которой формируются файлы с расширениями, поддерживаемыми Proteus (HEX, ASM, COF, ELF/DWARF2, UBROF и т.д).

Недостатком HEX-файлов является сложность отладки в Proteus, так как не выполняется дизассемблирование программы, а результаты отладки можно наблюдать в соответствующих окнах без привязки к тексту программы.

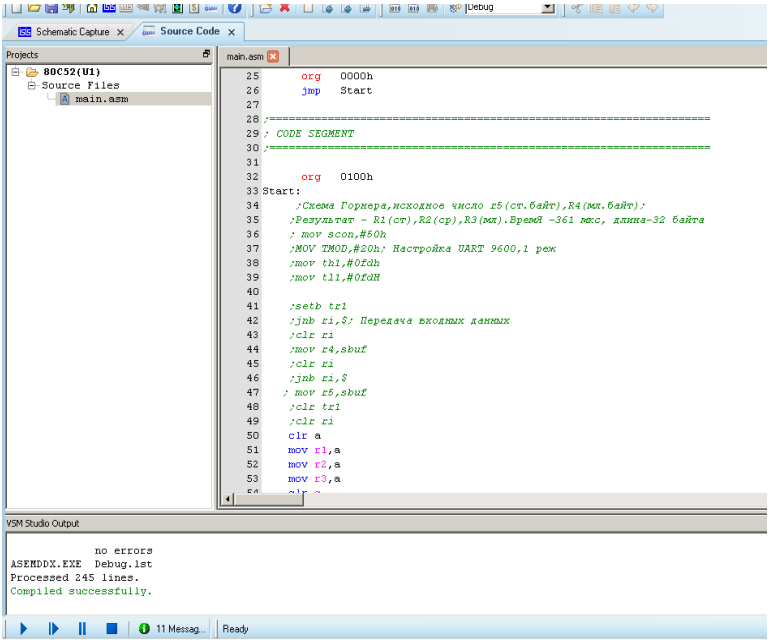


Рис. 6. Окно Source Code

Более рационально применение файлов с расширениями ASM, COF, ELF/DWARF2, UBROF, которые обеспечивают отладку с полным сервисом. (рис.7)

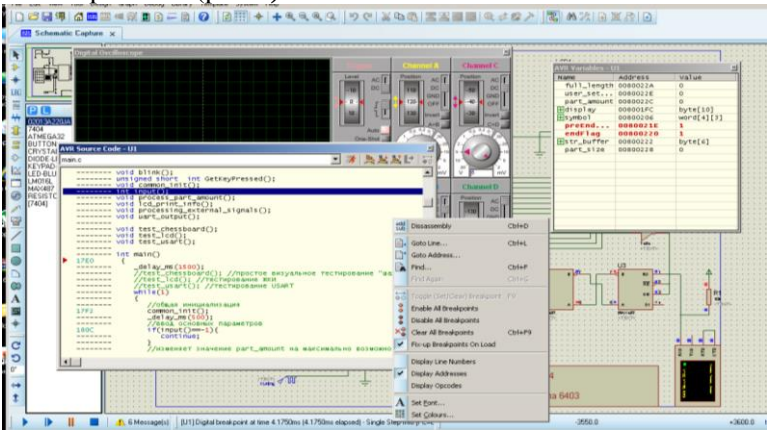


Рис.7. Окна отладки при программировании на СИ

При необходимости операторы СИ могут быть развернуты в ассемблерный код. (рис. 8). Для этого следует щелчком правой кнопки мыши в поле **Source Code в режиме паузы** вызвать контекстное меню (среднее окно на рис.7) и установить в нем функцию **Dissassembly.**

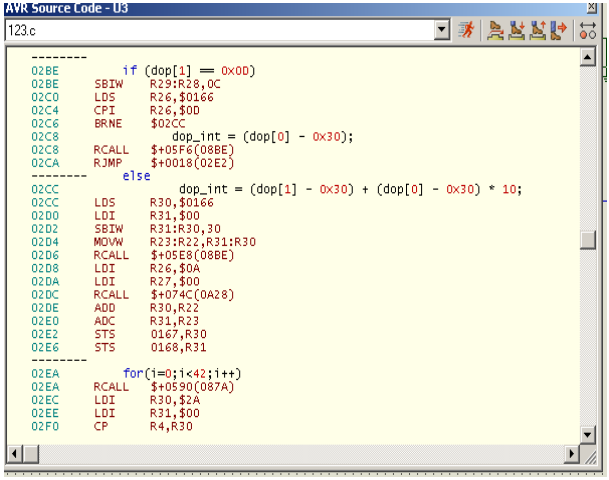


Рис. 8. Дизассемлирование программы

2. Отладка микропроцессорных устройства

Отладка начинается с обращения к верхней строке меню – **Debug** (рис. 9) и последующего запуска отладки - **Start VSM Debugging**. При повторном вызове **Debug** откроется меню, позволяющее установить режим отладки и выбрать необходимые окна.

Меню Debug открывается только во время паузы.

Режим отладки задается в меню **Debug** или в окне **Source Code**:

Run simulation - запуск симуляции в непрерывном режиме и её продолжение после паузы или с точки останова (**Breakpoint**). Точки останова позволяют прерывать выполнение программы в строго указанных местах.

Run simulation (no breakpoints) - вызывает запуск симуляции с игнорированием установленных точек останова.

Run simulation (timed breakpoints) - вызывает окно, в котором предварительно задается время для перевода симуляции в режим паузы. По умолчанию предлагается одна секунда.

Step Over Source Line, **Step Into Source Line**, **Step Out from Source Line** -позволяют выполнить код пошагово соответственно до конца текущей строки, в текущей строке, с выходом из текущей строки.

Run To Source Line -выполнить код до текущей строки**. Неактивная кнопка** станет доступной, если выделить, щелкнув левой кнопкой мыши, какую- либо строку кода программы. При этом курсор текущей выполняемой строки остается на месте, а подсветка переместится на выбранную строку. Запуск кнопкой **Run To Source Line** вызовет выполнение программы до достижения выделенной строки.

Переключатель точек останова располагается в окне **Source Code** (верхняя правая позиция)**.**

Для задания точки останова необходимо выделить требуемую строку программы и нажать красную кнопку поля переключения, отмена- белая кнопка.

Точки останова можно установить **непосредственно в тексте программы**. Для этого с внешней стороны номера строки следует выполнить двойной щелчок левой кнопкой мыши, отмена -такие же действия.

Активная точка останова выглядит как закрашенный красный круг, неактивная - как красная окружность с белой серединой.

Animate Single Step – запускает симуляцию в непрерывном режиме, но **переключение строк замедленное.** В окне **Source Code** подсветка строки пошагово показывает процесс выполнения, а результаты можно наблюдать в открытых окнах.

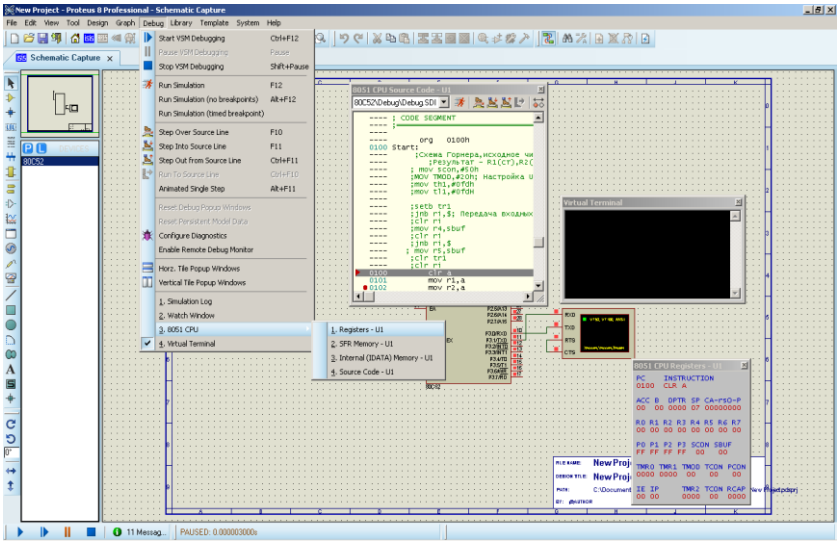


Рис. 9. Окно начала отладки

Reset Popup Windows и **Reset Persistent Model Data** на картинке неактивны. Они позволяют сбросить в исходное состояние соответственно всплывающие окна и **данные** в ПЗУ (**но не в памяти программ**) моделей микроконтроллеров, флэш-памяти и других устройств, содержащих ПЗУ. **Это возможно только когда симуляция не запущена**.

Configure Diagnostics - эта команда вызывает окно конфигурирования диагностических сообщений **Simulation Log** (рис. 10). Опция **Configure Diagnostic** доступна как в режиме останова, так и при запущенной симуляции. Более подробно функции этой команды будут рассмотрены далее.

Horiz. Tile Popup Windows и **Vertical Tile Popup Windows** выстраивают находящиеся на экране всплывающие окна соответственно горизонтально и вертикально.

Нижняя часть меню Debug позволят управлять размещением окон, необходимых при отладке, и их содержанием.

Simulation Log – окно, содержащее информацию о ходе процесса симуляции: сообщения о выполнении определенных действий устройства, ошибках симулятора и моделей.

По умолчанию вся диагностика стоит в режиме **Warnings Only** (только предупреждения), что индицируется желтыми треугольниками с восклицательными знаками, лог ведется в течение одной минуты (время внизу от 0 до 60 сек), в непрерывном режиме.

Для того, чтобы увидеть конкретный перечень формируемых диагностических сообщений, необходимо щелкнуть по плюсу слева от нужного компонента. На рисунке 8 свернуты сообщения самого симулятора **PROSPICE** и развернуты для микроконтроллера. Чтобы не перегружать окно **Simulation Log** излишней информацией, время сбора информации корректируется в строке **Log from … for… seconds**.

Форму оповещения можно изменить для каждого типа сообщений: **Disabled-** отключена, **Warnings Only-** только предупреждения, **Full Trace** -полная трассировка, **Debug-** отладка.

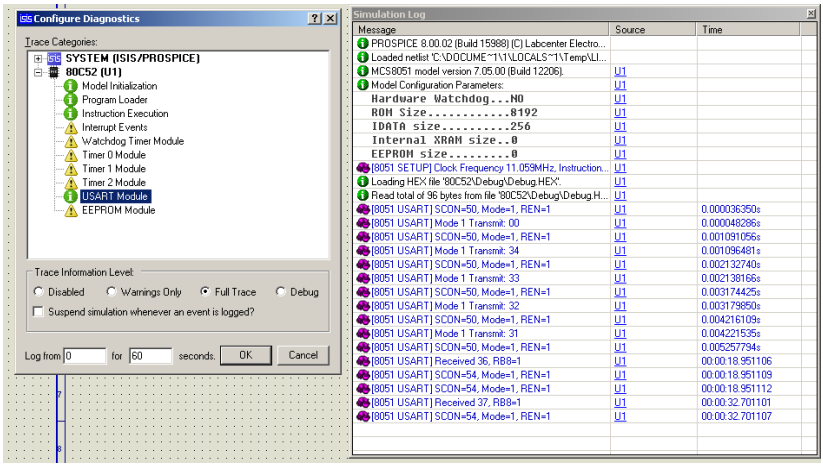


Рис. 10. Окна конфигурирования диагностики и протокол процесса симуляции

Приостановить симуляцию до наступления определенного события можно, установив в строке **Suspend simulation whenever an event is logged?** галочку. Этот режим может быть полезен при ручном вводе параметров в процессе отладки устройства. В непрерывном режиме эта функция должна быть заблокирована.

На рис. 10 окно **Simulation Log** соответствует режиму, установленному в **Configure Diagnostics.** В этой задаче необходимо было передать из UART в терминал пять байт, принять в UART два

байта и выполнить обработку. Более подробно контролируется вывод-ввод данных через UART с указанием времени выполнения команд записи-чтения.

Используя данные, приведенные в колонке Time, можно вычислить время выполнения определенного участка программы или отдельной команды

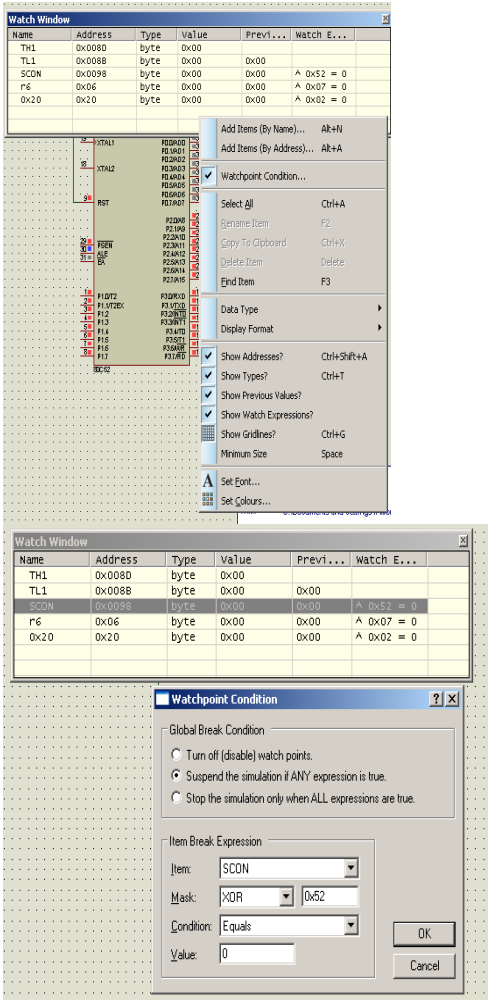
Следует обратить внимание на появление синих вопросительных знаков. При щелчке мышью по ним доступны всплывающие подсказки со ссылками на разделы **Help**.

При отладке программы наиболее важные параметры целесообразно хранить в окне **Watch Window. Его применение позволяет контролировать значения параметров в режиме реального времени (остальные окна отображают информацию только в режиме паузы).**

Для записи параметров необходимо вызвать это окно, в текущей строке щелкнуть правой кнопкой мыши. После этого откроется контекстное меню (рис. 11). Первые две строчки позволяют записать требуемый параметр по имени **(By name)** или по адресу **(By adress).**

При вызове **By name** откроется меню, из которого можно выбрать любой программно-доступный регистр микроконтроллера, а **By address -** открывает окно, в котором указывается имя и адрес ячейки **резидентной памяти данных.** В контекстных меню можно также задать тип, формат записываемых данных и ряд других параметров (рис. 11).

Например, по умолчанию окно **Watch Window** состоит из двух столбцов. Для повышения информативности можно добавить столбцы адреса, типа, предыдущего значения (**Previous Value),** наблюдаемого (контролируемого) выражения (условия) **(Watch Expressions)** для функции **Watchpoint Condition.**



а) б)

Рис. 11. 0кна слежения и точек останова

Эта функция позволяет остановить симуляцию по достижению внутренним регистром **определенного значения.** Условием останова может быть, как значение одного бита (например, флага готовности в регистре управления, флага в регистре состояния), так и

значение байта или слова (например, проверка допустимого значения).

На рис. 11 задана остановка симуляции по флагу готовности TI, находящегося в управляющем слове SCON последовательного порта.

После вызова функции **Watchpoint Condition в** окне **Global Break Condition** (глобальное условие точки останова) следует установить переключатель **Suspend the simulation if ANY expression is true** (приостановить симуляцию, если **ЛЮБОЕ** из выражений истинно). Верхнее положение **Turn Off** соответственно отключает остановку по условию, а нижнее - **… only when ALL…** - останавливает по совпадению сразу **всех** условий, записанных в окне **Watch Window**.

При выбранной строке **SCON в** поле **Item** автоматически записывается имя регистра, а значение условия останова формируется в строках **Mask (**маскирование), **Condition (**Условие), **Value** (Значение). Значение SCON при установленном флаг готовности и заданном режиме равно 0x52. Поэтому установки на рисунке 9 реализуют выражение (SCON)xor(0x52)=0.

