РАДИО-ЕЖЕГОДНИК

FLOWCODE 6
- новые полёты

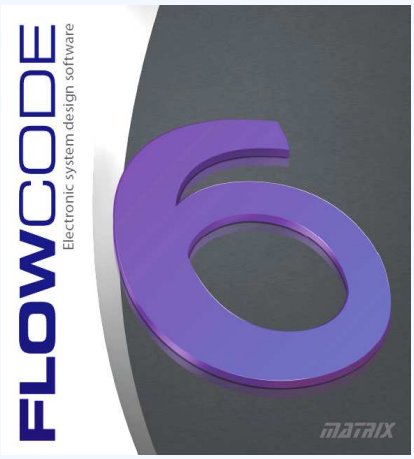
ВЫПУСК
29

2013

РАДИО - ЕЖЕГОДНИК 2013 выпуск 29

ТЕМАТИЧЕСКИЙ ОБЗОР ПЕЧАТИ И ИНТЕРНЕТ-РЕСУРСОВ

ТЕМА НОМЕРА:



FLOWCODE 6 - новые полёты

Выпуск приурочен к выходу новой версии популярной программы графического программирования микроконтроллеров **FLOWCODE 6**

Выпускающий редактор: *С. Степанов*

Над выпуском работали: *С. Муратчаев*, *С. Скворцов*, *В. Гололобов*, *В. Смирнов*

Художник: *О. Агафонов*

E-mail: ***radioyearbook@gmail.com***

Октябрь 2013

Информационная поддержка: *Портал "РадиоЛоцман"* ***www.rlocman.ru***

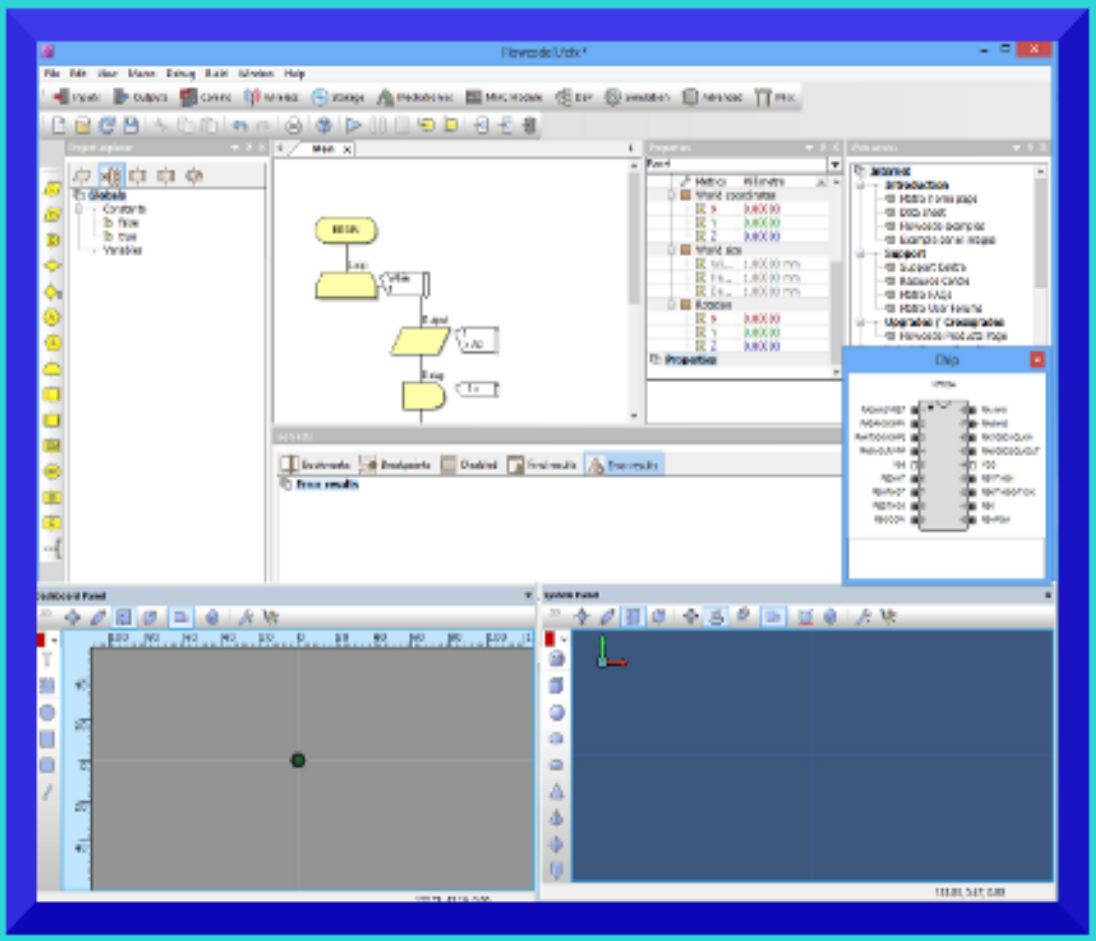
Официальные версии журнала доступны для свободной загрузки: ***www.rlocman.ru/radioyearbook***

СОДЕРЖАНИЕ

Все материалы для сборника любезно предоставлены автором и переводчиком
Владимиром Николаевичем Гололобовым *http://vgololobov.narod.ru/*
e-mail: vgololobov@yandex.ru

В.Н. Гололобов

Flowcode 6



(Заметки к появлению новой версии)

Москва 2013

Оглавление

Предисловие

Не так давно появилась шестая версия программы Flowcode. К использованию графического языка программирования и любители, и профессионалы относятся по-разному, кому-то нравится, кому-то нет, между тем популярность программы Flowcode растёт.

Рассказывая о программе, желательно повторять все операции, о которых идёт речь – так можно избежать ошибок и опечаток, связанных с изменениями в программе на момент её выхода в свет. То, что опечатки могут иметь место в руководствах и файлах помощи, можно убедиться самостоятельно, если внимательно прочитать, например, упражнения.

Для рассказа я использую пробную версию, а подробность и полнота описания будут ограничены сроком действия пробной версии.

Получить пробную версию программы Flowcode 6 можно на сайте Matrix Multimedia: http://www.matrixmultimedia.com/index.php, достаточно зарегистрироваться на сайте, чтобы активировать программу после установки. Установка проста и ничем не отличается от установки любой программы в Windows. С первым запуском программы после установки потребуется ввести ваш логин и пароль, которые вы использовали при регистрации. С приобретением программы больше проблем. Как заявлено производителем, расчёт только в валюте Великобритании. Правда, есть возможность обратиться к региональным распространителям программы. Но начинать лучше с демонстрационной версии, чтобы убедиться в необходимости покупки.

Вот, что пишут о программе её создатели:

Программа Flowcode позволяет вам быстро и легко разрабатывать сложную электронику и электромеханические системы. Инженеры используют Flowcode для разработки систем управления и измерения на базе микроконтроллеров, сложных индустриальных интерфейсов или совместимых с Windows персональных компьютеров.

Разработка

Создайте модель электронной системы, используя разделы System designer и Component library. Затем разработайте программу для управления системой. Dashboard designer позволяет вам разработать HMI.

> Человеко-машинный интерфейс (ЧМИ) (англ. Human machine interface, HMI) — широкое понятие, охватывающее инженерные решения, обеспечивающие взаимодействие человека-оператора с управляемыми им машинами (Wiki).

Component creator позволит вам собрать части и протестировать движение электромеханической системы.

Симуляция

Симулятор покажет вам, как ваша система будет работать. Трёхмерный симулятор покажет движение вашей электромеханической системы под воздействием электронных стимулов. Dashboard HMI покажет реальные общие значения. Интерфейс Application Program позволит вам соединить вашу встроенную систему с функциями PC и инструментами других производителей.

Внутрисхемный тест

Скомпилируйте вашу разработку в микроконтроллер или соедините его с интерфейсом PC. Пройдите программу в чипе и одновременно на экране. Dashboard HMI поможет вам наблюдать воплощение вашей системы, а API позволит вам интегрировать данные от сторонних производителей инструментов в вашу систему.

Реализация

Когда вы убедились, что с вашей разработкой всё в порядке, воспроизведите вашу систему для использования другими. Если вы полноправный автор, вы можете изготовить один или сотни тысяч устройств. Используйте автоматический документатор, чтобы помочь другим лучше разобраться с вашим проектом.

PROJECT EXPLORER

- сразу можно видеть все порты, макросы, переменные, константы и компоненты в вашем проекте.

COMPONENT TOOL BAR

- выберите ваш электромеханический компонент из нашей большой библиотеки, от простого выключателя до модуля Bluetooth.

FLOWCHART PROGRAM

- перетащите и отредактируйте стандартные иконки программных компонентов для вашей программы. Создайте макросы подпрограмм, которые можно вызвать из других иконок. Используйте мощный язык Flowcode для управления, для внешних инструментов и наблюдения за вашей системой.

ICON TOOL BAR

- перетащите стандартные иконки в вашу программу. Щёлкните, чтобы отредактировать свойства для синтаксически правильной программы.

C CODE PROGRAM

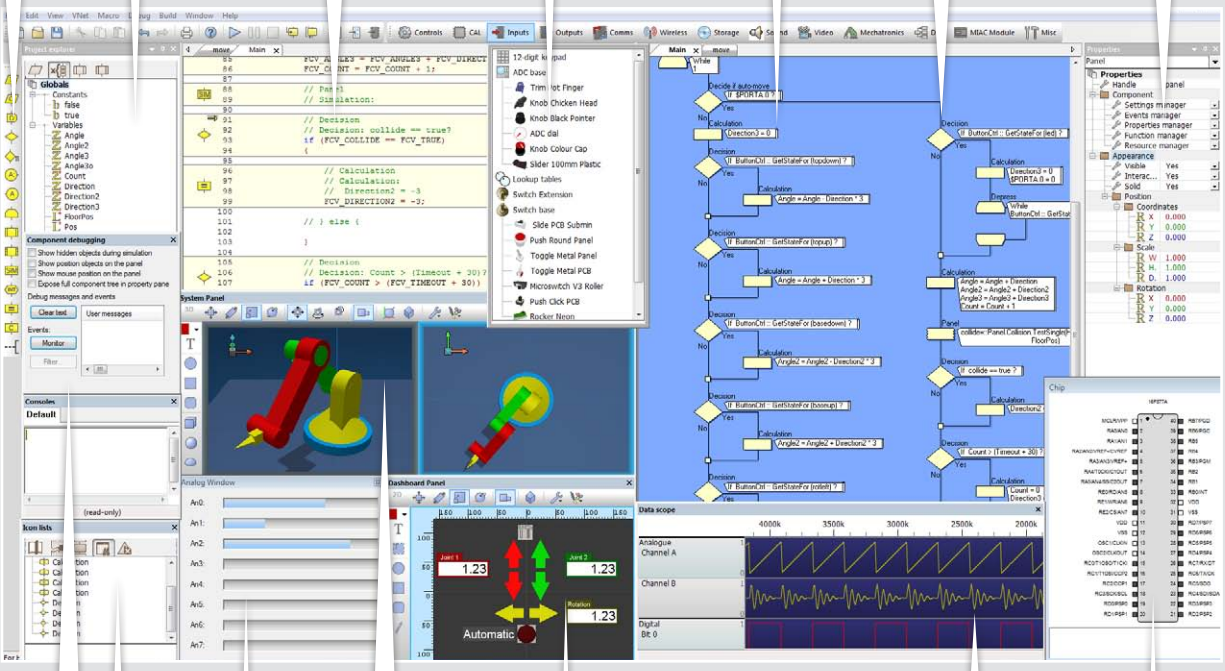
- наблюдайте за эквивалентным Си кодом вашей программы; очень быстро синтаксически правильный код генерируется автоматически на основе иконок программных компонентов.

CONTROL TOOL BAR

- используйте стандартную инструментальную панель для редактирования вашей программы, а также для симуляции программы и запуска внутрисхемной отладки.

PROPERTIES EDITOR

- просматривайте и редактируйте свойства всех компонентов.



COMPONENT DEBUG

- наблюдайте за вызовами API в вашей программе и разработанных компонентах.

SYSTEM PANEL

- разрабатывайте вашу систему, используя системную панель в нескольких проекциях. Используйте готовые электромеханические компоненты или создайте собственные. Импортируйте вашу модель из программы подобно Sketchup или Solidworks.

SCOPE WINDOW

- используйте окно осциллографа для наблюдения за разными сигналами в вашей системе. Подключите осциллограф для симуляции данных или воспроизведения реальных данных при внутрисхемной отладке.

ICON LIST WINDOW

- для поиска результатов, сообщений об ошибках, точек остановки и закладок.

DASHBOARD PANEL

- управляйте и наблюдайте за вашей программой при симуляции или внутрисхемной отладке. Пишите программы, используя симуляцию команд API, чтобы показать эквивалентное поведение в реальном мире ваших данных в формате, понятном человеку.

CHIP

- используйте окно микросхемы, чтобы видеть и управлять состоянием входов и выходов вашего микроконтроллера при симуляции и внутрисхемной отладке.

ANALOGUE WINDOW

- наблюдайте за состоянием аналоговых входов в вашей разработке.

Глава 1. Что нового на первый взгляд?

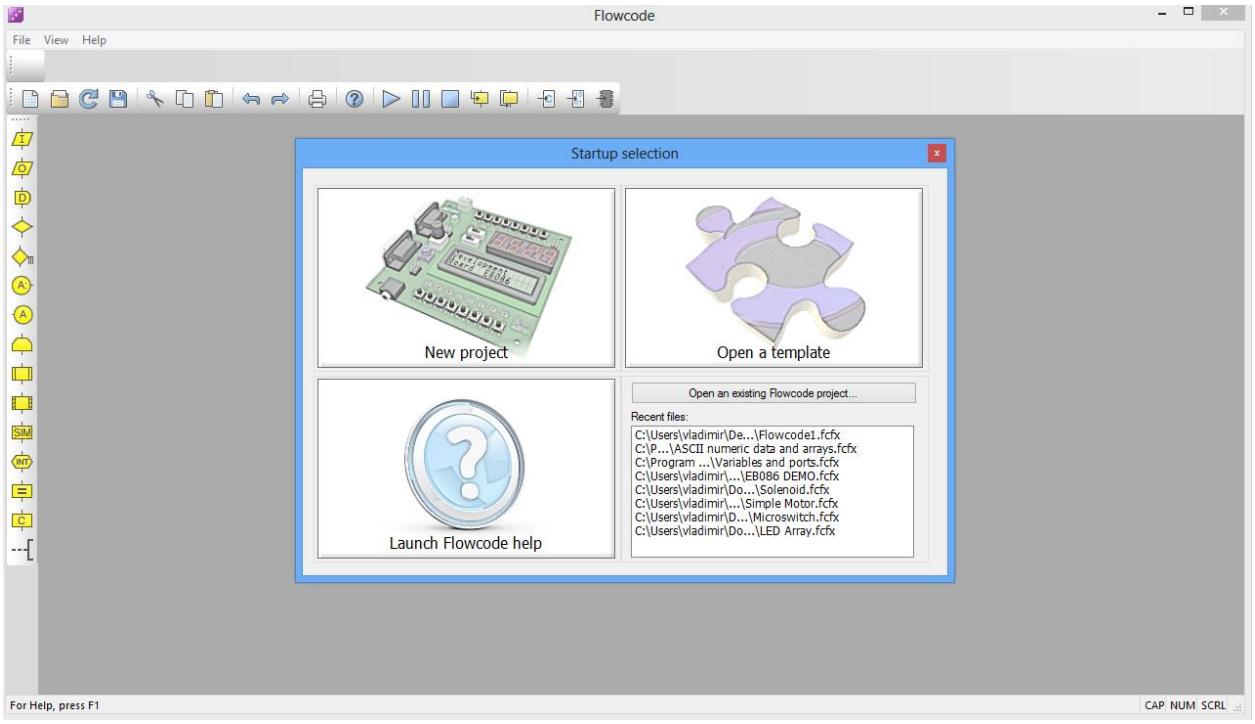


Рис. 1.1. Программа Flowcode 6 после запуска

Да, несколько изменилось меню выбора дальнейшей работы. Посмотрим, что предлагает каждый из «пунктов» этого меню. New project подразумевает создание нового проекта.

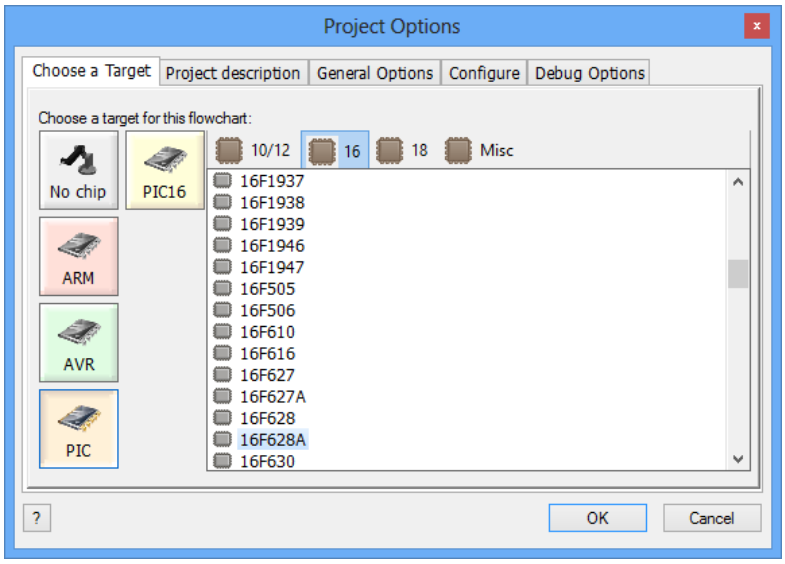


Рис. 1.2. Меню создания нового проекта

Это, скажем, первая неожиданность – вы можете выбрать создание проекта на основе PIC, AVR или ARM микроконтроллера. И в каждой группе есть из чего выбрать. На закладках вы найдёте необходимые настройки проекта, как его описание, конфигурация микроконтроллера или опции отладки.

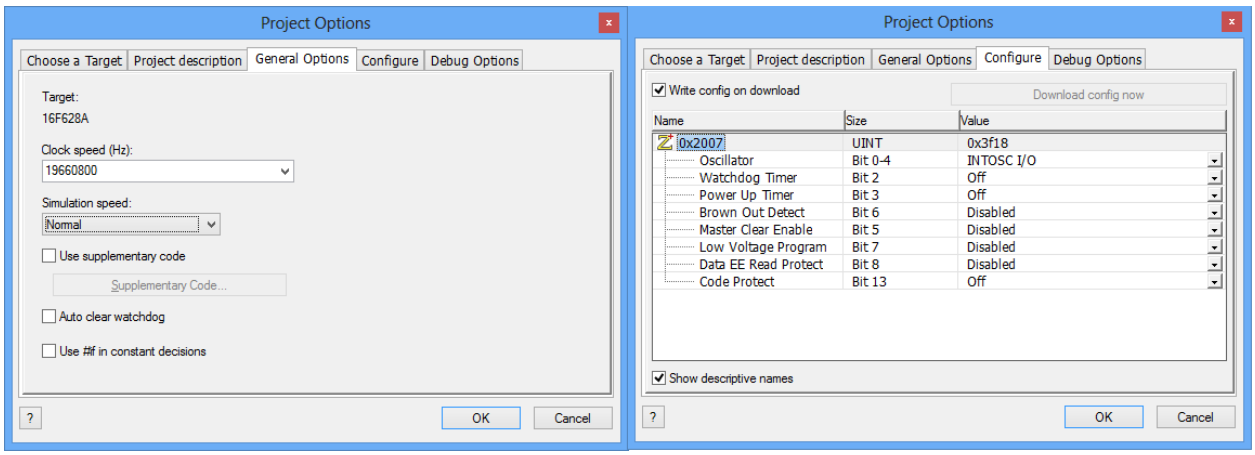


Рис. 1.3. Некоторые закладки для настройки проекта

Если вы не готовы определить все свойства вашего проекта, то можно будет сделать это позже, используя основное меню программы. После нажатия на кнопку **ОК** вы видите несколько окон программы – рабочее поле, системную панель и панель управления (две панели появились в этой версии). Если вы их не увидели, то можете вызвать их в основном меню программы (View). При выборе компонентов обрамления (не компонентов программы, а дополнительных устройств) вы может решить, на какую панель их добавить:

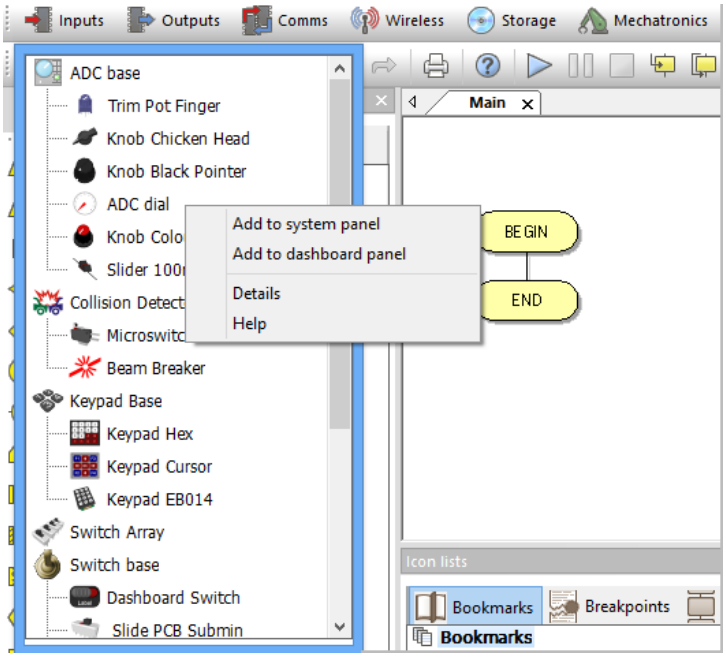


Рис. 1.4. Выбор места размещения дополнительных элементов

Если вы выбрали панель управления, то выбранный элемент появится на ней. К слову, все окна – и рабочее окно для сборки программы, и системная панель, и панель управления – масштабируемые, можно легко изменить размер окна, поместив мышку к краю окна или в угол окна, нажать левую клавишу мышки и перетащить курсор, увеличив окно до нужного размера. И окна могут оставаться плавающими, их легко перемещать по экрану, а могут быть закреплены в нужном вам месте основного окна, как вам больше нравится, как вам удобнее работать с программой.

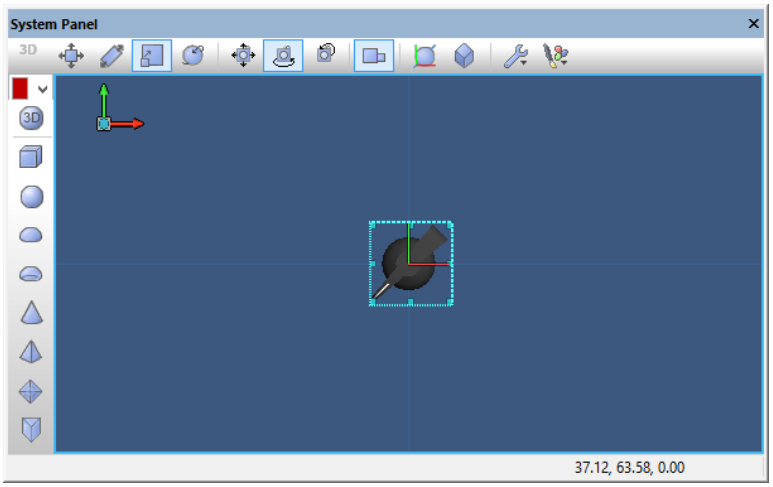


Рис. 1.5. Добавление нового элемента на системную панель

Выделив элемент щелчком мышки, обратите внимание на окно свойств, где появляются свойства этого элемента.

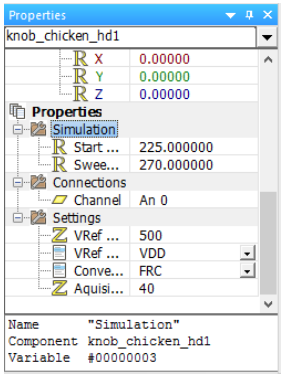


Рис. 1.6. Окно свойств добавленного элемента

В свойствах компонента вы можете задать его положение:

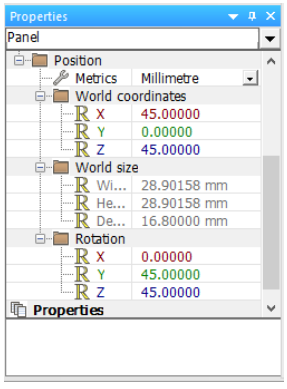


Рис. 1.7. Изменение угла наблюдения в свойствах компонента

Для многих элементов очень важен пункт Connections (подключение), если вы не задали подключение, например, светодиода к выводу порта, то программа не будет транслироваться в hex-файл. И, наконец, на системной панели вы можете увидеть кнопку с надписью «3D». Да, можно видеть трёхмерный вид добавленных трёхмерных компонентов.

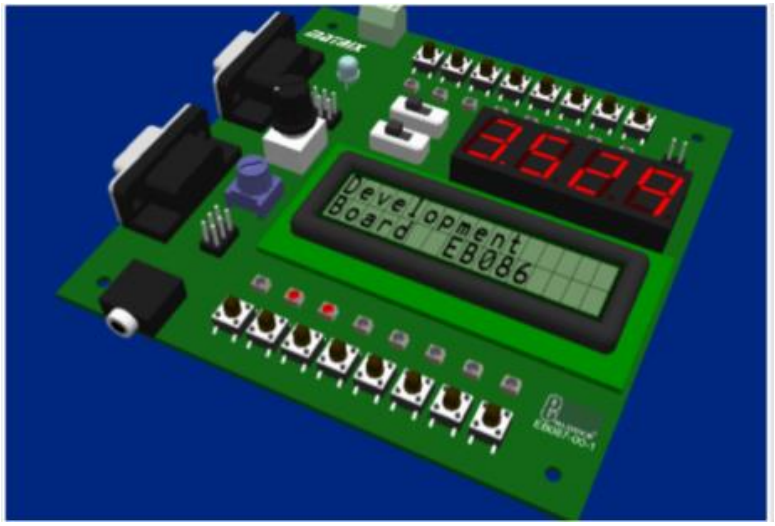


Рис. 1.8. Трёхмерный вид компонентов

Но вернёмся к началу работы с программой. Мы использовали создание нового проекта, но это не единственный выбор. Начиная работу, можно загрузить шаблон – раздел «Open a template»:

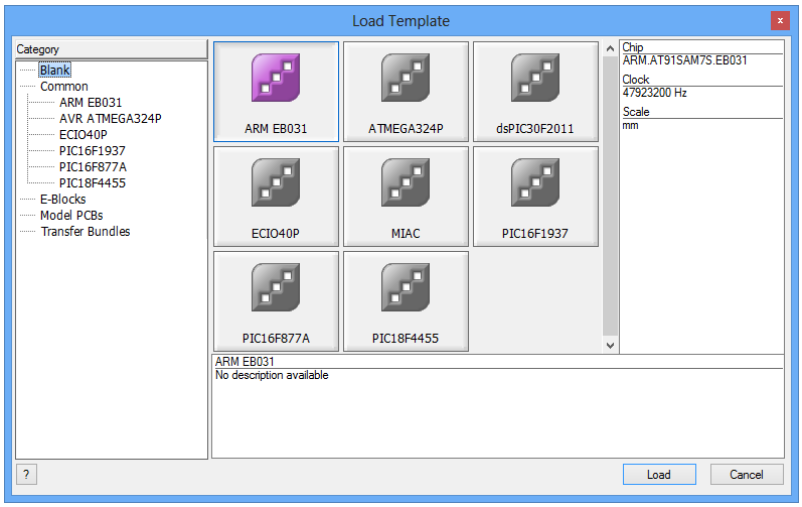


Рис. 1.9. Выбор шаблона для работы

И здесь, как можно видеть, богатый выбор – переберите все представленные разделы, чтобы увидеть возможности в окне справа.

Предпоследний раздел – это помощь.

Все современные средства помощи очень широко используют Интернет, программа Flowcode 6 не исключение. Описание в Wiki, рассказ на YouTube. Но и Интернет сегодня не диковинка, а привычное средство общения, хороший справочник и удобный инструмент для разработчиков программ.

Рис. 1.10. Окно выбора подсказок и помощи

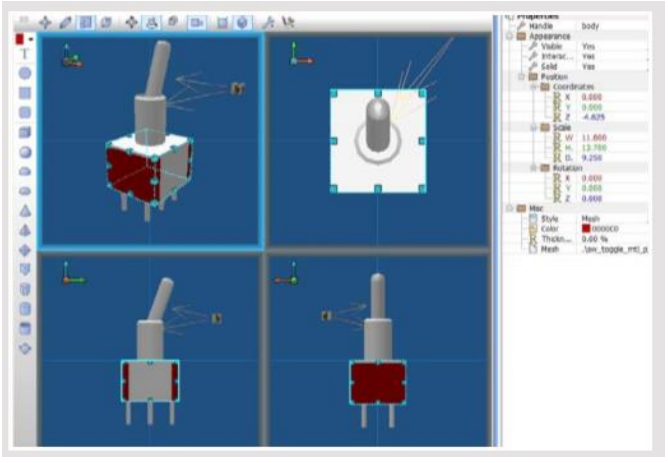
И файлы примеров, и видео уроки, и форум – всё это будет вам полезно, если вы подключены к Интернету. То, что урок на английском, не мешает увидеть основные приёмы работы с программой, а вы знаете, что лучше один раз увидеть, чем сто раз услышать.

В последней четверти приглашения к началу работы вы найдёте те проекты, с которым недавно работали. Если они, конечно, есть.

Чем эта версия программы отличается от предыдущих версий. Слово её разработчикам:

Что нового в Flowcode 6?

Теперь вы можете создавать свои собственные электронные компоненты и добавлять их в библиотеку компонентов (component library). Вы можете использовать вызов симулятора API для определения электрического и механического поведения при симуляции.



Расширение библиотеки компонентов

Библиотека компонентов (component library) была существенно расширена, чтобы включить в неё много новых электронных компонентов и их симуляцию. Компоненты могут разрабатываться и распространяться через наш web-сайт.

Системные компоненты

В дополнение к PCB стилю компонентов вы теперь имеете доступ к смонтированным на панели переключателям, измерителям и дисплеям для индустриального контроля.

HMI компоненты приборной панели

Поправьте или создайте компоненты приборной панели подобные графическим, цифровым и измерительным элементам, что позволит вам наблюдать, как функционирует ваша система при симуляции и внутрисхемном тестировании.

Симуляция ближе к реальному времени

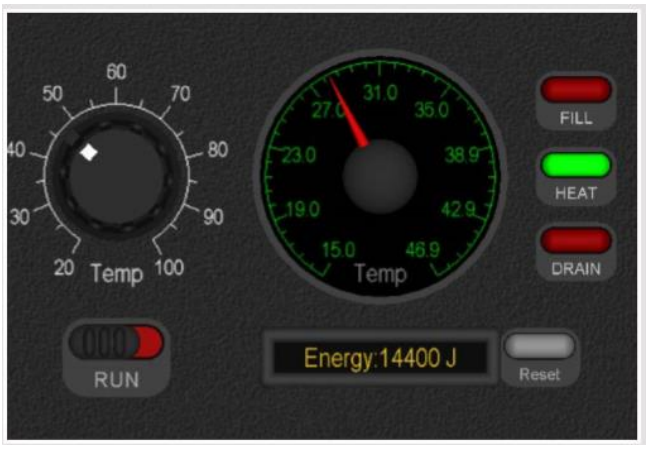
Удобство в скорости симуляции означает, что ваша симуляция работает ближе к реальному времени, позволяя вам проверить вашу разработку в живом виде.

API (Application Programming Interface, программный интерфейс приложения)

Мощный API позволяет управлять событиями симуляции и поддержкой компонентов в широком диапазоне функций на стороне PC.

HMI панели управления

Компоненты Human Machine Interface показывают параметры вашей системы в процессе симуляции и внутрисхемного тестирования, используя интуитивное отображение, включающее измерители, графики, осциллограммы и таблицы.



Консоли

Используйте текстовые консоли, чтобы увидеть данные вашей системы: прекрасно подходит для разработки систем с цифровыми коммуникациями.

Создание электромеханических систем

Обеспечивает симуляцию вашей модели, движущейся в трёхмерном пространстве под воздействием электрических стимулов от микроконтроллера и других компонентов вашей системы. Используйте двухмерную панель управления для наблюдения за вашей системой в режиме реального времени.

Системная панель

Используйте системную панель для разработки примера вашей собственной трёхмерной модели. Осмотрите вашу разработку под разными углами при её симуляции.

Лазерные резаки и 3D принтеры

Создайте дешёвые части с помощью лазерных резаков и 3D принтеров, и посмотрите их в работе с вашей электроникой на экране.

Окружение 3D разработки

Импортируйте 3D модели и определите движение и симуляцию, используя API.

Поддержка инструментов сторонних производителей

Есть доступ к чтению и данным внешних инструментов, используя DLL поддержку. Отображайте данные на приборной панели или с помощью программ сторонних производителей.

Полная интеграция E-BLOCKS

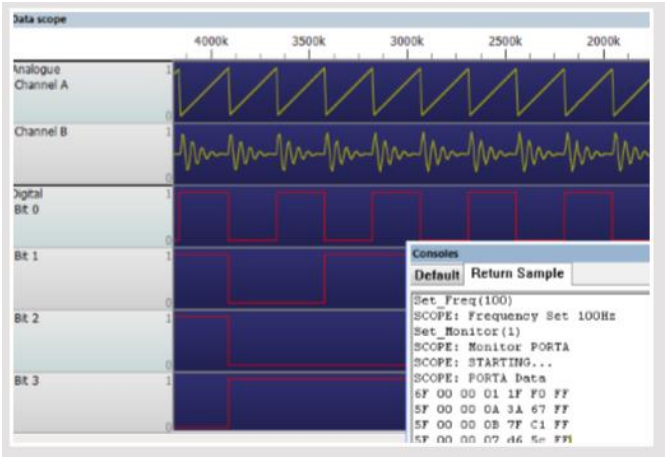
Используйте наш новый EB006 программатор для наблюдения за каждым выводом вашей разработки PICmicro, наблюдайте и интерпретируйте последовательные данные ввода/вывода.

Приборная панель HMI

Наблюдайте, как функционирует ваша система при симуляции и внутрисхемном тестировании. Используйте новые компоненты, как графики, консоли и прокручиваемые текстовые окна, чтобы проверить вашу разработку.

Осциллограф и консоли

Используйте программный осциллограф и консоли, чтобы видеть данные в виде сигналов или в текстовом формате. Используйте API, чтобы транслировать входные данные в шестнадцатеричном или ASCII эквиваленте. Подключите Softscope и консоли к оборудованию сторонних производителей, используя DLL, для создания полной SCADA системы.



Ещё один верный способ посмотреть, что нового в этой версии – скачать примеры, загрузить их в программу, как, например, такой:

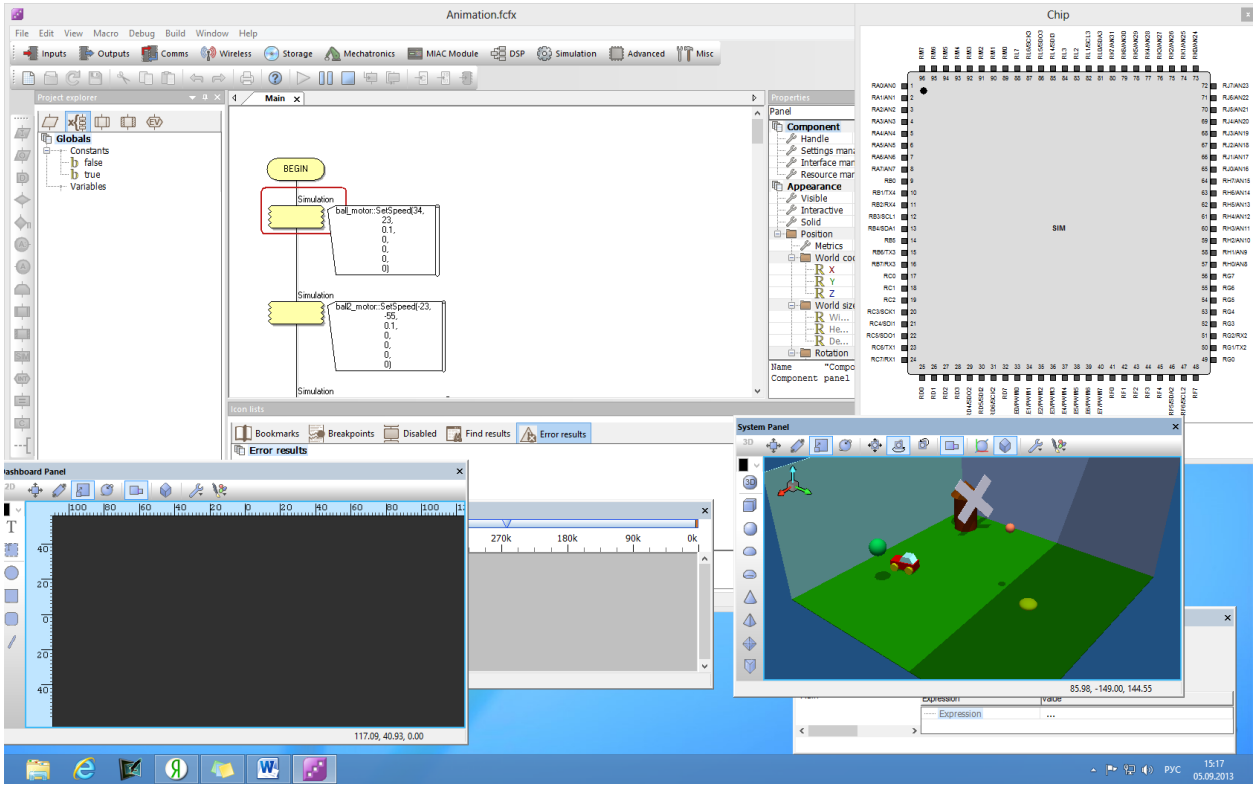


Рис. 1.16. Пример из подборки Flowcode 6

И мельница, и мячи, и автомобиль – всё в движении на площадке 3D.

Ещё один пример анимации с линейками светодиодов в трёхмерном пространстве, думаю, тоже отличительная черта новой версии.

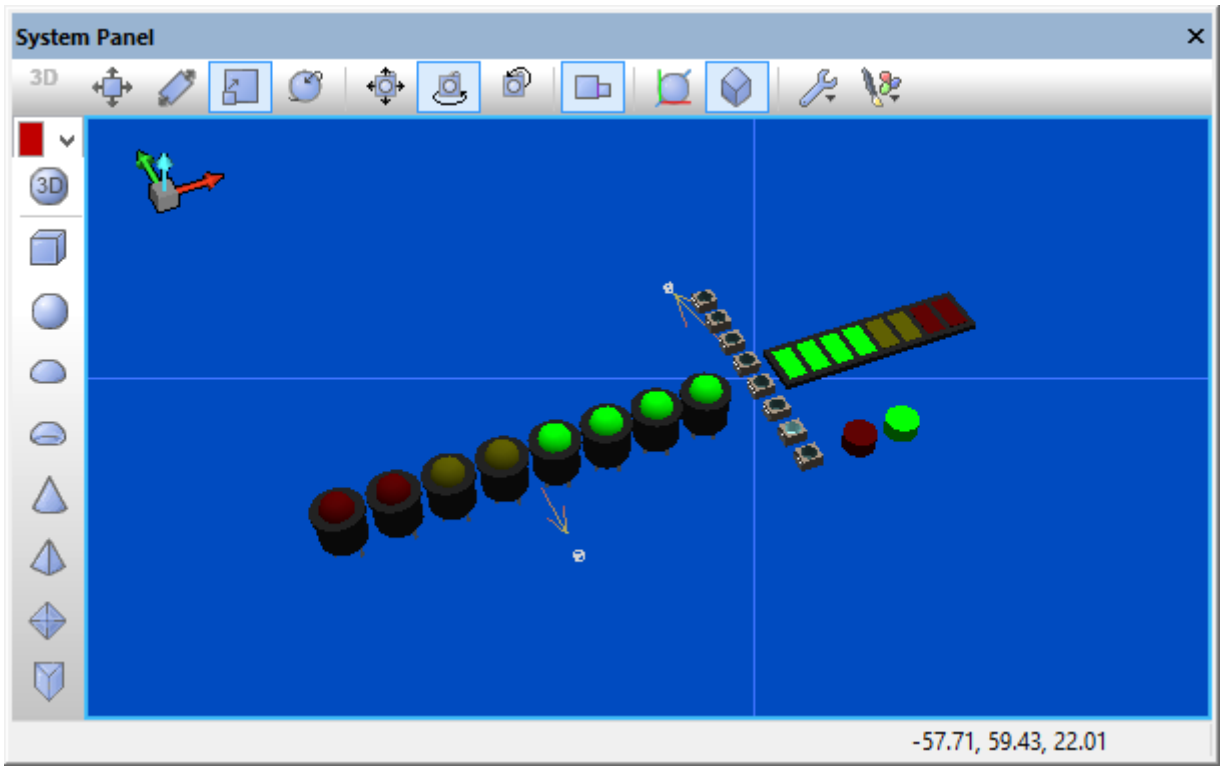


Рис. 1.17. Моделирование работы линейки светодиодов

Внимательно разбирая примеры, всегда можно найти что-то новое, если не в версии, то для себя… или всё-таки новое в версии?

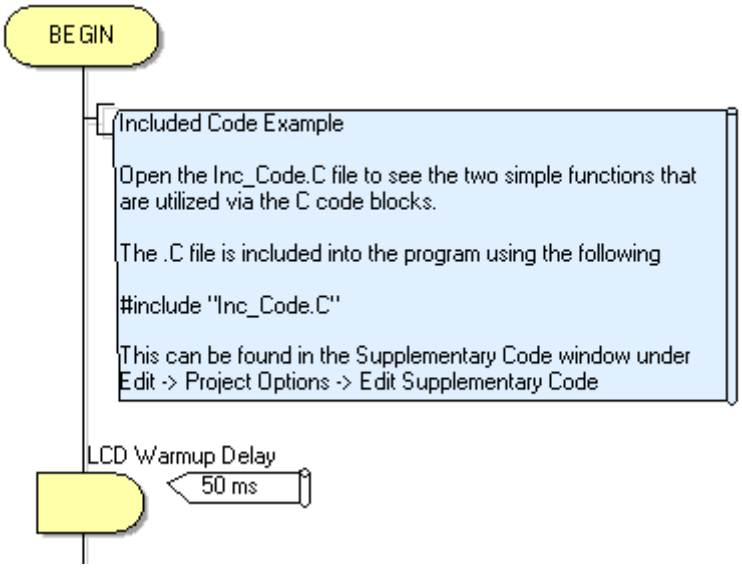


Рис. 1.18. Вставка внешнего текста на Си

Начинающим бывает порой трудно разобраться с тем, как добавить текст на языке Си или ассемблере, если не хватает возможностей Flowcode, из-за особенностей использования переменных. В примерах есть подробное пояснение.

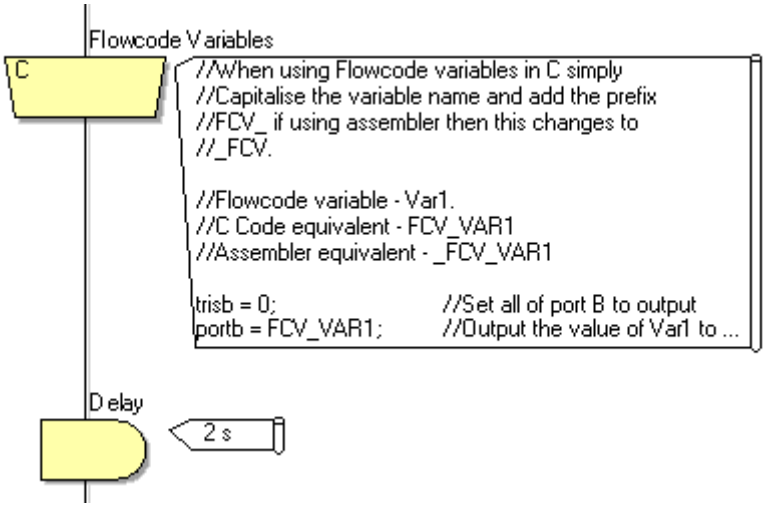


Рис. 1.19. Как используются переменные во вставках на языке Си и ассемблере

Примеры, их, наверное, создают профессиональные программисты, работающие над программой, показывают ряд эффектных решений, которые наверняка будут стимулировать к созданию не менее удачных собственных приложений.

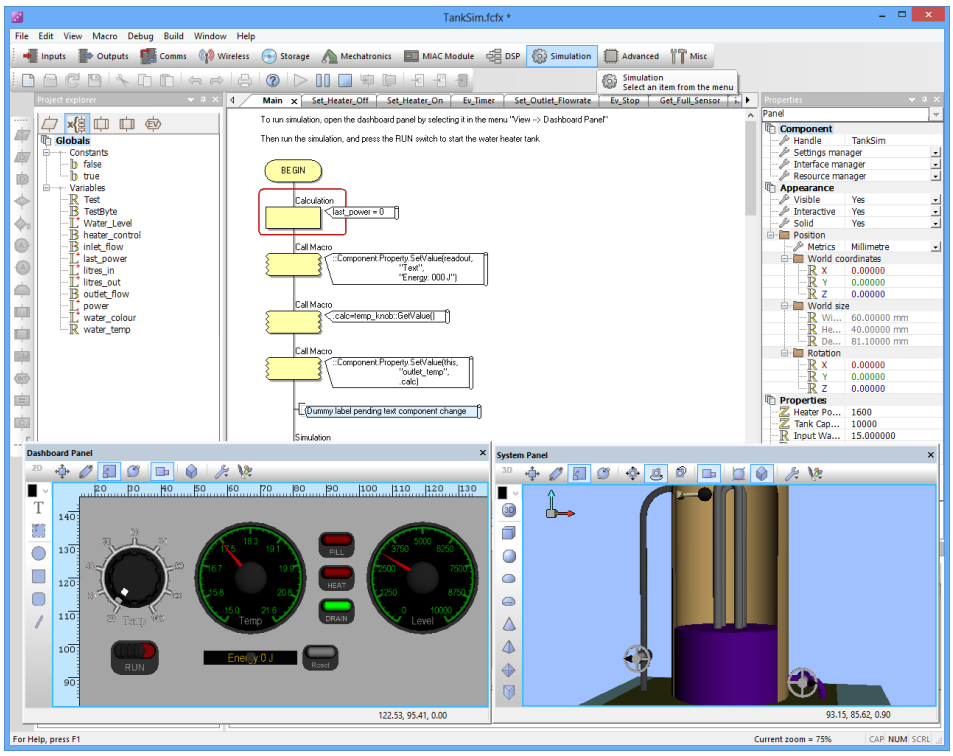


Рис. 1.20. Один из примеров использования панели управления и анимации

И, наконец, для тех, кто сомневается, стоит ли тратить время на знакомство с программой, есть пример перевода числа с плавающей точкой в символьный вид для вывода на дисплей.

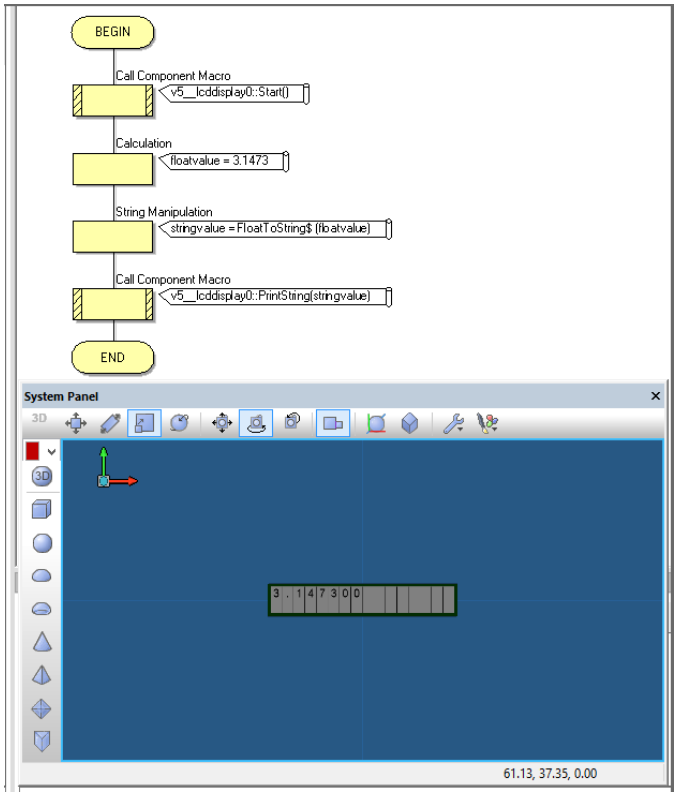


Рис. 1.21. Перевод десятичного числа в символьный вид для вывода на дисплей

Глава 2. Основное меню (File, Edit)

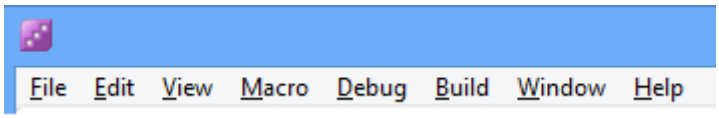


Рис. 2.1. Основное меню программы Flowcode 6

Если в вашем распоряжении окажется русифицированная версия, то вид основного меню может немного измениться, но сохранит, надо полагать, расположение всех пунктов; другое дело выпадающие списки каждого из пунктов, если они располагаются в алфавитном порядке.

Посмотрим, что в этих выпадающих списках.

Пункт File основного меню

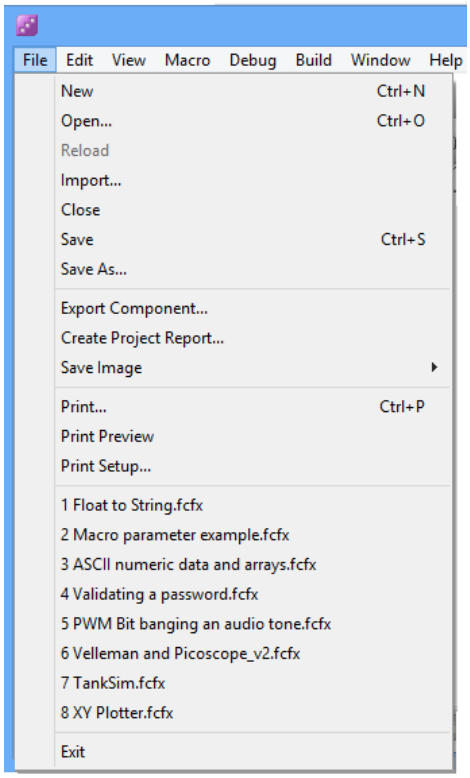


Рис. 2.2. Пункт основного меню File и список его операций

Как правило, согласно принятому стандарту, все операции с файлами во многом одинаковы для всех программ. Это операции создания нового файла (New), операции открывания (Open) и закрывания файла (Close), и две основные операции сохранения файла Save (просто сохранить) и Save as (сохранить под другим именем). Стандартными будут и операции печати Print, настройки печати Print Setup, просмотра перед печатью Print Preview. Но, как правило, специализированные программы всегда имеют дополнительные команды (или разделы) в традиционных пунктах основного меню.

Первый из таких разделов Import позволяет импортировать файлы проектов. Нажав это раздел пункта File основного меню, можно посмотреть, какие файлы можно импортировать в проект:

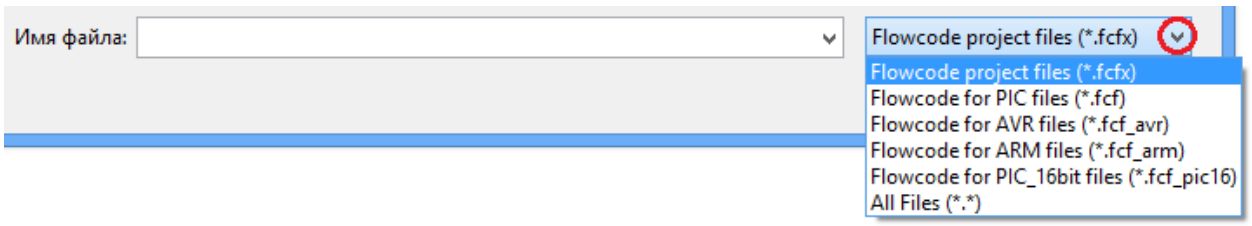


Рис. 2.3. Файлы, импортируемые в новый проект

Список импортируемых файлов вы получите, если воспользуетесь отмеченной на рисунке кнопкой. Одной из особенностей всех версий программы Flowcode было то, что программа, написанная для микроконтроллера AVR, может импортироваться в проект для микроконтроллера PIC. Если возможности микроконтроллеров совпадают, то правка может либо не понадобиться, либо быть минимальной.

Раздел Export Component (экспортировать компонент) появился в этой версии программы; он вызывает появление менеджера компонентов.

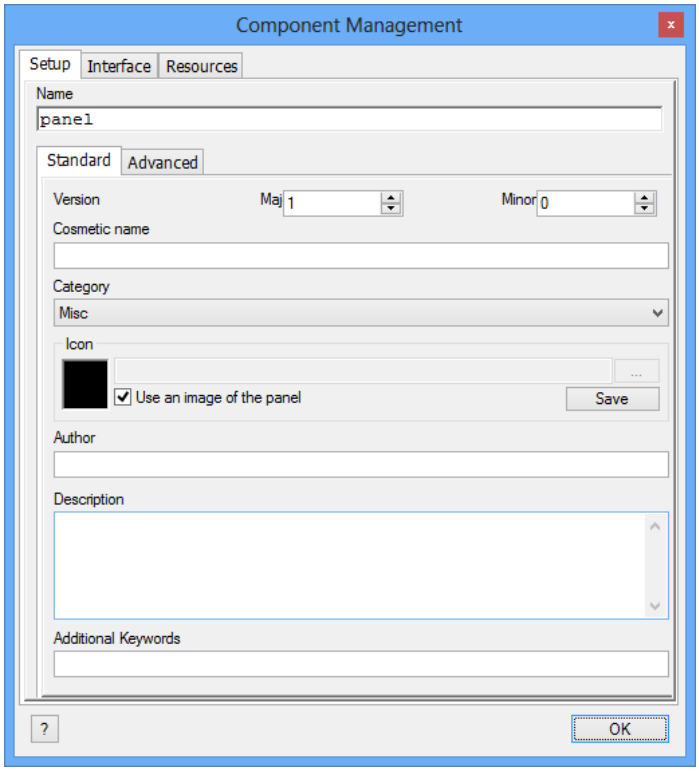


Рис. 2.4. Использование раздела экспорта пункта File

Раздел создания отчёта о проекте (Create Project Report) может быть полезен не только при профессиональном использовании программы, но и в любительских разработках. Далеко не все любители готовы тратить время на создание описания своего проекта хотя бы краткого. Используя этот раздел меню, вы получите файл в формате HTML, который можно открыть своим web-проводником, чтобы увидеть готовое описание. Есть несколько вариантов отчёта, по умолчанию это default, которые можно выбрать из списка с помощью кнопки справа от окна, отображающего выбранный формат отчёта: графический вид программы, исходный код или формат по умолчанию.

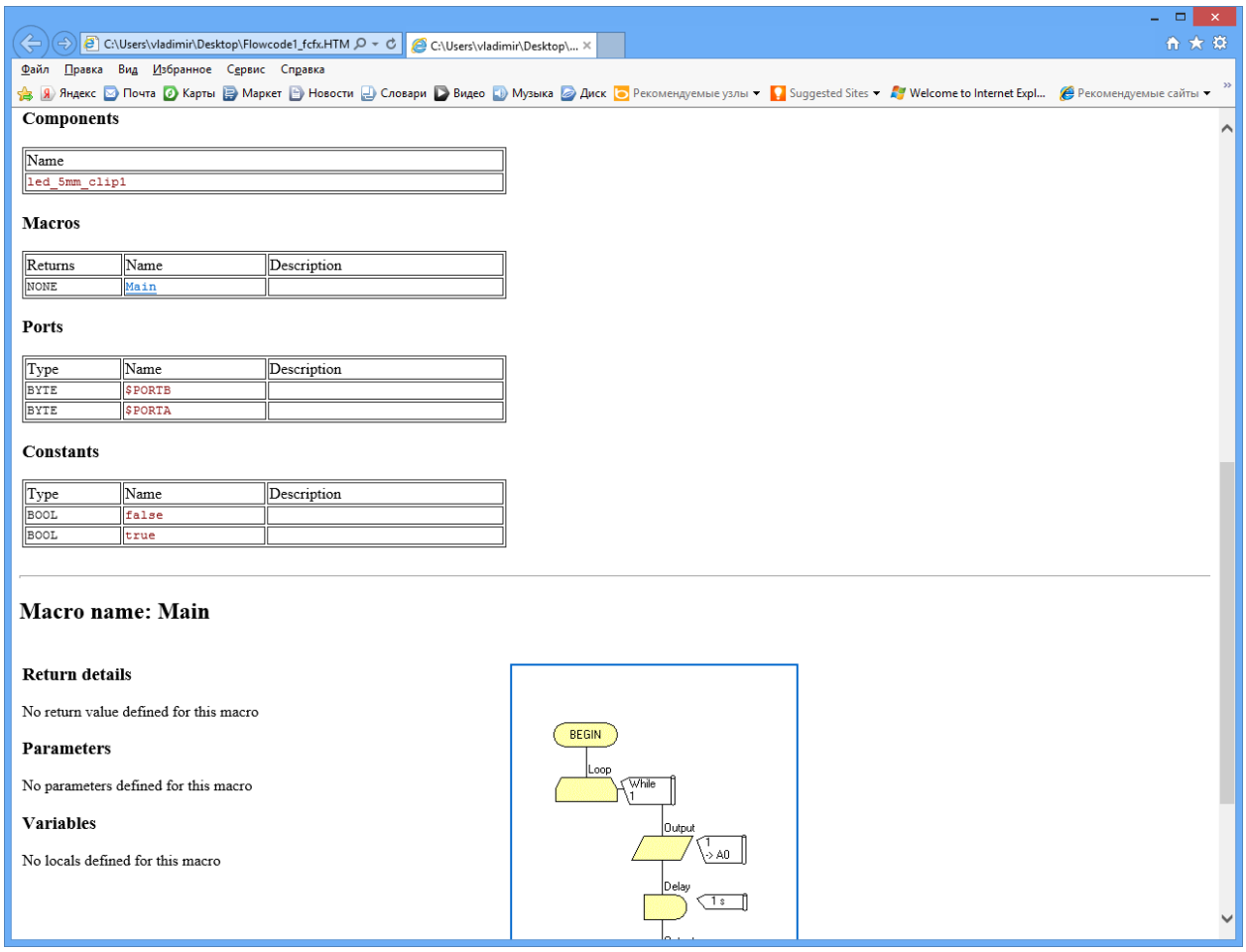


Рис. 2.5. Сохранённый с помощью раздела создания отчёта файл в IE

Раздел сохранения картинки (Save Image) даёт вам выбор, что следует сохранить:

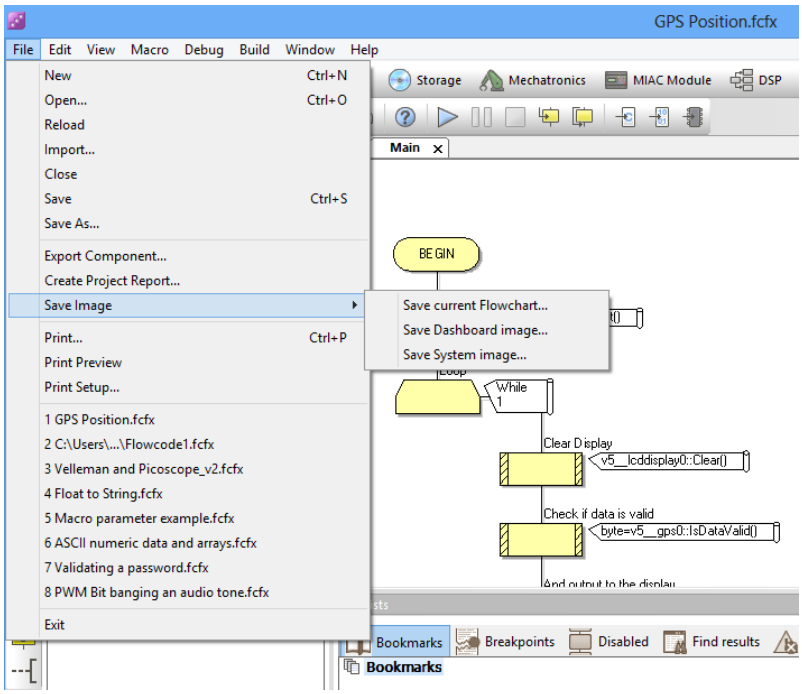


Рис. 2.6. Сохранение картинки

И это полезно даже любителям – сохранить рисунок быстро, а иметь его под рукой в бумажном виде или в виде картинки на компьютере, всегда можно обменяться мнением с друзьями, внести поправки, составляя план дальнейших действий. А в профессиональной работе он пригодится для расширенного отчёта, при написании статьи или для хранения в бумажном виде, как документ сопровождения проекта.

В следующих разделах пункта File вы найдёте список недавно открывавшихся файлов и последнюю команду Exit, закрывающую программу.

Пункт Edit основного меню

Пункт основного меню Edit (редактирование) есть в любом редакторе, текстовом, графическом или специализированном, как в среде разработки, подобной Flowcode.

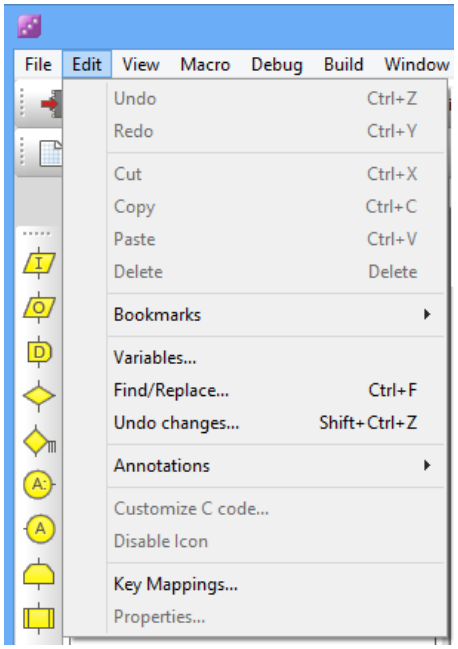


Рис. 2.7. Пункт редактирования основного меню

Все разделы любого пункта меню, как это принято в современных приложениях, контекстно-чувствительны, то есть, активным раздел становится тогда, когда команду можно применить. Команда Undo (отменить) пока бледно серого цвета станет ярче тогда, когда будет что-то отменять. Например, если удалить фрагмент программы, то активизируется эта команда в пункте Edit:

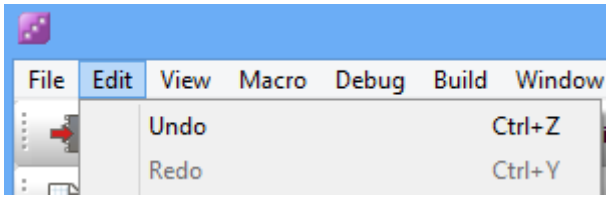


Рис. 2.8. Активизация команды при изменениях в проекте

Команда Redo (вернуть) имеет прямо противоположный смысл – она возвращает предыдущую команду, а не отменяет ошибочно сделанное действие, как команда Undo.

Следующий набор команд: Cut (вырезать), Copy (копировать), Paste (вставить), Delete (удалить), - все они входят в набор команд любого редактора. Если команда Cut вырезает элемент рабочего поля в буфер обмена, то команда Delete удаляет его. Первая команда удобна тогда, когда вам нужно переместить элемент из одного места в другое, а отменить вторую команду можно с помощью команды Undo.

Следующий раздел Bookmark относится к закладкам.

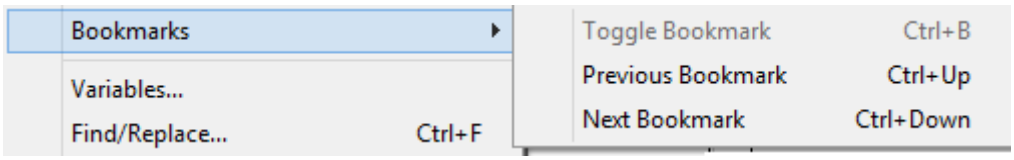


Рис. 2.9. Раздел Bookmarks

Вот, что можно прочитать о закладках в разделе помощи.

Использование закладок

Закладки, в основном, используются для ускорения навигации по создаваемой программе. Вы можете перепрыгивать по заложенным иконкам щелчком по ним в списке; это можно также сделать щелчком правой клавиши мышки по ним или щелчком по стрелке Next и затем по Show (показать).

Иконки с закладками можно также найти или перемещаться по ним непосредственно, используя саму программу, вы можете щёлкнуть правой клавишей мышки по иконке в программе и выбрать Next Bookmark или Previous Bookmark (Bookmarks > Next/Previous Bookmark).

Добавление и удаление закладок

Закладки могут добавляться или просто переключаться правым щелчком мышки по иконке, их можно удалить простым повторным переключением (Bookmarks > Toggle Bookmark).

Можно удалить закладку из списка правым щелчком мышки или нажатием на стрелку рядом с именем иконки и выбором команды Remove.

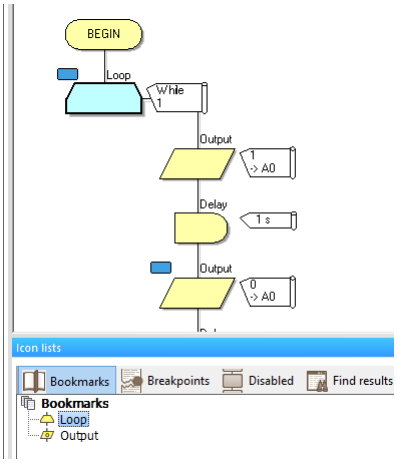


Рис. 2.10. Закладки в «тексте» программы

Следующий раздел (Variables) открывает менеджер переменных. Как в окне навигации по проекту, вы можете увидеть глобальные переменные и локальные переменные, с которыми можно работать. Если есть лишняя переменная, то её следует удалить из программы, а затем можно удалить из списка переменных, щёлкнув по ней правой клавишей мышки и выбрав команду из выпадающего списка.

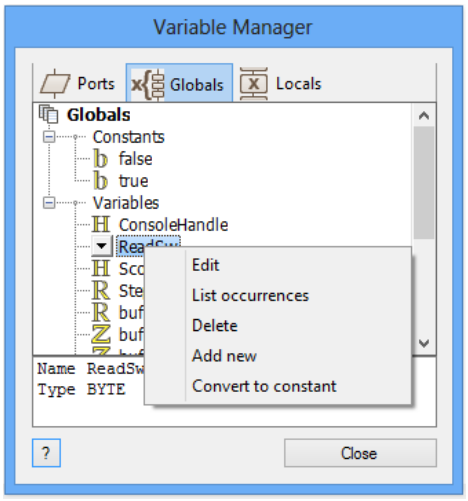


Рис. 2.11. Менеджер переменных

Если переменная используется в программе, то вы получите уведомление об этом:

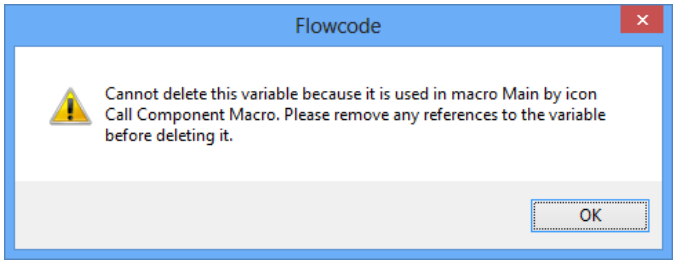


Рис. 2.12. Переменная используется в программе, и её нельзя удалить

Следующая команда пункта Edit – это команда поиска и замены.

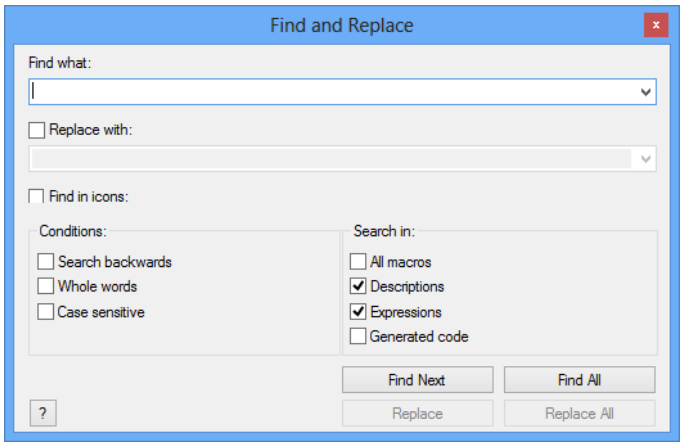


Рис. 2.13. Диалоговое окно поиска и замены

Первое окно для ввода того, что следует искать, ниже окно для ввода того, чем следует заменить найденное. Опция поиска в иконках будет работать, если вы её отметите, а ниже условия поиска:

искать в обратном направлении, искать слово целиком и ориентироваться на регистр. Справа предложено выбрать место поиска: все макросы, описания, выражения, сгенерированный код.

Используя кнопки **Find Next** и **Find All**, вы можете найти следующее слово или найти все искомые слова. Также вы можете заменить отдельное слово кнопкой **Replace**, а можете заменить все с помощью кнопки **Replace All**.

Следующая команда Undo Changes (отменить изменения) использует буфер истории операций. Если операций не было, буфер будет пуст.

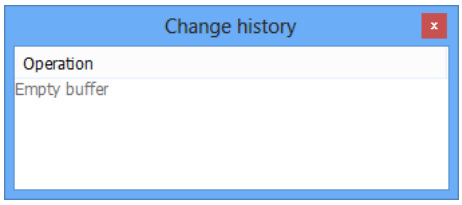


Рис. 2.14. Отмена сделанных операций

Следующая операция относится к примечаниям.

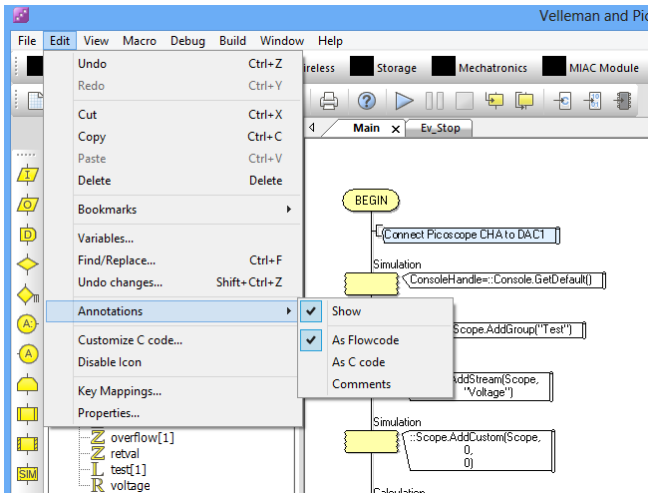


Рис. 2.15. Раздел Annotations

Выделив примечание, его можно увидеть, как оно есть в программе, или увидеть в виде примечания в Си коде.

Выделив компонент программы, вы можете использовать следующие разделы пункта Edit, которые активируются при выделении.

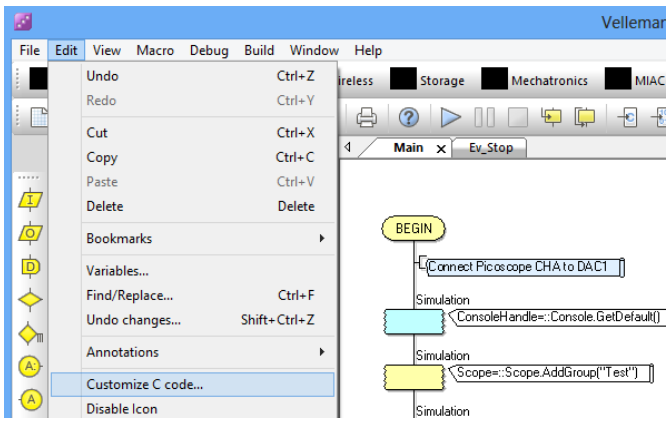


Рис. 2.16. Разделы обработки кода Си и удаления иконки из обработки

Если выбрать команду Disable Icon, то меняется внешний вид иконки, и она появляется в списке удалённых из обработки программных элементов.

