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

ОТЛАДКА ПРОГРАММ НА СИМУЛЯТОРЕ VMLAB 3.12

Самара 2015

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

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

ОТЛАДКА ПРОГРАММ НА СИМУЛЯТОРЕ VMLAB 3.12

Составитель *В.Г. Иоффе*

Самара 2015

УДК СГАУ: 004.312

Составитель ***В.Г. Иоффе***

Рецензент: доцент, к.т.н. Зеленко Л.С.

Отладка программ на симуляторе VMLab 3.12 [Электронный ресурс]: метод. указания / сост. *В. Г. Иоффе*. М-во образования и науки РФ, Самар. гос. аэрокосм. ун-т им. С.П. Королева (нац. исслед. ун-т). Электрон. текстовые и граф. дан.(1,23 Мбайт).- Самара, 2015.- 88 с., ил. 1 эл.опт. диск (CD-ROM).

Методические указания содержат описание симулятора VMLab 3.12 и технологию его использования при программировании однокристальных контроллеров AVR Atmel на ассемблере. Приведены примеры программирования для ATmega8535.

Методические указания предназначены для бакалавров, обучающихся по направления 09.03.01–"Информатика и вычислительная техника.

Электронные методические указания разработаны на кафедре информационных систем и технологий.

Стр.88, Ил.36, , Библ.4 назв.

УДК СГАУ: 004.312

Содержание

Введение

При проектировании микропроцессорных систем и выборе соответствующих МП и ОМК основное внимание следует уделять средствам, обеспечивающим разработку и отладку систем, так как грамотный выбор этих средств позволяет сократить время разработки в несколько раз. Особенно трудоемка разработка программного обеспечения, которая составляет 90-95% стоимости проекта.

Средства разработки можно разделить на программные и программно-аппаратные.

К числу программных средств относятся ассемблеры, макроассемблеры, компиляторы, в состав которых обязательно должен входить симулятор (эмулятор). Разработка программ выполняется на ЭВМ.

Симулятор – программное средство, способное имитировать работу МП или ОМК и его памяти. Как правило, в состав симулятора входят: модели процессора, памяти и периферийных устройств, отладчик, выполняющий функции управления процессом отладки и анализа результатов моделирования.

Развитые симуляторы содержат в своём составе также модели внешних устройств (АЦП, ЦАП, усилители, клавиатуру и так далее), средства моделирования сигналов (генераторы сигналов различной формы, регулируемые источники постоянного напряжения) и отображения результатов моделирования: алфавитно-цифровые индикаторы, осциллографы, анализаторы эффективности разработанной программы (профайлеры) и так далее.

Наиболее эффективными способами разработки и отладки программ является использование графических сред и языков высокого уровня (СИ, Pascal, Fort).

Для относительно простых задач используются графические среды разработки, программирование в которых сводится к созданию граф-схем алгоритмов на основании определенных правил. Примером подобной среды для микроконтроллеров фирмы ATMEL с архитектурой AVR является «Algorithm Builder», предназначенная для производства полного

цикла разработки начиная от ввода алгоритма, включая процесс отладки и заканчивая внутрисхемным программированием кристалла.

В отличие от классического ассемблера программа вводится в виде алгоритма с древовидными ветвлениями и отображается на плоскости, в двух измерениях. Сеть условных и безусловных переходов отображается графически, в удобной векторной форме. Это к тому же освобождает программу от бесчисленных имен меток, которые в классическом ассемблере являются неизбежным балластом. Вся логическая структура программы становится наглядной.

Графические технологии раскрывают новые возможности для программистов. Визуальность логической структуры уменьшает вероятность ошибок и сокращает сроки разработки. Появляется такое понятие, как дизайн алгоритма, предполагающее некоторый художественный вкус программиста.

По оценке пользователей, по сравнению с классическим ассемблером, время на разработку программного обеспечения сокращается в 3-5 раз, что делает его в этом соизмеримым с языком СИ, при этом эффективность кода несравненно выше.

При выборе программных средств разработки необходимо контролировать наличие библиотеки стандартных функций.

Недостаток программных средств – невозможность отладки программ в режиме реального времени с учетом физических особенностей конкретного МП и ОМК.

Эти недостатки устраняются при использовании программно-аппаратных средств.

Наиболее простым и дешевым средством этого класса являются оценочные платы (EVB) или модули (EVM). В их состав входят: ОМК или МП, ОЗУ, РПЗУ, средства контроля и управления, средства связи с ЭВМ, макетное поле или разъём для устройств пользователя. Программа, отлаженная на ЭВМ, по каналу связи загружается в оценочный модуль, а далее производится отладка на реальном оборудовании.

Оценочные платы могут использоваться в качестве контроллеров автоматизированных систем.

Лучшими характеристиками обладают внутрисхемные эмуляторы (ВСЭ). ВСЭ – программно-аппаратное средство, способное замещать собой эмулируемый МП или ОМК в реальной схеме. Его применение делает процесс разработки и отладки МПС легко контролируемым и управляемым. Стыковка ВСЭ с отлаживаемой системой производится при помощи кабеля со специальной эмуляционной головкой, которая вставляется вместо МП или ОМК в отлаживаемую систему. Как минимум, ВСЭ содержит средства разработки и отладки программ, загрузчик, эмулятор ПЗУ, средства контроля и диагностики состояния отлаживаемой системы. ВСЭ могут быть выполнены в конструктиве ПЭВМ или в виде выносного блока с последовательным или параллельным каналом связи. Однако это достаточно дорогое устройство - (1 – 10) тысяч долларов.

При разработке и отладке программ на ассемблере микроконтроллеров семейства AVR Atmel наиболее широкое распространение получили симуляторы AVR Studio, VMLab, Proteus.

К недостаткам AVR Studio следует отнести практически отсутствие средств моделирования и контроля внешних устройств.

VMLab лишен этих недостатков. Пользователю предоставлена обширная библиотека различных внешних компонентов , включая осциллограф, терминалы для работы с последовательными интерфейсами и так далее. Разработчики VMLab предлагают инструментальные средства для программирования пользовательских компонентов, не входящих в состав библиотеки симулятора [1].

В методических указаниях рассмотрен симулятор VMLab 3.12, распространяемый свободно в Интернете.

VMLab 3.12 – полностью интегрированная профессиональная среда разработки.

Система поддерживает следующие основные особенности:

- **Язык программирования ассемблер или СИ AVR**.
- **Объектные файлы**. После компиляции проекта создаётся obj файл, который может быть

подключен к другим проектам без вставки ассемблерного кода.

- **Библиотечные модули**. VMLab поддерживает концепцию библиотечных модулей. Библиотечный модуль – стандартный OBJ файл, который подключен к проекту. Это позволяет создавать свои библиотеки кодов, которые могут использоваться в других проектах.
- **Мощный многооконный и многофайловый редактор**. Проект разбивается на несколько файлов, следуя логике – файл кода, файл проекта с обозначенными используемыми компонентами и т.д. Дополнительно, редактор выделяет жирным шрифтом ключевые слова, комментарии – серым цветом, а директивы и используемые компоненты – зелёным цветом для удобства чтения;
- **Текстовый отладчик с проверкой кода**. После компиляции проекта результирующий код автоматически загружается в симулятор и тестируется средствами VMLab. Правильные и неправильные строки кода помечаются разным образом. Во время отладки на тех строках кода, которые исполнялись, дорисовывается жёлтая полоса, и она тем длиннее, чем больше раз исполнялась та или иная строка кода. Это очень удобно при отладке, т.к. сразу можно заметить, в каком месте программа зациклилась, а до какого места не дошла.

При работе с симулятором разработчику предоставлен доступ к любому ресурсу микроконтроллера, имеется возможность покомандного и пофрагментного исполнения программ, останова по условию, подсчета числа тактов выполнения тех или иных фрагментов программы, инициирования прерывания, дизассемблирования содержимого памяти программ и т.д.

Симулятор позволяют промоделировать практически все возможные варианты работы программы и, тем самым, убедиться в ее работоспособности. Однако, для наиболее полной доводки прикладного программного обеспечения микроконтроллера

необходимы комплексные и всесторонние испытания разработанной системы в реальном окружении и во всевозможных режимах. Это объясняется тем, что симулятор не учитывает реальные задержки, возникающие в аппаратуре в процессе передачи сигналов. Оценка временных характеристик выполняется только в модельном времени.

Методические указания разработаны, в основном, с использованием фирменного описания VMLab 3.12.[1]

1 Основное окно программы

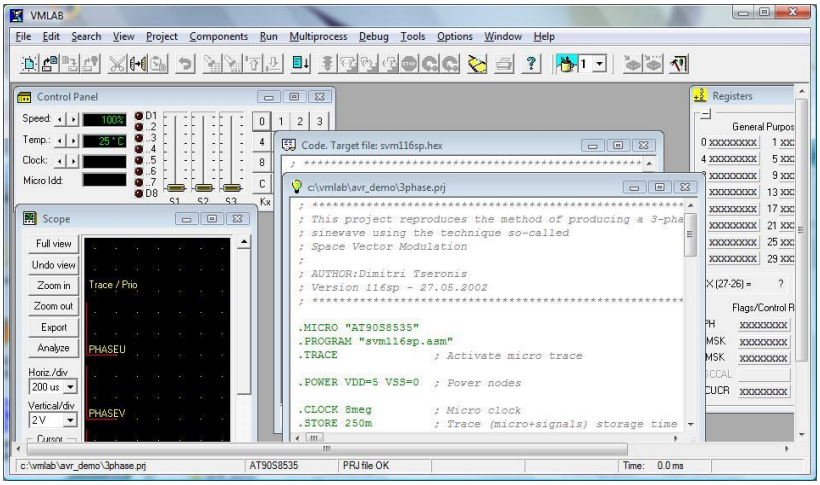

Рис. 1. Основное окно программы

Основное окно программы (рис.1)состоит из следующих основных частей

- Главное меню программы;
- Панель функциональных кнопок;
- Рабочий стол программы с набором необходимых окон;

Рассмотрим последовательно все перечисленные части программы.

1.1 Главное меню программы

Рис. 2. Главное меню программы

Главное меню программы(рис.2) состоит из следующих пунктов;

- **File**. Содержит основные команды для работы с файлами;
- **Edit**. Редактирование, вставка и др. операции для работы с текстом программы;
- **Search.** Содержит основные опции поиска и замены фрагментов текста программы.
- **View**. Набор основных окон, необходимых для отладки программы;
- **Project**. Основные настройки по управлению проектом;
- **Components**. Содержит основные встроенные компоненты для подключения к проекту;
- **Run.** Данный пункт содержит команды для компиляции, выполнения, пошаговой отладки программы;
- **Multiprocess.** Запускает второй VMLab для моделирования мультипроцессного режима.
- **Debug.** Содержит инструменты для отладки программы – управление точками останова, менеджером наблюдения и т.д.;
- **Tools.** Содержит полезные дополнительные сервисные инструменты;
- **Options.** Пункт содержит основные IDE;
- **Window.** Позволяет манипулировать размещением окон на экране;
- **Help.** Справка по программе.

1.2 Панель функциональных кнопок

Рис. 3. Панель функциональных кнопок

На рис.3 отражены следующие окна:

 - создание нового текстового файла;

 - открытие нового файла;

 - сохранение файла на диске;

 - открытие нового файла проекта;

 - удаление выделенного текста;

 - копирование выделенного текста;

 - вставка текста из буфера;

 - отмена предыдущей операции;

 - поиск заданного фрагмента текста;

 - поиск следующего фрагмента заданного текста;

 - переход к предыдущей метке;

 - переход к следующей метке;

 - сборка проекта;

 - запуск/продолжение выполнения программы;

 - шаговый режим без захода в подпрограмму (блок);

 - шаговый режим выполнения программы;

 - шаговый режим с заходом в подпрограмму (блок); выполняется одна команда (если команда представляет собой вызов подпрограммы, то

подпрограмма выполняется, а указатель команд встает на конец подпрограммы);

 - остановка выполнения программы;

 - рестарт программы в симуляторе с сохранением данных в памяти;

 - рестарт программы в симуляторе со сбросом данных в памяти;

 - включение/выключение анимации кода (При включении - в процессе выполнения программы выделяется цветом исполняемая строка);

 - печать файла, на окне которого стоит курсор;

 - вызов справки;

 - выбор рабочего стола программы;

 - проверка флэш-памяти ОМК с использованием STK500;

 - проверка ОМК автоматической последовательностью сигналов;

 - запуск инструментов пользователя.

Команда  (или ***Запуск /Перезапуск (глубокий) (Shift + F8)***) останавливает процесс отладки программы, устанавливает в исходное значение (XX) все регистры и загружает в счетчик команд **PC** в нулевое значение.

Команда  (или ***Запуск /Перезапуск (лёгкий) (F8)***) останавливает процесс отладки программы, сбрасывает счетчик команд **PC** в нулевое значение, но сохраняет последние значения регистров и памяти данных, полученные в ходе выполнения программы.

По команде  (***Запуск / Начать/Продолжить (F5)***) начинается выполнение программы без остановки или до первой

точки останова. Приостановить выполнение (пауза) можно по команде  или из пункта меню ***Запуск / Приостановить)***

1.3 Пункты главного меню

File

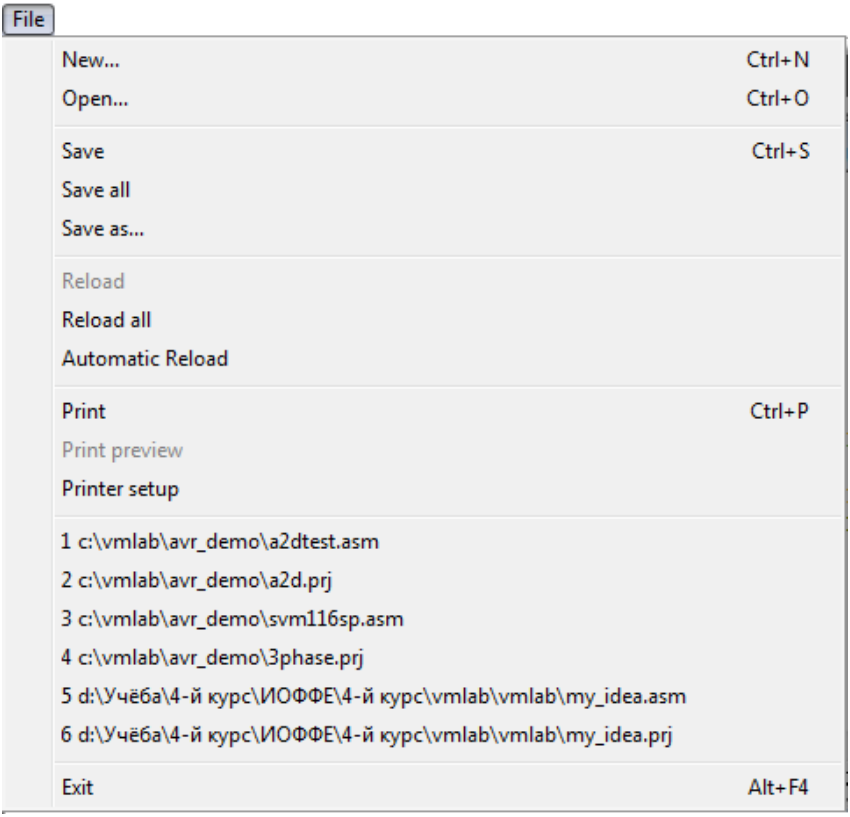

Рис. 4. Пункты подменю File (Файл)

На рис.4 показаны :

New – Создание нового файла.

Open – Загрузить файл с диска.

Save – Сохранить изменения в текущем файле.

Save all – Сохранить изменения во всех открытых окнах с текстом программы.

Save as – Сохранить текущий текст программы под другим именем.