После нажатия **ОК** условие записывается в столбец **Watch Expressions** окна **Watch Window.**

Чтобы отменить точку останова, надо снова зайти в **Watchpoint Condition** и задать функцию **Turn Off…**.

Остальные функции окна **Debug** не требуют особых пояснений. Здесь визуализируются основные ресурсы микроконтроллера, подключенных терминалов и виртуальных приборов. Пример на рисунке 7 отражает доступные ресурсы ОМК 8051/52, а выбран - блок регистров.

Окна ОМК и виртуальных приборов **отображаются только во время паузы** (в отличие от **Watch Window и Simulation Log).**

Изменить содержимое регистров или ячеек резидентной памяти ОМК в процессе отладки невозможно. Поэтому для изменения значений переменных необходимо перекомпилировать программу или использовать для ввода данных внешние источники информации (терминалы последовательных портов или паттерн-генератор).

2.1. Контекстное меню окна пошаговой отладки

Кроме описанных выше возможностей управления отладкой, некоторые опции спрятаны в контекстное меню, вызываемое по щелчку правой кнопкой мыши внутри окна **CPU Source Code в режиме паузы**. Вид меню представлен на рис. 12.

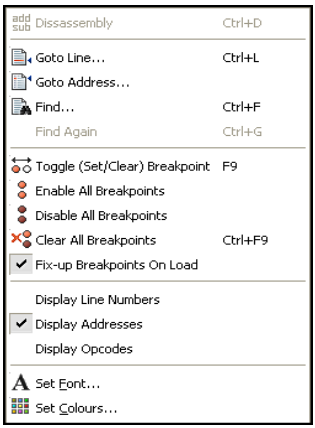


Рис. 12. Меню пошаговой отладкой

Окно поделено на пять секций.

Верхняя опция **Dissasembly** неактивна, если программа написана на ассемблере. **При программировании на языке высокого уровня СИ эта функция позволяет развернуть строку Си в коды ассемблера.**

Следующие две команды **Goto Line…** и **Goto Addresss…** переводят текущее положение выбранной строки (подсветку) по указанному номеру (адресу в ПЗУ программ ОМК). В первом случае вводится **десятичный номер строки**, а во втором - **шестнадцатиричный адрес ПЗУ программ**, например, так: **0x008E**. Опции **Find…** (Поиск) и **Find Again…** (Повторный поиск- становится активной после выполнения первого поиска) ничем не отличаются от аналогичных в других программах и ищут слово (фразу) и т.д. в соответствии с заданными условиями: регистр, слово целиком, вперед, назад…

Следующая группа команд управляет брекпойнтами (точками останова). Первая из них **Toogle…** – фактически повторяет кнопку в верхнем меню. Следующие три: **Enable** (активировать), **Disable** (дезактивировать) и **Clear** (очистить) выполняют данные действия

сразу для всех установленных брекпойнтов. Установка флажка **Fix-Up Breakpoins On Load** – активизирует все установленные точки при перезапуске симуляции.

В следующей группе установка флажков расширяет информационные возможности окна **CPU Source Code**.

Display Line Numbers – покажет номера строк, включая и комментарии.

Display Addresses –показывает адреса, по которым в ПЗУ программ ОМК расположены команды ассемблера,

Display Opcodes - отобразит шестнадцатиричные коды команд.

Последняя группа **Set Font…** и **Set Colours…** - позволяет настроить вид окна **CPU Source Code**: шрифт текста и цветовую гамму.

2.2. Настройка анимации

После запуска Proteus основные настройки устанавливаются автоматически, но их можно скорректировать, чтобы обеспечить требуемые параметры анимации.

В разделе **System**, используя команду **Set Animation Options,** можно установить следующие функции (рис. 13):

Show Voltage&Current on Probes? -выводить значения тока и напряжения в пробниках.

Show Logic State of Pins? - отображать состояние логических выводов в виде цветных квадратов. Цвета могут быть изменены командой **Set Design Colours** раздела **Template** основного меню. По умолчанию синие квадраты используются для отображения логического 0, красные - логической 1 и серые – высокоимпедансного состояния.

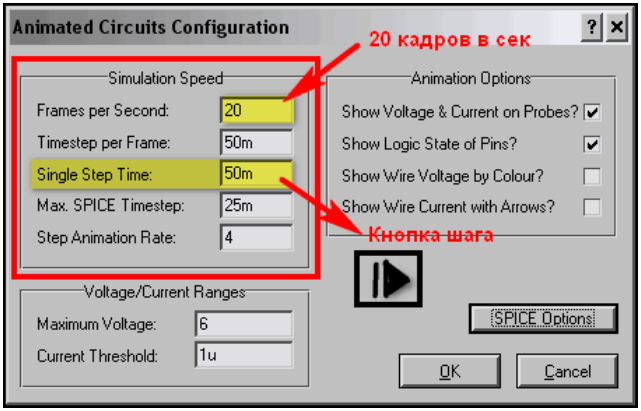


Рис.13. Окно настройки анимации

Show Wire Voltage bi Colour? - отображать напряжение в проводниках цветом. По умолчанию цветовой спектр задан от синего при –6V через зелёный при 0V к красному при +6V. Напряжение можно изменить в диалоге **Set Animation Options**, а цветовую гамму командой **Set Design Colours** в разделе **Template**.

Show Wire Current with Arrows? - показывать направление тока в проводниках стрелками. Стрелка появляется, если амплитуда тока превышает пороговое значение. Пороговое значение по умолчанию — 1uA, но может быть изменено в окне Set **Animation Options.**

Frames Per Second - количество кадров в секунду. Значение по умолчанию - 20 кадров в секунду. Это оптимально для большинства современных компьютеров, так как обеспечивает наилучшее соотношение между сглаженным воспроизведением графики и нагрузкой на графический процессор компьютера. Увеличение значения (max=50) увеличивает нагрузку на процессор ПЭВМ, уменьшение – снижает нагрузку и замедляет анимацию.

Timestep Per Frame – временной интервал на кадр (длительность кадра*)* анимации. Параметр определяет временной интервал, на который продвинется симуляция за один кадр. По умолчанию устанавливается значение 50 мс.

Single Step Time– время одного шага. Этот параметр связан с кнопкой Step пошаговой анимации внизу слева в основном окне

ISIS. Такими шагами симуляция будет продвигаться при каждом нажатии кнопки, а не по шагам микропрограммы контроллера, если таковой присутствует в схеме – для этого существует другая кнопка и в другом месте.

Step Animation Rate - частота кадров анимации в сек. По умолчанию установлено 4 кадра в сек. Максимальное число, которое может быть установлено – 10.

Изменять временные параметры анимации следует осторожно, контролируя загрузку процессора ПЭВМ.

Установка параметров симулятора ProSPICE выполняется в разделе **System** командой **Set Simulation Options** или из окна **Animated Circuit Configuration,** нажав кнопку **SPICE Options**. Окно содержит пять вкладок: **Tolerances** (допуски по точности), **MOSFET** (параметры МОП транзисторов), **Iteration** (количество шагов в вычислениях), **Temperature** (температурные параметры), **Transient** (параметры переходных процессов), **DSIM** (параметры симуляции цифровых цепей). Подробно эти параметры описаны в [6].

В большинстве случаев для моделирования достаточен режим **Default Setting, который устанавливается по умолчанию**. Кроме этого, возможна установка режимов

Setting for Better Accuracy (установки для наилучшей точности) – но при этом увеличивается время вычислений и нагрузка на ЦП компьютера;

Setting for Better Convergence (установки для наилучшей сходимости решений) – здесь соответственно все наоборот.

3. Особенности работы с памятью данных

При работе с внешней памятью 8051 порты P0, P2, P3 необходимо настроить на системный режим (режим микропроцессора), записав в регистр-защелку портов код «все единицы». Этот код записывается в регистры-защелки портов **после сброса аппаратно**, но в процессе работы их содержание **может измениться.**

После настройки на системный режим по команде **MOVX** через порт P0 передаются с разделением времени младший байт адреса и данные, выход P2 содержит старший байт адреса, а на линиях P3.7 и P3.6 формируются соответственно сигналы записи (WR) и чте-

ния(RD) данных. В момент передачи адреса на выходе ALE ОМК формируется импульс, который следует использовать для управления внешним регистром, в котором будет фиксироваться младший байт адреса. При работе с **одной** микросхемой памяти электрическая схема представлена на рис. 14.

Элементы схемы выбираются из библиотеки компонентов: U2 (74LS 373) - регистр младшего байта адреса, U3(MEMORY_13_8) – статическое ОЗУ. На рис. 12 разряды A12, A13 внешней памяти заземлены. Если необходим больший объем памяти, то их соединяют с соответствующими разрядами порта P2.

После соединения элементов необходимо задать свойства элементов. Двойным щелчком левой кнопкой по элементу открывается окно для редактирования.

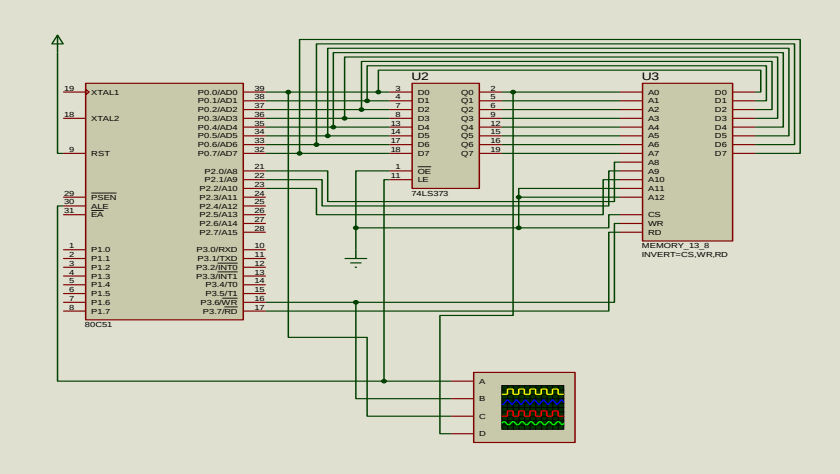


Рис. 14. Схема соединения микроконтроллера 8051 с внешней памятью

В настройках ОЗУ следует указать Advanced Properties – (Default), Other Properties – INVERT = CS, WR, RD, а в регистре - Initial Output State of Latch- (Default). Схему сохраняют и приступают к разработке программы.

Proteus не поддерживает функции загрузки данных в резидентную и внешнюю память из файла или в ручном режиме. Поэтому при тестировании программ необходимо разработать процедуру загрузки данных. При **работе с массивами** наиболее мобильным

вариантом является формирование массива данных в резидентной памяти программ и последующая его загрузка.

Пример подпрограммы загрузки данных в РПД или ВПД 8051 приведен ниже:

Входные данные

Адрес массива в памяти команд – dptr

Адрес массива в РПД – r0

Адрес массива в ВПД – r1- младший байт адреса, P2 – старший байт адреса

Длина массива – r3

Признак приёмника массива РПД (1) или в ВПД (0) – бит с адресом 0

Используемые дополнительные регистры – нет

Выходные данные:

Массивы в РПД или ВПД

Длина программы - 61 байт

Время выполнения на частоте кварцевого резонатора 12 МГц – для ВПД –(20+ 10n) мкс, РПД–(20+ ;10n) мкс, где n –число элементов массива.

```
Ljmp Start
org   30h
     db 23h, 34h,35h, 37h,40h,55h,60h,65h
    org 40h
Start:
mov sp,#70h
mov dptr, #30h;параметры подпрограммы - адрес в ПЗУК
mov r3,#8; Длина массива
clr 0; 0 -запись в ВПД
;setb 0; 1- запись в РПД
mov r0,#30h; адрес массива в РПД
mov r1,#30h; мл. байт адреса массива в ВПД
mov p2,#0;ст. байт адреса массива в ВПД
call load; подпрограмма загрузки данными РПД или ВПД
jmp loop
     org 80h
 load:
     push acc; сохранение в стеке
     push psw
```

```
        jb 0,k; проверка места сохранения массива
    l:
         clr a
        movc a,@a+dptr; чтение первого элемента
        movx @r1,a; Запись в ВПД
        inc dptr;организация цикла
        inc r1
        djnz r3,l
        sjmp l1
    k:  clr a
        movc a,@a+dptr; чтение первого элемента
        mov @r0,a; Запись в ВПД
        inc dptr; организация цикла
        inc r0
        djnz r3,k
    l1: pop psw;восстановление
        pop acc
        ret;выход из подпрограммы
Loop:
        jmp Loop
        END; конец трансляции
```

Аналогичная процедура может использоваться для AVR

```
.org $40;
string: .dw $2211,$4433,$6655,$8877,$99aa,$0011,$3311,$0aabb
.org $50
Start:
    ldi xl,$60; Начальный адрес в РПД
   clr xh
   ldi r18,8; Длина массива
    ldi zl, low(string<<1); Загрузка адреса массива в памяти данных
    ldi zh, high(string<<1); и сдвиг влево
m:
    lpm r2,z+; Запись массива в РПД
    st x+,r2
    dec r18
    brne m
Loop:
```

```
rjmp  Loop
```

При использовании макрокоманды**. dw** в AVR следует учитывать, что **в ОЗУ данных младший байт записывается в ячейку с младшим адресом.**

4.Особенности отладки программ ввода-вывода

При выполнении ввода-вывода **в последовательном коде** следует использовать отладчики Virtual Terminal для работы с UART(USART) **в асинхронном режиме**, SPI Debugger, I2C Debugger- соответственно для SPI и I2C. Дополнительным средством контроля могут служить осциллограф или логический анализатор. Особенности работы с ними отражены в приложении А.

Синхронный режим работы UART(USART) можно отлаживать только с осциллографом или логическим анализатором.

При отладке **ввода** информации в последовательном коде могут быть полезны **DPattern,** находящийся в папке **Generator Mode** (синусоида в окружности) и **Pattern Generator** (см. Приложение А), **каждый бит** которого может генерировать независимую последовательность импульсов.

Единственным виртуальным устройством **ввода информации в параллельном коде** является **Pattern Generator.** Паттерн-генератор ПГ является буфером FIFO, управляемым сигналами внутренней CLKOUT или внешней синхронизации CLKIN, HOLD, OE. В задачах ввода-вывода он может быть ведущим или ведомым устройством.

При работе ПГ в **режиме ведомого** ОМК может инициировать обмен, формируя сигналы внешней синхронизации, подаваемый на вход CLKIN, управление сдвигом HOLD и состоянием выходного буфера OE.

В **режиме ведущего** ПГ может задавать начало обмена ОМК с помощью сигнала CLKOUT.

При наличии достаточного количества параллельных портов ввода-вывода (например, AVR) соединение ОМК и ПГ не вызывает затруднений. (Рис.15) ПГ используется в режиме с внутренней синхронизацией. Инициатором обмена является ОМК, которое формирует сигналы OE и HOLD. Сигнал синхронизации CLKOUT поступает на вход PB0.