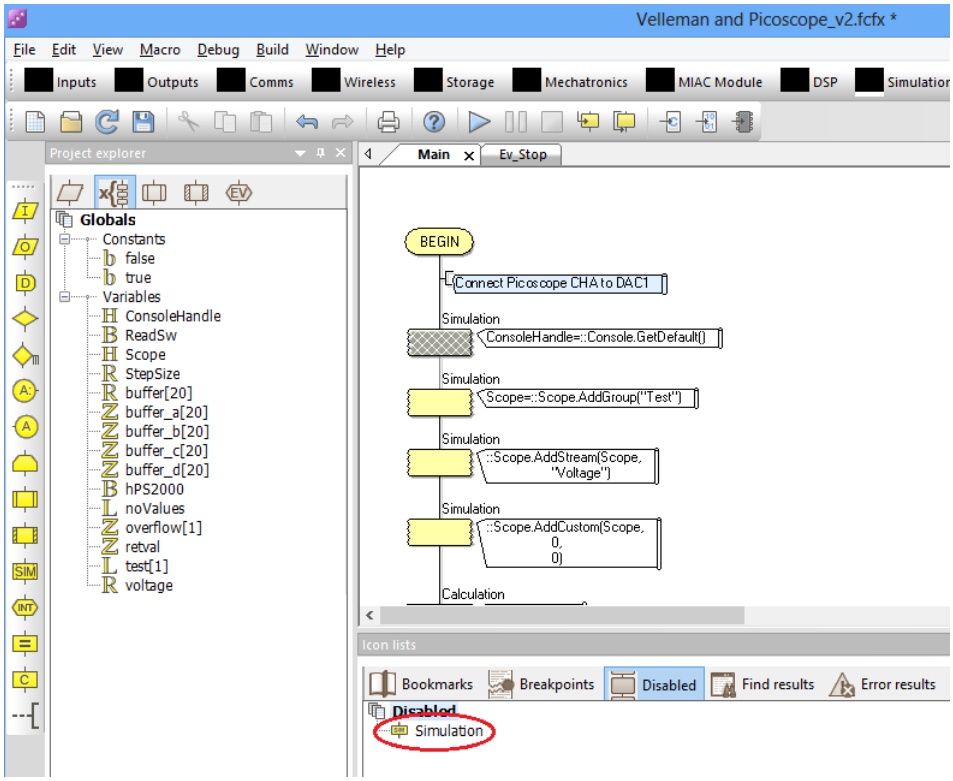


Рис. 2.17. Использование команды Disable

Вывод компонентов программы из обработки может быть полезен при отладке программы, чтобы оценить его влияние на работу программы или в тех случаях, когда появляются сомнения в правильности применения этого элемента программы.

Отладка один из самых важных этапов при разработке программ, поэтому любая среда разработки имеет отладочные средства. При работе над программой, если появляются сомнения, часть программы, написанной на языке Си, блокируется с помощью операции выделения комментария. В данном случае используется команда Disable Icon.

Чтобы вернуть часть программы на место, можно воспользоваться списком раздела Disabled в нижней части рабочего окна. Щёлкните правой клавишей мышки по нужному фрагменту программы и из выпадающего списка выберите команду Enable.

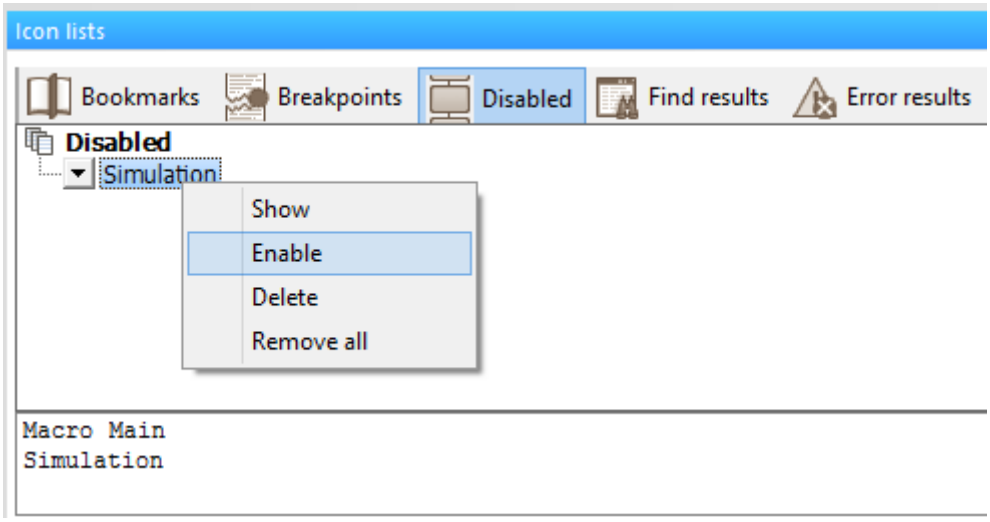


Рис. 2.18. Возвращение фрагмента программы в работу

За этими разделами следуют операции привязки клавиш (Key Mappings), если есть клавиатура, и вызов диалога свойств компонента (Properties). Обе операции станут активны при выделении элемента в программе.

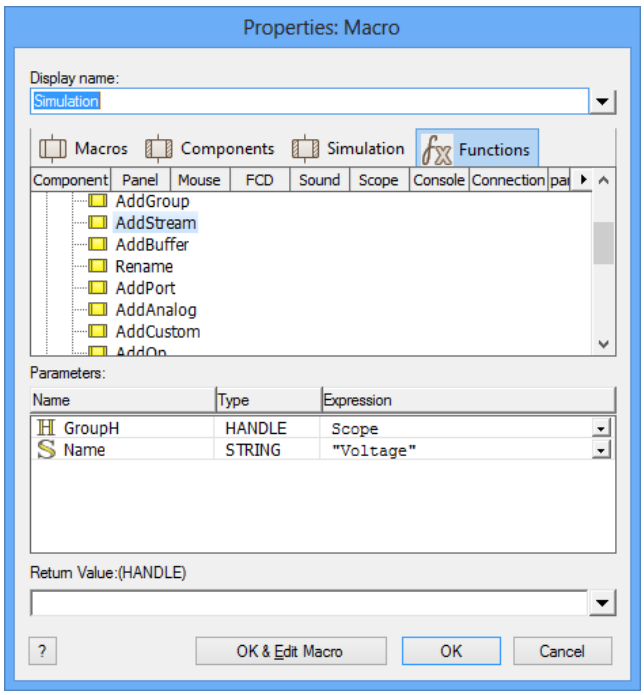


Рис. 2.19. Диалоговое окно свойств макроса в программе

Как и многие диалоги, этот диалог имеет своё собственное меню и может иметь ряд закладок. Выделим, например, в основном меню пункт Components.

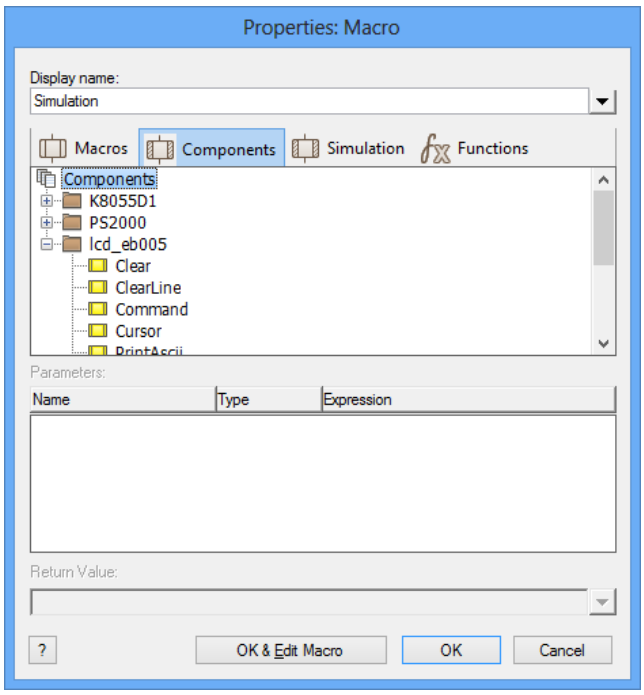


Рис. 2.20. Диалог свойств внешних компонентов

В окне есть список всех дополнительных компонентов, а открыв список (значок «+»), можно увидеть все свойства компонента. Особенно много свойств можно узнать в пункте Functions, где есть ряд закладок:

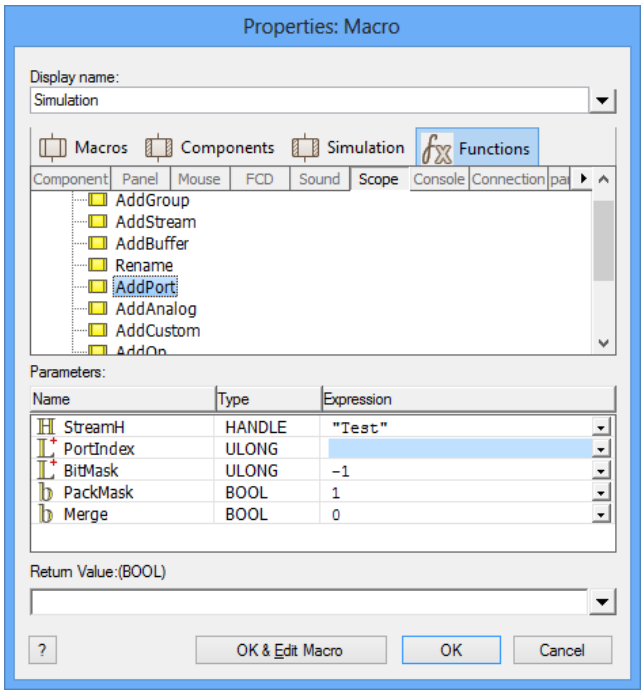


Рис. 2.21. Диалог свойств в пункте Functions

Глава 3. Основное меню (View, Macro)

Пункт View

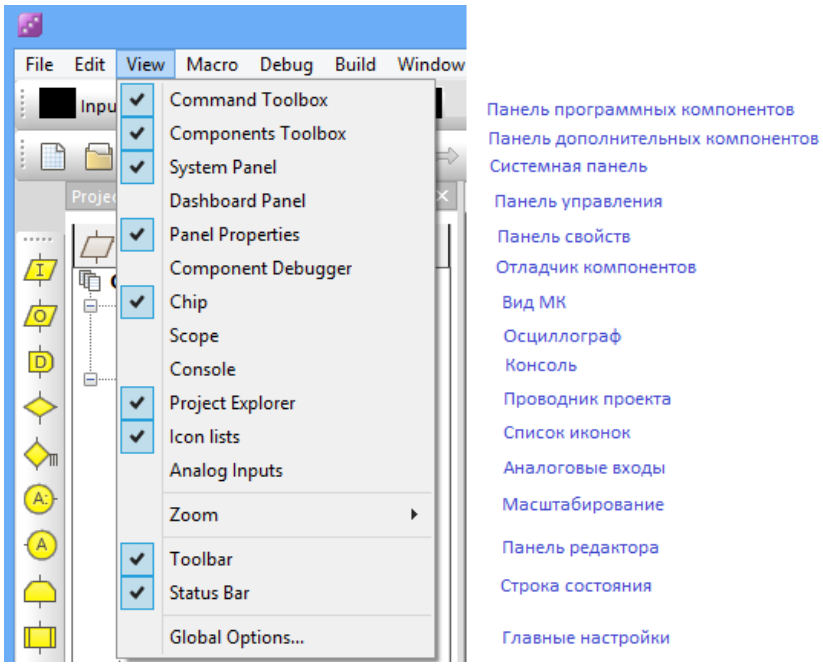


Рис. 3.1. Подменю пункта вида

В этом пункте основного меню можно включить или выключить инструментальные панели, включить окно свойств компонентов и т.д. Чтобы увидеть, например, панель управления (Dashboard Panel) достаточно щёлкнуть по её названию в списке. Панель появится на экране, а рядом с этим разделом появится галочка, как сейчас она есть рядом с теми окнами, которые открыты в программе.

В этом списке панелей есть разделы, требующие пояснения, но мы рассмотрим только главные настройки редактора, а к остальным обратимся позже.

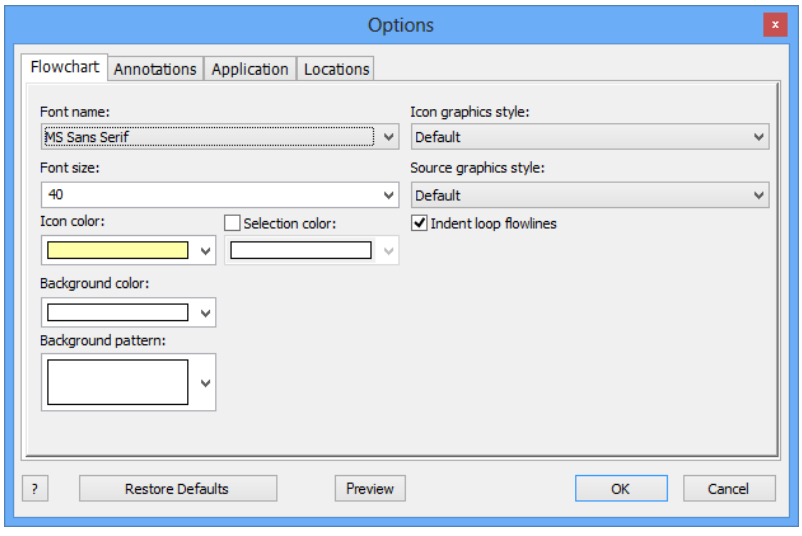


Рис. 3.2. Диалог главных настроек редактора

Кроме выбора шрифта и его размера, цвета иконок и фона, вы можете изменить графический стиль иконок:

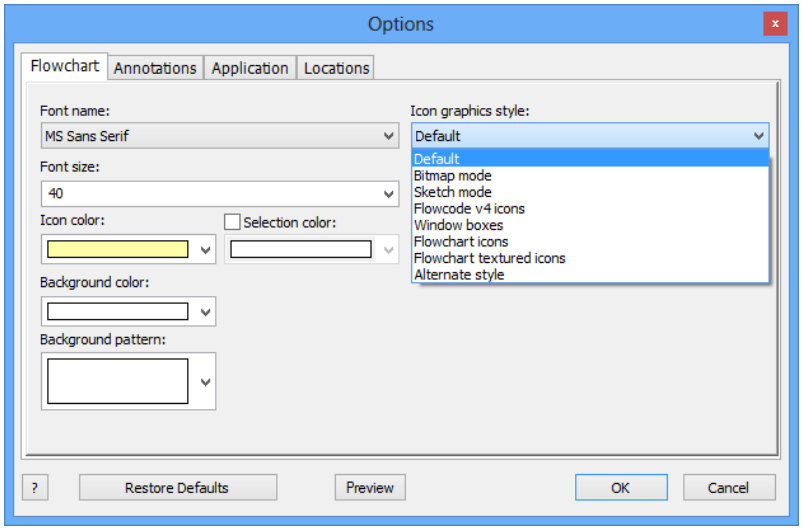


Рис. 3.3. Список возможных стилей оформления иконок

Так стиль Sketch mode выглядит иначе, чем привычный уже стиль.

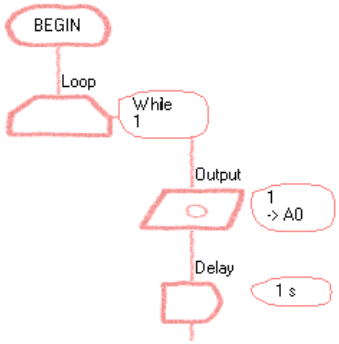


Рис. 3.4. Изменение стиля оформления программы

Предварительный просмотр осуществляется с помощью кнопки **Preview**, а вернуться к исходному виду программы позволит кнопка **Restore Defaults**. Вы можете прервать свои настройки, воспользовавшись кнопкой **Cancel**, а можете принять новые настройки, если нажмёте кнопку **OK**.

Аналогично можно изменить стиль интерфейса и на двух других закладках, первая относится к отображению аннотаций (описание иконок, включение текста внутри иконок, отключение примечаний, ограничение размера примечаний и т.п.):

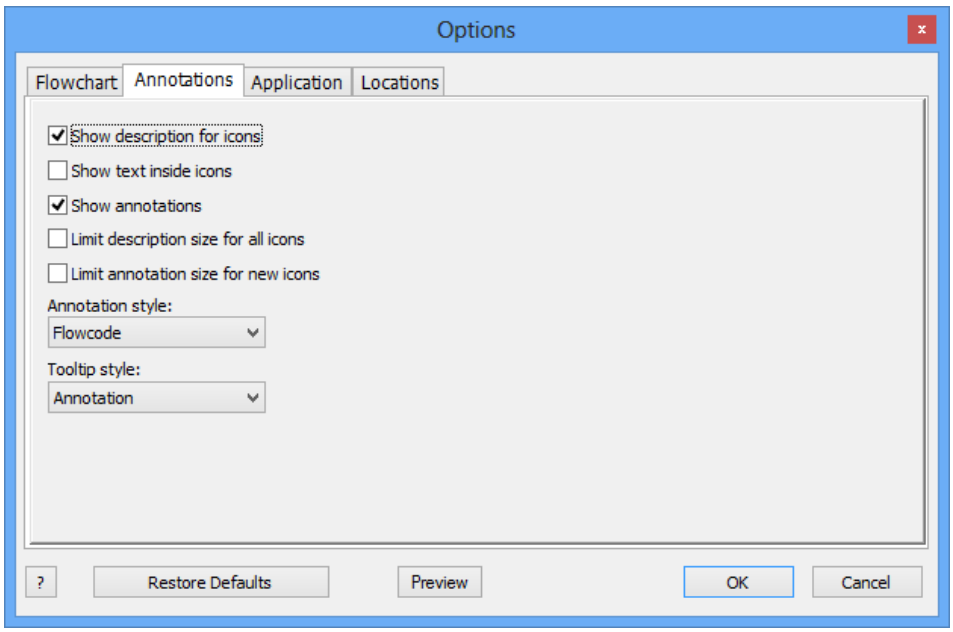


Рис. 3.5. Закладка изменения стиля примечаний

На следующей закладке можно внести настройки в стиль приложения:

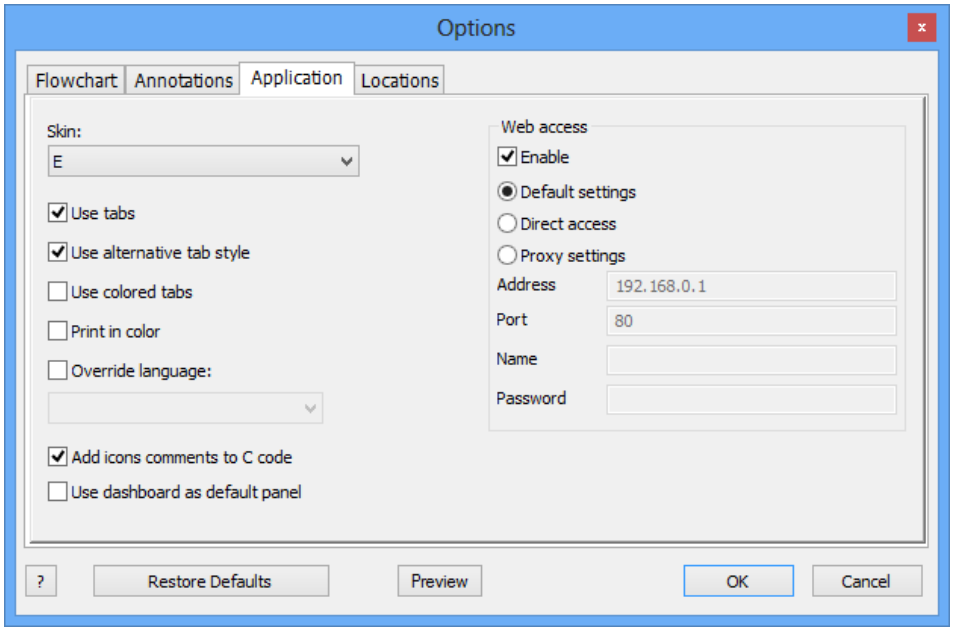


Рис. 3.6. Закладка изменения стиля приложения

Здесь же вы можете настроить подключение приложения к Интернету или отключить доступ, убрав галочку Enable.

Последняя закладка позволяет перенаправить выход компилятора:

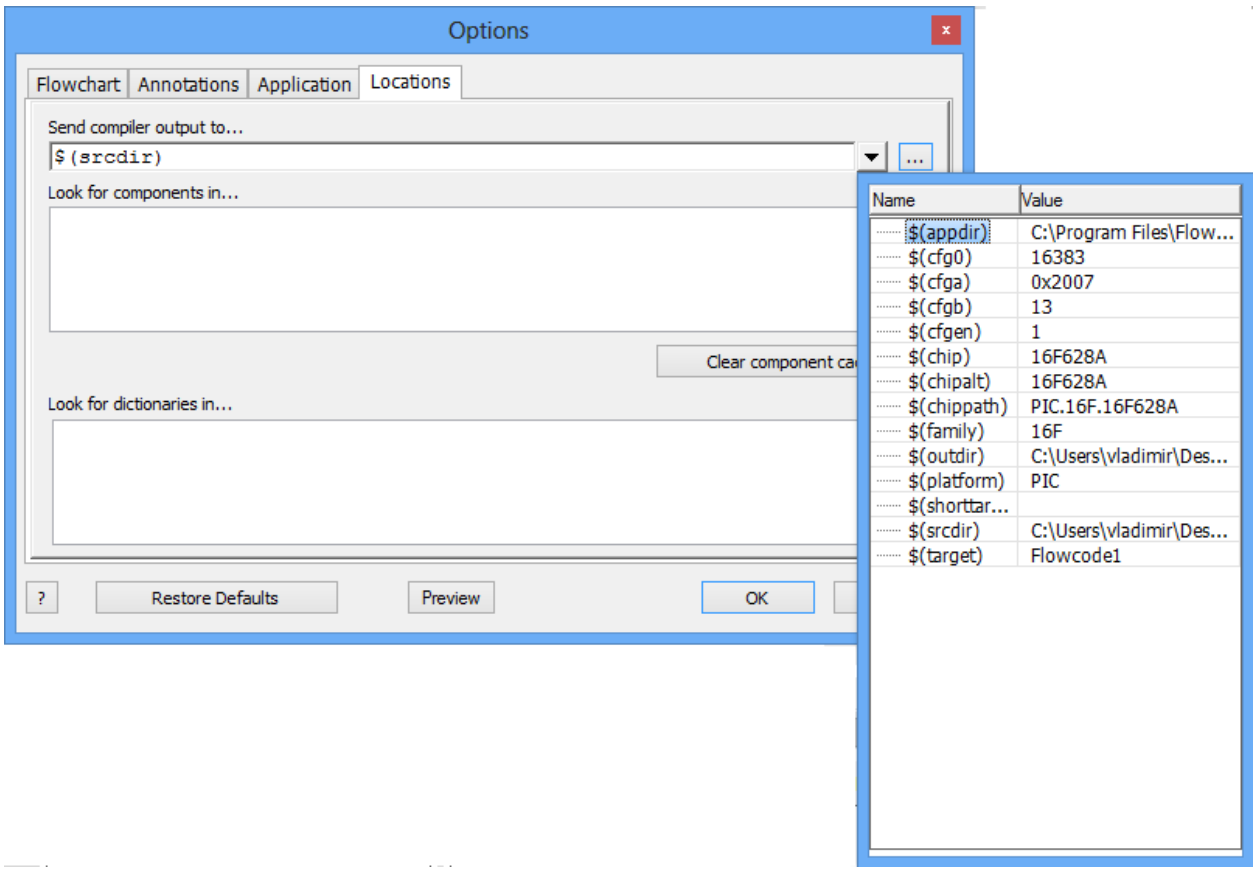


Рис. 3.7. Закладка Locations

Пункт основного меню Macro

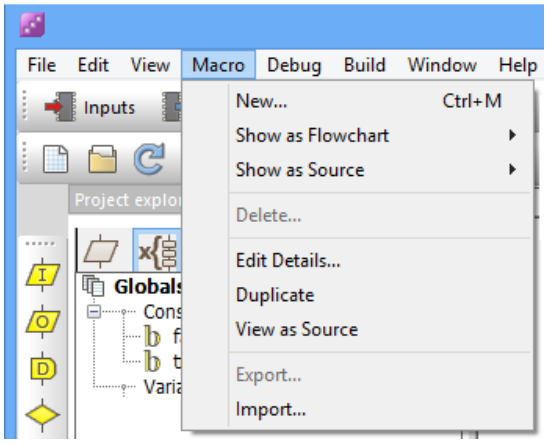


Рис. 3.8. Содержание пункта Macro

Использование подпрограмм, функций и макросов, обычная и полезная практика при создании даже небольших программ, но обязательная с ростом размера программы. Если вы не хотите запутаться в длинной последовательности операций, то их можно разбить на блоки, оформив каждый как макрос. Чтобы создать новый макрос, используйте раздел New пункта Macro основного меню.

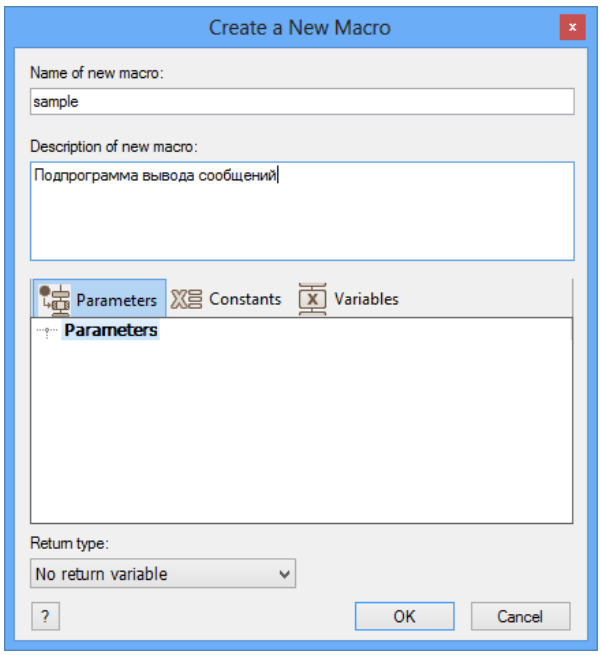


Рис. 3.9. Создание нового макроса

Каждая функция (процедура, подпрограмма, макрос) должна иметь имя, которое вы впишите в окно Name of new macro. Полезно будет и описать этот макрос в следующем окне. Позже, когда у вас наберётся много блоков программы, вы сможете тем лучше в них разобраться, чем лучше опишите каждую из составляющих.

Функция может иметь параметры, константы и переменные, которые появятся в следующем окне, если они есть. Кроме того, макрос может возвращать переменную, которую вы определите (если функция возвращает что-то) в окне Return type. По умолчанию макрос не возвращает переменной.

Следующие два раздела подменю позволяет вам посмотреть макросы, если они есть, в виде графическом (Show as Flowchart) или в виде исходного текста (Show as Source). При этом основная программа рассматривается в этом случае тоже как макрос.

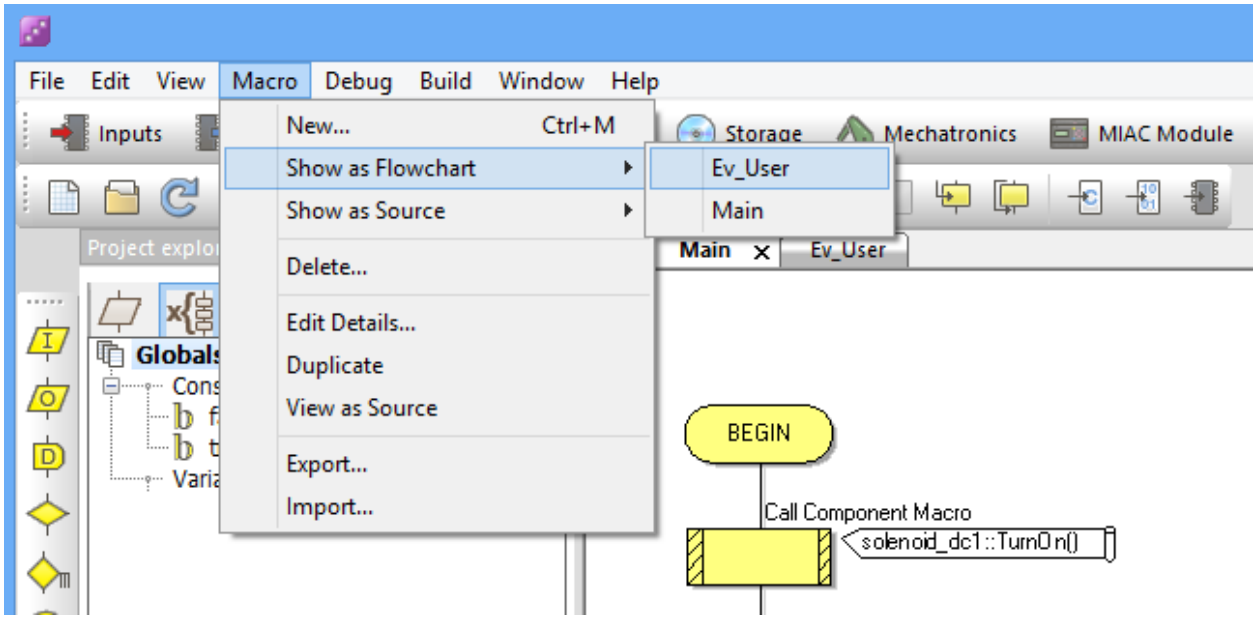


Рис. 3.10. Разделы просмотра макросов

Впрочем, просмотреть макросы можно и используя закладки в рабочем поле программы. А просмотр макроса в виде исходного текста, видимо, возможен после трансляции программы.

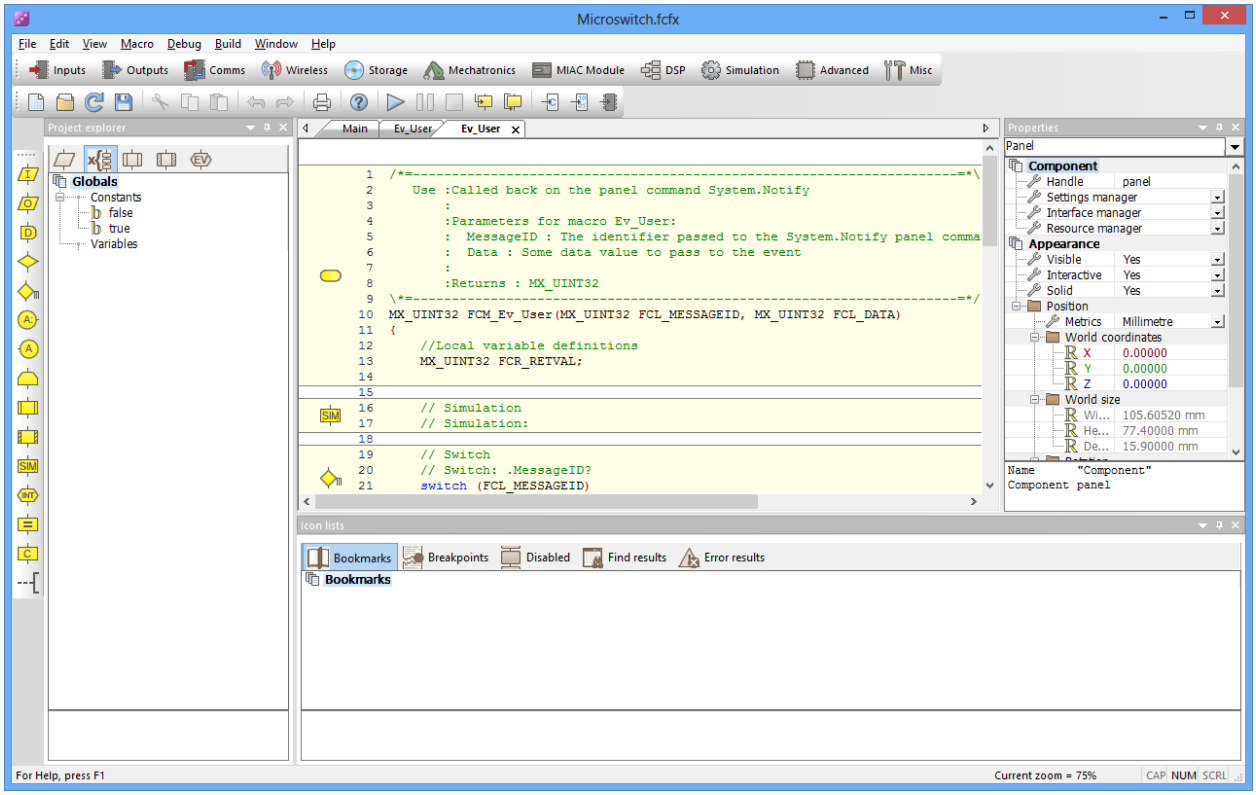


Рис. 3.11. Просмотр макроса в виде исходного текста на языке Си

Любой из созданных вами макросов вы можете удалить, используя раздел Delete, если макрос больше не нужен, и можете поправить его настройки, используя раздел Edit Details пункта Macro основного меню:

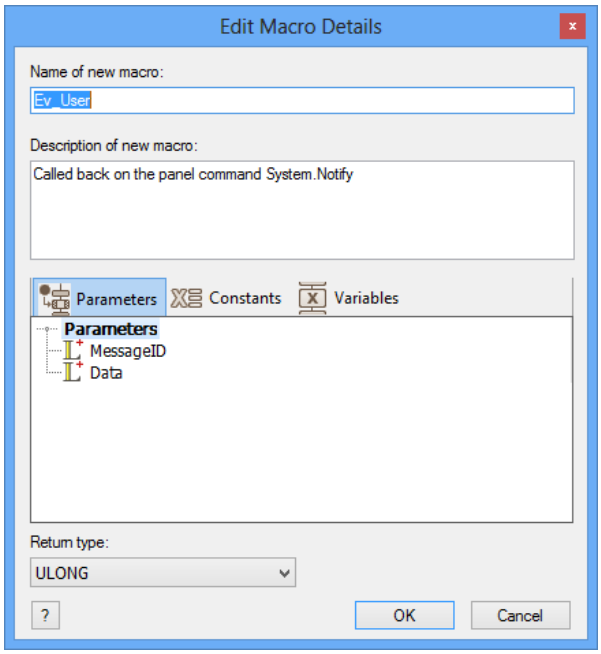


Рис. 3.12. Настройки макроса

Вы можете продублировать макрос (Dublicate) или включить его вид на языке Си, что бывает полезно при отладке программы (View as Source). Давайте посмотрим, чем полезно дублирование макроса. Положим, вы создали новый макрос с именем m1. Он появляется как заготовка, где вы дополняете его нужными операциями.

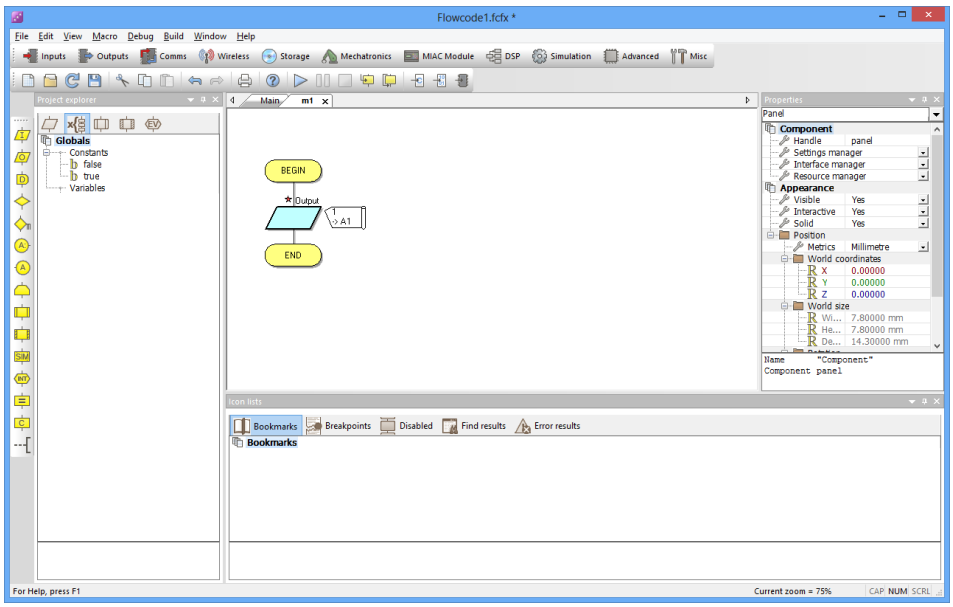


Рис. 3.13. Пример создания нового макроса

В программе вам понадобилось создать похожий, но несколько иной макрос. Можно, конечно, создать его как новый, если это простая часть программы, но в более сложном случае…

После использования раздела Dublicate появится диалоговое окно создания нового макроса, где вы дадите ему имя, а, нажав кнопку **ОК**, получите точную копию макроса.

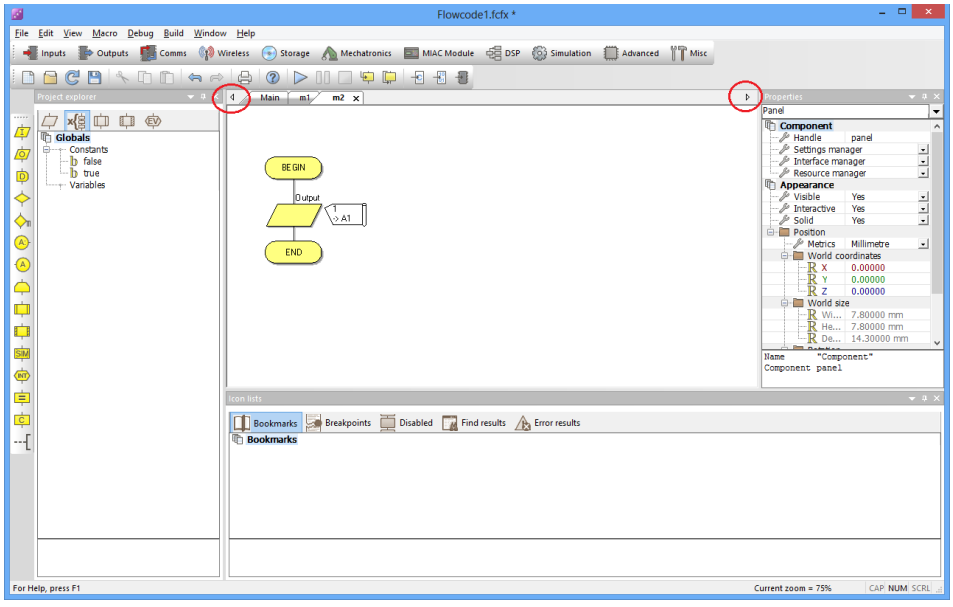


Рис. 3.14. Пример дублирования макроса

Внеся необходимые правки, что сделать проще, чем собирать новый фрагмент программы, вы получаете необходимое. Каждый из макросов появится в рабочем поле со своей закладкой. В большой программе этих закладок набирается много. Перемещаться по ним можно, например,

используя отмеченные на рисунке стрелки. Но, если они вам мешают, вы можете закрыть ненужные блоки.

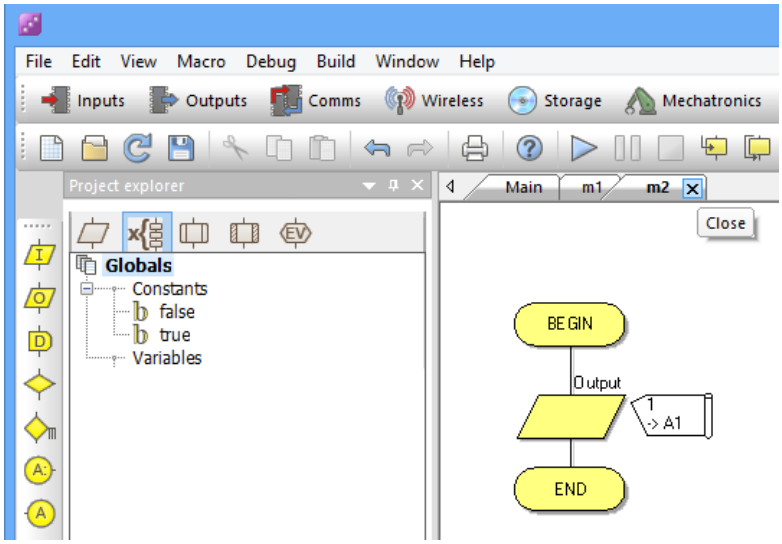


Рис. 3.15. Кнопка, закрывающая ненужные в настоящее время фрагменты программы

Позже, если появится необходимость вернуться к макросу, вам и потребуется предыдущий раздел пункта Macro:

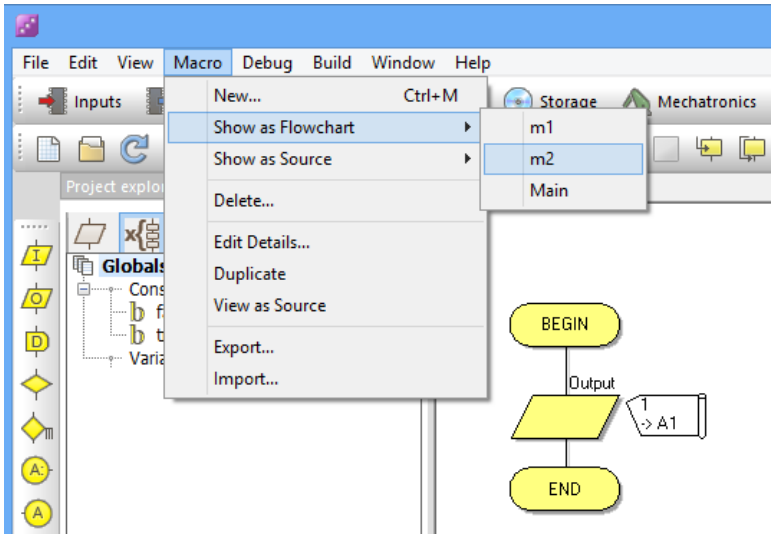


Рис. 3.16. Возвращение макроса в рабочее поле

Кроме того, работая над программой, стараясь не упустить удачное решение или пытаясь быстрее реализовать план программы, имена подпрограмм даёшь не всегда удачно, как это сделано выше. Подпрограмма с именем m1 или m2 хороша в очень короткой программе, но в большой программе лучше исправить имя, используя пункт Edit Details. В появившемся диалоговом окне достаточно стереть прежнее имя и вписать новое. После того, как вы нажмёте кнопку **ОК**, имя макроса изменится.

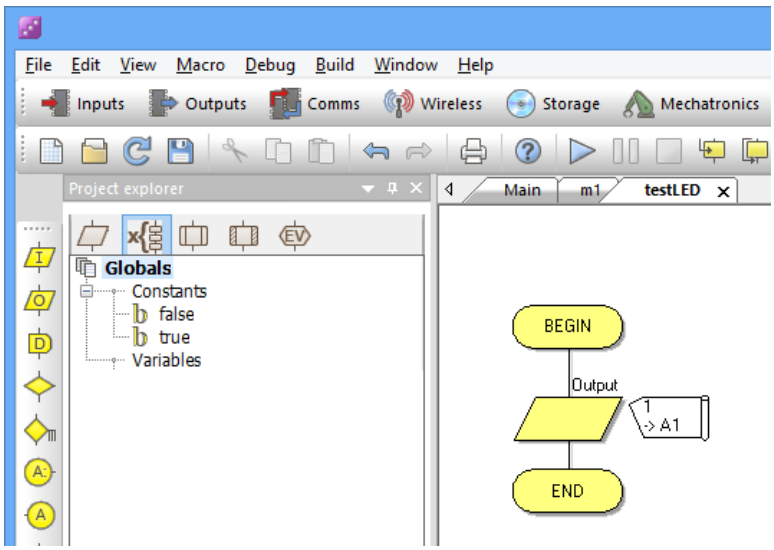


Рис. 3.17. Исправление имени макроса

Довольно часто, работая над программой, вы находите очень удачное решение. Оформляя это решение в виде подпрограммы, вы можете экспортировать макрос, чтобы при необходимости импортировать его в другую программу. Для этого служат последние две команды Export и Import. В диалоговом окне вы выбираете, какой из макросов следует экспортировать, нажимаете кнопку **Export** и указываете место, где будет храниться ваша процедура.

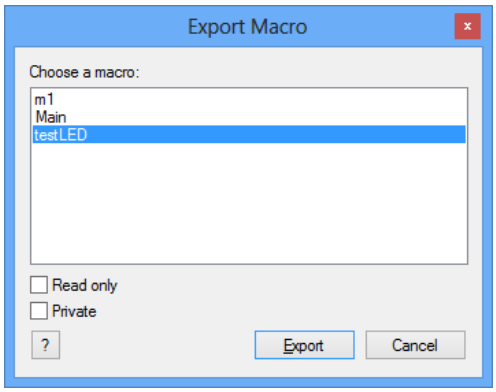


Рис. 3.18. Диалоговое окно экспорта макроса

Вы можете воспользоваться дополнительными опциями, Read only и Private. Сохранив макрос, вы, создавая новую программу, можете вставить эту нужную вам функцию, используя команду Import, где в диалоговом окне укажете место хранения макросов, и какой из файлов с расширением fcm, следует включить в программу. И, думаю, согласитесь, что имя m1 – это очень неудачный опыт с именем макроса.

Опытные программисты всегда выбирают некий единый стиль для обозначения переменных, функций и т.д. Любая современная программа даёт возможность использовать довольно длинные имена. Поэтому после первых опытов в программировании следует подумать о собственном стиле. Вы можете найти рекомендации в книгах о программировании, можете придумать что-то своё, но в дальнейшем постарайтесь придерживаться выбранного стиля. Это сэкономит вам много времени впоследствии, когда вы будете использовать свои наработки, а именно они и позволяют быстро и эффективно программировать.

Позже мы вернёмся к рассмотрению подпрограмм, когда будем рассматривать встроенные в Flowcode макросы. И вы сможете убедиться в пользе этой практике, избавляющей вас от необходимости описывать такой, например, довольно сложный протокол, как CAN. Конечно, ознакомиться с ним придётся в любом случае, но проще использовать его из библиотеки программы, чем создавать самостоятельно.

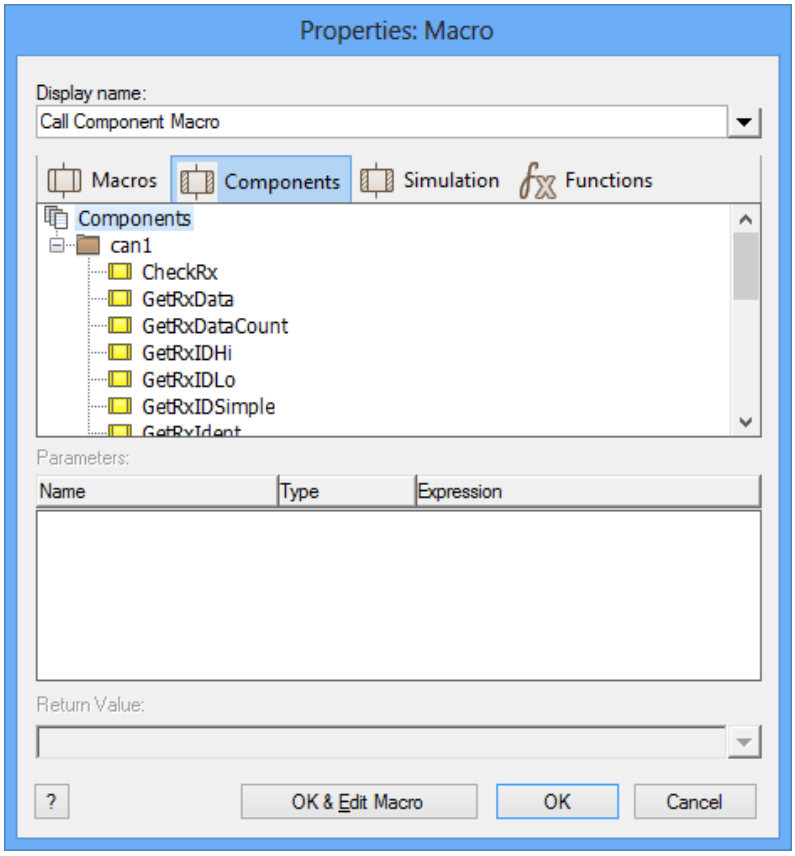


Рис. 3.19. Пример библиотечного макроса Flowcode

Глава 4. Основное меню (Debug, Build)

Пункт Debug

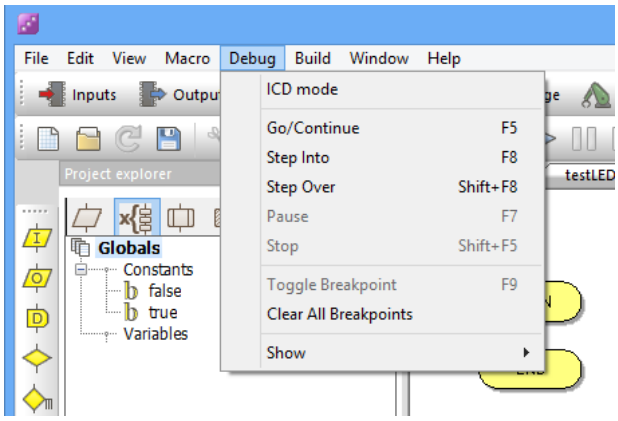


Рис. 4.1. Выпадающее меню пункта отладки

Отладка очень важный этап создания программы даже для опытных программистов, всегда можно опечататься в тексте программы, а проверять программу в работе с реальными устройствами, например, с космическими кораблями – это слишком дорогое удовольствие.

Для начинающих отладчик – лучший друг и советник. В зависимости от того, насколько удачно сделана отладочная часть среды разработки, зависит и успех программы. Особенность создания программ для микроконтроллеров в том, что этот компонент электрической схемы часто бывает сердцем устройства, но последнее не будет работать без остальных составляющих.

Разделы пункта Debug основного меню начинаются с ICD mode. При симуляции вы можете переключаться между симулятором и внутрисхемным отладчиком, используя этот раздел меню. Для использования ICD mode следует в свойствах проекта включить эту возможность.

Начать симуляцию программы можно полным её запуском, используя команду Go/Continue, но можно начать пошаговое прохождение программы командой Step Into.

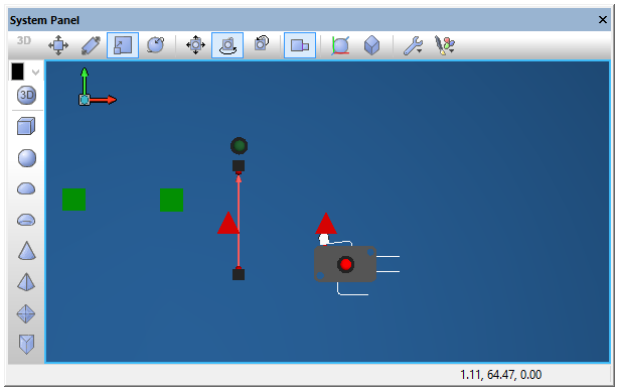


Рис. 4.2. Полный запуск программы командой Go

При таком запуске отладчика вы можете на системной панели, где размещены все дополнительные компоненты схемы, видеть, как работает устройство – индикаторные светодиоды зажигаются и гаснут, на дисплей выводится информация, заложенная в программе и

т.д. При пошаговом прохождении программы (скажем, командой Step Into) отладка выглядит несколько иначе.

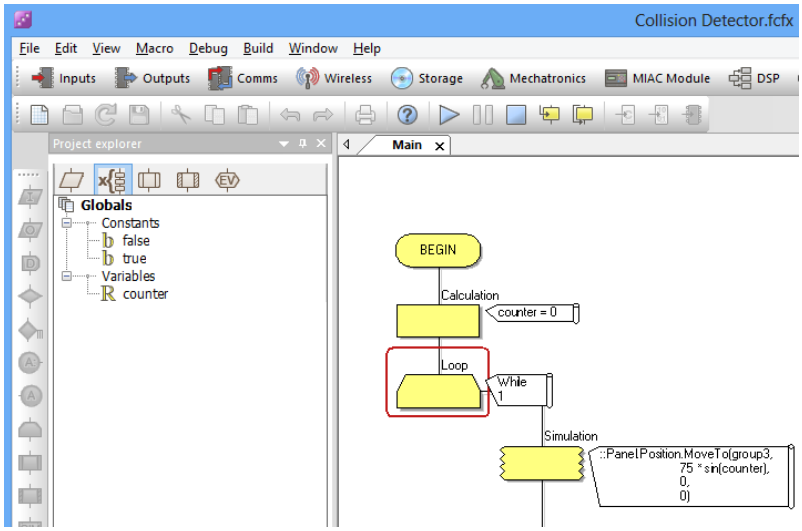


Рис. 4.3. Пошаговая отладка

При каждом шаге вы будете переходить от одного элемента программы к другому (отмеченному красной рамкой). Вы можете при этом нажимать кнопки, поворачивать регуляторы, смотреть, как меняются переменные, по каким ветвям программы проходит её выполнение. Порой, поскольку кнопки вы нажимаете мышкой, удобно использовать функциональные клавиши для пошагового перемещения. Горячие клавиши прописаны в меню рядом с командами.

Команда Step Over позволяет перешагнуть через некоторые элементы программы. А команды Pause и Stop не нуждаются ни в переводе, ни в пояснениях.

При пошаговом прохождении программы можно столкнуться с ситуацией, когда на пути встречается цикл, повторяющийся много раз. Повторять сотни шагов, чтобы завершить цикл, занятие утомительное. Проще сразу после цикла поставить точку остановки (Breakpoint), выделив нужный элемент программы. Сделать это можно командой Toggle Breakpoint. Рядом с выделенной иконкой программного элемента появится точка (это и есть точка останова).

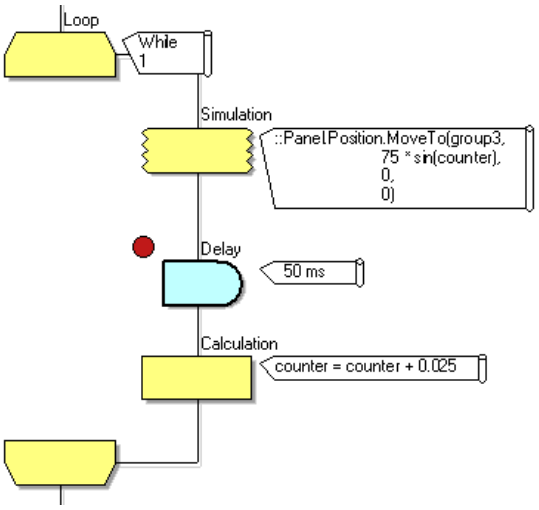


Рис. 4.4. Создание точки остановки

После запуска программа остановится в этом месте, вы можете пройти часть программы по шагам, а можете проверить то, что нужно и продолжить работу программы командой Go/Continue. Точек останова может быть много. Выключить ненужные вы можете, выделив элемент программы с точкой останова, той же командой Toggle Breakpoint (переключение точки остановки). И можно выключить все точки остановки, если они вам больше не нужны, командой Clear All Breakpoints из пункта Debug основного меню.

Последняя команда включает окно наблюдения.

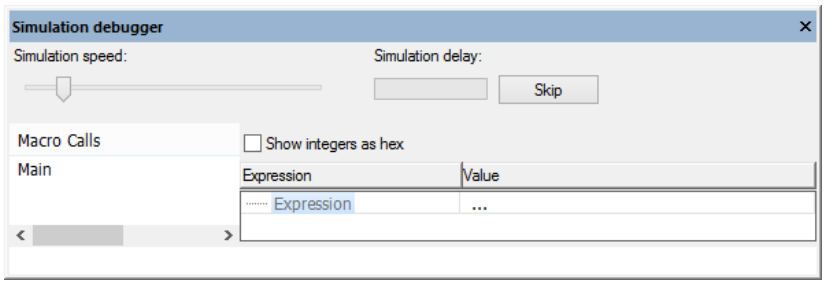


Рис. 4.5. Окно наблюдения

В окне наблюдения можно видеть, как меняются переменные при выполнении программы. Для этого следует добавить переменную в окно наблюдения. Для добавления переменной запустите выполнение программы, а в окне наблюдения щёлкните правой клавишей мышки там, где видите Expression и Value.

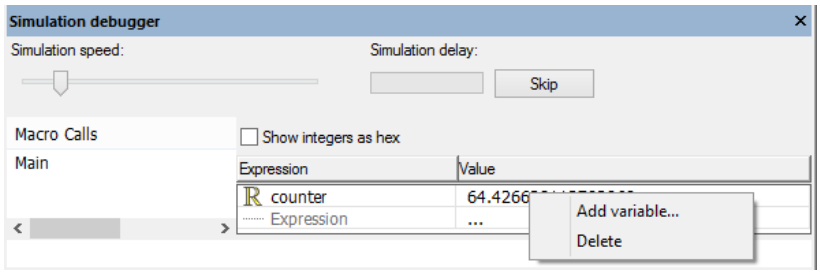


Рис. 4.6. Добавление переменной для наблюдения

Используйте команду добавления переменной (Add variable), или удалите ненужную переменную командой Delete. При добавлении переменной вы попадёте в окно со списком всех переменных, щёлкните по нужной вам дважды или выделите её и нажмите кнопку **ОК**.

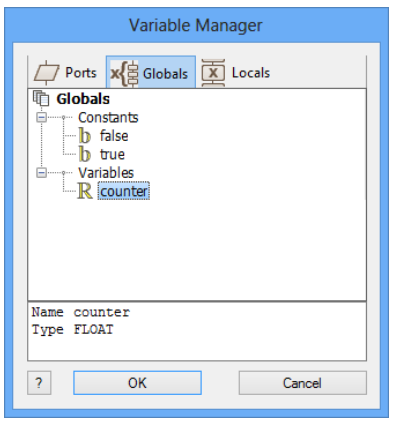


Рис. 4.7. Окно выбора переменной для добавления в окно наблюдения

Позже мы ещё вернёмся к отладчику, а сейчас рассмотрим следующий…

Пункт Build

Сборка программы и даже её отладка не являются целью работы в программе Flowcode. Конечной целью будет получение hex-файла для загрузки его в микроконтроллер с помощью программатора или загрузка программы в микросхему непосредственно из программы Flowcode.

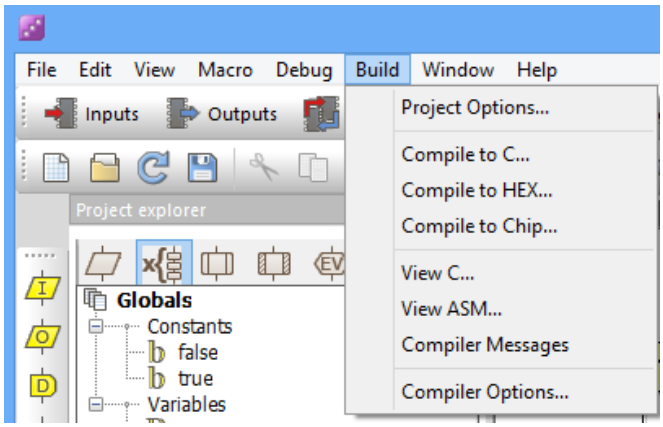


Рис. 4.8. Пункт создания работающей программы

Открывается этот пункт разделом настройки проекта, что справедливо, поскольку без настроек нет смысла транслировать программу.

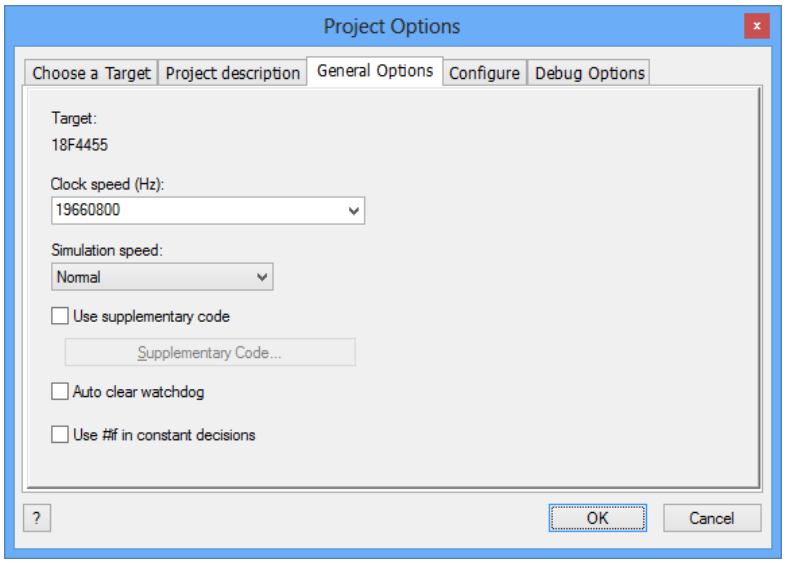


Рис. 4.9. Настройки проекта Project Options

Обязательно следует выбрать частоту тактового генератора (Clock speed). Всё, что в программе связано со временем, например, длительность пауз, будет определено по этой тактовой частоте. В качестве примера я настрою микроконтроллер для работы с внутренним тактовым генератором для микроконтроллера PIC16F628A.

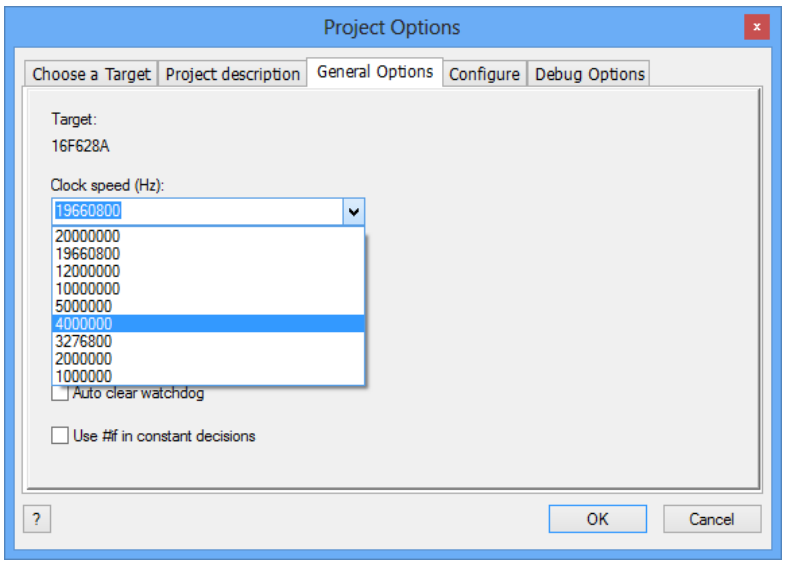


Рис. 4.10. Выбор тактовой частоты

Выбор самого микроконтроллера осуществляется на закладке Choose a Target.

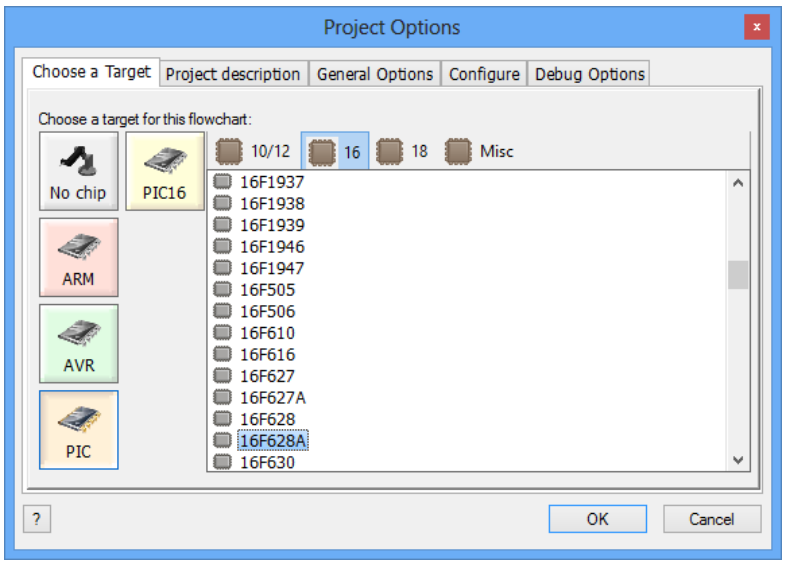


Рис. 4.11. Выбор модели МК

Для выбора режима работы с внутренним генератором следует перейти на закладку Configure. На этой же закладке выбираются все остальные настройки слова конфигурации. Слово конфигурации записывается при программировании микросхемы и не меняется в процессе работы программы. Такие элементы слова конфигурации, как защита кода (Code Protect), могут создавать неудобства при макетировании или перепрограммировании микроконтроллера. А выбор источника тактовых сигналов скажется, например, на скорости работы микроконтроллера при сетевом подключении. К заданию слова конфигурации следует относиться ответственно, ясно понимая, что воспоследует за включением или выключением каждого из элементов на закладке Configure.

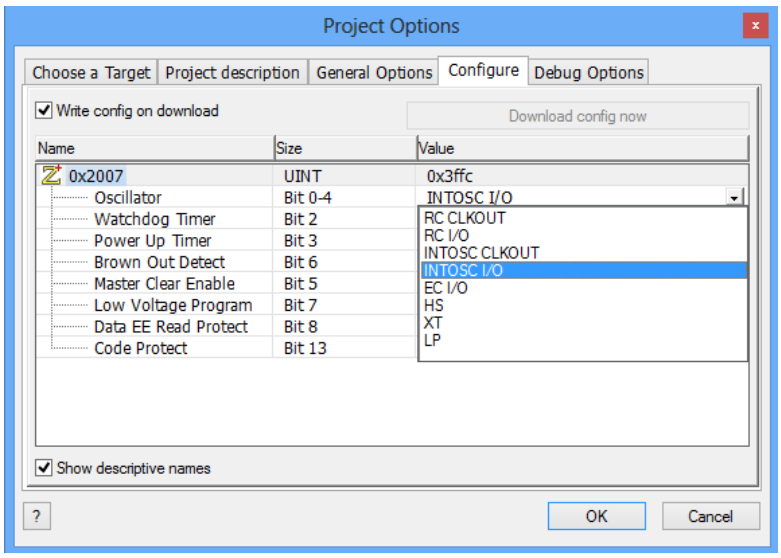


Рис. 4.12. Настройка слова конфигурации микроконтроллера

Закладка Project description (описание проекта) даёт возможность дать название проекту и описать его. А на закладке Debug Options, мы уже упоминали, можно сделать доступной внутрисхемную отладку.

Возвращаясь к основной странице настроек проекта, можно упомянуть, что в проекте используется добавочный код.

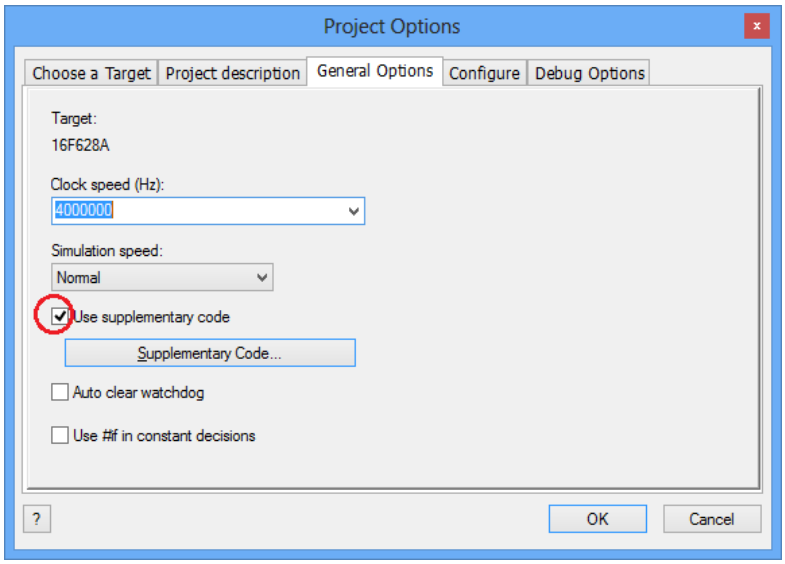


Рис. 4.13. Основные настройки проекта

Отметьте использование дополнительного кода, теперь, нажав на кнопку Supplementary Code, вы получите возможность ввести дополнительный код.

Основной язык высокого уровня программы Flowcode – это язык программирования Си. Поэтому диалог ввода дополнительного кода ориентирован на этот язык.

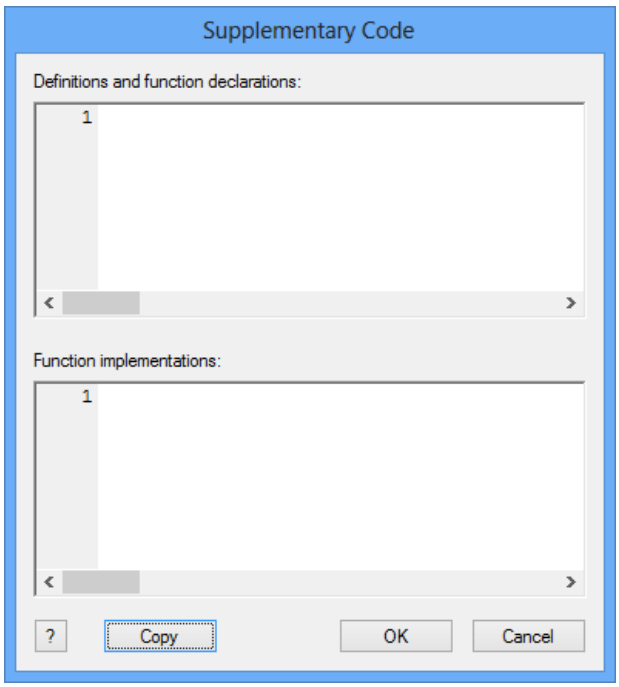


Рис. 4.14. Добавочный код

В верхнем окне диалога вы вводите определения и декларации функций, как это принято, например, в файле заголовка языка Си, а в нижнем окне выполняете реализацию функции.

Настроив проект, вы можете транслировать графический язык Flowcode в код на языке Си, используя следующий раздел пункта Build – это и есть Compile to C. Можно транслировать программу в hex-файл, трансляция в Си и на ассемблер будет выполнена автоматически. И, если у вас программатор, работающий с Flowcode, то вы можете сразу загрузить программу в микросхему, выполнив команду Compile to Chip.

Следующие три раздела из меню Build позволяют вам увидеть программу на языке Си, на ассемблере и прочитать сообщения компилятора. Последнее особенно полезно, если появляются проблемы с получением hex-файла.

Для начинающих изучать программирование микроконтроллеров не менее полезно, создавая простейшие программы, просматривать код программы на языке Си и ассемблере. Освоить эти языки программирования хотя бы на уровне близкого знакомства не только полезно, но и необходимо.

Провести подобные операции позволяет любая demo-версия Flowcode. Например, соберите программу мигающего светодиода: включить вывод, сделать паузу в секунду, выключить вывод, сделать паузу в секунду. Всю эту простую программу выполнить в бесконечном коде. Используя просмотр на языке Си, вы увидите, как программа может быть написана, чтобы компилятор, который используется в программе Flowcode, смог её оттранслировать. Не следует забывать, однако, что написание однотипных команд (синтаксис) и функции могут различаться для разных компиляторов. Как правило, компилятор транслирует текст программы на ассемблер.

Аналогично можно посмотреть и то, как выглядит текст программы на ассемблере.

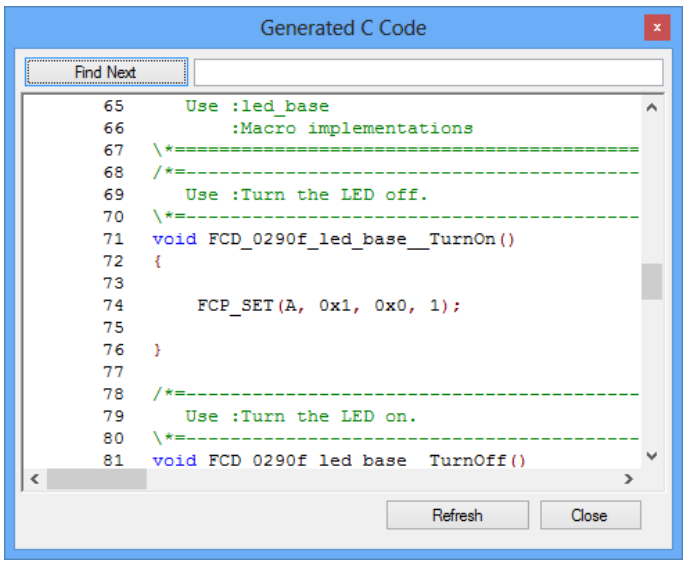


Рис. 4.15. Просмотр программы на языке Си

Программа Flowcode может работать не только с компилятором, установленным по умолчанию. Вы можете использовать и другой компилятор. Для этого используйте раздел Compiler Options (опции компилятора).