Reload – Перезагрузить открытый файл с диска, не сохраняя изменений.

Reload all – Перезагрузить все файлы проекта, не сохраняя изменений.

Automatic Reload – Если отметить «да», то все файлы будут перезагружаться автоматически с диска каждый раз, когда они будут изменены любым другим приложением; «нет» - вы будете информироваться каждый раз при возникновении такой ситуации.

Print – Печать текста программы.

Print preview – просмотр печатаемого документа.

Print setup – настройки для печати документа. При выборе этого пункта меню появляется окно, в котором необходимо выбрать настройки печати и параметры страницы (рис. 5).

Exit – Выход из программы.

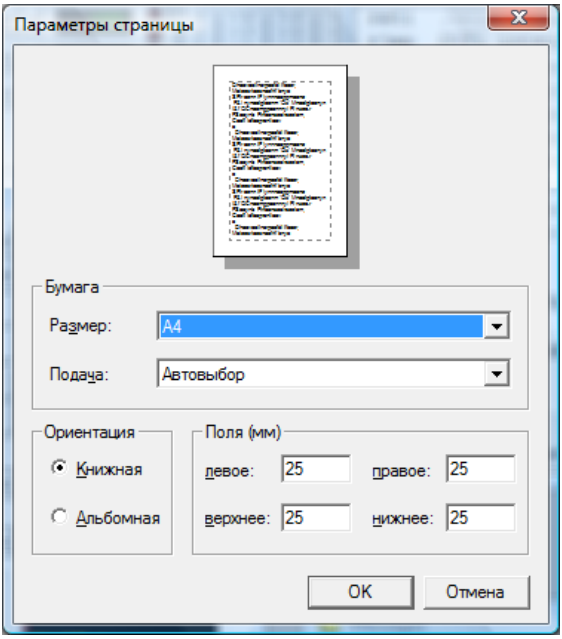

Рис. 5. Настройки печати

Edit

Пункт меню Edit относится к работе с текстом программы (рис.6).

Undo – Отменить последнее действие.

Cut – Вырезать фрагмент текста и поместить в буфер обмена.

Copy – Копировать фрагмент текста в буфер обмена.

Paste – Вставить фрагмент текста из буфера обмена.

Clear – Удалить фрагмент текста.

Select all – Выделить весь текст.

Auto indent – Если выбран этот пункт меню, то текст будет автоматически выравниваться по предыдущей строке, образуя «лесенку». Если не выбран – каждая строка будет начинаться от левого края окна редактирования.

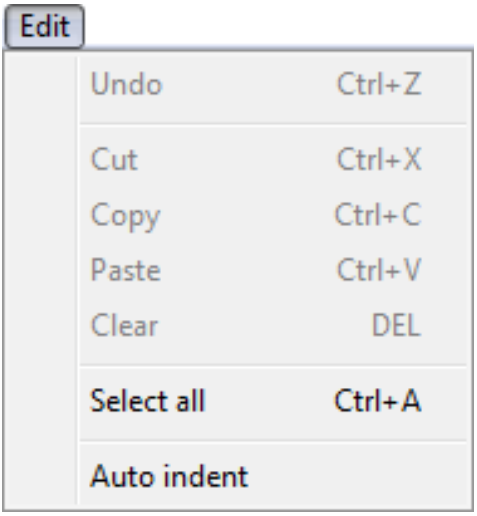

Рис. 6. Пункты подменю Edit (Редактирование)

Search

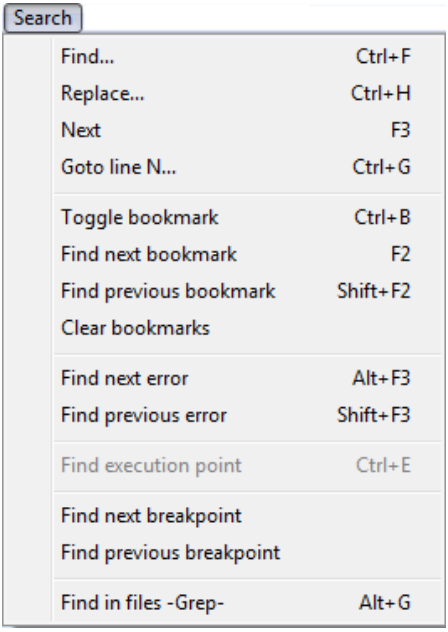

Рис. 7. Пункты подменю Search (Поиск)

Пункт меню Search содержит опции для поиска фрагментов текста и точек останова в программе (рис.7).

Find – Поиск определённого фрагмента текста.

Replace – Поиск определённого фрагмента текста и замена другим текстом.

Next – Поиск следующего фрагмента текста, удовлетворяющего условиям поиска.

Goto line N – Переход к строке с номером N.

Toggle bookmark – Переход к первой метке в тексте программы.

Find next bookmark – Переход к следующей метке по тексту программы.

Find previous bookmark – Переход к предыдущей метке по тексту программы.

Clear bookmarks – Удаление всех меток по тексту программы.

Find next error – Поиск следующей ошибки в тексте откомпилированной программы.

Find previous error - Поиск предыдущей ошибки в тексте откомпилированной программе.

Find execution breakpoint – Переход к строке, на которой остановилась программа.

Find next breakpoint – Переход на следующую строку, где поставлена точка останова.

Find previous breakpoint - Переход на предыдущую строку, где поставлена точка останова.

Find in files -Grep- – Поиск указанного текста в файле (файлах).

View

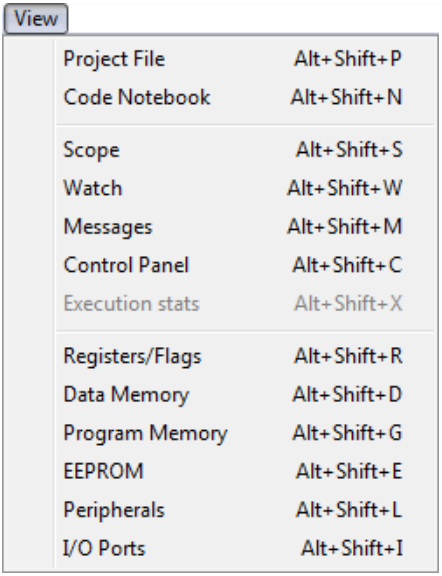

Рис. 8. Пункты подменю View (Просмотр)

Этот пункт содержит все окна, необходимые для работы с программой (рис8).

Project File – Окно проекта.

Code notebook – Окно редактирования кода.

Scope – Окно осциллографа.
Watch – Окно слежения.
Messages – Окно сообщений.
Control Panel – Окно панели управления.
Execution states – Окно порядка выполнения.
Registers/Flags – Окно регистров и флагов.
Data Memory – Окно памяти данных.
Program memory – Окно кодов.
EEPROM – Окно электрически перепрограммируемого ПЗУ.
Peripherals – Окно периферийных устройств.
I/O Ports - Окно портов ввода-вывода

Project

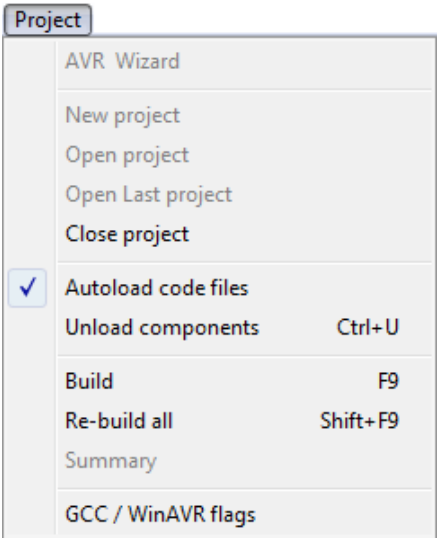

Рис. 9. Пункты подменю Project (Проект)

Пункт Project (рис.9) содержит средства для создания, редактирования и управления проектом.

AVR Wizard – Мастер создания проекта для ОМК AVR (доступен только в зарегистрированной версии программы).

New project – Создание нового проекта.

Open project – Открыть существующий проект.

Open Last project – Открыть проект, который был закрыт последним.

Close project – Закрыть текущий проект.

Autoload code files – При выбранном пункте файл кода будет загружаться автоматически вместе с файлом проекта

Unload components – Отключение компонентов

Build – собрать проект. Компиляция и линковка всех файлов, входящих в проект.

Re-build all – перекомпиляция всех файлов проекта.

Summary – вывод подробных сведений о статусе проекта.

GCC/WinAVR flags – настройка параметров взаимодействия VMLAB и компилятора для C GNU (GCC)

Components

Этот пункт меню (рис.10) предназначен для добавления различных компонентов к проекту.

Resistor –внешнего резистора**.**

Capacitor –Конденсатора с заземлением**.**

LED diode –Светодиода**.**

Key –интерактивного переключателя, контролируемого одной из 16 кнопок контрольной панели**.**

Opamp –операционного усилителя**.**

Comparator –компаратора.

A/D Converter –аналого-цифрового преобразователя АЦП (недоступно в незарегистрированной версии).

D/A Converter –цифро-аналогового преобразователя ЦАП.

Inverter –инвертора (недоступно в незарегистрированной версии).

NAND 2 –двухвходового элемента И-НЕ

NOR 2 –двухвходового элемента ИЛИ-НЕ (недоступно в незарегистрированной версии)

V Sinewave –генератора синусоидального напряжения

V Pulse –генератора импульсного напряжения

V PWL –разъема питания PWL (недоступно в незарегистрированной версии)

V Slider –движкового генератора напряжения (интерактивного)

V Digital NRZ –цифрового формирователя сигнала

TTY – интерактивного телетайпа на RS232

LCD module –ЖКИ модуля

I2C monitor –интерактивного I2C монитора

Keypad –интерактивной клавиатуры 4x4

TTY 2 (big screen) –интерактивного телетайпа на RS232 с большим экраном отображения

Multiprocess – Включение второгоVMLAB для имитации мультипроцессорного режима (дублирует одноимённый пункт меню).

Create new – Создание нового компонента (см. файл «Создание новых компонентов»)

Рис. 10. Пункты подменю Components (Компоненты)

Run

Этот пункт меню (рис.11) позволяет управлять процессом отладки программы.

Go / Continue – Начать/продолжить симуляцию

Pause program – Приостановить выполнение программы

Restart (light) – Рестарт программы в симуляторе с сохранением данных в памяти, счётчик команд сбрасывается в 0

Restart (deep) - Рестарт программы в симуляторе со сбросом данных в памяти, счётчик команд сбрасывается в 0

Step over - Выполнение одной команды без захода в подпрограмму

Step into - Выполнение одной команды с заходом в подпрограмму ; шаговый режим

Step out - Выполнение одной команды (если команда представляет собой вызов подпрограммы, то подпрограмма выполняется, а указатель команд встает на конец подпрограммы);

Run to Cursor – Выполнение программы до той строки, на которую поставлен курсор

Animate code – Анимация кода, в процессе выполнения программы выделяется цветом исполняемые строки.

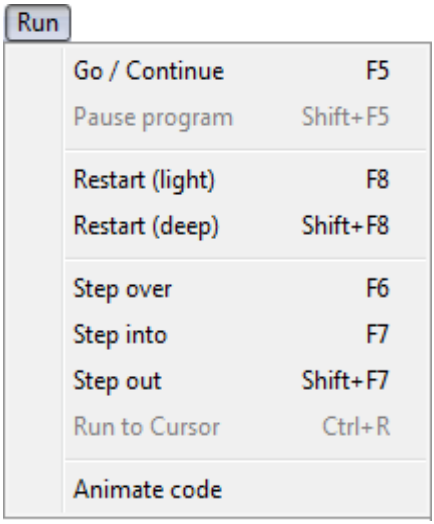

Рис. 11 – Пункты подменю Run (Запуск)

Multiprocess

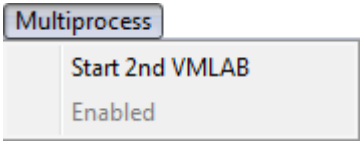

Рис. 12. Пункты подменю Multiprocess (Многопроцессорный режим)

Этот пункт меню (рис12)предназначен для работы в мультипроцессорном режиме.

Start 2nd VMLAB – Открытие второго VMLAB.

Enabled – Показывает, открыт ли второй VMLAB (если да, то ставится галочка)

Debug

Возможности режима отладки показаны на рис.13.

Watch manager – Менеджер наблюдения, позволяет отслеживать значение выбранных переменных.

Remove watch – Отмена наблюдения

Remove all watches – Отмена всех наблюдений

Toggle breakpoint – Переход к точке останова

Remove all breakpoints – Убрать все точки останова

Enable all breakpoints – Активизация всех точек останова

Disable all breakpoints – Отключение всех точек останова

Show all breakpoints – Показ всех точек останова

Break if micro runtime error – При выбранном пункте меню будет происходить прерывание исполнения программы, если обнаружена ошибка в процессе выполнения программы

Clear all on-R/W breaks – Очищение всех R/W-меток

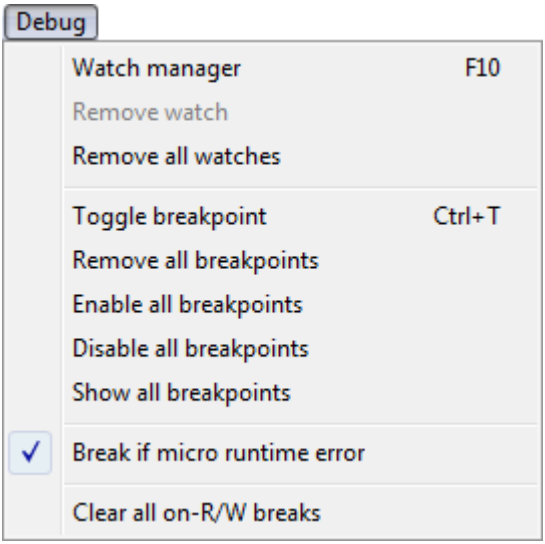

Рис. 13. Пункты подменю Debug (Отладка)

Tools

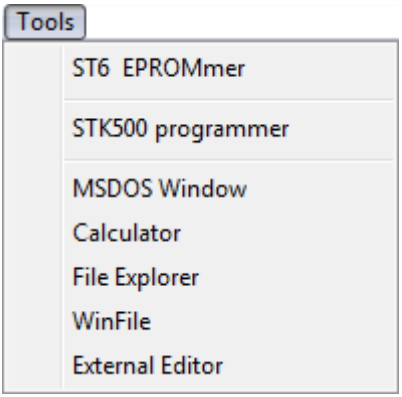

Рис. 14. Пункты подменю Tools (Инструменты)

Этот пункт содержит полезные дополнительные сервисные инструменты (рис.14).

ST6 EPROMmer – Запуск микро EPROM/OTP инструмента программирования (нужна дополнительная установка этого инструмента)

STK500 programmer - STK500 программатор

MSDOS Window – Окно MSDOS (командная строка)
Calculator – Калькулятор Windows
File Explorer – Файловый менеджер (окно Windows)
WinFile – Запуск файла
External Editor – Внешний редактор (блокнот Windows)

Options

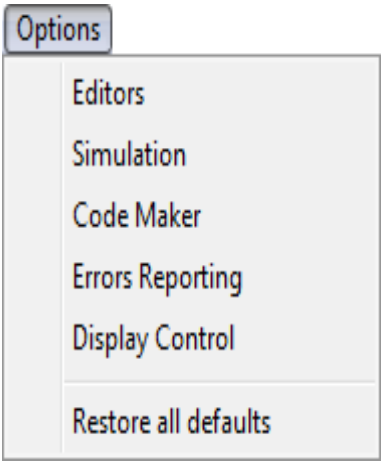

Рис. 15. Пункты подменю Options (Настройки)

Пункт (рис.15) содержит основные IDE (Integrated Development Editors).