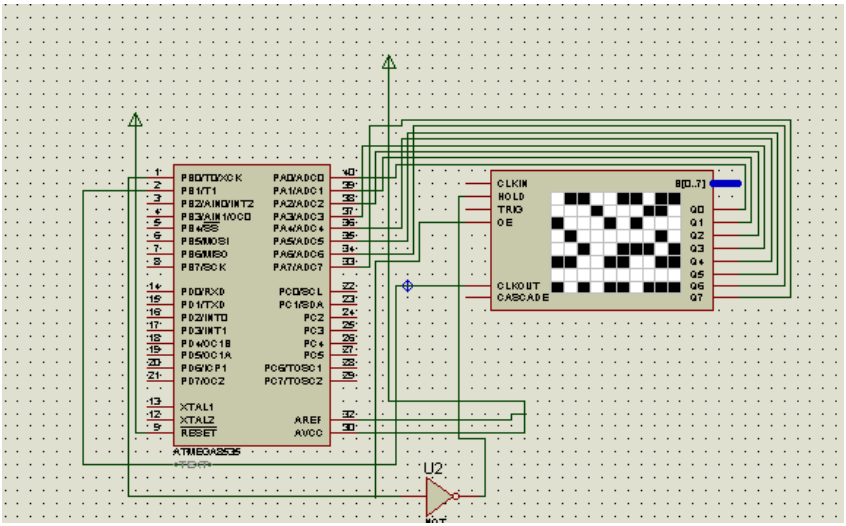


Рис.15. Схема контроллера на базе ATmega 8535

При работе ОМК в режиме микропроцессора или расширенного микроконтроллера, когда данные передаются по шине от нескольких устройств в режиме с разделением времени, требуется дешифрация адреса. Пример организации контроллера представлен на рис. 16

Микроконтроллер 80C51 соединен с внешним ОЗУ U3 и ПГ с помощью шины данных, организованной на базе порта P0. Регистр U2 служит для хранения младшего байта адреса, а старший байт формируется на выходе порта P2. Дешифрация устройств выполняется разрядом A14. Буфер данных ОЗУ будет открыт при A14=0, а паттерна – при A14=1. ПГ работает в режиме с внутренней синхронизацией, формируя на выходе CLKOUT выходной сигнал, с помощью которого осуществляется запись данных в ОМК. Инвертор U4 формирует запрос прерывания по фронту CLKOUT.

При программном вводе CLKOUT должен поступать на свободный вход порта ОМК. По фронту CLKOUT выполняется запись информации в ОМК, а по срезу – подача на выход ПГ очередного байта.

Виртуальными **приемниками информации в параллельном коде** могут служить логический анализатор, микросхемы ОЗУ и/или регистры.

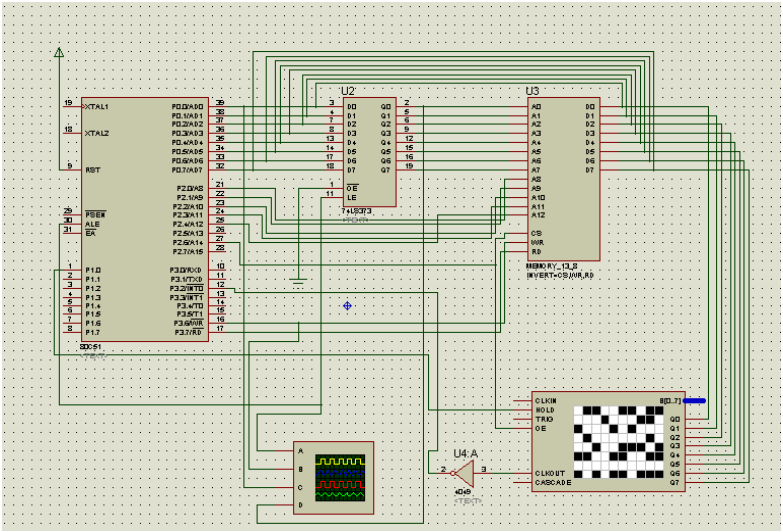


Рис.16. Схема контроллера на базе ОМК 8051

Пример присоединения к шине данных дополнительного ОЗУ и регистра приведен на рис.17

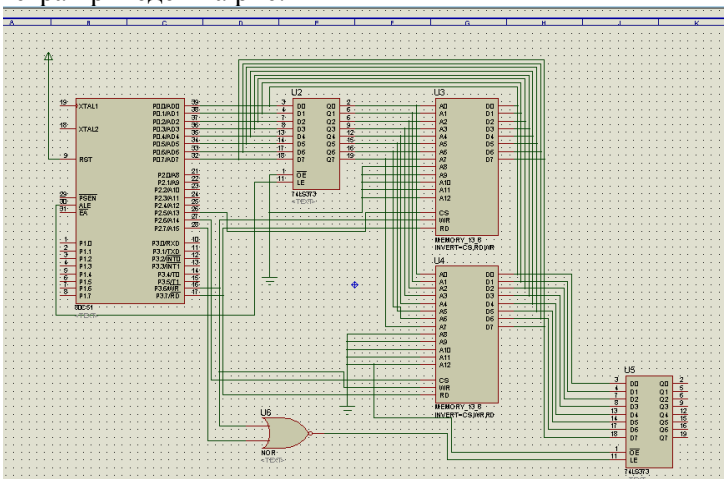
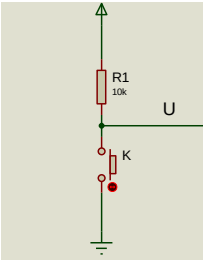


Рис. 17. Схема присоединения дополнительных устройств к 8051

В целях упрощения схемы для обращения к ОЗУ используется только младший байт адреса и **неполная дешифрация**. Биты старшего байта адреса, формируемые на выходе порта P2, равны для U3 - 0d0h, U4 - 0b0h, U5 - 70h. Управление записью данных в регистр-защелку U5 выполняет схема U6, реализующая функцию «ИЛИ-НЕ».

Управляющие выходные сигналы можно контролировать с помощью осциллографа, а **формирование входных сигналов** выполняют с помощью генераторов или соответствующих схеме. Пример формирователя отрицательного импульса приведен на



(Рис.18)

Рис. 18. Формирователь отрицательного импульса

В исходном состоянии U=Uпит, при нажатии кнопки К U=0, а при отпускании- Uпит

Отладку контроллеров, предназначенных для измерения частотно-временных сигналов, рекомендуется выполнять с помощью **Signal Generator** (см. Приложение А), который позволяет более оперативно управлять параметрами в процессе отладки. Измерение (или контроль) частоты, интервалов времени, счета входных импульсов можно выполнять цифровым частотомером **COUNTER TIMER**

Источниками сигналов при отладке микроконтроллеров, предназначенных для работы с аналоговыми сигналами, могут быть аналоговые генераторы, расположенные в папке **Generator Mode** (синусоида в окружности), потенциометры, аттенюаторы, делители напряжения. Для измерения уровней напряжения следует использовать цифровые вольтметры и амперметры.

5. Краткий обзор библиотеки компонентов

При выполнении лабораторных работ и курсового проекта могут быть необходимы устройства, входящие в состав библиотеки компонентов **P.** Приведенная ниже информация является **кратким «путеводителем»** по этой библиотеке с учетом задач, решаемых в ходе учебного процесса.

Analog ICs. Информация об основных аналоговых компонентах: усилители различных типов, компараторы, фильтры, мультиплексоры/демультиплексоры, источники питания, стабилитроны, источники опорного напряжения, таймеры

Connectors. Pазъемы для параллельных и последовательных соединений

Data Converters. АЦП и ЦАП с различными техническими характеристикам, параллельным и последовательным (I2C, SPI) интерфейсами, аналоговые и цифровые датчики температур, фотоэлементы.

Electromechanical. Двигатели различного назначения, включая и шаговые.

Memory ICs. ОЗУ статического и динамического типов, EPROM, EEPROM с параллельными и последовательными (I2C, SPI) интерфейсами, буферы FIFO,

Microprocessor ICs. Модели наиболее распространенных микроконтроллеров, цифровые потенциометры с интерфейсами I2C, SPI, 1-Wire, часы реального времени, мониторы питания и сброса, прецизионные цифровые термометры и термостаты, программируемые генераторы напряжений различной формы, драйверы ЖКИ и полупроводниковых индикаторов, преобразователи и расширители интерфейсов, преобразователи двоичного кода в 7-сегментный с последовательным интерфейсом, датчики угла поворота (энкодеры) и так далее.

Miscellaneous. Батареи, кварцевый резонатор, модель инфракрасного порта, светофор, разъем COM-порта, элементы нечеткой логики, контроллер диска с интерфейсом ATA/IDE, источник света и фоторезистор

Modelling Primitives. Абстрактные модели (не привязанные к конкретной элементной базе) АЦП, ЦАП и основных цифровых элементов.

Operational Amplifiere. Операционные усилители различного назначения

Optoelectronics. Сегментные, алфавитно-цифровые и графические индикаторы с контроллерами и без них, с параллельными и последовательными интерфейсами, дисплеи и панели ЖКИ, оптроны, полупроводниковые индикаторы.

Simulator Primitives. Абстрактные модели генераторов, логических элементов, источников напряжения и тока. Генераторы частично дублируют функции, описанные в папке **Generator Mode.**

Speakers&Sounders. Звонки(зуммеры) и динамики

Switces&Relays. Клавиатуры (4x3, 4x4,4x6), кнопки, переключатели, тумблеры, реле.

Transducers. Устройства для измерения температуры, давления, влажности с аналоговым и цифровым выходами, с последовательными интерфейсами, терморезисторы и термопары различных градуировок, фоторезисторы, омметр, звуковой регулятор

Пассивные элементы представлены в папках **Inductors** (индуктивности и трансформаторы)**, Capacitors** (конденсаторы)**, Resistors** (резисторы).

Содержимое остальных папок соответствует их названию и не требует пояснений.

6.Порядок выполнения работы

1. Получить задание у преподавателя
2. Выполнить анализ полученного задания: определить состав необходимых компонентов и выбрать алгоритм решения задачи
3. Начать разработку проекта (см. раздел 1)
4. Разработать электрическую схему с учетом необходимых источников сигналов и средств отладки (см. раздел 1.2)
5. Задать требуемые свойства компонентам (см. раздел 1.2 и соответствующие разделы приложения А)
6. Написать программу в соответствии с рекомендациями, приведенными в [7,8]. **При выполнении лабораторных работ**

программа должна быть написана на ассемблере. Программа, написанная для 8051 и AVR, должна заканчиваться бесконечным циклом, так как у этих микроконтроллеров отсутствуют команды останова.

Входные тестовые данные формируются с помощью подпрограммы загрузки (раздел 3) и/или соответствующих виртуальных устройств (приложение А).

При **курсовом проектировании** язык программирования выбирается студентом.

1. Выполнить компиляцию проекта и устранить возникающие ошибки

2. Запустить отладку проекта, активизировать необходимые окна микроконтроллера и виртуальных устройств (см. раздел 2 и приложение А). Рекомендуется использовать **Watch Window,** в котором при симуляции **непрерывно** можно отображать требуемые регистры и ячейки памяти микроконтроллера. Это особенно **актуально для AVR, так как значения регистров ввода-вывода отображаются только по прямым адресам.**

Первоначальный запуск рекомендуется выполнять в **непрерывном режиме с точкой остановки в конце программы**. При обнаружении ошибок разместить **дополнительные точки останова** в критических местах программы. Если точки останова не помогают устранить ошибки, то следует перейти в **шаговый режим отладки**.

Если необходим детальный анализ процесса работы устройства, следует использовать опцию **Configure Diagnostics** и окно диагностических сообщений **Simulation Log.**

Дополнительными источниками информации при отладке могут быть пробники и осциллограф.

3. После отладки программы следует определить её параметры: объём программы и время выполнения. По умолчанию объём программ 8051 отражается в окне **Simulation Log**, а AVR –в окне **VSM Studio Output** при компиляции. Время фиксируется таймером, расположенным в нижней части экрана отладки. Дополнительную информация о процессе отладки можно получить, если открыть **Configure Diagnostics** и скорректировать содержимое окна **Simulation Log.**

4. Отчет должен содержать задание к лабораторной работе, обобщенную граф схему алгоритма, параметры и используемые ресурсы **подпрограммы вычислительной процедуры,** электрическую схему проекта, **подпрограммы ввода-вывода.**

7.Контрольные вопросы

Введение

1. В чем преимущества и недостатки программного симулятора? внутрисхемного эмулятора? интегрированной среды разработки
2. Какие преимущества обеспечивает применение оценочного модуля?
3. Какие требования предъявляются к виртуальной среде моделирования ВСМ?
4. Какие задачи можно решать с помощью ВСМ?
5. Какие преимущества обеспечивает применение ВСМ?
6. Какие параметры необходимо контролировать при выборе ВСМ?
7. В чем преимущества и недостатки Proteus Design Suite 8.0?

Средства разработки электрической схемы

8. Какова процедура создания проекта? Показать на компьютере
9. Какие микроконтроллеры поддерживает Proteus? Как задать требуемые свойства микроконтроллеру?
10. Как выбрать элементы электрической схемы?
11. Как организовать связь между элементами электрической схемы, используя линии? шину? терминалы?
12. Как выбрать и подключить требуемый генератор?
13. Почему не рекомендуется использовать без необходимости аналоговые генераторы?
14. В каких случаях более рационально использовать **Signal Generator**
15. В каких случаях следует использовать **DPattern**?
16. Какие средства предусмотрены для измерения напряжения, тока? Как их подключить к схеме?
17. Как измерить частоту, период, временной интервал?

18. Какие задачи можно решить с помощью осциллографа? В каких случаях его следует использовать?

19. Для каких целей предназначен логический анализатор? Его функциональные особенности.

20. В чем преимущества ждущего режима запуска?

21. Для каких целей предназначен паттерн-генератор ПГ? Его функциональные особенности.

22. Как настроить паттерн-генератор на режим с внутренней синхронизацией? внешней?

23. Как организовать с помощью ПГ программный ввод информации в ОМК? По прерыванию? В параллельном коде? последовательном?

Отладчик асинхронного порта UART (USART)

24. Какие средства предусмотрены для контроля ввод-вывода информации через последовательный асинхронный порт UART (USART)?

25. Какие технические характеристики Virtual Terminal?

26. Какие параметры необходимо задавать при настройке терминала?

27. Как организовать ввод информации в терминал? вывод?

28. Каков алгоритм обмена информацией, если терминал является приемником? передатчиком?

29. Как реализовать аппаратный метод управления потоком? программный?

30. Как подключить терминал к RS-232? RS-485?

Отладчик интерфейса SPI

31. Принцип обмена информацией по интерфейсу SPI.

32. Как соединить отладчик SPI.с микроконтроллером в режиме ведущего? Ведомого?

33. Как организовать обмен информацией по SPI.с несколькими ведомыми устройствами?

34. Какие параметры необходимо задавать при настройке отладчика SPI.?

Отладчик интерфейса I2C

35. Принцип обмена информацией по интерфейсу I2C.

36. Какова структура кадра I2C?

37. Какие параметры необходимо задавать при настройке отладчика I2C?

38. Как соединить отладчик I2C.с микроконтроллером в режиме ведущего? Ведомого?

39. Как организовать обмен информацией по I2C с несколькими ведомыми устройствами?

40. По заданию преподавателя нарисовать электрическую схему, подключить к ней необходимые для отладки виртуальные устройства и установить свойства используемых элементов.

Отладка микропроцессорных устройства

41. Как активизировать окна наблюдений при отладке программы?

42. Для каких целей следует использовать опции Configure Diagnostic и Simulation Flag? Какая информация в них отображается?

43. Какие преимущества обеспечивает использование окно Watch Window? Какая информация в нем отображается?

44. Как задать непрерывный режим работы с точками останова? шаговый режим? Дать рекомендации по их использованию.

45. Какая информация находится в контекстном меню окна пошаговой отладки? В каких случаях это окно следует использовать?

46. Как организовать загрузку тестовых данных в микроконтроллеры 8051 и AVR?