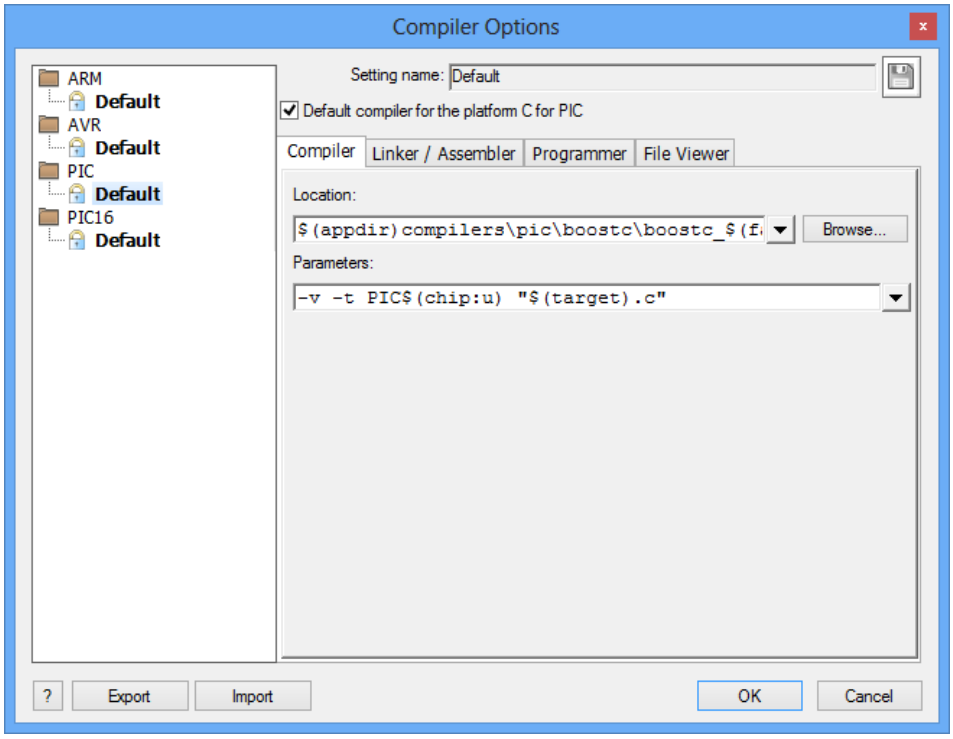


Рис. 4.16. Настройки компилятора

Компилятор потребует добавления параметров, их следует знать, если вы используете другой компилятор. Аналогично вы можете использовать другие компоновщик и ассемблер, настроив их на закладке Linker/Assembler. Это особенно полезно в тех случаях, когда вы умеете пользоваться языком Си и привыкли работать с другим компилятором, но эту возможность следует проверить, будет ли работать ваша версия Flowcode с другим компилятором.

На закладке Programmer вы можете задать свою модель программатора, если установки по умолчанию не поддерживают работу с вашим программатором.

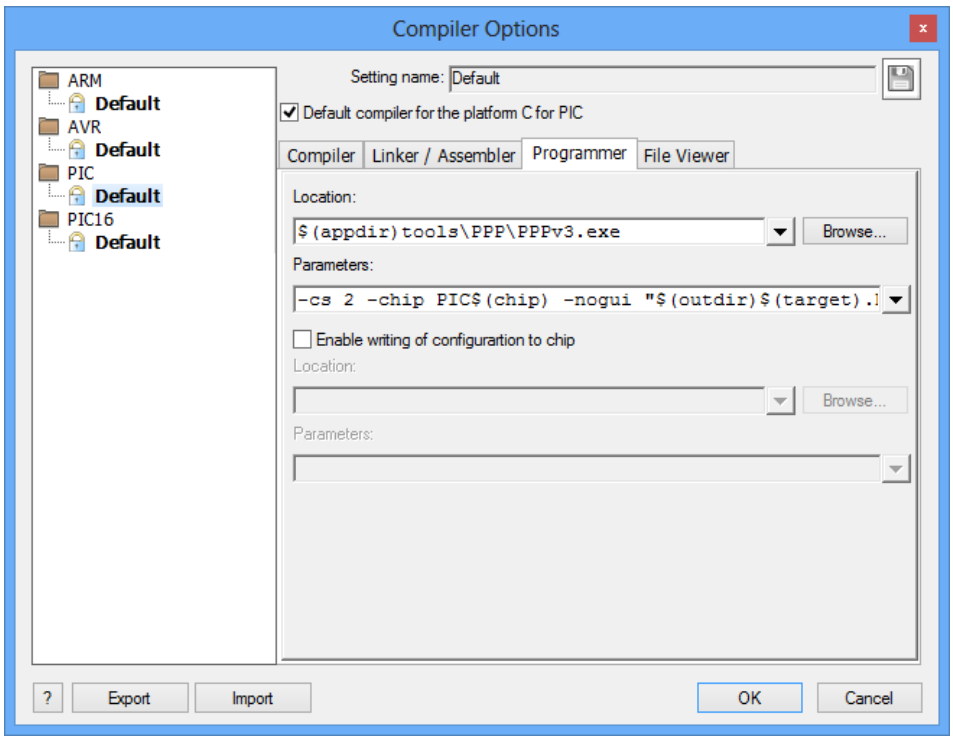


Рис. 4.17. Страница настройки программатора

Предыдущие версии Flowcode, например, можно было настроить для микроконтроллера AVR, чтобы программировать модуль Arduino. Но, как и с компилятором, следует знать дополнительные параметры и проверить правильность работы программатора до того, как вы приступите к работе. Многие среды разработки ориентированы на программаторы, которые выпускают разработчики программы. Так MPLAB работает с программатором PICKit, причём последние версии ориентированы на модель PICKit 3.

На сайте Matrix Multimedia можно найти программатор, предлагаемый разработчиками программы:

Рис. 4.18. Программатор Matrix Multimedia

Предыдущие версии Flowcode, изначально не ориентированные на продажу и распространения в нашей стране, были очень чувствительны к кириллице. Если вы помещали проект в папку «Документы», то возникали проблемы с трансляцией программы. И даже комментарии, написанные на русском языке, могли вызывать проблемы, хотя по определению комментарии игнорируются при всех трансляциях.

Есть два пути решения этого: либо не используйте кириллицу ни в указании пути к файлам, ни в программе. Либо проверьте работу программы, и будьте готовы к неожиданностям.

Эти рекомендации касаются и использования сторонних компиляторов, и использования программаторов. Если вы не уверены, скажем, что PICKit 2 будет работать с программой, лучше используйте отдельную программу для «прошивки» микросхемы или внутрисхемной отладки. Впрочем, последнее решается несколькими удачными или неудачными попытками. Не забудьте только сохранить исходные параметры настроек, чтобы вернуться к ним в случае неудачи.

Если настраивая внешний вид программы, вы стараетесь выбрать цвет по своему вкусу, и что-то пошло не так, вы можете смириться с неудачным видом программы, но для случая, когда ваша программа перестала транслироваться… вы потеряете смысл работы с Flowcode. Будьте осторожны в ваших желаниях.

Глава 5. Основное меню (Window, Help)

Пункт Window

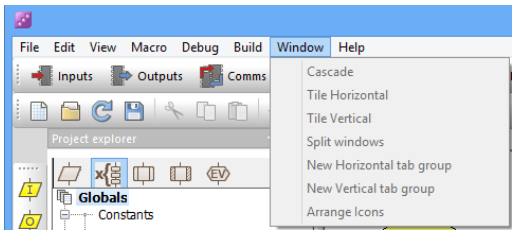


Рис. 5.1. Меню пункта Window

Некогда обязательный пункт основного меню позволял организовать окна программы: выстроить их каскадом, по горизонтали или вертикали, разделить их, - словом, организовать расположение окон наиболее удобным образом. Окна Flowcode могут быть закреплены в основном окне или оставаться в «свободном плавании». Если кроме основной программы есть подпрограммы, то «оживают» разделы меню:

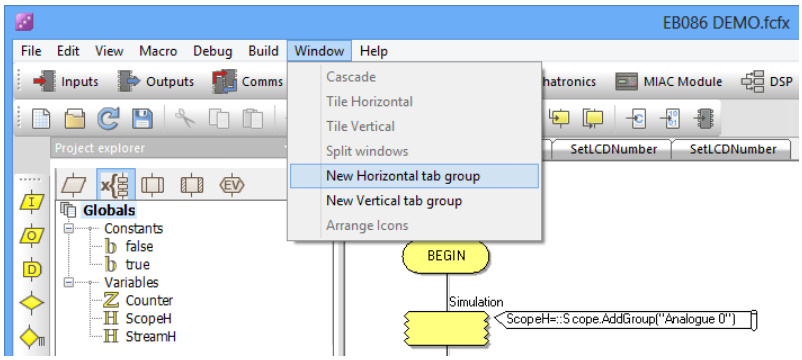


Рис. 5.2. Вид меню при наличии подпрограмм

Выделив одну из закладок, можно изменить вид рабочего поля:

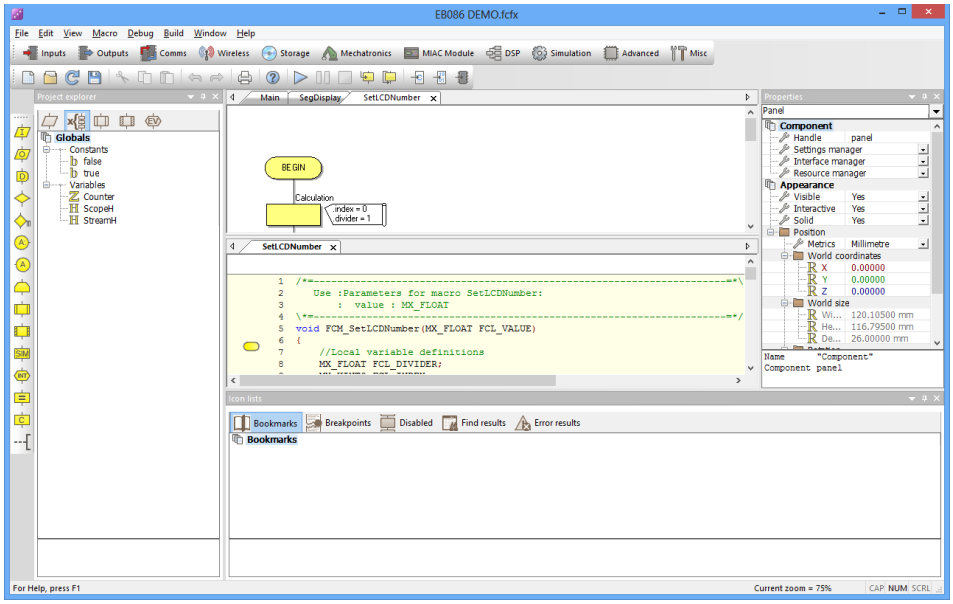


Рис. 5.3. Использование раздела New Horizontal tab group

Программа позволяет запустить одновременно и две копии.

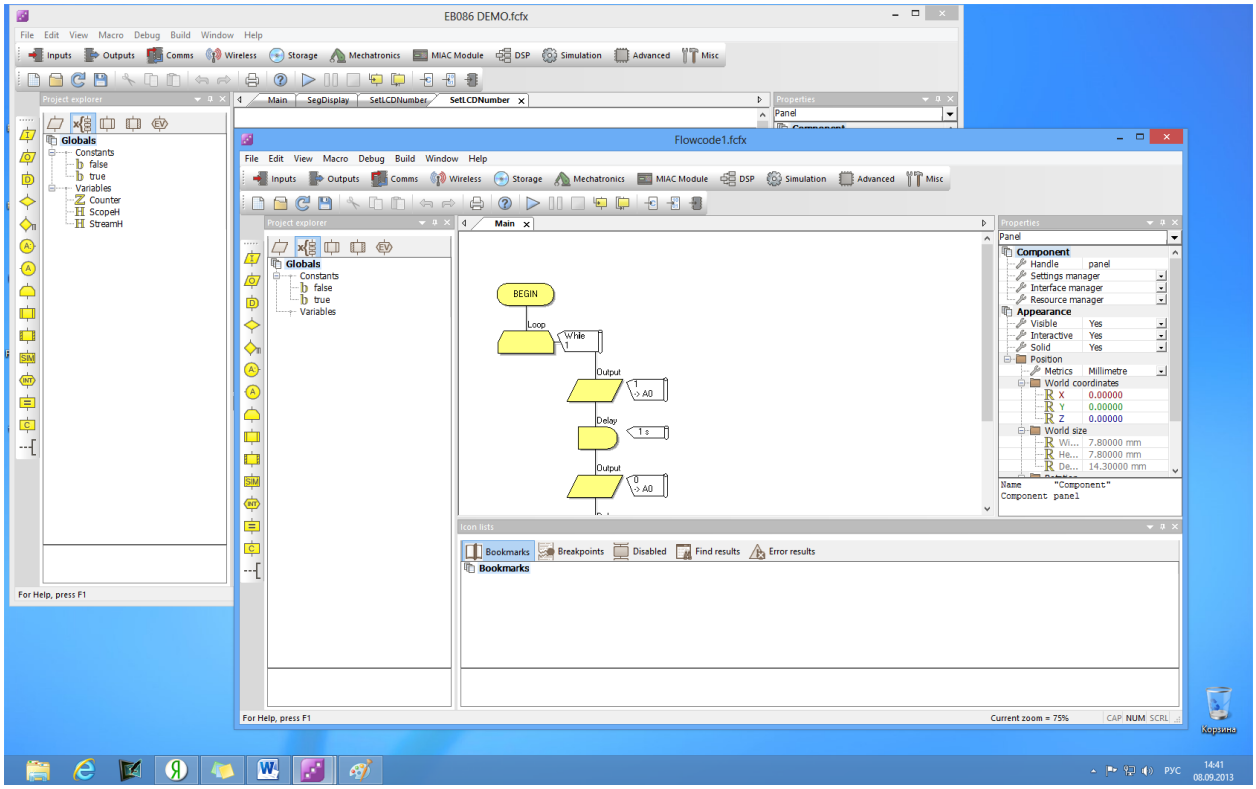


Рис. 5.4. Две копии программы одновременно

Пункт Help

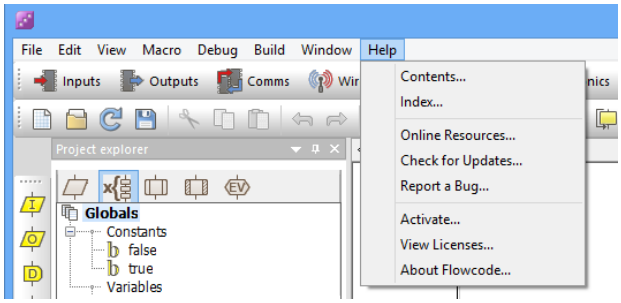


Рис. 5.5. Меню Help

Раздел Contents (содержание) вызывает появление окна выбора:

Рис. 5.6. Выбор содержания помощи

Всё содержание будет работать при подключении к Интернету. Примеры программ, о которых упоминалось выше, можно скачать, используя Flowcode Examples Files.

Полезно, начиная работать с программой, посмотреть Training Videos (учебные уроки).

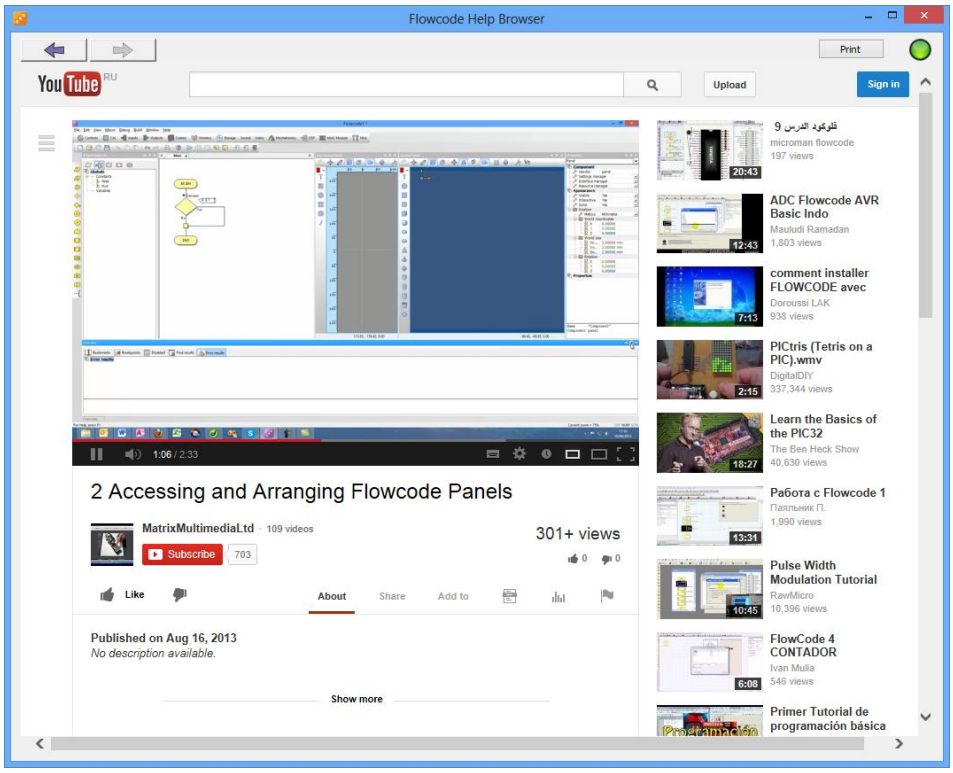


Рис. 5.7. Один из уроков видео курса

Раздел Index открывает окно поиска. Если в текстовое поле ввести слово для поиска, можно увидеть все разделы, где это слово встречается:

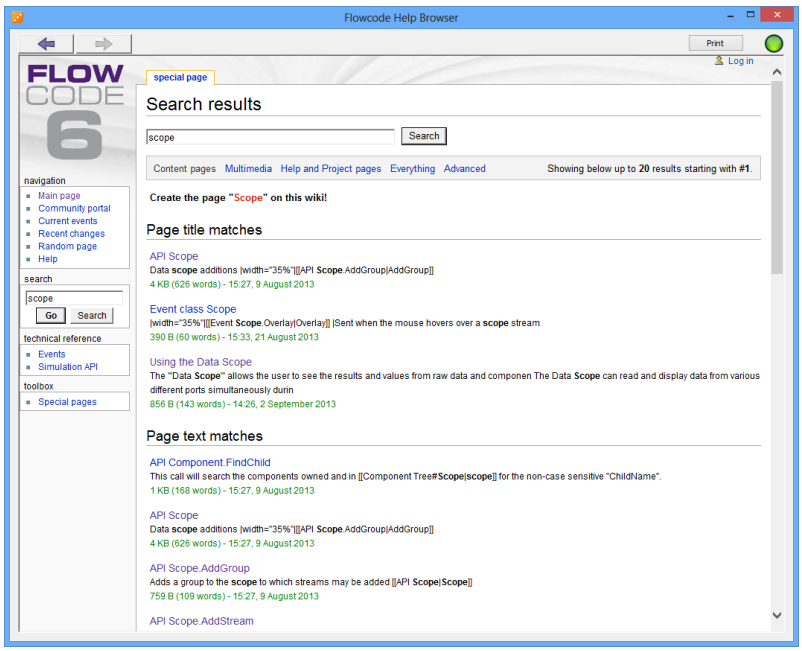


Рис. 5.8. Окно поиска Help

On line Resources открываются в том случае, когда в настройках программы установлена опция, разрешающая доступ в Интернет.

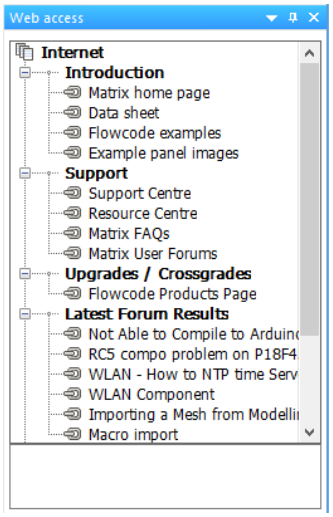


Рис. 5.9. Ресурсы программы в Интернете

Следующие два раздела позволяют проверить наличие обновлений и сообщить об ошибке, активировать вашу копию программы и проверить лицензию:

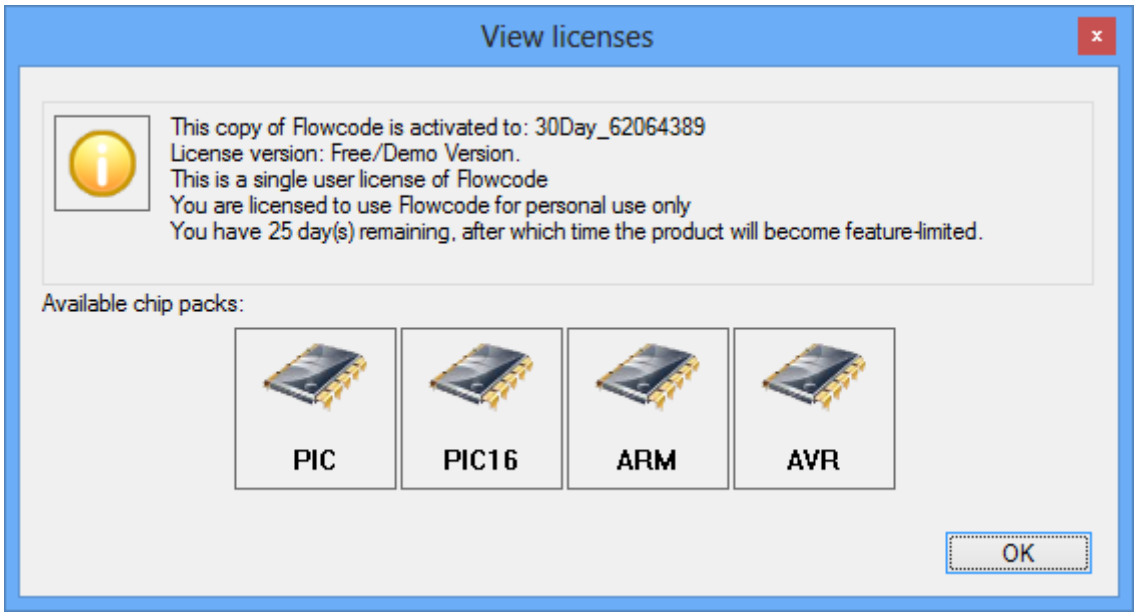


Рис. 5.10. Лицензия на программу Flowcode

Моя пробная версия завершит работу через 25 дней, тогда и рассказ закончится. А пока…

Инструментальная панель основных команд редактора

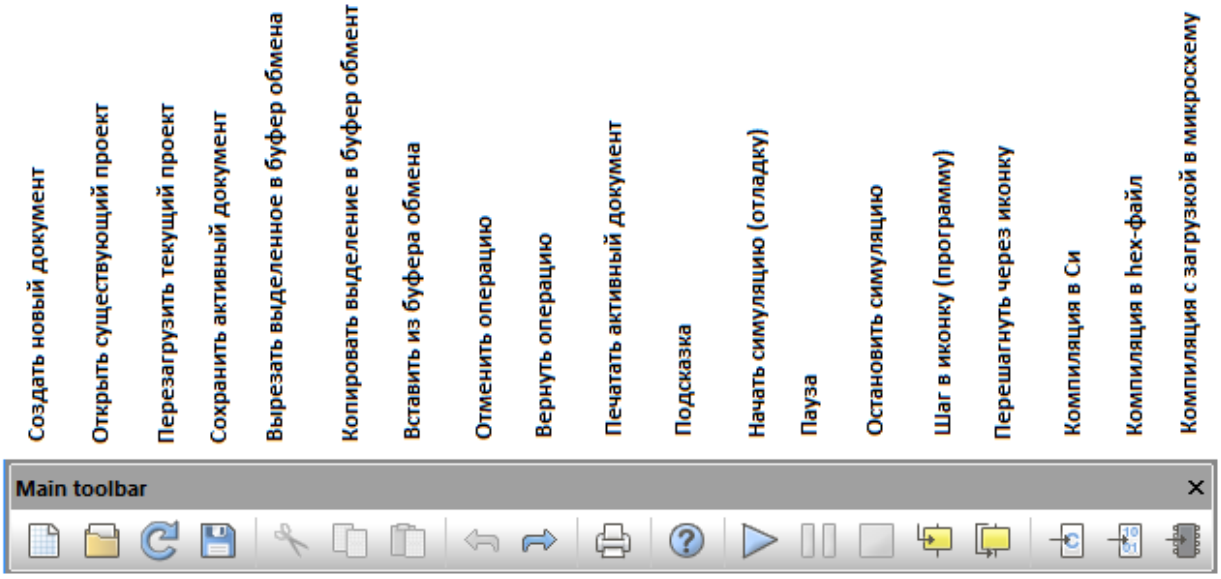


Рис. 5.11. Назначение иконок инструментальной панели редактора

Каждая из иконок, если на неё навести курсор мышки, даёт подсказку о своём назначении:

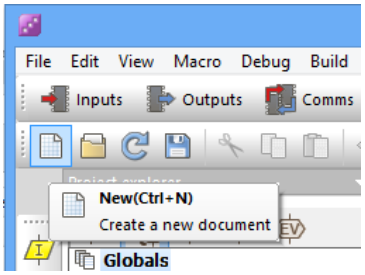


Рис. 5.12. Подсказки при наведении курсора мышки

Аналогичными свойствами обладают все элементы программы. Если навести курсор мышки на команды программы (иконки элементов программы), то можно увидеть их содержание:

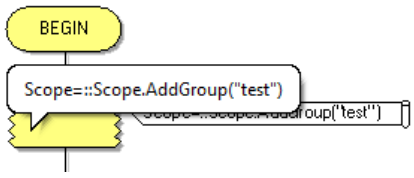


Рис. 5.13. Подсказка по команде

Впрочем, если вы не отключили эту возможность в настройках, то содержание команды выводится и без наведения курсора мышки на иконку команды. Но в тех случаях, когда вы выбираете операции из свойств компонентов, такая подсказка не будет лишней.

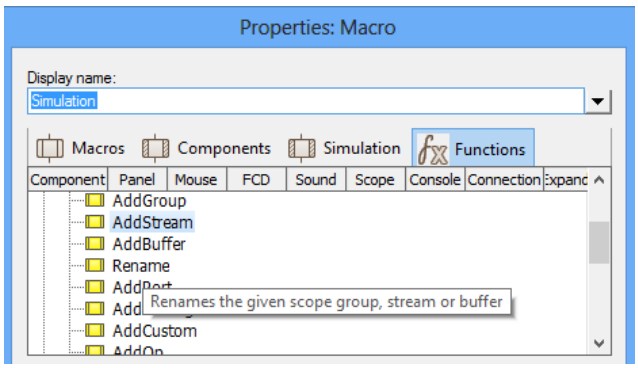


Рис. 5.14. Подсказка по командам макроса

Возвращаясь к основной инструментальной панели, многие операции редактора программы удобнее выполнять одним нажатием на иконку, чем выводить список из основного меню и выбирать нужный пункт.

Как и окна программы, инструментальные панели можно прикрепить к основному окну программы, а можно переместить в удобное для вас место.

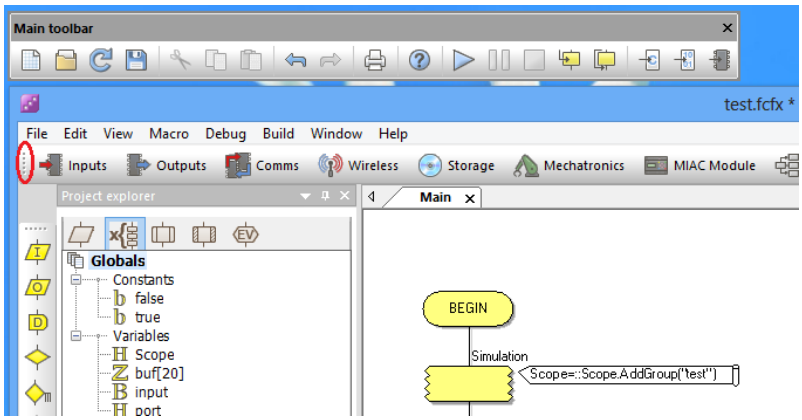


Рис. 5.15. Перемещение инструментальной панели по экрану монитора

Чтобы переместить инструментальную панель, достаточно навести курсор мышки на отмеченную на рисунке область панели (появится значок перемещения), нажать и удержать левую клавишу мышки, подцепив мышкой панель, и перенести её. Аналогично, если вам стало неудобно работать с панелью после её переноса, вы можете навести курсор мышки на область заголовка панели (курсор изменит свой вид), подцепить панель левой клавишей, и, удерживая клавишу мышки, вернуть панель на прежнее место. Когда вы переместите панель в нужное место, вид основного окна несколько изменится, освобождая место для панели.

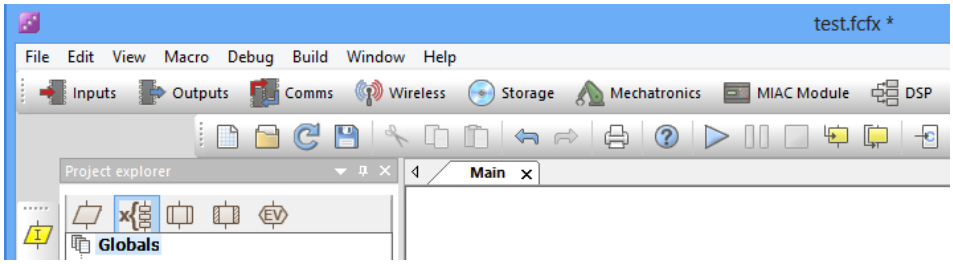


Рис. 5.16. Изменение вида основного окна при возвращении панели на прежнее место

Вы можете переместить панель к левой кромке основного окна программы.

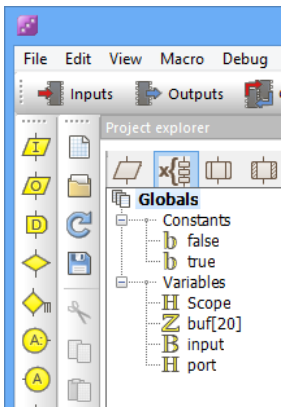


Рис. 5.17. Размещение инструментальной панели у левой кромки окна программы

Вы можете поместить основную панель под инструментальной панелью компонентов или над ней, как вам удобнее.

Завершая рассказ об основном меню программы, я хочу отметить одну особенность – у меня не получилось, когда потребовалось, из подсказки скопировать и перенести текст в текстовый процессор. Выделив нужный мне фрагмент текста, я использовал правую клавишу мышки для вызова меню, где есть команда копирования:

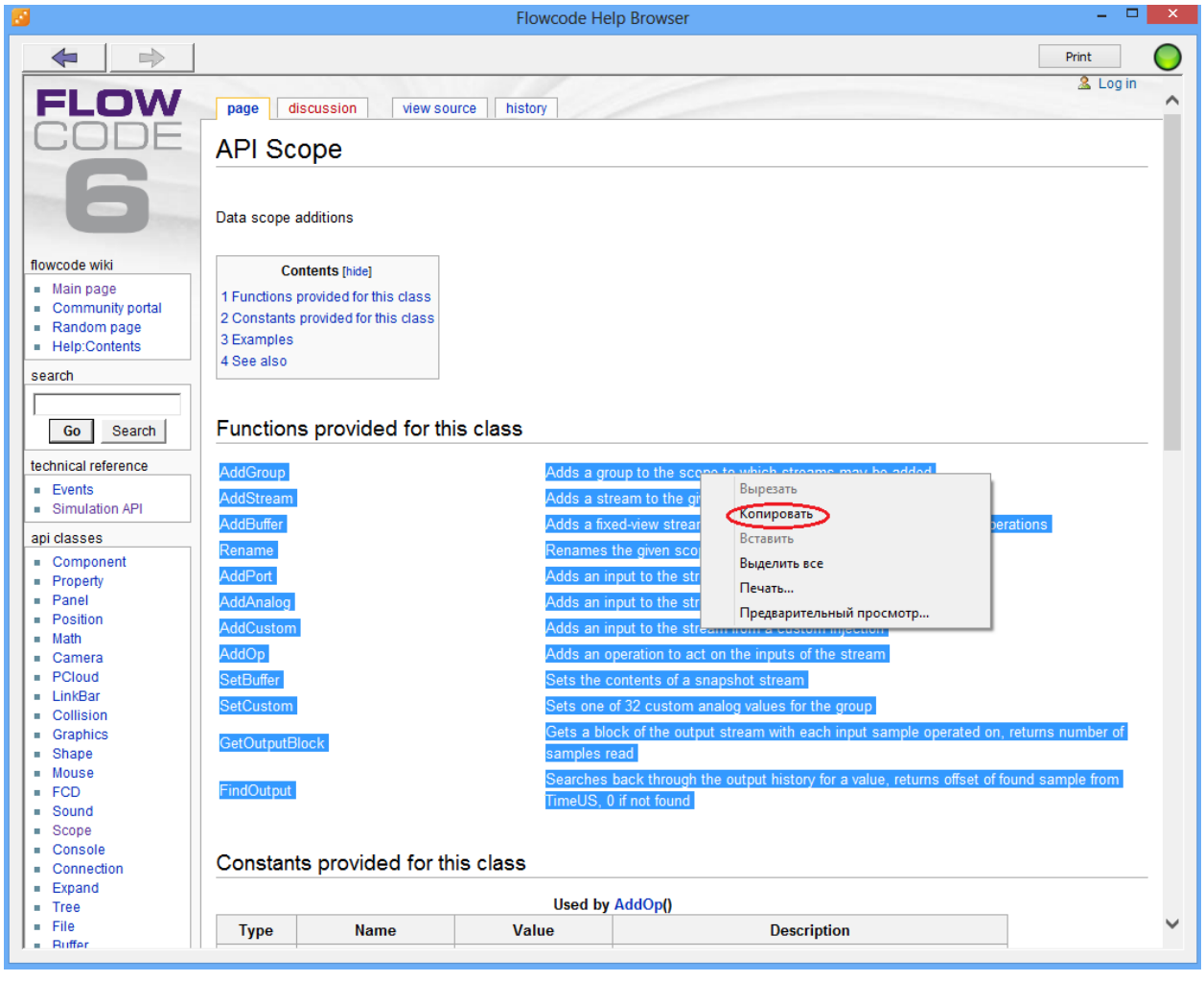


Рис. 5.18. Копирование текста подсказки

Но при переходе в редактор текста я обнаружил, что буфер обмена пуст. Или копирование текста из справки происходит в другой буфер, или есть запрет на копирование – я не знаю, но знаю, что многие не могут легко и свободно читать текст на английском, для них удобнее скопировать текст и спокойно перевести его. Можно, конечно, распечатать подсказку на принтере, а затем на бумаге добавить перевод, но можно поступить иначе: с помощью web-браузера зайти в Wiki, где и скопировать нужный текст. Вот адрес в Интернете с той же страницей подсказки:

http://www.matrixmultimedia.com/wiki/index.php?title=API_Scope

Выделив нужный текст, можно добавить его на страницу в текстовом редакторе…

AddGroup – Adds a group to the scope to which streams may be added

…и перевести: (Команда AddGroup) добавляет группу (в окне осциллографа) к осциллографу, к которой могут быть добавлены потоки.

Глава 6. Панель элементов программы (Input, Output, Decision, Loop)

Инструментальная панель элементов программы при всей важности остальных панелей является, видимо, самой главной.

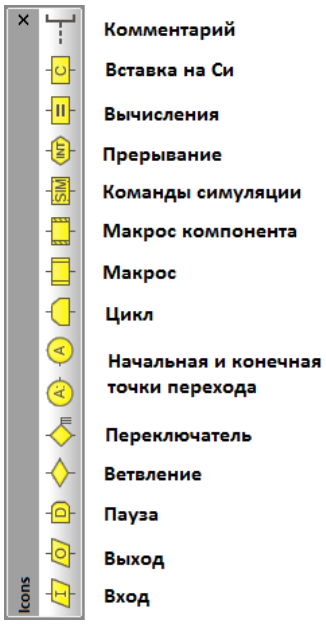


Рис. 6.1. Назначение элементов программы

Все эти элементы программы являются строительными её кирпичиками. В идеальном случае вы используете только эти элементы при создании программы для микроконтроллера. Насколько это возможно, зависит от вашего опыта работы, сложности программы и опыта работы с Flowcode.

Разберём назначение «строительных кирпичей» программы.

Input (вход)

Начиная работу над проектом с выбора микросхемы (не забудьте правильно настроить конфигурацию):

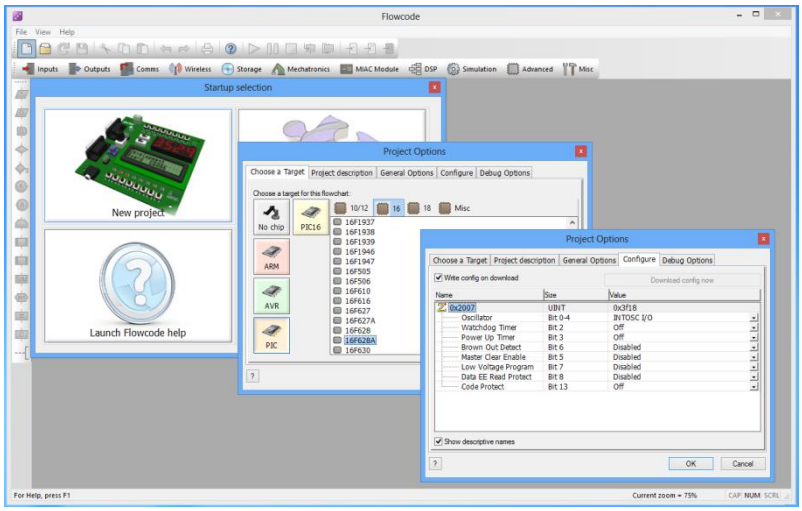


Рис. 6.2. Начало работы над новым проектом

Вы получите заготовку для программы в виде только двух элементов: начало и конец программы:

Рис. 6.3. Заготовка программы

Добавим в программу элемент работы с входом: «подцепите» левой клавишей мышки иконку Input (вход) и перенесите элемент к линии соединения двух элементов шаблона; курсор приобретает вид элемента, а при приближении к линии соединения появляется стрелка, указывающая место, где программный элемент будет вставлен.

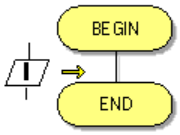


Рис. 6.4. Добавление элемента в программу

Программный элемент Input предназначен для обслуживания выводов микроконтроллера, работающих на вход. Вы должны определить, будете ли вы работать со всем портом сразу, будете ли вы работать только с одним выводом и создать переменную, которая будет ассоциироваться с этим входом. Входы микроконтроллера обязательно требуют связи с переменной. Переменную можно создать разными способами. Например, вы можете в окне навигации по проекту щёлкнуть правой клавишей мышки по разделу переменных (Variables) и выбрать из выпадающего меню пункт создания новой (Add new).

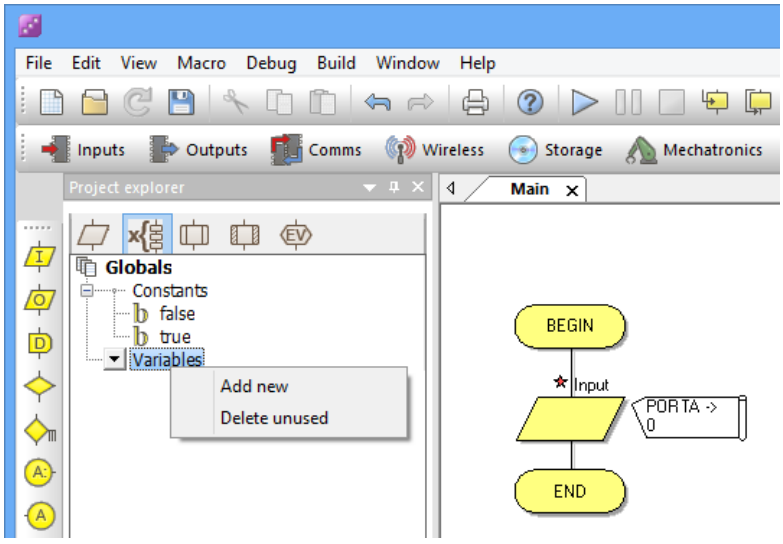


Рис. 6.5. Создание новой переменной в окне навигации

Но можете двойным щелчком левой клавиши мышки открыть диалоговое окно элемента Input, в окне переменных открыть (стрелка справа) диалог, где и открыть выпадающее меню правым щелчком мышки по тому же разделу:

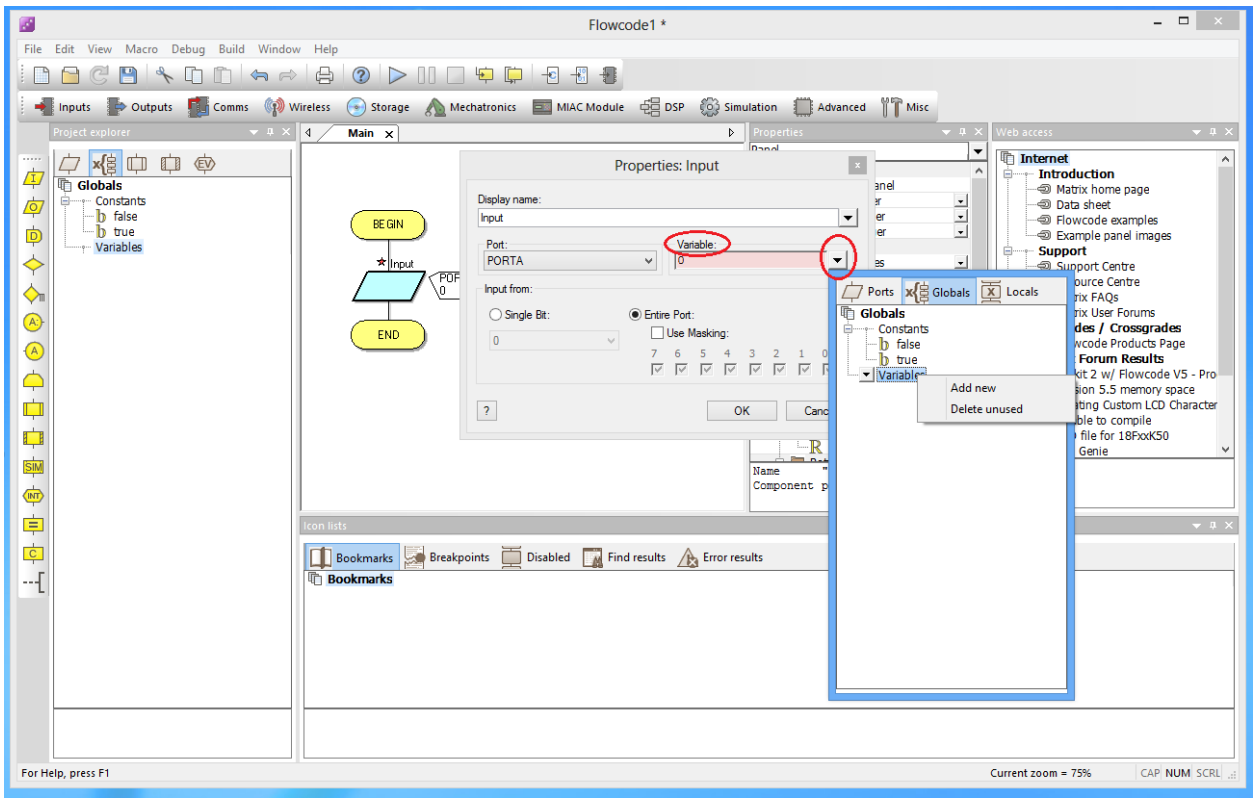


Рис. 6.6. Создание новой переменной в диалоге свойств входа

Положим, я назову переменную but1. Я собираюсь к этому входу подключить кнопку, поэтому я могу выбрать два типа переменных Byte или Bool. В диалоговом окне вы легко увидите разницу в этих двух типах переменных – первая может изменяться в диапазоне 0..255, вторая принимает значения 0 и 1.

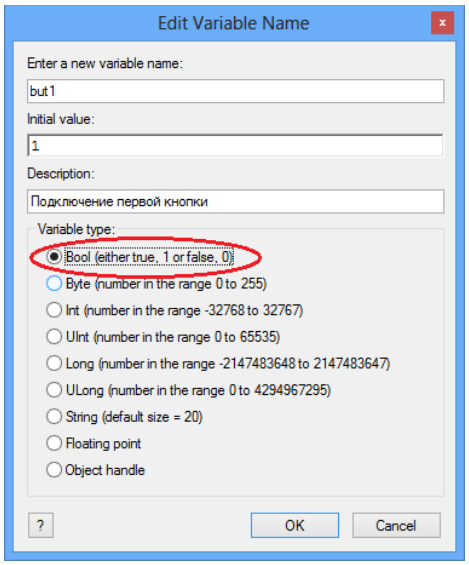


Рис. 6.7. Задание имени и типа переменной

Кнопка будет либо нажата (при этом она соединит вход с общим проводом), либо отпущена. Для этой цели достаточно только двух состояний, и мне хватит двух значений. Память у любого микроконтроллера не столь велика, как у современных компьютеров. Если можно сэкономить память, то это следует сделать – переменная булева типа занимает один бит, а байтовая переменная займёт восемь бит, то есть, одну ячейку памяти. Если Flowcode булевы переменные

собирает в одной ячейке памяти, то я смогу добавить ещё несколько кнопок, которые уместятся в одной ячейке памяти. Не поленитесь и добавьте описание этой переменной, позже вам пригодится описание, когда кнопок будет много. Если вы уже знаете, зачем будет нужна кнопка, то добавьте это в описание. Позже, выделив переменную щелчком правой клавиши мышки, вы можете выбрать из выпадающего меню команду редактирования переменной. Здесь вы прочитаете ваше примечание, а можете при необходимости изменить тип переменной.

После создания переменной двойным щелчком левой клавиши мышки отправьте её в окно диалога настройки входа (если в нём вы создавали переменную). Если переменную вы создали в окне навигации, то стрелка справа от окна переменной откроет диалог, в котором вы увидите вашу переменную, которую двойным щелчком левой клавиши мышки добавьте в окно ввода переменной. Вы можете, впрочем, просто впечатать в этом окне имя переменной, если создали её раньше.

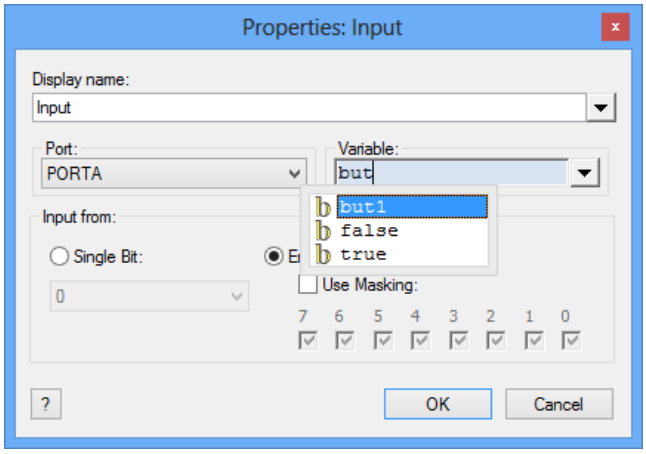


Рис. 6.8. Добавление переменной в текстовом окне

Обратите внимания при вводе имени на цвет фона – изначально он голубоватый, если вы ошибаетесь, цвет фона станет розовым, а если вы правильно ввели имя переменной (список переменных появляется снизу), фон окна станет белым. Аналогичная цветовая гамма поможет вам позже, когда вы обратитесь к добавлению, например, свойств макросов дополнительных компонентов; не всегда ясно, в каком виде добавляется параметр, а цвет укажет путь к правильному значению.

Осталось определить, будет ли переменная следить за всем портом или за одним из битов порта, будет ли использоваться маска или нет, задать маску, если она нужна. В данном случае устраивает один бит для подключения одной кнопки к выводу порта, и в окне выбора вывода можно выбрать из выпадающего списка, какой именно вывод нужен.

Теперь любые изменения входа будут отражаться на состоянии переменной. Входы микроконтроллера, если пока умолчать об аналоговых входах встроенного АЦП, обычно обслуживают кнопки, клавиатуру, датчики, имеющие два состояния, включён и выключен. К таким датчикам относятся концевые датчики, пожарные и охранные датчики, датчики температуры некоторых нагревателей и т.п.

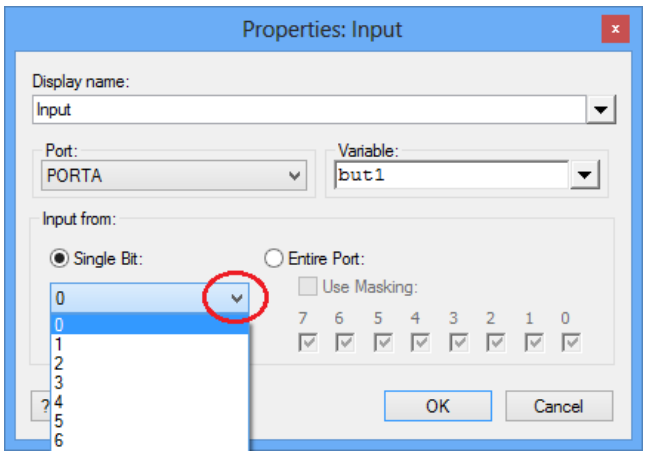


Рис. 6.9. Выбор вывода порта

Все выводы (или почти все) микроконтроллера, как правило, могут работать на вход или на выход. Будет ли вывод входом или выходом определяется при программировании микроконтроллера. И, как правило, входы и выходы работают как цифровые, то есть, на входах отслеживается их состояние, в высоком или низком состоянии находится вход, и по состоянию входа определяется состояние выхода. Исключение для аналоговых входов встроенного модуля АЦП и, например, для выходов встроенного модуля USART, который передаёт информацию в сеть.

После того, как мы соединили вход микроконтроллера с переменной, обслуживающей его, выбрали бит порта и решили вопрос с маской, мы можем закрыть диалог, нажав на кнопку **ОК**. Программа изменит свой вид:

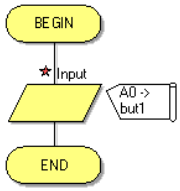


Рис. 6.10. Начало сборки программы

Output (выход)

Элемент программы выход служит для управления выходом, который может принимать два значения 0 и 1, состояние низкого и высокого уровня. Обычно к выходу микроконтроллера подключают исполняющее устройство, например, реле или индикаторный светодиод (используя дополнительные элементы).

Программный элемент Output добавляется в программу аналогично элементу Input. И он имеет аналогичное диалоговое окно для настройки состояния выхода (0 или 1), для привязки выхода к выводу микроконтроллера. Все операции аналогичны тем, что описаны выше. Добавим к нашей программе выход и откроем диалоговое окно его свойств, дважды щёлкнув левой клавишей мышки по иконке выхода.

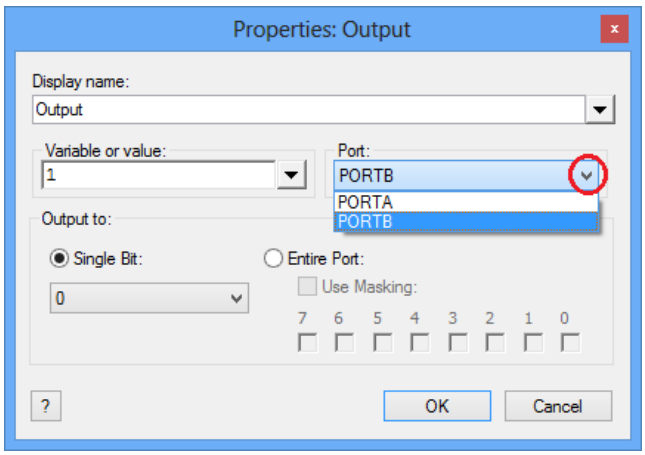


Рис. 6.11. Диалоговое окно настройки выхода

В окне Port вы можете выбрать порт, к которому будет подключен исполняющий элемент схемы. Для выбора из списка служит стрелка справа, отмеченная на рисунке.

В окне Display name (отображаемое имя) вы можете задать любое имя, которое отобразится в графике программы, помогая вам понять назначение программных элементов. Я не исключаю, что проблем не возникнет, если вы будете писать по-русски, но это в том случае, если производитель поддерживает кириллицу. Иначе проблемы могут возникнуть, но попробовать стоит.

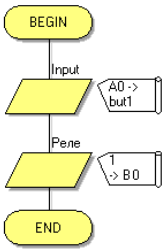


Рис. 6.12. Программа с добавленным выходом

Пока программа не имеет смысла. Некоторый смысл программе придаст следующий элемент.

Decision (ветвление)

Этот элемент графического языка соответствует конструкции если..то..иначе (if..then..else) языков высокого уровня. Добавим этот элемент к программе.

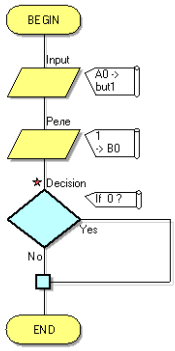


Рис. 6.13. Добавление ветвления программы

Ветвление программы произойдёт при выполнении условия ветвления. Откроем диалоговое окно этого элемента двойным щелчком по иконке левой клавиши мышки.

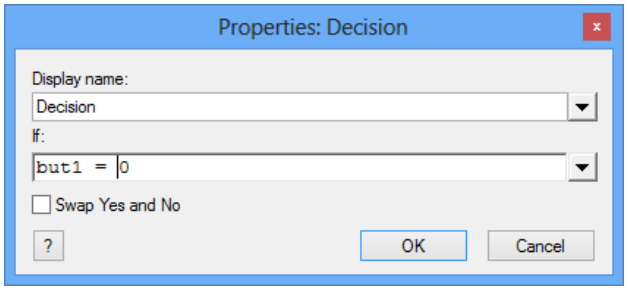


Рис. 6.14. Диалоговое окно настройки ветвления

В окне записи условия я добавил его – равенство нулю переменной, связанной с входом микроконтроллера. Напомню, что кнопка соединяет вход с общим проводом.

При выполнении этого условия выход будет принимать высокий уровень. Кнопка **ОК** завершит настройку элемента Decision. Для переноса элемента выхода в ветвь программы его можно «подцепить» его мышкой и перенести в новое место.

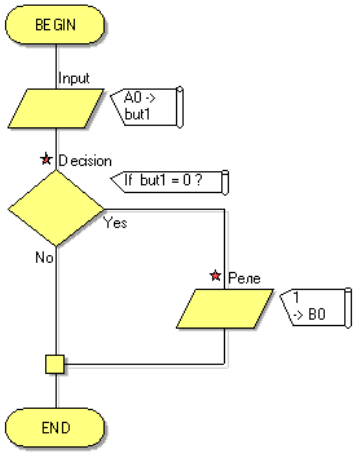


Рис. 6.15. Изменение программы

Для разумного завершения программы в ветвь, обозначенную как No, то есть, условие не выполняется, добавим противоположное состояние выхода. Для этого с панели элементов программы перенесём ещё один элемент Output и настроим его.

Теперь смысл программы таков – когда кнопка нажата, исполняющий элемент схемы будет включаться, а когда кнопка отпущена, этот элемент будет выключен.

Программа приобретёт вид:

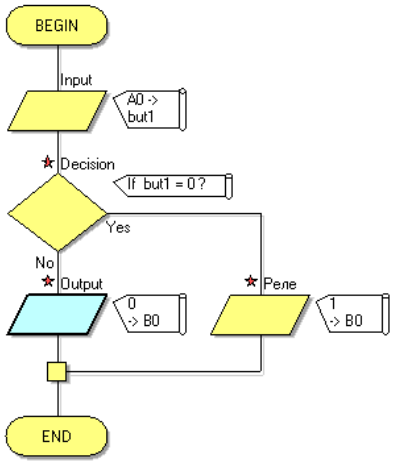


Рис. 6.16. Дополнение программы

Хотя программа очень проста, она вполне осмыслена. Остался один недостаток – программа будет выполнена только один раз, после чего перестанет работать, скорее всего, вы даже не успеете нажать кнопку. Чтобы устранить эту недоработку, используется простой приём – программа работает в бесконечном цикле.

Loop (цикл)

Этот элемент программы не следует за ветвлением, но он сейчас нужен. Перенесём его в программу, а все предыдущие элементы программы перенесём внутрь цикла.

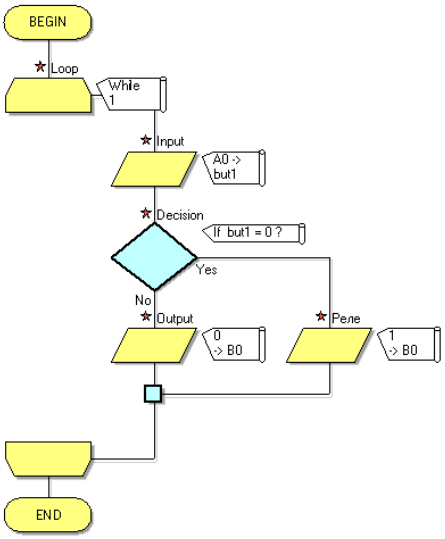


Рис. 6.17. Окончательный вид программы

По умолчанию условие выполнение цикла записывается так, как это видно на рисунке. Откроем диалоговое окно цикла.

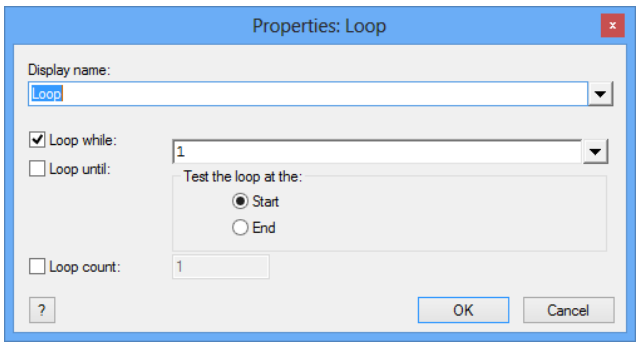


Рис. 6.18. Диалоговое окно цикла

Диалог позволяет выбрать один из типов цикла: цикл выполняется когда (когда выполняется условие, записанное в текстовом поле), цикл выполняется пока (выполняется условие), цикл выполняется заданное число раз (Loop count). Кроме того, вы можете выполнять проверку в начале цикла (Start), а можете проверку сделать в конце (End).

При том условии, что сейчас, цикл будет работать бесконечно. Проверка условия осуществляется по принципу: истинно или ложно, единица или ноль. Единица означает, что условие работы цикла выполняется всегда. Вы можете записать другое условие, как мы записывали, например, условие ветвления программы.

Те, кто знаком с электрическими схемами, сочтут программу лишенной смысла – нажал на кнопку и светодиод загорелся, зачем микроконтроллер? Достаточно кнопки, светодиода и источника питания. И они правы. Но предположим, что светодиод должен загореться не тогда, когда мы нажимаем кнопку, а когда отпускаем. В программном варианте нам достаточно поменять две команды управления выходом местами, а без микроконтроллера? И ещё, кнопка может быть датчиком, срабатывающим, когда поворачивается рычаг, за который цепляются заготовки на ленте транспортёра, и датчик подключён к электронному счётчику, подсчитывающему количество заготовок. Чтобы подсчёт был точен, следует блокировать дребезг контактов, который есть у всех механических контактов. В программе можно легко исправить ситуацию, а без микроконтроллера это потребовало бы дополнительных схемных компонентов и т.д. И наконец, чем сложнее программа, тем сложнее её понять, а нам важно понять назначение программных элементов.

Любую программу, как бы ни была она проста, следует проверить. Проверить в первую очередь средствами отладки программы, этим и займёмся.

Проверка программы

Начнём проверку с того, что включим системную панель.

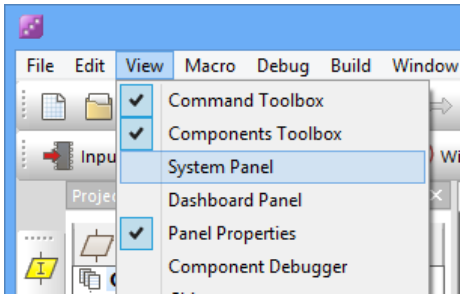


Рис. 6.19. Включение системной панели

Из набора дополнительных компонентов (к ним вернёмся позже) добавим на панель светодиод и кнопку. Кнопку можно найти в раздел Input дополнительных компонентов.

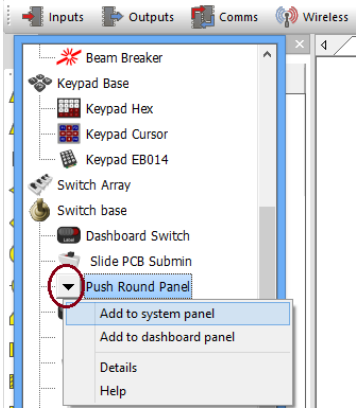


Рис. 6.20. Добавление на системную панель кнопки

Я выбрал простой выключатель, а из выпадающего списка (щелчок правой клавишей мышки по названию или левой клавишей мышки по отмеченной стрелке) выбрал Add to system panel. В разделе Outputs можно найти LED 5mm Panel (светодиод), который тоже добавим на системную панель. Системная панель приобретает вид:

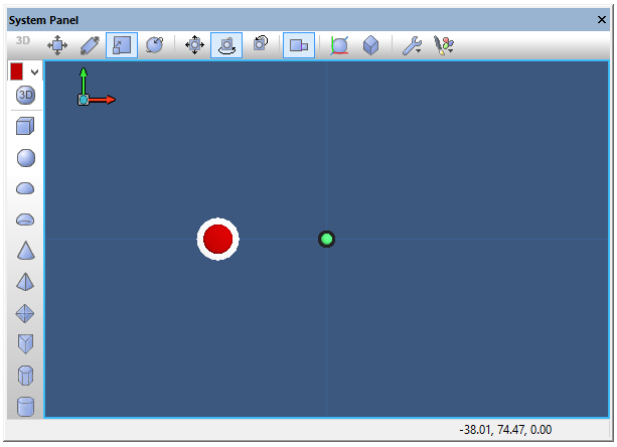


Рис. 6.21. Вид системной панели для проверки программы

Но добавить компоненты на панель мало, нужно их подключить к микроконтроллеру. Выделим выключатель на системной панели и обратимся к его свойствам в окне свойств, обратив внимание на свойство Connection. В этой версии Flowcode процедура изменила свой вид. Достаточно указать стрелкой курсора на вывод RA0 и подключение состоялось. Кнопка будет подключена к выводу 0 порта А.

Аналогично мы подключим светодиод. На макетной плате светодиод следовало бы подключить через токоограничительный резистор, но при отладке в программе этого не требуется. Да и светодиоды бывают разные, как, впрочем, и микроконтроллеры.

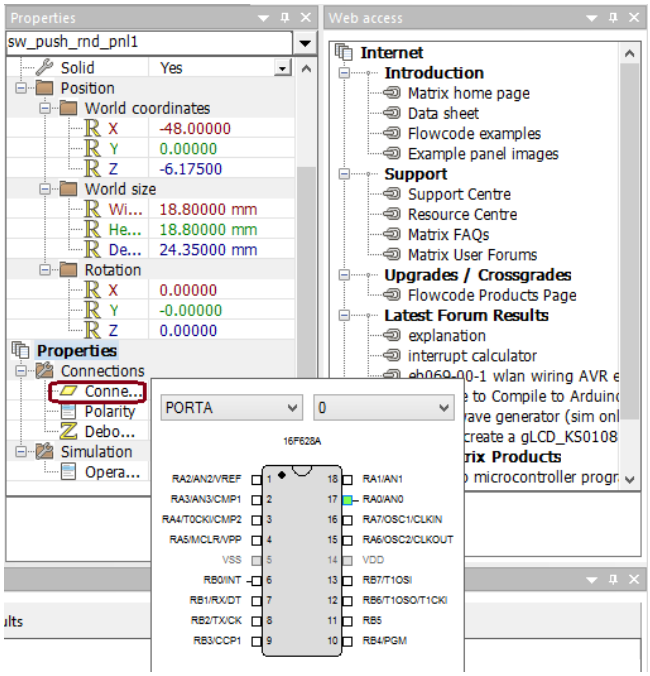


Рис. 6.22. Подключение кнопки к выводу входа

Подключив кнопку, следует обратить внимание ещё на одно свойство, которое называется Polarity.

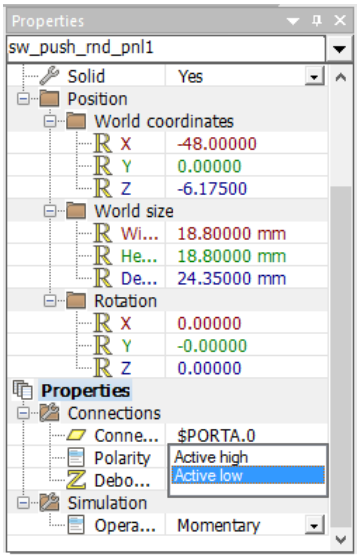


Рис. 6.23. Полярность подключения кнопки

Формулировка Active low означает, что нажатая кнопка даст низкий уровень напряжения на входе микроконтроллера.

Свойства светодиода можно увидеть на его странице свойств. Аналогично, как мы это сделали с выключателем, можно изменить полярность (то, как будет подключен светодиод).

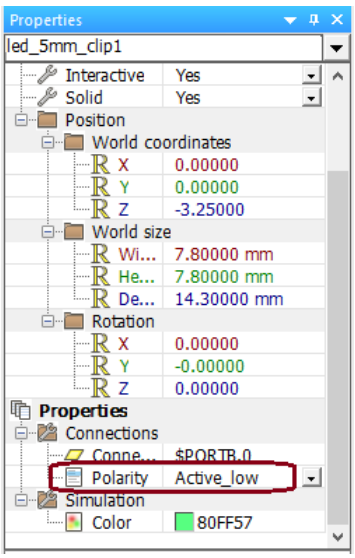


Рис. 6.24. Свойства светодиода

Переключение страниц свойств происходит при выделении элемента на системной панели. Но при необходимости вы можете выбрать нужную страницу, нажав на кнопку со стрелокой справа от окна элемента и выделив нужный компонент в выпадающем списке.

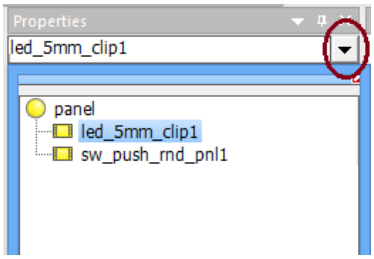


Рис. 6.25. Выбор элемента для отображения страницы свойств

Запустим симуляцию, либо используя иконку инструментальной панели, либо используя раздел основного меню в пункте Debug. Нажимая левой клавишей мышки «кнопку» на инструментальной панели, проверим работу программы.

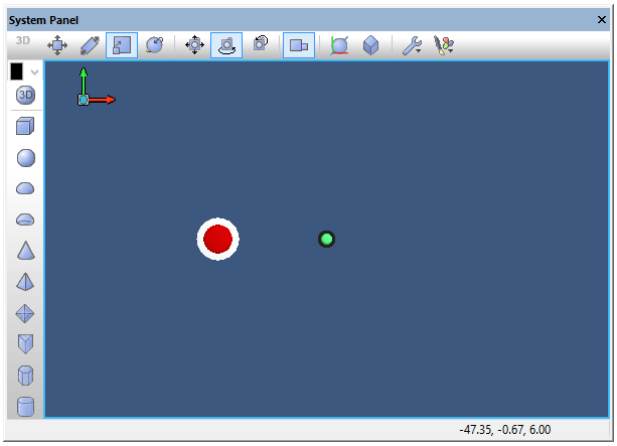


Рис. 6.26. Проверка программы с помощью симуляции

Глава 7. Панель элементов программы (Switch, Connection, Macro)

Перед тем, как продолжить рассмотрение программных компонентов, я хочу напомнить, что каждый проект, который вы создаёте, по завершению будет содержать ряд файлов.

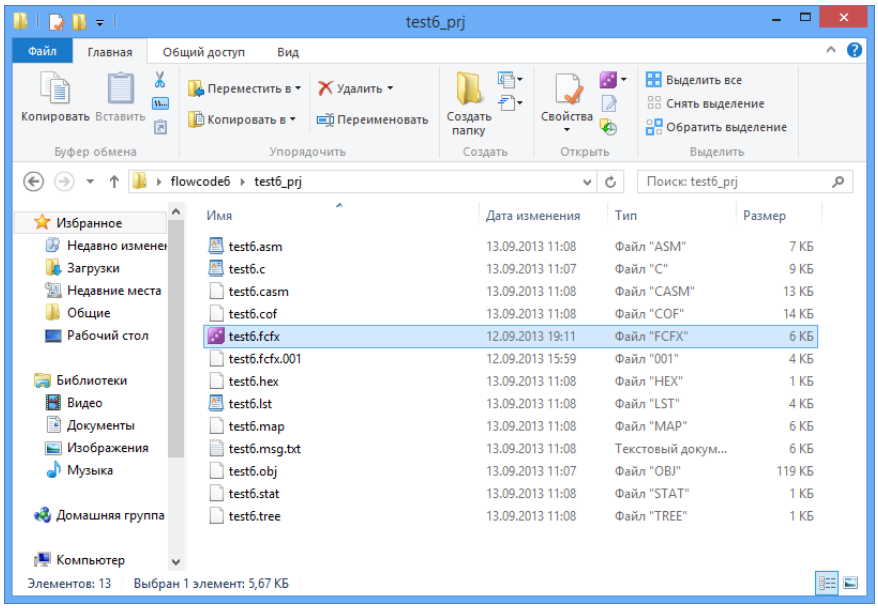


Рис. 7.1. Пример файлов проекта test6 после компиляции

Кроме исходного файла проекта, кроме файла для загрузки в микросхему проект создаст файл на языке Си и на ассемблере, объектный файл и отладочные файлы. Начиная работу над проектом, создайте папку, которую укажете программе, сохраняя рабочий файл программы. При работе даже над небольшой программой появляется много вариантов, которые хотелось бы сохранить, а держать их все в одной папке – простой, но эффективный способ быстро запутаться.

Операционная система Windows 8, в которой я работаю, все системные названия использует с англоязычными именами. В проводнике папки видны под именами «Документы» или «Рабочий стол», а при сохранении файлов используются названия с латиницей.

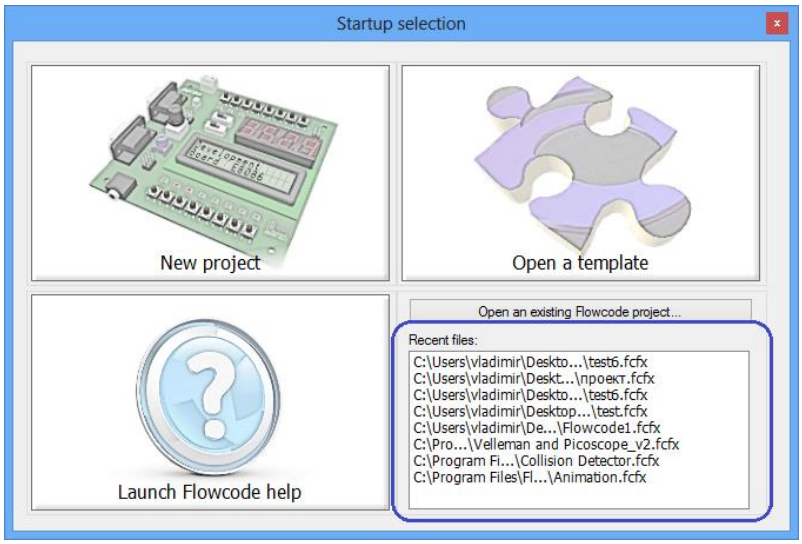


Рис. 7.2. Название системных папок в Windows 8

Поэтому я могу хранить рабочий проект в любой из системных папок, проблем не возникает. Но попробуйте в этом случае создать папку на рабочем столе с именем «мой проект». Сохраните в ней новый проект, который тоже можете назвать «проект». Первая же попытка компиляции проекта, надеюсь, убедит вас в том, что названия лучше использовать в написании латиницей.