Editors –Редакторы (вызов окна настроек текстовых редакторов программы)

Simulation – Эмуляция (вызов окна настроек времени исполнения)

Code Maker – Редактор кода (различные настройки создания кода)

Errors Reporting – Отчёт об ошибках (настройки вывода отчётов об ошибках)

Display Control – Экран управления (настройки вывода различных параметров – типы вывода числовых значений, цветов и т.д.)

Restore all defaults – Восстановление настроек по умолчанию

Window

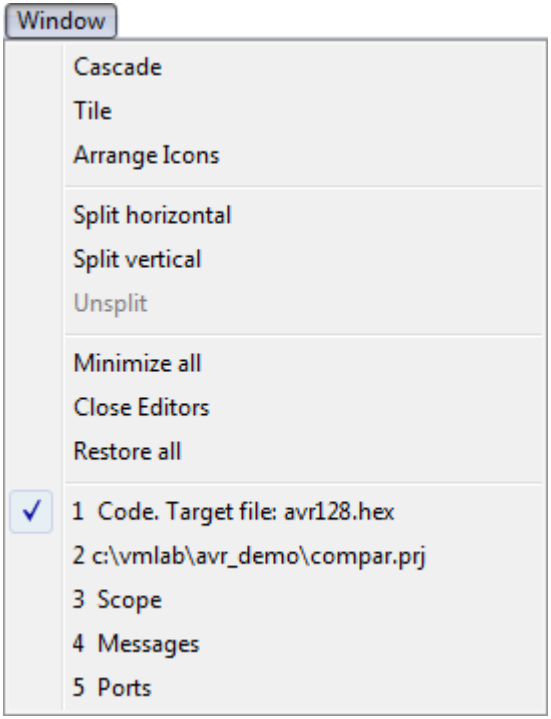

Рис. 15. Пункты подменю Window (Окна)

Содержит основные функции работы с окнами программы VMLAB (рис.15).

Cascade – Расположить открытые окна каскадом на рабочем столе VMLAB

Tile – Разместить на рабочем столе VMLAB открытые окна горизонтально (черепица)

Arrange Icons – Упорядочить значки окон

Split horizontal – Разделить окно горизонтально

Split vertical - Разделить окно вертикально

Unsplit – Объединить разделенное окно

Minimize all – Свернуть все окна

Close Editors – Закрыть окно файла проекта

Restore all – Восстановить все окна (из свёрнутого состояния)

Help

Этот пункт предоставляет информацию о программе VMLAB, об ассемблере AVR и другую справочную информацию (рис.16).

Contents – Подробная справка о программе и её возможностях

Using help – Использование помощи, вызов файла WinHil32.hlp

WinAVR – Ссылки на сайт GNU online

User-defined components – Помощь по созданию собственных компонентов

AVR assembler – Справка по ассемблеру AVR

About (s/n) – О программе

About licenses – Справка о получении лицензии на VMLAB (наличие лицензии открывает некоторые возможности, недоступные в бесплатной версии)

AMTools website – Ссылки на домашнюю страницу программы

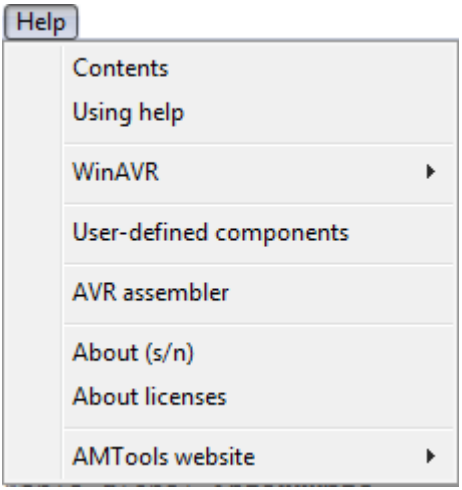

Рис. 16. Пункты подменю Help (Помощь)

2 Начало работы

Чтобы начать работу, необходимо создать проект.
Для этого нужно войти в пункт меню ***Project / New project***.
Появится окно следующего вида (рис.17).

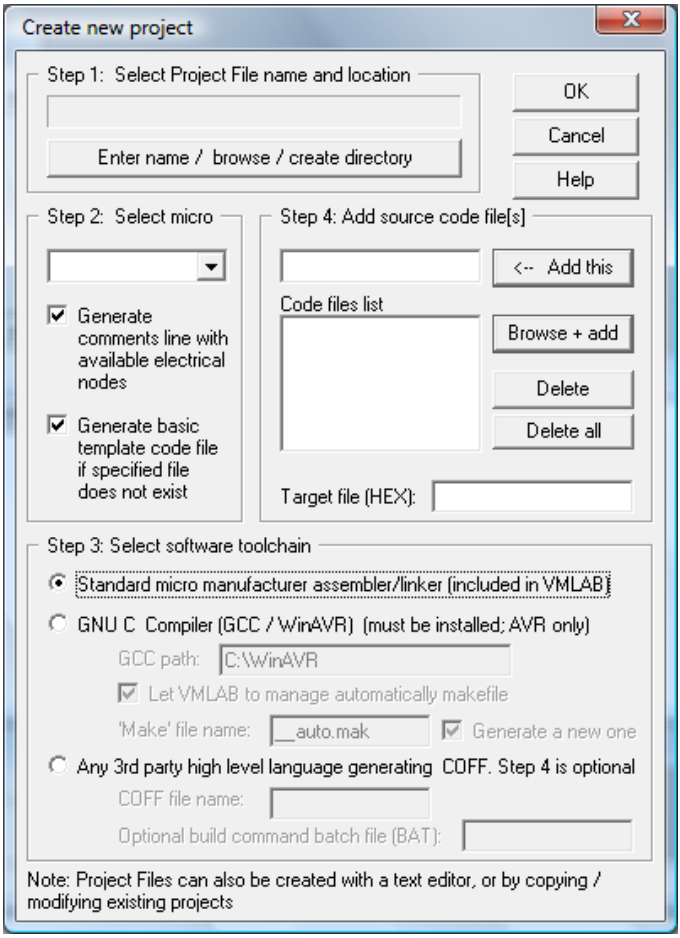

Рис. 17. Создание нового проекта

Здесь в соответствующих пунктах необходимо:

1. Выбрать директорию расположения проекта, а также определить его название;
2. Выбрать микроконтроллер из списка;

галочками отметить, нужно ли генерировать комментарии об электрической совместимости и шаблон кодового файла, если этот файл пока не существует;

3. Выбрать набор инструментальных средств:
   a. стандартный микроассемблер (включенный в VMLAB);
   b. GNU C компилятор (он должен быть установлен, только для ОМК AVR);
   c. любой другой компилятор, генерирующий файл в общем формате объектных файлов. Здесь необходимо ввести имя файла компилятора и пакетного запускающего файла (bat-файл);
4. Добавить файлы исходного кода.

VMLAB автоматически генерирует шаблон файла исходного кода. По умолчанию его имя совпадает с именем проекта. При необходимости можно изменить имя выходного hex файла. Файлы исходного кода и выходной hex файл будут размещены в той же папке, что и файл проекта.

Проект создан. Нажмите ОК

Если нужно открыть уже имеющийся проект, то необходимо выбрать пункт меню ***Project / Open project***, и с помощью Проводника открыть соответствующий файл проекта, либо выбрать пункт меню ***Project / Open Last project***, тогда откроется последний открывавшийся проект. Работа с проектом начинается с редактирования шаблона проекта.

2.1 Окно проекта

Окно проекта (рис.18) можно вызвать из пункта меню ***View / Project File.*** Оно отображает описание всех компонентов, входящих в проект.

Выбрать компоненты можно, зайдя в пункт меню ***Components*** и выбрав соответствующие компоненты из предложенного списка.

При выборе требуемого компонента в окне проекта автоматически появляется его шаблонное описание, в которое необходимо внести название и необходимые параметры (см. главу «Компоненты»).

Также в этом окне отображаются настройки (параметры) проекта:

- модель ОМК;
- название файла кода;
- название целевого файла;
- возможность трассировки;
- напряжение питания (Power nodes),
- частота генератора ОМК (Micro clock),
- время трассировки (Trace (micro+signals) storage time) и другие параметры, которые записываются по умолчанию, но их значения при необходимости можно изменить.

В приведенном примере между нулевым потенциалом и входом PD3 включена кнопка К0, а PD2 – кнопка К4. Резисторы 10 Ком соединяют линию питания и PD2,PD3.

В исходном состоянии на входах PD2,PD3 напряжение будет равно 0, а при нажатой кнопке – уровень логической единицы.

Активизирован терминал UART и заданы параметры последовательного порта с именем XTTY: скорость -9600бит\с, 8-разрядный символ, используется бит паритета по четности, один стоповый бит, символы отображаются в hex-формате.

После завершения редактирования необходимо выполнить компиляцию проекта (Project/Build). При этом откроется окно редактирования кода.

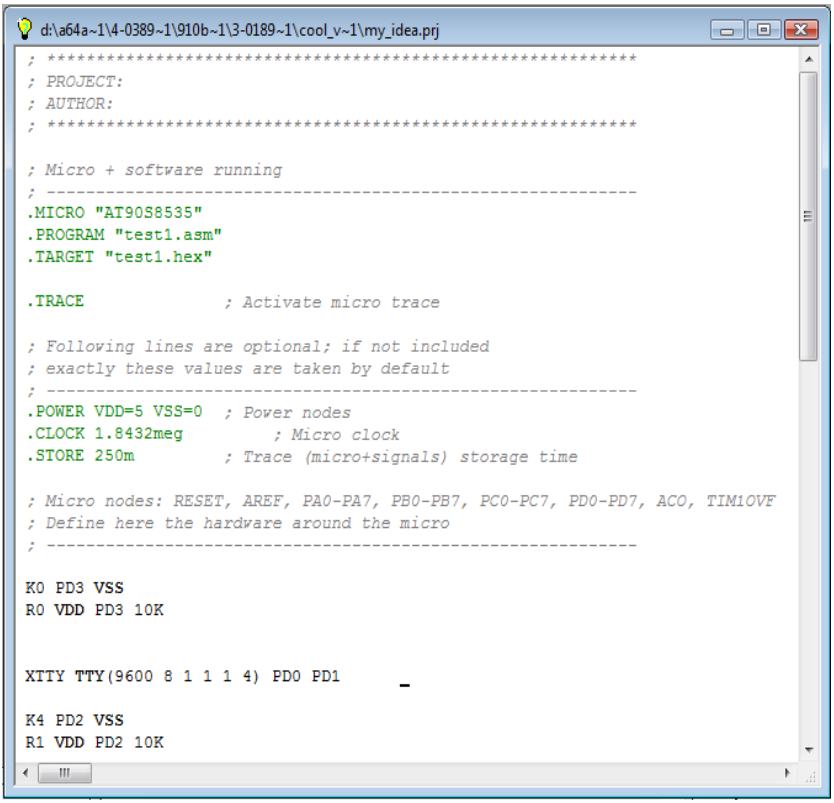

Рис. 18. Окно проекта

2.2 Окно редактирования кода

Окно редактора кода можно вызвать из пункта меню ***View / Code notebook.*** Редактор кода (рис. 19) имеет различную подсветку ключевых слов, операторов программы и комментариев.

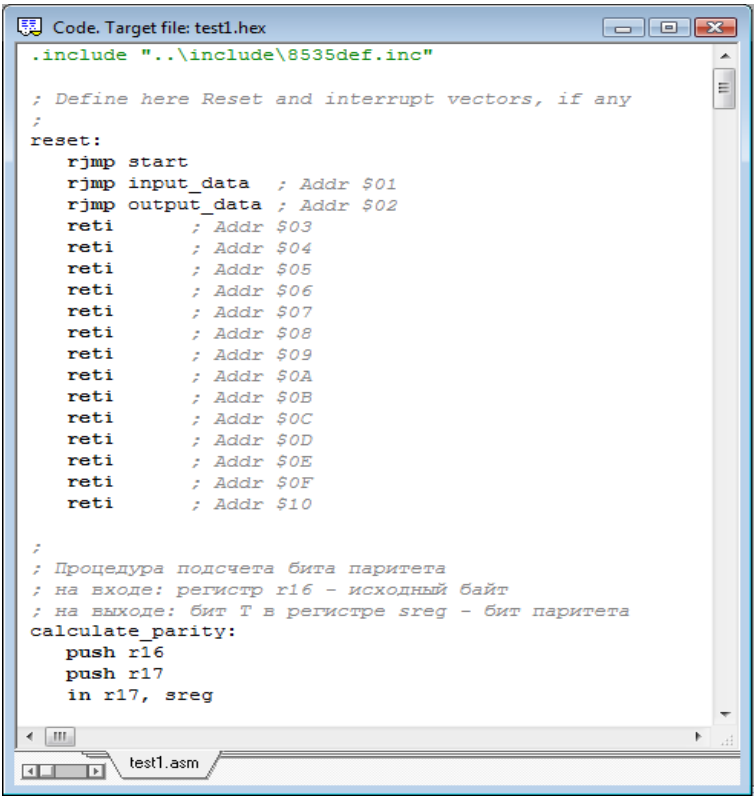

Рис. 19. Окно редактирования кода

Прежде всего, необходимо отметить следующее:

- перед компиляцией программы (пункт меню ***Project / Build***) сохранять текст не нужно, т.к. при компиляции текст сохраняется автоматически;
- по каждой команде в системе имеется контекстно-зависимая справка (на английском языке). Необходимо поставить курсор в позицию команды и нажать клавишу F1, а потом в появившемся окне ввести название команды. По регистрам такой справки не предусмотрено ;

- информация о ходе компиляции и удаче/неудаче её будет выведена в окно сообщений. Если компиляция прошла успешно, то все строки кода будут помечены зелёной точкой на полях. Если же в какой-то строке будет обнаружена ошибка, то эта строка будет отмечена красным восклицательным знаком, и другие строки зелёными точками помечены не будут.

2.3 Окно сообщений

Окно сообщений можно вызвать из пункта меню ***View / Messages.*** Оно подразделяется на пункты, которые (рис. 20) содержат информацию о:

- файле проекта;
- компиляции кода;
- времени исполнения программы;
- инструментах и поисках фрагментов текста в файлах.

Каждый из пунктов появляется по мере изменения состояния программы. Так, если проект был только создан, а код ещё не компилировался (либо компиляция не была проведена успешно из-за ошибок в коде), то доступен для просмотра только пункт Project File (файл проекта). Этот пункт содержит информацию о том, были ли произведены изменения в файле кода и проекта, выбранном ОМК, параметрах электрических узлов и состоянии файла проекта (есть ли в нём ошибки).

Если компиляция прошла успешно, разворачивается пункт Code Maker (получение исполняемого кода). Здесь представлена информация о файле кода (полный путь к нему), целевом файле (с расширением .hex), компиляторе, наличии / отсутствии ошибок в коде, готовности к исполнению и др.

Если программа была запущена, то становится доступным для просмотра пункт Run Time (время исполнения). Здесь выводится сервисные сообщения о ходе исполнения программы (например, Starting hardware-software co-simulation – начало симулирования, User break – остановка выполнения пользователем,

End of Simulation. System restarted – конец симулирования, система перезагружена), а также все предупреждения (например, о том, что не задан из каких-то необходимых параметров или слишком большая погрешность у производимой операции).