Список использованных источников

1. Средства разработки и отладки для однокристальных микроконтроллеров URL: http://www.gaw.ru/html.cgi/ txt/publ/eqump/phuton2.htm
2. Инструментальные средства р азработки и отладки микроконтроллеров URL: http://www.phyton.ru/development-tools
3. Мамаева Т. Программные и аппаратные средства поддержки разработок компании IAR System. [Текст] Компоненты и технологии №4,2008г., с.108-109
4. FLOUCODE 6 – новые полеты. Радиоежегодник. Вып.29,2013г. URL: . http://www.rlocman.ru/book/book.html?di=150138
5. Пакет проектирования TINA. URL:www.ru.tina.com/document/Tina_10_manual.pdf
6. Proteus по-русски Радиоежегодник.Вып.24, 2013г.. URL: http://www.rlocman.ru/book/book.html?di=148418
7. Иоффе В.Г. Архитектура и программирование однокристальных микроконтроллеров MCS-51/52. Рег номер 18-015ЭИ.Самар. гос. аэрокосм. ун-т им. С.П. Королева (нац. исслед. ун-т)...- Самара, 2015.- 126с.:ил.
8. Магда Ю.С. Микроконтроллеры серии 8051: практический подход- М.: ДМК Пресс, 2008.- 228с.,ил.
9. Мортон Дж.Микроконтроллеры AVR. Вводный курс/Пер. с англ.. - М: Издательский дом «Додека -XXI», 2006.-272 с.:ил.
10. Евстифеев А.В.Микроконтроллеры AVR семейства Mega. Руководство пользователя. -М: Издательский дом «Додека -XXI», 2007.-592 с.:ил.
11. Евстифеев А.В.Микроконтроллеры AVR семейства Tiny. Руководство пользователя.. -М: Издательский дом «Додека -XXI», 2007.-432 с.:ил.
12. Ревич Ю. В. Практическое программирование микроконтроллеров Atmel AVR на языке ассемблера. . — 2-е изд., испр. — СПб.: БХВ-Петербург, 2011. — 352 с: ил.
13. Иоффе В.Г. Отладка программ на симуляторе VMLab 3.12. Рег номер 14-015ЭИ.Самар. гос. аэрокосм. ун-т им. С.П. Королева (нац. исслед. ун-т)...- Самара, 2015.- 85с.:ил.

14. Программирование на языке С для AVR и PIC микроконтроллеров. Сост. Ю.А. Шпак. .. –К.: «МК-Пресс», 2006. -400 с.:ил.

15. Лебедев М.Б. Code Vision AVR: пособие для начинающих... -М.: Додека-XXI, 2008,-592с.: ил.

Приложение А. Виртуальные средства отладки

1. Генератор сигналов

Генератор сигналов **Signal Generator** используется при моделировании в качестве многофункционального источника периодического напряжения с регулируемой частотой и амплитудой. По сравнению с генераторами, представленными в меню **Generator Mode, Signal Generator** позволяет более оперативно управлять параметрами в процессе отладки.

Характеристики генератора:

- формы сигнала: прямоугольная, пилообразная, треугольная, синусоидальная;
- регулируемая выходная частота - в диапазоне (0-12) МГц;
- регулируемая амплитуда выходного сигнала - в диапазоне (0-12) В;
- возможна амплитудная и частотная модуляция выходных сигналов.

Генератор выбирается из списка виртуальных устройств (**Signal Generator**), размещается в поле схемы и соединяется с соответствующими линиями ОМК (рис. А.1), где **+, -** - соответственно положительная и отрицательная выходные линии, **AM** – вход управления амплитудной модуляцией, **FM** – вход управления частотной модуляцией. При работе с однополярным сигналом отрицательный выход следует заземлять. Если не требуется модуляция, то входы AM, FM – остаются неподключенными.

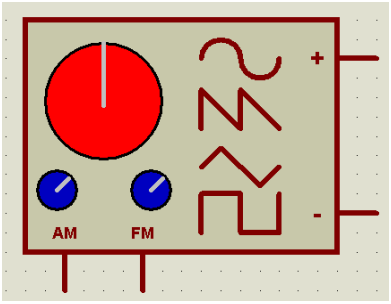


Рис. А.1. Условное изображение генератора сигналов

После запуска симуляции открывается окно, в котором устанавливаются параметры генератора (рис. А.2)

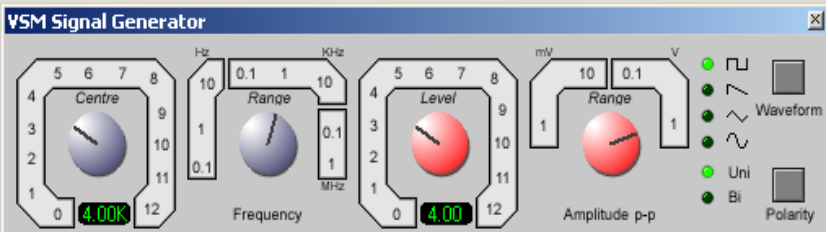


Рис. А.2. Панель управления генератором

Ручка регулировки Range задает масштаб для частоты и напряжения, Centre – значение частоты, **Level** – значение амплитуды. С помощью кнопок переключения **Waveform и Polarity** устанавливается вид выходного сигнала и его полярность (Unix –однополярный сигнал, **Bi** – дифференциальный сигнал).

Входы амплитудной и частотной модуляции имеют следующие возможности:

- усиление входа модуляции в терминах Hz/V или V/V задаются с помощью управления **Frequency Range** и **Amplitude Range**, соответственно;
- входное напряжение модуляции ограничено в пределах +/- 12V;
- входы модуляции имеют бесконечное входное сопротивление;
- напряжение на входе модуляции добавляется к установкам управления верньером до умножения установками диапазона, чтобы определить мгновенное значение частоты амплитуды.

Например, если частотный диапазон выбран в 1KHz, а частотный верньер установлен в 2.0, тогда уровень 2V частоты модуляции даст выходную частоту 4kHz.

Примеры частотной и амплитудной модуляции иллюстрируют соответственно рис. А.3 и рис. А.4.

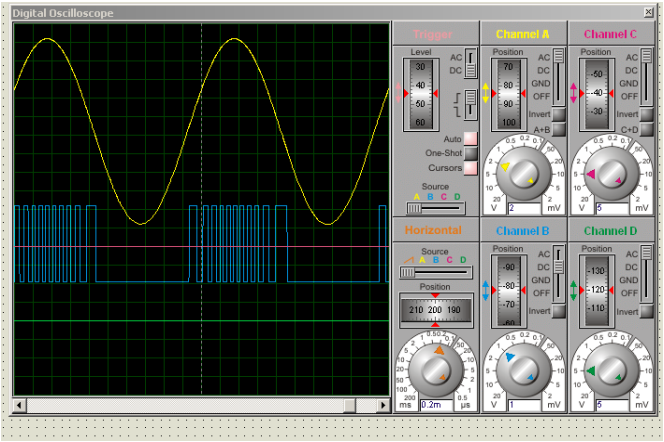


Рис. А.3. Частотная модуляция

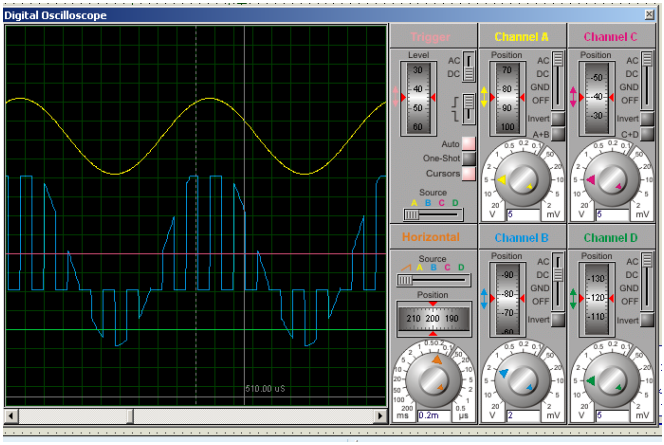


Рис. А.4. Амплитудная модуляция

2. Генератор цифровых данных (паттерн-генератор)

Паттерн-генератор (генератор шаблонов) **Pattern Generator** используется при отладке микропроцессорных устройств как источник цифровой информации в параллельном или последовательном кодах. С его помощью можно проектировать и отлаживать устройства, реализующие ввод информации в

программном синхронном или асинхронном режимах и по прерыванию.

Генератор позволяет создавать произвольную байтовую (до 1 КБайта) или восьмиканальные битовые последовательности .

Особенности генератора:

- интерактивный и графический режимы работы;
- режимы внутренней или внешней синхронизации;
- точные настройки временных параметров (временной развертки);
- ручная установка длины периода паттерна;
- сохранение и загрузка паттернов;
- возможность внешнего управления для приостановки передачи паттерна;
- отображение информации в шестнадцатиричной и десятичной формах;
- редактирования свойств генератора и значений паттерна в процессе работы;
- контроль текущего состояния генератора;
- генератор выбирается из списка виртуальных устройств **(pattern generator)**, размещается в поле схемы и соединяется с соответствующими линиями омк.

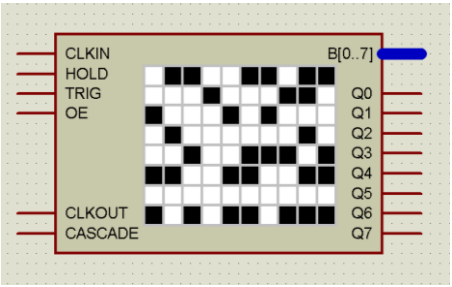


Рис. А.5. Условное изображение паттерн-генератора

Условное изображение генератора представлено на рисунке А.5, где

B [0-7] – выводы для связи с **шиной**, **Q0-Q7** – **индивидуальные** выводы. **Выходы** с тремя состояниями.

CLKIN (вход) - используется для подключения внешнего тактового генератора.

Hold (вход) - может использоваться для остановки (паузы) паттерн-генератора. Если на входе **высокий уровень,** прекращается формирование тактовых импульсов и данные на выход не поступают. Передача возобновится, если на входе **Hold** будет **низкий уровень.**

TRIG (вход) - используется для подачи **внешнего импульса** запуска. Запуск тактового генератора синхронизируется по этому входу.

OE (вход) – управление выходным буфером паттерн-генератора. При OE=1 данные передаются на выход генератора. Если OE=0, то генератор работает с заданным шаблоном (последовательностью), но данные на выход не передаются (выход находится в третьем состоянии).

CLKOUT (выход) – выход внутреннего тактового генератора. По умолчанию эта опция не установлена, так как затрудняет моделирование, особенно на высоких частотах. Этот выход может быть активизирован при задании свойств генератора (**Edit Component).**

Cascade (выход) - переходит в высокое состояние в течение первого такта (один период синхрогенератора) после начала симуляции. Фронт указывает на начало передачи первого байта (бита) паттерна, а срез – на её окончание. Далее сигнал находится в состоянии нуля до начала следующего цикла или сброса.

Перед началом симуляции двойным щелчком левой кнопки мыши открыть окно редактирования свойств генератора и задать соответствующие параметры (рис. А.6). Перевод параметров приведен в таблице А1.

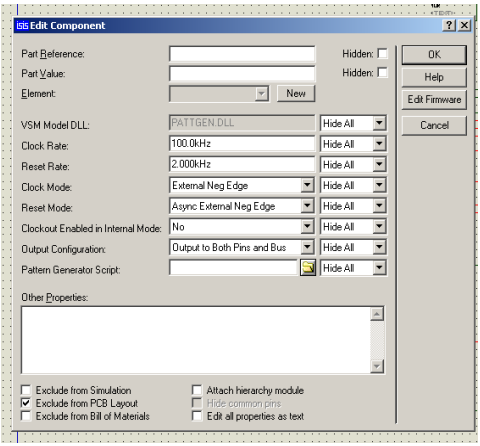


Рис. А.6. Окно редактирования свойств генератора

Таблица А.1. Свойства паттерн-генератора

Clock Rate	Частота генератора	100kHz	
Reset Rate	Частота сброса	2kHz	
Clock Mode	Режим генератора	Default Internal External Pos Edge External Neg Edge	 Внутренний Внешний по фронту Внешний по спаду
Reset Mode	Режим сброса	Default Internal Async External Pos Edge Sync External Pos Edge Async External Neg Edge Sync External Neg Edge	 Внутренний Асинхронный внешний по фронту Синхронный внешний по фронту Асинхронный внешний по спаду Синхронный внешний по спаду
Clock-out	Включение вывода Clockout	Default No	

Enabled in Internal Mode	при использовании внутреннего таймера	Yes	
Output configuration	Конфигурация выходных линий данных	Default Output to Both Pins and Bus Output to Pins Only Output to Bus Only	 Вывод на все выходные линии Вывод на индивидуальные линии Вывод на шину
Pattern Generator Script	Скрипт генератора паттернов	*File*	

После запуска симуляции появится окно, в котором можно оперативно изменить параметры настройки и задать требуемую последовательность байт, особенности ввода-вывода управляющих сигналов и ряд других функций (рис. А.7).

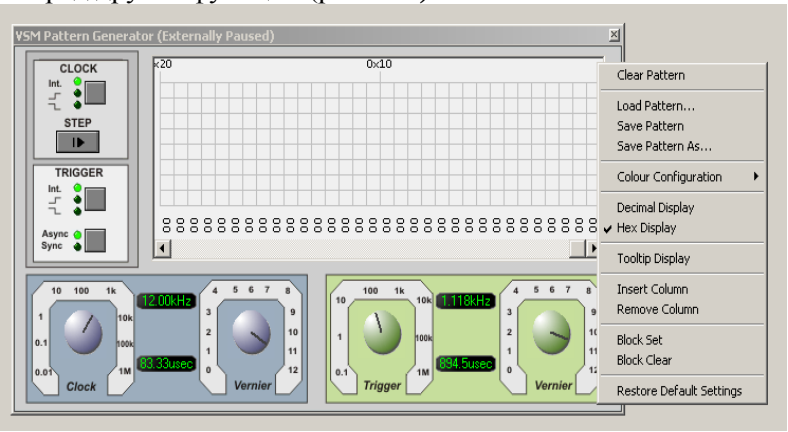


Рис. А.7. Окно настройки параметров генератора

Передаваемая информация (паттерн) может быть загружена вручную или из текстового файла.

В первом случае информацию вводится в поле данных побитно, нажимая левой кнопкой мыши на квадраты, или побайтно, нажимая левой кнопкой мыши на номер столбца.

Для загрузки из файла необходимо это свойство задать в окне редактирования (рис.А.6), двойным щелчком правой кнопкой в поле данных вызвать выпадающее меню и выполнить команду загрузки. **Загружаемый файл должен иметь расширение. prn.**

Файл представляет собой перечень байт, разделенных запятыми, где каждое значение определяет содержимое колонки на сетке данных. По умолчанию формат байта шестнадцатиричный, но можно использовать и десятичный формат. Для ввода комментария необходимо перед ним поставить с точку с запятой, тогда эти данные будут игнорироваться анализатором программы.

В этом меню можно задать и другие свойства поля данных паттерна: очистить, сохранить, установить блок в логическую 1или сбросить в 0, изменить форму отображения данных, вывести указатель контекстного окна (**Tooltip Display**), в котором отображается информация о текущих столбце и строке, и так далее.

Дополнительно можно **вручную** выставить период работы, кликнув левой кнопкой слегка выше сетки, над столбцом, на котором требуется остановить генератор (рис. А.8). Для отмены кликните правой кнопкой по той же области.

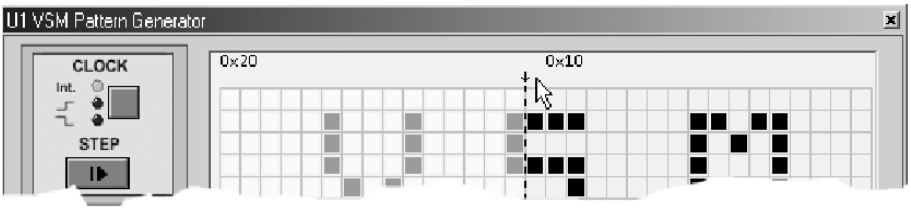


Рис. А.8. Задание периода генератора

Генератор может работать в режиме внутренней или внешней синхронизации.

Внутренняя синхронизация может быть задана **до симуляции** в окне редактирования свойств (рис. А.6) или **в процессе симуляции** - (рис. А.7)

В последнем случае необходимо в поле **CLOCK** с помощью кнопки установить **Int.**, выставить требуемую частоту. Наиболее простой способ задания частоты - текстовый режим, при котором двойной щелчок мышки по шкале **Clock** или **Vernier** вызывает

всплывающее окно редактирования. В нем указывается значение частоты в Гц, КГц, МГц. **Целая и дробная части разделяются точкой.**

Задать частоту можно также, используя управляемые мышкой дисковые шкалы **Clock** и Vernier. **Clock** определяет коэффициент масштабирования (Гц, КГц, МГц), а с помощью **Vernier** задается конкретное значение частоты.