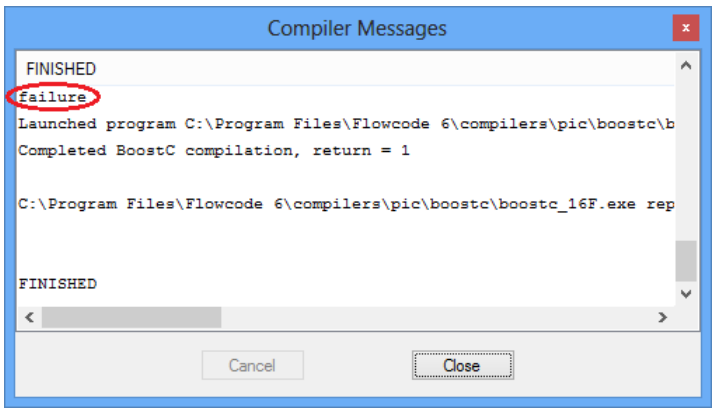


Рис. 7.3. Сообщение компилятора при кириллическом названии папки

Switch (переключатель)

Часто при создании программы приходится использовать ветвление несколько раз подряд. Если параметр в условии ветвления остаётся постоянным, то гораздо удобнее использовать такую программную конструкцию, как переключатель. Хотя предыдущий пример этого не требует, давайте заменим обычное ветвление программы переключателем.

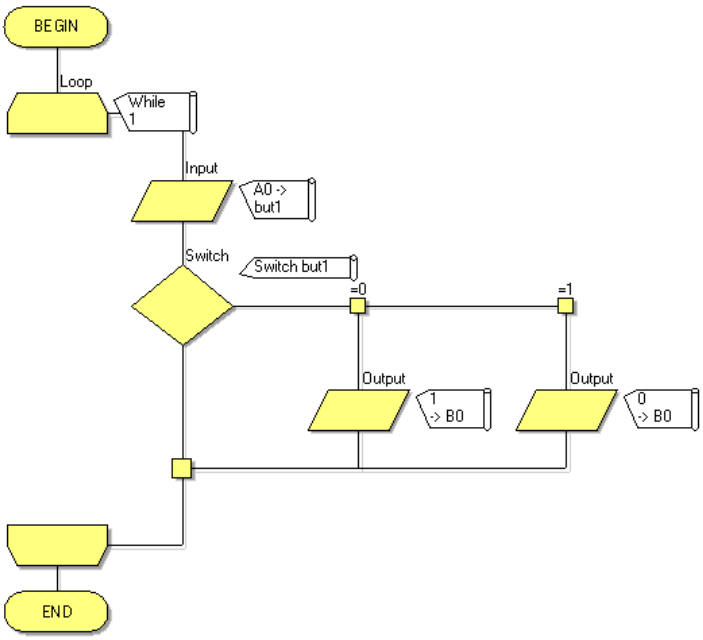


Рис. 7.4. Простейший пример использования переключателя

Переменная but1 принимает только два значения: кнопка нажата 0, кнопка не нажата 1. Эти два случая и отображает переключатель, в ветвях которого записано, что должен делать выход B0 в каждой из этих ситуаций. Но часто переменная может принимать ряд значений, их проще описать с помощью конструкции Switch, чем записывать с помощью обычной конструкции ветвления.

Чтобы добавить необходимое в свойства переключателя, используется обычный диалог, который появляется после двойного щелчка мышкой по компоненту.

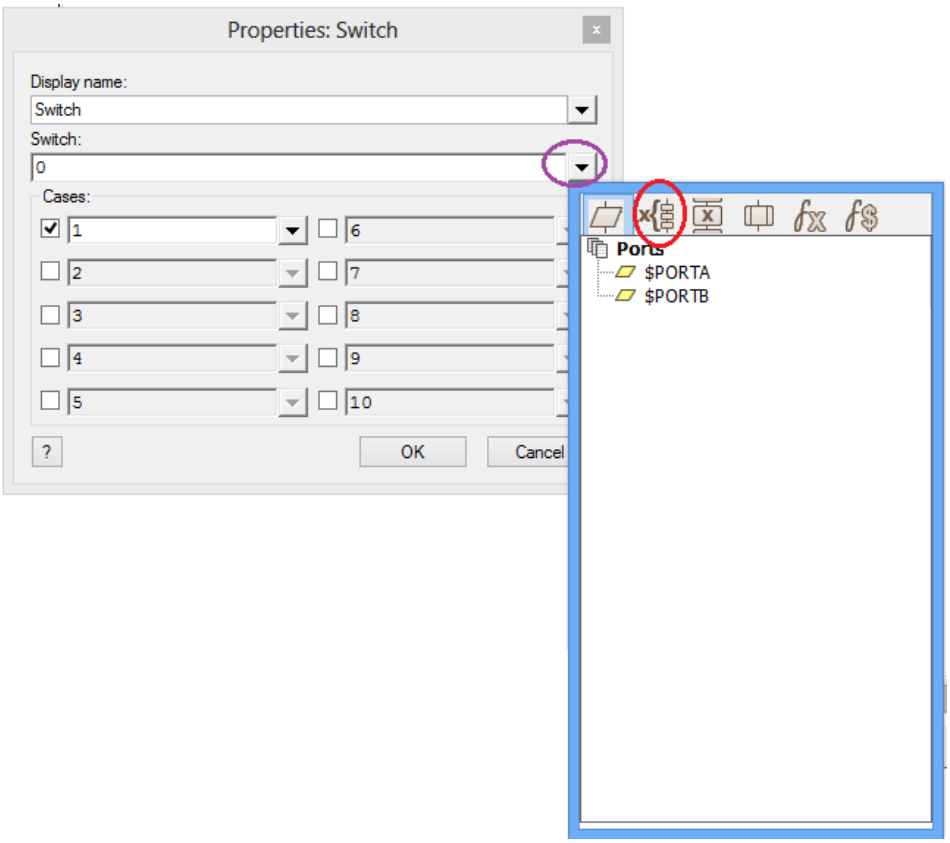


Рис. 7.5. Диалоговое окно описания переключателя

Если нажать кнопку справа от текстового поля Switch:, то открывается окно навигации, но в разделе выбора порта. Отмеченная на рисунке закладка откроет список переменных. Переменную можно и просто впечатать, удалив ноль, который записан по умолчанию. Ниже в разделе Cases (случаи) вы можете выбрать, отметив их галочкой, нужные вам переключения, записав значения, принимаемые переменной, вместо тех, что заданы по умолчанию.

Представьте, что к порту А подключена не кнопка, а к двум входам подключены два пожарных датчика. Когда срабатывает датчик, на пульте в охране высвечивается индикаторный светодиод, отмечающий номер помещения. Даже в этом случае использование переключателя в программе даёт более удобную для чтения картинку. И, к вопросу о применении микроконтроллера в простых случаях, представьте, что высвечивать следует не только номер помещения в охране, но и план эвакуации. Если срабатывает один датчик, план эвакуации один, другой датчик, другой план эвакуации. Всё это легко собрать без использования микроконтроллера. Но, если сработали и первый и второй датчики, план эвакуации меняется. С помощью переключателя в программе вы легко реализуете и такой вариант. Попробуйте набросать схему без микроконтроллера. И этот же вариант для нескольких датчиков. Надеюсь, вас убедит такой простейший пример и в удобстве использования микроконтроллера, и в удобстве использования программного компонента Switch.

Connection Point (точка соединения)

Следом за переключателем на инструментальной панели программных компонентов появляется элемент, который существует только в паре со следующим за ним. Подсказки (при наведении курсора мышки) поясняют их применение.

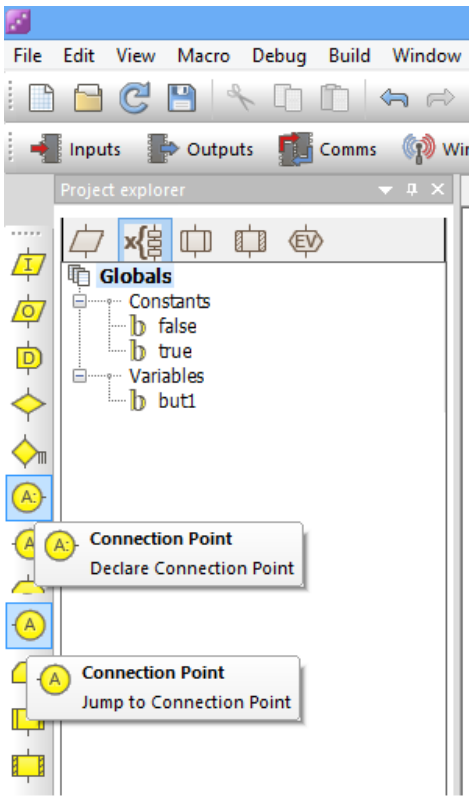


Рис. 7.6. Точки соединения

Первая точка – это объявление точки соединения, вторая добавляет переход к точке соединения. Вот пример использования точек соединения из примеров Flowcode.

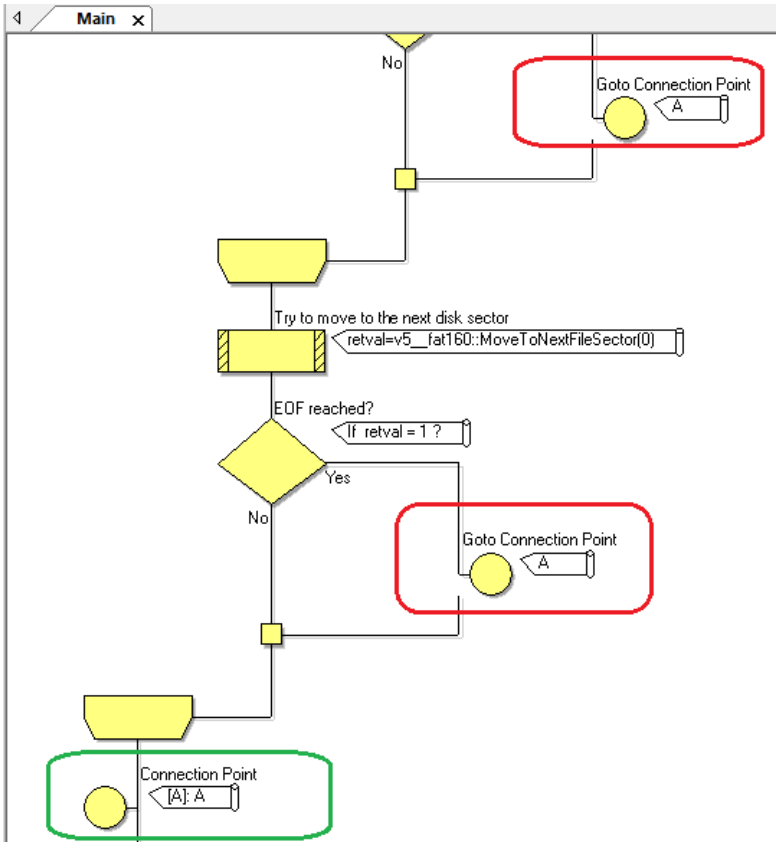


Рис. 7.7. Использование компонентов точки соединения

Внизу использована точка объявления, а выше две точки указания перехода к точке объявления. Хотя считается дурным тоном в программировании использование оператора Goto, но иногда это приходится делать. Обычно в программе создаётся метка, на которую и указывает оператор Goto. Вы можете видеть, что выше использована эта конструкция.

Loop (цикл)

Мы уже использовали цикл, и в одной из предыдущих глав рассказывалось, зачем используется бесконечный цикл.

Любая форма цикла позволяет быстро записать повторяющиеся операции. Когда-то, на заре эры цифровых компьютеров и программирования, вероятно, программы записывались только в виде линейных конструкций, операции следовали одна за другой от начала работы программы до её завершения. Но очень часто операции выглядят так:

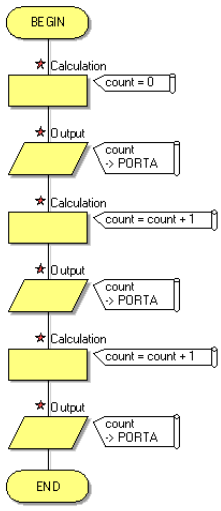


Рис. 7.8. Пример повторяющихся однотипных операций

В этом примере переменная count каждый раз увеличивается на единицу, и значение переменной выводится в порт А. Если эту операцию следует проделать только 10 раз, вы согласитесь, что и программа будет выглядеть безобразно, и вам надоест её писать, пусть даже вы используете такой удобный приём, существующий в любом редакторе, как копирование с последующей вставкой.

Для случая заведомо известного количества повторов однотипных фрагментов программы удобно использовать счётный цикл. Вот, как настраивается этот цикл в Flowcode.

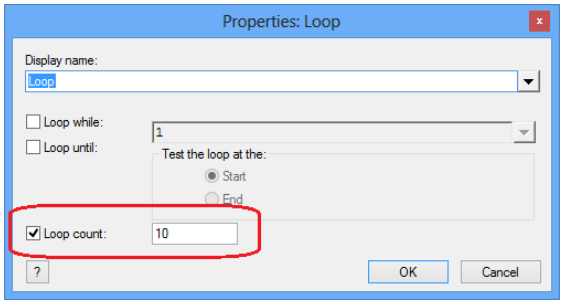


Рис. 7.9. Диалоговое окно настройки цикла

С использованием счётного цикла программа выглядит значительно аккуратнее.

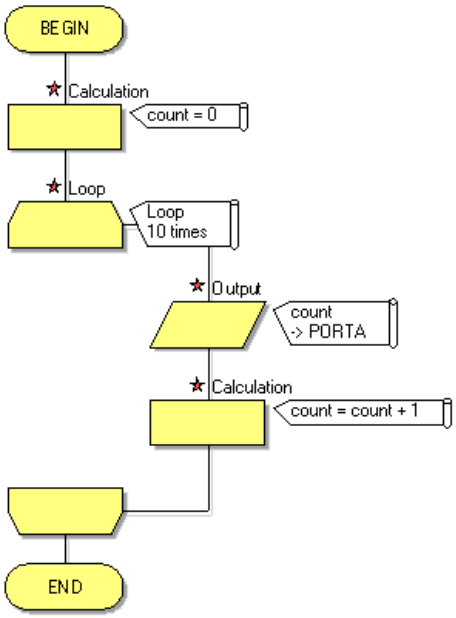


Рис. 7.10. Вид программы при использовании счётного цикла

Использование бесконечного цикла в форме «пока (что-то)» для работы микроконтроллера связано не столько с удобствами программирования, сколько мера вынужденная. Альтернативой могло бы быть использование оператора Goto (точек перехода). Иногда использование цикла в этой форме позволяет избежать каких-либо изменений до появления некоторого события. Например, нам нужно в программе ожидать изменения входа, к которому подключен пожарный датчик, а когда это произойдёт, включить тревогу. После завершения работ по устранению угрозы пожара систему можно перезапустить. Для этого можно использовать ранее созданную программу, но можно видоизменить её.

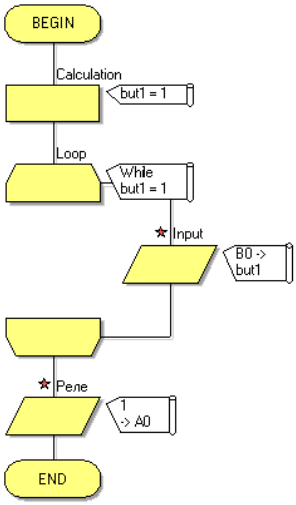


Рис. 7.11. Модификация программы с кнопкой

Измените в настройках цикла проверку (не в начале цикла, а в конце), используйте пошаговое прохождение программы, и вы увидите разницу. Попробуйте изменить цикл по условию Loop until (выполнять цикл до тех пор, пока), используйте пошаговое прохождение программы,

видоизмените программу, чтобы увидеть разницу в этих формах цикла. И, есть одно, что не вполне очевидно – при задании свойств выключателя проверьте активное состояние, высокое или низкое, наблюдая за переменной в окне наблюдения.

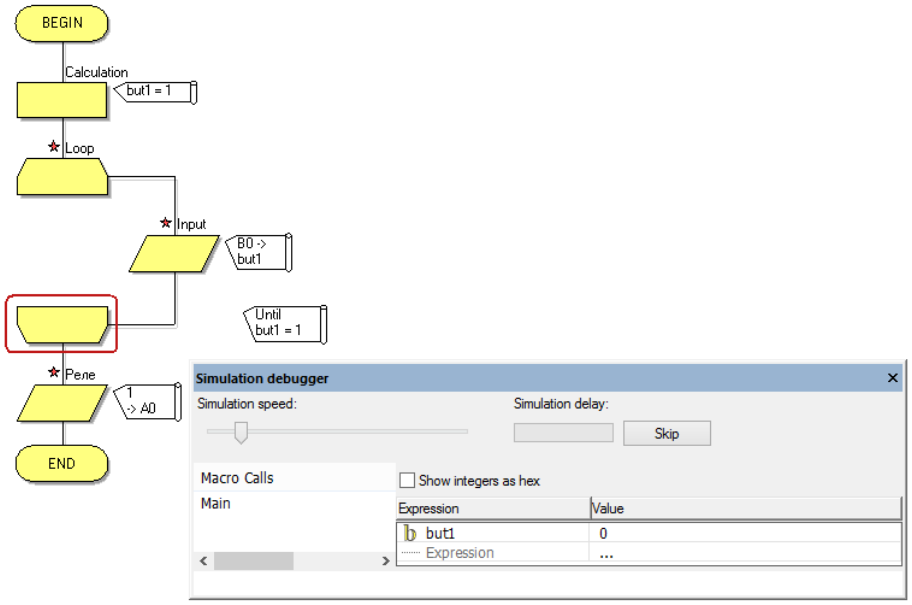


Рис. 7.12. Наблюдение за переменной в окне наблюдения

Чтобы добавить переменную в окно наблюдения достаточно щёлкнуть правой клавишей мышки в текстовом поле и выбрать из выпадающего списка переменную. Я, например, не совсем ясно понял смысл активного состояния выключателя, надеюсь, вам повезёт больше. Дело в том, что порт B микроконтроллера PIC16F628A позволяет подключить на вход подтягивающие резисторы, а порт A подтягивающих резисторов не имеет. Подключение резисторов требует использования дополнительной команды, а в свойствах выключателя, как я их понимаю, должно указываться, каким должно быть соединение выключателя, например, с общим проводом или с плюсом питания (в этом случае подтягивающий резистор должен соединяться с общим проводом).

Впрочем, всегда можно добавить нужную команду с помощью вставки на языке Си.

Macro (макрос)

Следующий компонент на инструментальной панели программных иконок предназначен для работы с подпрограммами (или макросами).

Использование макросов приём удобный во многих отношениях. Любая программа по мере её создания, пишите ли вы её на языке Си, собираете ли её из графических элементов, разрастается и быстро превращается в трудно воспринимаемую сущность. Поэтому подпрограмма, которая вызывается обращением к ней, существенно упрощает вид основной программы.

Изменим простую программу примера для использования макроса. Для этого обратимся к пункту Macro основного меню.

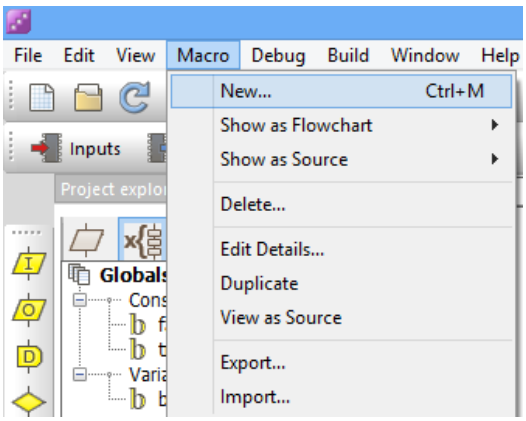


Рис. 7.13. Создание нового макроса

В диалоговом окне нужно задать имя макроса, хорошо бы добавить его описание, и, если макрос должен возвращать переменную, указать её. Создаётся новый лист с двумя элементами Begin и End. В основном окне программы вы увидите закладку с новым листом. Добавив в подпрограмму необходимые компоненты, можно вызвать выполнение этой подпрограммы из основной программы:

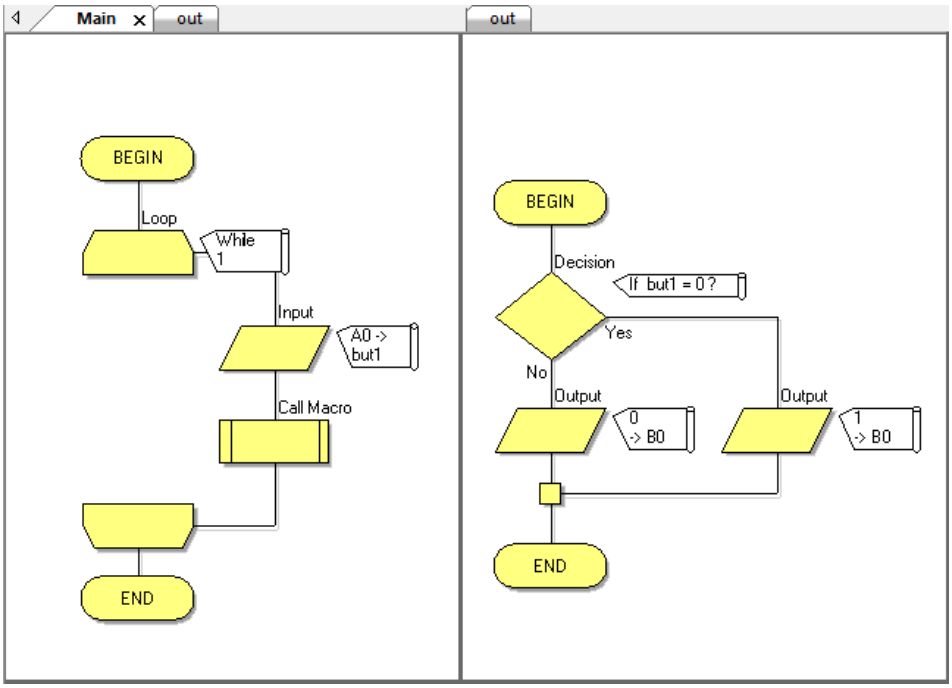


Рис. 7.14. Программа и подпрограмма

Откройте файл в примере из набора Flowcode: X-Y Plotter.fcfx. Эта программа использует большое количество макросов. Все закладки не помещаются в рабочем окне, и чтобы их просмотреть следует использовать стрелки на панели закладок.

Рассмотрите пример, чтобы решить для себя, использовать ли макросы, как и когда это делать. И ещё, я часто применяю слова подпрограмма, макросы, в сущности, не делая между ними разницы. Конечно, суть подпрограммы, процедуры или функции одинакова, но называют их по-разному, чтобы подчеркнуть их назначение. Так процедура ориентирована на выполнение неких действий, ей не нужно возвращать что-либо в программу, а функция выполняет вычисления и

возвращает в программу результат; макрос – это макрокоманда, состоящая из других команд. О точном значении этих терминов вы можете прочитать в книгах о программировании.

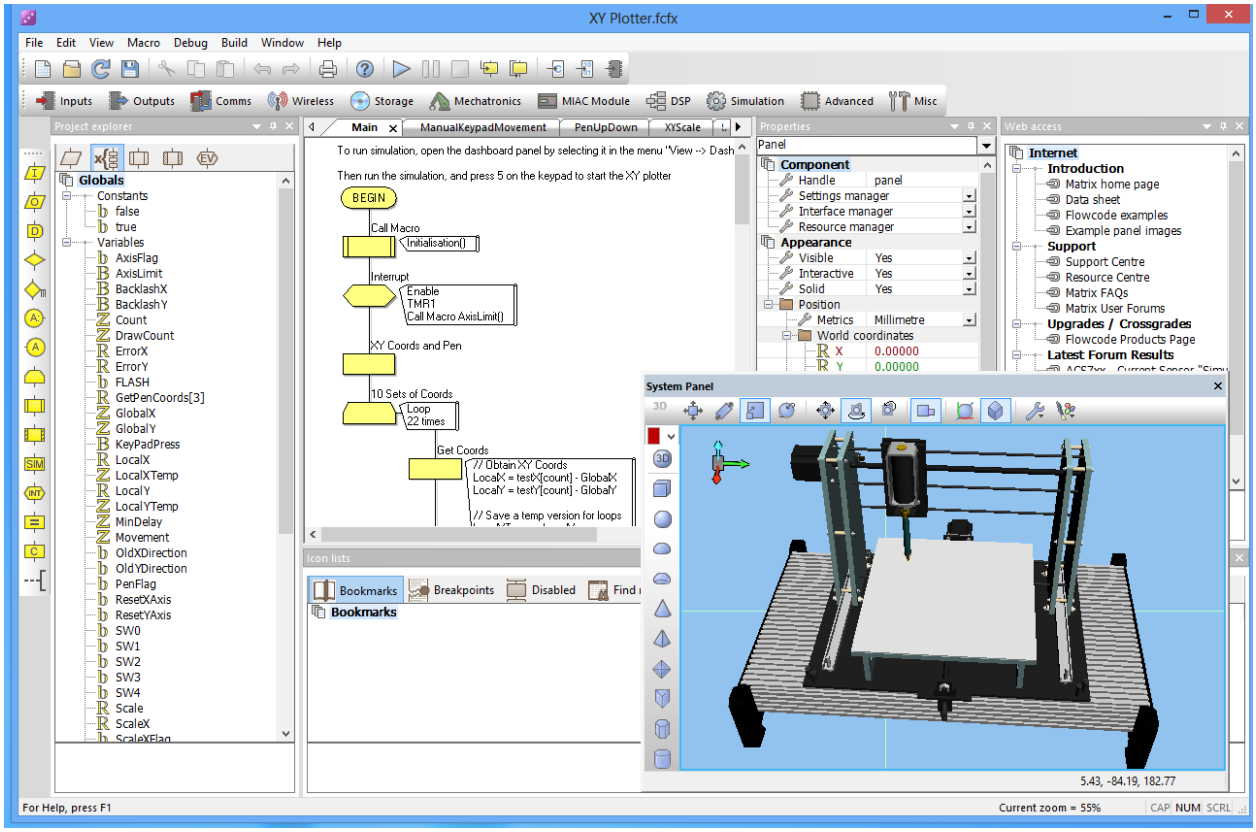


Рис. 7.15. Пример из набора Flowcode 6

Создавая программу, далеко не всегда получается вовремя определить, какую часть программы выделить в подпрограмму, пока программа не разрастётся достаточно, чтобы этот вопрос стал актуален. Попробуйте вырезать то, что можно перенести в макрос (создав новый макрос), и вставьте с помощью соответствующих команд выпадающего меню (щелчок правой клавишей мышки по выделенной области):

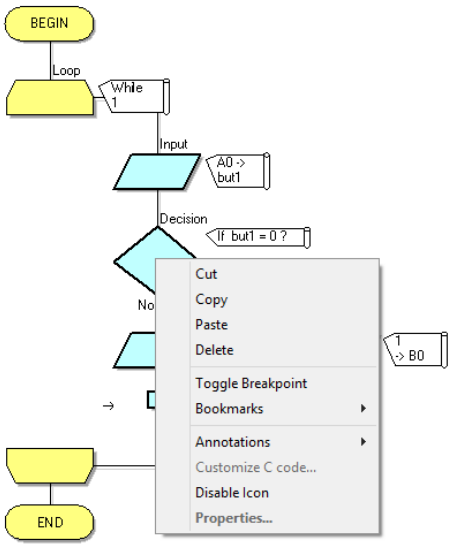


Рис. 7.16. Выпадающее меню для обработки выделенной области

Или используйте команды основного меню, или используйте кнопки инструментальной панели редактирования.

Рассказ о подпрограммах, я хочу завершить таким замечанием, что есть один случай, когда вопрос об использовании или нет подпрограммы не зависит от ваших предпочтений. Это использование прерывания. О прерываниях мы поговорим в следующей главе, но подпрограмму обработки прерывания вы вынуждены использовать.

И в завершение этой главы ещё одно замечание. Я считаю, что программа полезна начинающим для освоения работы с микроконтроллерами. В программе можно быстро понять назначение основных программных конструкций, создать и перенести на макет какое-нибудь простое устройство, использующее микроконтроллер. И посмотреть, как выглядят на языке Си и на ассемблере базовые операции. Но последнее зависит от компилятора. В новой версии простые конструкции языка очень привязаны к библиотеке компилятора. Вот пример:

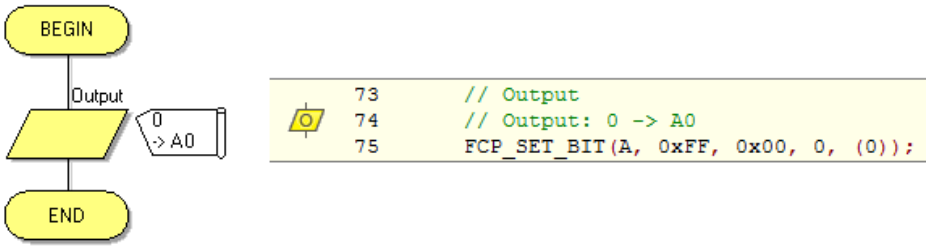


Рис. 7.17. Изменение состояния выхода в графике и на языке Си

Аналогичный фрагмент в ранних версиях выглядит иначе:

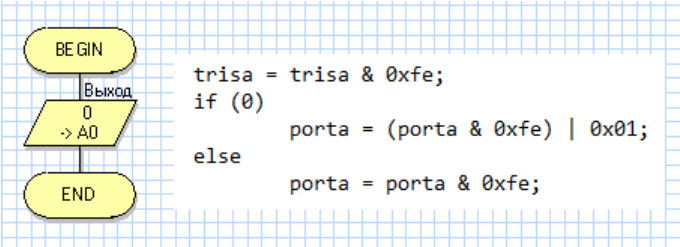


Рис. 7.18. Вид аналогичного фрагмента в ранних версиях программы

Такое написание фрагмента на языке Си будет понятно многим компиляторам (может быть, придётся изменить регистр или немного исправить написание), а фрагмент выше… не думаю, что будет полезен в другой среде разработки программы. Так что, начинающим я бы посоветовал не торопиться с переходом на новые версии программы Flowcode.

Глава 8. Панель элементов программы (Component Macro, Sim и т.д.)

Среда разработки программы тем ценнее, чем лучше её библиотека. Особенно это важно для среды разработки программы для микроконтроллеров. Чем больше библиотечных функций работы с внешними компонентами, тем легче создавать свою программу. В программе Flowcode каждый внешний элемент связан со своим макросом.

Component Macro

Если у вас на системной панели нет ни одного компонента, вы не сможете использовать компонентный макрос. Добавим в программу этот элемент и откроем его свойства двойным щелчком левой клавиши мышки.

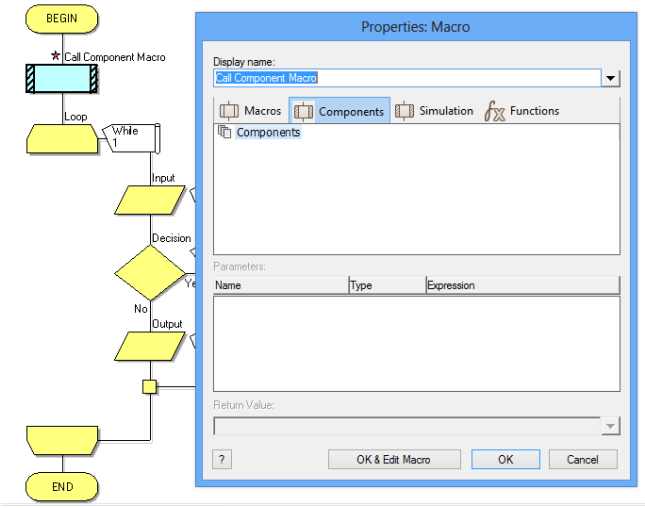


Рис. 8.1. Component Macro в отсутствии дополнительных компонентов

Добавим на системную панель выключатель и светодиод.

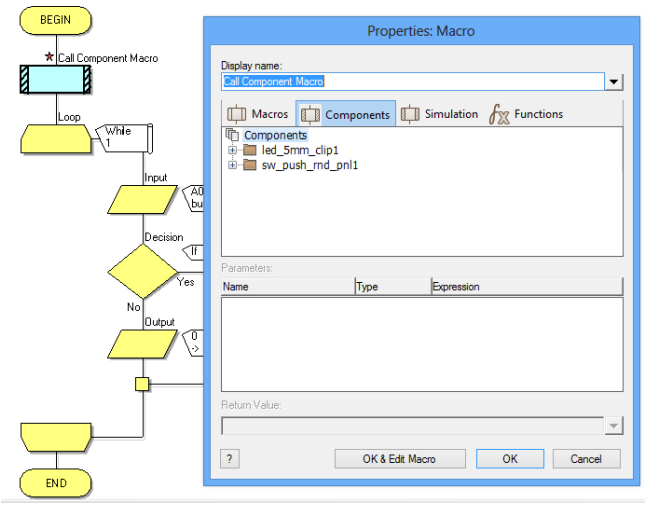


Рис. 8.2. Component Macro при наличии дополнительных компонентов

Каждый из этих компонентов приходит со своим набором команд и функций, которые можно использовать в макросе компонента.

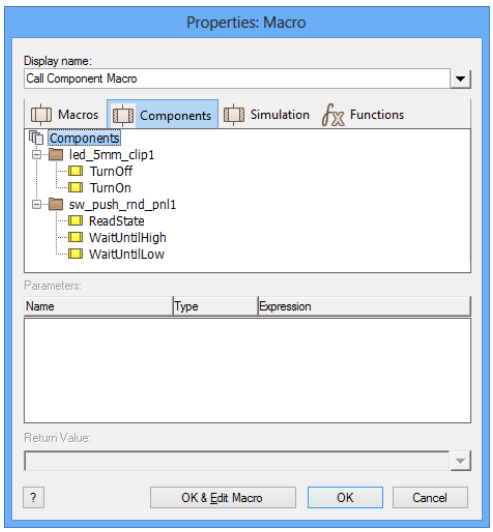


Рис. 8.3. Команды компонента светодиод и выключатель

Эти команды можно использовать с компонентным макросом. Позже мы подробнее остановимся на использовании некоторых функций компонентов, когда будем рассматривать панель дополнительных компонентов.

Simulation (симуляция)

Этот программный элемент появился только в шестой версии программы. Добавим его в программу:

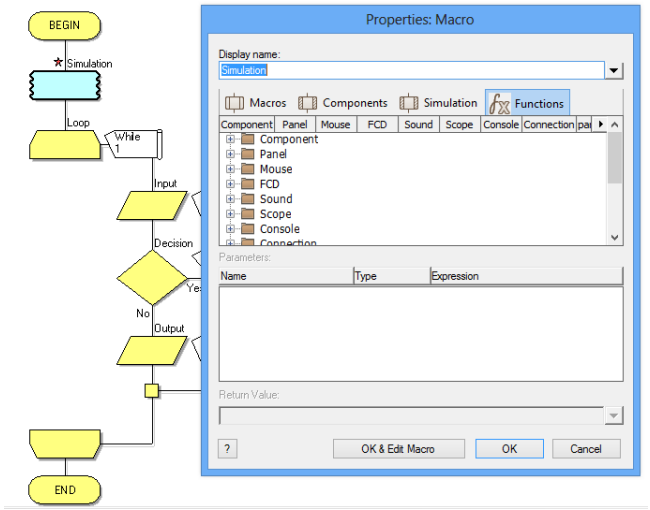
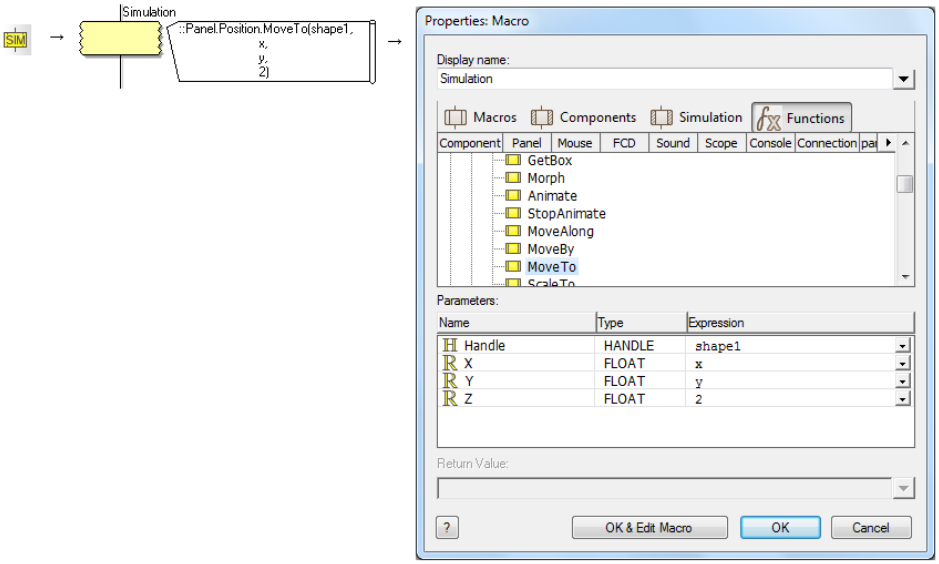


Рис. 8.4. Свойства программного элемента Sim

Длинный список раздела функций позволяет рассмотреть их, если нажать на значок с плюсом, открывающий те функции, что доступны для компонента, функции которого нас интересуют.

Обратимся вновь к описанию компонента Sim разработчиками программы:

Свойства иконки макроса Simulation



Макросы разделяются на Macros и Component/Simulation.

Component Macro (компонентные макросы) – это предопределённые макросы, связанные с компонентами. Например, макрос LCD используется для отображения чисел и букв на LCD дисплее.

Компонентные макросы доступны только для использования со специфическим компонентом…

Макросы Simulation предназначены только для симуляции компонентных макросов; обеспечивая аналогичные цели, они могут использоваться только при симуляции, и не могут работать с оборудованием…

Как и вызываемые только для симуляции компонентные макросы, Simulation Macros могут также поддерживаться Simulation Functions (функции симуляции), добавляющими расширенную программную симуляцию, хотя они и не загружают код, и не компилируются в микросхему. Events (события) используются для создания продвинутых цифровых программ и могут быть использованы для проверки концепций приложений, которые будут созданы позже с использованием аппаратных средств.

Interrupt (прерывание)

Прерывание предназначено для прерывания работы программы с целью выполнения операций, требующих только кратковременного внимания процессора, или операций, связанных с событиями, которые крайне важны и которые нельзя пропустить.

Если вернуться к простой программе работы с кнопкой в расширенном варианте, то есть, при использовании всех выводов порта с пожарными датчиками, то можно сказать, что основная работа программы – это постоянный опрос состояния датчиков. Но нам хотелось бы для дополнительного наблюдения передавать время от времени в сеть информацию о состоянии датчиков. Для этого мы используем прерывание по таймеру, а подпрограмма обработки прерывания будет передавать в сеть данные о состоянии датчиков.

Любой из микроконтроллеров обеспечивает разные типы прерываний. Добавим в программу прерывание и посмотрим свойства прерывания.

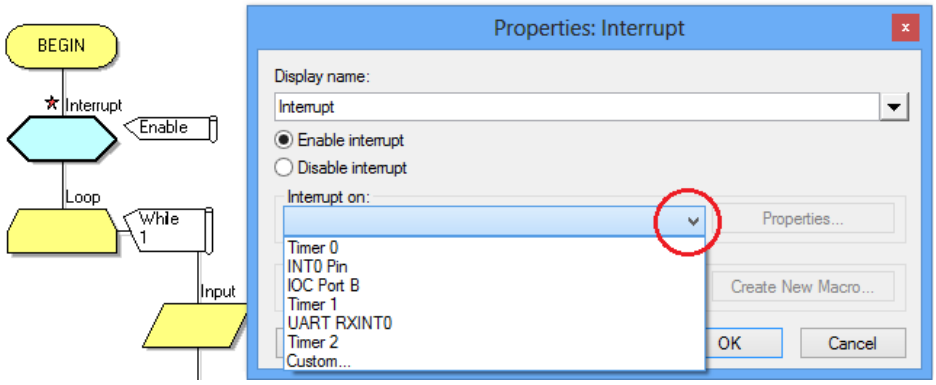


Рис. 8.6. Страница свойств прерывания

Компонент Interrupt разрешает прерывание, тип которого вы выбираете из списка в разделе Interrupt on. В приведённом примере мы предполагали использовать прерывание по таймеру, положим, Timer 1. Мы знаем, что должны выполнить при возникновении прерывания. Поэтому заготовили подпрограмму обработки прерывания, которую назвали send. Эту подпрограмму мы укажем при настройке свойств прерывания.

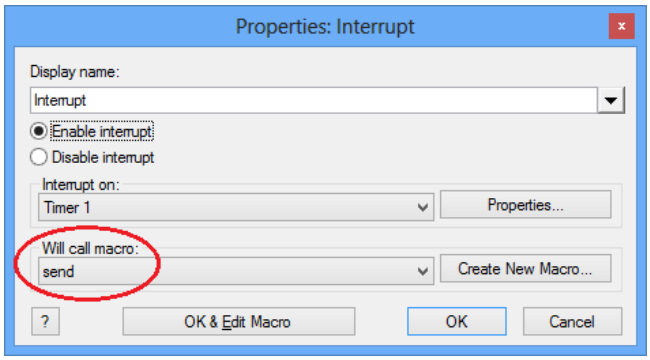


Рис. 8.7. Выбор типа прерывания и указание подпрограммы обработки

Нам остаётся только обратиться к свойствам таймера, чтобы настроить его. Для этого служит кнопка Properties.

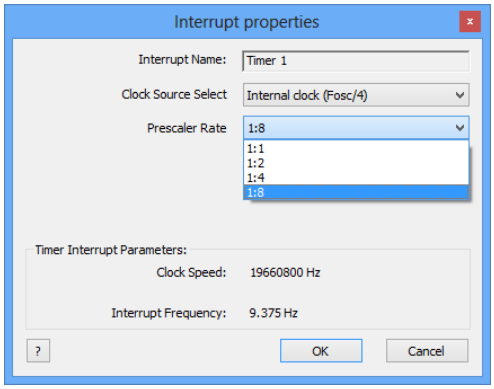


Рис. 8.8. Настройка таймера

На странице настройки таймера видно, что таймер будет использовать (можно выбрать и другой вариант) внутренний тактовый генератор и предделитель. В этом случае частота прерываний при тактовой частоте 19,6 МГц окажется равной 9,3 Гц.

В подпрограмме обработки прерывания мы отправим в сеть информацию о состоянии и перезагрузим таймер, разрешив прерывание.

Прерывание по изменению состояния порта будет полезно тогда, когда мы загрузим процессор другой работой, а за состоянием датчиков будет следить это прерывание. Срабатывание пожарного датчика изменит состояние порта, а возникшее прерывание в подпрограмме обработки произведёт все необходимые операции по оповещению об опасности пожара.

Посмотрим, как превратить нашу простейшую программу (с кнопкой) в аналогичную, но передающую сообщение в сеть при срабатывании кнопки.

Используем прерывание по изменению состояния порта. Вот программа и настройка прерывания.

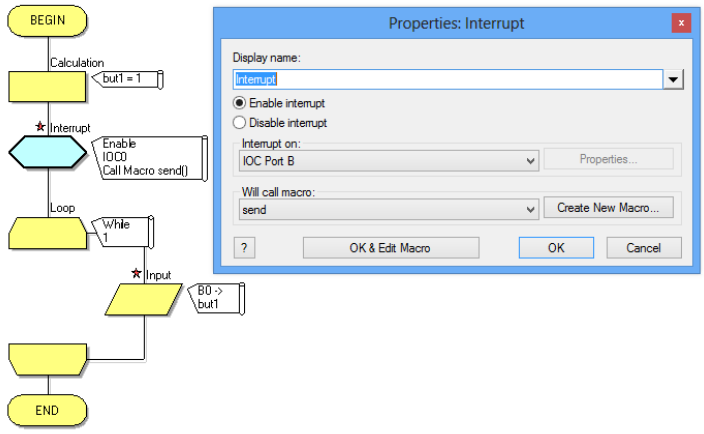


Рис. 8.9. Настройка прерывания по изменению порта

Для общения с сетью мы используем дополнительный компонент RS232, который можно найти на инструментальной панели дополнительных компонентов в разделе Comms.

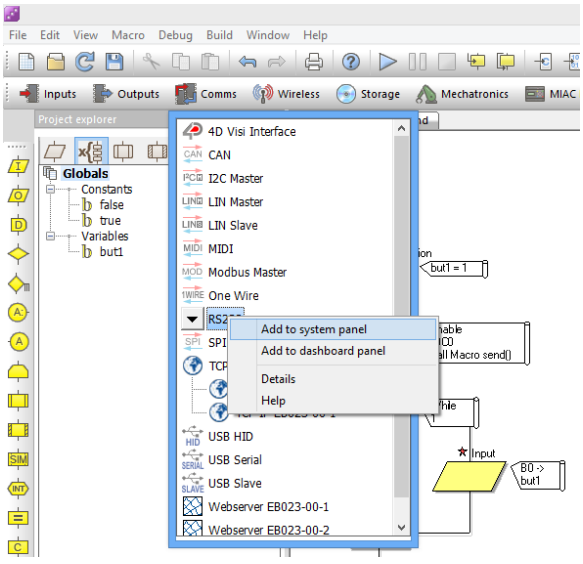


Рис. 8.10. Компонент для работы с модулем USART

Компонент мы добавим на системную панель. После этого можно добавить в начальную часть программы, которую назовём «Инициализация», компонентный макрос.

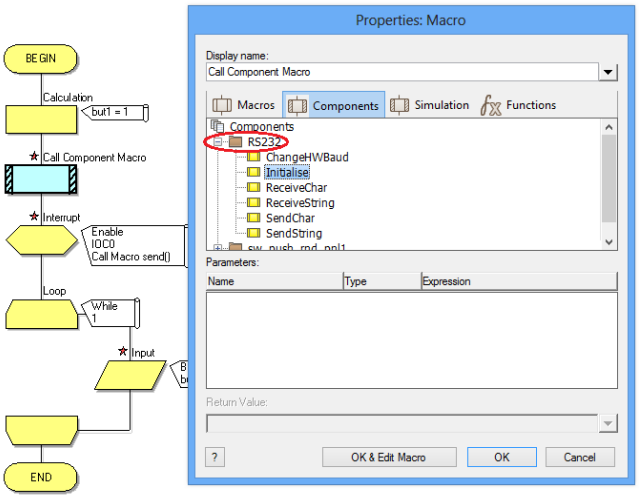


Рис. 8.11. Добавление макроса RS232

Следовало бы задать скорость передачи данных, как это положено для интерфейса RS232. Проблема в том, что в свойстве ChangeHWBaud предлагается использовать байт. Как в байт вложить скорость, скажем, 9600… Однако есть подсказка:

> *ChangeHWBaud*
> *Changes the hardware UART baud rate allowing for dynamic speed changes.*
> *Parameters*
> *BYTE NewBaud 0=1200, 1=2400, 2=4800, 3=9600, 4=19200, 5=38400, 6=57600, 7=115200*

Теперь понятно, что в качестве параметра следует использовать 3.

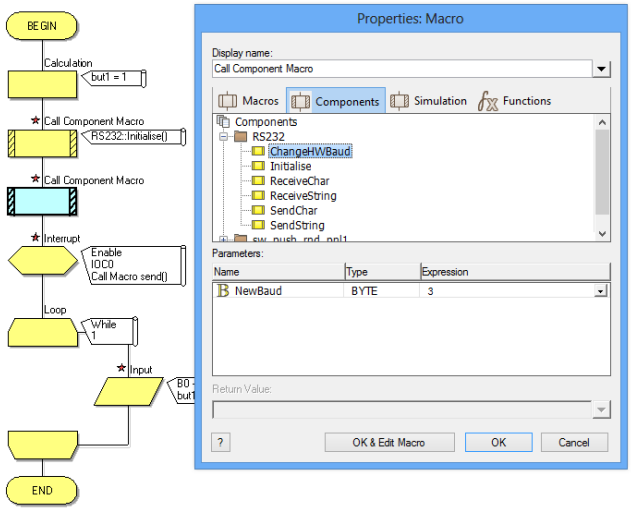


Рис. 8.12. Задание скорости передачи данных

Осталось дописать подпрограмму обработки прерывания. Для этого используется команда макроса, который добавлен в подпрограмму. При возникновении прерывания в сеть отправится символ «Y». В реальной программе можно отправить, например, переменную, в которой записано состояние порта.

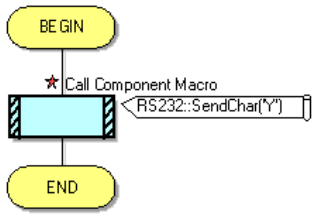


Рис. 8.13. Подпрограмма обработки прерывания

Есть ещё одно, что вам нужно сделать – указать подключение выводов. Программа Flowcode работает с разными микроконтроллерами. Некоторые из них имеют не единственный модуль USART. Кроме того вы можете самостоятельно использовать выводы микросхемы для этой цели.

Чтобы правильно подключить USART, достаточно в свойствах RS232 выбрать первый канал, программа знает, какие выводы ваша микросхема использует.

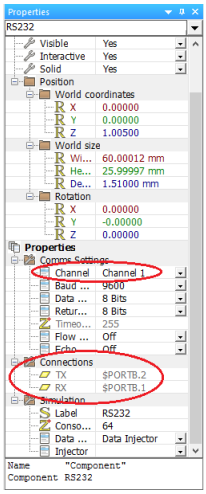


Рис. 8.14. Задание подключения модуля

Теперь можно перейти к проверке работы программы. Выводиться информация будет в консоль, поэтому следует позаботиться о том, чтобы консоль появилась в основном окне программы:

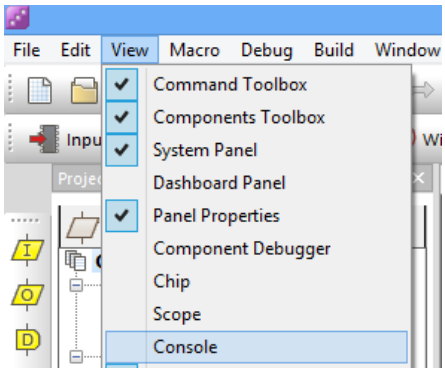


Рис. 8.15. Окно консоли

При каждом нажатии на кнопку должно возникать прерывание, а в консоли появляться символ «Y». Не забудьте только, что у PIC16F628A прерывание по изменению состояния порта B обрабатывается для выводов RB4-RB7. Кнопку следует подключить к выводу, скажем, RB4.

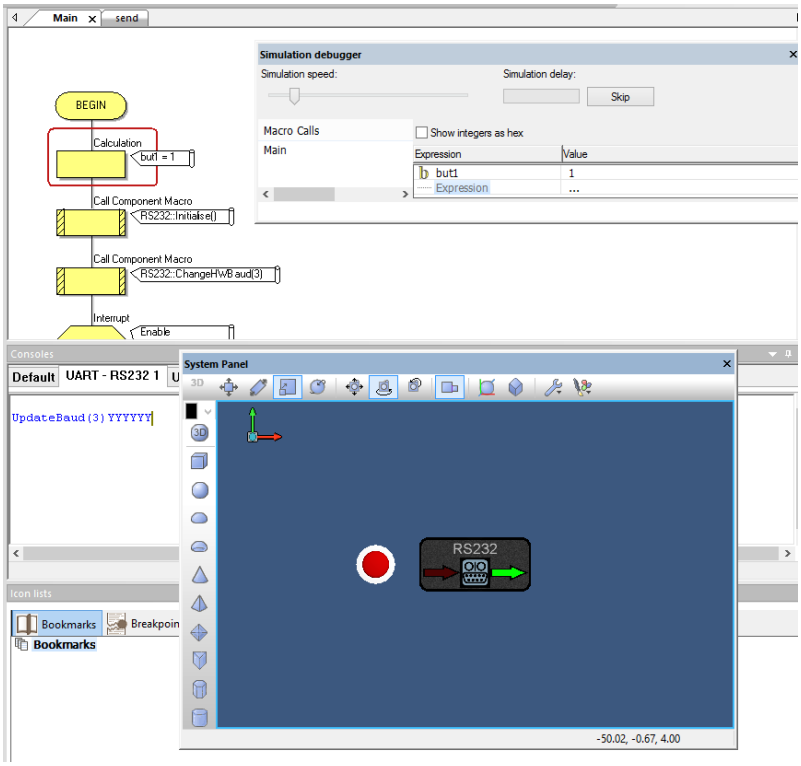


Рис. 8.16. Проверка работы программы

Calculation

Этот программный компонент предназначен для вычислений. Самое простое применение компонента – это инициализация переменной, присвоение начального значения. Но компонент поддерживается многочисленными функциями, обеспечивающими, практически, вычисления любой сложности. Откройте свойства компонента, чтобы посмотреть доступные функции:

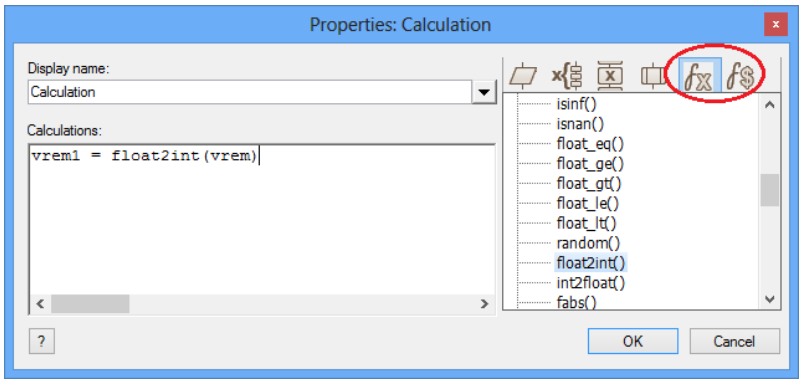


Рис. 8.17. Функции в разделе свойств компонента Calculation

C Code (блок текста на языке Си)

Если по каким-то причинам вам недостаточно возможностей графического языка, вы можете добавить фрагмент на языке Си. Например, микроконтроллер PIC16F628A для порта B может использовать на цифровых входах подтягивающие резисторы. Добавив кнопку в устройство, разумно использовать эти встроенные в микроконтроллер резисторы. Но для этого необходимо сделать запись в один из регистров. Сделает ли это Flowcode, уверенности нет, поэтому можно добавить простую запись на языке Си, включив подтягивающие резисторы.

Кроме того, если у вас есть отлаженный фрагмент программы на языке Си, если вам не хочется тратить время на создание его аналога на графическом языке Flowcode, вы можете вставить этот фрагмент программы, используя компонент C Code.

Выше об этом упоминалось, но напомню ещё раз: все переменные и функции на языке Си после трансляции графического исходного текста приобретают префиксы. Таким образом, вам придётся поправить вставленный текст на языке Си. Вот пример, в программе есть переменная vrem. Добавим компонент вставки кода Си и запишем в него простое присваивание:

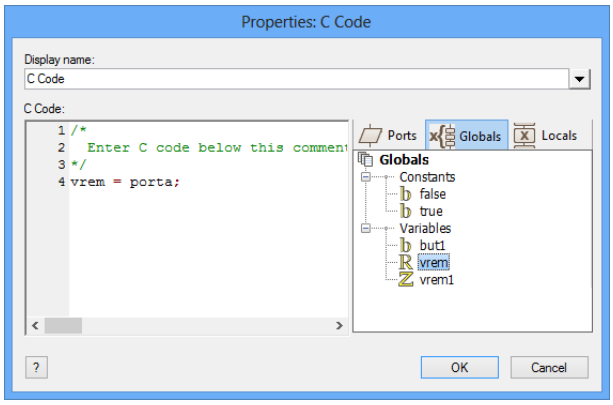


Рис. 8.18. Добавление текста на Си

Если теперь оттранслировать текст программы, то будет получено следующее сообщение:

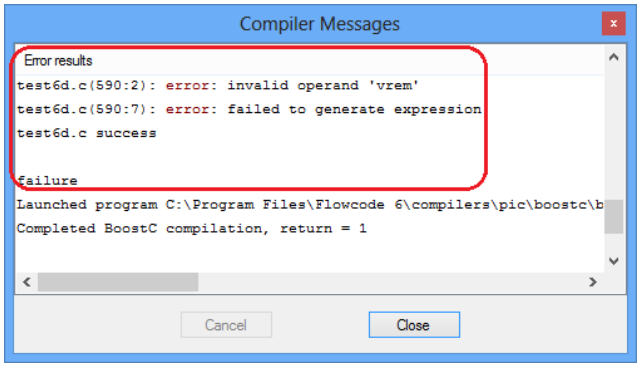


Рис. 8.19. Сообщение компилятора

Мало того, это же сообщение вы будете получать до тех пор, пока не исправите написание переменной в верхнем регистре. Отлаживая программу, вы можете видеть её в графике, но можете использовать текст на языке Си после компиляции. Достаточно указать это:

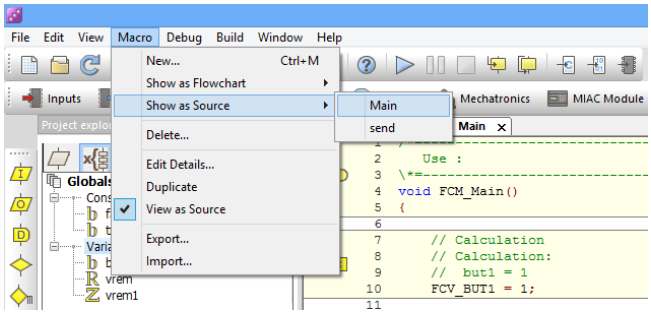


Рис. 8.20. Вид основной программы на языке Си

Comment (комментарий)

Даже в том случае, когда вы пишите программу для себя, наилучшее решение – сопровождать программу документацией. Самый простой вид документирования программы – использование примечаний. Этой цели служит Comment, компонент, который можно добавить в любом месте программы.

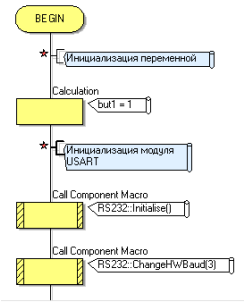


Рис. 8.21. Примечания к программе

К сожалению, комментарии кириллицей плохо воспринимаются программой. Если вы откроете текст программы на языке Си, то увидите следующее:

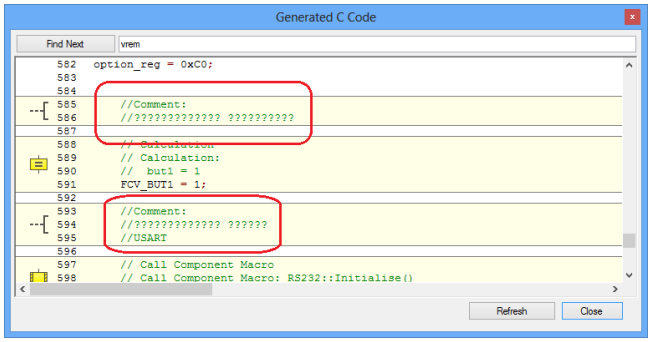


Рис. 8.22. Текст примечаний на языке Си

И обратите внимание, обозреватель кода на языке Си имеет механизм поиска (Find Next), которым удобно пользоваться при просмотре текста. Если вы будете активно пользоваться текстом на языке Си, пишите примечания латиницей. И пользуйтесь англоязычной версией программы, в русифицированной версии могут появиться значки вопроса вместо названия программных компонентов. Их, впрочем, можно исправить при сборке программы.

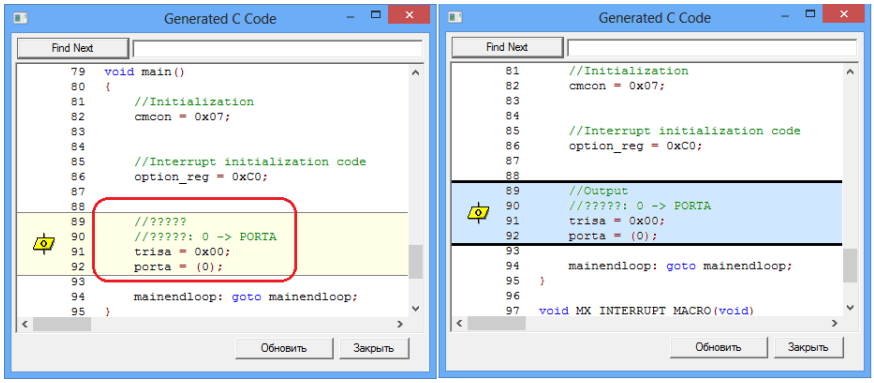


Рис. 8.23. Вид кода на языке Си русифицированной программы

Глава 9. Дополнительные компоненты (Inputs, Outputs, Comms)

Для этой главы я попытался придумать какое-либо осмысленное устройство, которое не использовало бы дополнительные компоненты электрической схемы, но только микросхему контроллера. Единственное, что приходит на ум – это генератор прямоугольных импульсов. Но такой генератор проще собрать на любой подходящей цифровой микросхеме, добавив пару резисторов и конденсатор, что будет гораздо дешевле.

Любое устройство на базе микроконтроллера использует внешние компоненты: датчики, кнопки на входе, светодиоды или дисплей на выходе. Кроме этих дополнительных компонентов сам микроконтроллер имеет встроенные модули АЦП, USART и т.д.

Для работы с этими дополнительными компонентами в программе Flowcode есть панель:

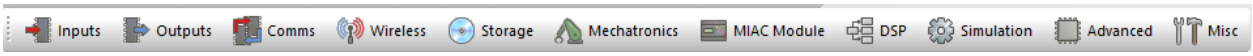


Рис. 9.1. Панель дополнительных компонентов

Все компоненты разбиты на группы. Рассмотрим их подробнее.

Inputs (входы)

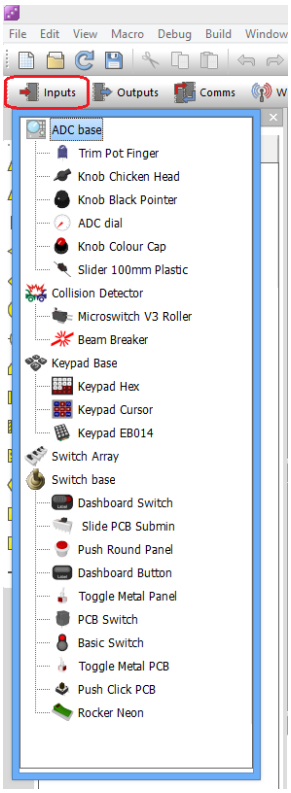


Рис. 9.2. Дополнительные компоненты для входов

Если вы видели предыдущие версии, то отметили, что и вид, и содержание этого списка изменились существенно.

Отчасти это, видимо, связано с появлением второй панели, панели управления. Часть дополнительных компонентов предназначено для размещения на этой панели.

Отчасти это, видимо, связано с изменившимся подходом к свойствам объектов. В предыдущих версиях в свойствах, например, кнопки вы могли изменить её вид.

И вспомним, что в этой версии появилась возможность трёхмерного вида дополнительных компонентов, и что существенно расширены возможности анимации.

Добавим на системную панель компонент ADC dial. Если микроконтроллер имеет встроенный модуль АЦП, то этот компонент, надо полагать, позволит с ним работать.

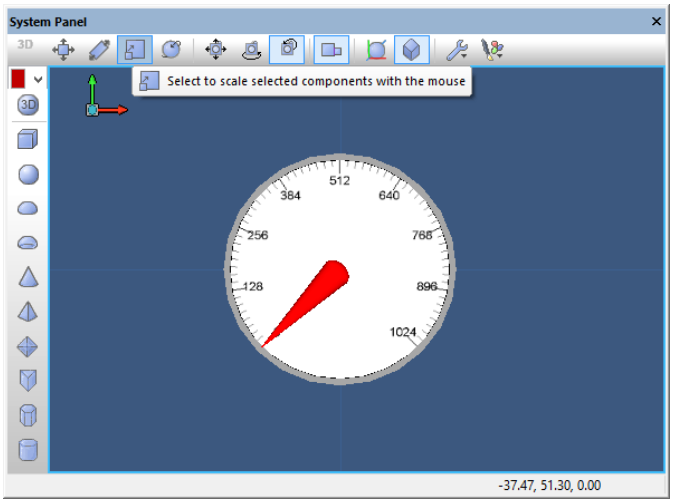


Рис. 9.3. Системная панель с компонентом шкалы АЦП

Если при появлении компонента он слишком мал, чтобы различить детали, то вы можете воспользоваться масштабированием с помощью мышки. Кнопка инструментальной панели отмечена на рисунке выше. Остальные кнопки на инструментальной панели позволяют изменить положение компонентов, придать им трёхмерный вид в удобном для наблюдения ракурсе и т.п.

Вторая панель (слева) позволяет добавить на панель разные фигуры и переключить вид этих фигур, трёхмерный или двухмерный. Панель при переключении меняет свой вид, предлагая соответствующие фигуры. В примерах Flowcode есть примеры анимации. Теперь вам понятно, как можно собрать на системной панели компоненты для подобной анимации.

Добавим в программу переменную и макрос:

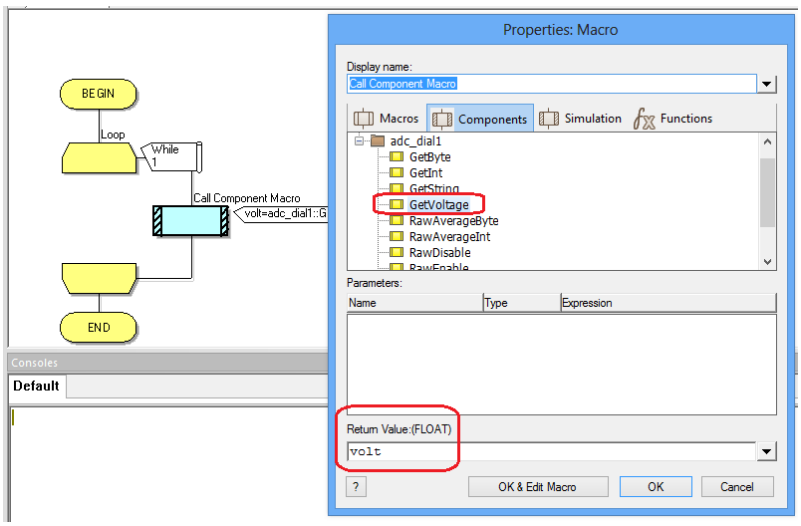


Рис. 9.4. Программа для работы с АЦП

Теперь можно запустить пошаговую проверку, с помощью мышки перевести стрелку на шкале, а в окне наблюдения (добавив переменную) наблюдать полученное значение.

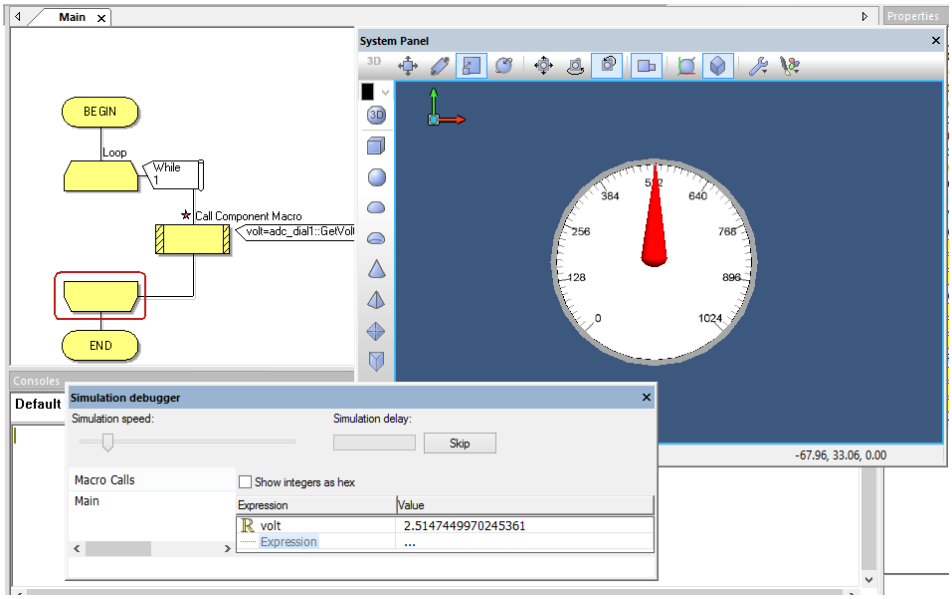


Рис. 9.5. Наблюдение за переменной при пошаговой проверке

Шкала, это видно на рисунке выше, градуирована не в вольтах. Но переменная отражает напряжение в десятичном виде.

Примерами датчиков могут служить два компонента: Microswitch и Beam Breaker.

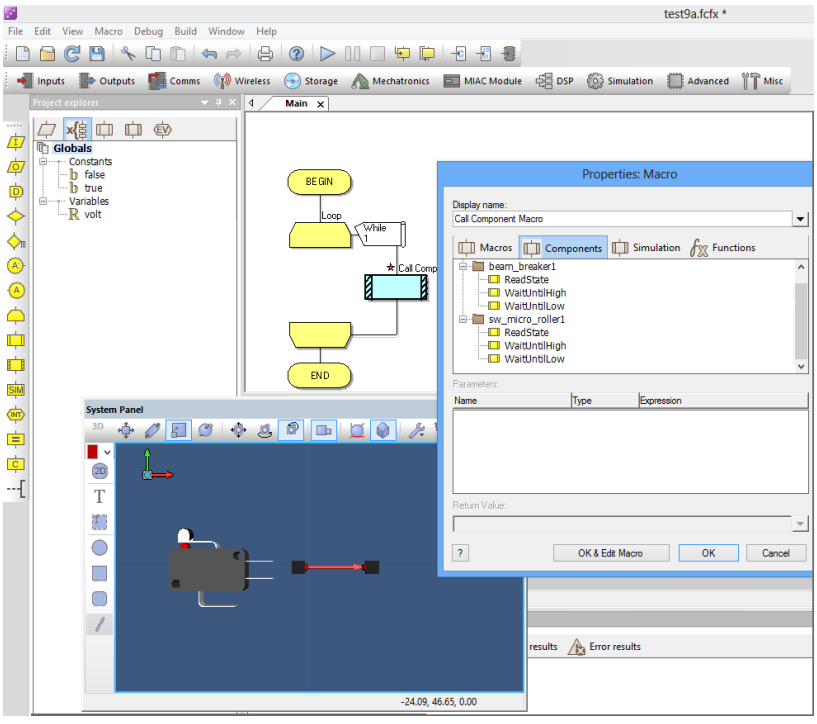


Рис. 9.6. Датчики и их свойства

Особую роль компонентов для входа играет клавиатура с рядом кнопок. Даже простое устройство, генератор прямоугольных импульсов, не устроит вас без настройки частоты и скважности импульсов. А в этом случае не обойтись одной или двумя кнопками. В разделе Inputs несколько видов клавиатуры. В окне свойств следует подключить клавиатуру к выводам микроконтроллера, а в свойствах макроса можно найти ряд удобных функций.

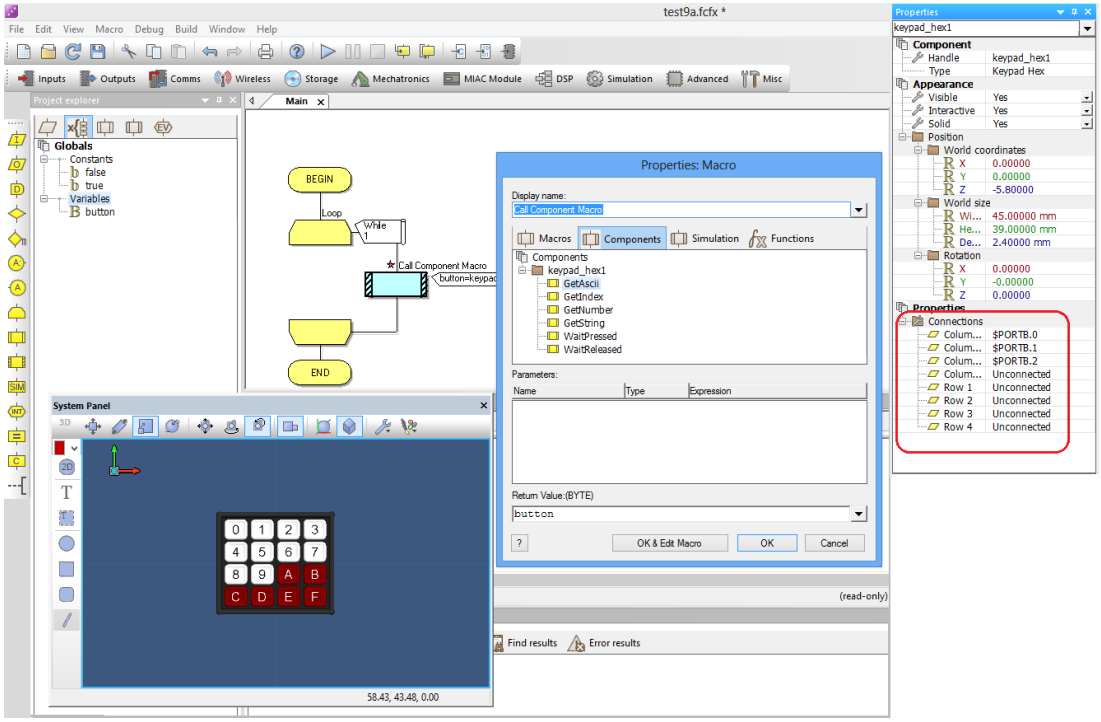
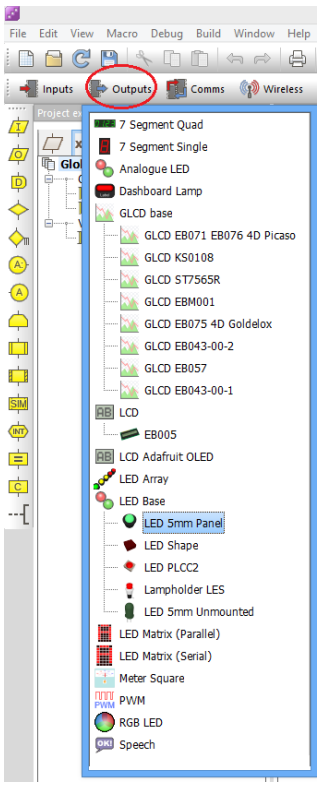


Рис. 9.7. Клавиатура, свойства макроса и подключение клавиатуры к выводам контроллера

Outputs (выходы)



Кроме исполняющих устройств, это могут быть реле или тиристоры, к выходу микроконтроллера часто подключают индикаторы.