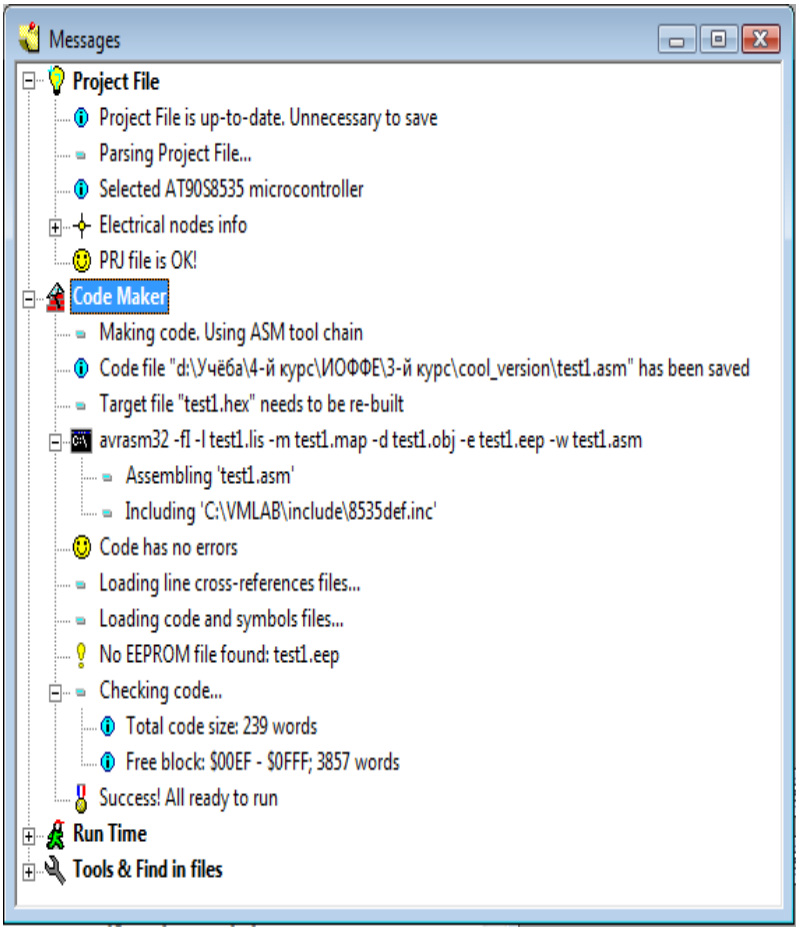

Рис. 20. Окно сообщений

Пункт Tools & Find in files доступен, если производился поиск фрагмента текста во всех файлах проекта (подпункт Search / Find in files -Grep-)

2.4 Окно осциллографа

Окно осциллографа (рис.21) можно вызвать из пункта меню ***View / Scope.***

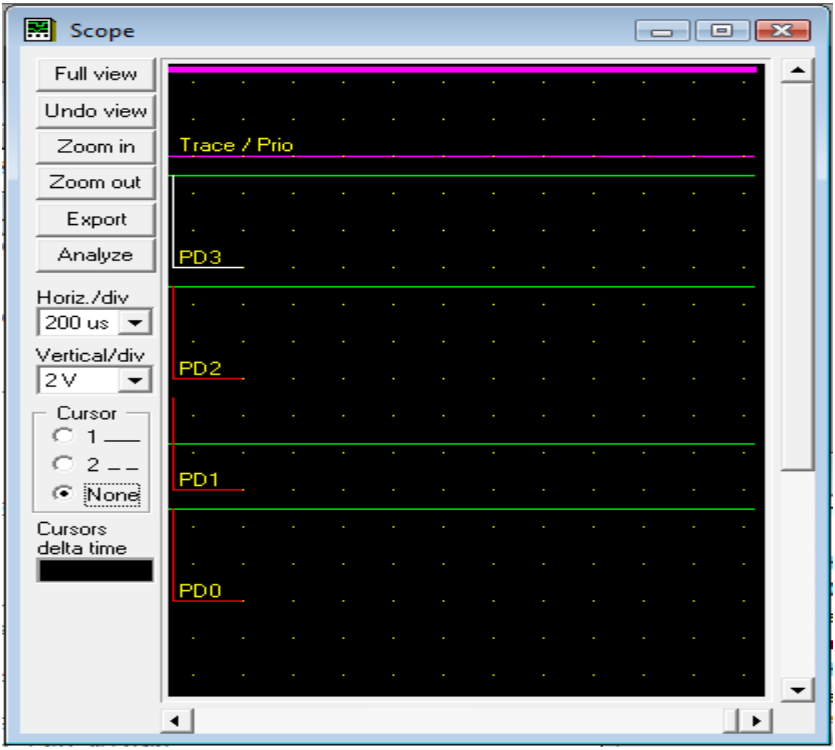

Рис. 21Окно осциллографа

Осциллограф в VMLAB – это имитатор обычного осциллографа, который измеряет различные сигналы с портов ввода/вывода.

Кнопка *Full View* позволяет отобразить графическую информацию в масштабе 100%.

Undo view возвращает предыдущий вид отображения сигналов.

Zoom in – кнопка увеличения масштаба, при нажатии на эту кнопку курсор на осциллографе отображается в виде лупы, и можно выделить фрагмент, чтобы отобразилась более подробная информация (функции кнопки доступны только когда остановлена эмуляция).

Zoom out при нажатии уменьшает масштаб времени в 2 раза.

Export позволяет сохранить данные с осциллографа в файл.

Analyze выводит сведения о параметрах, выделенного на осциллографе сигнала. Окно со сведениями представлено на рис.22.

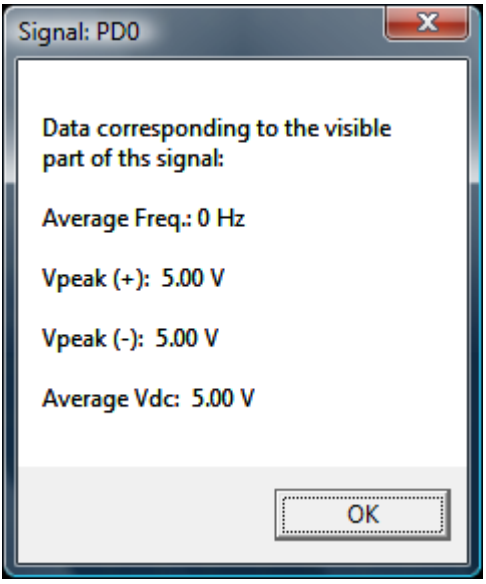

Рис. 22. Данные о сигналах с порта PD0

Здесь представлены сведения о средней частоте сигнала, амплитудных значениях напряжения и среднем значении напряжения.

Также в окне осциллографа есть 2 выпадающих списка, в первом из которых выбирается масштаб по горизонтали (в единицах времени), а во втором – по вертикали (в Вольтах).

Панелька *Cursor* отвечает за отображение курсора в области осциллографа (здесь – вертикальная черта, которая указывает момент времени).

Если выбрать сначала одно отображение курсора, а потом сменить его на другое, то на осциллографе будет отображено последнее положение предыдущего вида курсора наряду с положением текущего вида. При этом разница значений будет отображена в окошке «Cursors delta time».

2.5 Окно слежения

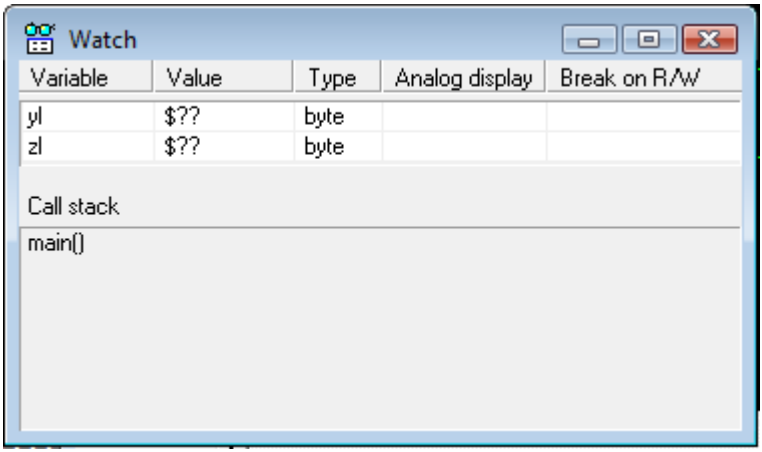

Рис. 23. Окно слежения

Окно слежения (рис23) можно вызвать из пункта меню ***View / Watch,*** оно необходимо для того, чтобы отслеживать и изменять значения регистров и переменных в ходе выполнения программы, а также отслеживать содержимое стека. Чтобы добавить наблюдаемую регистр (переменную), необходимо нажать

правой кнопкой мыши область окна слежения и выбрать регистр (переменную) из списка (см. рис. 24).

Закладка *Symbols* (имена регистров) отображена на рисунке 24.

Кнопка *Watch all* включает в наблюдение все регистры (переменные),

Clear all отменяет наблюдение всех регистров (переменных),

Clear breaks убирает все точки останова, поставленные на чтение и на запись наблюдаемых переменных (регистров) – в таблице столбцы R и W,

Clear bars – очищает аналоговое значение выражения (жёлтую полосу, длина которой пропорциональна значению).

В таблице можно выбрать вид отображения (шестнадцатеричный, двоичный и т.п.), чтение, запись и аналоговый штрих.

Закладка *C expressions* будет доступна в следующих версиях программы.

Закладка *Tips* выводит подсказки об окне наблюдений.

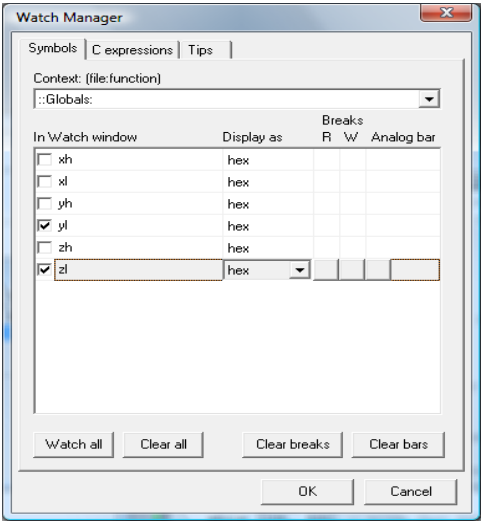

Рис. 24. Управление наблюдениями

2.6 Окно панели управления

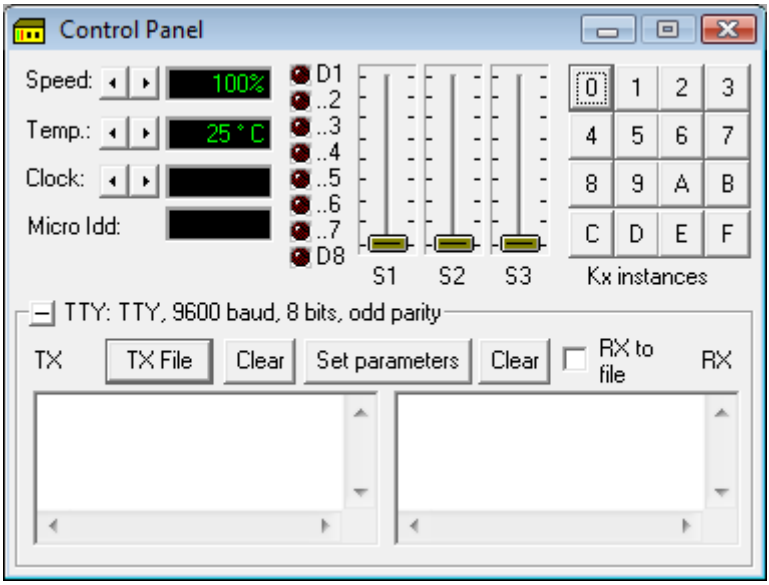

Рис. 25. Окно панели управления

Окно панели управления (рис.25) можно вызвать из пункта меню ***View / Control Panel.*** В окне панели управления всегда отображаются:

- *Speed* – скорость отображения эмуляции (в процентах). По умолчанию устанавливается максимальная скорость - 100%,
- *Temp* – позволяет эмулировать температуру работы системы, оказывая воздействие на элементы, зависящие от температуры окружающей среды;
- *Clock* – частота генератора. Позволяет изменять (даже в процессе эмуляции) частоту генератора ОМК. Обычно установлено то значение, которое прописано в директиве .MICRO файла проекта;
- *Micro Idd* – значение потребляемого тока. Рассчитывается на основании типовых параметров ОМК.

Величина потребляемого тока зависит от частоты внутреннего генератора, режима выполнения и т.д.

- 8 светодиодов, которые должны быть описаны в файле проекта как компоненты‘Dx’;
- 3 регулятора уровня S1-S3, используемые с компонентами, в которых эти регуляторы применяются ;
- 16 кнопок, пронумерованных с 0 по F. Кнопки могут использоваться автономно или в составе компонента «Интерактивная клавиатура», подключаемого в файле проекта.

Если в проекте используются другие компоненты, указанные в файле, то средства управление ими тоже выводится на панель. На рис.е 25 представлена панель управления интерактивным телетайпом TTY.

2.7 Окно порядка исполнения

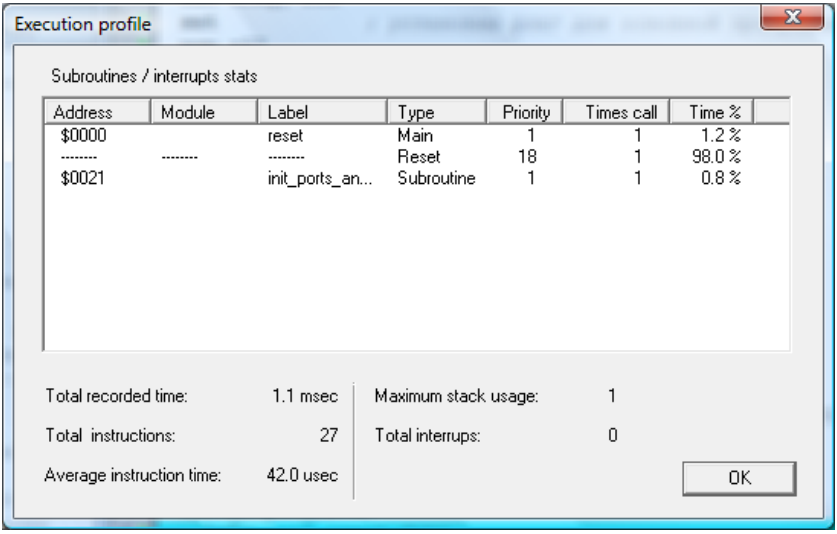

Рис. 26. Окно порядка исполнения

Окно порядка (рис.26) исполнения можно вызвать из пункта меню ***View / Execution States.*** Чтобы это окно стало

доступно, необходимо в файл проекта добавить строчки .TRACE и .STATS. В таблице выводится информация времени исполнения:

- Адрес исполняемой команды;
- Название модуля;
- Метка;
- Тип блока, в котором находится выполняемая команда (подпрограмма, прерывание и т.д.);
- Приоритет команды;
- Сколько раз исполнялась текущая строчка (команда);
- Процент времени, которое занимает выполнение текущей команды.

Также здесь приводится информация об общем времени исполнения, общем количестве исполненных команд, среднем времени исполнения команды, максимальной заполненности стека, общем количестве вызванных прерываний.

2.8 Окно регистров/флагов

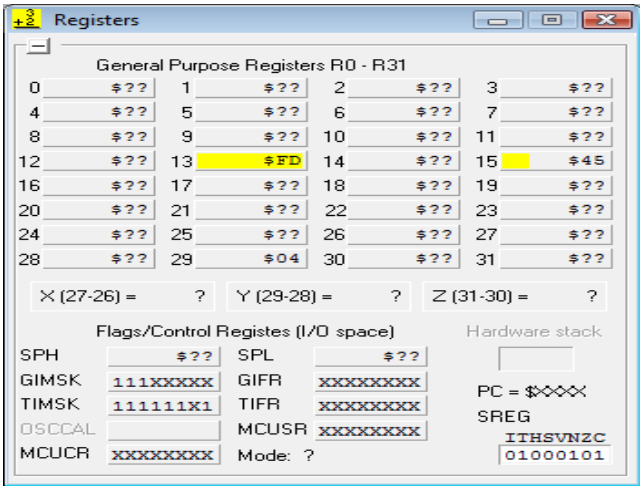

Рис. 27. Регистры/флаги

Окно регистров/флагов (рис.27) можно вызвать из пункта меню ***View / Registers/Flags.***

R0-R31 – Регистры общего назначения.