Управление входом **TRIG** выполняется установкой в поле **TRIGGER** режима **Int.** и аналогично предыдущему - параметрами запуска с помощью шкал **Trigger и Vernier**. Дополнительно, чтобы триггер работал кратно битам тактовой частоты, в всплывающем окне можно указать количество бит с добавлением суффикса **bits**, например, (10bits). В этом случае через 10 импульсов генератор тактовой частоты сбрасывается, а на выходе **Cascade** формируется фронт импульса.

При запуске программы **в шаговом режиме** необходимо сначала нажать кнопку "Step" на панели управления анимацией, а затем кнопку “Step” в окне генератора.

В этом режиме передача данных выполняется по срезу импульса генератора синхронизации. Сигнал синхронизации можно использовать в процессе отладки, если установлено разрешение вывода **CLKOUT** (рис. А.6)

Внешняя синхронизация предполагает наличие внешнего генератора, выходная частота которого подается на вход **CLKIN.**

В зависимости от установки в поле **CLOCK** передача данных выполняется по фронту или срезу входного импульса.

Момент запуска генератора паттернов определяется параметрами, задаваемыми в поле **TRIGGER**. При **асинхронном** запуске работа генератора начинается сразу после прихода фронта или среза внешнего сигнала на вход **TRIGGER**, а при синхронном – следующим за сигналом на входе **TRIGGER** срезом импульса внешнего генератора. Более подробно вопросы синхронизации изложены в [6].

Применение паттерн-генератора

В режиме **внутренней синхронизации** взаимодействие ОМК и генератора обеспечивается сигналами **HOLD, OE, TRIG, CLKOUT, CASCADE**.

Если инициатором передачи данных является ОМК, то необходимо

соединить на схеме линии портов ОМК и генератора: запуск генератора выполняется сигналами на входах **HOLD, OE, TRIG,** данные считываются с шины или индивидуальных линий, запись данных в ОМК должна выполняться **по фронту** импульса **CLKOUT,** а переход к следующему циклу - **по срезу.** Анализ сигнала **CASCADE** позволяет определить момент окончания передачи первого байта (срез импульса) и начало сброса (фронт импульса).

установить в **Edit Component** частоту передачи и частоту сброса (триггера), задать разрешение вывода частоты передачи на выход **CLKOUT**, конфигурацию выходных линий и, если используется текстовый файл данных, его адрес.

Частота сброса (триггера) определяет количество повторяющихся импульсов. Например, при частоте генератора 100 КГц, частоте триггера10КГц количество импульсов -10. Те же параметры можно задать указанием в поле триггера 10bits.

Для реализации асинхронной передачи можно, например, сигнал **HOLD** использовать для начала работы, а **CLKOUT** – для записи данных в ОМК. При этом OE должен быть равен 1.

Если при отладке используется несколько паттерн-генераторов, передающих данные на один порт, управление подключением выполняется коммутацией сигнала **OE**.

Если инициатором обмена является паттерн-генератор, то начало обмена задает сигнал **CLKOUT,** а сигналы обратной связи подаются на входы **HOLD, OE, TRIG.**

В режиме **внешней синхронизации** принципы управления аналогичны. Источником внешней синхронизации могут быть внешний генератор или ОМК.

3. Цифровой осциллограф

Осциллограф используется для измерения и анализа аналоговых **AC** и дискретных **DC** сигналов. При работе с дискретными сигналами можно устанавливать переключатели как **AC**, так и DC. Однако необходимо учитывать, что реализация аналогового режима требует больших ресурсов ПЭВМ.

Осциллограф имеет следующие характеристики:

- Число каналов – 4 (A, B, C, D)

- Усиление канала - от 20В/деление до 2мВ/деление
- Временная развертка - от 200мс/деление до 0.5мкс/деление.
- Условия запуска- временная развертка (непрерывная, ждущая, по фронту или срезу сигнала), развертка аналоговым сигналом (например, для фигур Лиссажу)
- Режим работы каналов- автономный, суммирующий (A+B, C+D),
- Типы входов – открытый **AD** (для анализа сигналов в полном диапазоне их изменения) и с разделительным конденсатором **AC** (для анализа только переменной составляющей, если постоянная составляющая значительно больше переменной).

Для подключения осциллографа следует щелкнуть по кнопке **Virtual Instruments Mode**, выбрать в окне - **OSCILLOSCOPE (**рис. А.9), установить его на свободное место рабочего поля схемы и соединить его входы с точками наблюдения.

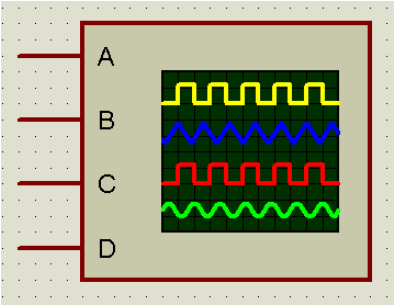


Рис. А.9. Условное изображение осциллографа

После сохранения и компиляции запустить проект, нажав кнопку **Debug/Start**.

Вид панели управления осциллографом представлен на рисунке А.10.

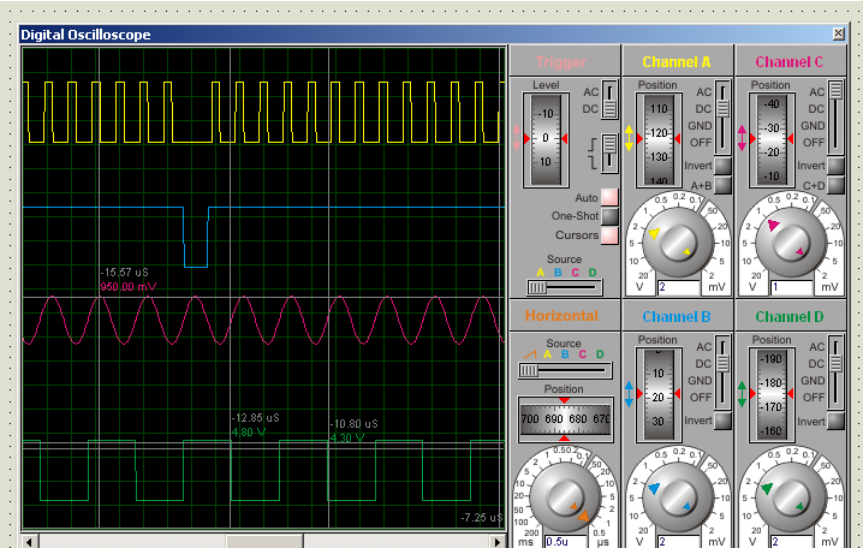


Рис. А.10. Панель управления осциллографом

Принцип работы с виртуальным осциллографом ничем не отличается от реального:

В секторе **Trigger** следует задать уровень (напряжение) запуска (**Level**) тип входа (**AD** или **AC),** канал источника запуска (**A, B, C, D**), особенности запуска (фронт или срез импульса), режим работы. Наиболее часто используется режим непрерывно развертки **Actel.** При этом **в** секторе **Horisontal** необходимо переключатель установить в положение пилообразной развертки и задать время развертки, соответствующее частоте (периоду) анализируемых сигналов.

Эти настройки являются общими для всех каналов.

Для **каждого** канала **Channel A - Channel D** можно установить начальное положение канала на экране осциллографа (**Position**), усиление (дисковая шкала), тип входа: **AD, AC, GND (**вход заземлен), **OFF (**канал отключен**), Invert** (инверсное значение входного сигнала**).**

Дополнительно каналы **A** и **D** могут отражать суммарное значение сигналов соответственно (**A+B) и (C+D).**

Так как размер окна осциллографа ограничен, то следует тщательно выбирать коэффициенты усиления наблюдаемых каналов.

Для удобства наблюдения каждый канал может отображаться индивидуальным цветом. Установка цвета канала выполняется

двойным щелчком правой кнопкой мыши в поле осциллографа и выбором из выпадающего меню функции **Setup.**

В тех случаях, когда не удаётся синхронизировать изображение, можно произвести снимок экрана и далее выполнить анализ зафиксированного процесса. Для этого режима на панели **Trigger** используется кнопка **One-Shot**.

Там же имеется кнопка **Cursors,** с помощью которой можно определить амплитуду сигнала и время его появления **относительно середины экрана**, отмеченную вертикальной пунктирной линией. Точность измерения – до третьего десятичного знака.

Таким образом, для анализа сигналов с помощью осциллографа необходимо:

- Выбрать тип входа
- Задать условия запуска и длительность развертки
- Установить режим работы, начальное значение и усиление каждого канала
- Запустить проект
- Произвести соответствующие измерения (см. Рис. А.10)

При работе с осциллографом не следует забывать об установке точек останова в программе, чтобы ограничить время наблюдения процесса. Если процесс короткий, то в конце развертки осциллографа его можно не увидеть.

4. Логический анализатор

Логический анализатор представляет собой устройство, предназначенное для записи и отображения последовательности **цифровых сигналов**.

В отличие от осциллографов, которые предназначены для амплитудного, фазового, временного измерения и анализа **ограниченного** числа **аналоговых** сигналов, логические анализаторы имеют значительно **большее количество каналов** (от 16 до нескольких сотен), но при этом **фиксируют только логические уровни сигнала**: 0, 1, иногда состояние Z (высокое сопротивление).

Анализатор — полностью цифровой прибор, имеющий в своем составе быстродействующее буферное ОЗУ. Логические сигналы в исследуемых точках с заданной частотой дискретизации регистри-

руются в буферном ОЗУ. После того как регистрация остановлена, данные, сохраненные в буферном ОЗУ, отображаются на экране монитора в виде временной диаграммы или таблицы состояний.

Запуск регистрации может осуществляться в автоматическом или в ждущем режимах. В первом случае запуск регистрации данных осуществляется автоматически по нажатию кнопки старт. Во втором - запуск регистрации осуществляется только в случае появления во входном потоке данных заданной логической последовательности.

Значение логического условия задается непосредственно перед запуском регистрации.

Кроме этого, в ждущем режиме возможна регистрация данных как до, так после момента выполнения условия.

Остановка регистрации данных анализатором происходит после заполнения всего объема буферного ОЗУ.

Логический анализатор применяется для отладки и диагностики цифровых систем, в которых необходимо контролировать одновременно большое число линий передачи информации. Например, шины адреса, данных, управления.

Логический анализатор можно использовать как средство регистрации информации в параллельном коде, так как в Протеусе подобный терминал отсутствует.

В Протеусе могут использоваться два типа терминала.

В Proteus 8.x используется анализатор со следующими параметрами:

Количество каналов – 16 - битовых и 4 - байтовых канала (рис. А.11).

Объем буферного ОЗУ – 40000x52

Период дискретизации – регулируемый от 0,5 нс до 200 мкс, что обеспечивает сохранение данных в буфере за время соответственно 10нс-4 с

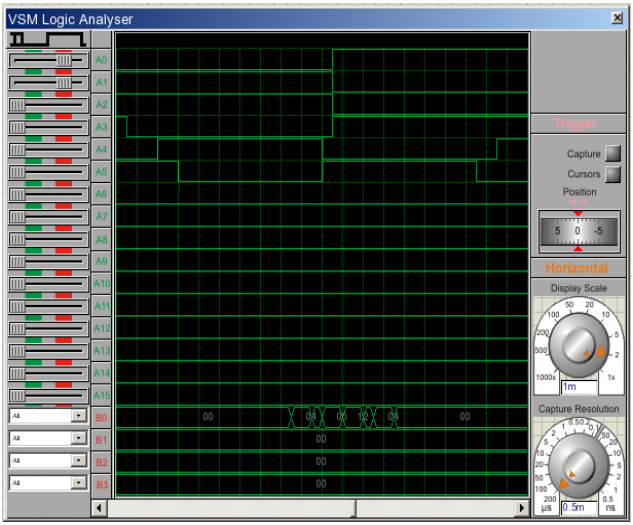


Рис А.11. Логически анализатор Proteus 8.x

Условие запуска - формируется как **конъюнкция входных сигналов.** Для **автоматического запуска** переключатели условий должны быть установлены в **крайнее левое положение.** Переключатели условий располагаются на лицевой панели слева.

Запуск анализатора высоким уровнем канала 1 и фронтом (изменение напряжения от низкого уровня к высокому), приходящим на канал 3, иллюстрирует рисунок А.12

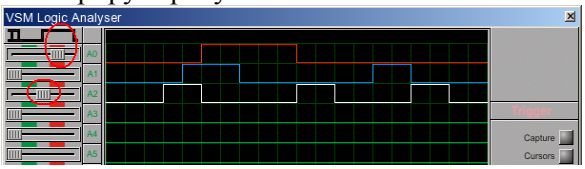


Рис. А.12. Сигналы запуска анализатора

Режимы начала работы буфера – (0-100) % буфера захвата.

Анализатор будет захватывать входные данные непрерывно, пока не обнаружатся логические условия для ждущего режима захвата. Когда это случится, захват данных продолжится до тех пор, пока не заполнится оставшаяся часть буфера после выполнения условия. Таким образом, при 0%-фиксируются события **после выполнения условия запуска,** при 100% **- до выполнения условия запуска,** а при остальных значениях – **до и после.**

Точное измерение времени с помощью курсоров. Установив кнопку **Cursors** в активное состояние, имеется возможность точного измерения времени между событиями.

Возможность изменения цвета отображаемых кривых, курсоров, текста и т.д.Цвет устанавливается щелчком правой клавишей мышки по дисплею и выбором из выпадающего меню **Colours Setup.** Из этого меню можно удалять курсоры и выводить на печать содержимое окна.

В других версиях Протеуса используется анализатор (рис. А.13).

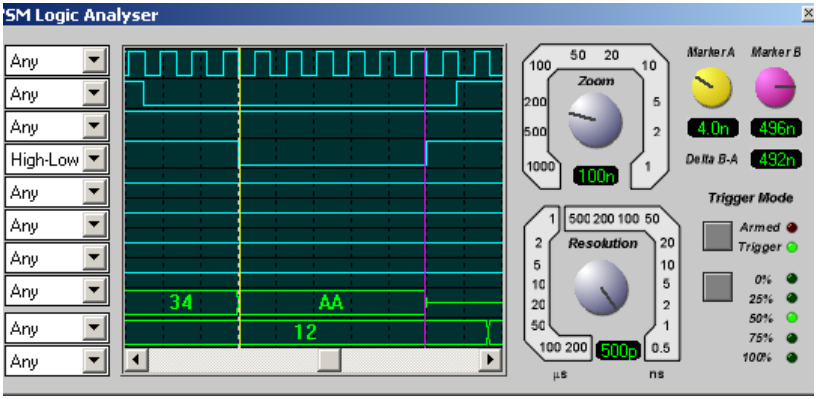


Рис. А.13. Окно управления анализатором

Количество каналов - 24 (8 каналов для битовых сигналов и 2 канала байтовых данных).

Объем буферного ОЗУ – 10000x24

Период дискретизации – регулируемый от 0,5 нс до 200 мкс, что обеспечивает сохранение данных в буфере за время соответственно 10нс-4 с

Условие запуска - формируется как **конъюнкция входных сигналов.** Для **автоматического запуска** переключатели условий должны быть установлены в **состояние Any**. Переключатели условий располагаются на лицевой панели слева.