От простого индикаторного светодиода до графических дисплеев в программе Flowcode представлены почти все, наиболее часто применяемые индикаторы.

Многие дисплеи работают на основе достаточно сложных протоколов обмена данными. Самостоятельно реализовать любой из подобных протоколов – задача интересная, но редко бывает целью разработки. Поэтому удобнее пользоваться готовыми макросами, встроенными в программу Flowcode.

Рис. 9.8. Список дополнительных компонентов, подключаемых к выходу

Вот некоторые из индикаторов, указанных в списке:

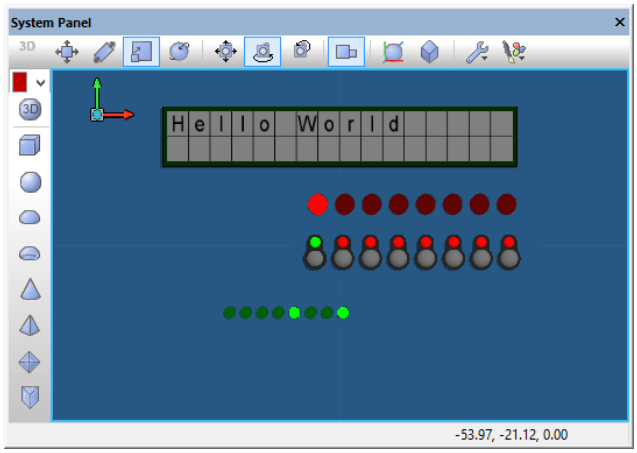


Рис. 9.9. Индикаторы для работы с Flowcode

Кроме индикаторов в этом разделе есть такой компонент, как модуль ШИМ, встраиваемый во многие микроконтроллеры. От регулировки напряжения, например, двигатель постоянного тока меняет скорость вращения, до синтеза звуков – модуль полезен в разных ситуациях. Среди примеров из набора Flowcode есть и пример работы с PWM.

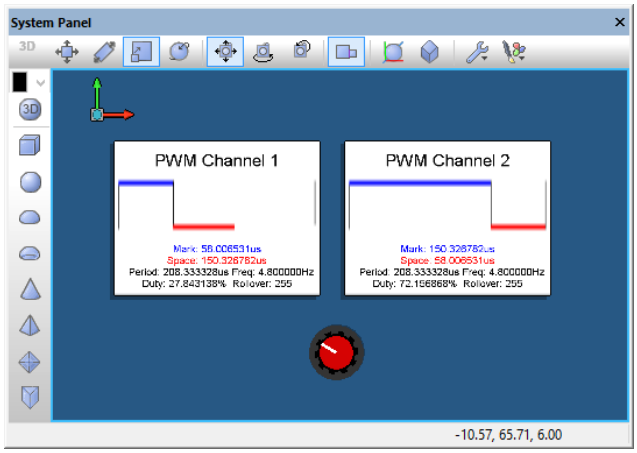


Рис. 9.10. Пример работы с PWM

Comms (коммуникация)

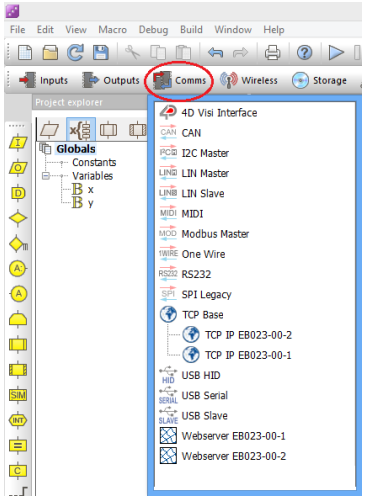
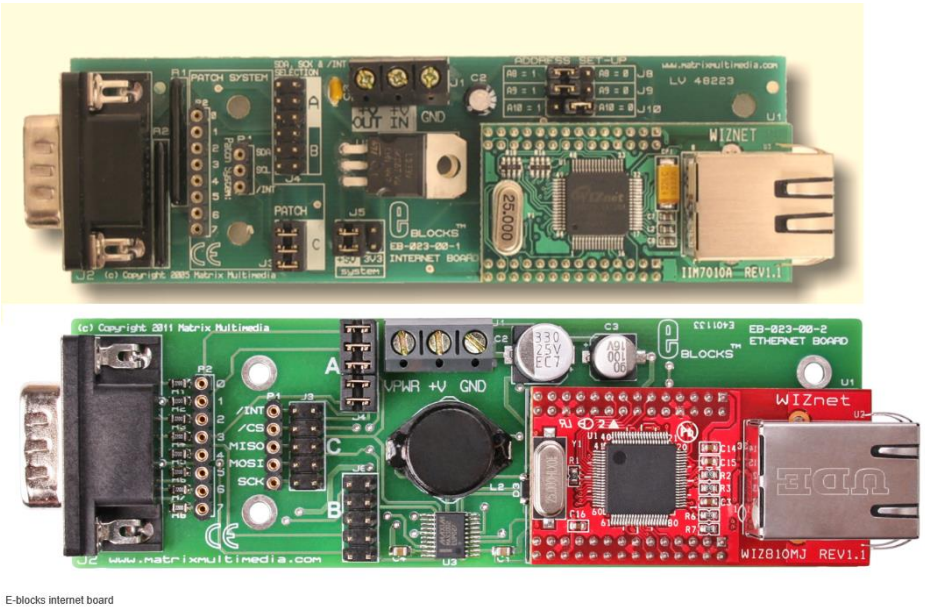


Рис. 9.11. Список компонентов проводной коммуникации

Об использовании интерфейса RS232 уже упоминалось выше. Это удобный протокол как для соединения микроконтроллеров в сеть, так и для подключения микроконтроллера к компьютеру.

Некоторые датчики температуры используют для передачи данных протокол One Wire или I2C. Начинающим для знакомства с этими протоколами лучше выбрать микроконтроллер, который поддерживает их на аппаратном уровне.

Ещё более сложные протоколы, как TCP или USB, конечно лучше изучать, используя все возможности, предоставляемые Matrix, и программу, и модули. Если есть возможность их приобрести.



E-blocks internet board

Рис. 9.12. Модули для работы с протоколом TCP

Опытные радиолюбители могут самостоятельно создать подобные модули. Мой давний знакомый Афанасьев Игорь Анатольевич, увлёкшись работой с микроконтроллерами, собрал сервер, а его товарищ написал программу для работы с этим сервером.

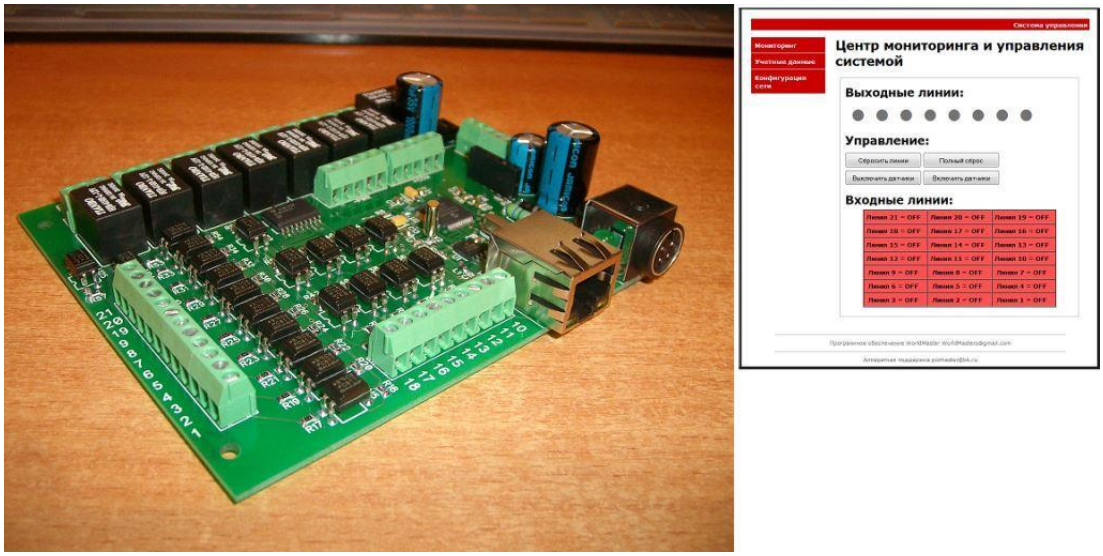


Рис. 9.13. Модуль сервера и страница управления

Теперь они работают над добавлением модуля WiFi, чтобы осуществить постоянное подключение модуля к Интернету, и как базу для управления роботом; модуля GSM для работы со смартфоном и планируют многое другое.

Но я считаю, что начинающим радиолюбителям не следует увлекаться слишком сложными протоколами, например, интерфейс CAN работает с кадрами и разрешением коллизии. Являясь промышленным стандартом, он требует определённого опыта и знаний, а его применение должно быть оправдано необходимостью. Сравните свойства двух макросов:

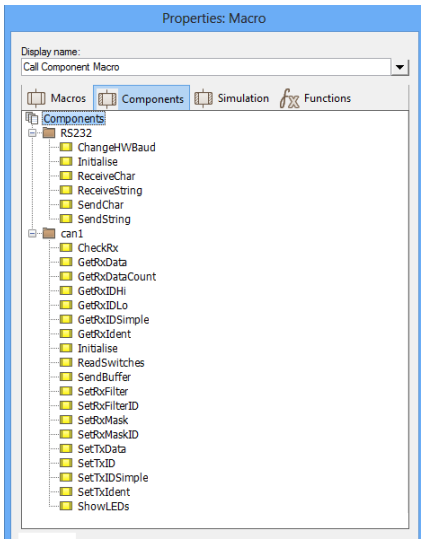


Рис. 9.14. Свойства макросов RS232 и CAN

С моей точки зрения самый удобный для первого знакомства протокол RS232. На небольших расстояниях, чтобы познакомиться с сетевым подключением микроконтроллеров, можно соединить два контроллера короткими проводами, создать две программы для каждого из них, и посмотреть, как это всё работает в «живом» виде. Например, проверить такие две программы:

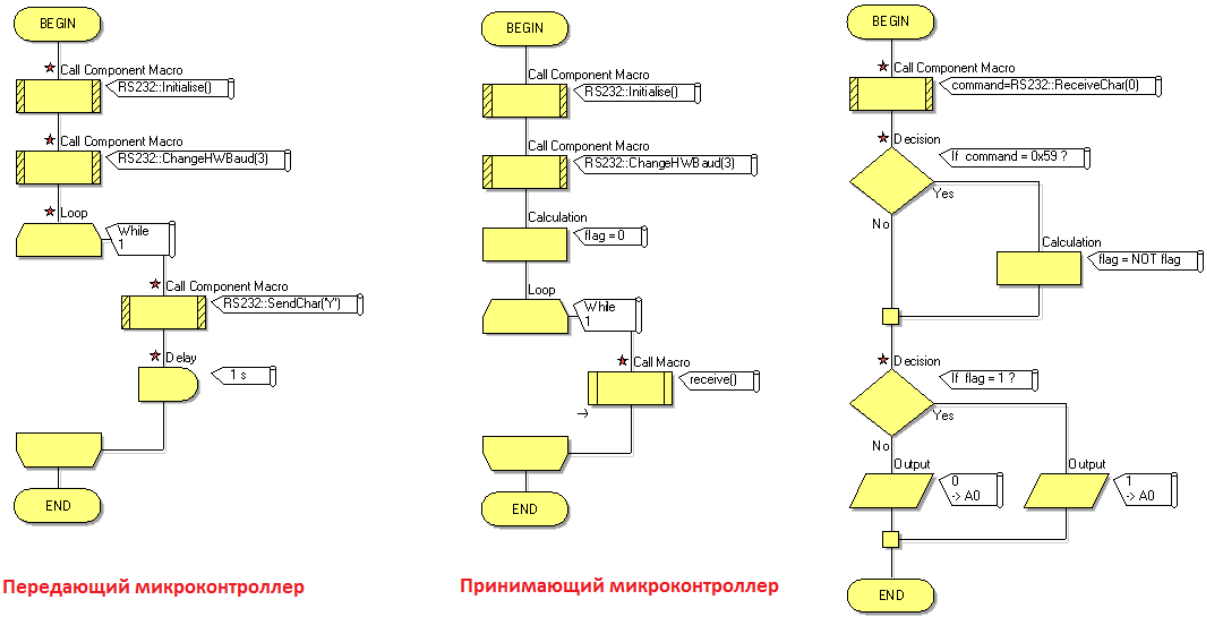


Рис. 9.15. Программы передачи команды и её приёма

Когда первый микроконтроллер передаёт команду, второй её принимает и отрабатывает, устанавливая в высокое состояние вывод RA0, а следующая (такая же) команда сбросит этот вывод. Пауза между отправкой команд заставит светодиод на выводе RA0 мигать. Программу легко можно модифицировать, отправляя запрос от первого микроконтроллера на состояние порта А второго микроконтроллера, а по состоянию порта менять команду. И можно придумать ещё много вариантов программы, которые позволят сделать задел для будущих разработок.

Если добавить несколько микросхем для перехода от RS232 к физическому интерфейсу RS485, то соединение потребует два провода, к которым вы сможете подключить несколько модулей, разнесённых на большое расстояние. И никто не запрещает предавать сигналы на частоте радиоуправления моделями там, где проводное соединение невозможно. При этом вся программная часть модулей сохраняется. Во всяком случае, подобные эксперименты с моей точки зрения принесут больше пользы в части освоения микроконтроллеров, чем знакомство начинающего со сложными протоколами из списка доступных в разделе Comms.

Глава 10. Дополнительные компоненты (Wireless, Storage и др.)

Wireless (беспроводные)

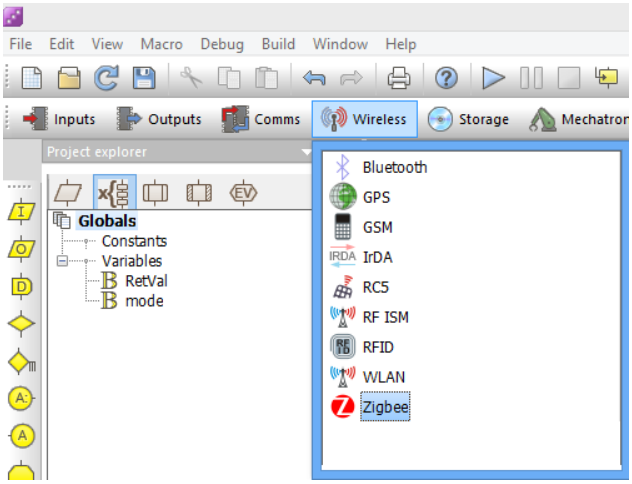


Рис. 10.1. Макросы поддержки беспроводного обмена данными

Программа Flowcode успешно используется в учебных заведениях, для которых приобретение сопутствующих модулей не составляет проблем.

Пример работы с модулем из списка беспроводной сети (RFID Get and Set buffer.fcfx) можно запустить на симуляцию:

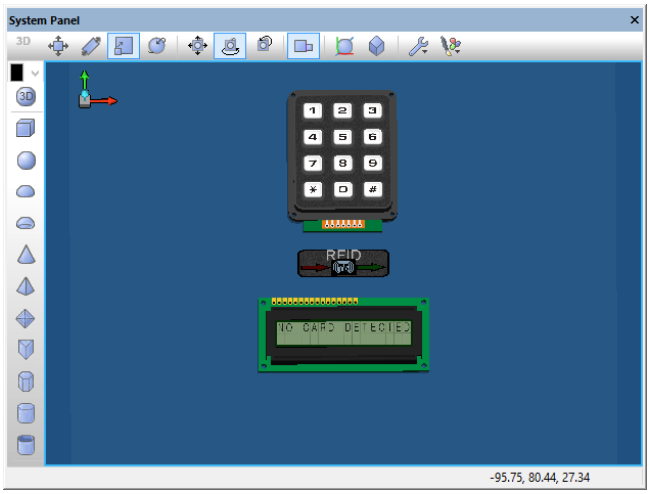


Рис. 10.2. Пример использования беспроводного обмена данными

Но надпись, которая появляется при запуске программы на симуляцию, гласит: модуль не обнаружен. Если посмотреть в свойствах компонента RFID раздел подключения, то можно увидеть:

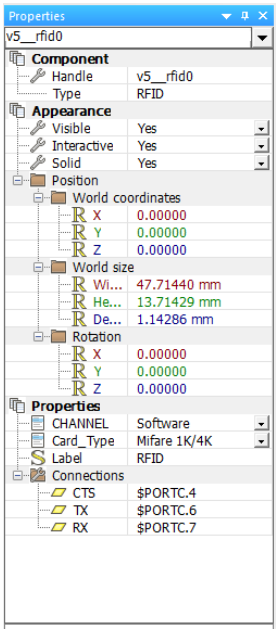


Рис. 10.3. Подключение компонента RFID

Хорошим экспериментом, мне кажется, было бы использовать второй микроконтроллер, который отвечал бы на запрос, имитируя работу компонента RFID, для которого есть готовый макрос.

Storage (устройства хранения)

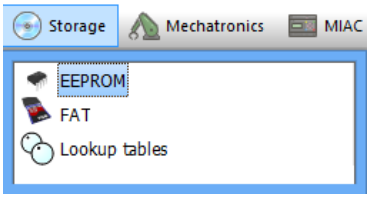


Рис. 10.4. Список макросов для устройств хранения

Первый макрос имеет две функции:

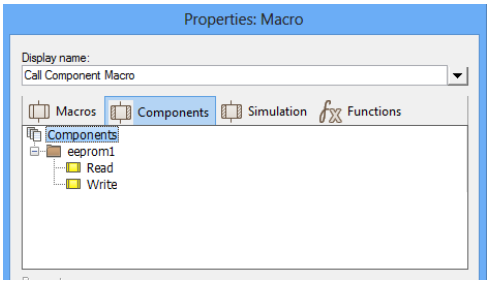


Рис. 10.5. Функции макроса EEPROM

То, что для EEPROM есть только операции чтения и записи, объясняется тем, что EEPROM, как правило, встроено в микроконтроллер для хранения разного рода констант или данных, которые должны оставаться доступными и после перезапуска микроконтроллера.

FAT (таблица расположения файлов), название макрос получил от таблицы, которой раньше снабжались только жёсткие диски, а теперь имеют флэшки и другие аналогичные носители.

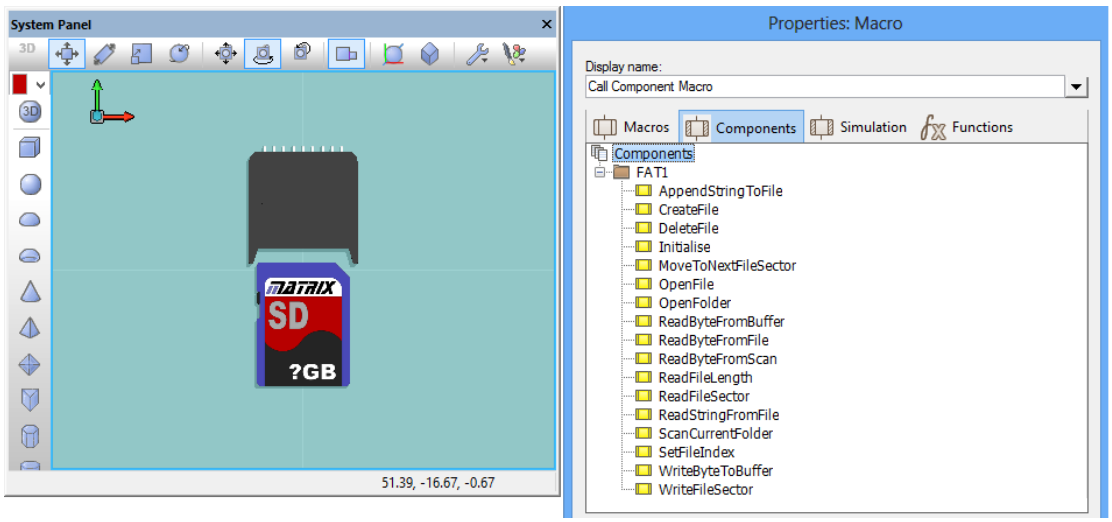


Рис. 10.6. Компонент FAT на системной панели и свойства макроса

В частности это SD-карта, которую вы вставляете в фотоаппарат или планшет. Secure Digital Memory Card – это полное имя SD носителя. Этот вид хранителя данных и использован в Flowcode. Расширенная анимация позволяет при симуляции программы, как это сделано в примере FAT Data Logger_Simple_Slow.fcfx, наглядно показать работу с этой картой памяти.

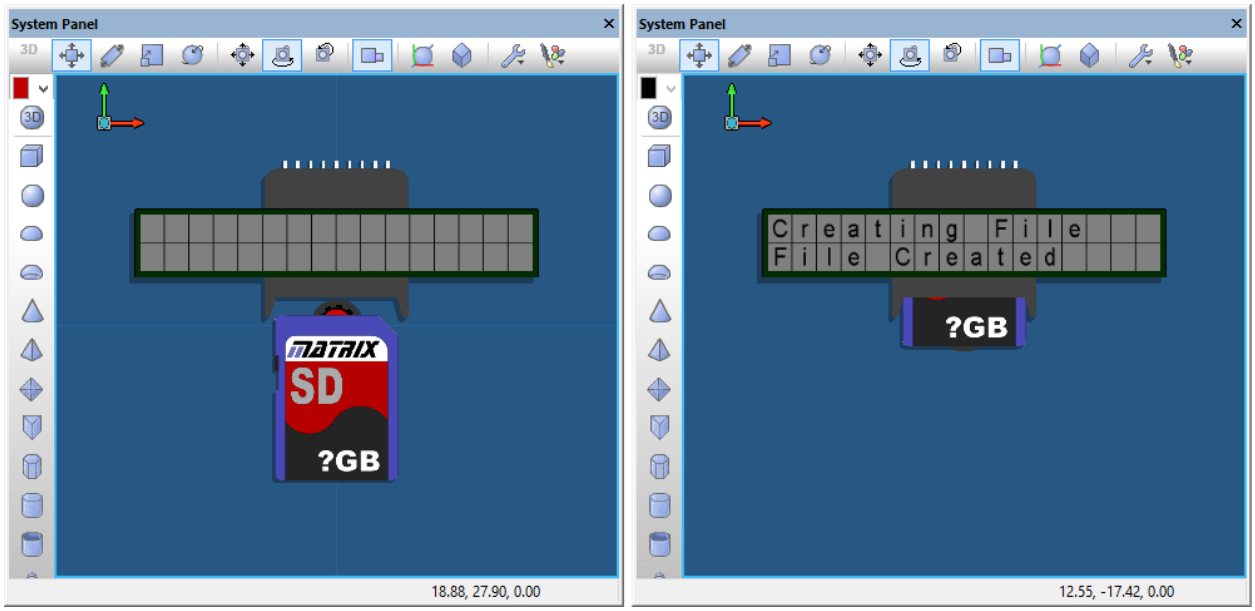
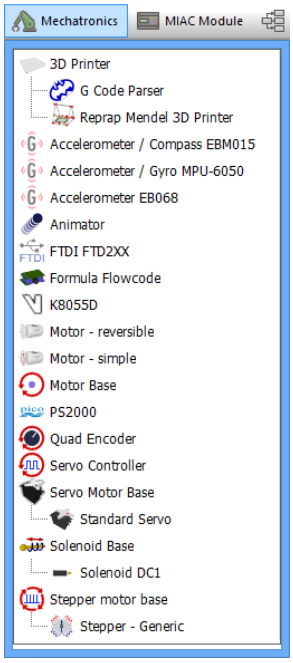


Рис. 10.7. Симуляция примера из набора Flowcode 6

Последний компонент Lookup tables – просмотр таблиц.

Mechatronics (мехатроника)

Сегодняшний бум в создании роботов даже в школьных кружках обязан своим появлением бурному развитию производства микроконтроллеров. Цена микросхемы стала такой, что можно позволить себе применить, скажем, один микроконтроллер для управления моторами простого робота-танка, второй микроконтроллер использовать для работы с датчиками-глазами робота и т.д. Несколько микроконтроллеров, объединённые с помощью USART, делают робота достаточно интеллектуальным.



Обилие компонентов этого раздела – не только повышенный интерес к теме, но и наличие ряда модулей, для которых созданы макросы.

В эту группу, впрочем, попали и такое устройство, как трёхмерный принтер, и осциллограф-приставка к компьютеру, и элемент симуляции, отвечающий за анимацию, и мост USB-COM, и многое другое.

Однако преобладающим элементом остаётся мотор. Есть несколько разновидностей: от простейшего до шагового двигателя.

Рис. 10. 8. Компоненты раздела Mechatronics

Моторы раздела Mechatronics используются при работе с проектами, использующими эти компоненты.

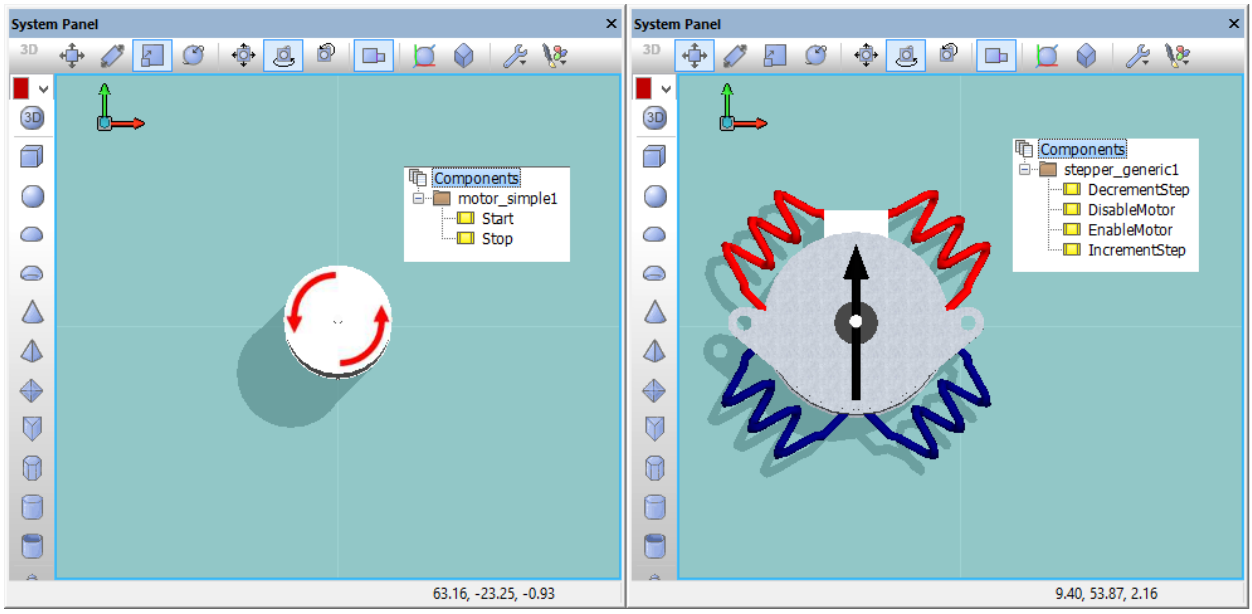


Рис. 10.9. Два мотора и их макросы из раздела Mechatronics

MIAC module

Рис. 10.10. MIAC

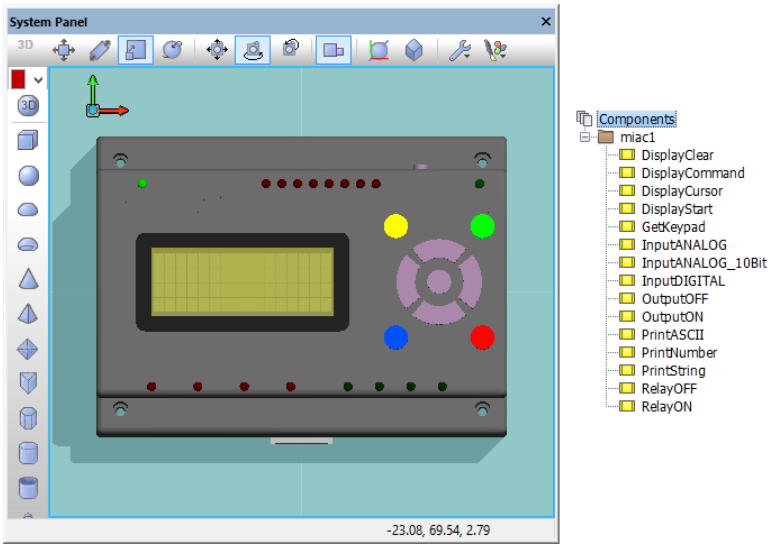


Рис. 10.11. Компонент раздела и свойства его макроса

Вот, что можно прочитать об этом на сайте Matrix:

> *MIAC E-system комплект разработки состоит из MIAC (управляющее устройство промышленного класса), нескольких модулей расширения и программы, которые позволяют инженерам быстро разрабатывать промышленные изделия управления и регистрации данных.*
>
> *Модули расширения: базовый (TTL), расширенный, ZigBee (согласующее устройство и маршрутизатор), GPS и Bluetooth. Вскоре появятся модули Ethernet и GSM.*
>
> *Наиболее привлекательным в MIAC E-system является то, что она имеет очень гибкий набор составляющих, которые могут использоваться для создания широкого круга электрических систем в очень короткие сроки. Не требуется серьёзной подготовки в части программирования, хотя система использует шину CAN для коммуникации, а при разработке не требуется даже знания шины CAN.*

Раздел содержит два компонента:

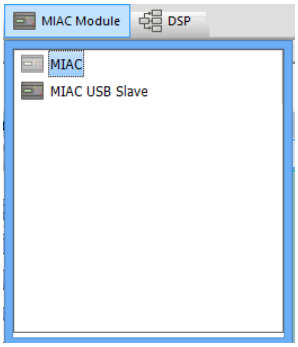


Рис. 10.12. Раздел MIAC module

DSP (цифровой сигнальный процессор)

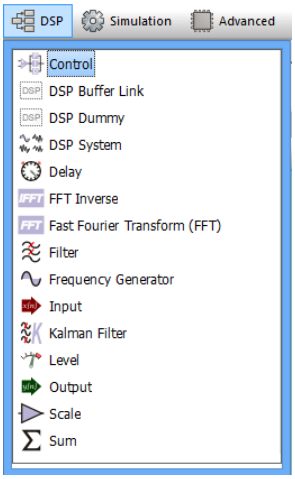


Рис. 10.13. Компоненты для работы с сигнальными процессорами

Широкое распространение средств оцифровки аналоговых сигналов и преобразование цифровых данных в аналоговые сигналы позволило не только создать новые накопители, мы давно привыкли к тому, что CD-диски вытеснили когда-то обязательные дискеты, но и привыкли к музыке на CD-дисках, привыкли пользоваться MP3-проигрывателями. Поэтому учебные заведения весьма заинтересованы в помощи в этом направлении от производителей обучающих программ.

Если при создании проекта заглянуть в перечень микроконтроллеров, обозначенных как PIC16, то:

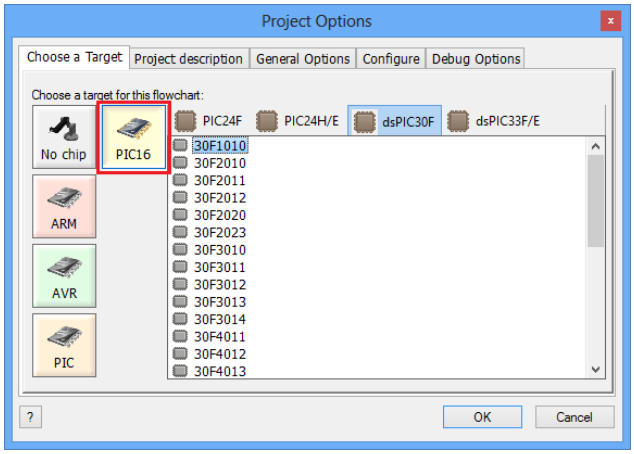


Рис. 10.14. Сигнальные процессоры Microchip

А в разделе DSP такие привычные инструменты для работы с аналоговыми сигналами, как генератор, быстрый анализ Фурье или фильтры. Впрочем, как и с осциллографом, не очень ясно, как пользоваться этими инструментами. Вот, что можно найти в подсказке:

> *Компонент frequency generator создан для вставки в DSP систему и позволяет генерировать несколько дискретных сигналов.*

Действительно, в свойствах компонента можно найти разные формы сигналов:

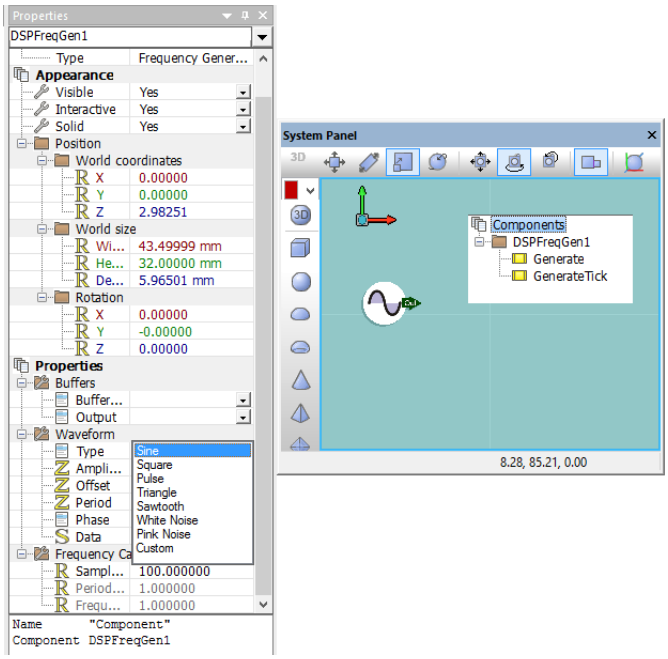


Рис. 10.15. Компонент, свойства макроса и выбор формы сигнала

Simulation

Этот раздел компонентов появился в версии 6.

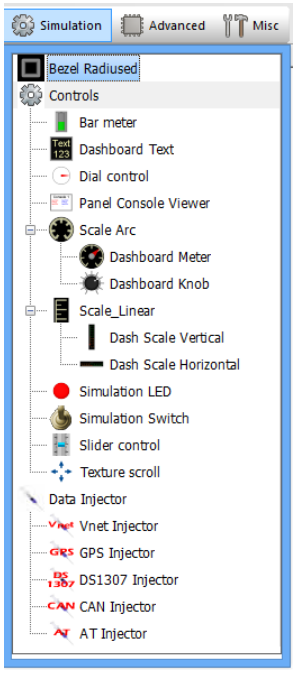


Рис. 10.16. Simulation

Некоторые компоненты этого раздела, видимо, предназначены для размещения на новой панели этой версии – панели управления. Та часть, что относится к Data Injector, предназначена для генерации данных. А в остальном попробуем разобраться в следующих главах.

Advanced

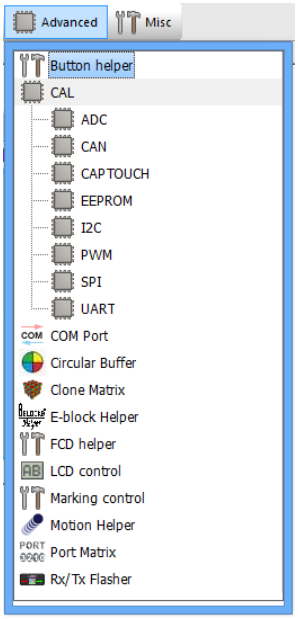


Рис. 10.17. Компоненты раздела «продвинутых» компонентов

Вот два компонента и свойства их макросов:

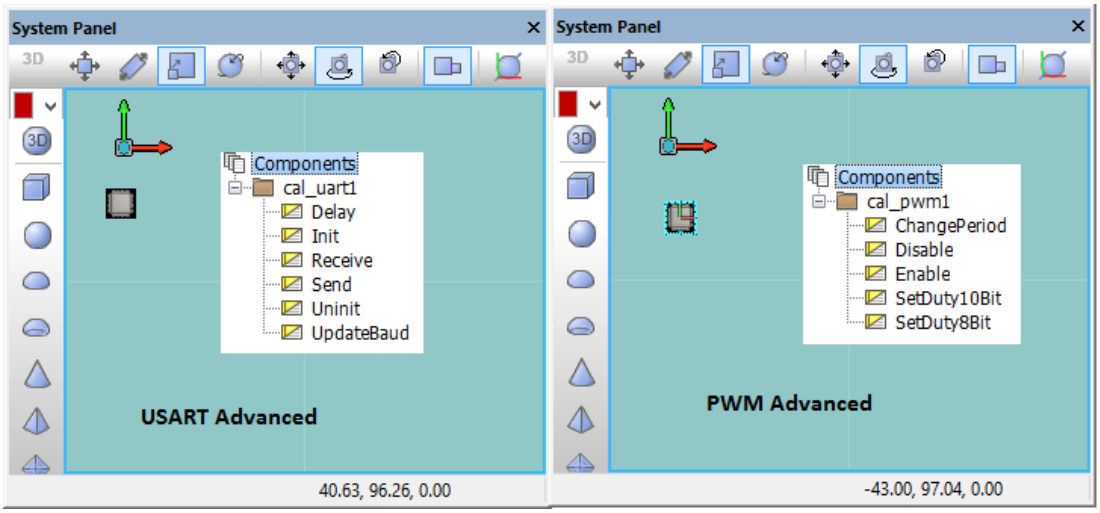


Рис. 10.18. USART и PWM раздела Advanced

Их можно сравнить со свойствами макросов RS232 и PWM:

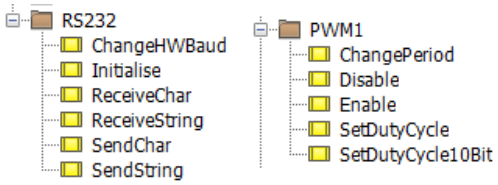


Рис. 10.19. Свойства макросов RS232 и PWM

Можно сравнить и свойства самих компонентов:

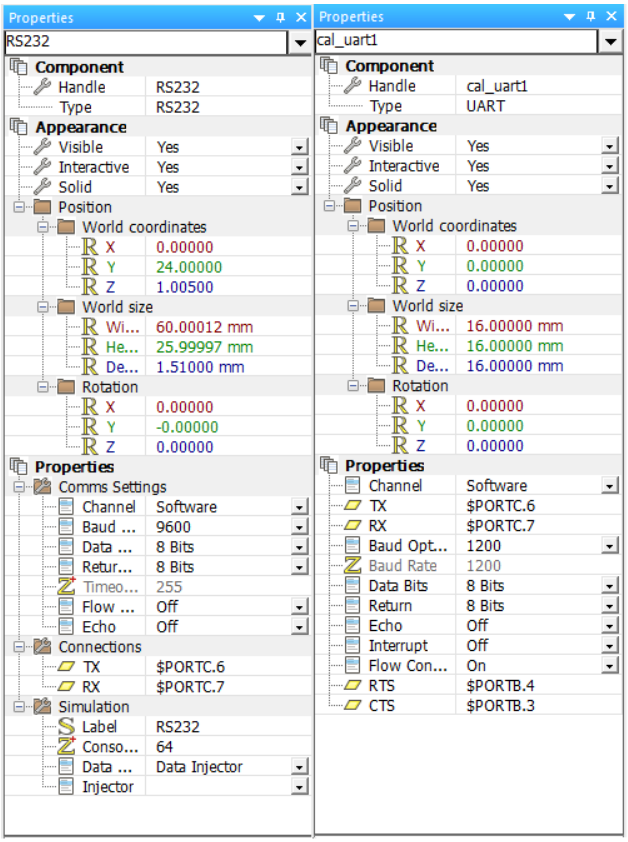


Рис. 10.20. Сравнение свойств компонентов RS232 и USART

О компоненте COM Port написано, что он используется при симуляции последовательного обмена с такими системами, как RS232, Bluetooth и USB Serial.

О том, что представляют собой другие компоненты, можно прочитать в разделе Help.

Misc

Эта группа компонентов пока пуста, видимо, с развитием версии она будет пополняться или…

Глава 11. Продолжение знакомства с новшествами

Просматривая примеры, полезное занятие, я обратил внимание на то, что в одном из них консоли, отображающие работу UART, показывают не только то, что на выходах микросхемы.

Пример работы с RS232

Вот этот пример:

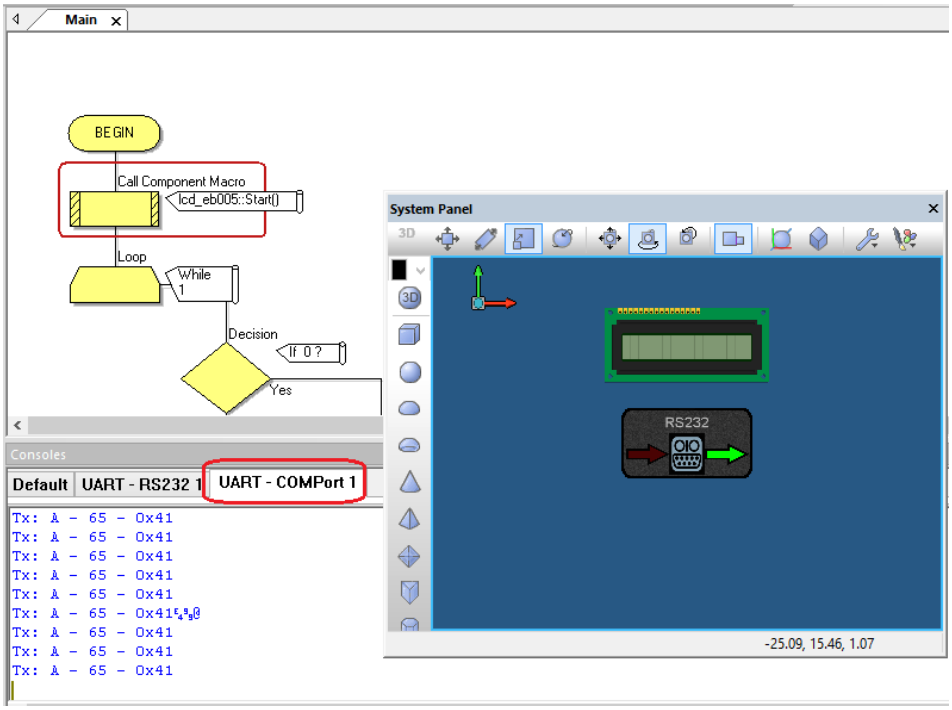


Рис. 11.1. Симуляция примера RS232 Software UART

Я попробовал подключить порт COM1:

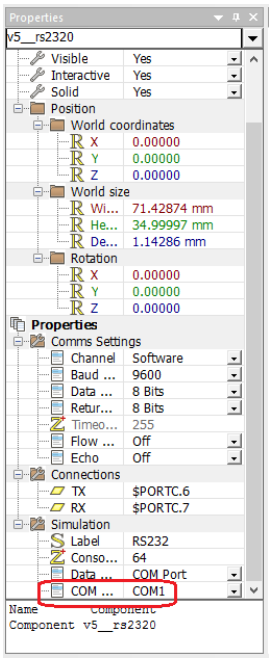


Рис. 11.2. Окно свойств RS232

И, запустив симуляцию программы, подключил реальный осциллограф к порту COM1:

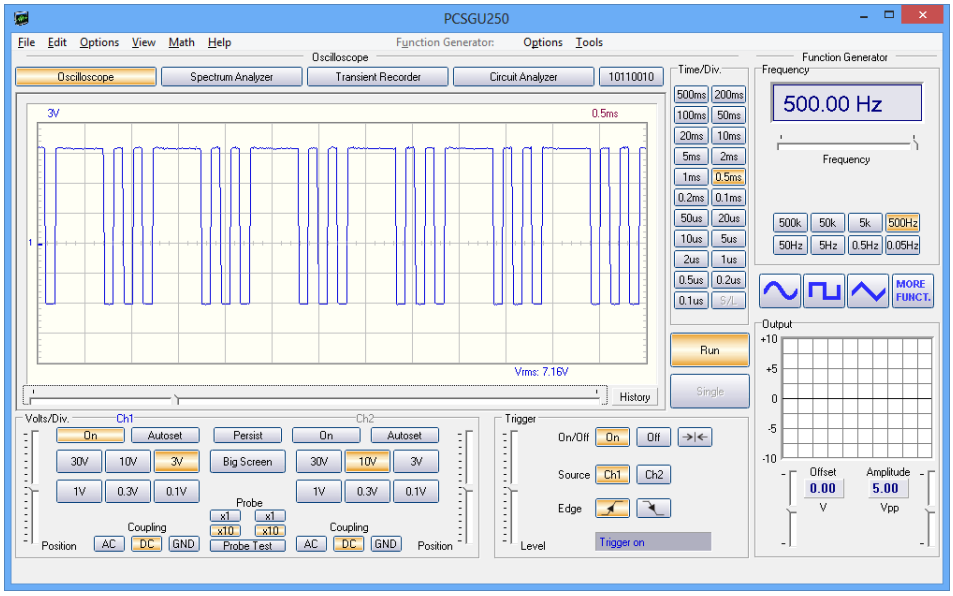


Рис. 11.3. Сигнал на выводе COM-порта

Я не знаю, интересно ли это, но это есть.

Arduino

Если для учебных заведений в стране, где Matrix предлагает модули, работающие с Flowcode, доступны по цене, то этого не скажешь о начинающих российских радиолюбителях. Вместе с тем, есть модуль доступный и им по цене – это Arduino. База модуля – микроконтроллер AVR. Несколько лет назад меня заинтересовало, можно ли работать с модулем Arduino в программе Flowcode. Оказалось, что можно.

Просматривая начало работы с проектами в Flowcode 6, я обнаружил такой вариант начала работы:

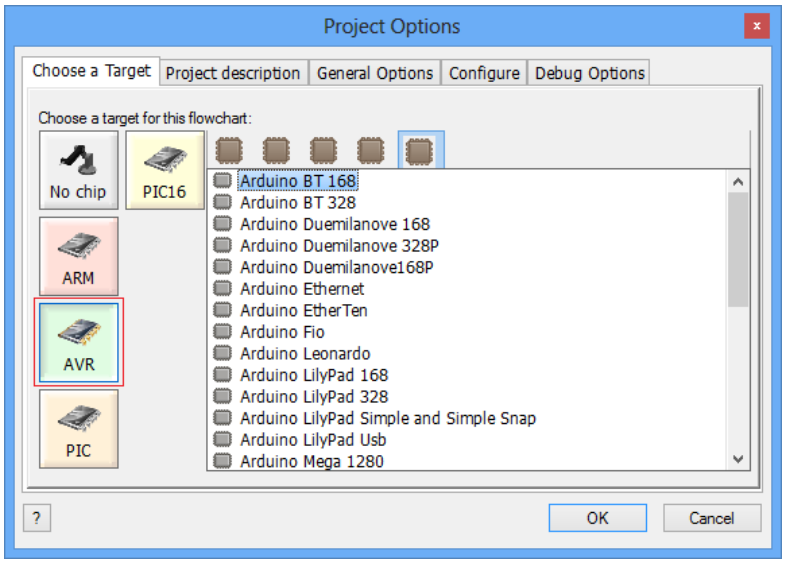


Рис. 11.4. Создание нового проекта для модуля Arduino

Проект Arduino хорошо поддерживается компилятором и библиотекой функций на языке Си, но и графический язык программирования микроконтроллеров не станет обузой.

От предыдущего рассказа о проекте Arduino у меня остался модуль, с которым интересно попробовать что-нибудь в среде Flowcode, где он находится под именем Arduino Nano 168.

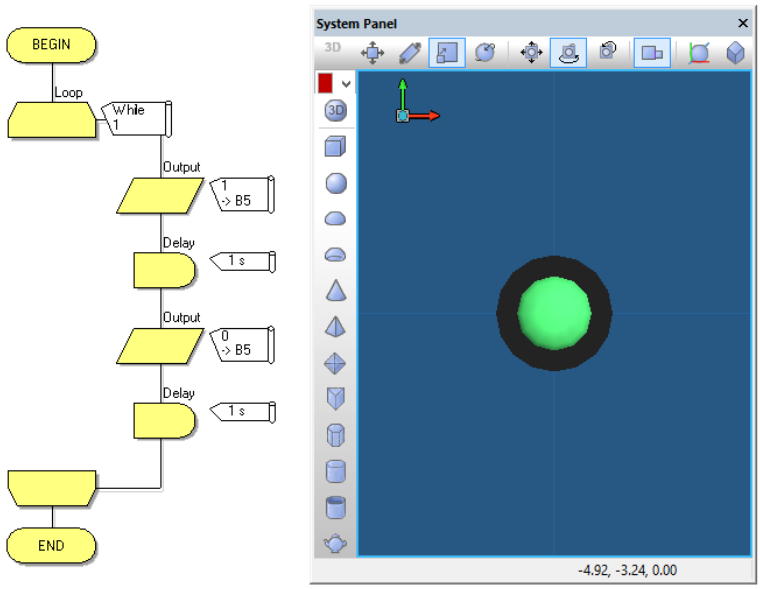


Рис. 11.5. Начало работы с новым проектом

В первую очередь я намерен проверить, подключив модуль, насколько быстро можно начать с ним работать. Простейшая программа. На плате Arduino для её проверки есть светодиод. Осталось оттранслировать программу в микросхему:

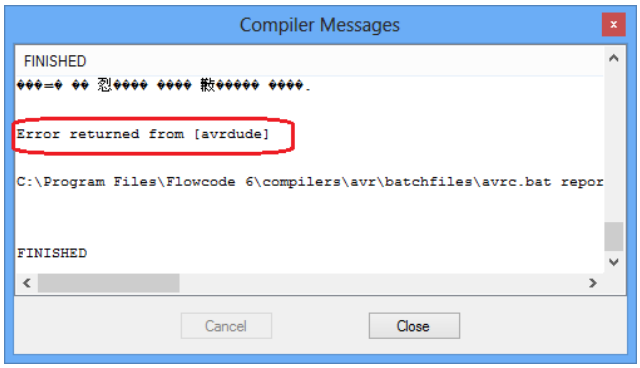


Рис. 11.6. Сообщение об ошибке

Обращение к подсказке не привело к решению проблемы, настройки, указанные в ней…

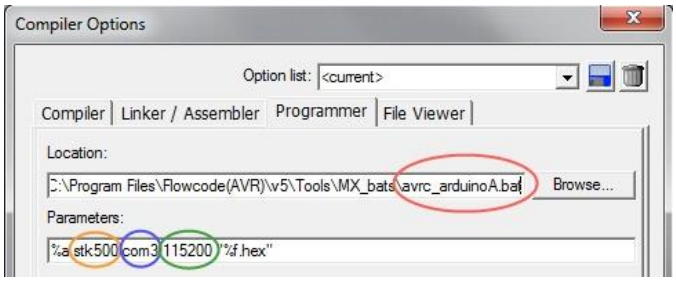


Рис. 11.7. Настройки из подсказки к программе

…дали сообщение:

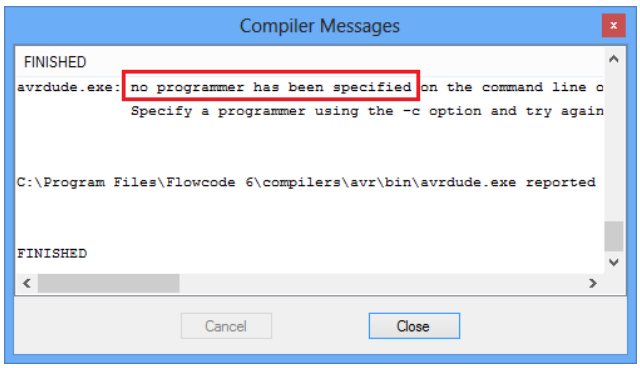


Рис. 11.8. Новое сообщение об ошибке

Я бы связал эту неудачу с тем, что указанный путь в подсказке к программе обслуживания программатора, отсутствует. А, указав путь к программе avrdude в составе Flowcode 6, вновь получаешь сообщение о том, что не задан программатор.

От прошлого рассказа о проекте Arduino у меня остались и программа avrdude, и настройки к Flowcode для работы с программой обслуживания программатора. Но начать этот процесс лучше с того, чтобы заглянуть в диспетчер устройств после подключения модуля Arduino:

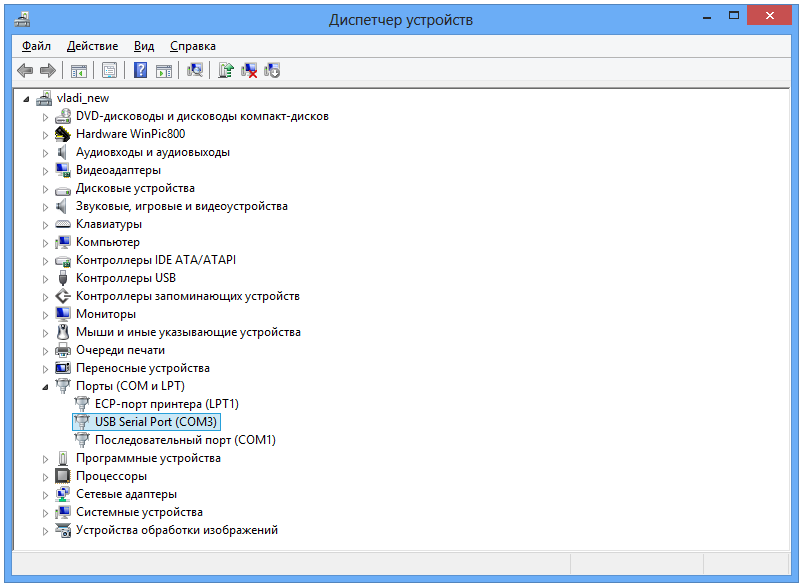


Рис. 11.9. Появление нового устройства после подключения Arduino Nano 168

В настройках понадобится указать виртуальный порт COM3. Начинается настройка с выбора настроек компилятора…

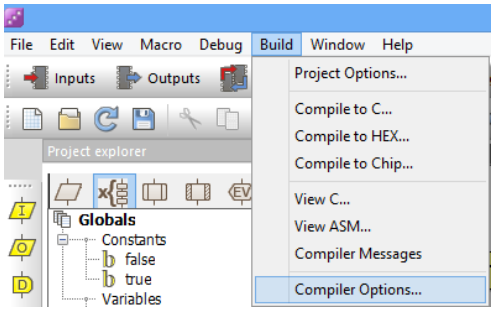


Рис. 11.10. Раздел настроек компилятора основного меню

…где есть закладка программатора:

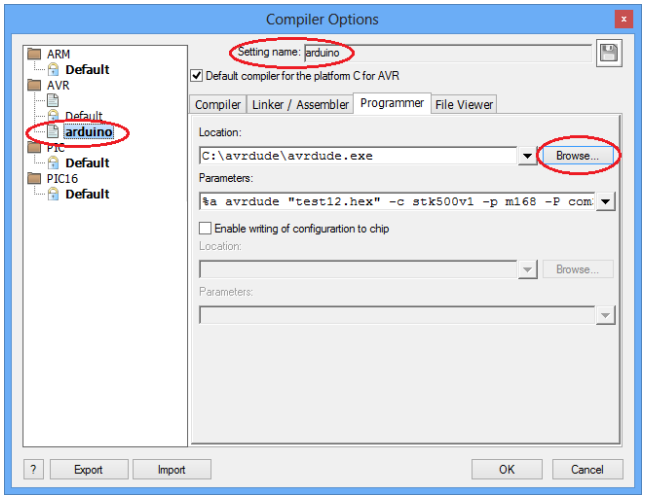


Рис. 11.11. Закладка программатора в настройках компилятора

Имя установок для программатора можно задать любое, но arduino, мне кажется, подходящее имя для новых настроек программатора. Расположение программы avrdude можно указать, используя проводник. Для этой цели служит кнопка **Browse** рядом с текстовым полем Location.

А параметры я использую те, что работали в прошлый раз. К сожалению, без ложки дёгтя не обошлось – имя файла пришлось вводить вручную в настройках. Возможно, и эту проблему как-то можно обойти. Хотя было бы лучше, если бы её не было:

```
%a avrdude "test12.hex" -c stk500v1 -p m168 -P com3 -b 19200 -
Uflash:w:"test12.hex":i -C C:\avrdude\avrdude.conf
```

Теперь при компиляции программы в модуль загрузка выполняется полностью:

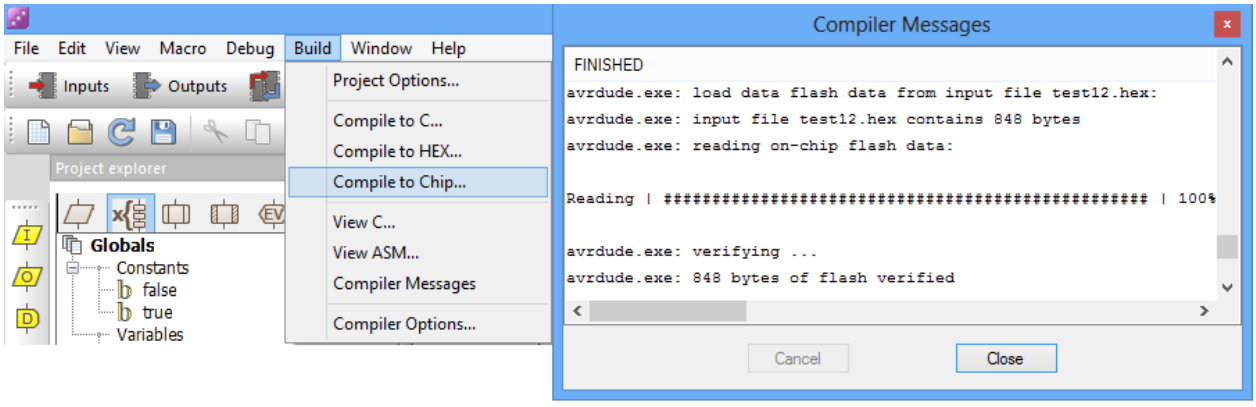


Рис. 11.12. Компиляция программы в модуль Arduino

Ниже в рассказе о программаторе есть строка параметра. Если исправить выше показанный текст раздела параметров:

```
%a avrdude "$(target).hex" -c stk500v1 -p m168 -P com3 -b 19200 -
Uflash:w:"$(target).hex":i -C C:\avrdude\avrdude.conf
```

То всё работает без проблем.

Программатор

Долгое время я пользовался программатором JDM, который успешно работал под управлением программы ICProg. Я использовал микросхему PIC16F628A в своих рассказах и про «умный дом», и про работу в операционной системе Linux, и в остальных своих рассказах. Чтобы не усложнять рассказ и макетную плату, я использовал микросхему с встроенным тактовым генератором. Проблемы не возникали. Но недавно обнаружилась проблема с JDM программатором: микросхему использовали с внешним кварцем, затем с внутренним тактовым генератором. После этого микросхема перестала перепрограммироваться. Проблема решаемая, достаточно её стереть с помощью программатора PICKit2. Но для этого его нужно иметь.

Какой программатор работает в среде Flowcode по умолчанию, я не знаю, возможно, тот, что выпускает разработчик программы. Но и PICKit2 (PICKit3) должен работать в среде Flowcode 6, о чём свидетельствует раздел Help:

PICkit2

Location:
C:\Program Files\Flowcode 6\Tools\PICkit2\pk2cmd.exe
Заметьте, что для 64-битового компьютера путь меняется с Program Files\ на Program Files (x86)\.
Parameters:
-BC:\Program Files\Flowcode 6\Tools\PICkit2\ -PPIC$(chip) -F$(target).hex -M -A5 -R
Заметьте, что для 64-битового компьютера путь меняется с Program Files\ на Program Files (x86)\.

PICkit3

Location:
C:\Program Files\Flowcode 6\Tools PICkit3\PK3CMD.exe
Заметьте, что для 64-битового компьютера путь меняется с Program Files\ на Program Files (x86)\.
Parameters:
-P$(chip) -F$(target).hex -E –M
Или для 16-битовых устройств
Parameters:
-P$(chip) -F$(target).hex -E -M -L

При настройке программатора следует учитывать модели микроконтроллеров, для которых настраивается программатор.

Настройка программатора для микросхемы PIC16F628A выглядит так:

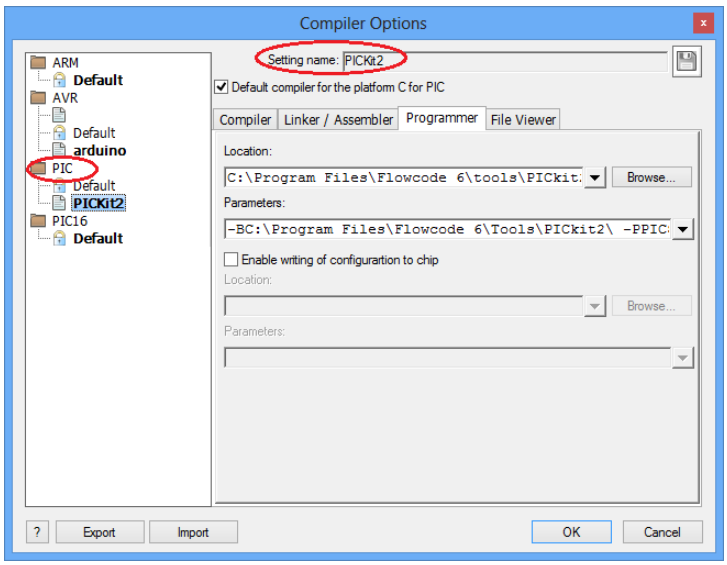


Рис. 11.13. Настройка программатора в настройках компилятора

Программа компилируется и загружается в микросхему, завершаясь сообщением:

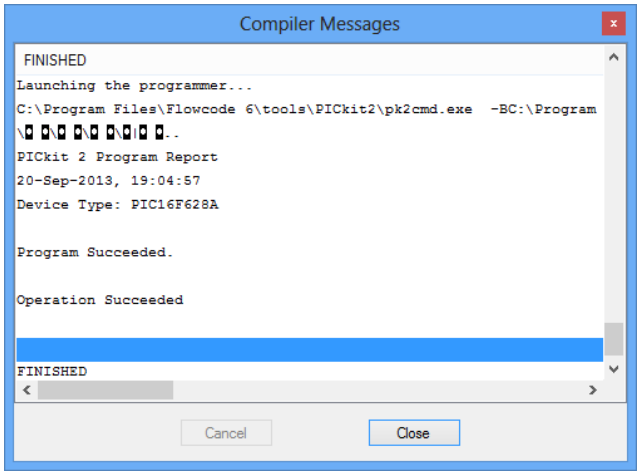


Рис. 11.14. Сообщение о завершении компиляции программы

Шаблоны

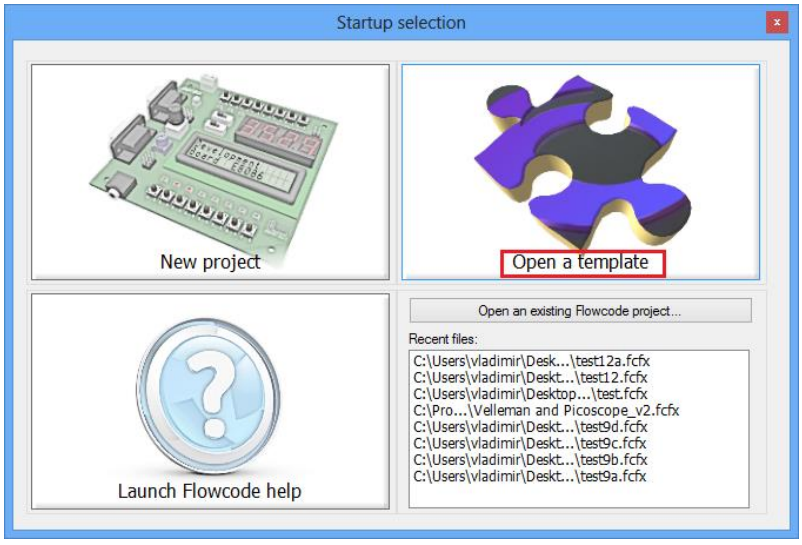


Рис. 11.15. Раздел шаблонов при создании нового проекта

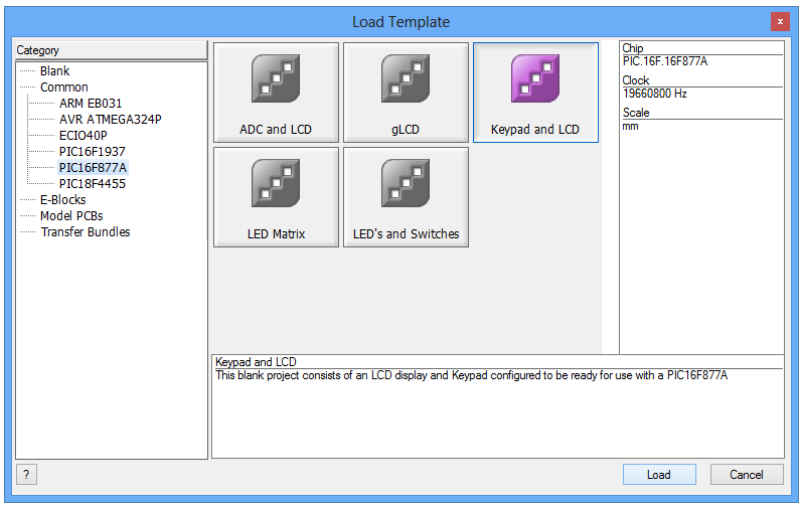


Рис. 11.16. Доступные шаблоны

Загрузим (кнопка **Load**) выделенный выше шаблон.

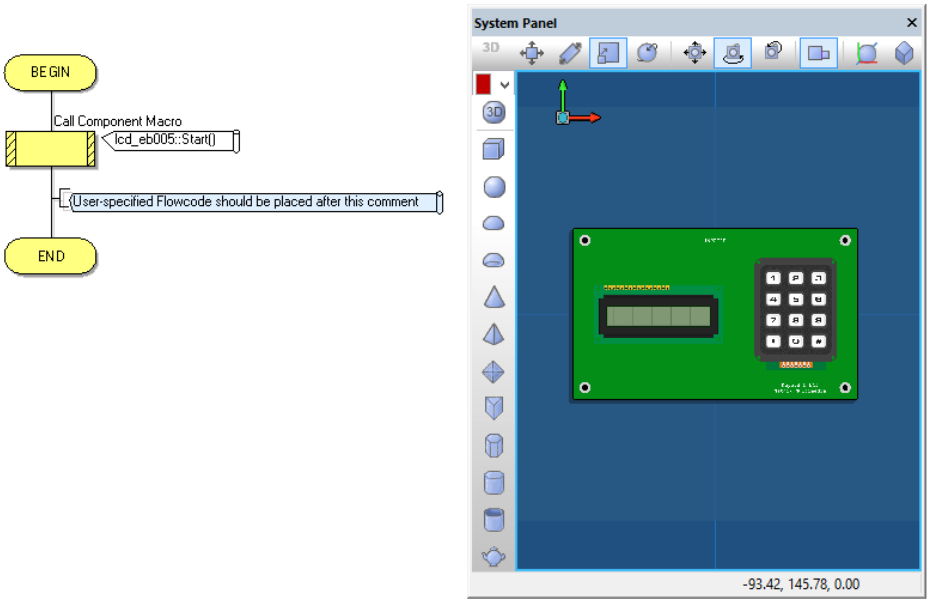


Рис. 11.17. Шаблон программы, использующей клавиатуру и дисплей

Значительная часть шаблонов предназначена для работы с модулями Matrix. Что не удивительно.

Ещё одно замечание

В примере с RS232 из набора примеров Flowcode использован компонент, который имеет название v5_RS2320. Меня в этом примере заинтересовало то, что данные появляются на выходе реального COM-порта компьютера.

Повторив нечто похожее с компонентом RS232 из компонентов Flowcode 6, можно увидеть, что данные отправляются по UART, но COM-порт «молчит».

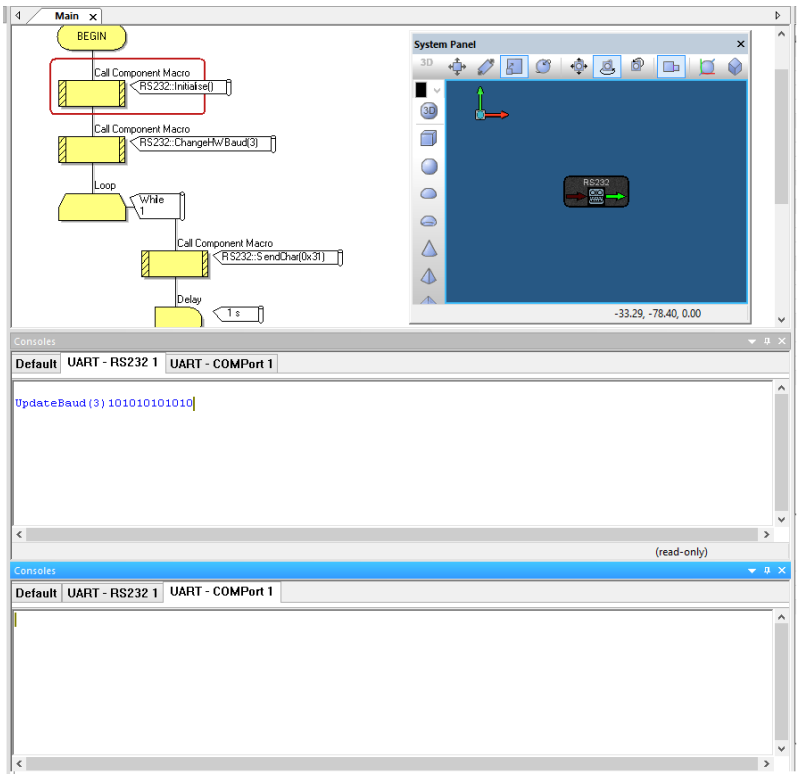


Рис. 11.18. Моделирование RS232

Но, если поправить свойства компонента, то ситуация изменится.

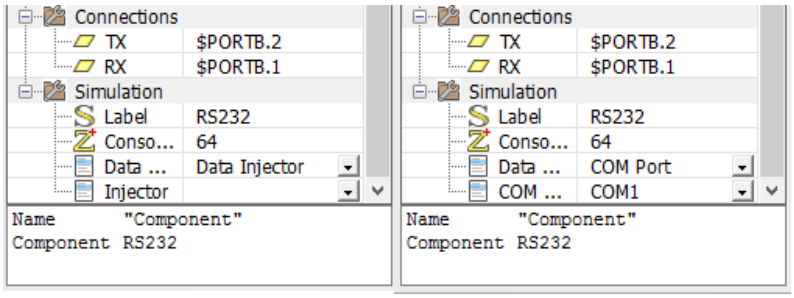


Рис. 11.19. Настройка свойств RS232

Результат моделирования:

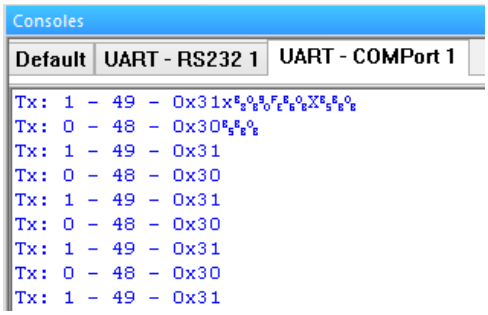


Рис. 11.20. Появление отправляемых символов в COM-порте

Теперь понятно и назначение такого компонента, как Injector данных.