Особенностью организации зоны РОНов является использование R26 –R31 в качестве регистров косвенной адресации (они обозначаются буквами: R27-R26 – X, R29-R28 – Y, R31-R30 – Z).

SPH, SPL – Старший и младший байт указателя стека;

GIMSK – Регистр маски прерываний;

TIMSK – Регистр масок прерываний по таймеру/счётчику;

OSCCAL – Недоступен в незарегистрированной версии;

MCUCR – Управляющий регистр ОМК;

GIFR – Регистр прерываний;

TIFR – Регистр прерываний от счётчика/таймера;

MCUSR – Статусный регистр ОМК;

Mode – Режим работы ОМК;

PC – Значение счётчика команд;

SREG – Регистр флагов, название каждого флага указано над соответствующим битом.

Значения регистров можно изменять во время выполнения программы. Запись в РОНы и указатель стека выполняется двойным щелком по соответствующему полю (при этом открывается окно изменения значений), а в остальные регистры запись производится побитно.

Способ вывода значений (шестнадцатеричные данные, двоичные и т.д.) задаётся в ***Options/Display Controls.***

2.9 Окно памяти данных

Окно резидентной памяти данных (рис.28) вызывается из пункта меню ***View / Data Memory.*** Это окно показывает содержимое резидентной памяти данных микроконтроллера. Различными цветами подсвечивается:

- Жёлтый фон – ячейка, которая читалась во время исполнения программы;
- Голубой фон – ячейка, в которую производилась запись;
- Зелёный фон – ячейка, которая читалась и в которую производилась запись;

- Розовый шрифт – ячейки памяти, в которых хранятся регистры периферийных устройств;
- Зелёная черта – указатель стека (как правило, это старший адрес резидентной памяти, доступной в бесплатной версии программы – 0258h);

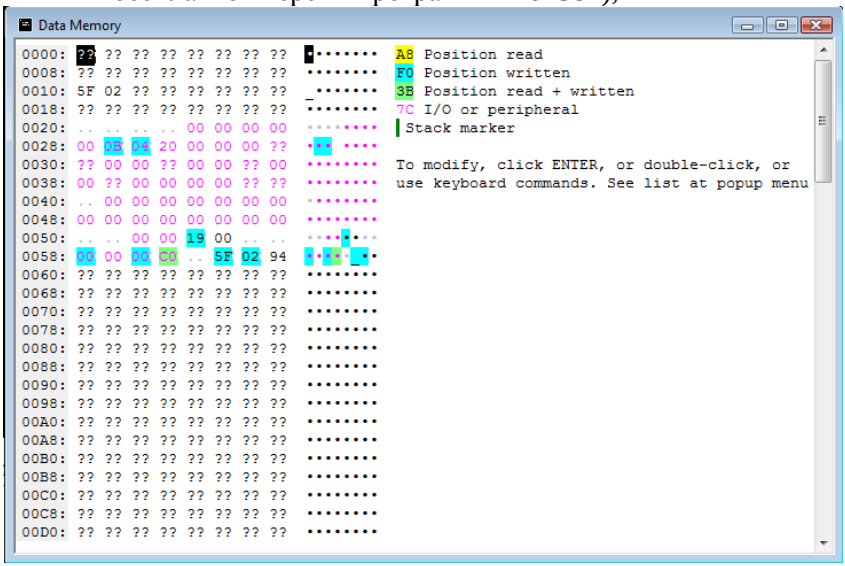

Рис. 28. Резидентная память данных

Все ячейки продублированы – слева показывается их 16-ричное значение, а справа – ASCII-коды.

Изменение содержимого ячейки выполняется двойным щелчком мыши или нажатием клавиши Enter. При этом откроется окно, в котором можно изменять значение.

При нажатии на правую кнопку мыши предоставляется дополнительный сервис для работы с ячейками (рис. 29):

- перейти от 16-ричного отображения данных *Entry: HEX area* (левые столбцы с вопросительными знаками по умолчанию) к ASCII-кодам *Entry: ASCII area* («точки» справа) для удобства ввода данных;
- выбрать количество ячеек в строке (8 или 16);

- установить значения в ячейках «00» (*Set RAM to 00*), «FF» (*Set RAM to FF*) или «??» (*Set RAM to ??*);
- *Byte operations* – выполнить операцию над выделенной ячейкой (инверсия, дополнительный код, инкремент, декремент, обмен тетрадами, установка в состояние «все единицы», сброс, изменение порядка следования бит на противоположный).

По стрелке доступны все операции;

- *Modify / Browse* – изменить значения в выделенной ячейке;
- остановить выполнение, когда будет чтение из выделенной ячейки (*Break on read*) или запись в выделенную ячейку (*Break on write*);
- *Clear R/W coverage* – отменить выделение цветом ячеек, из которых производилось чтение или запись.

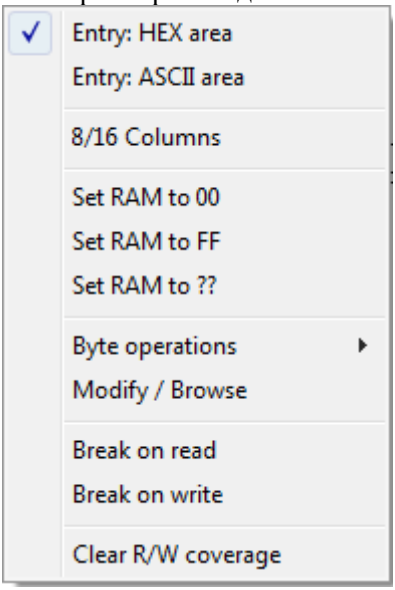

Рис. 29. Работа с ячейками резидентной памяти данных

2.10 Окно памяти команд

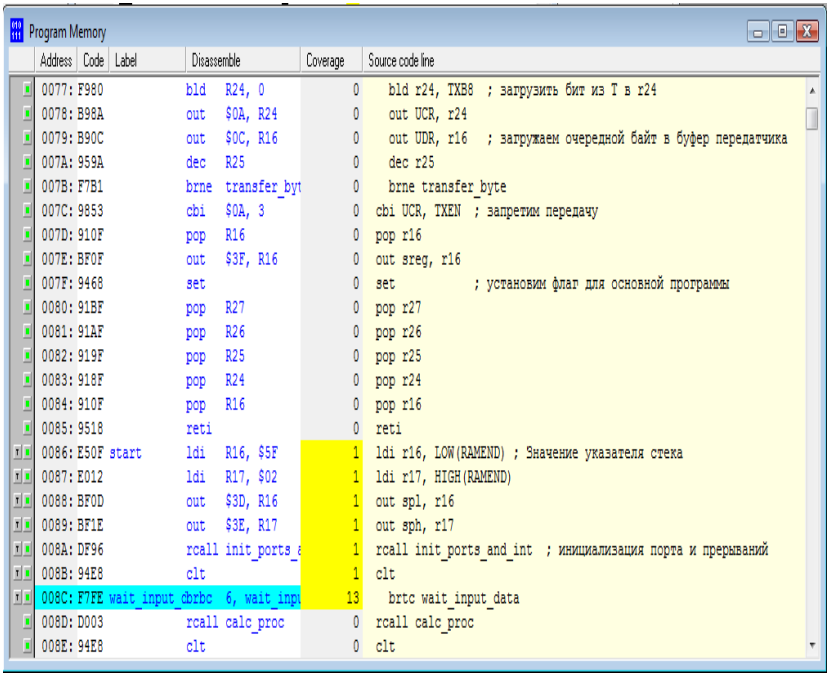

Рис. 30. Память команд

Окно памяти команд (рис.30) вызывается с помощью пункта меню ***View / Program memory.*** Здесь представлены команды программы в дизассемблированном виде. Команды в память загружаются после компиляции программы, поэтому и окно памяти команд «заполняется» только после компиляции проекта. Столбцы в окне памяти команд обозначают следующее:

- *Address* – адрес команды;
- *Code* – код команды;

- *Label* – метка;
- *Disassemble* – дизассемблированная команда;
- *Coverage* – показывает (во время исполнения), сколько раз выполнилась команда в ходе программы;
- *Source code line* – строка текста программы, соответствующая команде.

При нажатии правой кнопки мыши доступны следующие функции (рис.31):

- *Go to source* – выделяет строку в окне редактирования кода проекта, которая соответствует выделенной команде (той, на которой было осуществлено нажатие правой кнопки мыши);
- *Save disassembly* – сохраняет код на языке ассемблера в файл с расширением .asm (следует отметить, что сохранённый код будет несколько отличаться от того, что было написано Вами в VMLAB – как минимум, там не будет комментариев и директив, а переменные будут преобразованы в номера регистров, в которые компилятор их запишет);
- *Clear coverage* – очищает столбец Coverage.

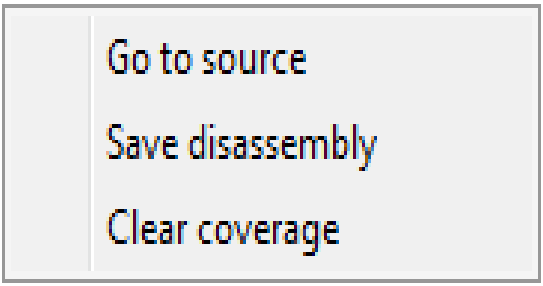

Рис. 31. Опции окна памяти команд

2.11 Окно EEPROM

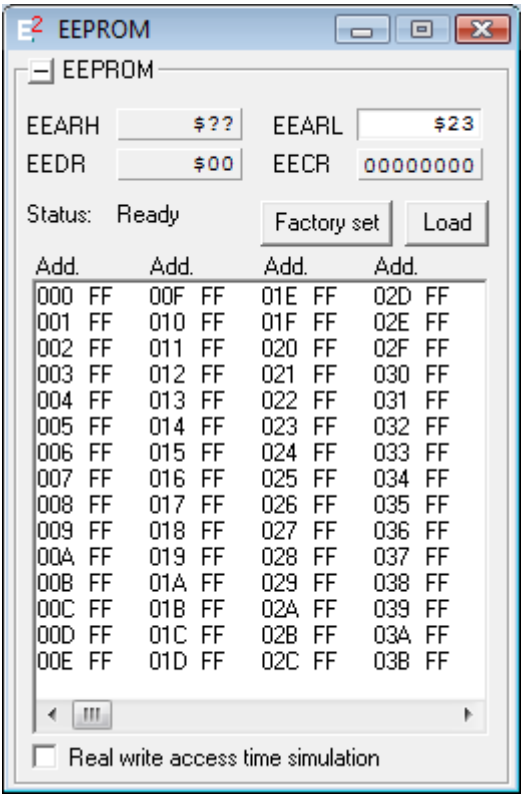

Рис. 32. EEPROM

Окно EEPROM (электрически перепрограммируемое ПЗУ) (рис.32) вызывается из пункта меню ***View / EEPROM.*** Это окно служит для отображения содержимого EEPROM. Большую часть окна занимает таблица с адресами EEPROM и значениями (в 16-ричном представлении), лежащими в его ячейках. Кроме того, в этом окне отображаются следующие регистры:

- EEARH (EEPROM Address Register (High byte)) и EEARL (EEPROM Address Register (Low byte)) – старший и младший байты регистра адреса.

Адреса памяти данных $1F ($3F) – Read/Write;

При сбросе (EEARH, EEARL)= 0.

Адресные регистры EEPROM определяют адреса в пространстве адресов EEPROM. В зависимости от объёма пространства адресов используется меньшая или большая часть битов этих регистров. Старший регистр адреса (EEARH) доступен, только если в EEPROM более 256 байт;

- EEDR (EEPROM Data Register) – регистр данных.

При сбросе (EEDR)=0.

- EECR (EEPROM Control Register) – управляющий регистр.

При сбросе (EECR)=0.

- Биты с 7 по 3 зарезервированы и всегда читаются как нули.
- Бит 3 – EERIE. Разрешение прерывания от EEPROM. Если этот разряд установлен в 1, то возникает прерывание в момент завершения цикла записи (если установлено общее разрешение прерывания в регистре SREG). При сброшенном разряде EEWE прерывание генерируется постоянно.
- Бит 2 – EEMWE: EEPROM Master Write Enable Управление разрешением записи в EEPROM). Если этот бит установлен, то будет разрешена запись в ячейку EEPROM при установленном в 1 бите EEWE. Если же этот бит сброшен, то установка бита EEWE в 1 не произведёт никакого эффекта. После программной установки EEMWE сбрасывается аппаратно через 4 машинных цикла.
- Бит 1 – EEWE: EEPROM Write Enable (Разрешение записи в EEPROM). Если в регистре адреса записан корректный адрес, бит EEWE может быть установлен (при EEMWE=1).
- Бит 0 – EERE: EEPROM Read Enable (Разрешение чтения из EEPROM). Если в регистре адреса записан корректный адрес, бит EERE может быть установлен. По окончании чтения этот разряд сбрасывается аппаратно.

○ Для выполнения записи в EEPROM необходимо

дождаться готовности к приему данных (EEWE =0),

загрузить в регистр EEDR данные, а в регистр EEAR - адрес.

Установить бит EEMWE в 1,

В течении четырех машинных циклов после установки флага EEMWE записать «1» в разряд EEWE.

Для операций чтения из EEPROM необходимо

дождаться окончания текущей записи,

загрузить адрес,

установить бит разрешения чтения EERE.

Регистр EEDR будет содержать данные, которые были прочитаны из ячейки EEPROM .

Надпись «Status» («Статус») выводит состояние EEPROM (Ready – все готово к работе, Error – ошибка в регистрах EEPROM, ? – статус не определён).

Кнопка «Factory set» заполняет все ячейки EEPROM ‘FF’.

Кнопка «Load» загружает значения EEPROM из файла с расширением .eep.

2.12 Окно портов ввода/вывода

Окно портов ввода/вывода (рис.33) вызывается из меню ***View / I/O Ports***. Здесь представлены регистры параллельных 8-битных портов ввода/вывода, которые поддерживаются ОМК AVR.

Регистр DDR определяет направление передачи данных через порт (0 – ввод, 1 – вывод).

В регистр PORT помещаются данные при выводе информации (DDRxn=1) . При вводе данных (DDRxn=0) разряды PORT определяют состояние внутреннего подтягивающего резистора для данного вывода (PORTxn=1 резистор включен между выводом ОМК и напряжением питания, PORTxn=0 –выключен)

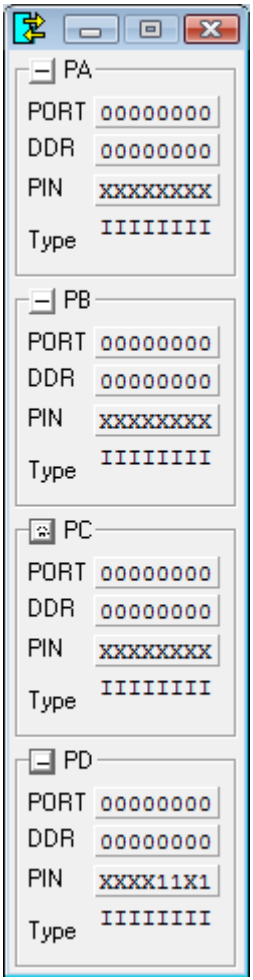

Рис. 33. Параллельные порты ввода/вывода

Чтение регистра PIN позволяет определить состояние внешних выводов ОМК.

Физически регистр PIN отсутствует. По его адресу осуществляется доступ к внешним выводам порта, состояние которых сохраняется во внутренних регистрах ОМК.