Режимы работы буфера - 0, 25, 50, 75 и 100% буфера захвата

Анализатор будет захватывать входные данные непрерывно, пока не обнаружатся логические условия для ждущего режима за-

хвата. Когда это случится, захват данных продолжится до тех пор, пока не заполнится оставшаяся часть буфера после выполнения условия. Таким образом, при 0%-фиксируются события **после выполнения условия запуска,** при 100% **- до выполнения условия запуска,** а при остальных значениях – **до и после.**

Двухканальное измерение текущего времени с разрешающей способностью, определяемой периодом дискретизации. Для точного измерения времени используются **маркеры A и B**. Каждый маркер можно размещать, используя соответственно окрашенную шкалу. Показания отображают точку времени, отмеченную маркером, относительно времени выполнения условия запуска, а показания **Delta A-B** — разницу между маркерами. Установка места расположения маркера выполняется соответствующими ручками настройки. На рисунке А.16 показано измерение сигнала чтения внешней памяти данных **RD.**

Возможность просмотра содержимого буфера захвата. Необходимость в панорамировании и масштабировании буфера захвата возникает потому, что он удерживает 10000 отсчетов, а отображение имеет ширину только в 250 пиксел. Для этих целей служат ручки **Zoom, Resolition** и прокрутка, позволяющая перемещать изображение влево и вправо. Время одного деления вычисляется умножением установок шкалы **Zoom** на разрешение шкалы Resolition.

При **подключенной** функции **Graph Mode** окно наблюдений анализатора можно увеличить до нужного размера и изменять его свойства. Для этого следует включить **Graph Mode,** выбрать в селекторе **Digital** и на свободном месте листа проекта разместить окно графика, затем увеличить окно до нужных размеров. **В используемой версии эта функция отсутствует.**

Использование логического анализатора

Для подключения анализатора и отображения данных необходимо:

Щёлкнуть по иконке **Virtual Instruments Mode**, выбрать Logic **Analyser**, разместить объект на чертеже, соединить входы анализатора с требуемыми сигналами. Условное изображение анализатора представлено на рисунке А.14.

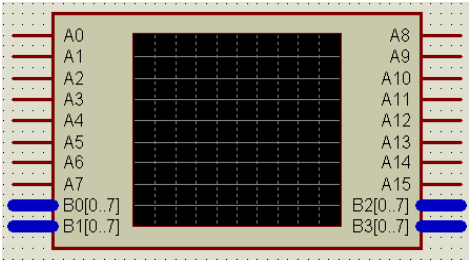


Рис. А.14. Условное изображение анализатора

Проверить свойства анализатора (двойной щелчок левой кнопкой мыши).

Запустить интерактивную симуляцию из меню **Debug** или используя кнопку **Play** на панели управления анимацией. Появится окно анализатора (рис. А.11, А.13).

Управление анализаторами в разных версиях Протеуса несколько отличается.

Для Proteus 8.x (рис. А.11)**:**

Задать период дискретизации, соответствующий наблюдаемому процессу (ручка **Capture Resolition**). Этим устанавливается самый короткий импульс, который будет записан в ОЗУ. Средняя ручка позволяет изменять масштаб в диапазоне от 1 (крайнее правое положение) до 2,5 (крайнее левое положение).

Установить на левой панели требуемые условия переключения (см. рис.А.12)

Выбрать режим работы буфера (дисковая шкала **Position**). Рекомендуемая начальная установка -50%

Ручкой **Display Scale установить** цену деления экрана, которая определяет размер окна наблюдений.

Включить анализатор кнопкой **Capture** и запустить требуемый фрагмент программы в непрерывном режиме с точкой останова, чтобы не перегружать буфер.

Используя прокрутку дисковой шкалы, выполнить анализ записанного процесса. Для измерения времени включить курсор **Cursors** в требуемых точках временной диаграммы. Поместив мышь в окно наблюдения, с помощью колесика можно изменять масштаб временной диаграммы.

Пример использования логического анализатора для контроля сигналов обмена между ОМК 8051 и внешней памятью данных приведен на рисунке А.15.

Для анализатора (рис.А.13):

Задать период дискретизации, соответствующий наблюдаемому процессу (ручка **Resolition**). Этим устанавливается самый короткий импульс, который будет записан в ОЗУ.

Установить на левой панели требуемые условия переключения. В примере на рисунке А.16 запуск осуществляется по срезу импульса чтения данных из внешнего ОЗУ **RD**.

Выбрать режим работы буфера (правая нижняя кнопка).

Ручкой **Zoom определить** размер окна наблюдений.

Запустить анализатор кнопкой **Trigger Mode**. В момент запуска включится светодиод **Armed,** а окончания записи **– Trigger.**

Используя прокрутку, выполнить анализ записанного процесса. Для измерения времени использовать маркеры **А и В.**

Пример использования логического анализатора для контроля сигналов обмена между ОМК 8051 и внешней памятью данных приведен на рисунке А.16.

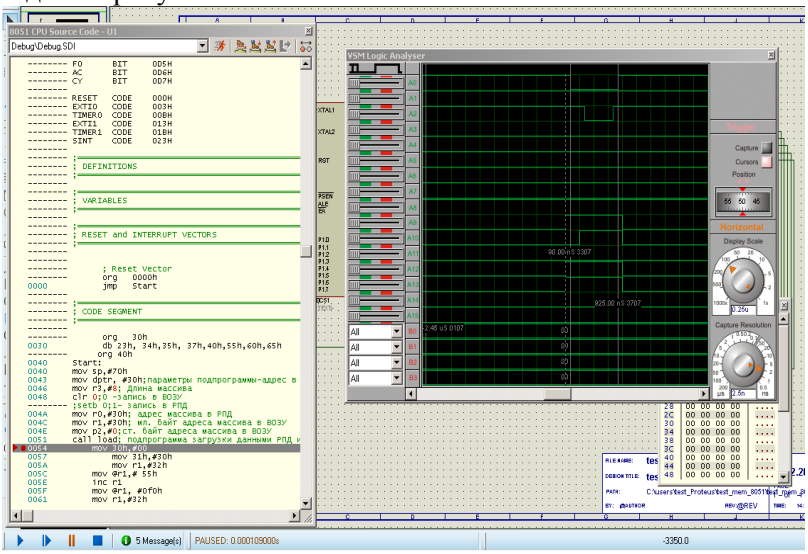


Рис.А.15. Пример использования для Proteus 8.x

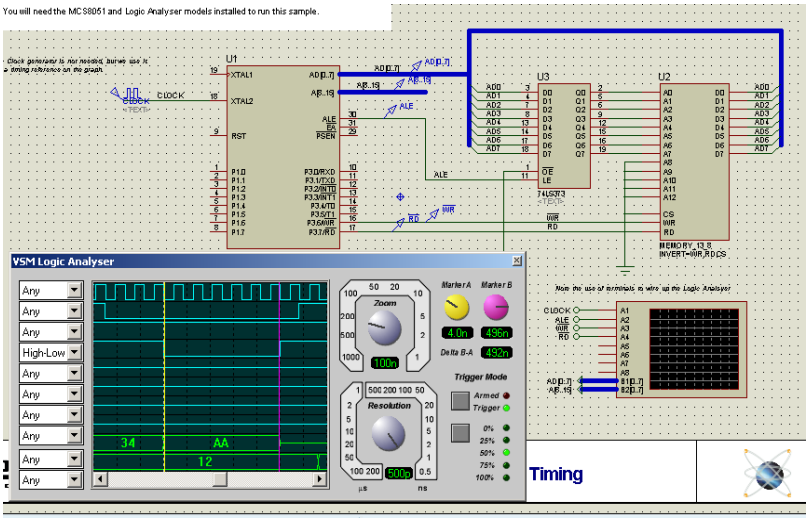


Рис. А.16. Пример использования анализатора

5. Цифровой частотомер

Цифровой частотомер **COUNTER TIMER** — это многофункциональный цифровой прибор, предназначенный для измерения интервалов времени, частоты, счета входных импульсов. Цифровой частотомер поддерживает следующие режимы работы:

- режим таймера (секунды), разрешающая способность 1мкс;
- режим таймера (часы, минуты, секунды), разрешающая способность 1ms;
- режим частотомера, разрешающая способность 1Гц;
- режим счетчика, максимальный счет 99 999 999.

Частотомер выбирается из списка виртуальных устройств (**COUNTER TIMER**), размещается в поле схемы и соединяется с источниками измеряемых сигналов (ОМК, генераторами и так далее). Условное изображение частотомера приведен на рисунке А.17, **CLK** – вход измеряемых сигналов, **CE** – вход разрешения работы, **RST** – вход сброса.

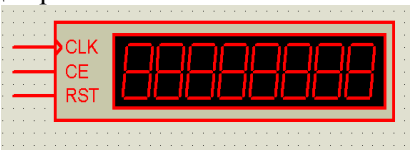


Рис. А.17. Условное изображение частотомера

Перед началом симуляции двойным щелчком левой кнопки мыши открыть окно редактирования свойств частотомера и задать соответствующие параметры: режим измерения, особенности управляющих сигналов **CE, RST** (рис. А.18).

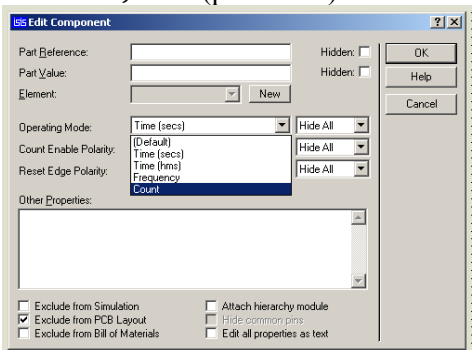


Рис. А.18. Окно редактирования свойств частотомера

После запуска симуляции появится окно, в котором можно оперативно изменять параметры настройки частотомера (рис. А.19), не вызывая окно редактирования. **Окно управления можно вызвать также, кликнув левой мышкой в изображение частотомера**. Параметры частотомера задаются переключателями **Reset Polarity** - полярность импульса сброса, **Gate Polarity** – полярность сигнала разрешения, **Manual Reset**- ручной сброс, **Mode** –режимы работы: Time **(secs)** - таймер с повышенной разрешающей способностью 1 мкс,**Time (hmcs)** - таймер с разрешающей способностью 1 мс, **Frequency** – частотомер, **Count** – счетчик.

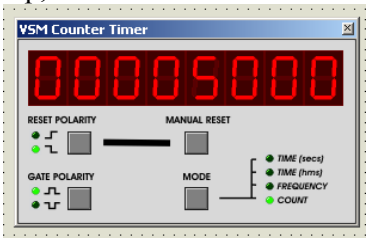


Рис. А.19. Окно управления частотомером

Значения времени, частоты или счета отображаются на схеме (рис. А.14) и в окне управления (рис. А.16), если оно открыто.

5.1. Режим измерения интервалов времени

Измерение интервалов времени может быть реализовано в двух режимах - с использованием внешнего сигнала **CE** и с открытым входом **CE.**

В первом случае необходимо выбрать полярность сигнала **CE**, определить разрешающую способность таймера и задать их в свойствах частотомера или окне управления, определить источник внешнего сигнала и подключить его к входу **CE.** Отсчет времени выполняется **внутренним генератором**. После запуска моделирования показания частотомера пропорциональны длительности сигнала на входе **CE**. Во втором – **CE** никуда не присоединяется и показания частотомера постоянно изменяются.

Сброс показаний можно выполнить в ручном режиме (**Manual Reset**) или сформировать внешний импульс. Сброс запускается фронтом или срезом импульса. Этот сигнал должен быть противоположным уровню запуска.

Если необходимо удерживать таймер в нуле, то это реализуется управлением по входам **CE** и **RST**.

5.2. Режим измерения частоты

Чтобы измерить частоту, необходимо подключить вывод **CLK** к источнику измеряемого сигнала и задать режим **Frequency**. Выводы **CE и RST** в режиме частотомера не используются.

Время подсчета импульсов фиксировано и равно 1 сек. Счет выполняется **по фронту** входного импульса. Поэтому для измерения частоты, например, синусоидального сигнала, на входе частотомера необходимо установить формирователь импульсов.

Использование частотомера для измерения аналоговых сигналов на частотах выше 50 КГц приводит к большой загрузке процессора ПЭВМ. Измерение выполняется не в режиме реального времени. Поэтому для определения частоты цепей аналоговых генераторов, работающих на частотах значительно выше 10кГц, рекомендуется использовать виртуальный осциллограф или анализатор.

5.3. Режим счетчика

Для работы в режиме счетчика необходимо задать режим **Counter,** установить параметры **CE и RST**, если это необходимо, подключить **CLK** к источнику входного сигнала.

Если функции запуска счета и сброса не требуются, эти входы оставляют неприсоединенными.

Если необходимо удерживать счетчик в нуле, то это реализуется управлением по входам **CE и RST**.

Значение счетчика изменяется **по фронту** сигнала на входе **CLK**, **если CE активен.**

В режиме счетчика можно измерять частоту, если на входе **CE** формировать эталонный временной интервал. Теоретически подобный режим будет более быстродействующим, чем стандартный режим счетчика, так как возможно управлять длительностью временного интервала.

Аналогично можно измерять длительность импульса, подавая на вход **CE** измеряемый сигнал, а на вход **CLK** – частоту эталонного генератора. И в этом случае возможности управления измерением расширены по сравнению с режимом таймера, так как частоту внешнего генератора можно регулировать.

6. Цифровые приборы

В состав виртуальных приборов Proteus входят цифровые приборы для измерения сигналов постоянного и переменного тока: вольтметры, амперметры, ваттметр. Они работают в режиме реального времени и могут быть подключены в схему, как и любой другой компонент.

Приборы выбираются из списка виртуальных устройств (DC VOLTMETER, AC VOLTMETER, DC AMMETER, AC AMMETER, WATTMETER), размещаются в поле схемы и соединяется с источниками измеряемых сигналов. Условные изображения приборов приведены на рисунках А.20– А.22.

Рис. А.20. Условное изображение вольтметров переменного и постоянного (АС) токов

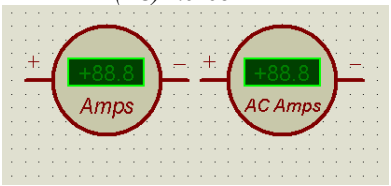


Рис. А.21. Условное изображение амперметров переменного и постоянного(АС) токов

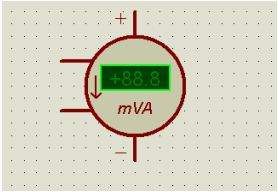


Рис. А.22. Условное изображение ваттметра

Все приборы имеют **три десятичных разряда** и могут настраиваться с масштабом 1, mili, micro. Редактирование свойств приборов выполняется перед началом моделирования. Окно свойств вызывается двойным щелчком правой кнопки мыши. Для всех приборов можно выбрать диапазон измерения и входное сопротивление. В вольтметрах по умолчанию устанавливается нагрузка 100Мом, но это значение может быть отредактировано. В приборах постоянного тока дополнительно указывается постоянная времени. В качестве примера на рисунке А.23 приведено окно свойств вольтметра постоянного тока.

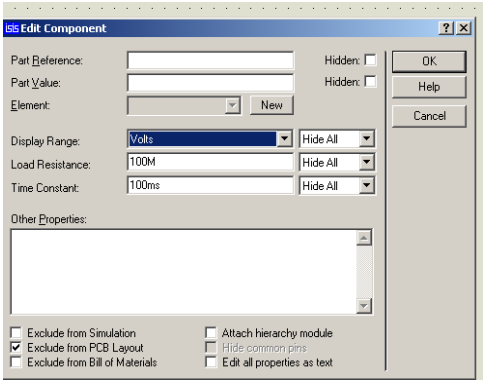


Рис. А.23. Окно редактирования свойств вольтметра постоянного тока

После начала моделирования экраны приборов в цифровом формате показывают текущие измеренные значения.

Ваттметр предназначен для использования только с синусоидальными сигналами. В процессе редактирования свойств ему можно задать режимы измерения активной (W), реактивной (VAR), полной (VA) мощности и фазовый угол.

7. Отладчик интерфейса SPI

Интерфейс SPI является одним из самых распространенных последовательных интерфейсов, применяемых в ОМК. Более 85% микроконтроллеров имеют в своем составе порты SPI. Широкое применение SPI связано с минимальными требованиями к аппаратной части устройств, подключаемых к ОМК (как минимум, требуется регистр сдвига), и относительно простым протоколом обмена. Интерфейс SPI синхронный, дуплексный, обмен по принципу «ведущий – ведомый», скорость обмена изменяется в широких пределах. В ОМК скорость обмена зависит от особенностей организации порта SPI и может достигать величины до 5 Мбит/с и более.