Глава 12. То, что осталось непонятным ранее

Время действия лицензии на пробную версию медленно, но упрямо перемещается к нулю.

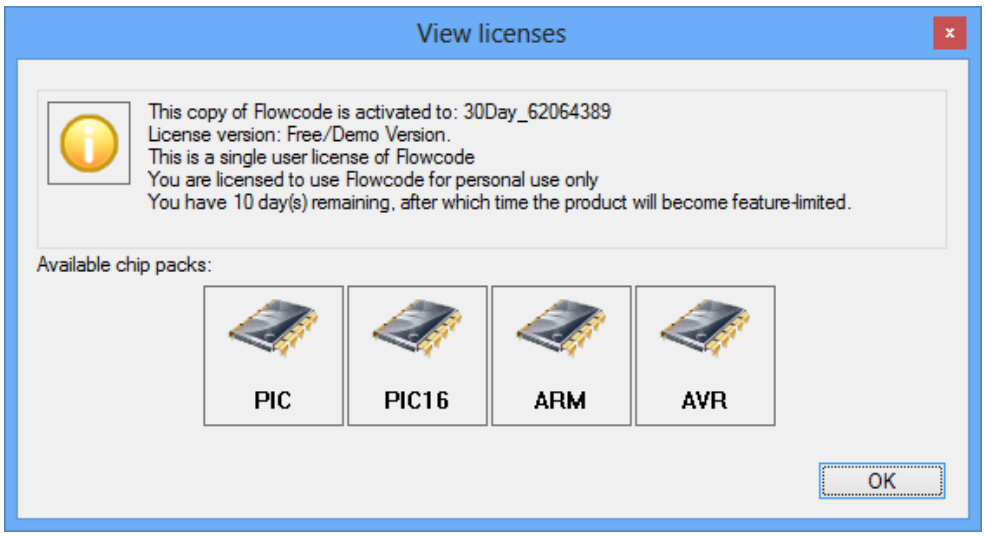


Рис. 12.1. Оставшееся время работы пробной версии

Было бы интересно попробовать что-то сделать на AVR или ARM контроллерах, но остались невыясненными некоторые вопросы, которыми я хочу заняться.

Использование Data Injector

Некоторые из общих компонентов, основанных на коммуникации, имеют встроенные опции для использования «впрыскивателя» данных, что позволяет симулировать внешние коммуникационные шины. Data Injectors разработаны для копирования функциональности специфических внешних интерфейсов.

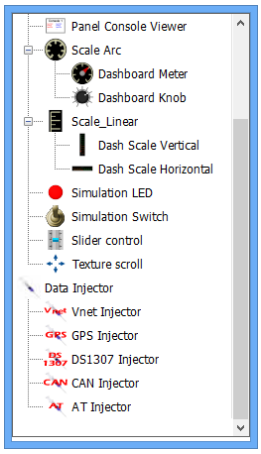


Рис. 12.12. Состав группы Simulation

В настоящее время в Flowcode включены:

- Injector Base – пустой компонент, используемый для создания «заглушек» в компонентах коммуникации, которые могут быть замещены актуальным injector при экспорте компонента.
- AT Injector – простой, на основе AT команд injector, который ждёт возврата каретки и затем эхом возвращает любые данные, что получил вместе с ответом OK.

- CAN Injector – метод декодирования CAN ID через обращение к внешнему CSV файлу, содержащему данные в формате ID,DecodeString\n.
- DS1307 Injector – простое I2C устройство, компромисс между часами реального времени на основе времени PC системы и разделом пользовательской памяти, который может быть доступен.
- GPS Injector – метод динамического добавления в компонент данных NMEA типа GPS сообщения.
- Vnet Injector – метод передачи связи с другими узлами Flowcode или другими устройствами, использующими TCP-IP стиль сообщений в сети.

С примером создания пользовательского «впрыскивателя» данных можно ознакомиться на форуме Matrix.

Некоторые компоненты группы Advanced

Компонент RxTx Flasher 

Это простая индикаторная панель, которую можно добавить на dashboard панель. Вызов макроса Flash заставит мигать LED заданное время. Индикатор может иметь этикетку, есть иконка, в которую вы можете загрузить файл картинки. Светодиоды могут также мигать, когда компоненту отправляются события User Notify. Событие Message ID игнорируется.

Данные: Flash Rx = 1, Flash Tx = 2.

LCD компонент управления 

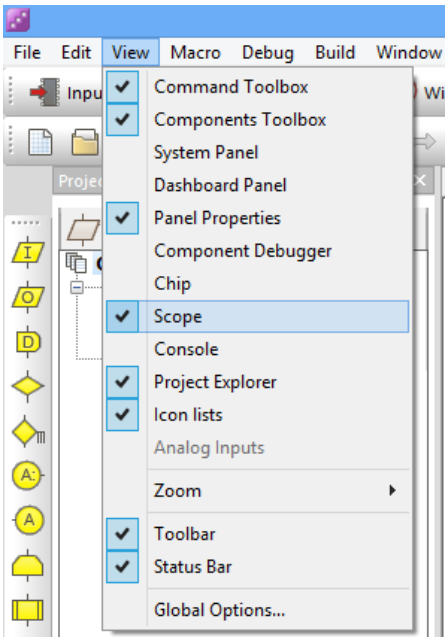
LCD пульт управления поддерживает до 64 x 64 единиц, подходящих для симуляции.

CAL – ADC

Компонент позволяет вам использовать всю доступную функциональность аналогово-цифрового преобразователя вашего микроконтроллера самым простым образом. С помощью этого компонента аналоговое напряжение на входе может считываться как цифровое значение. Этим достигается возможность в проекте включить аналоговый вход от потенциометра с помощью более мощных инструментов измерения и датчиков, чем просто вывод аналогового значения.

Компонент ADC CAL – это основа для других компонентов, которые добавят видимые эффекты при симуляции, такие как представление аналоговых вводов ползунками или наборными дисками.

Scope



Первое упоминание об осциллографе я встретил в одном из примеров из набора Flowcode 6.

Меня заинтересовал этот пример, благо и делать ничего не нужно, запустить моделирование и наблюдать результат.

К сожалению, с наблюдением результата не повезло – я ничего не увидел на экране виртуального осциллографа.

Иногда посмотреть с помощью осциллографа, что происходит на выводе микроконтроллера, например, полезно. Поэтому я попытался как-то посмотреть происходящее на выводе Tx.

Увы. Ничего у меня не получилось.

Рис. 12.2. Осциллограф в основном меню

Осциллограф не является компонентом, имеющим набор свойств. Попытка использовать программный компонент Simulation, добавляя что-то, что казалось подходящим к случаю, позволила только заполнить окно осциллографа начальным изображением «луча»:

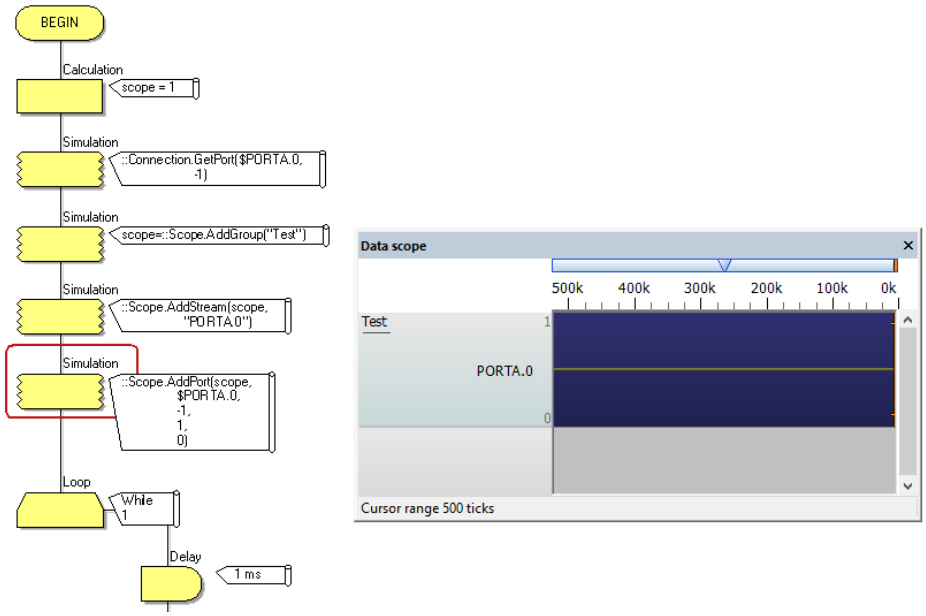


Рис. 12.3. Начальное изображение

Это начальное изображение, уж не знаю почему, переходит в нуль, если программа встречает компонент Delay.

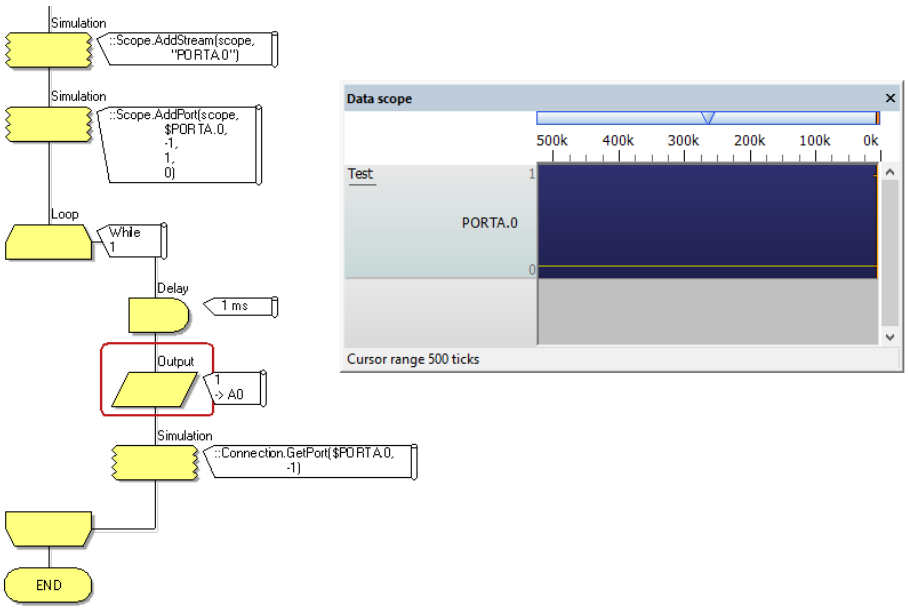


Рис. 12.4. И конечное изображение

Большего мне достичь не удалось.

Я не думаю, что достиг бы большего, перебирая варианты использования разных функций. Не думаю. И я расскажу, почему так…

Я искал ответ на вопрос, как пользоваться компонентом Scope, в файлах помощи, искал на форуме Matrix, но не преуспел, пока не наткнулся на тему: Squarewave generator (sim only).

Как я понимаю, один из разработчиков Jonny W предложил дополнить программу компонентом, позволяющим просмотреть на экране виртуального осциллографа прямоугольные импульсы. Я скачал этот компонент, добавил его в папку components, следуя инструкции, но получил тот же результат, что и участник форума, задававший вопрос об использовании Scope – компонент не удалось добавить ни на системную панель, ни на панель управления. Вы можете проверить это.

Однако, скачав там же программу squarewave.fcfx, я убедился, что шансы на удачу есть. Программа работала. Я советую скачать программу, чтобы посмотреть, что требуется для работы Scope. Вот некоторые из фрагментов программы:

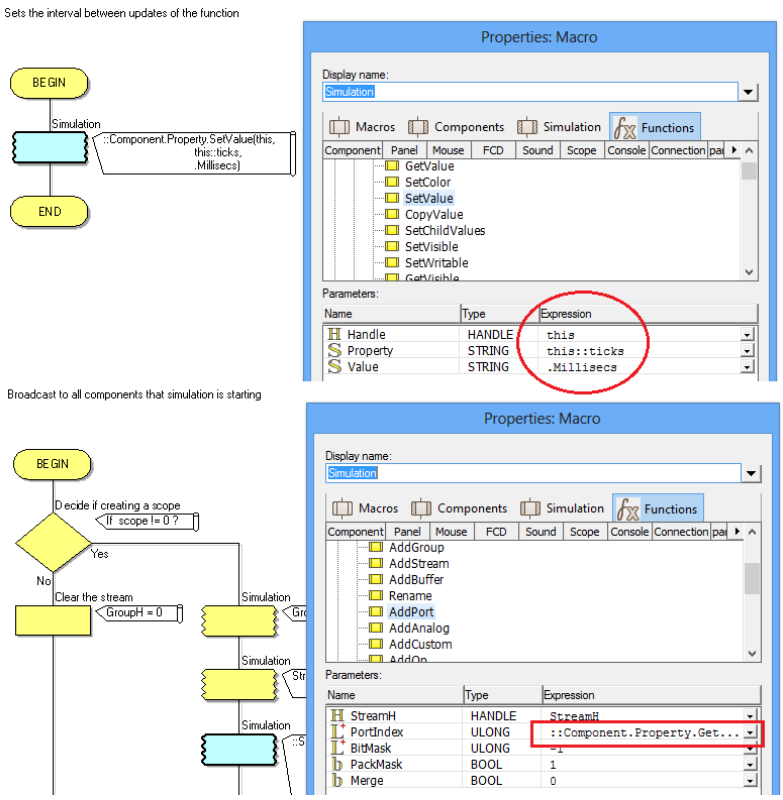


Рис. 12.5. Фрагменты программы squarewave.fcfx

Для тех, кто свободно себя чувствует в программировании на C++, видимо, всё понятно, хотя и они, как мне кажется, должны иметь лучшее представление об исходном тексте программы Flowcode.

Но не всё так плохо. Достаточно экспортировать компонент из этой программы (картинку для иконки я сделал в прошлый раз), чтобы получить искомое.

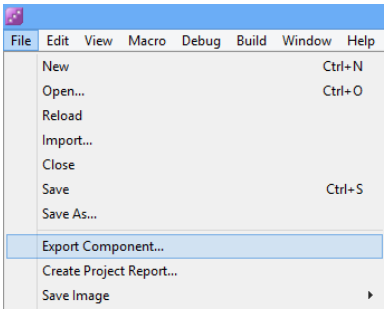


Рис. 12.6. Раздел экспорта в пункте File основного меню

Я ничего не трогал ни в программе, ни в диалоге экспорта компонента (кроме иконки).

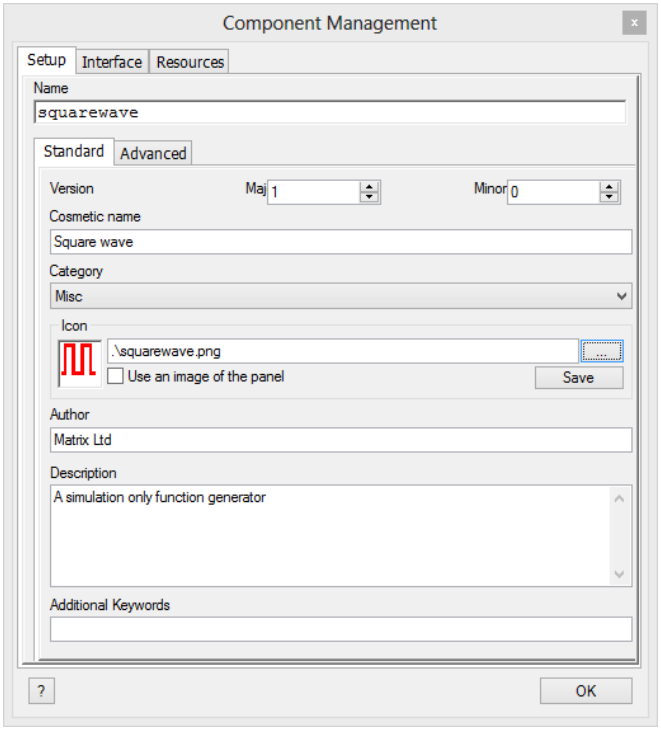


Рис. 12.7. Экспорт компонента из программы squarewave.fcfx

Кнопка **ОК** позволяет получить новый компонент.

И, я уже упоминал об этом, в Windows 8 я не могу сохранить результат сразу в нужной папке, поэтому сохраняю на рабочем столе:

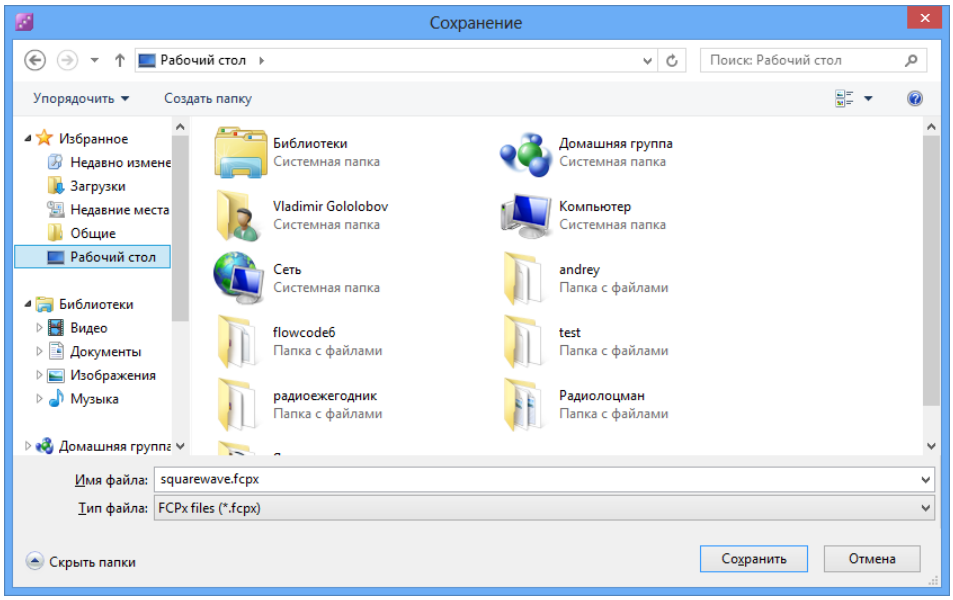


Рис. 12.8. Сохранение компонента на рабочем столе

С рабочего стола я могу перенести его в место установки программы Flowcode 6 в папку components.

Теперь, запустив программу, в группе Misc можно найти компонент, который можно добавить и на системную панель, и на панель управления.

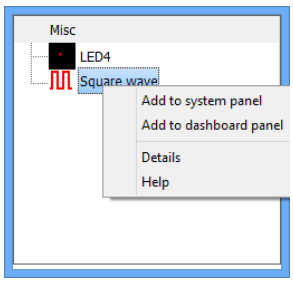


Рис. 12.9. Появление нового компонента в группе Misc

Кстати, в той же группе у меня остался светодиод, который я создавал в одной из предыдущих глав. Создав новую простейшую программу, можно проверить, работает ли новый компонент.

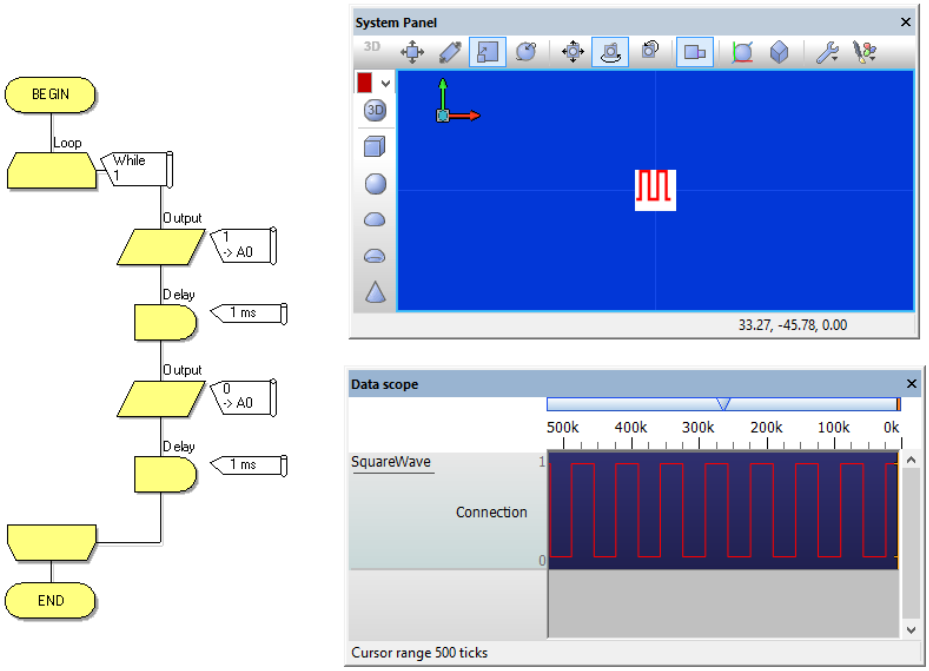


Рис. 12.10. Проверка работы компонента Square Wave

Теперь компонент не только добавляется на панель, но и имеет свойства, как любой другой:

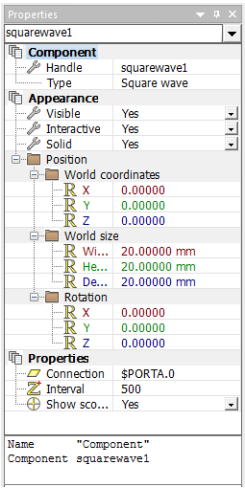


Рис. 12.11. Свойства компонента Square Wave

Вместо заключения

Конечно, не обо всём новом, что появилось в этой версии Flowcode, я успел рассказать. Но за оставшиеся несколько дней я едва ли существенно дополню рассказ. Да и с любой программой следует поработать, решая конкретные задачи.

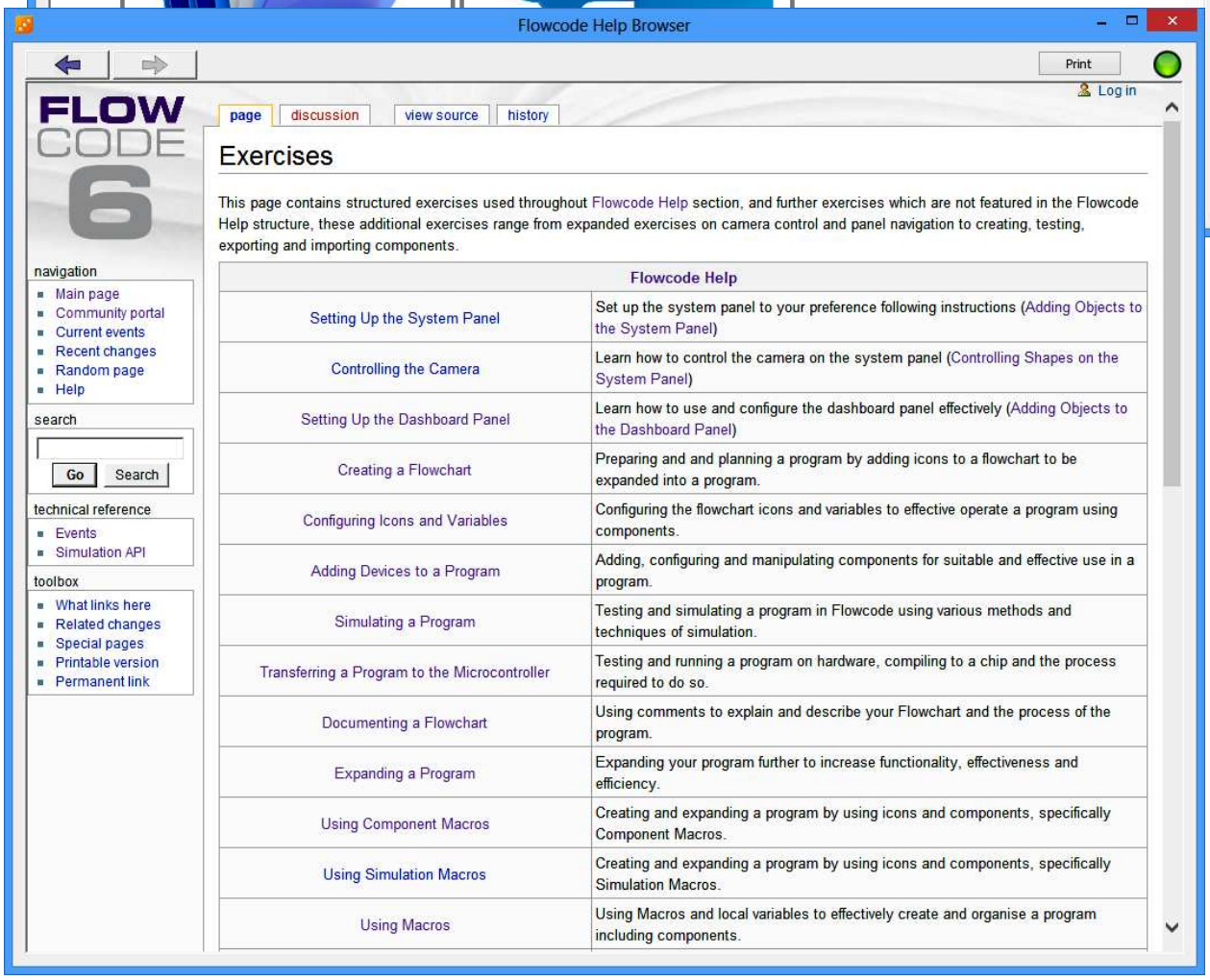
Появление новой версии операционной системы или программы всегда интересно. Появляется что-то полезное, что-то меняется, как вам кажется, напрасно. У каждого свои предпочтения, и все мы разные, чем друг другу и интересны.

Программа Flowcode 6 имеет много нововведений. Интересны они вам или нет, решать вам.

Flowcode 6. Exercises

Flowcode 6. Упражнения

Перевод В.Н. Гололобова



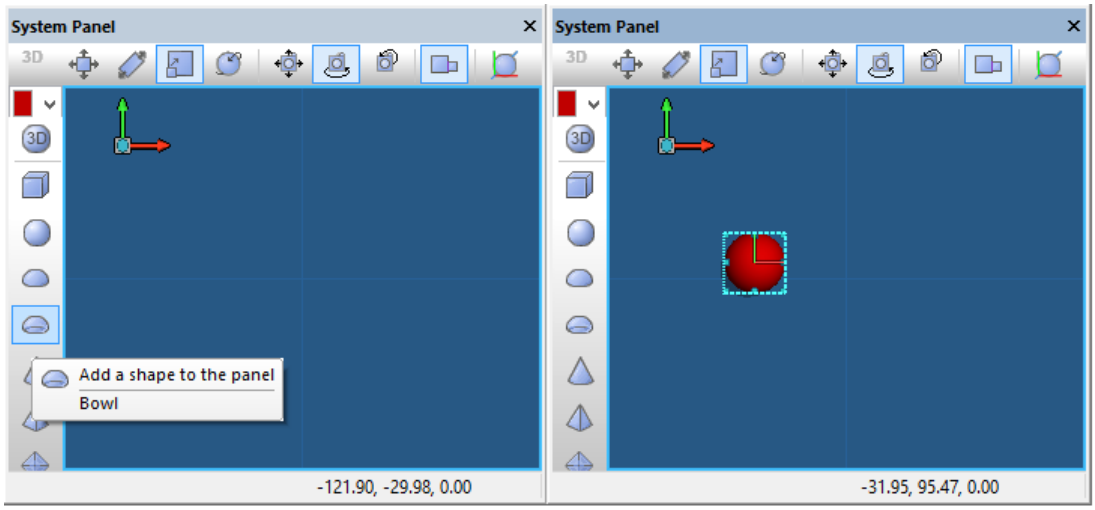
Список упражнений

1. Создание собственного компонента - Create an LED Component
2. Экспорт компонента LED - Exporting the LED Component
3. Импорт компонента - Importing the LED Component
4. Добавление объектов на системную панель - System Panel - Adding Objects
5. Системная панель – управление формами - System Panel - Controlling Shapes
6. Добавление объектов на панель управления - Dashboard Panel - Adding Objects
7. Управление множеством объектов - Dashboard and System Panel - Controlling Multiple Objects
8. Создание сложного компонента - Building a Complex Component - The Traffic Cone
9. Импорт и тестирование дорожного конуса - Importing and Testing the Traffic Cone
10. Конфигурирование иконок и переменных - Configuring Icons and Variables
11. Добавление устройств в программу - Adding Devices to a Program
12. Симуляция программы - Simulating a Program
13. Загрузка программы в микроконтроллер - Transferring a Program to the Microcontroller
14. Документирование программы - Documenting a Flowchart
15. Расширение программы - Expanding a Program
16. Использование устройств с аналоговым входом - Using Analogue Input Devices
17. Использование макросов - Using Macros
18. Использование компонентного макроса - Using Component Macros
19. Использование прерываний - Using Interrupts
20. Вставка кода в Flowcode - Inserting Code Into Flowcode

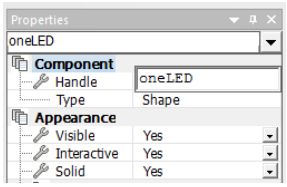
1. Создание собственного компонента

Создание изображения

- Создайте новый Flowcode flowchart. В подсказке по этому пункту, если вы загляните, написано, что вы создаёте новый проект для выбранного вами микроконтроллера.
- Убедитесь, что видимы панели системная и панель свойств.
- Если это не так, то обратитесь к разделу View основного меню.
- Щёлкните по полусфере в меню фигур и перетащите её на системную панель.



- Голубоватый точечный прямоугольник вокруг полусферы показывает, что объект в данный момент выделен.
- Измените имя компонента на «oneLED».



Чтобы это сделать:

- Найдите окно handle компонента в верхней части Panel Properties.
- Щёлкните по имени по умолчанию.
- Измените его на «oneLED».

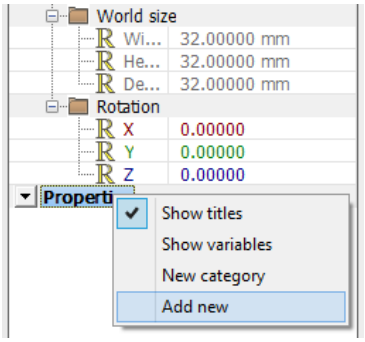
Добавление свойств

Далее, добавьте два свойства компонента LED. Первое ассоциировано с цветом. Второе задаёт соединение светодиода с микроконтроллером.

Вначале добавьте цвет по умолчанию для компонента, используя новое свойство, названное «color».

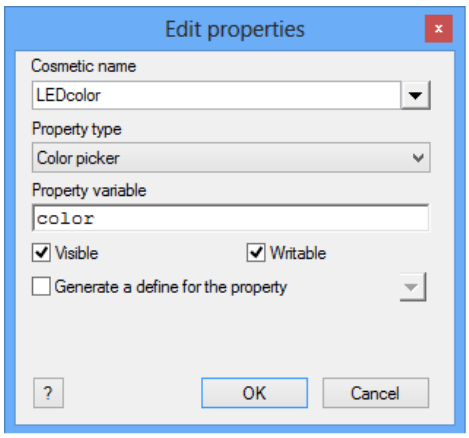
Чтобы это сделать:

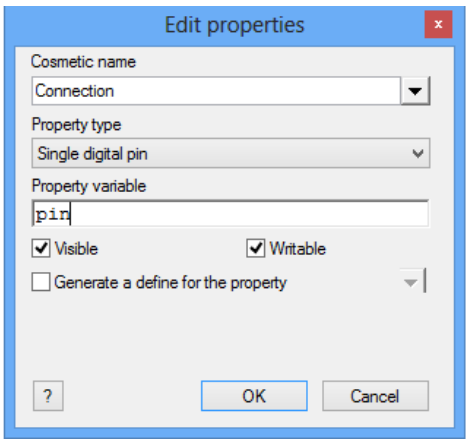
- Снимите выделение с полусферы, щёлкнув в любом свободном месте системной панели. Переместитесь в окно свойств.
- Наведите курсор мышки на иконку справа от названия Properties.
- Щёлкните по стрелке вниз.
- Щёлкните по «Add new».



- В окно Cosmetic name впечатайте «LEDcolor».
- Выберите «Color picker» для «Property type».
- Впечатайте color для имени «Property variable».

Аналогично добавьте второе новое свойство, чтобы задать подключение по умолчанию. Дайте ему Cosmetic name «Connection» с типом свойства «Single digital pin» и переменной pin.

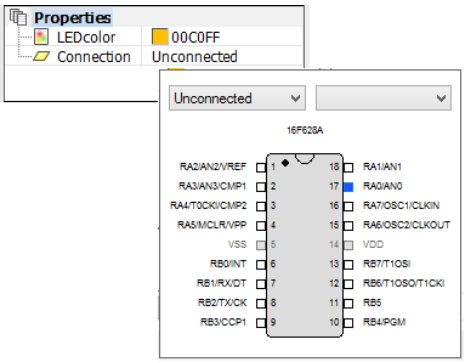




В заключение сконфигурируйте эти свойства.

Чтобы это сделать:

- Щёлкните по цвету, показанному для свойства color.
- Выберите цвет из палитры, которая появляется.
- Щёлкните по слову «Unconnected» справа от свойства pin.
- Выберите подходящий вывод, как RA0, на изображении микросхемы.

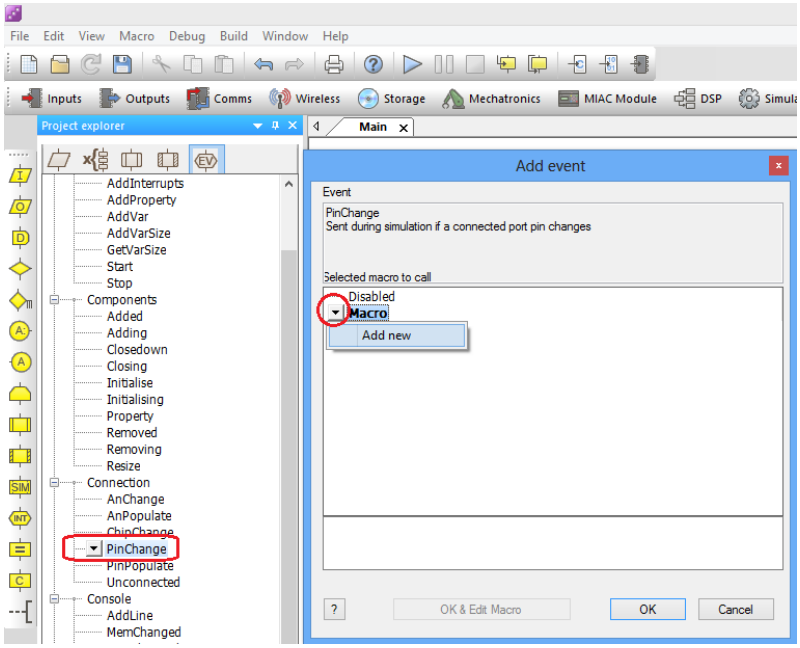


Реакция, когда меняется состояние вывода

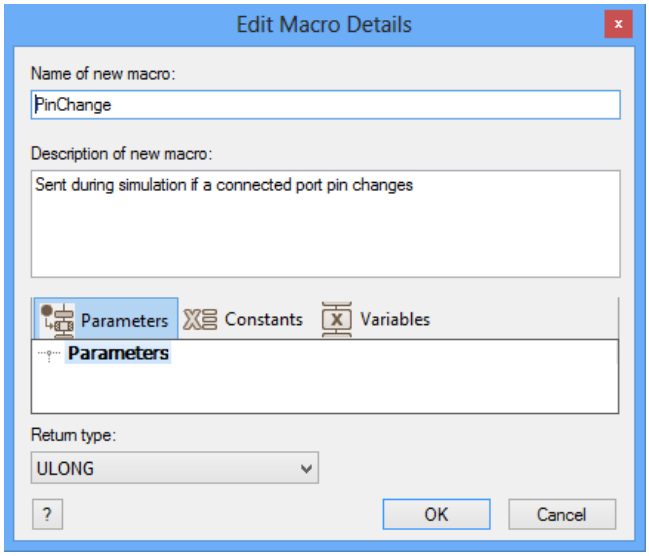
Этим затрагивается два макроса. Один, называемый PinChange, распознаёт, когда вывод меняет логический уровень. Второй, называемый Property, сказывается на свойствах LED, в данном случае меняется цвет.

Вначале создадим макрос PinChange.

- Откройте Project Explorer, используя меню View, если проводник по проекту не открыт.
- Дважды щёлкните по разделу «Connection -> PinChange». Откроется диалог «Add event».

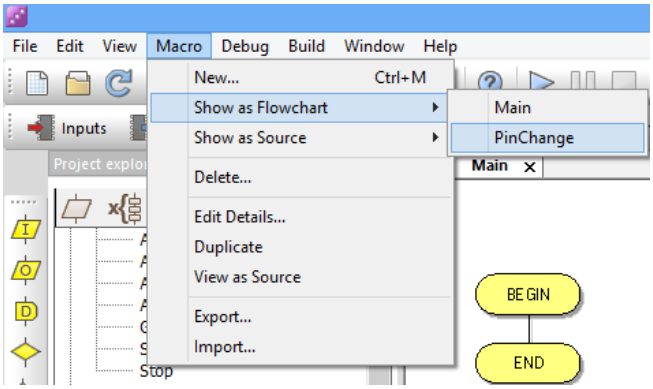


- Поместите курсор мышки на иконке слева от названия Macro. Появится выпадающее меню.
- Щёлкните по стрелке вниз и выберите опцию «Add new». Откроется диалоговое окно «Edit Macro Details».
- Измените имя макроса на PinChange.
- Щёлкните по кнопке **OK**.



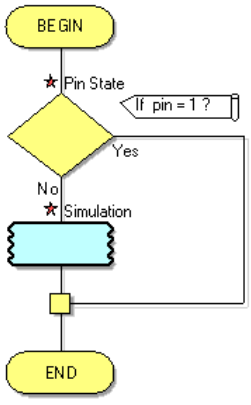
Затем «обустроим» макрос PinChange.

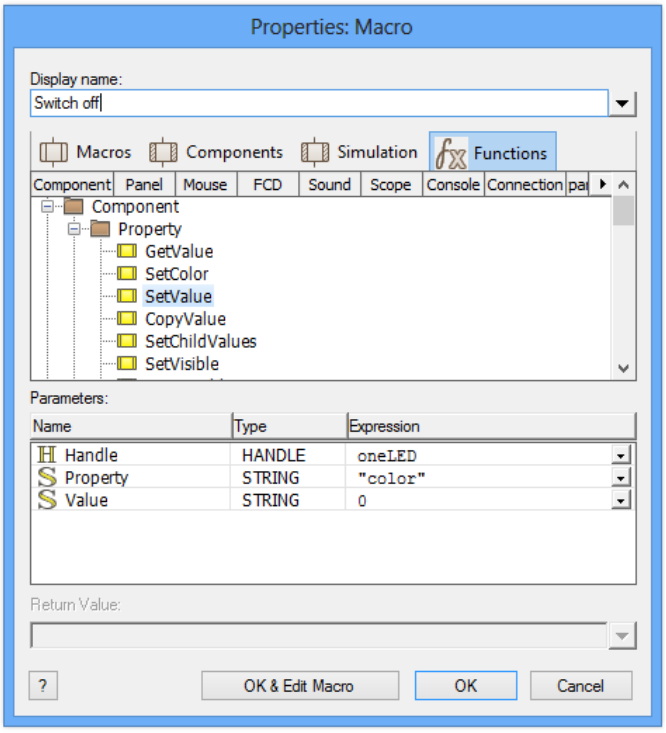
Выберите Show as Flowchart из Macro основного меню, и выберите PinChange макрос.



Добавьте следующие детали:

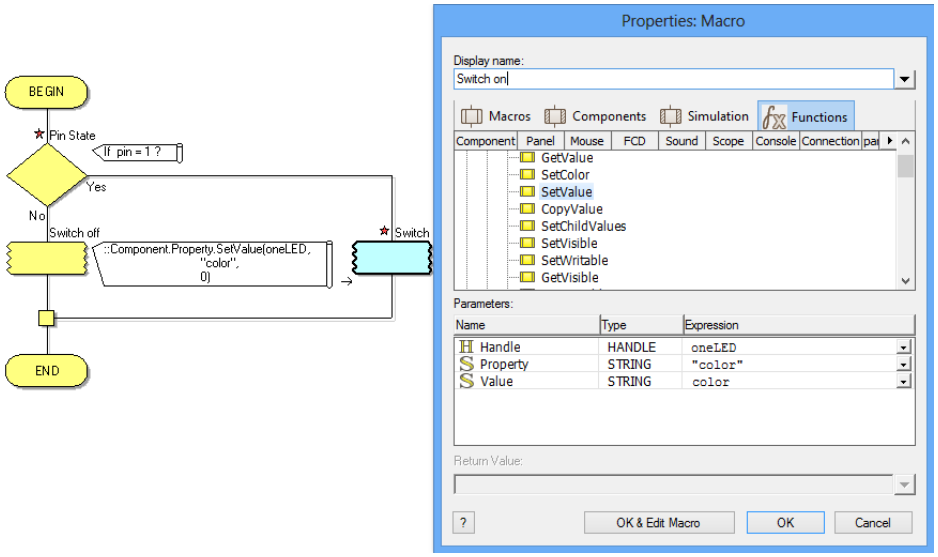
- Добавьте иконку Decision с панели программных компонентов, переименуйте её в Pin state, задайте условие If pin = 1.
- Перетащите команду SIM (иконку панели программных компонентов) в ветку «No». Сконфигурируйте команду так:
  - Переименуйте её в Switch off.
  - Щёлкните по этикетке Fuctions.
  - Раскройте раздел Component, а затем Property.
  - Щёлкните по команде SetValue, заполните её, добавив oneLED как Handle, “color” (включая кавычки) как Property и 0 (ноль) как Value. Затем щёлкните по кнопке **OK**.



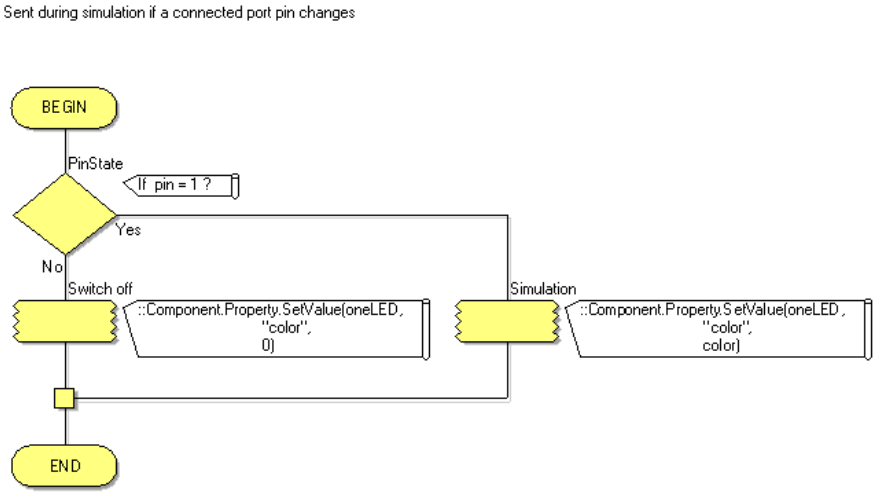


- Щёлкните по элементу SIM в макросе, скопируйте его и вставьте в ветку «Yes».
- Измените конфигурацию следующим образом:
  - Переименуйте команду в Switch on.

- Измените Value с нуля на color (без кавычек).
- Затем щёлкните по **ОК**.



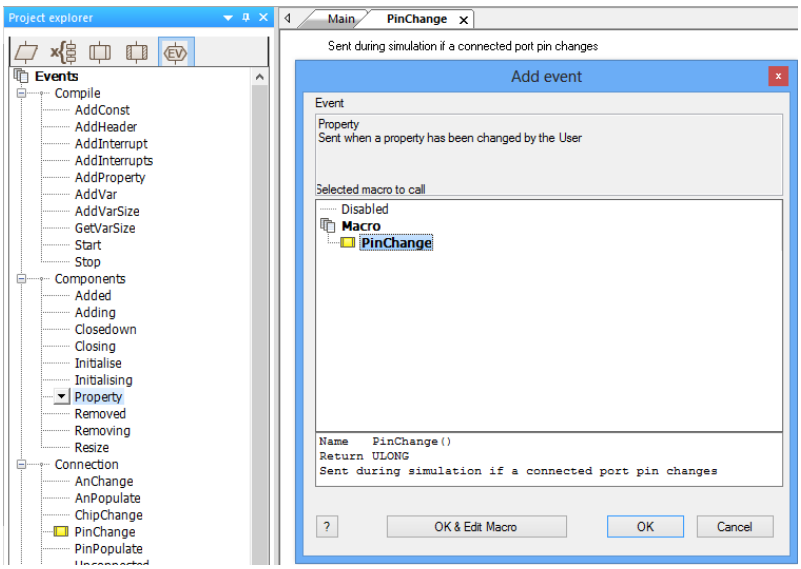
Подпрограмма (макрос) будет выглядеть следующим образом:



Обновление LED

Когда состояние вывода LED меняется, вы хотели бы, чтобы цвет LED тоже менялся. Это выполняется с помощью макроса «Component/Property». Этим будет заново использован код уже созданный в макросе PinChange.

- Откройте Project Explorer, как и раньше.
- Щёлкните по иконке Events.
- Раскройте раздел Components.
- Дважды щёлкните по команде Property, чтобы открыть диалоговое окно Add event.
- Щёлкните по макросу PinChange (для повторного использования), а затем щёлкните по **ОК**.



Добавление интерфейса

Компонент не может использоваться, если ни у кого к нему нет доступа. Всё, что нужно сделать, это создать два макроса, LEDon и LEDoff, которые устанавливают и сбрасывают состояние вывода. Это станет *общим интерфейсом* для компонента.

- Щёлкните по Macro в основном меню и выберите New (Macro > New...). Этим откроется диалоговое окно создания нового макроса.
- Назовите макрос LEDoff и щёлкните по кнопке **OK**. В рабочем поле появится шаблон новой подпрограммы.

  Настройте его следующим образом:

  - Перетащите иконку Calculation в подпрограмму.
  - Дважды щёлкните по ней и переименуйте в Reset pin.
  - В окне элемента впечатайте pin = 0 и щёлкните по **OK**.

- Создайте второй макрос таким же образом. Его назовите LEDon. В содержимое иконки calculation, названной Set pin, внесите pin = 1.

Но это не всё. Следующее упражнение называется: Exercise - Exporting the LED Component.

2. Экспорт компонента LED

Теперь, когда мы создали светодиод в предыдущем примере, мы можем экспортировать компонент для постоянного использования.

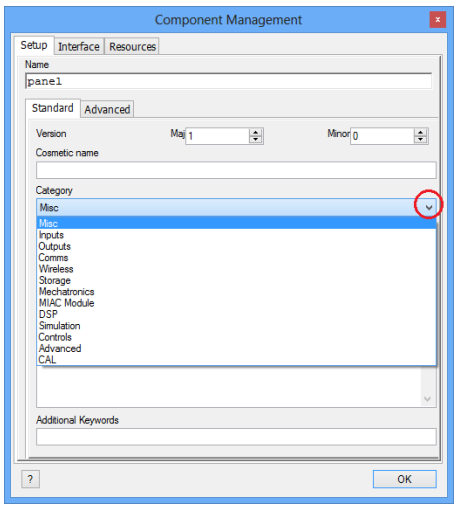
Предварительные замечания

- Создайте компонент LED, как показано в статье Exercise - Create an LED Component.

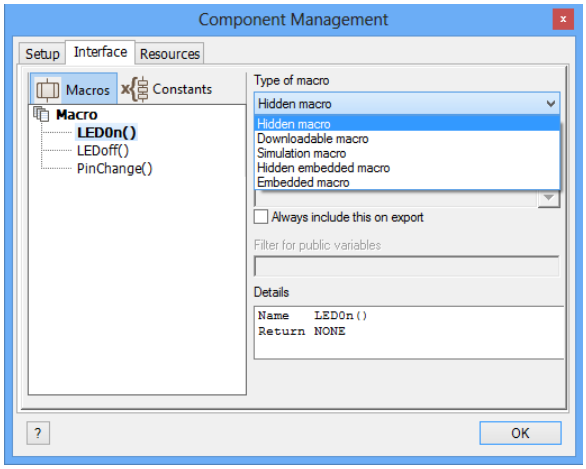
Объявление интерфейса

Перед экспортом компонента необходимо декларировать (объявить) интерфейс. Интерфейс – это подборка макросов и файлов, доступных в общем пользовании. Объявление интерфейса означает решение, какие из подпрограмм компонента сделать доступными пользователю, и решить, будут ли они подходить для симуляции или для загрузки.

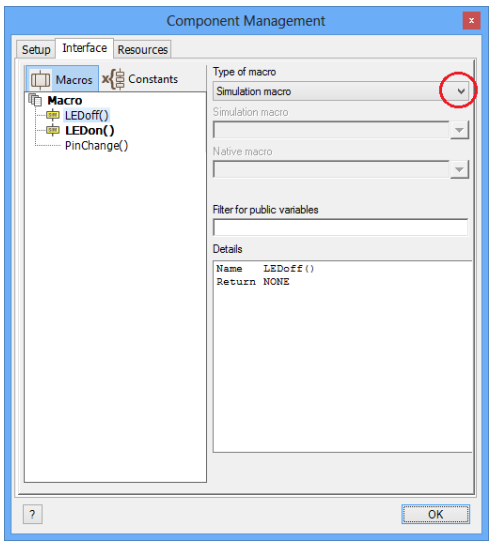
- Переместите курсор мышки на системную панель и щёлкните правой клавишей мышки.
- Выберите раздел Export component. Откроется диалог Component Management (обслуживание компонента). Здесь повторяются установки, Settings Manager, Interface Manager и Resource Manager доступные в Properties Panel.



- Сконфигурируйте его для компонента LED следующим образом:
  - Щёлкните по закладке Setup.
    - Сопоставьте компонент с категорией, щёлкнув по стрелке вниз рядом с надписью Category. В нашем случае компонент лучше поместить в категорию Outputs.
    - Установите флажок (щелчком левой клавиши мышки) в окошке Use an image of the panel.
  - Щёлкните по закладке Interface.
  - Задайте макросы LEDoff и LEDon как макросы Component, выбрав каждый из них последовательно и щёлкнув по Component macro (этого нет в списке, есть Simulation macro) в выпадающем меню Type of macro.



  - Аналогично задайте макрос PinChange как Hidden macro.
  - В данном случае нет необходимости использовать закладку Resources, поскольку нет дополнительных файлов для сохранения вместе с компонентом.



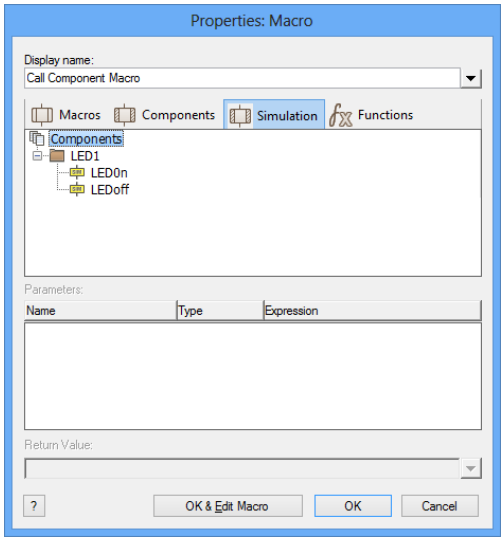
- Щёлкните по кнопке **ОК**.

Сохранение компонента

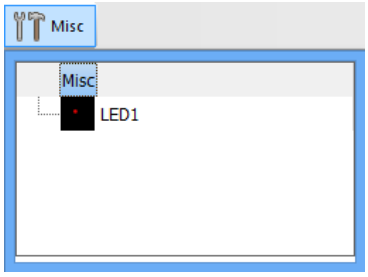
- Вы можете получить запрос на сохранение программы. Если хотите, сохраните.
- Откроется диалог Save As. Это автоматически укажет на место, где Flowcode ищет компоненты. Выберите подходящее имя для компонента.
- Щёлкните по кнопке Save, и компонент сохранится с расширением .fcpx.

3. Импорт компонента

- Создайте новый файл Flowcode.
- В дальнейшем подразумевается, что вы создали и экспортировали компонент, как описано выше.
- Откройте группу Outputs, где вы сохранили созданный компонент (если вы не указали эту группу, то компонент появится в группе Misc).
- Выберите компонент по имени (я использовал LED1), под которым вы его сохранили, перенесите его на системную панель. Вы можете его использовать, как и другие электронные компоненты.

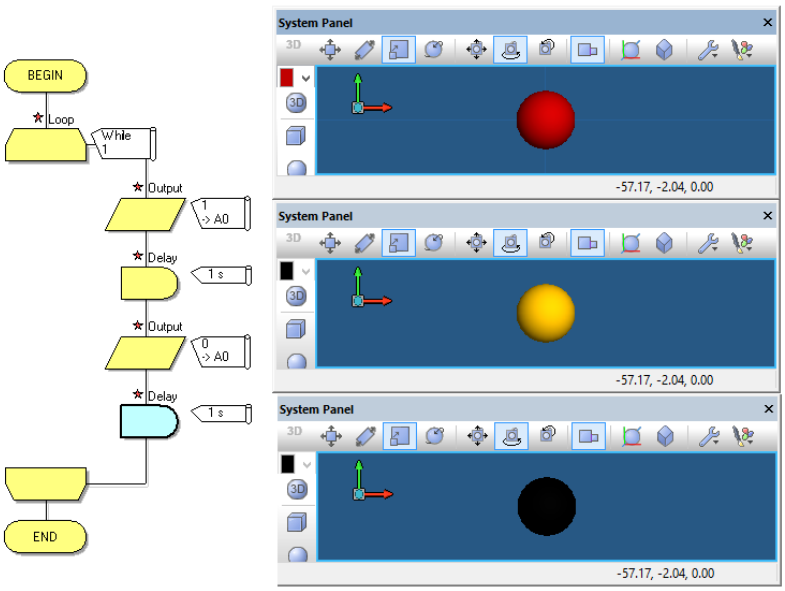


Если не добавлять новый компонент в группу элементов для выхода, то при новой загрузке он появляется в прежде пустой группе Misc.



Новый компонент можно добавить на системную панель, можно менять его свойства, и он в компонентных макросах отображается так, как показано в рассказе об импорте нового компонента. Отображается на закладке симуляции, но не на закладке компонентов.

Вот результат использования вновь созданного компонента с простейшей программой:



4. Добавление объектов на системную панель

Это упражнение начинает процесс создание индикаторной панели для автомобиля.

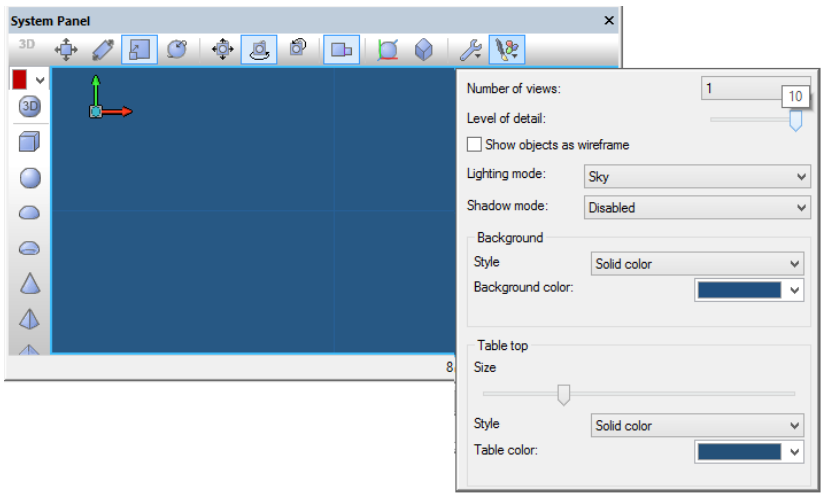
Новая программа

- Начните новый проект, используя предопределённый микроконтроллер.
- Убедитесь, что системная панель видима. Если необходимо, используйте раздел View основного меню, где отметьте System Panel.

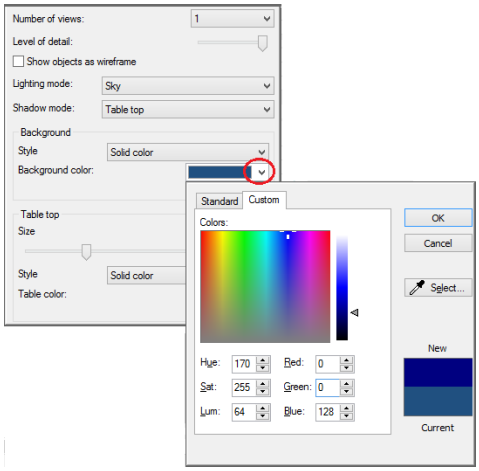
Основные опции

- Щёлкните по закладке the General options 

  Сделайте следующие изменения:
  - Передвиньте ползунок Level of detail в крайнее правое положение (к «10»).



- Выберите Table top в shadow mode.
- Выберите Background стиль как Solid color и задайте цвет тёмно-синий (Red=0, Green=0, Blue=128 на закладке Custom).



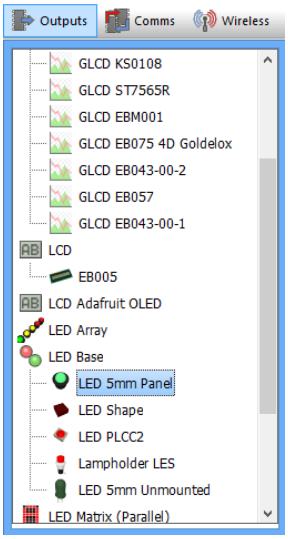
- Выберите Table top стиль Solid color и задайте цвет светло-синий (Red=0, Green=70, Blue=255 на закладке Custom).

Настройка сетки и привязки

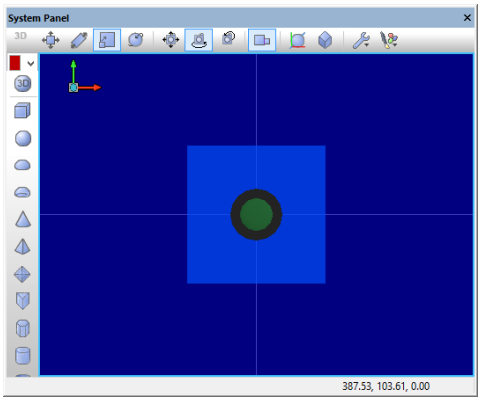
- Щёлкните по закладке Grid and Snap options 
  Сделайте следующие изменения:
  - Задайте привязку к Table top, щёлкнув по этикетке Table top.
  - Оставьте Show для линий сетки не отмеченным.

Добавление светодиода

- Выберите LED, такой как LED 5mm Panel, из группы Outputs.
- Щёлкните по стрелке рядом с названием компонента и выберите Add to system panel.

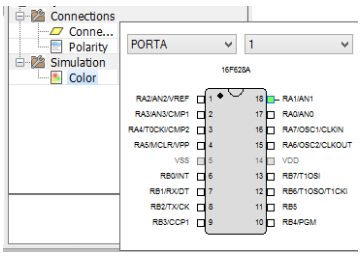


- Щёлкните где-нибудь по системной панели и выберите опцию Zoom to 100%. Системная панель должна походить на показанную ниже.



Измените свойства LED

- Щёлкните по LED, чтобы выделить его. На панели свойств появятся два свойства для LED – color и pin.
- Свойство color установлено как зелёный (0x80FF57). Щёлкните по этому значению, и измените цвет на оранжевый (0x0080FF). Вы можете заметить только небольшое изменение в виде LED, поскольку он сейчас выключен.
- Свойство pin определяет, к какому выводу микроконтроллера LED подключён. По умолчанию светодиод отключён. Щёлкните по этикетке, появится микроконтроллер с выводами. Щёлкните по биту 1 (RA1/AN1), чтобы задать соединение.



- Панель свойств также даёт информацию о размерах и позиции светодиода. Щелчок правой клавишей мышки по этикеткам даёт представление о возможных изменениях.

Увеличение

Светодиод может выглядеть очень маленьким на системной панели. Вы можете увеличить его. Удерживайте клавишу **Ctrl** на клавиатуре, курсор изменит свой вид, и появится иконка  ..

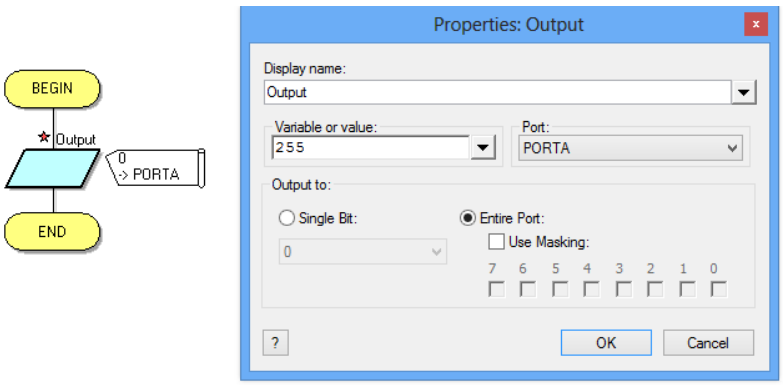
Поместите курсор на эту иконку. Теперь, когда вы перемещаете курсор мышки вверх, вид компонента увеличивается, а когда вниз, уменьшается. Подстройте вид так, чтобы Table top занимал большую часть системной панели, делая LED более заметным.

Включение

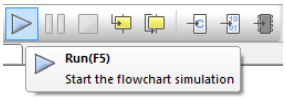
Перенесите в программу иконку Output.

Дважды щёлкните по ней и измените заданное значение Variable or value с 0 на 255. Этим установится логическая 1 на всех выводах PORT A.

Программа примет вид, показанный ниже:

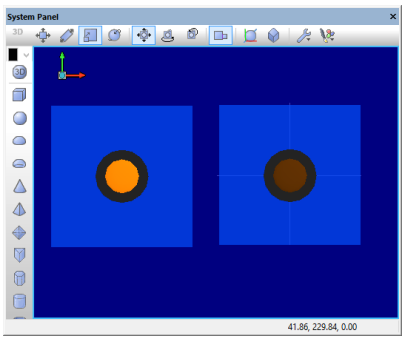


- Запустите симуляцию, щёлкнув по иконке Run:



Светодиод зажжётся.

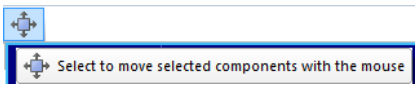
- Дважды щёлкните по иконке Output ещё раз и измените значение на 0. Теперь при симуляции программы LED погаснет.
- Сохраните программу как «System_Panel_Add_LED».



5. Системная панель – управление формами

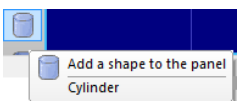
Откройте проект LED

- Откройте проект System_Panel_Add_LED, который вы создали в предыдущем упражнении.
- Проверьте, что на панели свойств отображаются размеры как World size.
- Переместите LED в правый верхний угол, щёлкнув по иконке Move with the mouse.



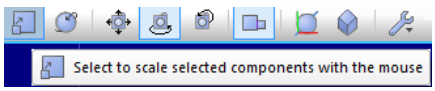
Использование инструментальной панели

- Щёлкните и перетащите цилиндр с инструментальной панели на системную панель.



- Щёлкните, чтобы выделить цилиндр на системной панели. Вы увидите пять свойств, перечисленных в панели свойств: Shape, Color, Outline, Thickness и Image.
- Предустановленное свойство Color – это красный (0x0000C0). Щёлкните по его значению и смените цвет на чёрный (0x000000).

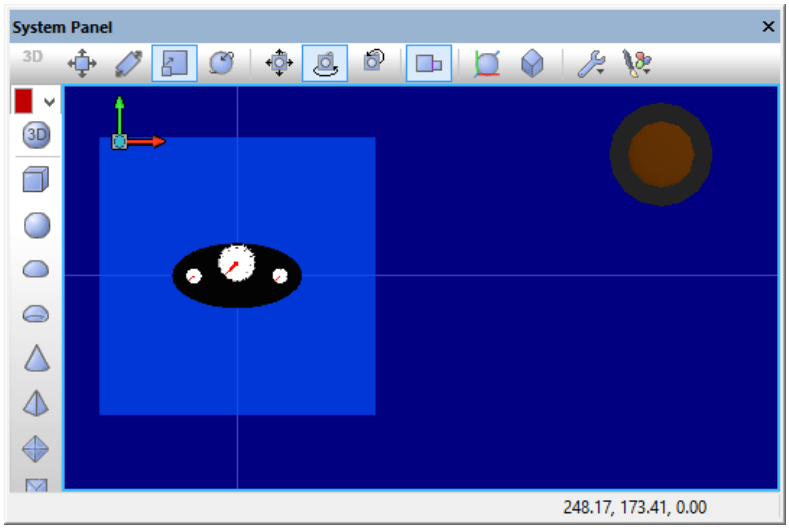
Изменение фигуры инструментальной панели



- Щёлкните по иконке Scale with mouse в верхней части горизонтальной панели.
  Теперь, когда вы выделите цилиндр, по краям появляется восемь ручек. Они позволяют вам придать новый вид цилиндру, перетаскивая подходящую ручку.
- Перетащите горизонтальную ручку, пока ширина цилиндра не станет 120 мм.
- Перетащите вертикальную ручку, пока высота цилиндра не станет 60 мм.
- На панели свойств задайте толщину (De...) 3 мм.
- Отцентрируйте панель, задав координаты позиции в X=0, Y=0, Z=0 на панели свойств.
- Чтобы сделать панель более реалистичной, найдите ADC dial в группе компонентов Inputs инструментальной панели дополнительных компонентов. (Этот компонент можно использовать как устройство ввода, но сейчас он будет использован в декоративных целях).
- Щёлкните по стрелке вниз слева от названия, выберите вариант Add to system panel.
- Щёлкните по циферблату на системной панели, чтобы выделить его, и задайте следующие свойства:
  'Coordinates': X=0, Y=25, Z=5;
  'World size': 'Wi...'=35, 'He...'=35, 'De...'=5;

- Теперь добавьте ещё два циферблата тем же путём.
- Задайте координаты первого: X=-100, Y=0, Z=5, a World size Wi...=15, He...=15, De...=5.
- Задайте для другого: Coordinates X=100, Y=0, Z=5, a World size Wi...=15, He...=15, De...=5.

Системная панель должна выглядеть так:



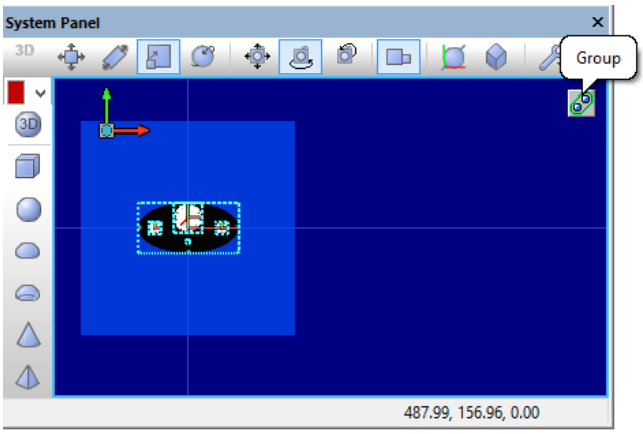
<u>Примечание:</u> При переводе упражнений из Help одновременно повторялись все предложенные операции. Но не всегда получается вид панели, как это приведено в

оригинале. Чтобы получить нечто похожее, приходится перетаскивать циферблаты. Центр смещён, а размеры светодиода велики. Видимо, что-то сделано не так в предыдущем примере, но это поправимо.

Добавление Led на панель

- Теперь переместим LED назад, выделив его и задав его новое положение координатами: X=0, Y=-60, Z=0.

  Проделывая это, убедитесь, что светодиод находится поверх нашей инструментальной панели.
- Проведите курсором поверх всех объектов и сгруппируйте их, щёлкнув по появившейся иконке Group  .



- Осталось сохранить проект, как System_Panel_Add_MountedLED.

Изменение точки наблюдения

- Flowcode позволяет вам наблюдать объекты с разных направлений, изменяя положение камеры (viewpoint).

  Первый метод – использование красной, зелёной и синей стрелок в верхнем левом углу системной панели.
  Вначале щёлкните по наконечнику зелёной стрелки.
  Это даст вам возможность увидеть объект с «зелёного» направления.
  Теперь повторите это с красной стрелкой, чтобы увидеть с «красного» направления.
  И, наконец, щёлкните по синей стрелке, что вернёт вас к первоначальной точке обзора.

- Есть три метода управления «камерой», которые дают лучший результат.

Первый – это управление pan, которое позволяет вам перемещать вид на объект. Чтобы это увидеть, щёлкните по иконке pan. Удерживайте клавишу **Ctrl** и щёлкните мышкой где-нибудь на системной панели.

Когда вы перемещаете курсор вверх и вниз или из стороны в сторону, точка зрения соответственно смещается.
Заметьте, что координаты объекта не меняются. Объект остаётся на том же месте, но меняется вид с разных мест.

Второй и третий методы – это управление поворотом камеры.

Щёлкните по первому из них. Удерживайте клавишу **Ctrl** и щёлкните мышкой где-нибудь по системной панели.

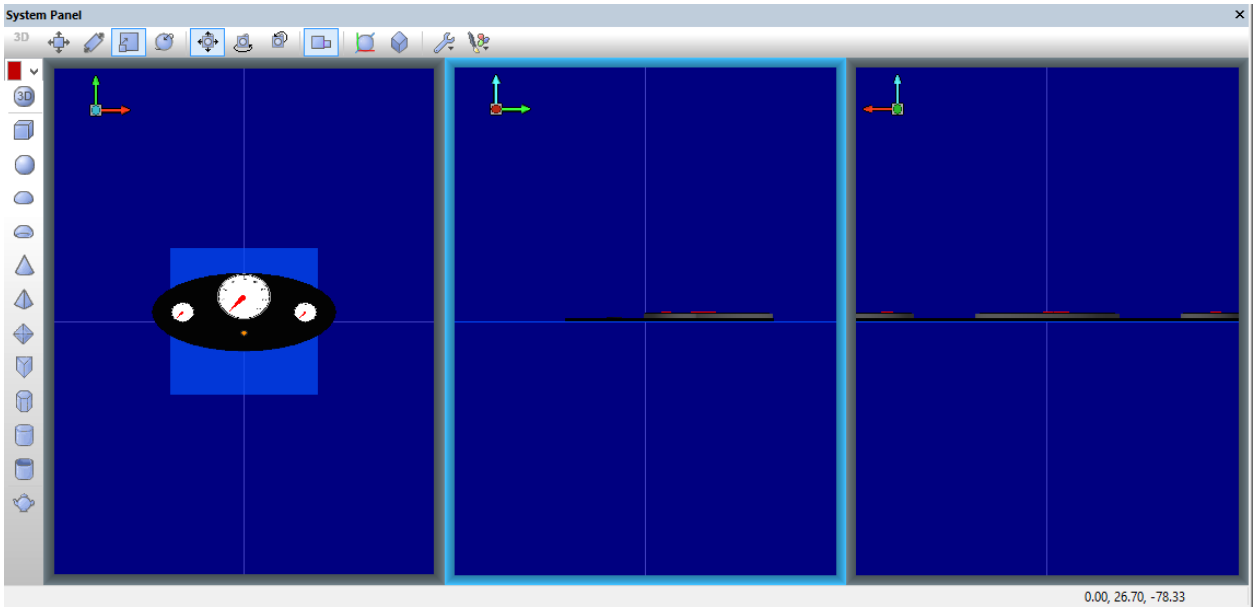
Когда вы перемещаете курсор, объект тоже поворачивается, хотя, если вы посмотрите на свойство Rotation панели свойств, то увидите, что установки не меняются. Вновь, меняется только точка зрения.

Теперь попробуйте то же самое с другим управлением.