Строка Type указывает, как настроены выходы портов (на рис. 32 все порты настроены на ввод – буква «I»).

2.13 Окно периферийных устройств

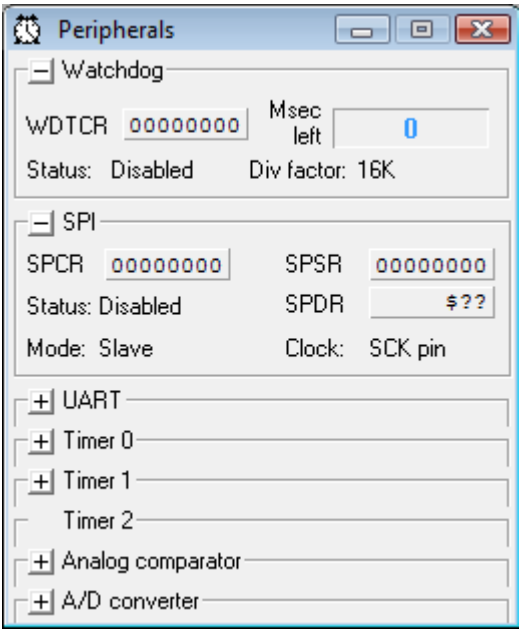

Рис. 34. Периферийные устройства

Окно периферийных устройств (рис34) можно вызвать из пункта меню ***View / Peripherals***. Здесь представлена информация обо всех периферийных устройствах, которые могут быть включены в состав проекта:

- Сторожевой таймер (Watchdog);
- Интерфейс SPI;
- Последовательный порт UART;
- 3 таймера (Timer 0, Timer 1, Timer 2);
- Аналоговый компаратор (Analog comparator);
- Аналого-цифровой преобразователь (A/D converter);
- Интерфейс I^2C (TWI);

- Последовательный порт (USART 0) и другие.

Набор периферийных устройств, отображаемых в этом окне, определяется моделью используемого ОМК.

Запись\чтение регистров выполняется аналогично предыдущему: регистры данных позволяют записывать информацию побайтно, а регистры управления – побитно.

Для отображения программной модели требуемого периферийного устройства в него необходимо «кликнуть» мышью.

3 Директивы ассемблера

Общие директивы:
.CLICKTOOL
.CLOCK
.MICRO
.PLOT
.POWER
.RAMINIT
.STORE
.TAB
.TARGET
.TOOLCHAIN
.TRACE
.SOURCE
.STATS

Директивы сборки проекта:
.PROGRAM

Директивы для компилятора C (GCC):
.GCCPATH
.GCCMAKE

Характерные для компиляторов 3-го типа (GENERIC) директивы:

.BAT
.COFF
Директивы ассемблера AVR изложены в [2].

.CLICKTOOL

Эта директива позволяет запускать консоль / MS DOS процесс нажатием кнопки  , доступной на панели функциональных кнопок. Это наиболее удобно в тех случаях, когда необходимо использовать внешние программы, чтобы записать информацию в электрически перепрограммируемое ПЗУ (EEPROM).

Для того чтобы запустить команду, определённую с директивой . CLICKTOOL, необходимо, чтобы проект был успешно скомпилирован командой ***Project / Build.***

Синтаксис:

.CLICKTOOL <MS-DOS / console command line>

Пример:

.CLICKTOOL “burn.exe –f myFile.hex”

.CLOCK

Определяет частоту внутреннего генератора ОМК.

Синтаксис:

.CLOCK <value>

Директива .CLOCK необязательная. Если она не определена, то используется значение частоты по умолчанию (например, для ST6 8 МГц). Во всяком случае, это значение может быть изменено позже на панели управления. На панели управления будут предложены только те значения частоты, которые поддерживает выбранный микроконтроллер.

Использование директивы .CLOCK предоставляет набор частот, которые поддерживает выбранный ОМК. Если в этой директиве задать частоту, которая не поддерживается выбранной моделью ОМК, то возникнет ошибка при компиляции.

.MICRO

Определяет модель микроконтроллера, используемого в проекте.

Синтаксис:

.MICRO "<micro name>" ["<micro options>"]

Примеры:

.MICRO "ST6210"

.MICRO "ST6225" "HWD"

.MICRO "Atmega8"

.MICRO "Atmega128" "BOOTRST=0"

Опции ОМК, определяемые в этой директиве, зависят от выбранного ОМК.

Директива .MICRO обязательная и единственная в каждом файле проекта.

.PLOT

Указывает VMLAB имена узлов, которые должны быть записаны и отображены в окне осциллографа.

Синтаксис:

.PLOT V(<nodeName>) [V(<nodeName>) ...]

Пример:

.PLOT V(node_1) V(PA0) V(PB0)

Директива .PLOT необязательна. Если она не представлена в проекте, никакие сигналы не будут записываться. Разрешается многократно использовать директиву PLOT.

.POWER

У VMLAB есть 3 глобальных значения напряжения: VDD, VSS и GND (или 0).

Директива .POWER устанавливает значения VDD и VSS привязанными к абсолютному нулю, GND.

GND (или число 0) указывает значение абсолютного нуля, опорное напряжение.

VDD – самое высокое напряжение (логическая единица).

VSS – самое низкое напряжение (логический ноль).

Все логические элементы (включая порты ввода/вывода) используют напряжение VDD/VSS для своих логических уровней.

В нормальном состоянии (по умолчанию) значения узлов VSS и GND совпадают. Но это может быть не всегда правильно. Определены альтернативные схемы напряжений (см. примеры).

Синтаксис:

.POWER VDD = <value> VSS = <value>

Примеры:

.POWER VDD=2.5 VSS=-2.5 ; GND is centered.

.POWER VDD=5 VSS=0 ; Positive, Default values.

.POWER VDD=0 VSS=-5 ; Negative

Директива .POWER необязательна. Если её нет, то устанавливаются значения по умолчанию для VDD (+5В) и VSS (0В).

.RAMINIT

Эта директива позволяет проинициализировать резидентное ОЗУ данных определёнными значениями до начала эмуляции. По умолчанию все ячейки резидентной памяти данных проинициализированы знаками «??» (неизвестное значение).

Директива .RAMINIT инициализирует только ячейки, в которых значения не были определены. Если же в память данных уже были записаны значения, то они не будут изменены.

Синтаксис:

.RAMINIT <десятичное целое число>

Обратите внимание, что **значение должно быть именно в десятичной форме**.

Пример:

.RAMINIT 15; Установит во все ячейки памяти значение $0F.

Используйте эту директиву осторожно: она призвана помочь разработчику, но в реальной жизни такой функции у ОМК нет. В жизни ОЗУ будет не проинициализировано, поэтому программисту необходимо написать код программы для инициализации памяти.

.STORE

Устанавливает время хранения для окна осциллографа (и для сигналов, и для трассировки ОМК).

Синтаксис:

.STORE <value>

Пример:

.STORE 200m ; Время хранения – 200 мс.

Эта директива необязательна. Если она не описана в проекте, будут установлены значения по умолчанию в зависимости от памяти вашего компьютера. Значения вне заданного промежутка будут округлены до максимального / минимального.

Сохранение на длительное время или слишком большого числа сигналов может привести к неверному отображению информации в окне осциллографа. Если вы отметили, что время перерисовки слишком большое или команды просмотра/масштабирования изображения медленно выполняются, сократите время хранения или отображайте меньшее число сигналов. Возможности по отображению большого количества сигналов и длительности времени хранения зависят от параметров вашего компьютера.

.TAB

Позволяет определить позиции табуляции для данного проекта. Эта директива полезна для автоматического изменения отступов при нажатии клавиши TAB для разных типов/стилей кода. Например, для кода на языке С позиция табуляции 3, а для языка ассемблера – 8.

Синтаксис:

.TAB <целое число от 1 до 16>

Пример:

.TAB 6

По умолчанию отступ равен 3.

.TARGET

Определяет файл с расширением .HEX, который будет создан в результате компиляции.

Синтаксис:

.TARGET <имя файла>

Эта директива обязательна только если используется набор инструментов GENERIC или GCC. Если используется ASM и не определена директива .TARGET, первый файл с расширением .ASM в директиве .PROGRAM с расширением .HEX будет использоваться как целевой файл.

.TOOLCHAIN

Определяет набор инструментов для создания исполняемого файла. Возможны 3 значения:

“ASM”: стандартный ассемблерный код;

“GCC”: компилятор GNU для языка C;

“GENERIC”: 3-й тип инструмента.

Синтаксис:

.TOOLCHAIN {<“ASM”>, <“GCC”>, <“GENERIC”> }

Пример:

..TOOLCHAIN “GCC”

.TRACE

Указывает, что история выполнения команд микроконтроллера должна быть трассирована. Трассировка будет видна в окне осциллографа.

Синтаксис:

.TRACE

Директива .TRACE необязательна. Если она не описана, то трассировка записываться не будет.

.SOURCE

Определяет список файлов, которые будут загружены в окно редактора кода. Эта директива доступна только в инструментах GCC и GENERIC. Не используйте её вместе с директивой .PROGRAM.

Поскольку директива .PROGRAM используется для указания VMLAB, какие файлы кода будут использоваться для создания целевого файла с расширением .HEX, директива .SOURCE только указывает, какие файлы нужно открыть в окне кода.

Синтаксис:

.SOURCE <имя файла> [<имя файла> …]

Пример:

.SOURCE “file1.c” “file2.c” “file3.c”

Если вы загрузили много файлов, используйте «+», чтобы сделать ваш файл более читаемым. Например:

```
.SOURCE “file1.c” “file2.c” “file3.c”
+       “file4.c” “file5.c” “file6.c”
+       “file7.c”
```

.STATS

Эта директива позволяет производить запись данных об исполнении программы и выводить их при вызове команды ***View / Execution States.*** Она должна быть использована вместе с директивой .TRACE.

Синтаксис:

.STATS

Предостережение: Директива STATS записывает данные с самого начала симуляции. Поскольку время исполнения программы может быть достаточно большим, может быть сгенерирован огромный файл _trace.dat. Поэтому используйте эту директиву осторожно.

.PROGRAM

Директива .PROGRAM определяет, каким образом компилировать конечный программный продукт, который будет загружен в ОМК, используя стандартный инструмент ASM/

Интегрированная среда VMLAB соберёт проект и слинкует его, чтобы сделать конечный файл с расширением .HEX, который в итоге будет использован для программирования ПЗУ ОМК или электрически перепрограммируемого ПЗУ.

Все файлы, описанные в этой директиве будут загружены в окно редактора кода.

Синтаксис:

.PROGRAM “<имя файла>”[: “<Опции для линкования>”] [“<имя файла>”[: “<Опции для линкования>”]]

Примеры:

.PROGRAM “example.asm”

.PROGRAM “example.hex”

.PROGRAM “example1.asm” “example2.asm” “example3.obj”

.PROGRAM “example1.asm”: “-P2 : 80-07ff” “example2.asm”

Имя первого модуля определяет загружаемый целевой файл (с расширением .HEX), если опущена директива .TARGET. В первом и втором примерах целевым файлом будет файл “example.hex”.

Опции линкования, если они есть, отправляются в программу Линкера, если используется более одного модуля. Они чаще всего используются для изменения используемых адресов. Здесь должен быть использован тот же синтаксис, что и для Линкера.

Файл с расширением .HEX может быть помещён прямо в директиву .PROGRAM. Делайте так, когда у вас есть только этот файл, и нет никакого дополнительного кода или информации об отладке. В этом случае вы сможете запустить программу, но не сможете отслеживать значения переменных и т.д.

Когда у вас много файлов подгружается к проекту, используйте «+», чтобы сделать ваш файл более читаемым. Например:

.SOURCE “file1.c” “file2.c” “file3.c”
+ “file4.c” “file5.c” “file6.c”
+ “file7.c”

.GCCPATH

Определяет путь инсталляции в GCC. Эта директива доступна только для набора инструментов GCC.

Синтаксис:

.GCCPATH <путь к файлу>

Пример:

.GCCPATH “C:\WINAVR”

Если используется набор инструментов GCC и эта директива не найдена, то VMLAB примет путь C:\AVRGCC как путь инсталляции GCC.

.GCCMAKE

Определяет, либо файл будет компилироваться утилитой GCC ‘make’, либо включен автоматический режим. Если включён автоматический режим, то флаги компилятора GCC могут быть установлены через пункт меню Project / GCC making flags; затем VMLAB будет управлять автоматически созданием мейк-файла.

Синтаксис:

.GCCMAKE {<имя файла>, AUTO}

Примеры:

.GCCMAKE “myMake.mak” ; Мейк-файл пользователя

.GCCMAKE AUTO ; Автоматический режим

Если используется компилятор GCC и эта директива не найдена, то VMLAB будет предполагать, что имя мейк-файла ‘makefile’. Автоматический режим не устанавливается по умолчанию.

.BAT

Определяет bat-файл для запуска консоли MS-DOS, которая будет запущена чтобы скомпилировать необходимые .HEX и .COFF файлы.

Эта директива доступна с инструментами компиляции GENERIC.

Синтаксис:

.BAT <имя файла>

Эта директива необязательна. Если она не найдена, VMLAB будет считать, что файлы .HEX и .COFF уже скомпилированы и необходимо только загрузить их.

.COFF

Определяет COFF файл (Common Object Format File – файл общего формата объектов), где находится символическая информация для отладки. Эта директива используется только с инструментами GENERIC.

VMLAB может читать и классический COFF, и расширенный COFF-файл для AVR (со поддержкой структур/объединений компонентов). Всегда, когда это возможно, указывайте вашему компилятору делать расширенный COFF (совместимый с AVRStudio 4.x).

Если вы используете директиву COFF, а директива .SOURCE отсутствует, то VMLAB будет загружать автоматически файлы кода, совместимые с COFF файлом.

Синтаксис:

COFF <имя файла>

Пример:

COFF “myFile.cof”

4 Библиотека компонентов аппаратного обеспечения

VMLAB предоставляет большинство компонентов, которые необходимы для эмуляции реальной работы ОМК.[3] Эти компоненты делятся на 3 типа:

- Базовые аналоговые компоненты

- Резистор (Resistor)
- Конденсатор с заземлением (Grounded capacitor)
- Переключатель/ключ, кнопка (Switch / key, button activated)
- Светодиод (LED diode)

- Аналоговые генераторы напряжения
  - Импульсный источник напряжения (Pulsed voltage source)
  - Источник синусоидального напряжения (Sine wave voltage source)
  - Движковый источник напряжения (интерактивный) Slider dependent voltage source (interactive) – со скользящим контактом (реостат).
  - Цифровой формирователь сигнала (Digital pattern generators) NRZ-генератор(интерактивный) (Non-ruturn-to-zero (NRZ) generator (interactive))
- Макро-модели (Macro-models)
  - Операционный усилитель (Operational amplifier)
  - Компаратор (Comparator)
  - Двухвходовой И-НЕ (2 inputs NAND gate)
  - 8-битный ЦАП (8 bits D to A converter)
  - Телетайп на RS232(интерактивный) (RS232 based TTY (interactive))
  - ЖКИ модуль (LCD module)
  - I2C монитор(интерактивный) (I2C monitor (interactive))
  - Интерактивная клавиатура 4x4 (Interactive keypad 4x4)

4.1 Базовые аналоговые компоненты

Резистор (Resistor)

Синтаксис:

R [< Текущее имя >] < Имя Вывода > < Имя Вывода > < значение >

Примеры:

Rload node1 node2 10K

R node_a VSS 100

Замечания:

Имена резисторов должны начинаться с буквы R .Нулевое значение R недопустимо.

VDD, VSS и GND - зарезервированные имена выводов питания.

Конденсатор с заземлением (Grounded capacitor)

Синтаксис:

C[< Текущее имя >] < Имя Вывода > < Имя Вывода Питания > < значение >

Примеры:

C1 node1 VSS 10n ; Конденсатор C1=10 нФ включен между контактом node1 и VSS.