Промышленность выпускает большое количество периферийных устройств с SPI: датчики, АЦП, ЦАП, мосты, часы реального времени, память и так далее. Поэтому наличие терминала SPI существенно ускоряет отладку микропроцессорных устройств.

Условное изображение терминала представлено на рисунке А.24, где:

SCK – линия синхронизации. В режиме ведомого работает как вход, в режиме ведущего как выход;

DIN – ввод данных с шины SPI;

DOUT – вывод данных на шину SPI;

SS – линия выбора устройства. В режиме ведомого эта линия должна быть присоединена к логу.0, в режиме ведущего остаётся активной, пока передаются данные ;

TRIG – вход, с помощью которого можно управлять очередью последовательностей данных на выходе SPI.

Рис. А.24. Условное изображение терминала

Терминал (анализатор протоколов) SPI может работать в одном из трех режимов:

- режим ведомого устройства SPI (**Slave)**;
- режим ведущего (мастера) устройства SPI (**Master)**;
- режим монитора (**Monitor**) - терминал показывает информацию, передаваемую по шине SPI

Терминал выбирается из списка виртуальных устройств (**SPI Debugger**), размещается в поле схемы, соединяется с соответствующими линиями ОМК (рис. А.25), двойным щелчком левой кнопки мыши вызывается редактор свойств, в котором задаются параметры терминала (рис. А.26).

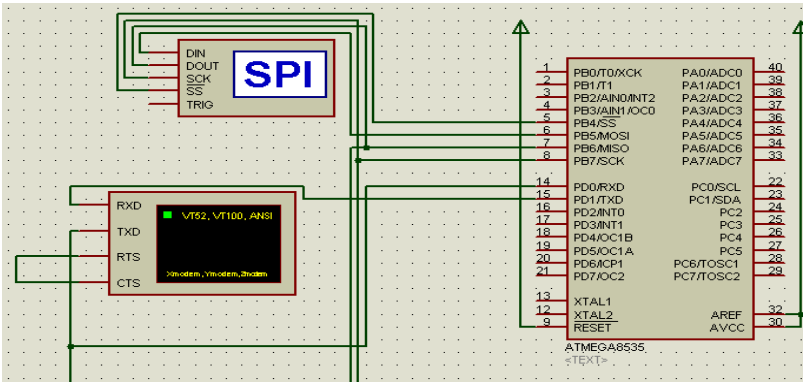


Рис. А.25. Схема соединения терминалов SPI и USART

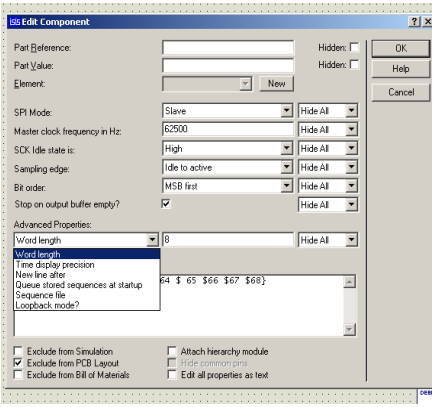


Рис. А.26. Окно редактирования свойств терминала

Таблица А.2. Функции редактирования

SPI Mode	Режим SPI	Default Monitor Master Slave	
Master clock frequency in Hz	Частота в Гц управляющего устройства	1000000	
SCK Idle state is	Уровень тактового сигнала	Default Low Hight	 Низкий Высокий
Sampling edge	Выбор типа запускающего сигнала	Default Idle to active Active to idle	 По фронту По спаду
Bit order	Порядок следования бит	Default MSB first LSB first	 Ст. разрядами вперед Мл. разрядами вперед
Stop on output buffer empty?	Нужна ли остановка при пустом выходном буфере?	?, no, yes	
Word length	Длина слова	Устанавливает количество бит в передаваемом слове (1-16)	
Time display precision	Точность отображения времени	Default	

New line after	Новая строка после передачи символов	Default	
Queue stored sequences at startup	Очередь хранит последовательности при запуске	Default No Yes	
Sequence file	Разрешает устанавливать имя текстового файла последовательности	*File*	
Loopback mode?	Цикличный режим Повторить режим?	Default No Yes	

После запуска отладки открывается экран, представленный на рисунке А.27. Экран разделен на несколько частей и позволяет контролировать состояние шины, принимать и передавать данные в режиме ведущего или ведомого.

Верхнее левое окно содержит основные данные о работе шины: направление передачи данных и их значения, время возникновения событий. Возможен анализ как байт , так и битовых последовательностей.

Если **терминал** используется **как передатчик**, то **значения данных** задаются в **правом нижнем окне**.

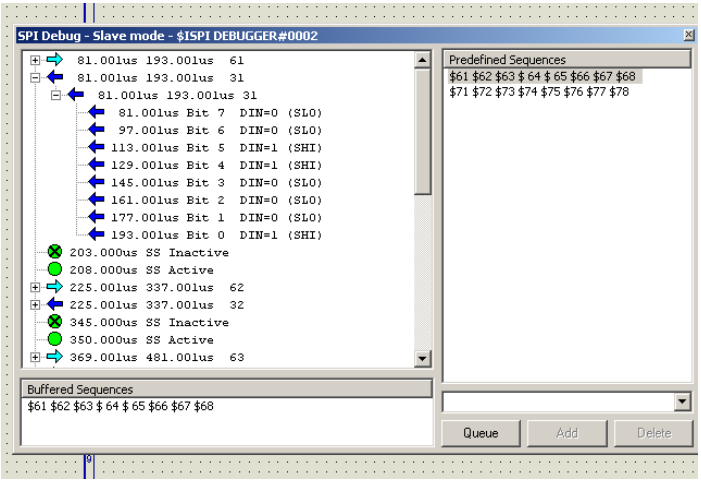


Рис.А.27. Экран отладчика

Таблица А.3. Правила

Синтаксис	Описание
‘0x’	Может использоваться как префикс для объявления шестнадцатеричных значений (0xFE)
‘$’	Может использоваться как префикс для объявления шестнадцатеричных значений ($FE)
‘h’	Может использоваться как суффикс для объявления шестнадцатеричных значений (FEh)
‘%’	Может использоваться как префикс для объявления двоичных значений (%1101)
‘b’	Может использоваться как суффикс для объявления двоичных значений (1101b)
‘d’	Может использоваться как суффикс для объявления десятичных значений (47d)

У шестнадцатеричных и двоичных значений должен быть либо префикс, либо суффикс. Десятичным значениям суффикс необязателен. После ввода очередного символа необходим пробел. Строки должны быть заключены в двойные или одинарные кавычки.

Данные из этого окна кнопкой **Add** могут быть помещены в буфер предварительно заданных последовательностей данных (**верхнее левое окно**) или кнопкой **Queue** в передающий выходной буфер (**правое нижнее окно**). Кнопка **Queue** используется также для передачи данных из Predefined Sequence в Buffered Sequence.

Кнопка **Delete** используется для удаления последовательности из **Predefined Sequence**

Использование отладчика SPI

Чтобы принять данные с помощью отладчика SPI:

Отображение данных на шине SPI требует подключения **SCK и DIN** терминала к соответствующим контактам схемы. Затем следует установить длину слова, порядок бит, статус **SCK** и настройки модели терминала. Данные на шине SPI будут отображены в окне терминала как шестнадцатеричные значения. Если принятое слово данных неполное, SS становится неактивным до тех пор, пока слово не передастся целиком, отображаться будут звёздочки вместо шестнадцатеричных значений.

Чтобы передать данные с помощью отладчика SPI:

Передача данных по шине требует подключения **SCK и DOUT** терминала к соответствующим контактам схемы. Далее следует установить длину слова, порядок бит, статус **SCK** и настройки модели терминала.

Записи могут быть отдельными словами, запятыми или пробелами, последовательностью слов или текстовых строк, заключенных в двойные или одинарные кавычки. Значения могут быть в десятичном, шестнадцатеричном или двоичном представлении. Шестнадцатеричные значения обозначаются как **"0x'** или с префиксом**' $'**, или суффиксом **'h '**. Двоичные значения обозначаются префиксом **'%'** или суффиксом **'b '**. Для десятичных значений не нужны ни префикс, ни суффикс, но при желании можно добавить суффикс ‘**d '.**

После ввода передаваемой последовательности можно непосредственно подготовить последовательность данных для отправки, поместив их в выходной буфер, или разместить в буфере предварительно заданных последовательностей для дальнейшего использования.

Чтобы скопировать данные из буфера предварительно заданных последовательностей в выходной буфер, убедитесь в том, что поле ввода пусто, выберите последовательность и выполните двойной щелчок или нажмите **Queue.**

Чтобы удалить запись из буфера предварительно заданных последовательностей, выберите определенную последовательность, затем нажмите кнопку **Delete.**

8. Отладчик интерфейса I2C

Интерфейс I2C широко используется для связи микропроцессорных устройств с периферийным оборудованием и организации многопроцессорных систем. Область применения аналогична SPI (датчики, АЦП, ЦАП, мосты, часы реального времени, память и так далее).

Интерфейс I2C синхронный, полудуплексный, обмен по принципу «ведущий – ведомый», число ведущих устройств ограничено только электрическими характеристиками интерфейса, скорость обмена изменяется в широких пределах.

По сравнению с SPI функциональные возможности значительно шире, но сложнее и его реализация. Значительная часть функций

порта I2C микроконтроллера может быть реализована программными средствами. Поэтому при сравнении технических характеристик портов I2C необходимо анализировать соотношение между программными и аппаратными средствами. Чем больше функций реализуется аппаратно, тем больше быстродействие порта. Например, в ATmega8535 с частотой тактового генератора 1 МГц максимальная частота шины I2C равна 55,5 КГц. В SPI при тех же условиях – 500 КГц.

Обмен данными по I2C выполняется кадрами, структура которого приведена на рисунке А.28, где S – стартовый бит (срез), ADR - поле адреса 7 или 15 бит (признаком 15-битовой адресации является наличие кода 11110b в старших битах), R/W*- чтение (1) или запись (0) данных, ACK* - сигнал подтверждения приема или передачи данных, D – 8-разрядное поле данных, P –стоповый бит (фронт)

S	ADR	R/ W*	ACK*	D	ACK*	……..	D	ACK*	P

Рис. А.28. Формат кадра I2C

Отладка устройств с I2C сложнее, чем с SPI. Поэтому применение отладчика значительно сокращает время разработки и тестирования устройств, использующих порты I2C.

К сожалению, протокол, реализуемый отладчиком Proteus, несколько упрощенный. Например, **при отладке программ с микроконтроллерами AVR не формируются коды состояния.**

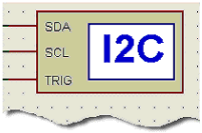


Рис. А.29. Условное изображение терминала I2C

Условное изображение терминала представлено на рисунке А.29, где **SCL** – двунаправленная линия синхронизации, **SDA** – двунаправленная линия данных, TRIG – вход, с помощью которого можно управлять очередью последовательностей данных на выходе терминала I2C.

Терминал выбирается из списка виртуальных устройств (**I2C Debugger**), размещается в поле схемы, соединяется с соответствующими линиями ОМК.

Так как линии SCL и SDA выполнены по схеме с открытым коллектором («монтажное И»), то к ним требуется подключать подтягивающие резисторы.

Двойным щелчком левой кнопки мыши вызывается редактор свойств, в котором задаются параметры терминала, соответствующие режиму работы порта ОМК (рис. А.30).

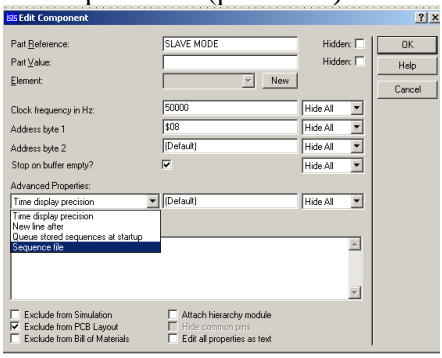


Рис. А.30. Окно свойств терминала

Таблица А.4. Перечень настроек

Clock frequency in Hz	Частота в Гц	50000
Adress byte 1	Байт адреса 1 (при 7-битовой адресации) и направление передачи	$08 (адрес -64, R/W*=0 -запись)
Adress byte 2	Байт адреса 2 (старший байт адреса при 10-битовой адресации)	Default
Stop on buffer empty?	Остановка моделирования, если выходной буфер пуст	Yes,?,No
Time display precision	Точность отображения времени	Default
New line after	Новая строка формируется после заданного числа символов	64
Queue stored sequences at startup	Очередь хранит последовательности при запуске	Default No Yes
Sequence file	Имя текстового файла с заготовленными выходными последовательностями	*File*

После запуска отладки открывается экран аналогичный отладчику SPI (рисунки А.31, А.32). Экран разделен на несколько частей и позволяет контролировать состояние шины, принимать и передавать данные в режиме ведущего или ведомого.

Внимание: во время симуляции с большим количеством действий на шине возможно снижение производительности терминала. Поэтому рекомендуется при отладке отключать устройства, процессы которых не связаны с шиной I2C, и выполнять обратные действия при отладке устройств, не подключенных к I2C.

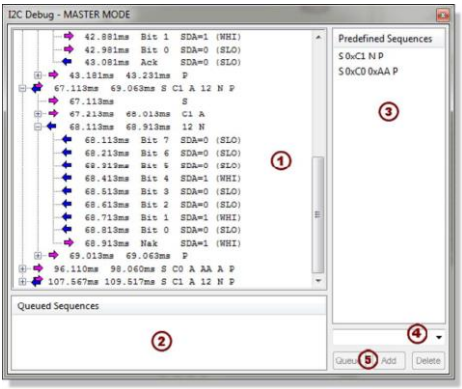


Рис. А.31. Экран отладчика в режиме ведущего

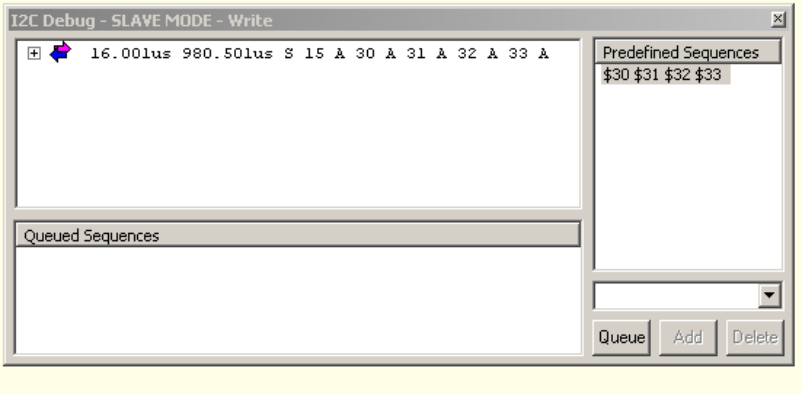


Рис. А.32. Экран отладчика в режиме ведомого

Верхнее левое окно содержит основные данные о работе шины: направление передачи данных и их значения, время возникновения событий. Возможен анализ как байт , так и битовых последовательностей.

Если **терминал** используется **как передатчик**, то **значения данных** задаются в **правом нижнем окне.**

Таблица А.5. Правила

Синтаксис	Описание
‘0x’	Может использоваться как префикс для объявления шестнадцатеричных значений (0xFE)
‘$’	Может использоваться как префикс для объявления шестнадцатеричных значений ($FE)
‘h’	Может использоваться как суффикс для объявления шестнадцатеричных значений (FEh)
‘%’	Может использоваться как префикс для объявления двоичных значений (%1101)
‘b’	Может использоваться как суффикс для объявления двоичных значений (1101b)
‘d’	Может использоваться как суффикс для объявления десятичных значений (47d)

У шестнадцатеричных и двоичных значений должен быть либо префикс, либо суффикс. Десятичным значениям суффикс необязателен. После ввода очередного символа необходим пробел. Строки должны быть заключены в двойные или одинарные кавычки.