Множество точек зрения

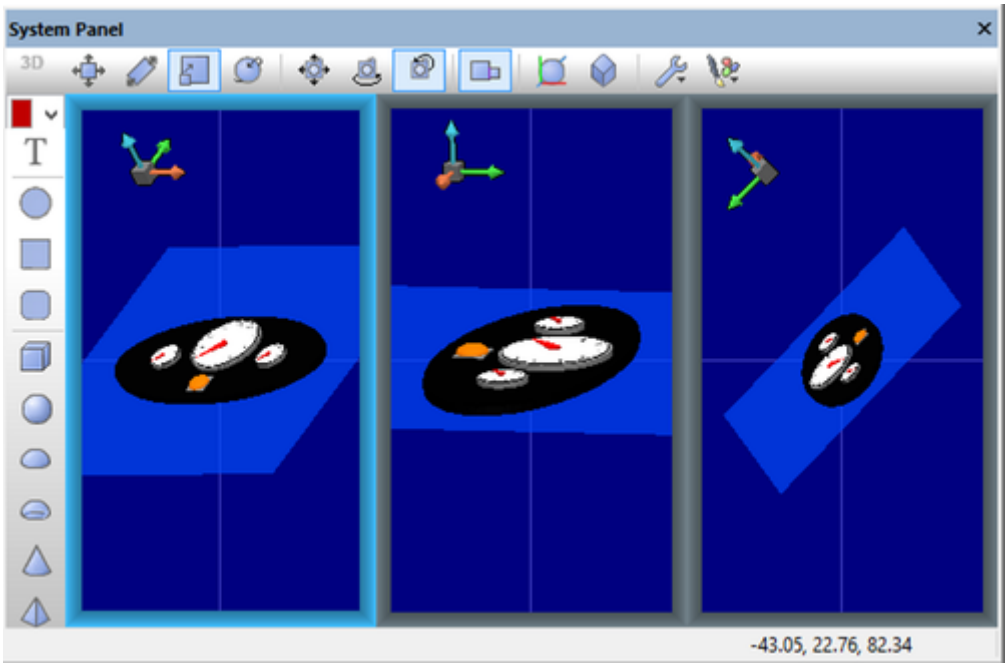
- Интересная особенность программы Flowcode 6 – это возможность рассматривать компоненты с разных направлений одновременно.
- Чтобы это увидеть:

Щёлкните по иконке General options.
В разделе Number of views щёлкните по стрелке вниз и выберите 3.
Щелчком по подходящему наконечнику стрелки сделайте точки обзора такими, как это показано ниже.
В зависимости от размера и формы системной панели три вида могут разделяться вертикально или горизонтально. Перетащите один край системной панели в сторону, чтобы увидеть этот эффект.



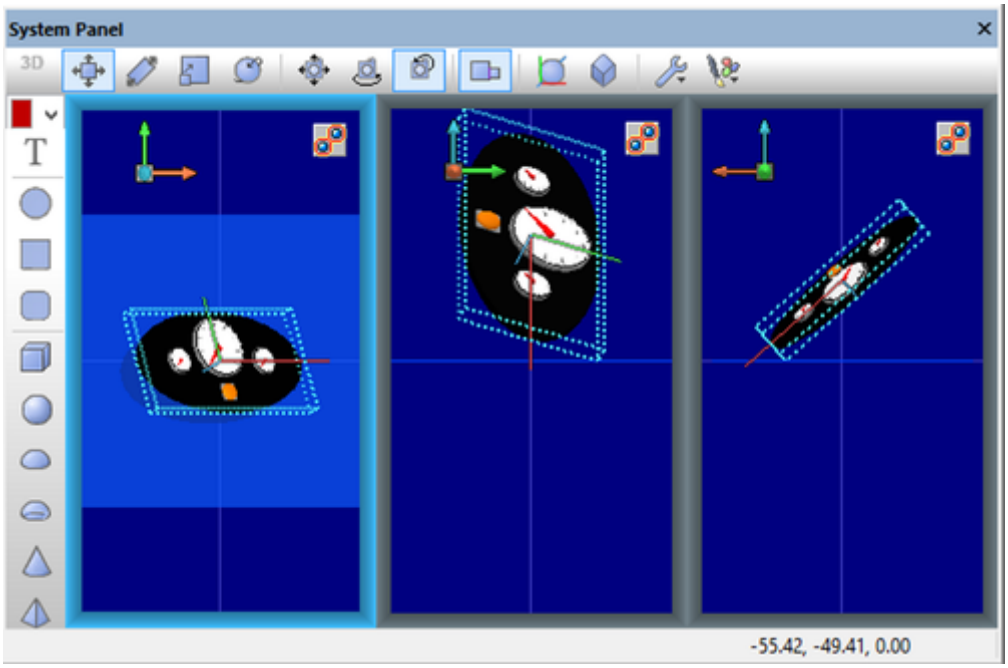
- Вы можете приспособить точку зрения для каждого вида индивидуально, оперируя позицией камеры.

Пример того, как это выглядит, ниже – обратите внимание на положение красной, зелёной и синей стрелок в каждом случае:



- Альтернативно вы можете внести изменения в каждый из объектов, в этом случае все три вида изменятся сразу.

В следующем примере инструментальная панель была повёрнута:



Вы можете видеть, что произошло, взглянув на панель свойств. Изменение собственно объектов меняет их свойства, как Coordinates и World size.

Перемещение камеры и её повороты меняют точку зрения, но не сказываются на свойствах самих объектов.

6. Добавление объектов на панель управления

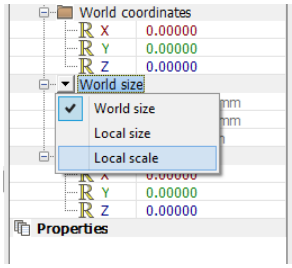
Это упражнение показывает, как добавить переключатель на Dashboard Panel (панель управления), и использовать её для управления LED на инструментальной панели автомобиля, которую мы создали ранее. Индикатор предупреждает водителя о том, что фары включены.

Смысл этой переделки в том, чтобы на операциях включения и выключения индикаторов не отражались повороты объекта на системной панели, дающие разные точки зрения на объекты.

Откройте предыдущий проект

- Откройте проект, который мы создали в предыдущий раз.
- Теперь задайте на панели свойств отображение размеров, как Scale.

  Чтобы это сделать, щёлкните по стрелке рядом с World size и выберите Local scale.



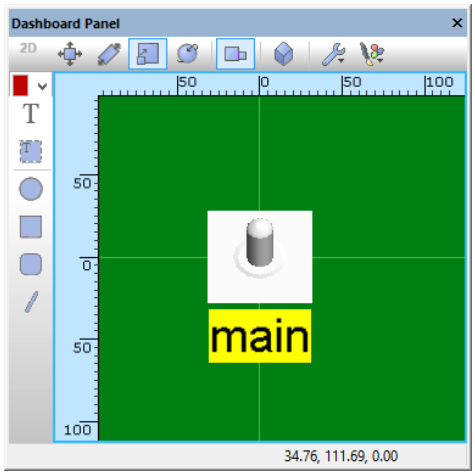
Настройка панели управления

- Убедитесь, что Dashboard Panel видима. Если нет, тогда воспользуйтесь разделом View основного меню, щёлкнув по Dashboard Panel.
- Выберите подходящий цвет для фона, например, тёмно-зелёный (red=0, green=128, blue=0), щёлкнув по закладке General options  .

Добавление выключателя

- Щёлкните по иконке Inputs инструментальной панели и найдите выключатель Toggle Metal Panel.
- Щёлкните по стрелке вниз рядом с компонентом и выберите Add to dashboard panel.

  Выключатель появится в центре панели управления.
  Щёлкните по нему, чтобы выделить компонент. Его свойства теперь показаны на панели свойств.
  Щёлкните по свойству pin. Появится микроконтроллер с выводами.
  Щёлкните по выводу RB0/Int, чтобы соединить выключатель с битом 0 порта B.



- Добавьте этикетку для идентификации компонента (будут и другие!).

  Чтобы это сделать:
  Щёлкните по цветному прямоугольнику в верхней части вертикальной панели слева Dashboard Panel.
  Выберите чёрный цвет.
  Щёлкните по иконке текста (с буквой «T») и перетащите её на панель управления.
  Компонент может оказаться «невидимкой» в этот момент, но на панели свойств компонент отобразится как label со списком свойств – Color, Background, Font и Text.
  По умолчанию текст содержит Please Change Caption. Щёлкните по нему и перепишите его как «main».
  Дополнительно измените:
  цвет фона на жёлтый (0x00FFFF);
  координаты на X=0, Y=-10, Z=0;
  масштаб на Width=5, Height=5, Depth=2.
- Dashboard Panel теперь должна стать похожа на показанную выше (выключатель пришлось увеличить).

Программа

- Добавьте в программу бесконечный цикл, а внутри:

  - Перетащите иконку Input с панели программных компонентов.

    Дважды щёлкните по ней, чтобы изменить конфигурацию.
    Переименуйте в Check the switch.
    Щёлкните по стрелке вниз в конце текстового поля Variable.
    Затем щёлкните по стрелке вниз перед Variables в появившемся окне и выберите Add new.
    В новом диалоговом окне Create a New Variable введите имя input для новой переменной.
    Оставьте поля Initial value и Description пустыми и оставьте тип переменной как byte.
    Щёлкните по кнопке **OK**.
    В окне свойств компонента Input в поле Variable введите имя input.
    Выберите PORT B в качестве порта, а для Input from: Single Bit, 0.
    Щёлкните по кнопке **OK**.

- Выберите и перетащите в программу иконку Decision.

  Дважды щёлкните по ней, чтобы настроить её.
  Переименуйте её в Is it on?.
  В окне If впечатайте input=1, используя переменную input, созданную ранее.
  Щёлкните по кнопке **ОК**.

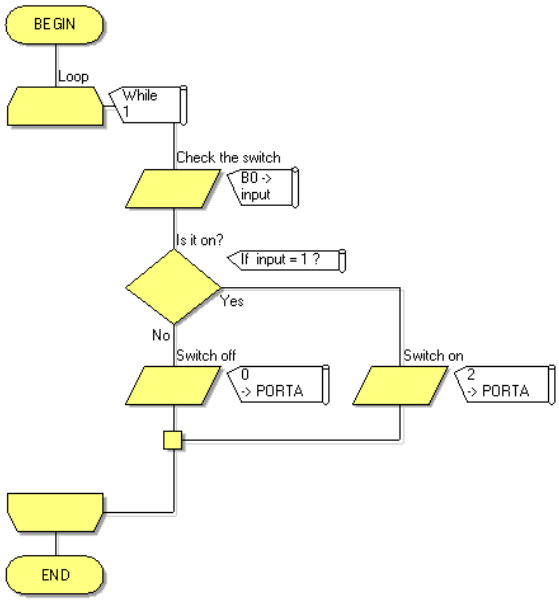
- Выделите и перетащите иконку Output в ветку Yes компонента Decision.

  Дважды щёлкните по ней, чтобы сконфигурировать.
  Переименуйте её в Switch on.
  Введите «2» в поле Variable or value.
  Оставьте другие настройки, как PORTA и Entire Port, как они есть.
  Щёлкните по кнопке **ОК**.
  (Когда выводится десятичное 2 в порт A, бит 1 порта установится в логическую 1.
  Поскольку LED подключён к биту 1, он включится).

- Перетащите второй компонент Output в ветку No компонента Decision.

  Дважды щёлкните по нему, чтобы сконфигурировать.
  Переименуйте его в Switch off.
  Введите 0 в поле Variable or value.
  Выберите PORTA и Entire Port.
  Щёлкните по кнопке **ОК**.

- Программа должна стать похожа на приведённую ниже. Сохраните её как Headlight warning.



7. Управление множеством объектов

Это упражнение покажет, как добавить ещё два светодиода на инструментальную панель автомобиля, каждый управляемый отдельным выключателем на Dashboard Panel. Они будут предупреждать водителя о включении подфарников и о том, что ремень безопасности не пристёгнут.

Откройте предыдущий проект

В нём использован один выключатель, названный main, и включающий оранжевый светодиод на инструментальной панели автомобиля.

Добавление светодиодов

- Добавьте ещё два светодиода LED 5mm Panel на системную панель.

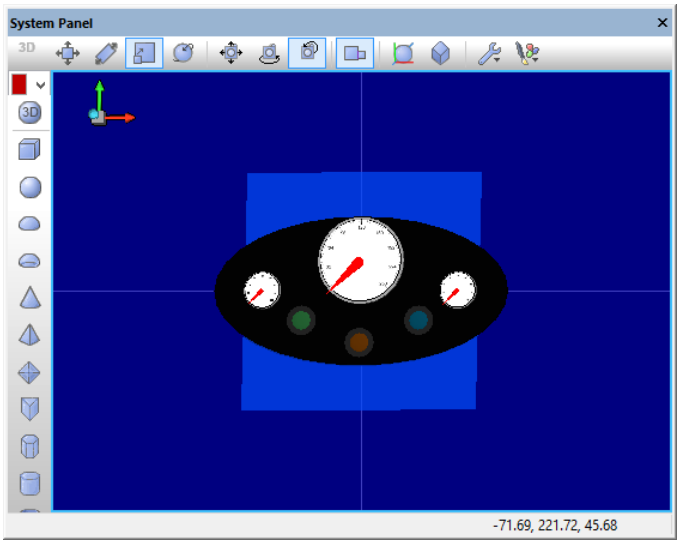
  Сделайте это с помощью Outputs инструментальной панели, как вы делали это раньше, или скопируйте и вставьте первый светодиод.

- В любом случае:

  задайте для первого координаты X= -60, Y=-55, Z=-1.75 и цвет зелёный;
  для второго координаты X= 60, Y=-55, Z=-1.75 и цвет синий.

  Все три светодиода должны иметь одинаковые размеры (Wi...=32, He...=32, De...=10). Подключение: RA0 – оранжевый; RA1 – зелёный; RA2 – синий.

  Системная панель должна принять вид:



Добавление выключателей

- Добавьте ещё два Toggle Metal Panel выключателя на Dashboard Panel.

  Как и выше, вы можете воспользоваться группой Input или скопировать и вставить их.
  В любом случае первому задайте координаты X= 30, Y=5, Z=-2, а второму X= 60, Y=5, Z=-2.

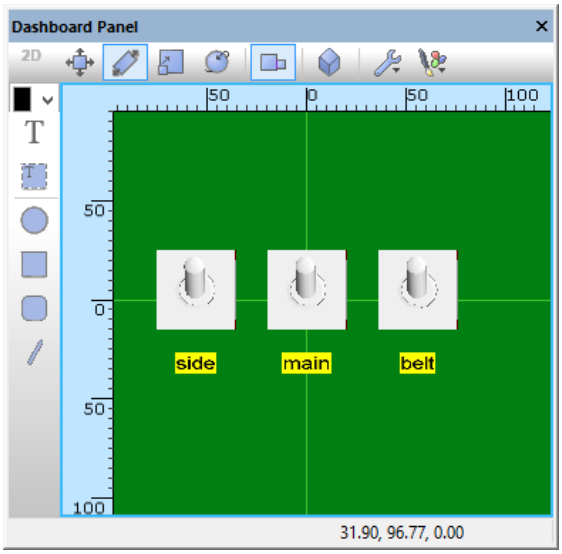
Все три выключателя должны иметь одинаковые размеры (Wi...=2, He...=2, De...=1, эти размеры приходится сделать гораздо больше).

- Создайте две этикетки, перетаскивая их с Dashboard panel, иконка «T», или скопируйте и вставьте ту, что есть.

  В любом случае переименуйте первую в «side», а вторую в «belt». Задайте первой координаты X=30, Y=-15, Z=0.5, а второй X=60, Y=-15, Z=0.5.

  Все три должны иметь одинаковые размеры (Wi...=10, He...=10, De...=2, что не очень получается).

  Dashboard Panel должна походить на следующую:



Подключение выключателей

Оригинальный выключатель main подключен к PORT B, бит 0.

- Подключите выключатель side к PORT B бит 1, а выключатель belt к PORTB бит 2.

Бинарная арифметика

Перед тем, как перейти к созданию программы, небольшое замечание по двоичной арифметике. Выключатели не должны «прочитываться» как отдельные сущности. Вместо этого они могут использоваться для управления первыми тремя битами бинарного числа, соответствующего состоянию PORT B. Таблица ниже показывает входное двоичное число, зависящее от комбинации включённых выключателей:

belt bit 2	main bit 1	side bit 0	binary no.	decimal no.
0	0	0	000	0
0	0	1	001	1
0	1	0	010	2
0	1	1	011	3
1	0	0	100	4

1	0	1	101	5
1	1	0	110	6
1	1	1	111	7

Как это работает:

Выключатель «belt» имеет десятичное значение 4, поскольку управляет третьей двоичной цифрой, и также значит 2^2 (=4).
Выключатель «main» имеет десятичное значение 2, поскольку управляет второй двоичной цифрой, и также значит 2^1 (=2).
Выключатель «side» имеет десятичное значение 1, поскольку управляет первой двоичной цифрой, и значит также 2^0 (=1).

Когда нажато более одного выключателя, генерируемое число (либо двоичное, либо десятичное) эквивалентно сумме этих значений. Таким образом, когда нажаты оба выключателя belt и side, воспроизводится число 5.

Программа Flowcode «чувствует» это число, используя иконку Switch.

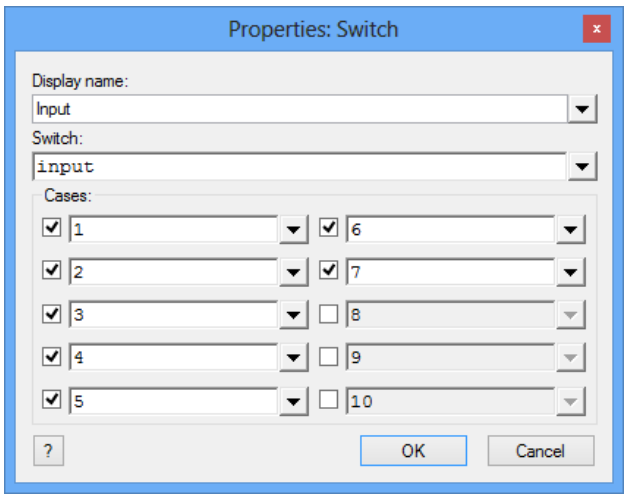
Создание программы

- Модифицируйте предыдущую программу следующим образом:
  - Дважды щёлкните по иконке Input, чтобы открыть диалоговое окно.
    - Измените Input from с Single Bit на Entire Port. Щёлкните по кнопке **ОК**.

Теперь программа читает все устройства, соединённые с портом B, и вводит число, генерируемое тремя выключателями.

- Щёлкните по иконке Decision в программе, чтобы выделить её и удалить (нажав, например, на клавиатуре клавишу **Delete**).
- Перетащите иконку Switch вместо неё.
  - Дважды щёлкните по иконке, чтобы открыть диалоговое окно, затем:
    - Переименуйте её в Input.
    - Вставьте имя переменной input в окне под этикеткой Switch.
    - Есть восемь доступных комбинаций переключателей, дающих входное число до 111 (десятичное 7), так что установите флажки от 1 до 7. Щёлкните по **ОК**.

      Диалоговое окно должно быть похоже на то, что ниже:



- В ветку по умолчанию добавьте компонент Output, назовите его Switch off и сделайте выходное значение 0 для всего порта A.
- В ветку «=1» добавьте компонент Output, назовите его Switch on и сделайте выходное значение 1 для всего порта A.

Эта ветка активируется, когда включаются только подфарники side. Зелёный светодиод предупреждает об их включении.

- В ветку «=2» добавьте компонент Output, назовите его Switch on и сделайте выходное значение 2 для всего порта A.

Эта ветка активируется, когда включается только выключатель main. Оранжевый светодиод предупреждает, что фары включены.

- В ветку «=3» добавьте компонент Output, назовите его Switch on и сделайте выходное значение 3 для всего порта A.

Эта ветка активируется, когда включены оба выключателя side и main. Оба светодиода, оранжевый и зелёный, зажигаются.

- В ветку «=4» добавьте компонент Output, назовите его Switch on и сделайте выходное значение 4 для всего порта A.
- Затем добавьте компонент Delay с задержкой в 1 секунду.
- Добавьте вторую иконку Output, сконфигурируйте выходное значение 0 для всего порта A, чтобы выключить предупреждающие сигналы.
- И, наконец, добавьте второй компонент Delay с задержкой в 1 секунду.

Эта ветка активируется, когда включён только выключатель belt, поскольку ремень безопасности не застёгнут. Синий светодиод будет мигать.

- В ветку «=5» добавьте компонент Output, назовите его Switch on и сделайте выходное значение 5 для всего порта A.
- Затем добавьте компонент Delay с задержкой в 1 секунду.
- Добавьте вторую иконку Output, сконфигурируйте выходное значение 1 для всего порта A.
- И, наконец, добавьте второй компонент Delay с задержкой в 1 секунду.

Эта ветка активируется, когда активируются оба выключателя, side и belt. Программа оставляет гореть зелёный светодиод, а синий начинает мигать.

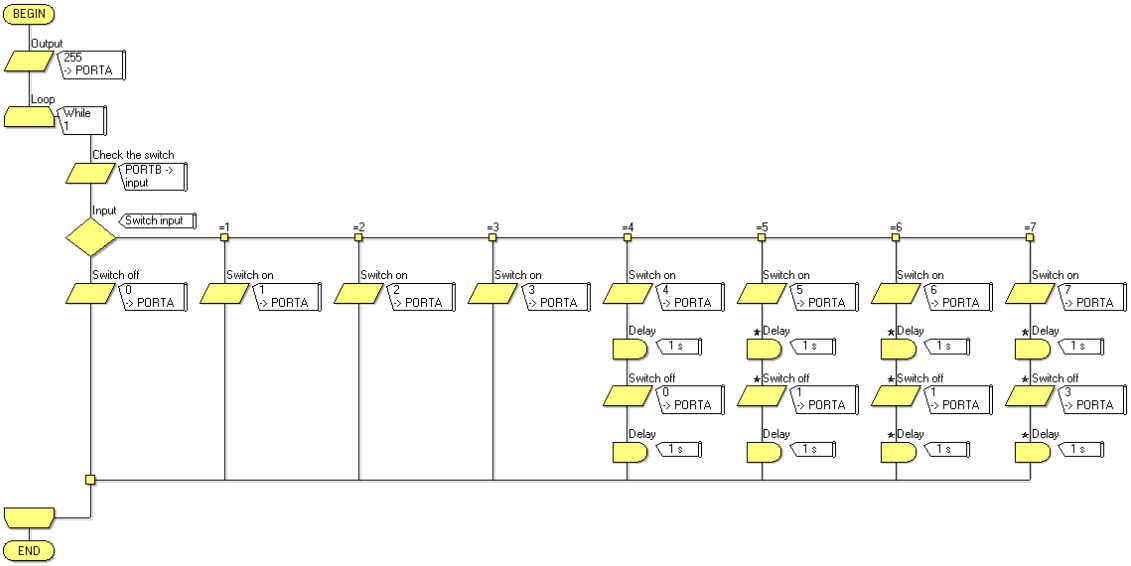
- В ветку «=6» добавьте компонент Output, назовите его Switch on и сделайте выходное значение 6 для всего порта A.
- Затем добавьте компонент Delay с задержкой в 1 секунду.
- Добавьте вторую иконку Output, сконфигурируйте выходное значение 2 для всего порта A.
- Затем добавьте второй компонент Delay с задержкой в 1 секунду.

Эта ветка активируется, когда выключатели main и belt включены. Программа оставляет включённым оранжевый светодиод, а синий начинает мигать.

- В ветку «=7» добавьте компонент Output, назовите его Switch on и сделайте выходное значение 7 для всего порта A.
- Затем добавьте компонент Delay с задержкой в 1 секунду.
- Добавьте вторую иконку Output, сконфигурируйте выходное значение 3 для всего порта A.
- Затем добавьте второй компонент Delay с задержкой в 1 секунду.

Эта ветка активируется, когда все три выключателя активированы. Зелёный и оранжевый светодиоды горят, синий светодиод мигает.

Программы выглядит подобным образом:



Запустите симуляцию программы, чтобы проверить правильность работы при разных комбинациях нажатых выключателей.

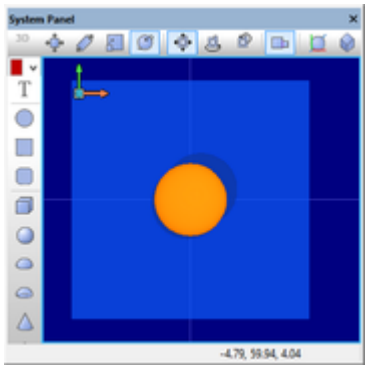
Поздравляем! Вы закончили разработку инструментальной панели автомобиля.

8. Создание сложного компонента

Это упражнение показывает, как создать дорожный конус с двойной мигающей лампой, используя компонент newLED. Заданные настройки и размеры можно найти методом проб и ошибок или использовать установки курсора и позиции курсора, показанные в нижней части системной панели.

Настройки компонента LED

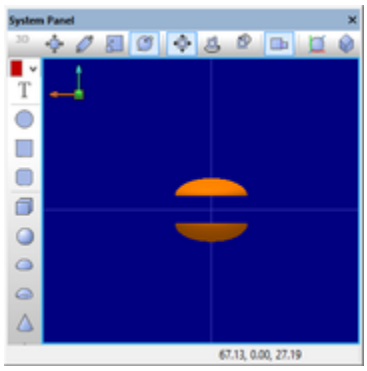
- Откройте новый проект.
- Убедитесь, что системная панель видима. Если это не так, используйте View основного меню.
- Допустим, что вы создали и экспортировали компонент LED, как это описано в упражнении «Создание компонента LED».



- Откройте группу, в которой вы сохранили экспортированный компонент newLED.
- Найдите его и используйте стрелку рядом с ним, чтобы добавить его на системную панель.
- Щёлкните по LED, чтобы выделить его. Измените свойство color на оранжевый (т.е. 0x0080FF).
- Используйте панель свойств (Panel Properties), когда LED выделен, чтобы задать: 'Position' как X=0, Y=0, Z=10, и 'Scale' в 'Wi...'=1, 'He...'=1, 'De...'=0.25.

Компонент должен приобрести вид, показанный на картинке (в зависимости от масштабирования «камерой»).

Добавление второго LED



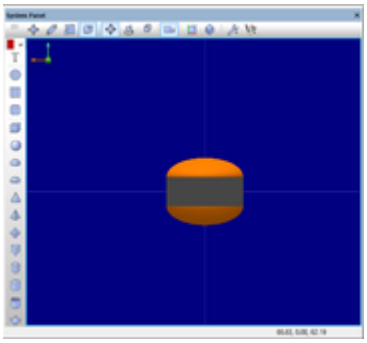
- Щёлкните по наконечнику вертикальной стрелки (зелёной), чтобы повернуть камеру для получения вида сбоку.
- LED должен быть выделен. Если нет, щёлкните по нему. Щёлкните по нему правой клавишей мышки и выберите команду Copy. Затем щёлкните правой клавишей мышки ещё раз и выберите Paste.

  Появится второй точно такой же светодиод.

- Задайте 'Position' с координатами X=0, Y=0, Z=-10, и 'Rotation' с X=0, Y=180, Z=0.

  Системная панель должна выглядеть так, как показано на картинке.

Монтаж LED

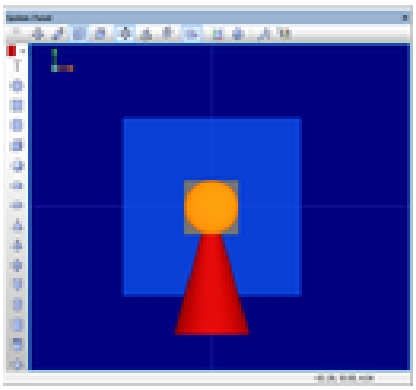


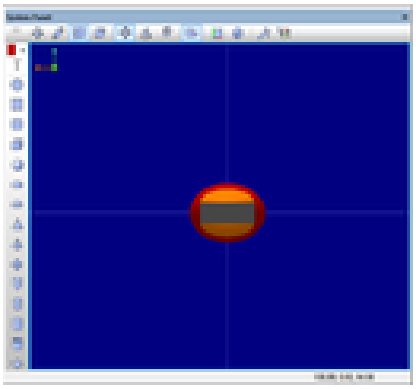
- Перетащите куб на системную панель.
- Выделите его задайте 'Position' с координатами X=0, Y=0, Z=0, и 'Scale' с 'Wi...'=32, 'He...'=12, 'De...'=32.
- Измените его цвет на серый (т.е. 0x5A5A5A).

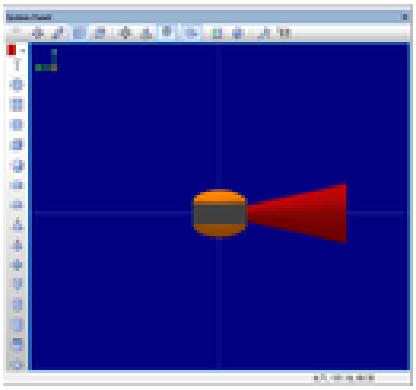
  Системная панель должна выглядеть так, как показано на картинке.

Создание дорожного конуса (traffic cone)

- Выделите курсором все три фигуры и сгруппируйте их, щёлкнув по иконке Group  .
- Заметьте, что Component Handle и Type изменились на Group. Переименуйте Handle как «flasher».
- Щёлкните по наконечнику вертикальной стрелки (синей), чтобы повернуть камеру назад к фронтальному виду.
- Перетащите конус на системную панель.
- Выделите его и задайте 'Position' с координатами X=0, Y=-12, Z=0, 'Scale' с 'Wi...'=45, 'He...'=45, 'De...'=120 и 'Rotation' с X=-90, Y=0, Z=0.
- Измените цвет на красный (т.е. 0x0000C0).
- Рассмотрите его с разных направлений камеры, щёлкая по наконечнику красной, зелёной и синей стрелок. Вид должен быть похож на тот, что показан на рисунках ниже:





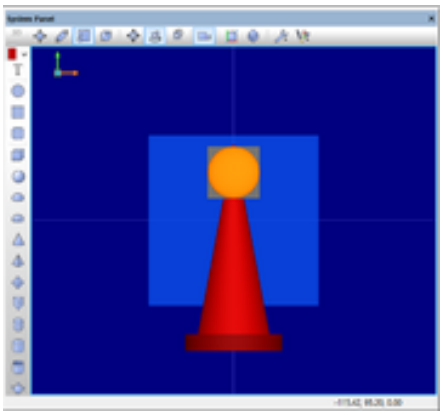


- Вернитесь к оригинальному виду (слева).
- Перетащите цилиндр на системную панель.
- Задайте следующие параметры:
  'Position' - X=0, Y=-75, Z=0;
  'Scale' -'Wi...'=60, 'He...'=60, 'De...'=10;
  'Rotation' - X=-90, Y=0, Z=0.
- Измените его цвет на тёмно-красный (т.е. 0x000080).

Завершение создания дорожного конуса

- Проведите курсором по всем трём фигурам, чтобы выделить их, и сгруппируйте их щелчком по иконке Group.

  Системная панель должна быть похожа на:

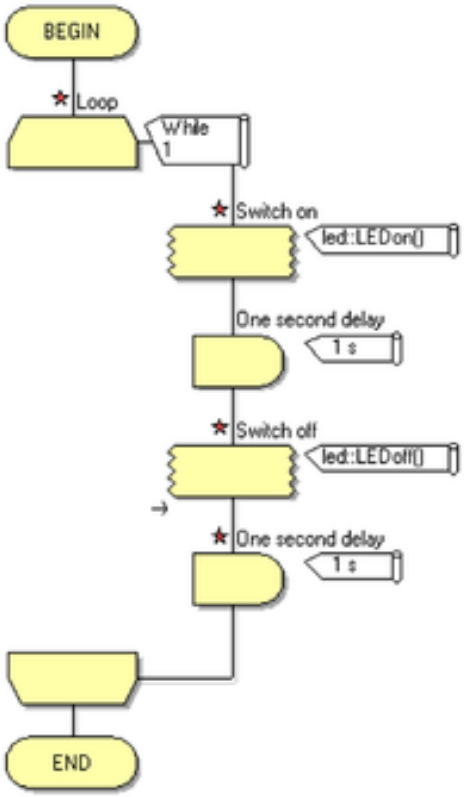


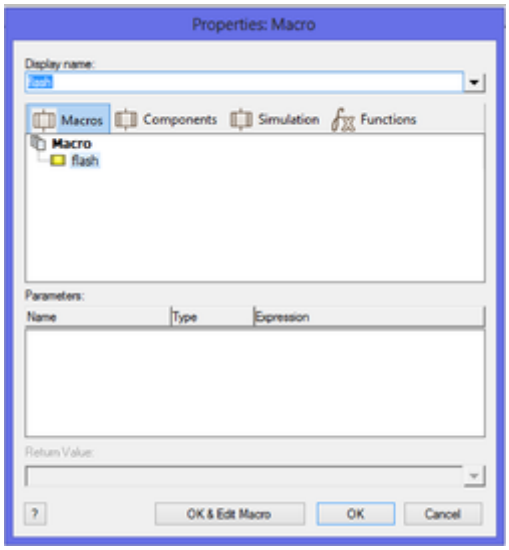
Добавление интерфейса

Следующая задача – заставить конус что-то делать: мигать, включаться и выключаться.

- Щёлкните по разделу Macro основного меню и выберите New, чтобы создать новый макрос.
- В диалоговом окне Create a New Macro назовите новый макрос flash и щёлкните по кнопке **ОК**.
- В программе нового макроса flash:
    - добавьте бесконечный цикл и внутрь цикла:
    - перетащите иконку Simulation macro;
    - дважды щёлкните по ней. Переименуйте её в Switch on. Щёлкните по закладке Simulation и по одной из этикеток LEDon, затем щёлкните по кнопке **ОК**;
    - перетащите иконку Delay;
    - дважды щёлкните по ней. Измените её имя на One second delay. Измените единицы на seconds и щёлкните по **ОК**.
    - перетащите вторую иконку Simulation macro;
    - дважды щёлкните по ней. Переименуйте её в Switch off. Щёлкните по закладке Simulation и по одной из этикеток LEDoff. Щёлкните по **ОК**;
    - перетащите вторую иконку Delay и сконфигурируйте её аналогично первой.

Подпрограмма flash должна выглядеть следующим образом:





- В основной программе Main добавьте иконку Macro.
    - Дважды щёлкните по ней и затем щёлкните по этикетке flash.
- Проверьте программу, запустив её симуляцию.

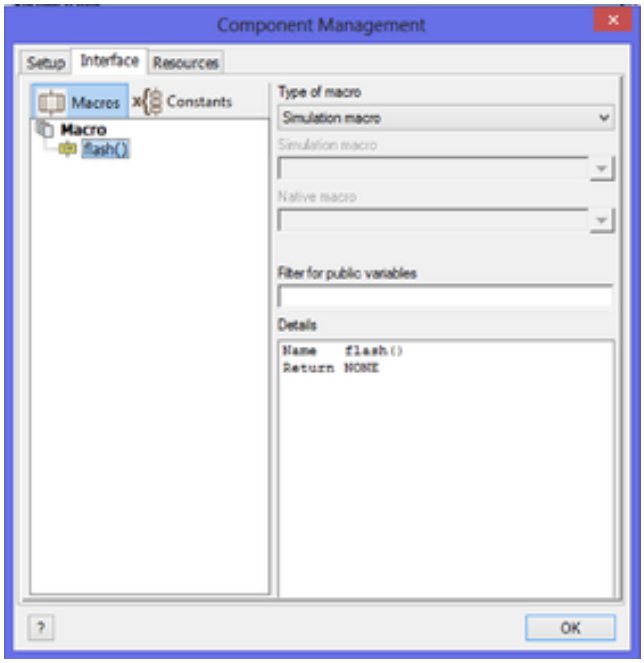
Экспорт компонента дорожный конус

Теперь, когда мы создали компонент дорожный конус, мы можем экспортировать компонент (который содержит также компонент newLED, созданный ранее) для его дальнейшего использования.

Объявление интерфейса

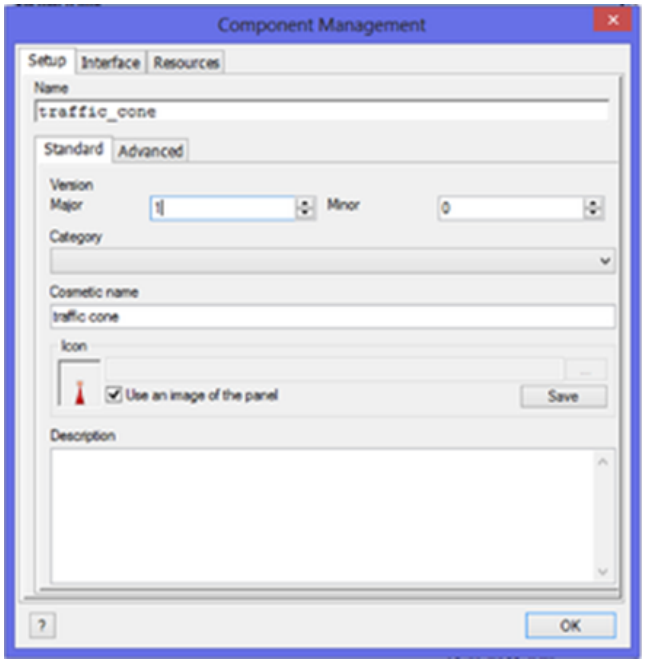
Перед тем, как экспортировать компонент, вам нужно объявить интерфейс (declare the interface).

- Поместите курсор на системную панель и щёлкните правой клавишей мышки.
- Выберите из выпадающего меню Export component, чтобы открыть диалоговое окно Component Management.



- Сконфигурируйте его для компонента «дорожный конус, traffic cone» следующим образом:
    - Щёлкните по закладке Setup.
    - Не добавляйте категорию. В этом случае дорожный конус появится в группе Misc.
    - Дайте компоненту Cosmetic name как traffic cone, которое впоследствии появится в группе Misc.
    - Щёлкните по окошку рядом с надписью Use an image of the panel.
    - Щёлкните по закладке Interface.
    - Задайте flash макрос, как макрос симуляции, щёлкнув по Simulation macro в выпадающем меню Type of macro.
    - В этом случае нет дополнительных файлов, которые следует сохранять вместе с компонентом, так что вам нет нужды использовать закладку Resources.

- Щёлкните по **ОК**.

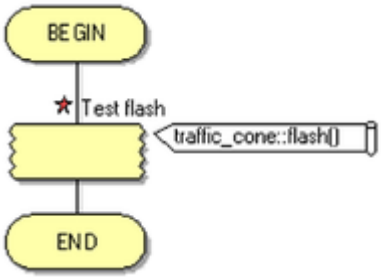


Сохранение компонента

- Вы можете получить запрос на сохранение текущей программы. В этом нет необходимости.
- Для компонента откроется диалоговое окно Save As. Автоматически открывается путь к установке Flowcode и папке «components». Выберите подходящее имя для компонента.
- Щёлкните по кнопке **Save**. Файл компонента сохранится с расширением .fcpx.

9. Импорт и тестирование дорожного конуса

- Создайте новый проект.
- Подразумевается, что вы выполнили все предыдущие операции и экспортировали дорожный конус.
- Откройте группу Misc и найдите компонент traffic cone.
- Используйте стрелку вниз рядом с компонентом и выберите добавление на системную панель.
- Создайте программу для тестирования импортированного компонента:
    - Добавьте компонентный макрос и дважды щёлкните по нему.
    - Щёлкните по закладке Simulation и затем по макросу flash, который появится в списке макросов.
    - Затем щёлкните по **ОК**.
- Ваша программа должна выглядеть теперь похожей на ту, что ниже:



Запустите симуляцию, чтобы проверить, будет ли работать импортированный компонент.

Создание программы

Задача в том, чтобы создать программу в Flowcode, которая станет базой для программы, что будет (когда будет сконфигурирована и запрограммирована правильно) включать лампу на десять секунд после нажатия на выключатель. Заметьте, что мы не будем в этом упражнении конфигурировать иконки для правильного функционирования, мы только зададим базовые настройки, которые впоследствии будут расширены.

Хотя это неразумно – использовать микроконтроллер в таком очевидном приложении, упражнение проиллюстрирует технику создания программы в Flowcode.

Сама программа может быть частью большей программы.

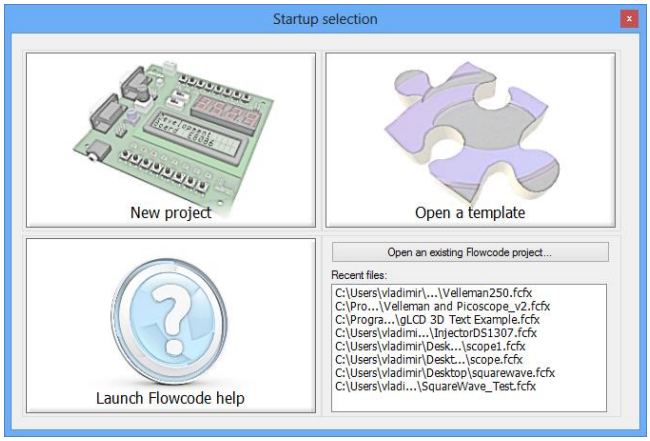
Сформулируем порядок работы программы

Порядок работы программы таков:

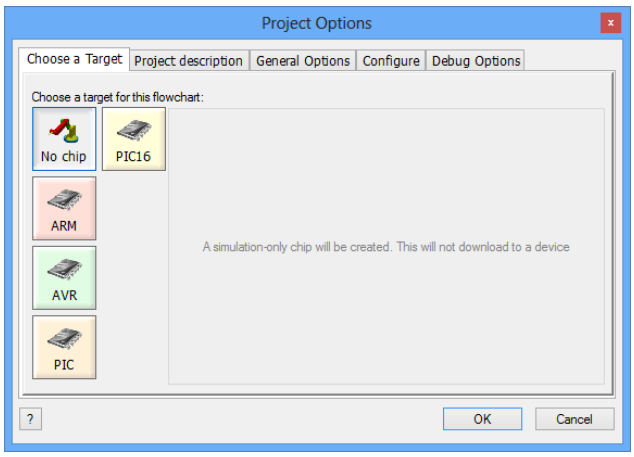
- Проверяем, нажат ли выключатель.
- Если нет, возвращаемся к началу.
- Если да, то
  - включаем лампу;
  - ждём 10 секунд;
  - выключаем лампу;
  - возвращаемся к началу.

Это занимает так мало времени у микроконтроллера, выполнить программу, что мы можем не беспокоиться удерживается ли выключатель или нет.

Настройки программы



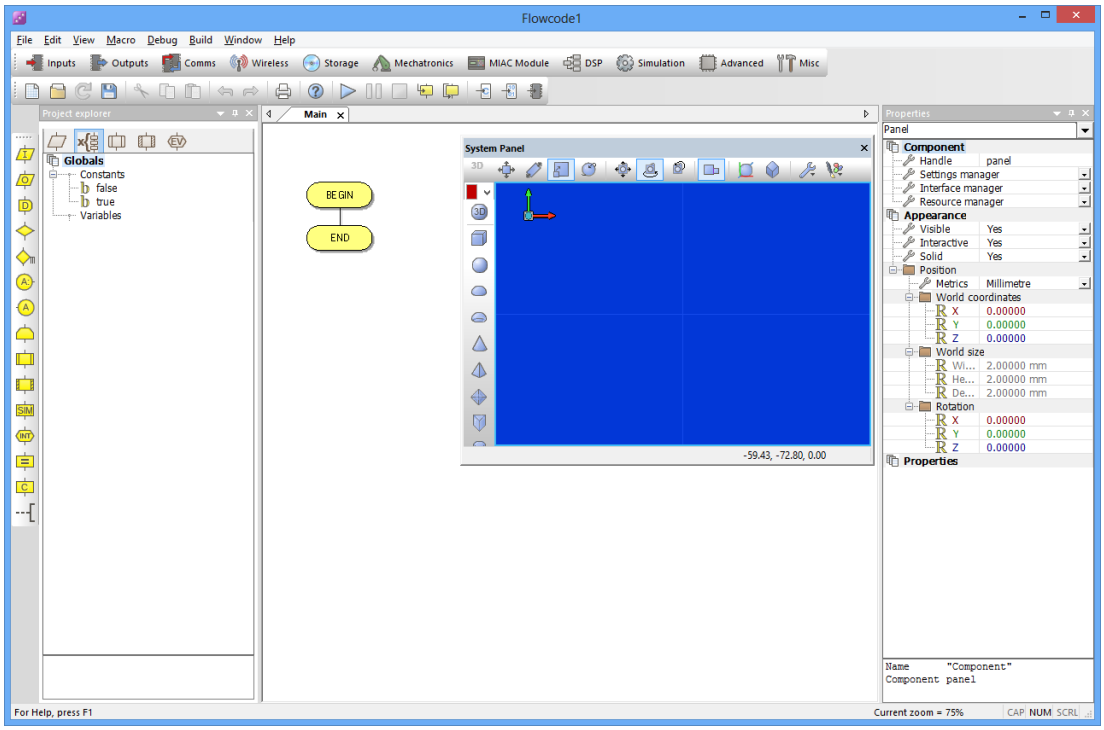
- Откройте Flowcode.
- На начальном экране щёлкните по New project.



- Откроется диалоговое окно Project Options. (Можете посмотреть полное описание диалога в Help).
- Задайте установки по умолчанию, щёлкнув по кнопке **OK**.

Устройте основное окно так, чтобы вы могли видеть системную панель (System Panel) и панель свойств (Properties Panel).

Основное окно должно быть похоже на показанное ниже (зависит от конфигурации).



Системная панель и панель свойств могут перемещаться – щёлкните по заголовку окна каждой панели и перетащите её за верхнюю часть.

Подготовка ввода

Больше информации об иконках, используемых в программах, вы можете найти в разделе о свойствах иконок (Icon Properties) в Help.

- Перетащите иконку цикла к линии между иконками BEGIN и END.

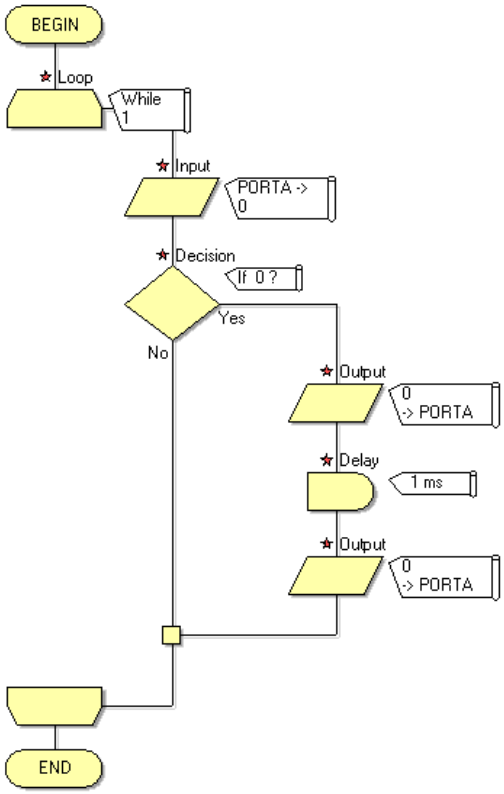
Иконка Loop заставит микроконтроллер повторять программу снова и снова (хотя цикл при необходимости можно сконфигурировать на выполнение конечного числа проходов).

- Внутрь цикла перетащите иконку Input с панели программных компонентов.
  - Мы позже сконфигурируем её и добавим значение переменной, которое будет использоваться для чтения состояния выключателя, чтобы обнаружить, когда выключатель будет нажат…

Подготовка выключателя

- Перетащите и вставьте иконку Decision после иконки ввода.

Эта иконка будет использоваться для выполнения решения, известного как условный переход, когда эта иконка будет сконфигурирована.



Когда иконки сконфигурированы и запрограммированы, эти иконки будут способны постоянно проверять, нажат ли выключатель или нет.
На следующем шаге мы добавим иконки, которые позволят нам управлять тем, что случится, если выключатель нажат. Заметьте, что программа не будет работать, пока иконки не будут сконфигурированы и запрограммированы.
После программирования – когда выключатель нажат, будет выполнено действие или процесс, и будет продолжаться проверка, нажат ли выключатель вновь.
Если выключатель не нажат, программа будет проходить по ветви No и продолжит выполнять цикл, проверяя, не нажат ли выключатель.

Настройка лампы

Теперь мы сосредоточимся на ветви Yes иконки Decision.

- Перетащите иконку Output в ветвь Yes.

Когда будет запрограммирована, эта иконка выполнит переключение лампы.
Далее мы хотели бы, чтобы лампа оставалась включена на десять секунд, а затем выключалась.

- Перетащите и вставьте иконку Delay после иконки Output.

    - Эта иконка позволит нам задержать выполнение программы на заданный период времени.
    - Задержка будет на 10 секунд, в это время программа не будет способна выполнять какие-то другие функции, пока время не истечёт.

    - Перетащите вторую иконку Output и вставьте после иконки Delay. Впоследствии она будет использоваться для выключения лампы после десятисекундной задержки.

Ваша программа должна теперь быть похожа на ту, что представлена выше.

10. Конфигурирование иконок и переменных

Это упражнение подразумевает, что вы уже создали программу Flowcode, описанную выше.

Чтобы настроить и запрограммировать программы вы должны сконфигурировать иконки и переменные, чтобы создать эффективный и функциональный процесс.

Это подготовит программу для взаимодействия с компонентами, соединёнными с заданными портами.

Загрузка программы в Flowcode

- Откройте программу, названную Lamp1.fcfx, которую вы создали в упражнении «Создание программы».

    Системная панель и панель свойств должны быть видимы. Если нет, используйте View основного меню для выбора включения этих окон.

Сформулируем порядок работы программы

Порядок работы программы таков:

- Проверяем, нажат ли выключатель.
- Если нет, возвращаемся к началу.
- Если да, то
    - включаем лампу;
    - ждём 10 секунд;
    - выключаем лампу;
    - возвращаемся к началу.

Это занимает так мало времени у микроконтроллера, выполнить программу, что мы можем не беспокоиться, удерживается ли выключатель или нет.

Программирование входа

- Дважды щёлкните по иконке Input.

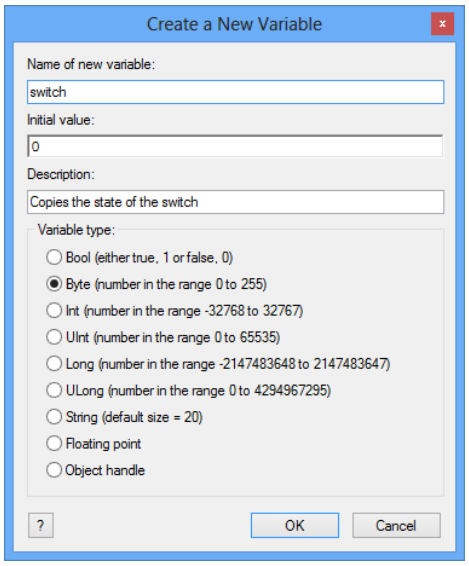
    Этим откроется диалоговое окно Properties: Input, позволяющее вам сконфигурировать, как программа получит информацию от выключателя.
    Эта информация будет содержаться в переменной, названной switch.

- Щёлкните по стрелке вниз справа от окошка Variable, чтобы открыть диалоговое окно переменных.
- Поместите курсор слева от названия Variables и щёлкните по стрелке вниз, которая появится.
- Щёлкните по разделу Add new, чтобы открыть диалог Create a New Variable.

    - Создайте новую переменную с именем switch и начальным значением «0», добавьте описание «Copies the state of the switch».
    - Оставьте Variable type как Byte.

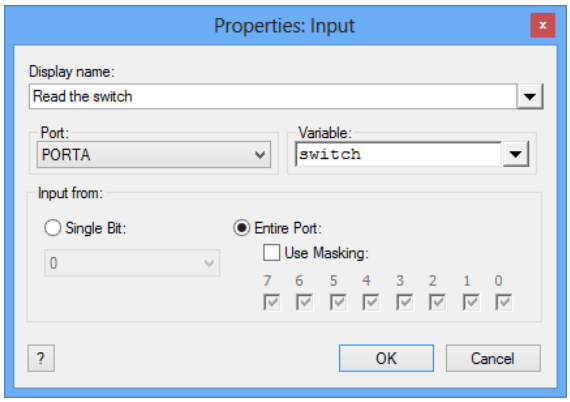
Больше информации о переменных можно найти в соответствующих разделах Help.

Результирующее диалоговое окно показано ниже.



- Завершите конфигурацию свойств входа следующим:
- измените Display name на Read the switch;
- в окне Variable введите имя переменной, которую вы создали, switch;
- оставьте в окне Port то, что есть, PORT A;
- выделите Input from: как Single Bit и выберите бит 0.

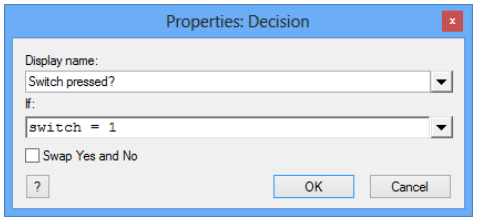
Окончательно диалоговое окно должно выглядеть так:



Настроенная таким образом программа отслеживает состояние выключателя, который будет подключён биту 0 PORT A микроконтроллера.
Когда программа обнаруживает на входе, что выключатель нажат, переменная switch становится равной логической 1. Если выключатель не нажат, она равна логическому 0.

Проверка выключателя

- Дважды щёлкните по иконке Decision, чтобы открыть диалоговое окно конфигурации, а затем:



- Переименуйте компонент в Switch pressed?.
- В окошке If впишите switch = 1.
- Оставьте Swap Yes and No окошко флажка пустым.
- Щёлкните по **ОК**.

Этим вы заставите программу выполнить то, что называют условным переходом. Последовательность прохождения программы зависит от выполнения условия, заданного в иконке Decision.

В данном случае оно зависит от того, будет ли или нет переменная switch = 1. Если будет, программа выполняется по ветви Yes, если нет, то выполняются команды ветви No.

Управление лампой

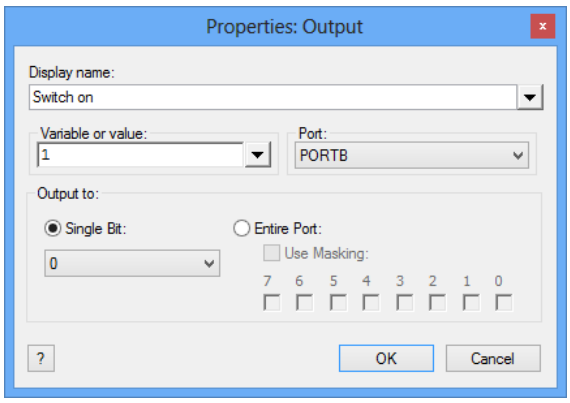
Ветвь No легко настроить. Всё, что мы хотим, это чтобы программа вернулась к началу (к верхней части иконки Loop).

Нет необходимости что-то конфигурировать.

В ветви Yes:

- Дважды щёлкните по иконке Output, чтобы открыть диалоговое окно конфигурации, а затем:
  - Измените Display name на Switch on.
  - В окне Variable or value задайте значение 1.
  - Измените Port на PORT B.
  - Выберите Single Bit в разделе Output to и бит 0.
  - Щёлкните по **ОК**.

Получившаяся конфигурация показана ниже:

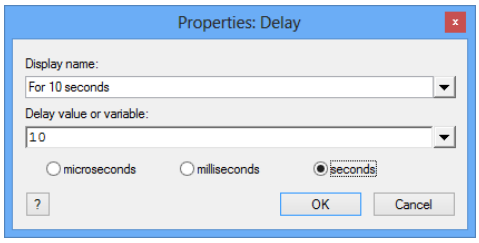


Эффект от работы иконки в том, что она отправляет сигнал логической 1 (высокий уровень) на лампу, подключённую биту 0 порта B микроконтроллера. Что включит лампу.

Теперь мы выполним требование, чтобы она оставалась включена десять секунд, а затем выключалась.

- Дважды щёлкните по иконке Delay, чтобы открыть диалоговое окно, затем:
  - Измените Display name на For 10 seconds.
  - Измените Delay value на 10.
  - Измените unit на seconds.
  - Щёлкните по **ОК**.

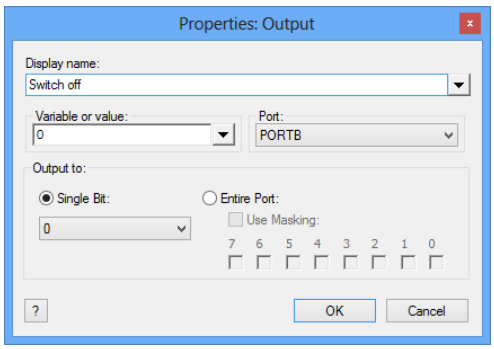
Результат конфигурации показан ниже.



В завершение мы должны выключить лампу после десятисекундной задержки.

- Дважды щёлкните по второй иконке Output, что позволит сконфигурировать её.
  - Переименуйте её в Switch off, оставьте значение 0 для единственного бита порта B.

Диалог должен выглядеть так:



Осталось сохранить программу под именем Lamp1.fcfx и закрыть программу.

11. Добавление устройств в программу

Это упражнение подразумевает, что вы уже создали в Flowcode программу, описанную выше.

Чтобы завершить программу, вам нужно добавить два электронных компонента: выключатель и лампу.

Загрузка программы в Flowcode

Откройте проект, названный Lamp1.fcfx, который вы создали ранее.

Добавление выключателя

- Щёлкните по группе Inputs инструментальной панели дополнительных компонентов и найдите Push Round Panel  выключатель.
- Поместите курсор поверх картинки и щёлкните по стрелке вниз, которая появится.
- Выберите Add to system panel.

- Щёлкните по выключателю на системной панели, чтобы выделить его. Его свойства появятся на панели свойств (Panel Properties).

  Заметьте, что имя (Handle) выбранного компонента появляется в верхней части панели свойств. В нашем случае имя выключателя показано как «sw_push_rnd_pnl»!

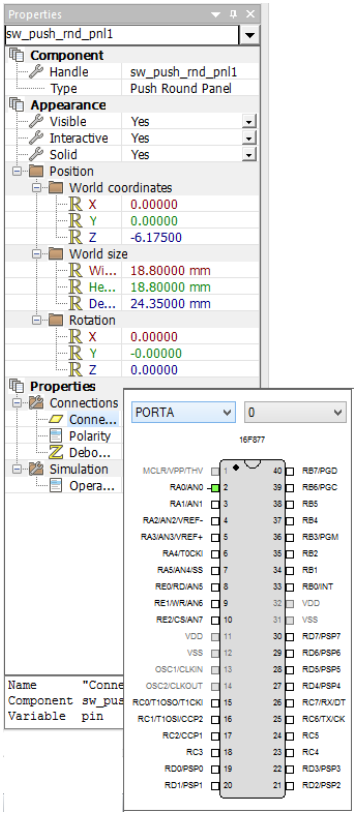
- Найдите свойство Connections на панели свойств, щёлкните в окошке правее Conne…

  Появится цоколёвка микроконтроллера.

  - Убедитесь, что подключение к порту A бит 0, используя выпадающее меню, или щёлкните по квадратику вывода бита 0 (названному RA0/AN0 на картинке).

Это подключит выключатель к биту 0 порта A микроконтроллера, что соответствует настройкам, которые вы использовали при создании программы.

Эти настройки выглядят так:



- Нет необходимости в других изменениях, хотя раздел Scale позволяет вам изменить размер (и фигуру) выключателя, если вам захочется.

Добавление лампы

- Щёлкните по группе Outputs инструментальной панели и добавьте LED, как компонент LED 5mm Panel, который мы будем использовать как лампу.
- Поместите курсор поверх картинки, щёлкните по стрелке вниз, которая появится.
- Выберите Add to system panel.
- LED появится в центре системной панели, возможно, правее выключателя.

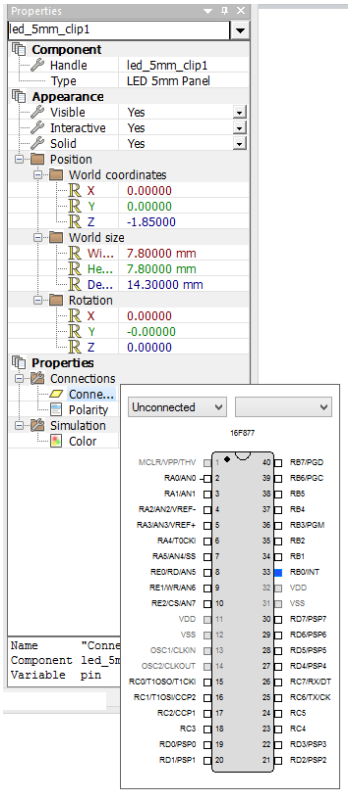
  Светодиод должен уже быть выделен, но если нет, щёлкните по нему.
  Перетащите его в подходящее место к одной из сторон.

- С выделенным светодиодом обратитесь к панели свойств.

  Заметьте, что Handle изменилось на имя поддержки компонента.
  Свойство Connection может быть задано неверно, по программе оно должно быть к биту 0 порта B.

  - Щёлкните по окошку рядом со свойством Conne…, как раньше, появится цоколёвка микроконтроллера.
  - Щёлкните по квадратику, представляющему вывод бита 0 порта B, показанному как RB0/INT, или используйте выпадающее меню для подключения компонента.

Панель свойств показывает теперь, что светодиод подключён к биту 0 порта B.



- Вновь, нет нужды изменять другие свойства, хотя раздел Scale может изменить размер компонента, а свойство Color (для других компонентов Bulb color) позволит изменить цвет светодиода.

Сохранение проекта

- Сохраните проект, используя то же имя. Самый простой способ сделать это – щёлкнуть по иконке  , Save.

12. Симуляция программы

Это упражнение показывает тестирование программы Lamp1.fcfx, которую вы создали в предыдущих упражнениях.

Первая часть процесса – это симуляция программы. Этим проверяется только программа, но не оборудование. Всё происходит внутри программы Flowcode, и не требует какой-либо аппаратной поддержки. В Help вы найдёте более подробное описание процесса симуляции.

Загрузка программы

- Откройте программу, созданную ранее и названную Lamp1.fcfx.

Запуск симуляции

- Щёлкните по иконке Run (или нажмите на клавиатуре функциональную клавишу F5)  .

  Может открыться окно Simulation debugger, игнорируйте его пока!

- Ничего не произойдёт, пока вы не включите лампу. Чтобы это сделать, щёлкните по кнопке выключателя.

Лампа зажжётся, окно Simulation Delay будет заполняться, поскольку Flowcode симулирует паузу в десять секунд.
В конце этого лампа погаснет.

- Вы можете повторять процесс столько раз, сколько хотите. Когда вы готовы, щёлкните по иконке  Stop simulation (или нажмите Shift и F5).

Пошаговая симуляция

Когда что-то идёт не по плану, полезно симулировать программу шаг за шагом (то есть, иконка за иконкой).

- Щёлкните по иконке Step Into  (или нажмите F8).

  Вокруг первой иконки в программе появится красная рамка.
  Когда вы вновь щёлкните по иконке шага, программа выполнит команду иконки программы, а красная рамка переместится к следующей иконке.
  Так вы можете проверить, будут или нет выполняться операции ожидаемым образом.

- В любом месте вы можете щёлкнуть по иконке Stop simulation, чтобы завершить симуляцию.

13. Загрузка программы в микроконтроллер

Это упражнение покажет загрузку в микроконтроллер программы Lamp1.fcfx, которую вы создали и проверили в предыдущих упражнениях.

Это потребует:

- Компиляцию программы (перевод в форму, которую может использовать микроконтроллер).
- Отправку её в подключённый микроконтроллер.
- Сохранение её в памяти микроконтроллера.

Загрузка созданной программы в Flowcode

- Откройте программу, названную Lamp1.fcfx, которую вы создали и протестировали раньше.

Выбор микроконтроллера

- Откройте Project Options через Build основного меню, а затем перейдите к закладке Choose a Target (Build > Project Options... > Choose a Target).

  Убедитесь, что вы выбрали правильный микроконтроллер, если выбрано No chip/SIM chip, тогда для компиляции микроконтроллер не выбран.
  Затем убедитесь, что нужный микроконтроллер подключен и готов к компиляции и загрузке программы.

Компиляция программы

- Щёлкните по иконке  Compile to chip.

  Появится окно Compiler Messages, в котором отобразится прогресс процесса компиляции.
  Сначала программа конвертируется на язык Си.
  Затем программа будет транслирована на другой язык, называемый ассемблер.
  И, наконец, результирующий код будет оттранслирован в микроконтроллер.
  Хорошая новость в том, что всё это происходит автоматически без необходимости вашего вмешательства!

14. Документирование программы

Это упражнение подразумевает, что вы создали и проверили программу, описанную выше.

В некоторых проектах необходимо документирование программы; хотя мы хорошо маркируем все иконки, несколько иконок комментариев помогают объяснить программу, а в дальнейшем детально помочь в её распространении, помочь другим понять систему.

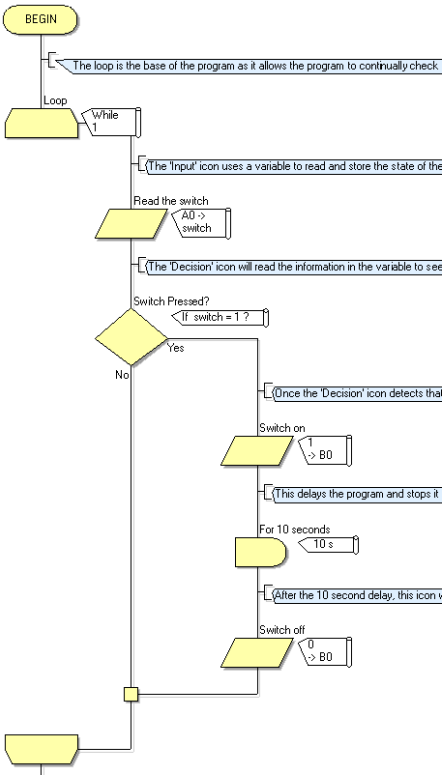
Загрузка программы

- Откройте программу, названную Lamp1.fcfx, которую вы создали и протестировали раньше.

Преимущества документации

Добавление комментариев в вашу программу крайне просто и очень полезно. Использование комментариев позволяет вам сделать прозрачнее процесс, назначение и функционирование вашей программы, позволяет другим понять её, так что они могут увидеть, что происходит, и даже помогут развить программу, применяя более эффективные методы или непосредственно редактируя иконки для более действенного функционирования.

Добавление комментариев



- Перетащите иконку Comment и вставьте над иконкой Loop.
- Дважды щёлкните по иконке Comment, чтобы открыть диалоговое окно Comment Icon Properties.
- Поясните назначение цикла, и как он организует ядро процесса.

Цикл – это основа программы, позволяя программе непрерывно проверять, нажат ли выключатель в повторяющемся процессе слежения за состоянием выключателя.

- Добавьте комментарий над иконкой Input.
- Откройте свойства иконки двойным щелчком по ней.
- Опишите использование иконки Input и то, как она читает состояние выключателя.

Иконка Input использует переменную для чтения и запоминания состояния выключателя, что позволяет информацию, содержащуюся в переменной, использовать для наблюдения, нажат ли выключатель или нет.

- Прокомментируйте иконку Decision.
  - Опишите, как иконка Decision определяет состояние выключателя и решает, какой следующий шаг следует выполнять.

Иконка Decision будет читать информацию в переменной, чтобы увидеть, был ли нажат выключатель, а затем решить, какой путь процесса выбрать на следующем шаге; если не

было нажатия, процесс перейдёт сразу к началу цикла и продолжит проверять состояние выключателя. Если выключатель был нажат, следующие иконки активизируют LED и выключат его после 10 секунд.

- Прокомментируйте иконку Output.
  - Опишите функции иконки Output.

Когда иконка Decision обнаруживает, что выключатель нажат, иконка Output включает LED и показывает, что выключатель был нажат.

- Прокомментируйте иконку Delay.
  - Опишите функции и назначение иконки.

Она задерживает программу, останавливая продолжение процесса её работы, что позволяет светодиоду оставаться включённым после предыдущей команды, эта пауза длится 10 секунд, после чего работа программы возобновляется.

- Прокомментируйте вторую иконку Output.
  - Поясните функцию этой иконки и дальнейший процесс.

После 10 секунд задержки эта иконка выключает LED, а затем продолжает процесс работы программы, который, в свою очередь, будет циклом новой проверки нажата ли кнопка, цикл продолжается пока не будет остановлен.

Сохранение программы

- Сохраните программу под тем же именем. Самый простой способ сделать это – щёлкнуть по иконке сохранения  .

15. Расширение программы

Это упражнение подразумевает, что вы создали, протестировали и задокументировали программу, описанную выше.

Поскольку вы задокументировали программу правильно, вы можете легко и аккуратно изменить её, поскольку вы знаете, как работает программа и через какие процессы проходит.

Вы не должны основательно переделывать программу – чтобы сделать её привлекательнее, иной раз небольшая правка даёт большой эффект, простое добавление или конфигурирование одного компонента могут изменить программу, сделав её значительно эффективнее и пригоднее для использования где угодно.

Загрузка программы

- Откройте программу Lamp1.fcfx, которую вы создали ранее и задокументировали выше.

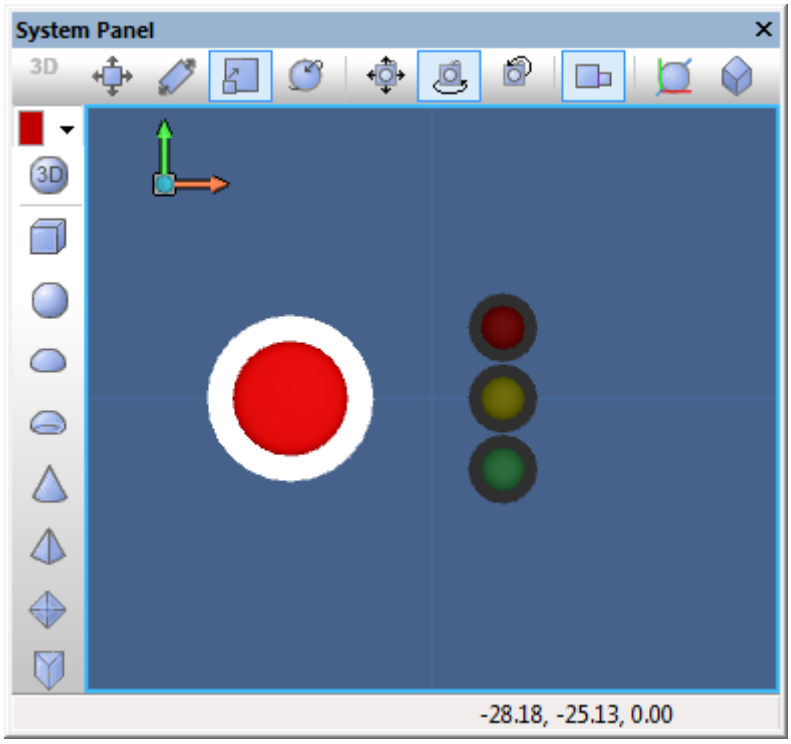
Улучшение программы

Теперь, когда у вас появилась полностью функционирующая программа, которая хорошо задокументирована и именами иконок, и подробными комментариями, вы можете

манипулировать программой, интегрировать её в другие системы или расширять её, превратить в собственную расширенную систему и даже превратить её в полное приложение.

Мы намерены расширить нашу программу так, чтобы она работала в качестве светофора. Чтобы это сделать, мы собираемся добавить ещё два светодиода и изменить иконку Decision, которая откликается на срабатывание выключателя, активируя светофор.

Добавление светодиодов



Сначала мы собираемся настроить системную панель, добавив ещё два компонента LED, чтобы расширить программу.

- Добавьте два LED, используя один из следующих методов:
    - Либо скопируйте и вставьте уже существующий светодиод на системную панель.
    - Либо добавьте ещё два компонента, используя панель добавочных компонентов.

Теперь расположите их соответственно на системной панели.

- Переместите LED, выстроив их вертикально по Y.
- Убедитесь, что первый, зелёный компонент LED внизу, и что он ещё подключен к $PORTB.0.

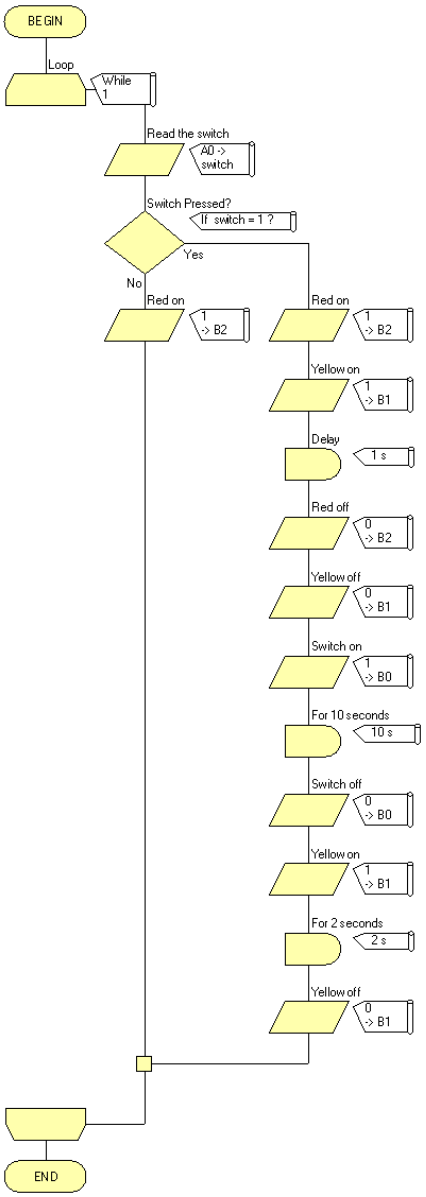
Далее измените свойства цвета и подключения двух новых светодиодов.