C 2 node2_ VSS 1u ; Конденсатор C2=1мкФ включен между контактом node2 и VSS.

Замечания:

Не разрешено использование переменных конденсаторов.

Второй вывод всегда должен быть VSS.

VDD и GND - зарезервированные имена выводов питания.

Переключатель/ключ, кнопка

Кнопка является интерактивным компонентом. Блок кнопок размещен на панели управления и нумеруется в16-ричной системе (0-F).

Синтаксис:

K{0 - 9, A - F} < Имя Вывода 1> < Имя Вывода2> [{NORMAL, LATCHED, MONOSTABLE(< timeValue >)}]

Кнопка нормально разомкнутая, включается между выводами 1 и 2.

Последние необязательные параметры указывают тип кнопки:

- NORMAL - **задается по умолчанию**. Цепь замыкается при нажатии кнопки.

- LATCHED - кнопка переключается в состояние замкнута/открыта после каждого нажатия кнопки.

- MONOSTABLE (значение времени) - при нажатии кнопка замыкается на время, определяемое опцией timeValue .

Примеры:

K1 node1 VSS LATCHED ;

K3 myNode GND monostable(10m) ;

Советы:

При работе с клавиатурой 4x4 следует выбирать макромодель клавиатуры KEY4X4, которая обеспечит автоматическое соединение матрицы клавиатуры.

Светодиод (LED diode)

Это визуальный компонент. Всего доступно 8 светодиодов в окне панели управления. Модель светодиода предполагает, что его сопротивление равно 1 ОМ.

Светодиод включается при токе больше 1 мА,.

Анод светодиода всегда должен быть подключен к VDD .

Синтаксис:

D{1 - 8} VDD < Имя Вывода >

Примеры:

D1 VDD node1 ; это светодиод "D1" в окне контрольной панели.

Советы:

При работе симулятора в **непрерывном режиме** включение светодиода можно не заметить. Чтобы убедиться в правильности работы программы , в месте включения светодиода рекомендуется задать точку останова (breakpoints).

Светодиод может быть напрямую подключен к порту микроконтроллера, но в этом случае не определяется уровень напряжения на входе порта.

D1 VDD PB0 ; можно, но не видно уровень напряжения на PBO

. Чтобы его увидеть, необходимо подключаться через резистор:

R2 n2 PB0 1K

D2 VDD n2 ;

4.2 Генераторы напряжения

Импульсный генератор напряжения (Pulsed voltage generator)

Синтаксис:

V[<текущее Имя>] <Имя Вывода> <Вывод Питания>PULSE(<vInitial> <vFinal> <tDelay> <tRise> <tFall> <tWidth> <tPeriod>)

<vInitial> -начальное значение напряжения,

<vFinal>- конечное значение напряжения,

<tDelay> - время задержки,

<tRise>-длительность фронта,

<tFall> - длительность среза,

<tWidth> - длительность импульса,

<tPeriod>) – период следования импульсов

Примеры:

V PA0 VSS pulse (0 5 40u 0 0 50u 100u)

Vpulse node1 VSS PULSE (2.5 3.5 0 1u 1u 1.5m 2.5m)

Замечания:

Второе имя вывода всегда должно быть выводом питания: VDD, VSS, GND (или 0).

Генератор синусоидального напряжения (Sine wave voltage generator)

Синтаксис:

V[< Текущее имя >] < Имя Вывода > < Вывод Питания > SIN(< vOffset > < vAmplitude > < frequency >) Смещение, амплитуда, частота

Примеры:

Vsin PA0 VSS sin(2.5 2.5 10K) ; 10 KHz, центрированный, амплитуда 5V

Замечания:

Второе имя вывода всегда должно быть выводом питания.

Смещение и амплитуда задаются в вольтах, а частота в герцах.

Регулируемый источник напряжения (интерактивный) (Interactive slider dependant voltage generator)

Синтаксис:

V[< Текущее имя>] < Имя Вывода >< Вывод Питания > SLIDER_< slider Number >(< vLow > < vHigh >) Номер ползунка, Uнижнее, Uверхнее

Примеры:

Vth minus vss SLIDER_1(0 5) ; с 0 до 5 вольт (Slider #1)

V v_node vss SLIDER_3(1.5 2.5) ; с 1.5 до 2.5 вольт (Slider #3)

Замечания:

Второе имя вывода всегда должно быть выводом питания.

На панели управления доступны только три источника (1,2,3).

К одному источнику напряжения можно подключать несколько выводов ОМК.

Формирователь цифровых сигналов (Digital pattern generators)

NRZ-генератор (интерактивный) (Interactive NRZ digital pattern generator)

Non-ruturn-to-zero (NRZ) – код без возврата к нулю (метод двоичного кодирования информации, при котором единичные биты

представляются положительным значением, а нулевые отрицательным).

Данный компонент генерирует цифровую диаграмму формы NRZ на специальном выводе.

Возможно задание трех уровней напряжения 1-VDD, 0 - VSS и X -(VDD+VSS)/2 .

Диаграмма активируется нажатием соответствующей кнопки KEY_x в окне панели управления.

Моделирование заданной последовательности выполняется, если был использован RESET.

Это позволяет задавать до 16 диаграмм, активизируемых клавишами, плюс выборочная инициализация, для каждого описания NRZ, расположенного в Файле проекта (Project File).

Синтаксис:

P[<текущее Имя>] NRZ(<time Bit>) <Имя Вывода> KEY_<keyNumber> "<pattern>"

период

номер клавиши, диаграмма

[+ KEY_<keyNumber> "<pattern>"]

[+ RESET "<pattern>"

[+ ....]

< keyNumber > должен быть от 0 до 9 от A до F.

"< pattern >" должна быть любой последовательностью значений 0,1 и X (максимальная длина 255), или строка вида "file:< fileName >", для диаграмм заданных во внешних файлах.

< timeBit > -длительность импульса.

Если RESET не использован, то диаграмма будет инициализирована как X.

Примеры:

```
P1 NRZ(20u) pb0  KEY_1 "01000110xx0000100111"

P2 NRZ(40u) myNode  KEY_1 "01000110xx000"
+           KEY_2 "111000111000111000111111"
+           KEY_A "10100001xxxx11110100"
+           RESET "101"
```

```
; Четырехфазный широтно-импульсный модулятора, активизируемый KEY_1
;
P0 NRZ(20u) bit_0  KEY_1 "01000110xx0000100111"
P1 NRZ(20u) bit_1  KEY_1 "00000000111000000000"
P2 NRZ(20u) bit_2  KEY_1 "11111111111110001111"
P3 NRZ(20u) bit_3 KEY_1 "XXXXXXXXXXXXX0000000"

; пример диаграммы, загружаемой из файла
P0 NRZ(20u) bit_0  KEY_1 "file:myPattern.txt"
```

Для задания диаграмм во внешних файлах можно использовать любой текстовый файл с расширением: TXT, DAT и т.п.

Файл должен быть расположен в вашей рабочей директории вместе с .PRJ файлом и кодом.

Правила написания подобных файлов приведены в примере:

Пример содержания предыдущего файла: myPattern.txt

```
100100 10 X 00 00 1111 ; используйте 0,1 и X в качестве двоичных значений
10000 11111  ; точка с запятой - комментарии
1 1 1 1 111 111100 ; пробелы и символы возврата каретки (конца строки) игнорируются
...
100 11 11 11 ; неограниченная длина; будьте осторожны: если этот
10111 001001 111 1 ; файл большой, VMLAB займёт много
100 10011 11 ; ресурсов памяти
```

Замечания:

Обычно строки получаются длинные, поэтому удобнее пользоваться конкатенатором “+”.

Несколько генераторов диаграмм могут быть активизированы одновременно с помощью уникальной клавиши. Это может быть полезно, например, для генерации диаграммы шириной в байт.

Те же кнопки могут использоваться и для иных целей, например, в качестве переключателя/ключа (Interactive switch/key).

В таком случае оба компонента будут активизированы одновременно

4.3 Макро-модели (Macro-models)

Операционный усилитель (Operational amplifier (OPAMP))

Синтаксис:

X[<Текущее имя>] OPAMP <nodePlus> <nodeMinus> <nodeOut>

<nodePlus> -прямой вход,
<nodeMinus> - инверсный вход,
<nodeOut> - выход

Пример:

Xop OPAMP node_a node_b node_c

Замечание:

Макромодель OPAMP предельно проста и оптимизирована для системы моделирования. Особенности типа ограниченной пропускной способности, скорость нарастания выходного напряжения и т.п. не моделируются. Выход OPAMP соединен с напряжениями VDD и VSS.

Компаратор (Comparator (COMP))

Синтаксис:

X[<Текущее имя>] COMP <nodePlus> <nodeMinus> <nodeOut>

Пример:

Xcomp COMP node_a node_b node_c

Замечание:

Макромодель COMP предельно проста и оптимизирована для системы моделирования. Такие особенности, как ограниченная скорость нарастания входного напряжения и т.п., не моделируются. Выход COMP соединен с напряжениями VDD и VSS.

Двухвходовой И-НЕ (2 inputs NAND gate (ND2))

Синтаксис:

X[<Текущее имя>] ND2 <input1> <input2> <output>

вход1, вход2, выход

Пример:

X ND2 node_a node_b node_c

8-битный ЦАП (8 bits Digital to Analog Converter (D2A8))

Синтаксис:

X[<Текущее имя>] D2A8 <i7> <i6> <i5> <i4> <i3> <i2> <i1> <i0> <nodeOut>

Вывод входа

Вывод выхода

8-битное значение считывается с входов < i0 > .. < i7 > и конвертируется в аналоговое значение. Шкала такого значения следующая: 0 -->VSS, 255 --> VDD

Пример:

XD2A D2A8 pa7 pa6 pa5 pa4 pa3 pa2 pa1 pa0 ana_out

Замечание:

Данный элемент может использоваться, например, для преобразования цифрового кода на выходе порта в аналоговый сигнал .

Интерактивный телетайп (Interactive TTY)

Интерактивный телетайп TTY – позволяет контролировать работу последовательного порта UART.

Этот виртуальный TTY содержит раздельные TX/RX экраны, кнопки очистки экранов Clear, кнопку конфигурации Set Parameters, позволяющую контролировать настройки телетайпа и изменять TX/RX параметры в случае необходимости (рис. 35).

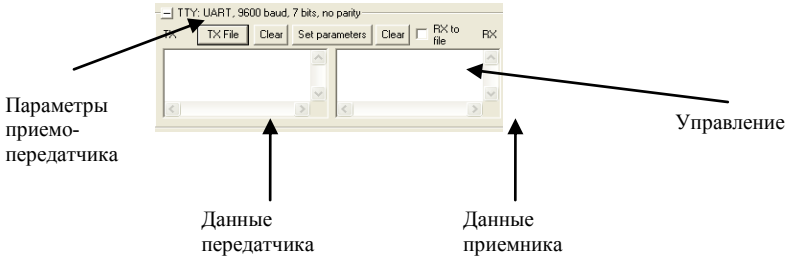

Рис. 35. Компонент UART

Подключив телетайп к системе через выводы TX/RX и установив курсор в соответствующем окне, имеется возможность задавать необходимые сообщения. Каждый набранный символ будет передаваться через вывод TX. Каждый символ принятый на вывод RX будет показан в окне RX.

Имеется возможность задавать два типа телетайпа TTY и TTY2. Отличие только в том, что TTY2 имеет окно RX больших размеров.

Синтаксис:

X[<Текущее имя>] TTY(<baudRate> [<7 or 8 bits> <parity> <oddParity> <stopBits> <RxDisplayAs>]) <nodeTx> <nodeRx>

Параметры:

Все параметры, за исключением первого, не обязательные. Если при выборе компонента они не указаны, то берутся значения по умолчанию.

< baudRate > скорость двоичной передачи (в бодах): 300, 600, 1200, 2400, 4800, 9600, 19200, 28800, 38400, 57600, 76800, 115200. **По умолчанию не задано (обязательное поле)**

<7or8bits> 7 или 8 бит передача: 7, 8. По умолчанию = 7.

< parity > паритет: 0: нет паритета, 1: паритет. По умолчанию = 0.

< oddParity > тип паритета: 0: чётный, 1: нечётный. По умолчанию = 0.

< stopBits >стоп биты: 1, 2. По умолчанию = 1.

< RxDisplayAs > опция отображения RX: 1, 2, 3, 4. По умолчанию = 1.

Последний параметр < RxDisplayAs > позволяет задавать формат

дисплея в окне RX, он может быть:
1: 8 бит ASCII-ANSI код (используемый Windows). По умолчанию.
2: Классический 7 бит ASCIIкод, коды >127 будут отображаться как шестнадцатиричные числа.
3: Десятичный код.
4: Шестнадцатеричный код

Пример:

Xpeter TTY(9600 7 0 0 1 2) tx_peter tx_paul Xpaul TTY(9600) tx_paul tx_peter

Пересылка содержимого файла

Кнопка TX File позволяет передавать содержимое файла.

Файл должен быть назван как имя с расширением TX. Например это будет PETER.TX или PAUL.TX.

Файл должен быть расположен в рабочей директории, вместе с остальными файлами: .PRJ, кодами и т.п.

Формат/время файла контролируется 4 параметрами в первой строке (пример): 1.0e-3 5.0e-3 TEXT CR ; первая строка для параметров Hello world, this text is to be transmitted

- Параметр 1: задержка между символами **в секундах** (время между стоп-битом и следующим старт-битом). По умолчанию = 0
- Параметр 2: задержка между строками **в секундах**. По умолчанию = 0.
- Параметр 3: формат данных: TEXT или BIN. По умолчанию = TEXT
- Параметр 4: Символ , указывающий на конец строки в режиме TEXT. Возможные значения: CR, LF, CRLF, NULL, NONE. NULL пошлёт 0x00 byte; NONE проигнорирует конец строки. Параметр игнорируется в режиме BIN.

В случае обнаружения ошибки в строке параметра, выдаётся сообщение предупреждение, и берутся значения по умолчанию.

Для режима BIN байты должны быть записаны в шестнадцатеричном формате, разделённые пробелами. Концы строк игнорируются.

Пример:

1.0e-3 5.0e-3 BIN ; пример режима BIN 0A DF 12 E3 98 08 18 FF FA FB

Приём в файл Возможна запись содержимого окна RX в файл с именем < Текущее имя >.RX Например, PETER.RX, PAUL.RX. Файлы должны быть в рабочей директории.

Замечания:

Малые окна TX/RX действуют как терминал TTY. Поэтому команды типа Copy/Cut/Paste и т.п. не действуют. Для очистки окон TX/RX следует использовать кнопку Clear.

Специальные символы типа < CR >, < ESC > и т.п. отображаются в виде соответствующего кода.

Уровни сигналов TX/RX равны VDD/VSS.

Если TTY настроен на 7 бит , а в программе используется 8 битный символ типа с, б, ц и т.п., в окне Messages появится предупреждение. 8 битные символы подчиняются стандартам Windows/ANSI, но не MS-DOS OEM.

Во время исполнения, можно изменять параметры TX/RX ,используя кнопку Set Parameters.

При задании параметров телетайпа вывод TX микроконтроллера должен быть присоединён к RX терминалу, а вывод RX - к TX терминала..

ЖКИ модуль (LCD module)

Синтаксис:

X[<ИмяЭкземпляра>] LCD(<chars> < lines > < osc_freq >) <RS> <RW> <E> <D7>...<D0>

где:

- inst_name – имя ЖКИ модуля; имя должно быть уникальным среди имен компонентов, начинающихся на X;
- chars – количество символов в строке индикатора;
- lines – количество строк индикатора;
- oscil_freq – частота работы контроллера ЖКИ;
- RS, RW, E – имена выводов, подключаемых соответственно к линиям RS, RW, E контроллера ЖКИ;
- D7, D6, D5, D4, D3, D2, D1, D0 – имена выводов,

подключаемых к соответствующим линиям данных контроллера ЖКИ.