Данные из этого окна кнопкой **Add** могут быть помещены в буфер предварительно заданных последовательностей данных (**верхнее левое окно**) или кнопкой **Queue** в передающий выходной буфер (**правое нижнее окно**). Кнопка **Queue** используется также для передачи данных из **Predefined Sequence** в **Buffered Sequence**.

Кнопка **Delete** используется для удаления последовательности из **Predefined Sequence**

Таблица А.6. Синтаксис для индикации

Syntax	**Description**
S	Начальное состояние (стартовый бит)
Sr	Состояние рестарта
P	Конечное состояние (стоповый бит)
N	Отказ в доступе/ отсутствие подтверждающего сигнала ACK*

A	Доступ разрешён/ ACK* получен
L	Потеря арбитража, переход в режим мастера
*	Получена часть данных
&	Обнаружены ошибочные логические уровни

Свойства модели

I2C Debugger имеет ряд настроек, позволяющих конфигурировать его при необходимости. Все настройки доступны через функцию **Edit Component** .

Таблица А.7. Синтаксис для индикации

Свойство	Описание
ADRESS1	Если терминал используется для моделирования ведомого устройства, эта настройка обозначает первый байт адреса ведомого устройства. Младший бит используется мастером, чтобы указать направление передачи (чтение или запись). Если оставить это свойство по умолчанию или пустым, то терминал будет работать как ведомое устройство.
ADRESS2	При использовании терминал в качестве ведомого устройства задаёт второй байт адреса при 10 битной адресации. Пустое поле означает 7-битную адресацию.
STOPONEMPTY	Указывает, что моделирование должно быть приостановлено, когда выходной буфер пуст, и требуется байт для отправки
CLOCKFREQ	В режиме ведущего устройства управляет тактовой частотой на линии SCL
SEQUENCE_FILE	Позволяет указать имя текстового файла, в котором хранятся заранее заданные выходные последовательности. Если поле пустое, то сохраняются последовательности, заданные при настройке терминала.
AUTOLOAD	Контролирует, будут ли заготовленные последовательности немедленно помещены в выходную очередь при начале симуляции.
TIMEPREC	Количество знаков после запятой в отображаемом времени.
WRAPLENGTH	Максимальное количество элементов в строке.

Применение отладчика I2C

Прием данных: Входные данные будут отображаться в верхней левой части окна анализатора, как **шестнадцатеричные значения и специальные символы,** соответствующие формату обмена.

Передача данных: Запись передаваемых данных осуществляется в нижней правой стороне окна терминала. Записями могут быть отдельные слова, разделенные запятой или пробелом последовательности слов или текстовые строки, заключенные в двойные или одинарные кавычки. Словом, могут быть числовые значения или специальные символы управления.

После ввода передаваемой последовательности она может быть сохранена в буфере предварительно заданных последовательностей (кнопка **Add**) или в выходном буфере (кнопка **Queue**).

Чтобы скопировать одну из заготовленных последовательностей в выходной буфер следует убедиться, что поле ввода последовательности пустое, выбрать заготовленную последовательность, затем нажать кнопку Q**ueue.**

Чтобы удалить запись из заготовок, необходимо выбрать запись, которую требуется удалить, и нажать кнопку **Delete**.

При работе в **режиме ведомого не отражаются символы, которые не формирует терминал: при приеме** в поле **ACK*** записывается **N**, хотя данные фиксируются правильно. При **передаче** не отражается признак окончания кадра **P**, а данные в ведущий ОМК передаются корректно.

Для задания **режима ведущего терминала**. необходимо установить порт ОМК в режим **ведомого**, а в терминале сформировать соответствующий кадр, используя приведенный выше синтаксис.

9. Отладчик асинхронного порта UART (USART)

Отладчик (терминал) асинхронного порта **(Virtual Terminal**) используется при разработке и отладке микропроцессорных устройств, в состав которых входит последовательный порт UART (USART), работающий в **асинхронном режиме**. **Синхронный режим** этих портов терминалом не поддерживается, но его можно отлаживать и контролировать с помощью паттерн-генератора, осциллографа или логического анализатора.

При отладке терминал можно использовать для ввода/вывода числовых данных и диагностических сообщений, формируемых в процессе работы микропроцессорного устройства.

Применение соответствующих преобразователей уровня позволяет с помощью терминала отлаживать устройства с интерфейсами RS-232, RS-485, RS-422, имеющими аналогичный формат кадра.

Характеристики терминала:

- режимы обмена – дуплексный, симплексный. **Выходная** информация отображается в виде hex-кода или ASCII символов, **входная** - передаётся как последовательность ASCII символов;
- связь с ОМК выполняется по линиям: **RXD** – вход приемника, **TXD** – выход передатчика, **RTS** (выход) – запрос передачи (готовность терминала к приему данных); **CTS** (вход) – разрешение передачи;
- возможность работы в режимах - с программным методом управления потоком (**XON/XOFF**) и аппаратным (**RTS /CTS**);
- скорость передачи - (110 -115200) бит/с или задается при настройке;
- регулируемый формат кадра: поле данных - 7 или 8 бит,1 или 2 стоповых бита, бит паритета –четность, нечетность, отсутствие контроля;
- возможность задания нормальной и инвертированной полярности для **RX/TX** и **RTS/CTS**;
- кодирование информации – ASCII, DUMB, ANSI, VT52, VT100;
- терминал поддерживает управляющие ASCII коды CR (0x0D) (возврат каретки), BS (0x08) (возврат) и BEL (0x07) (звуковой сигнал). Все другие коды, включая перевод строки LF (0x0A), игнорируются.

Терминал – это просто цифровая модель и не требует никаких специфических уровней напряжения на своих выводах, то есть он будет функционировать как с микроконтроллерами или другими универсальными асинхронными приемопередатчиками, так и с преобразователями уровней RS-232 или RS-485;

- исходный уровень на выводах RX и TX по умолчанию высокий. Таким образом, состояние холостого хода – высокий уровень, стартовый бит – низкий уровень, а стоповый бит – высокий

уровень. При кодировании данных значение 0 передается низким уровнем, а 1- высоким. Это совместимо со встроенными модулями UART во многих микроконтроллерах, и также с внешними универсальными асинхронными приемопередатчиками типа 6850 и 8250. В других случаях (например, при подключении преобразователя уровня RS-232 или RS-485), необходимо установить полярность RX/TX инверсной;

- **аппаратный метод** управления потоком используется, если необходима **быстрая реакция** на состояние приемника. Однако для его **реализации требуется 5 линий связи.** На выводах RTS и CTS по умолчанию исходный уровень высокий. Если необходимо подключиться к инвертированным шинам управления, полярность RTS/CTS должна быть отрицательной;
- **программный метод** управления потоком применяется, если необходимо минимизировать количество линий связи и невысоких требованиях по быстродействию**. Обмен информацией выполняется только с помощью линий TxD, RxD и земляного провода.** Передача знака XOFF (код DC3 ASCII или 13h) прекращает передачу данных, а знака XON (код DCl ASCII или 11h) сообщает о необходимости возобновить передачу;
- по умолчанию, терминал не отображает передаваемые символы. Предполагается, что отображение информации выполняется в устройстве, к которому он подключен. Для отображения информации в терминале необходимо выбрать **Echo Typed Characters** в контекстном меню;
- если необходимо вывести текст при запуске, то это можно выполнить, используя настройку ТЕКСТ (TEXT), например, TEXT="Hello World" пошлет в схему текст "Hello World". Текст скрипта посылается один раз после запуска, если сброс передатчика CTS не неактивен.

Условное изображение терминала представлено на рисунке А.33.

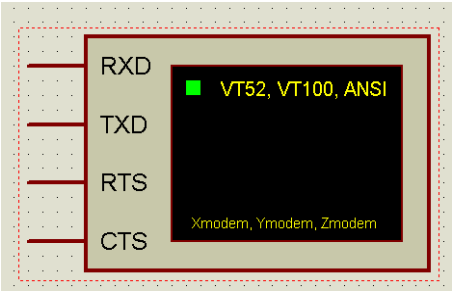


Рис. А.33. Условное изображение терминала

Для подключения терминала необходимо:

- выбрать иконку «Виртуальные инструменты»;
- в открывшемся меню выбрать **VIRTUAL TERMINAL** и поместить его на схему;
- соединить выводы **RX и TX** терминала с соответствующими выводами тестируемого устройства.

Если устройство использует **аппаратный метод управления потоком**, то необходимо соединить **RTS и CTS** с соответствующим линиям: **RTS** как выход, сигнализирующий, что терминал готов к получению данных, а **CTS** как вход, который при передаче информации должен быть установлен в «1».

Рис. А.34. Окно редактирования свойств терминала

При программном управлении эти линии не подключают к устройству.

В обычном режиме линии **RTS, CTS** должны быть соединены между собой.

Двойным щелчком правой кнопкой мыши открыть окно редактирования параметров (рис. А.34) и задать требуемые параметры, согласуя их с настройками порта ОМК (таблица А.2).

Таблица А.8. Настройки терминала

Baud Rate	Скорость передачи, бод	Default 110- 115200	
Data Bits	Количество битов данных	Default,7, 8	
Parity	Бит четности	Default NONE EVEN ODD	 Отсутствует Четный Нечетный
Stop Bits	Стоп-биты	Default, 1, 2	
Send XON/XOFF	Передавать последовательности XON/XOFF	Default, No, Yes	
Terminal Type	Тип терминала	Default, TELETYPE, DUMB, ANSI,VT52, VT100	
RX/TX Polarity	Полярность RX/TX	Default, Normal, Inverted	
RTS/CTS Polarity	Полярность RTS/CTS	Default, Normal, Inverted	
Trace Events	Трассировка событий (Отображаемые события)	Disabled, Warnings Only, Trace Messages, Debug Messages	
Auto Line Feed	Автоматический перевод строки	Default, No, Yes	
Local Echo	Локальное эхо	Default, No, Yes	
Cursor Visible	Видимость курсора	Default, No, Yes	
Show Hints	Показывать подсказки	Default, No, Yes	
Columns number	Количество столбцов	Default, 80, 40	
Hide Toolbar	Скрывать панель инструментов	Default, No, Yes	
ASCII Charset	Набор символов ASCII	Default, No, Yes	
Freeze Font	«Заморозка» шрифта	Default, No, Yes	

Запустить симуляцию Входная информация будет отображаться в окне терминала. Для передачи символов следует убедиться, что выбрано окно терминала и вводится нужный текст с клавиатуры.

После начала симуляции можно кликнуть правой кнопкой на окне терминала для доступа к контекстному меню (рис. А.35). Это меню позволяет очистить экран, редактировать входную и выходную и выходную информацию:

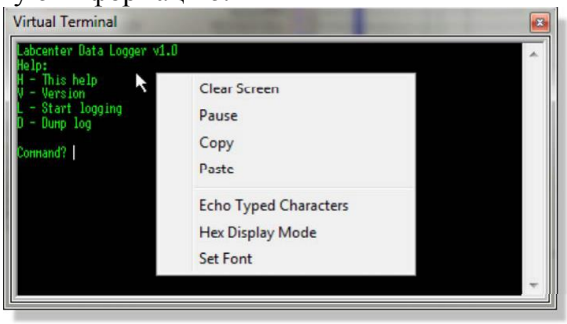


Рис. А.35. Меню экрана терминала

Пример подключения виртуального терминала к MAX232, преобразующего уровни ТТЛ/КМОП в сигналы RS232 (+/- 12 В), показан на рисунке А.36, где **TiIN, TiOUT** – соответственно вход ТТЛ и выход RS232, **RiIN, RiOUT** – соответственно вход RS232 и выход ТТЛ, i=1,2.

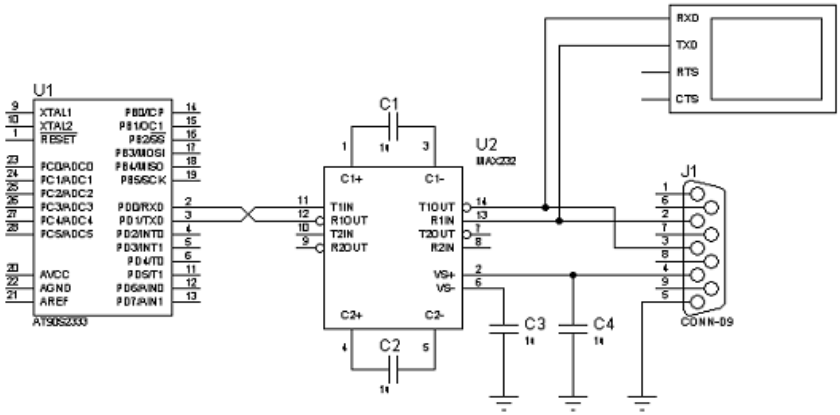


Рис. А.36. Подключение терминала к RS-232

Необходимо помнить, что MAX232 содержит логические инверторы, и поэтому нужно изменить полярность у выводов RX/TX виртуального терминала, для корректного обмена.

MAX232 смоделирован только на цифровом уровне. В модели отсутствует логика работы его внутренних преобразователей напряжения и т. д. Подобная симуляция очень сильно бы повлияла на производительность, при очень малой получаемой выгоде. Подобное ухудшение производительности произойдёт, если подключить любой аналоговый компонент (резистор, конденсатор, осциллограф и т. д.) к выводам **TxOut/RxIn**.

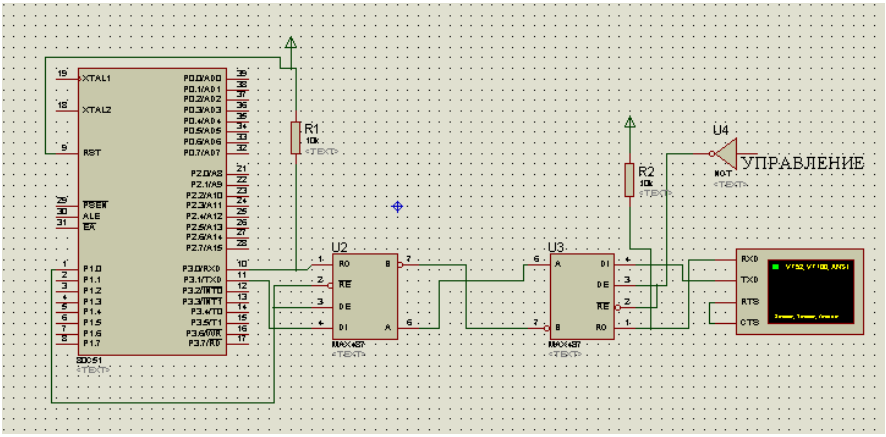


Рис. А.37. Схема соединения терминала с интерфейсом RS-485

На рис. А.37 показана схема соединения терминала с интерфейсом RS-485. Преобразователь уровня MAX487 преобразует уровни ТТЛ в дифференциальный сигнал, где **A, B** – соответственно неинвертирующий/инвертирующий вход/выход, **RO** — выход приемника, **RE/** — разрешение выхода приемника при **RE/=0** (при **RE/=1** выход **RO** находится в высокоимпедансном состоянии), **DI** — вход передатчика, **DE** — разрешение выходов передатчика при DE=**1** (если **DE=0**, то выходы находятся в высокоимпедансном состоянии). Так как интерфейс RS-485 полудуплексный, то уровни сигналов на входах **RE/, DE** определяют направление передачи информации. Подтягивающие резисторы **R1, R2** обеспечивают защиту от помех. При моделировании в Proteus их можно не подключать.

Методические материалы

РАЗРАБОТКА И ОТЛАДКА МИКРОПРОЦЕССОРНЫХ СИСТЕМ В ВИРТУАЛЬНОЙ СРЕДЕ МОДЕЛИРОВАНИЯ PROTEUS

Методические указания

Составитель ***Иоффе Владислав Германович***

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ имени академика С. П. КОРОЛЕВА»
(Самарский университет)
443086, САМАРА, МОСКОВСКОЕ ШОССЕ, 34.

Изд-во Самарского университета.
443086, Самара, Московское шоссе, 34..