Эта макромодель реализует модуль ЖКИ на основе контроллера с системой команд HD44780. Параметры экрана могут задаваться из следующего ряда: 8x1, 8x2, 16x1, 16x2, 16x4, 20x1, 20x2, 20x4, 24x2 , 40x2.

Возможность изменения частоты генератора позволяет имитировать реальные задержки выполнения операций.

Все возможности и команды ЖКИ в точности симулированы, за исключением знакогенератора (шрифт всегда фиксирован и не обязательно 100% соответствует реальному виду на ЖКИ)

Пример:

X1 LCD(24 2 250K) PD2 PB0 PD3 PA7 PA6 PA5 PA4 PA3 PA2 PA1 PA0

Советы:

При написании программы необходимо учитывать реальные характеристики ЖКИ модуля, поэтому следует внимательно изучить описание его команд и формировать необходимые временные задержки.

Используйте Log checkbox, чтобы отследить данные и операции ЖКИ в окне сообщений (Messages)

С помощью окошка метки (chekbox) по вашему выбору можно задать останов или продолжение на некорректные команды.

I2C монитор

Монитор I2C позволяет моделировать и отлаживать программы, использующие шину I2C (торговая марка Philips Semiconductors). Монитор может работать в режиме ведущего(Master) или ведомого(Slave) модуля.

Режим работы с несколькими ведущими в данной версии симулятора не реализован.

Синтаксис:

X[< Текущее имя >] I2C(< задающий_генератор > < адрес_ведомого >) < SDA > <SCL>

Параметры:

< задающий_генератор > - частота, которая будет использоваться модулем в процессе обмена. **Максимальная допустимая частота 400KHz.**

< адрес_ведомого > - **Адреса должны задаваться в десятичном формате (0-127).**

< SDA >, < SCL > выводы порта ОМК, реализующие I2C . К выводам необходимо подключить подтягивающие резисторы.

Пример:

Xone I2C(100K 24) PC1 PC0 ;

Частота 100КГц,

адрес ведомого = 24 (десятичный)

R1 VDD PC0 10K ;

R2 VDD PC1 10K ;

Экран монитора представлен на рис. 36, где показан прием байт 0xE1, 0xC0 ведущим от ведомого с адресом 72 с частотой 25 кГц.

Рис. 36. Монитор I2C

Это интерактивный элемент, с двумя кнопками Master TX и Master RX, которые позволяют задавать последовательности в этих режимах.

Кнопка Master TX задаёт передачу адреса и данных, взятых из двух строк ввода справа от кнопки. Байты данных должны быть шестнадцатеричного формата, разделяемые одним пробелом, например: 0A 3D 45 6A 78...

Кнопка Master RX задаёт приём (частота передачи, получаемые данные) из адреса, взятого в его соответствующей строке ввода (шестнадцатеричный).

В режиме ведомого Slave RX, элемент полностью пассивен, тогда кнопки не нужны. Все события, наблюдаемые на шине регистрируются в правом окошке: Start, Stop, Data RX, и т.д.

Замечание:

Шин I2C допускает подключение несколько ведомых модулей, использующих общие линии SDA SCL, по которым они могут обмениваться данными .

Клавиатура 4x4 (4x4 keypad matrix)

Представляет собой 4 строки и 4 столбца клавишного поля, соединённых в матрицу. Клавиатура панели управления (Control Panel) используется для интерактивного контроля 16 переключателями (кнопками).

Синтаксис:

X[Текущее имя] KEY4X4 < r0 > < r1 > < r2 > < r3 > < c0 > < c1 > < c2 > < c3 > < r0 > это вывод строки-0 (самый верхний) , и т.д. < c0 > это вывод столбца-0 (самая левая), и т.д.

Кнопка 0 панели управления соединяет выводы < r0 > , < c0 >.

Кнопка 1 соединяет выводы < r0 > , < c1 > , и так далее

Пример:

XkeyPad KEY4X4 PA0 PA1 PA2 PA3 PA4 PA5 PA6 PA7

Советы и ограничения:

Не подсоединяйте выводы питания (VDD / VSS / GND) к матрице клавиатуры. Для этой цели лучше использовать одиночный переключатель/ключ Kx.

Разрешена организация клавиатуры только 4x4, т.к. на панели управления доступны только 16 кнопок. Если используются элементы, активизируемые кнопками такие, как NRZ-генератор при нажатой кнопке выполняется два (или больше) действия одновременно: например, замыкание клавиши и запуск временной диаграммы. Клавишную матрицу 4x4 можно рассматривать как 16 автономных ключей (кнопок).

При отсутствии модели требуемого компонента пользователь может его разработать самостоятельно. Методика разработки новых моделей компонентов изложена в [4].

5 Порядок выполнения работы

1. Получите задание у преподавателя
2. Создайте Project File. Используйте опции New Project или Open Project (для открытия уже существующего проекта)

При создании нового проекта (New Project) необходимо реализовать основные этапы подготовки проекта, описанные в разделе 2, после выполнения которых формируются шаблоны файлов проекта и кодов.

Изменить предлагаемые настройки проекта можно директивами, изложенными в разделе 3.

При определении конфигурации и состава внешних периферийных устройств рекомендуется использовать Components menu.

При выборке соответствующего компонента предлагается шаблон, скорректировав который в соответствии с требованиями раздела 4 , формируется файл проекта.

Если требуемый компонент отсутствует, его можно создать на основании методики, приведенной в Приложении 2.

3. Создайте или отредактируйте файл кодов, используя встроенный текстовый редактор. Шаблон файла кодов располагается в окне Code Notebook
4. При создании файла кодов следует руководствоваться описанием программной модели ОМК и ассемблера.
5. При редактировании файлов проекта и кода можно использовать фрагменты ранее разработанных программ.
6. Выполните компиляцию проекта – командой Build.
7. Любые ошибки/предупреждения (error/warning) высветятся в окне Messages Window.
8. Нажав на сообщение об ошибке (двойной клик), можно перейти к строке кода, на которую ссылается ошибка.

9. Когда все ошибки устранены, VMLAB включает в светофоре зеленый цвет, разрешая отладку в выбранном режиме (меню RUN).

10. После компиляции проекта автоматически выбираются заданные в нем интерактивные компоненты, необходимые для управления процессом отладки.

Дополнительные средства можно установить из меню View.

Распределение окон по экрану выполняется в Window.

В Options задаются основные настройки эмулятора.

5. Если в процессе отладки программы не требуется точек останова , то можно начать отладку в непрерывном режиме.

При обнаружении ошибок или неверном результате выполнения тестовой программы обработки, рекомендуется перейти в один из пошаговых режимов, выполнив предварительно рестарт программы.

При этом можно отслеживать: переменные, аппаратные сигналы, регистры ОМК, память и периферию, наблюдать ход исполнения команд , помечать точки останова и т.п.

6. Если приложение работает нормально, то генерируется выходной файл (target .HEX file), готовый для записи во флэш-память ОМК , EEPROM, OTPROM и т.п.

7. Длину программы и время её выполнения можно определить в окне Messages. При этом следует учитывать, что при запуске программы вначале формируется сброс длительностью 2 мс (ATmega 8535). Это значение следует вычесть из общего времени работы программы.

В приложениях А, Б приведен пример проекта, реализующего цифроаналоговый преобразователь ЦАП на базе широтно-импульсного модулятора ШИМ, управление выходным напряжением которого выполняется с помощью двух кнопок. При нажатии кнопки К1 напряжение увеличивается, а К0 – уменьшается. Для преобразования выходного сигнала ШИМ в напряжение используется внешняя RC-цепь. Контроль выходных сигналов выполняется осциллографом.

Схема соединения внешних компонентов описывается в файле .prj (см. раздел 4, приложение А).

В качестве счетчика выбран СТ1, работающий в режим 8 – разрядного ШИМ. Для настройки на этот режим задаем соответствующие управляющие слова:

TCCR1A - 0b10000001;выход схемы сравнения А в исходном состоянии равен 0, режим1;

TCCR1B - $01- режим1, коэффициент предделителя 1

Частота ШИМ для этого режима Ftck/510, где Ftck-частота на выходе предделителя.

Длительность импульса определяется T= (OCR1A)*2*Ttck, где OCR1A –16 разрядный регистр сравнения, Ttck – период выходной частоты предделителя.

Значение регистра сравнения OCR1A в процессе работы определяется количество нажатий кнопок К0 и К1. Запись выполняется в младший байт- OCR1AL, а старший байт обнуляется.

Работа ЦАП выполняется по прерыванию. Кнопки К0 и К1 присоединены соответственно к входам внешних прерываний INT0 и INT1. Для настройки подсистемы прерывания необходимо :

в регистре управления MCUCR установить вид внешнего запроса, например , по срезу входного импульса - $0A,

в регистре маски внешних прерываний GIMSK установить разрешение -$C0,

в регистре маски таймеров TIMSK разрешить прерывание по переполнению СТ1 - $04,

выполнить общее разрешение прерываний командой SEI.

Контрольные вопросы

Какие средства отладки могут использоваться при разработке микропроцессорных систем?

В чем преимущества и недостатки симуляторов?

Какие компоненты должны входить в состав симуляторов?

В чем преимущества и недостатки оценочных модулей?

Какие преимущества обеспечивает совместное использование симулятора и оценочного модуля?

Какими возможностями обладает симулятор VMLab?

Какие основные этапы подготовки программы для отладки в симуляторе?

Каков алгоритм создания проекта?

Какие внешние компоненты можно подключать в VMLab?

С помощью каких средств можно эмулировать аналоговые сигналы? цифровые сигналы?

Какие последовательные каналы позволяет использовать VMLab? Какие особенности работы с ними?

Как подключить клавиатуру? кнопку?

Какие средства отображения информации входят в VMLab?

Как создать файл программы?

Как исполнить программу в шаговом режиме? непрерывном? с точками останова?

Какие средства контроля ресурсов ОМК предусмотрены в VMLab? Как их активизировать?

Как изменить содержимое ячеек памяти или регистров?

Как оценить параметры разработанной программы (объём, время выполнения)?

Как организовать работу в многопроцессорном режиме?

Список использованных источников

1.VMLab 3.12 for AVR.[Электронный ресурс]. www.amctools.com

2.Справка по ассемблеру AVR. [Электронный ресурс]. www.microcon.euro.ru

3.Зубарев А.А. Программная эмуляция устройств на микроконтроллерах в среде Visual Micro Lab. [Электронный ресурс]. www.bek.sibadi.org

4. Интерфейс программирования пользовательских компонентов в VMLab [Электронный ресурс]:http//www.avr.h15.ru

```
;
*****************************************************************
;  Программа демонстрирует реализацию ЦАП с помощью ШИМ
;  Сгенерированный сигнал можно наблюдать в окне Scope
;  Генерируемое напряжение изменяется кнопками 0 и 1
;В качестве счетчика используется CT1, работающий в режиме ШИМ.
;На выходе ШИМ установлен RC фильтр. Ввод - по прерыванию.
;
*****************************************************************

; -------------------------------------------------------
.MICRO "ATmega8535"; Используемый микроконтроллер
.PROGRAM "dac_m.asm"; Имя файла с программой для микроконтроллера
.TARGET "dac_m.hex"; Имя полученного в результате компиляции hex файла

; -------------------------------------------------------
.POWER VDD=5 VSS=0   ; Напряжение питания
.CLOCK 1meg; Частота тактового генератора микроконтроллер
.STORE 250m; Ширина временного окна осциллографа
; -------------------------------------------------------

; Кнопка K0 уменьшает скважность (отношение длительности выходного сигнала ШИМ к ;периоду)
; Подключаем кнопку K0 к входу INT0
K0 PD2 VSS
R0 VDD PD2 10K ; подтягивающий резистор
```

```
; Кнопка K1 увеличивает скважность
; Подключаем кнопку K1 к входу INT1
K1 PD3 VSS
R1 VDD PD3 10K ; подтягивающий резистор

; RC-фильтр. На вход фильтра поступает выходной сигнал ШИМ с вывода PD5, а выход - ;out1
R2 PD5 out1 10K
C1 out1 vss 500n; (n- нанофарада)

; Выводы сигналов, которые будут отображаться на осциллографе
.plot P1(PD5)
.plot P2(out1)
```

Приложение Б. Программа реализации ЦАП на ассемблере

```
include "C:\VMLAB\include\m8535def.inc"

.DSEG          ; Указатель на работу с сегментом данных
   .org $60
Value:.byte 1; Резервирование ячейки памяти, в которой хранится значение для ШИМ
.CSEG  ; Указатель на работу с сегментом кода

Reset:      rjmp Main
            rjmp OnINT0    ; Addr $01
            rjmp OnINT1 ; Addr $02
            reti     ; Addr $03
            reti     ; Addr $04
            reti      ; Addr $05
            reti      ; Addr $06
            reti      ; Addr $07
            rjmp OnTimer1Ovfl; Addr $08
            reti      ; Addr $09
            reti      ; Addr $0A
```

```
                reti    ; Addr $0B
                reti    ; Addr $0C
                reti    ; Addr $0D
                reti    ; Addr $0E
                reti    ; Addr $0F
                reti    ; Addr $10
        ; обработчик внешнего прерывания INT0
        ; по нажатию K0 уменьшается значение переменной Value
        OnINT0:
        push r16
        lds R16, Value
                tst R16
                breq OnINT0_L0
                subi R16, 0x0F;шаг регулирования напряжения
                sts Value, R16
         pop r16
        OnINT0_L0:
        reti
        ; обработчик внешнего прерывания INT1
        ; по нажатию K1 увеличивается значение переменной
Value
        OnINT1:
        push r16
        lds R16, Value
         cpi R16, 0xFF
         breq OnINT1_L0
         subi R16, -0x0F
                sts Value, R16
         pop r16
        OnINT1_L0:
        reti

        ; обработчик переполнения таймера 1
        ;  в регистр ШИМ загружается значение переменной Value
        OnTimer1Ovfl:
                push r16
                  clr R16
```

```
        out OCR1AH, R16
        lds R16, Value
        out OCR1AL,   R16
          pop r16
        reti

Main:       ; Установка указателя стека
        ldi R16,high(ramend)
        out SPH, R16
        ldi R16, low(ramend)
        out SPL, R16
        clr R16     ; Сброс переменной Value
        sts Value, R16
        out tcnt1h,r16 ;Сброс счетчика
        out tcnt1l,r16
; настройка выходов ШИМ D5(схема сравнения A), D4(схема сравнения B) на вывод
        sbi DDRD, 4
        sbi DDRD, 5
        ; Разрешение прерывания при переполнении Timer 1
        ldi R16,0b00000100
                out TIMSK, R16
        ; настройка Timer 1
        ldi R16, 0b10000001   ; При сравнении выв. D5 сбрасывается в 0,режим 8 битного ШИМ
        out TCCR1A, R16
        ; Настройка подсистемы прерываний
        ldi R16, 0b00001010   ; Настройка запросов INT0, INT1 на срез сигнала
        out MCUCR, R16
        ldi R16, 0b11000000   ; разрешение внешних прерываний INT0, INT1
        out GIMSK, R16
        sei ; Общее разрешение прерываний
        ldi R16, 0b00000001   ; Настройка делителя частоты(коэффиц. делен.= 1)
        out TCCR1B, R16
```

```
; цикл ожидания прерываний
Main_L0:    rjmp Main_L0
```

Методические материалы

Отладка программ на симуляторе VMLab 3.12

Методические указания

Составитель ***Иоффе Владислав Германович***

Самарский государственный
аэрокосмический университет.
443086 Самара, Московское шоссе, 34.