- Измените свойства среднего LED.
    - Подключите компонент к порту B бит 1, изменив свойство Connection на $PORTB.1.
    - Измените свойство Color на жёлтый (00FFFF), используя окно выбора цвета.
- Измените свойства верхнего LED.

- Подключите LED к Port B бит 2, изменив свойство Connection на $PORTB.2.
- Измените свойство Color на красный (000FF), используя окно выбора цвета.

Расширение программы

Теперь, когда компоненты настроены правильно, пришло время сконфигурировать программу и расширить её. Сначала нам нужно задать состояние светофора по умолчанию, когда при не нажатом выключателе горит только красный свет.



- Добавьте иконку Output в ветвь No иконки Decision.
  - Измените Display name на Red on.
  - Введите значение «1» в окно Variable or value.
  - Задайте Port соединение как PORTB.
  - В разделе Output to выберите Single Bit и измените бит на «2».
  - Щёлкните по **ОК**, чтобы подтвердить изменения иконки.

Теперь мы собираемся установить последовательность активизации огней светофора, сконфигурировав и расширив ветвь Yes иконки Decision.

- Добавьте 4 иконки Output над существующими иконками, затем добавьте иконки Delay между двумя иконками Output.
    - Мы сконфигурируем эти иконки сверху вниз следующим образом:
        - Измените Display name первой иконки на Red on.
        - Эта иконка такая же, как предыдущая, добавленная в ветвь No иконки Decision.
        - Установите значение «1» для Port B, бит 2.
        - Для следующей иконки измените имя на Yellow on.
        - Аналогично отправьте «1» в порт B, но на этот раз для бита 1.
        - Следующая иконка Delay.
        - Измените её имя на For 1 second.
        - Введите значение 1 в окно Delay value or variable.
        - Задайте единицы как seconds.
    - Выключите первые два (красный и жёлтый) светодиода, чтобы зажечь последний, зелёный, который программировался ранее.
        - Исправьте иконку, следующую за иконкой Delay.
        - Измените её имя на Red off.
        - Отправьте значение «0» в Port B, бит 1 – это выключит красный свет.
        - Сконфигурируйте следующую иконку, последнюю из добавленных нами.
        - Назовите её Yellow off.
        - Отправьте значение «0» в Port B, бит 2 – этим мы выключим жёлтый свет.

Далее, зелёный свет горит 10 секунд перед выключением, всё это благодаря иконкам, которые мы сконфигурировали и запрограммировали в предыдущих упражнениях.

- Обновите отображаемые имена существующих иконок следующим образом:
    - Замените Switch on на Green on.
    - Замените Switch off на Green off.
- Под этими иконками добавьте иконку Output с иконкой Delay за ней, и, наконец, добавьте последнюю иконку Output.
    - Сконфигурируйте первую добавленную иконку.
    - Измените Display name на Yellow on.
    - В Variable or value впечатайте «1».
    - Измените порт на PORTB.
    - В разделе Output to выберите Single Bit, измените значение на «1».
    - Теперь займёмся иконкой Delay.
    - Измените её имя на For 2 seconds.
    - Введите 2 в окно Delay value or variable.
    - Измените единицы на seconds.
    - В завершение сконфигурируйте последнюю иконку Output.
    - Измените имя на Yellow off.
    - В окне Variable or value пусть будет 0.
    - Измените Port на PORTB.
    - В разделе Output to выберите Single Bit и измените его на «1».

Окончательное тестирование

- Сохраните программу.
- Запустите симуляцию для проверки работы программы.
- Если вы намерены и дальше расширять программу или предложить её к всеобщему пользованию, убедитесь, что задокументировали её, пояснив процесс и описав, какое действие он окажет на компоненты.

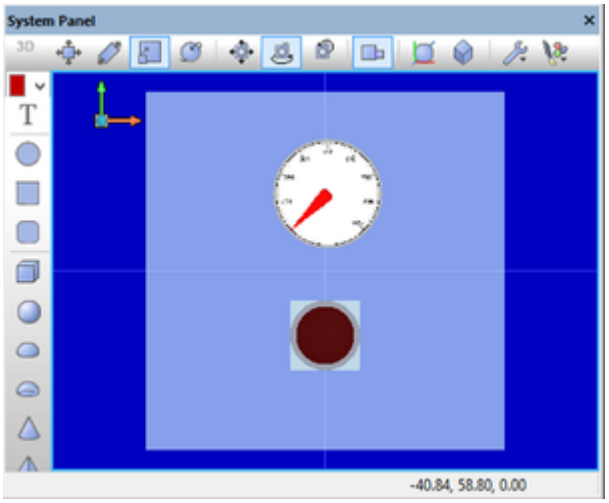
16. Использование устройств с аналоговым входом

Смысл этого упражнения в демонстрации использования аналогового входа в программе, созданной в Flowcode.

Цифровые входы проще в использовании, поскольку имеют ограниченный диапазон значений. Например, двухбитовый цифровой вход может принять только одно из четырёх возможных значений: 00, 01, 10 или 11.

Flowcode использует иконку  для работы с цифровыми входами.

Аналоговый вход, с другой стороны, может иметь любое из бесконечного набора допустимых значений. В результате он труднее в поддержке в Flowcode.



Для ввода данных от аналогового датчика используется компонентный макрос  .
Данные в этом случае хранятся в переменной.

Компонентные макросы – это фрагменты кода, которые были написаны для поддержки компонентов, включённых в Flowcode 6.

Они берут на себя все сложности использования этих компонентов.

В этом упражнении зажигается свет лампы, когда сигнал от диска ADC возрастает.

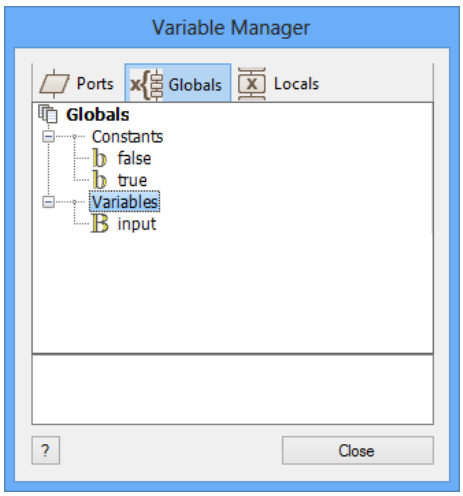
Порядок работы программы

Программа будет:

- Читать значение, задаваемое входным устройством, диском ADC dial.
- Сравнивать его с набором значений и:

если больше заданного значения, включать лампу;
если меньше, тогда выключать лампу.

- Возвращаться к началу программы и повторять процесс.



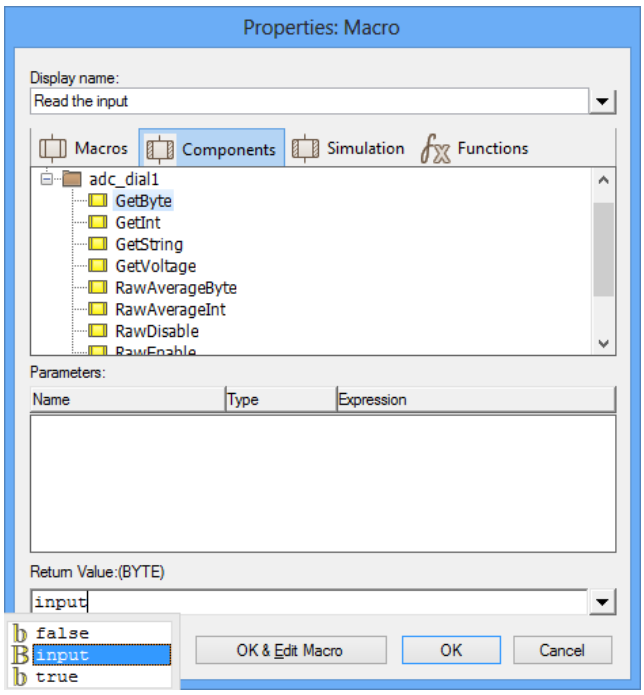
Создание переменной input

- В разделе Edit основного меню щёлкните по Variables..., чтобы открыть диалоговое окно Variable Manager.
- Поместите курсор поверх названия Variables и щёлкните по стрелке вниз слева, которая появится.
- Выберите Add new и появится диалоговое окно Create a New Variable.
- Назовите новую переменную input.
- Оставьте тип переменной как Byte.
- Щёлкните по **ОК**.
- Диалоговое окно показано выше.

Создание программы

В группе Inputs найдите и поместите на системную панель компонент ADC dial.

- Щёлкните и перетащите бесконечный цикл  между иконками BEGIN и END.
- Внутри цикла:
    - Щёлкните и перетащите иконку компонентного макроса  .
    - Дважды щёлкните по ней, чтобы открыть диалоговое окно, в котором вы можете конфигурировать компонент.



Программа «знает», какой компонент вы добавили на системную панель или панель управления, модифицируя список доступных для компонента команд.

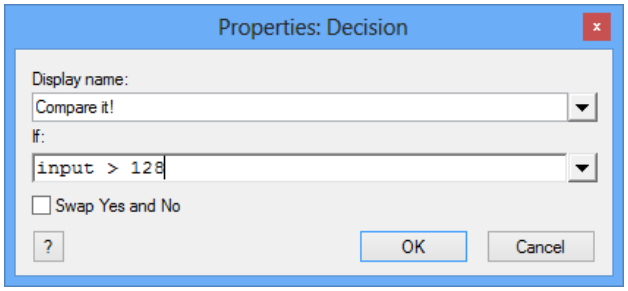
На закладке Components в списке появится ADC dial с раскрывающимся списком команд.

- Переместитесь по списку команд и щёлкните по GetByte.

Она прочитывает выход аналогового входного устройства, ADC dial в нашем случае, и сохраняет значение в байтовой переменной, названной input (её имя введите в окне Return Value).

- Переименуйте название в окне Display name как Read the input.
- Щёлкните по **ОК**.
- Диалоговое окно показано на рисунке выше.

- Следом перетащите иконку Decision  и дважды щёлкните по ней, чтобы открыть диалоговое окно.

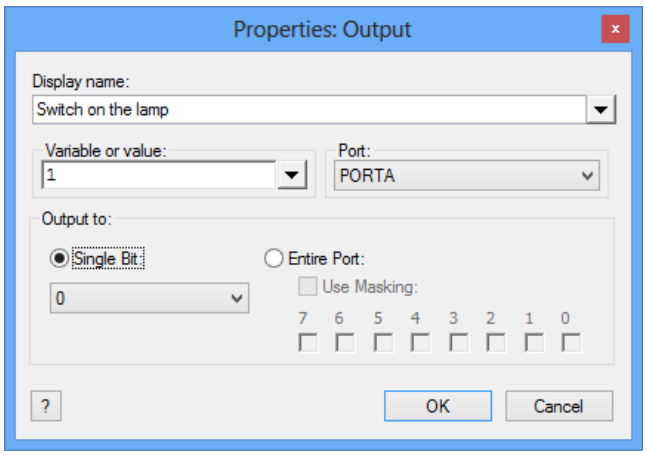


- Переименуйте его в Compare it!
- В условии запишите input>128.

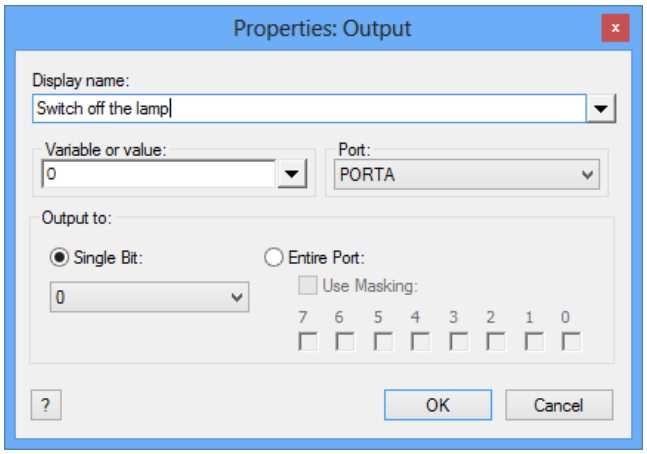
128 – это совершенно произвольное число в этой программе. Байтовая переменная может хранить любое значение от 0 до 255, так что 128 ровно половина.

В этом случае программа решает, будет ли значение, хранящееся в переменной input, больше или меньше 128, и поступит по-разному, в зависимости от полученного результата.

- Щёлкните по **ОК**.
- Вновь, диалоговое окно показано выше.

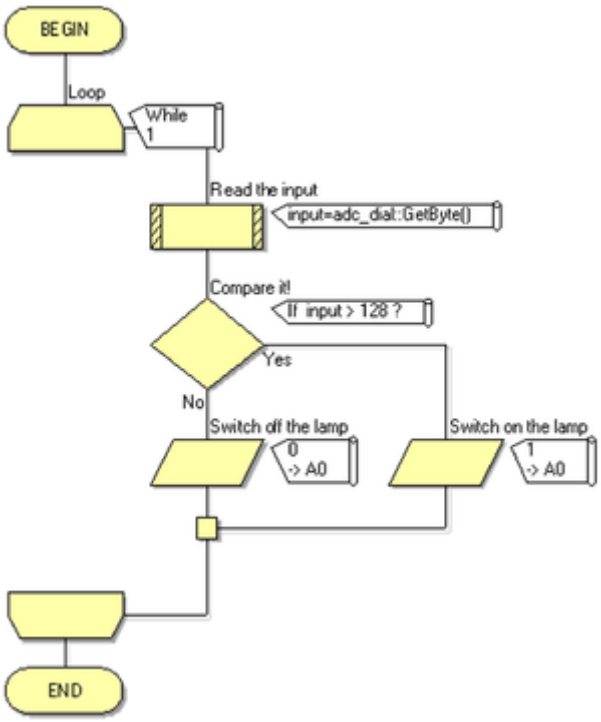


- В ветвь Yes перетащите и вставьте иконку Output  .
    - Дважды щёлкните по ней, чтобы открыть диалоговое окно.
    - Измените Display name на Switch on the lamp.
    - В окно Variable or value введите «1».
    - Сконфигурируйте оставшиеся разделы как PORT A, Single Bit, 0.
    - Щёлкните по **ОК**.



- В ветвь No перетащите и вставьте вторую иконку Output.
    - Дважды щёлкните по ней, чтобы открыть диалоговое окно.
    - Измените имя в Display name на Switch off the lamp.
    - В окне Variable or value оставьте значение «0».
    - Сконфигурируйте остальное так: PORT A, Single Bit, 0.
    - Щёлкните по **ОК**.
    - Вновь, диалоговое окно для иконки Output показано выше.

Программа теперь должна выглядеть так, как показано ниже.



Добавление светодиода

- Найдите Led, такой как LED 5mm Panel в группе Outputs инструментальной панели дополнительных компонентов.

    - Поместите курсор на картинку слева от компонента помеченного как LED 5mm Panel и щёлкните по появившейся стрелке вниз.
    - Щёлкните по Add to system panel, чтобы выбрать этот вариант.
    - Выделите LED на системной панели и перетащите его в удобное место.
    - Обратитесь к свойству Connection панели свойств для этого компонента, чтобы убедиться, что он подключён к Port A, бит 0 ($PORTA.0).

Оставьте это в виде, показанном в начале упражнения.

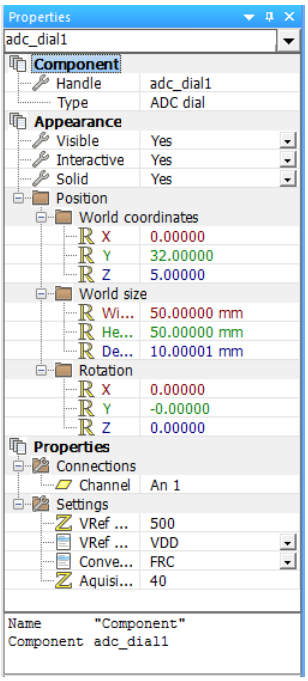
Добавление ADC dial

ADC dial представляет ряд входных устройств.

- Найдите ADC dial в группе Inputs инструментальной панели дополнительных компонентов.

    - Поместите курсор на картинку слева и щёлкните по стрелке вниз, которая появилась.
    - Щёлкните по Add to system panel.
    - Выделите ADC dial на системной панели и перетащите в удобное место.
    - Обратитесь к панели свойств, найдите свойство Connection. Сразу после него, свойство Connection показывает это, выход подключён к An 0 – первому биту микроконтроллера, способному работать с аналоговым вводом. Щелкните по An 0, чтобы открыть цоколёвку контроллера.
    - Щёлкните по квадратику, представляющему вход AN1, чтобы изменить подключение к этому каналу.

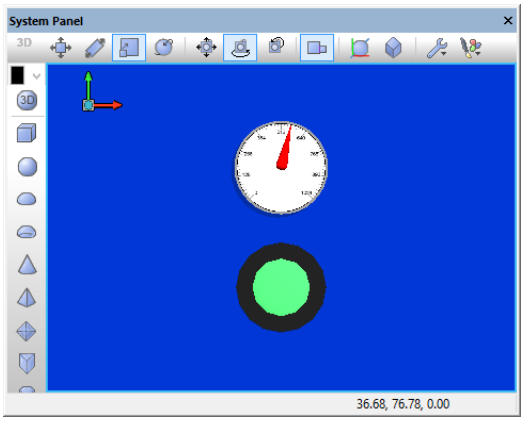
Оставьте это, как показано ниже:



Системная панель должна выглядеть так, как показано в начале упражнения.

Окончательно тестирование

- Сохраните программу как Analogue.fcfx.
- Щёлкните по кнопке симуляции  .
- Используйте мышку, чтобы повернуть указатель на ADC dial, и обратите внимание на то, что получится, когда вы пройдёте отметку половины шкалы.



- Щёлкните по кнопке остановки симуляции  , когда порадуетесь поведению программы.

17. Использование макросов

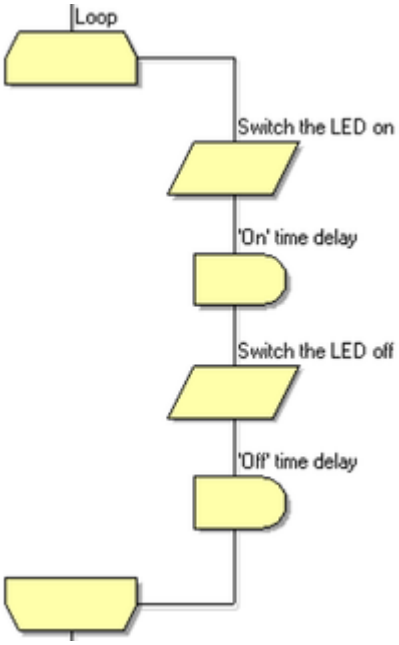
Макрос – это фрагмент кода, который повторяется несколько раз в программе.

Гораздо удобнее использовать макрос вместо того, чтобы раз за разом создавать код заново, когда он понадобится.

Это упражнение показывает, как использовать макрос с меню при создании простого мигающего светодиода. Светодиод может мигать с разной частотой при нажимании разных кнопок.

В первой части выполним ядро программы с меню. Во второй части покажем, как создать макрос.

Связывание частоты и задержки



Программа, управляющая миганием LED, показана выше.

Таблица ниже показывает связь между длительностью пауз для каждой иконки Delay и частотой мигания. LED включается на один период паузы, а затем выключается ещё на один период.

LED frequency and delay	
Delay in milliseconds	Frequency
1000	0.5
500	1
250	2
125	4

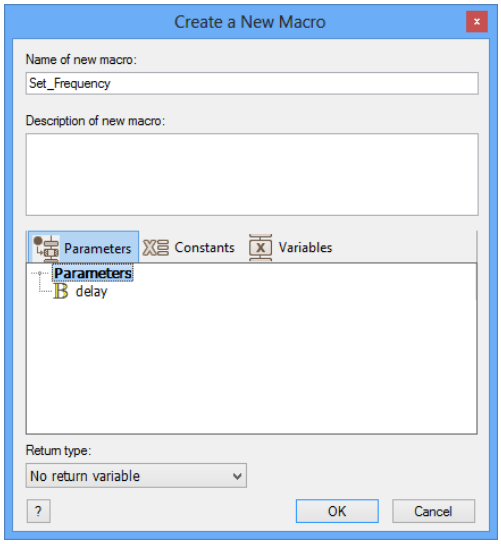
Создание переменных

Программа использует две переменные: selection для хранения состояния выключателей, используемых при выборе частоты; и delay, используемой для определения частоты, с какой будет мигать светодиод.

- В разделе Edit основного меню выберите Variables... (Edit > Variables...), чтобы открыть диалоговое окно Variable Manager или используйте Project Explorer, что рекомендуется, и будет доступно из раздела меню View (View > Project Explorer).
- Поместите курсор слева от метки Variables и щёлкните по стрелке вниз, которая появится.
- Выберите Add new, и появится диалоговое окно Create a New Variable.
- Назовите первую переменную selection.
- Оставьте тип переменной Byte.
- Щёлкните по **ОК**.

- Теперь повторите всё то же самое для второй переменной delay.

Создание макроса



- Откройте в основном меню Macro и щёлкните по New... Появится диалоговое окно Create a New Macro.
    - Назовите новый макрос Set_Frequency.

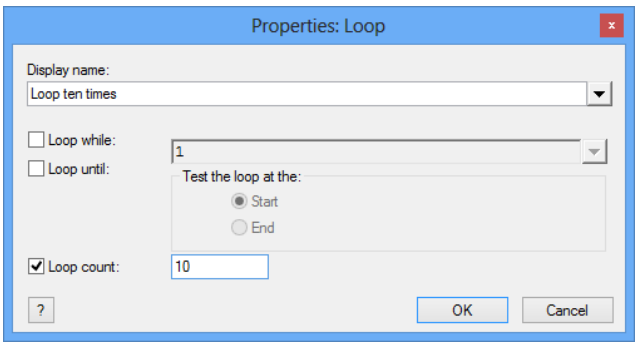
      Обратите внимание, что имя макроса должно быть однословным. Если нужно более одного слова, соедините их символом подчёркивания.

    - Поместите курсор слева от надписи Parameters и щёлкните по стрелке вниз, которая появится.
    - Щёлкните по Add new и впишите имя delay, как имя переменной в диалоговом окне, которое появится. Оставьте окно Description пустым, а тип предопределённым Byte.
    - Щёлкните по **ОК**.

Диалоговое окно создания нового макроса показано выше.

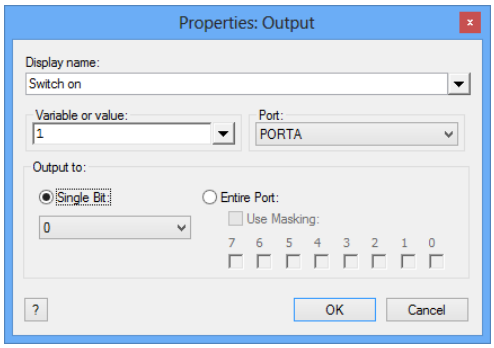
Заметьте, что это привело к созданию новой закладки, названной Set_Frequency, в рабочей области Flowcode.

- Щёлкните по этой закладке, так чтобы вы могли создать детали макроса.
- Перетащите и вставьте иконку Loop между BEGIN и END.
- Дважды щёлкните по ней, чтобы открыть диалоговое окно.
- Измените имя на Loop ten times.
- Щёлкните по **ОК**.
- Отметьте окошко Loop count и вставьте число 10.

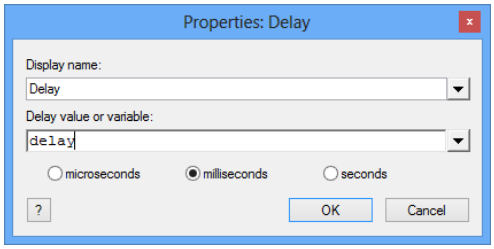


Внутри цикла:

- Перетащите иконку Output и дважды щёлкните по ней.
- Измените Display name на Switch on.
- Сконфигурируйте её на вывод значения «1» в бит 0 порта A.
- Щёлкните по **ОК**.



- За иконкой добавьте иконку Delay и дважды щёлкните по ней.
- Оставьте имя как Delay и единицы времени как milliseconds.
- В окне Delay value or variable введите имя переменной delay.
- Щёлкните по **ОК**.



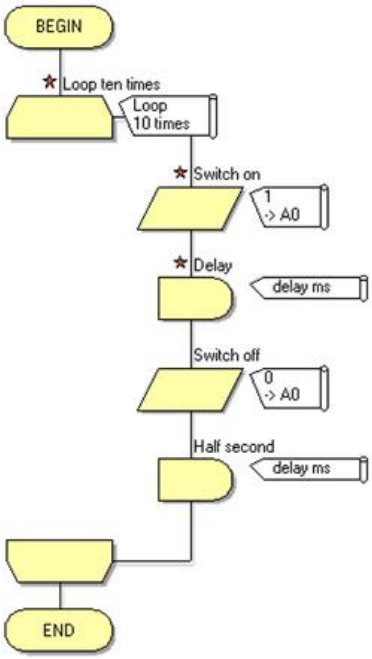
- Следом добавьте вторую иконку Output.

  Простейший способ сделать это – воспользоваться командами copy и paste, выделить предыдущую иконку, скопировать и перетащить её после вставки в нужное место.

- Дважды щёлкните по ней, переименуйте в Switch off.
- Сконфигурируйте её на вывод «0» в бит 0 порта A.
- Щёлкните по **ОК**.
- Добавьте ниже вторую иконку Delay, сконфигурировав её, как и первую.

И опять, используйте метод copy и paste. Скопируйте её, щёлкнув по иконке правой клавишей мышки и выбрав copy из выпадающего меню. Выделите иконку, за которой следует вставить скопированную. Щёлкните правой клавишей мышки и выберите paste из выпадающего меню.

Структура макроса должна быть похожа на ту, что показана ниже.



Порядок выполнения основной программы

Эта часть программы использует три выключателя для выбора частоты мигания светодиода. Порядок выполнения программы будет следующим:

- Прочитывается состояние выключателей.
- Используется это состояние для определения правильного перехода.
- Задаётся подходящее время задержки для выбранной частоты.
- Запускается макрос с заданной задержкой.
- Программа возвращается к началу для проверки состояния выключателей.

Создание основной (main) программы

- Откройте Flowcode и создайте новую программу, как описано в упражнении создание программы.
- Щёлкните и перетащите бесконечный цикл  между иконками BEGIN и END.

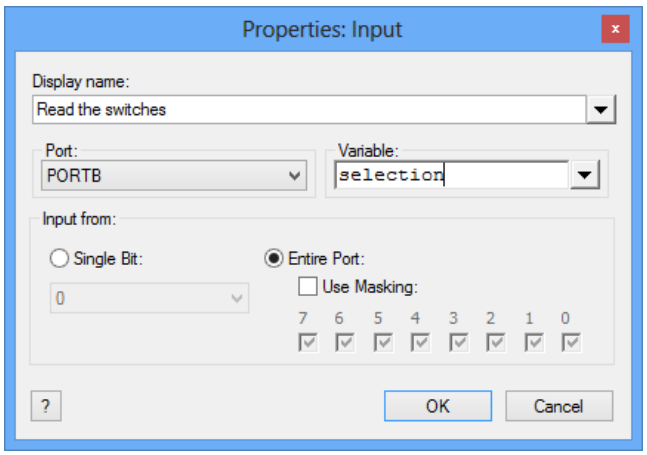
  Внутри цикла:

  - Щёлкните и перетащите иконку Input  .
  - Дважды щёлкните по ней, чтобы открыть диалоговое окно.
  - Переименуйте его в Read the switches и задайте ввод всего порта B.
  - В окне Variable введите selection.

  Три выключателя будут подключены к битам 0, 1 и 2 Port B.
  Состояние этих выключателей будет храниться в переменной selection.

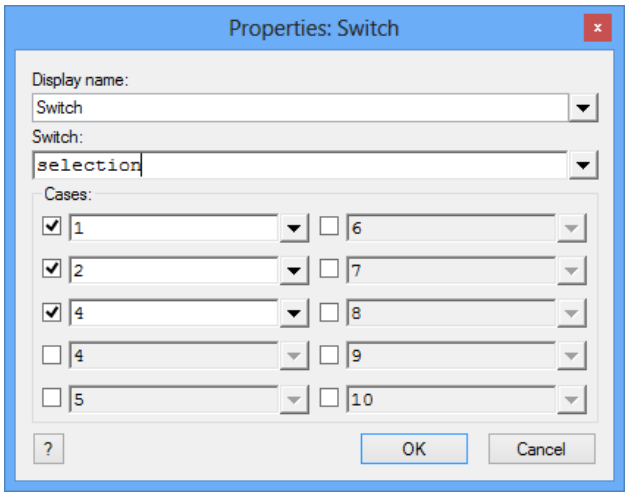
  - Щёлкните по **ОК**.

  Диалог конфигурации ввода иконки Input показан ниже.



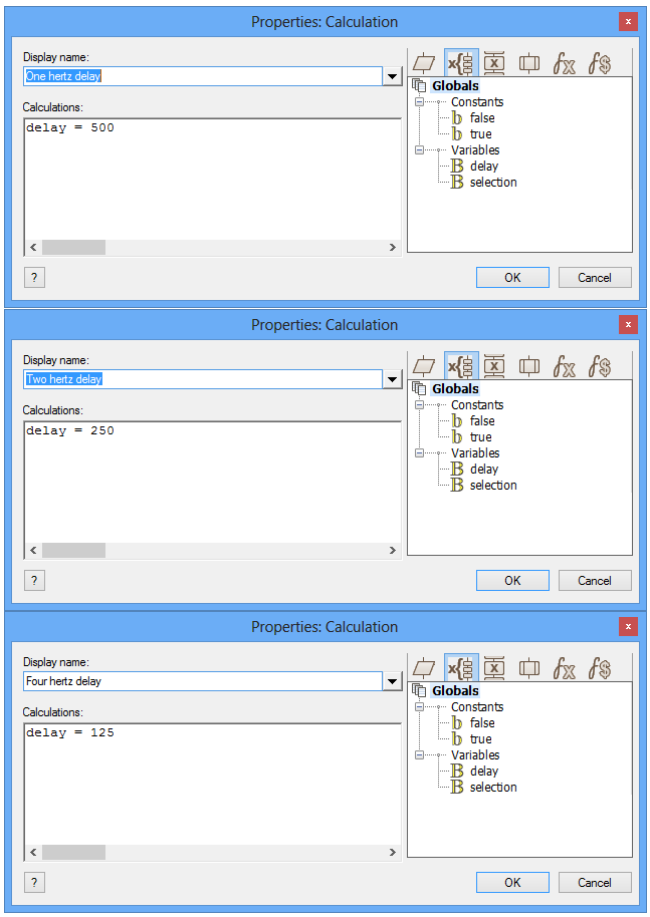
- Щёлкните и перетащите иконку Switch-case  после иконки Input.
- Дважды щёлкните по ней, чтобы открыть диалоговое окно.
- Оставьте Display name как Switch.
- В окне Switch введите имя переменной selection, которая будет переключать переходы.
- Отметьте первые три окошка Case и измените содержимое третьего с «3» на «4».
- Щёлкните по **ОК**.

Диалоговое окно Switch-case показано ниже:



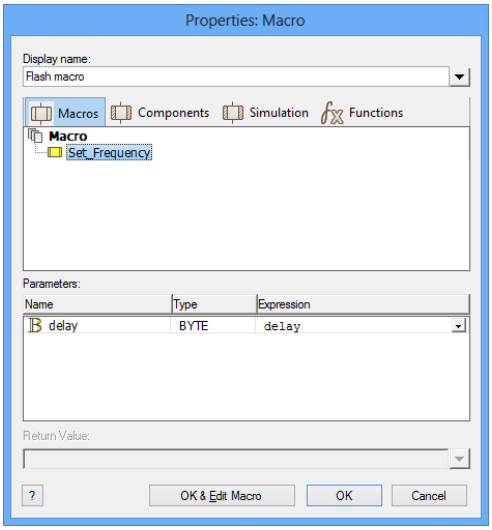
- Внутри каждой из трёх ветвей (=1, =2 и =3) щёлкните и перетащите иконку вычислений  и иконку макроса  .
- В ветви =1 дважды щёлкните по иконке Calculation и переименуйте её в One hertz delay.
- В окне Calculations введите delay = 500.
- В ветви =2 дважды щёлкните по иконке Calculation и переименуйте её в Two hertz delay.
- В окне Calculations введите delay = 250.
- В ветви =4 дважды щёлкните по иконке Calculation и переименуйте её в Four hertz delay.
- В окне Calculations введите delay = 125.

Три диалоговых окна Calculation показаны ниже:

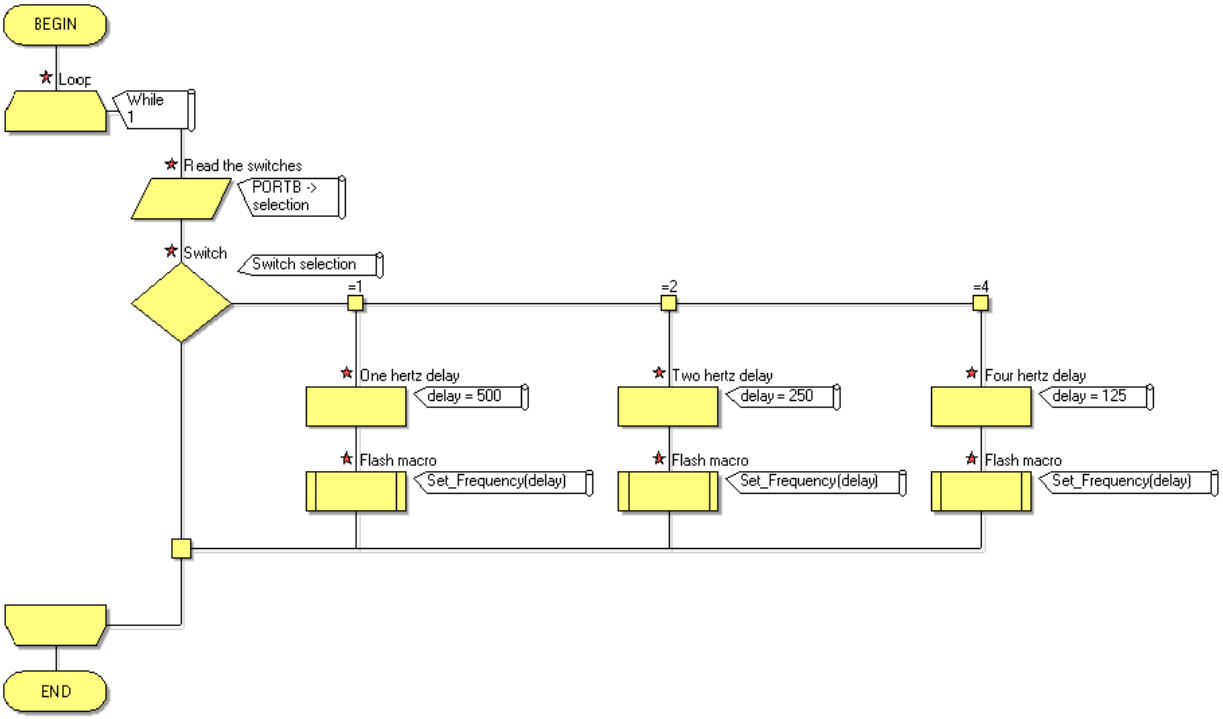


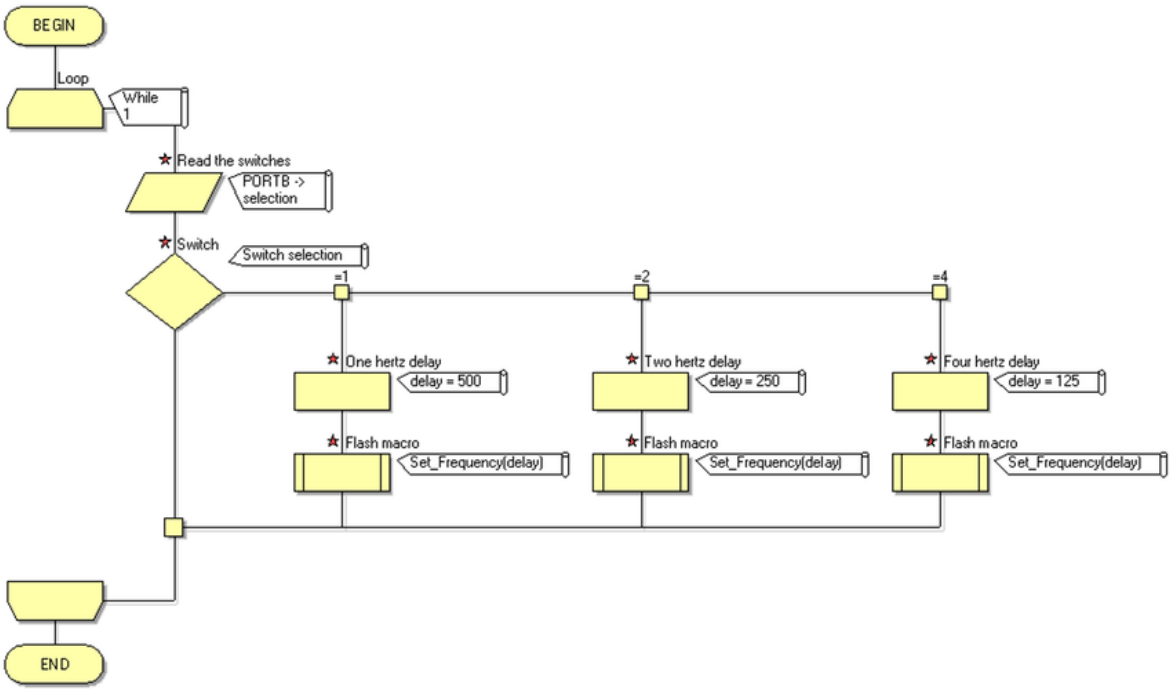
- Дважды щёлкните по каждой из иконок макро последовательно и:
    - измените Display name на Flash macro;
    - щёлкните по Set_Frequency этикетке, которая появится (единственная) в списке доступных макросов;
    - введите delay в окне Parameters в разделе Expression;
    - щёлкните по **OK**.

Результирующее диалоговое окно показано ниже:



Основная программа выглядит так





Добавление электроники

Добавление выключателей

- Найдите выключатель Push Round Panel в группе Inputs панели дополнительных компонентов.

    - Поместите курсор на картинку слева от названия выключателя и щёлкните по стрелке вниз, которая появится.
    - Щёлкните по Add to system panel.

На системной панели появится кнопка выключателя.

- Щёлкните по изображению выключателя на системной панели, чтобы выделить его.
- Щёлкните правой клавишей мышки, чтобы скопировать и вставить ещё два выключателя, используя выпадающее меню.
- Щёлкните по одному из них, чтобы выделить его, и используйте панель свойств, чтобы задать координаты X=-25, Y=10, Z=0.
- Аналогично для второго введите координаты X=0, Y=10, Z=0, а для третьего X=25, Y=10, Z=0.
- Щёлкните вновь по первому и затем щёлкните по свойству Connections на панели свойств.
- Используйте цоколёвку микроконтроллера, которая появляется, чтобы подключить выключатель к Port B, бит 0.
- Похожим образом подключите оставшиеся два выключателя к Port B, бит 1 и Port B, бит 2.

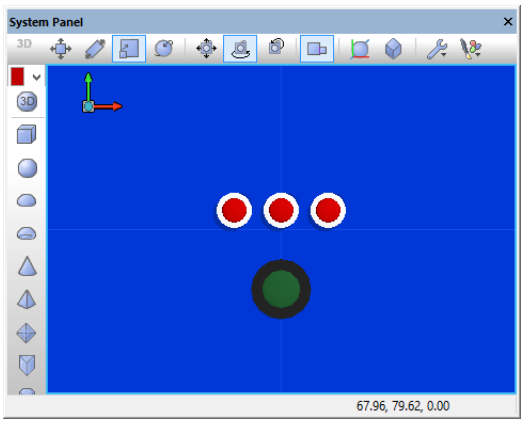
Добавление светодиода

- Найдите LED, такой как LED 5mm Panel в группе Outputs панели дополнительных компонентов.

  - Поместите курсор на картинку левее названия LED 5mm Panel и щёлкните по стрелке вниз, которая появится.
  - Щёлкните по Add to system panel.
  - Выделите LED на системной панели и перетащите его в подходящее место ниже выключателей.
  - Обратитесь к панели свойств, если свойство Connection показывает, что по умолчанию светодиод подключён к Port A, бит 0, оставьте так, иначе подключите светодиод к этому выводу порта.

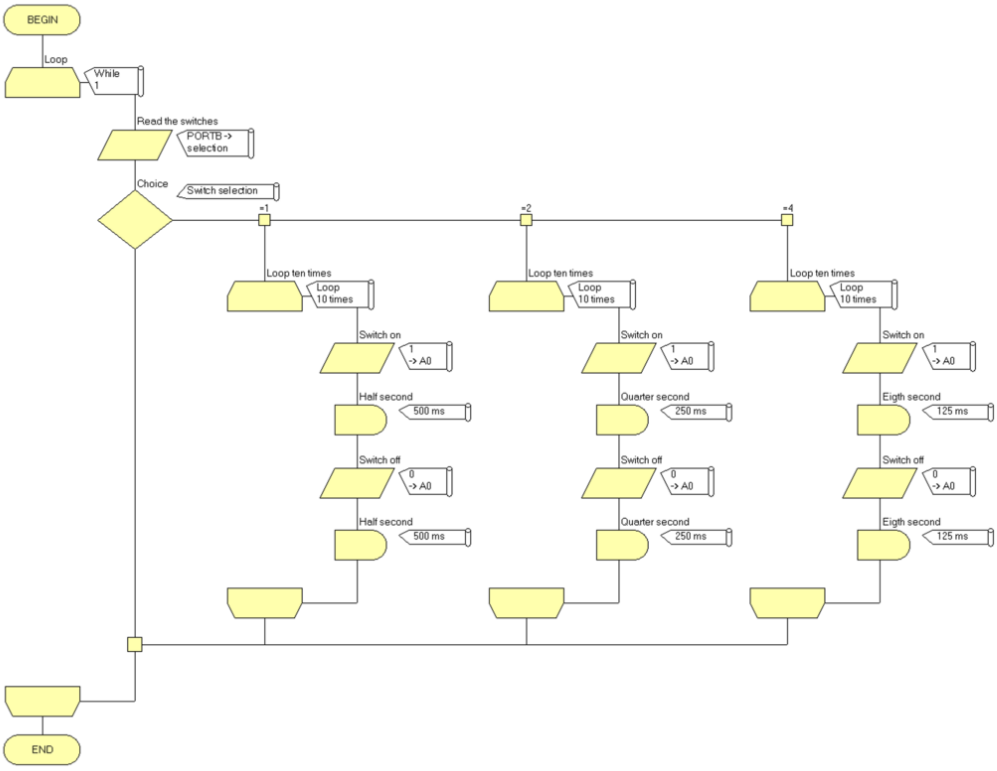
Системная панель

Системная панель должны выглядеть так:



Альтернатива

В конечном счёте, программа выглядит гораздо компактнее при использовании макроса. Программа, выполняющая ту же работу, но не использующая макрос:



18. Использование компонентного макроса

Компонентные макросы – это фрагменты кода, которые были написаны для поддержки компонентов, включённых в Flowcode 6.

Они берут на себя все сложности при использовании компонентов.

Это упражнение использует их для управления довольно сложным, но очень часто используемым устройством, Liquid Crystal Display (LCD) – жидкокристаллическим дисплеем. В нашем случае дисплей используется для отображения времени таймера.

Первая часть программы использует Component Macros для создания ядра временной последовательности. Вторая часть осуществляет включение лампы на десять секунд, когда выключатель нажат.

Основная временная последовательность

В основе программы секция, которая задаёт установки LED для отображения времени.

Порядок выполнения программы для отображения времени будет:

- Инициализация LCD, используя команду Start макроса.
- Установка курсора в заданную позицию на LCD, используя Cursor.
- Отображение текста Elapsed time, используя PrintString макроса.
- Установка времени в ноль.
- Изменение позиции курсора, используя Cursor макроса.
- Отображение времени.
- Пауза в секунду.
- Увеличение времени.

- Отображение нового времени.
- Возвращение к началу цикла и повторение процесса после секундной задержки.

Создание программы

- Откройте Flowcode и создайте новый проект.

Создание основной последовательности

Добавление LCD

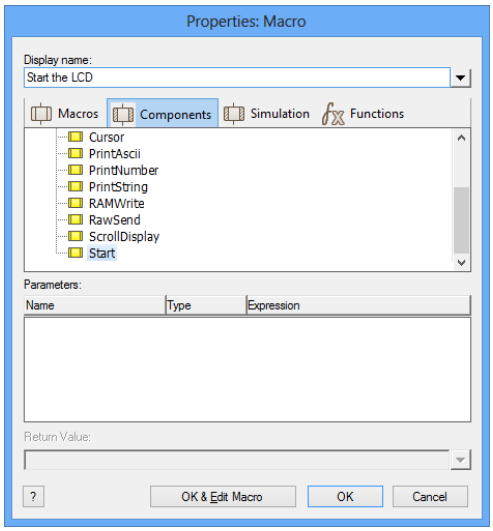
- Найдите LCD в группе Outputs панели дополнительных компонентов.
    - Поместите курсор на картинку левее надписи LCD и щёлкните по стрелке вниз, которая появится.
    - Щёлкните по Add to system panel.
    - Увеличьте образ LCD, чтобы он стал удобен для чтения.

Инициализация LCD

- Щёлкните и перетащите бесконечный цикл между иконками BEGIN и END.
- Внутри цикла:
    - Щёлкните и перетащите иконку компонентного макроса  .
    - Дважды щёлкните по ней, чтобы открыть диалоговое окно, так чтобы вы могли сконфигурировать иконку.

Программа знает, какой компонент вы добавили на системную панель или панель управления и соответственно модифицирует список доступных команд.
На закладке Components появится LCD с открывающимся списком команд.

- Переместитесь в нижнюю часть списка и щёлкните по команде Start.
- Переименуйте Display name в Start the LCD.
- Щёлкните по **ОК**.
- Диалоговое окно показано ниже.



Настройка дисплея

Первая задача – определить, где будет отображаться текст.

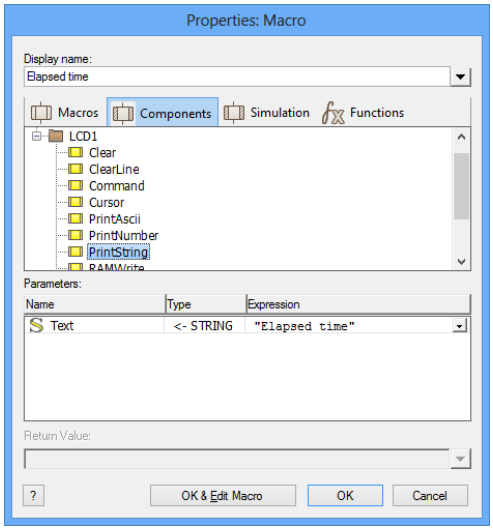
Это потребует другого компонентного макроса, названного Cursor. Этим определяется ячейка LCD, где начнётся отображение текста. LCD имеет сетку из 32 ячеек, организованных в две строки по 16.

Верхняя строка имеет значение y=0, а нижняя y=1. Шестнадцать горизонтальных ячеек имеют значения от x=0 до x=15.

- Щёлкните и перетащите вторую иконку Component macro, вставьте её под первой.
- Дважды щёлкните по ней, чтобы открыть диалоговое окно.
- Переместитесь к команде Cursor.
- Переименуйте Display name в Move the cursor.
- В разделе Parameters задайте значения x в 2, а y в 0.
- Щёлкните по **ОК**.

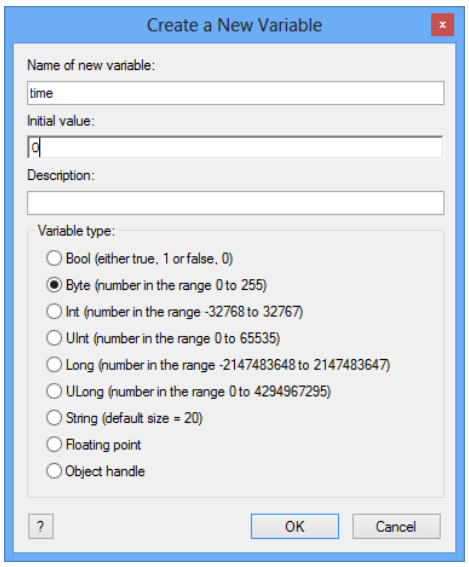
Следующий Component Macro задаёт выводимый текст, начинающийся с этого места.

- Щёлкните и перетащите третью иконку Component macro, вставьте её под предыдущей.
- Дважды щёлкните по ней, чтобы открыть диалоговое окно.
- Переименуйте Display name в Elapsed time.
- Переместитесь к команде PrintString и щёлкните по ней.
- В окне Expression введите "Elapsed time" (убедитесь, что кавычки на месте).
- Щёлкните по **ОК**.
- Диалоговое окно показано ниже.



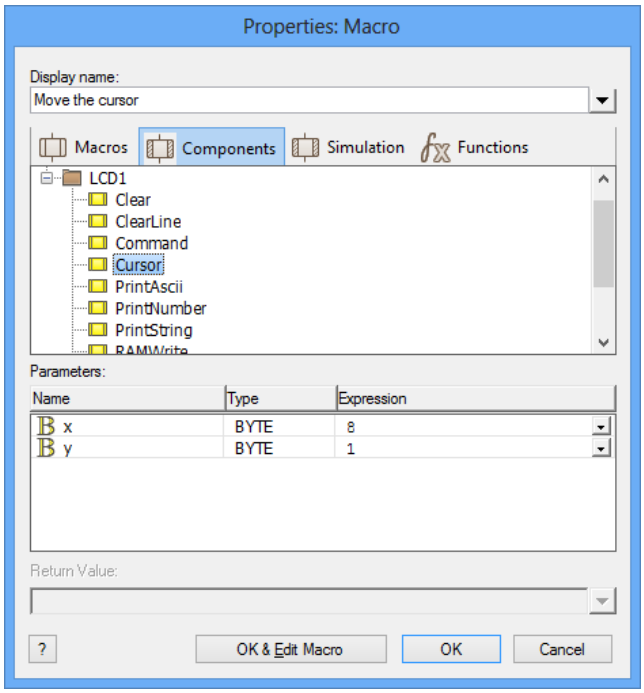
Отображение истекшего времени

- В разделе Edit основного меню щёлкните по Variables..., чтобы открыть диалоговое окно Variable Manager.
- Поместите курсор поверх названия Variables слева и щёлкните по стрелке вниз, которая появится.
- Выберите Add new, и откроется диалоговое окно Create a New Variable.
- Назовите новую переменную time с начальным значением равным нулю.
- Оставьте тип переменной Byte.
- Щёлкните по **ОК**.
- Диалоговое окно показано ниже.



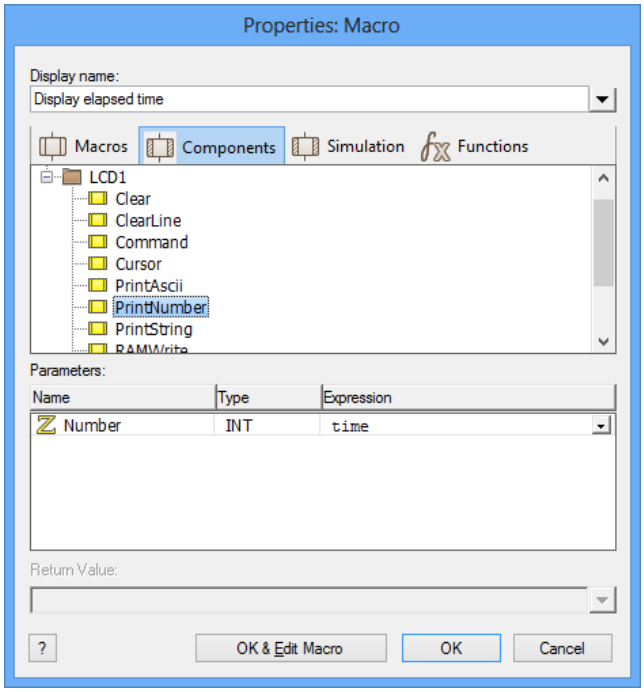
Следующий Component Macro перемещает курсор, чтобы показать истекшее время за названием Elapsed time. Значение y меняется на y=1, а значение x на x=8.

- Щёлкните и перетащите четвёртый Component Macro следом за третьим.
- Дважды щёлкните по нему, чтобы открыть диалоговое окно.
- Переместитесь к команде Cursor.
- Переименуйте Display name на Move the cursor.
- В разделе Parameters задайте значения x = 8, а y = 1.
- Щёлкните по **ОК**.



- Щёлкните и перетащите пятую иконку Component macro под четвёртую.
- Дважды щёлкните по ней, открывая диалоговое окно.
- Переместитесь к команде PrintNumber и щёлкните по ней.
- Переименуйте Display name в Display elapsed time.
- В окне Expression впечатайте имя переменной time.
- Щёлкните по **ОК**.

- Диалоговое окно пятого Component Macro показано ниже.



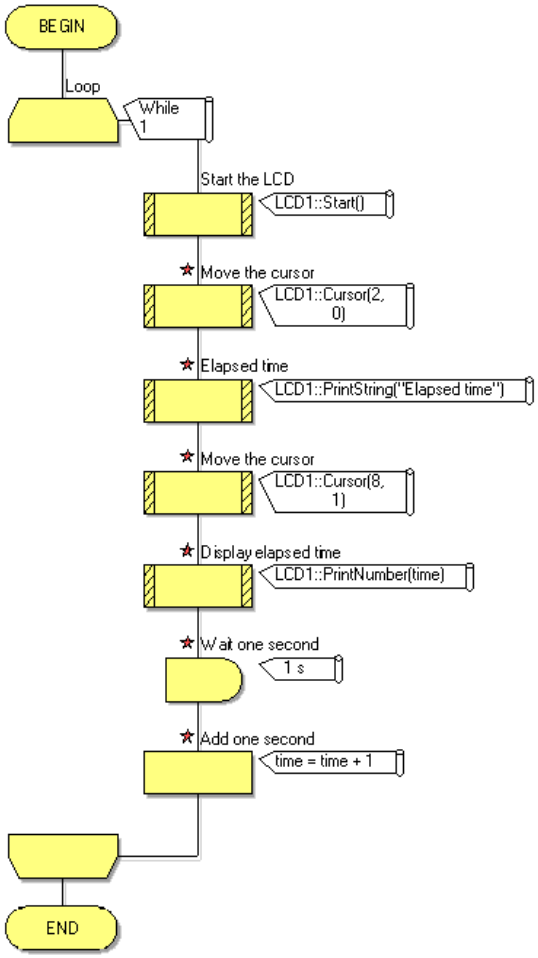
Последние штрихи

- Щёлкните и перетащите иконку Delay следом за пятым Component Macro.
- Дважды щёлкните по ней, чтобы открыть диалоговое окно.
- Переименуйте её в Wait one second.
- Сконфигурируйте её, чтобы получить задержку в одну секунду.
- Щёлкните по **ОК**.
- Щёлкните и перетащите иконку Calculation, вставьте за Delay.
- Дважды щёлкните по иконке, чтобы открыть диалоговое окно.
- Переименуйте её в Add one second.
- В разделе Calculations впишите time = time + 1.
- Щёлкните по **ОК**.

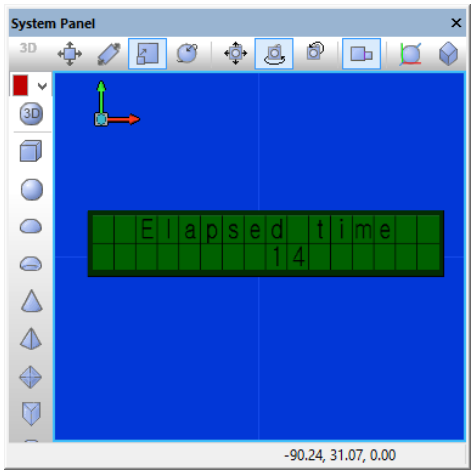
Эффект последнего вычисления в увеличении числа, которое хранится в переменной time, на единицу (это также называется инкрементированием переменной time).

Тестирование ядра

Диаграмма ниже показывает программу на этом этапе.



Сохраните программу как Timer и можете протестировать её.



Расширение программы

Полная программа создаётся для:

ожидания, пока выключатель будет кратковременно нажат;
включения лампы;
времени включения десять секунд;
выключения лампы.

Чтобы это сделать, порядок выполнения должен быть изменён следующим образом:

- Нажат ли выключатель?
- Если нет, вернуться назад и проверить выключатель вновь.
- Если был нажат:
    - включить лампу;
    - пройти через задержку в десять секунд;
    - выключить лампу.

Добавление выключателя

- Найдите Push Round Panel выключатель в группе Inputs панели дополнительных компонентов.

    - Поместите курсор на картинку слева и щёлкните по стрелке вниз, которая появляется.
    - Щёлкните по Add to system panel.
    - Щёлкните по выключателю на системной панели, чтобы выделить его, и перетащите в удобное место.
    - На панели свойств щёлкните по этикетке Unconnected, следующей за Connection.
    - Появится цоколёвка микроконтроллера.
    - Щёлкните по прямоугольнику, представляющему вывод RA1/AN1, чтобы подключить выключатель к Port A бит 1 микроконтроллера.

Добавление светодиода

- Найдите LED, например, LED 5mm Panel в группе Outputs панели дополнительных компонентов.
    - Поместите курсор на картинку слева и щёлкните по стрелке вниз, которая появляется.
    - Щёлкните по Add to system panel.
    - Выделите светодиод на системной панели и перетащите его в подходящее место.
    - Проверьте, чтобы в свойстве Connection LED был подключен Port A, бит 0.

Модификация программы

- Сразу за иконкой Begin перетащите и вставьте иконку Connection Point (Declare Connection Point)  .
- Следом добавьте иконку Input, дважды щёлкните по ней, открывая диалог.
- Измените Display name на Read the switch.
- Создайте новую переменную, назовите её switch. Один из способов – щёлкнуть по стрелке вниз в конце окошка Variable.

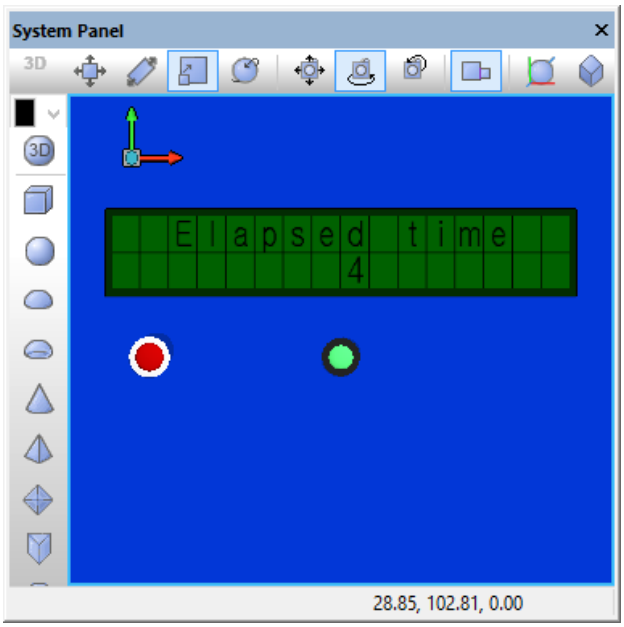
    - Затем щёлкните по стрелке вниз рядом с этикеткой Variables и выберите Add new.
    - Появится диалоговое окно Create a New Variable. Введите switch как имя переменой, оставьте тип переменной как Byte.
    - Щёлкните по **ОК**.

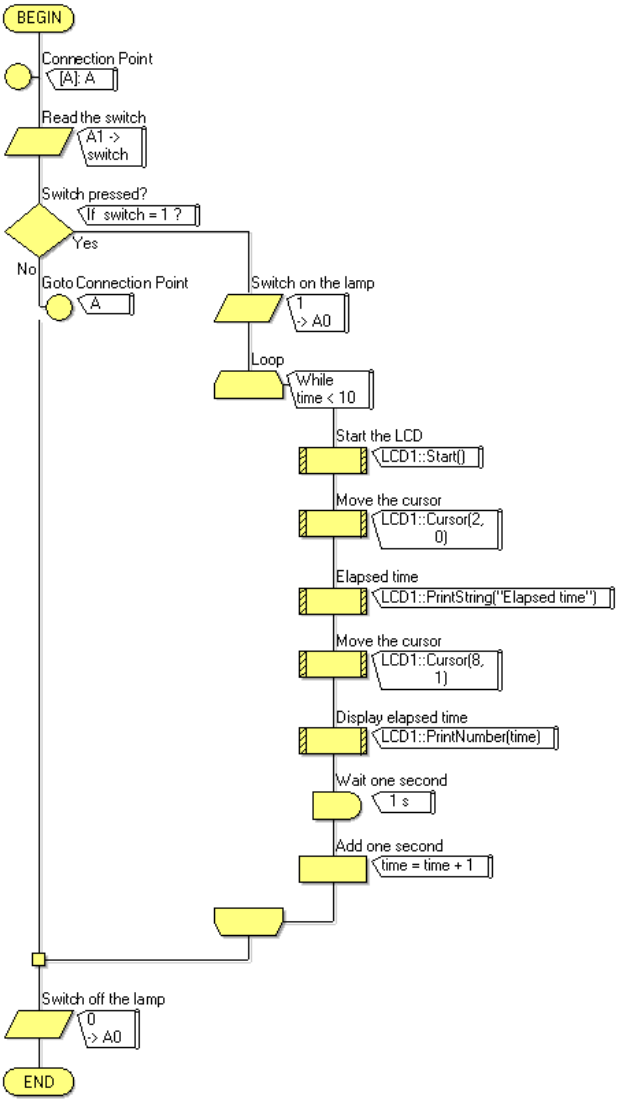
- В окне Variable добавьте имя новой переменной switch.
- В окне Port выберите PORT A. Сконфигурируйте остальное в разделе Input from как Single Bit, 1.

- Щёлкните по **ОК**.
- Теперь перетащите иконку Decision и двойным щелчком откройте диалог.
    - Переименуйте иконку в Switch pressed?.
    - В разделе условия If введите switch = 1.
    - Щёлкните по **ОК**.
- В ветвь No перетащите иконку Connection Point (Goto Connection Point)  .
- В ветвь Yes перетащите иконку Output.
    - Дважды щёлкните по ней, чтобы открыть диалоговое окно.
    - Измените Display name на Switch on the lamp.
    - В окне Variable or value впишите «1».
    - Сконфигурируйте остальное так: PORT A, Single Bit, 0.
    - Щёлкните по **ОК**.
- Выделите и перетащите иконку Loop с ранее созданным ядром программы в ветвь Yes ниже.
    - Дважды щёлкните по ней, чтобы сконфигурировать цикл.
    - Оставьте имя, как есть, и отмеченное окошко Loop while.
    - В окошко условия введите time<10.
    - Щёлкните по **ОК**.
- После цикла, в конце программы, поместите иконку Output.
    - Дважды щёлкните по ней, чтобы переименовать в Switch off the lamp и сконфигурировать: значение «0» в Port A, бит 0.
    - Щёлкните по **ОК**.

Окончательно программа должна быть похожа на приведённую ниже.

А вы можете запустить симуляцию и проверить работу программы:





19. Использование прерываний

Прерывание – это способ немедленно привлечь внимание микроконтроллера.

Прерывания используются очень широко, как в программах для микроконтроллеров, так и для микропроцессоров. Например, клавиатура и мышка вашего компьютера, скорее всего, используют прерывание для беседы с CPU.

Прерывания могут использоваться и для энергосбережения. Во многих устройствах, питающихся от батарей, микроконтроллер переходит в режим сна, когда он не активен, в котором потребляет минимум энергии. Прерывание используется для пробуждения микроконтроллера и приведения его в рабочее состояние, когда это требуется.

Это упражнение показывает, как использовать прерывание, чтобы определить, когда выключатель включён (внешнее прерывание).

Опрос против прерывания

Очень часто микроконтроллер подключен к большому числу периферических устройств ввода – кнопкам, датчикам, памяти, таймерам и т.п. Есть два основных пути обслуживания любого из этих устройств микроконтроллером:

- опрос – каждое устройство «опрашивается» последовательно, есть ли у него данные для передачи контроллеру;
- прерывание – позволяющее устройству прервать работу, выполняемую в данный момент контроллером.

Упражнение поможет увидеть разницу между опросом и использованием прерывания.

Система

Это упражнение создаёт систему со светодиодами, управляемыми двумя выключателями:

- Выключатель 1 опрашивается программой в регулярные интервалы.
- Выключатель 2 инициирует прерывание.

Опрашиваемый выключатель, выключатель 1, проверяется в регулярные (и предопределённые) интервалы времени – каждые шесть секунд в данном примере. Выключатель 2 подключен так, что как только он срабатывает, процессор останавливает свою работу и переходит на подпрограмму обработки прерывания, Interrupt Service Routine (ISR). В процессе этого запоминается адрес возвращения, так что по окончанию ISR процессор может вернуться к тому месту, где остановился в основной программе. В Flowcode подпрограмма обработки прерывания – это макрос, сконфигурированный, как любой другой.

Программа

Порядок работы программы:

- Проверить, нажат ли выключатель 1.

  Если нет, заставить медленно мигать красный светодиод.
  Если да, заставить медленно мигать жёлтый светодиод.

  - Вернуться к началу и повторить весь процесс.
  - Когда бы выключатель 2 не был нажат, немедленно заставить мигать оба светодиода быстро десять раз, а затем вернуться в основную программу.

Основная программа

- Начните в Flowcode новую программу, используя микроконтроллер по умолчанию.
- Убедитесь, что системная панель видна, если необходимо выберите её в разделе View основного меню. Когда системная панель видна, в меню рядом с ней стоит галочка.
- Перетащите иконку Loop и вставьте её между иконками BEGIN и END.
- Внутрь цикла перетащите иконку Input с панели программных компонентов.
- Перетащите и вставьте иконку Decision после иконки Input.

Далее, вставим иконки для управления тем, что происходит, когда выключатель не нажат (а программа будет следовать по ветви No).

- Перетащите иконку Output в ветвь No. Эта иконка будет включать красный LED.
- Перетащите иконку Delay и вставьте за иконкой Output. Она заставит гореть LED 3 секунды.
- Добавьте ещё одну иконку Output. Она выключит светодиод.
- Добавьте вторую иконку Delay следом. Это выключит светодиод на 3 секунды.

Затем программа возвращается к началу цикла и проверяет состояние выключателя вновь.

Теперь добавим иконки для управления процессом, когда выключатель нажат, а программа следует по ветви Yes:

- Перетащите и вставьте в ветвь Yes иконку Output.
- Перетащите иконку Delay ниже.
- Пополните это второй иконкой Output.
- И второй иконкой Delay.

Затем, как и раньше, программа вернётся к началу, и будет проверять состояние выключателя.

Добавление выключателя 1 и LED

- Найдите в группе Inputs и щёлкните по стрелке вниз рядом с выключателем, отмеченным как Toggle Metal PCB  .

  Щёлкните по Add to system panel.
  Сконфигурируйте свойства так, чтобы подключить к Port B, бит 1.

- Найдите в группе Outputs и щёлкните по стрелке вниз рядом с надписью LED Array  .

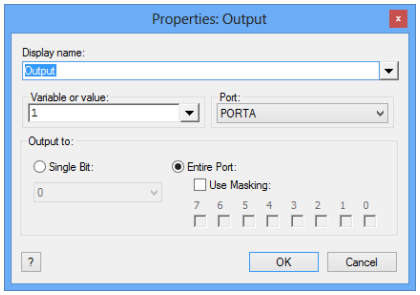
  Щёлкните по Add to system panel.

- Щёлкните по компоненту и сконфигурируйте его свойства следующим образом:

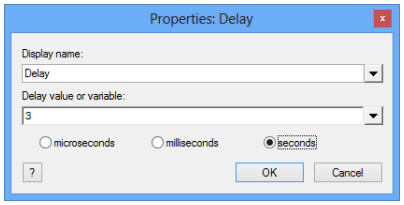
  пусть свойство Count будет «2», чтобы было 2 LED в массиве;
  измените цвет LED0 на красный, а LED1 на жёлтый (в разделе Same… должно быть No);
  LED0 будет зажигаться, когда в порт A отправляется значение 1, а LED1, когда отправляется 2;
  проверьте, что свойство Connections показывает, что они подключены к Port A.

Дважды щёлкните по каждой иконке в программе, чтобы сконфигурировать их так, как показано на следующих картинках:

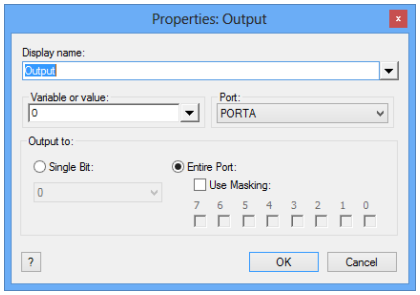
Конфигурация иконок



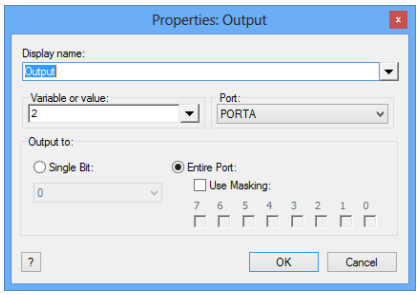
Включение красного светодиода



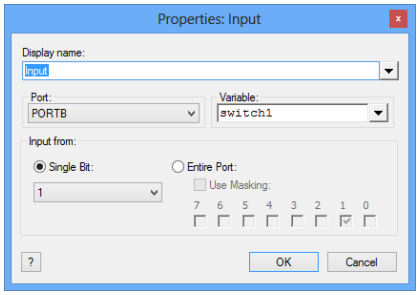
Пауза 3 секунды



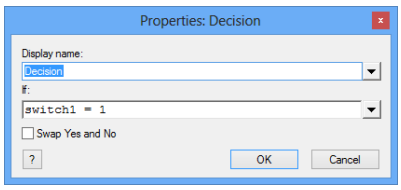
Выключение светодиодов



Включение жёлтого светодиода



Конфигурация выключателя 1



Конфигурация ветвления

Запустите симуляцию, чтобы проверить правильность работы программы.

Добавление выключателя 2

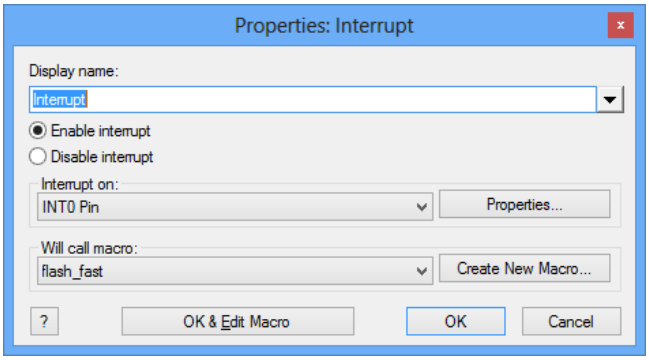
- Добавьте второй выключатель Toggle Metal PCB на системную панель.

  Сконфигурируйте его свойства так, чтобы он был подключен к Port B, бит 0 – биту аппаратного прерывания.

- Перетащите иконку прерывания 

между BEGIN и началом цикла (хотя прерывание работает в любом месте программы).

  Дважды щёлкните по иконке и сконфигурируйте её, как показано ниже.

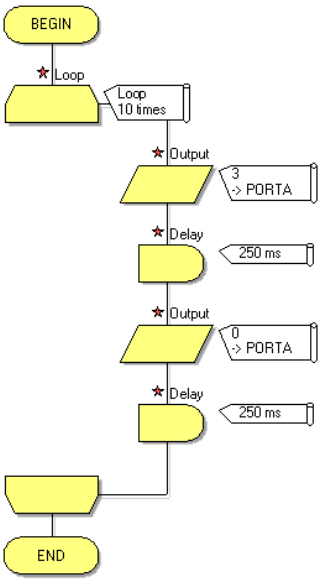


- Следующая задача – создать подпрограмму поддержки прерывания. Щёлкните по кнопке **Create New Macro**.
- Откроется диалог Create a New Macro.

  Дайте макросу имя flash_fast.
  Щёлкните по **ОК**.

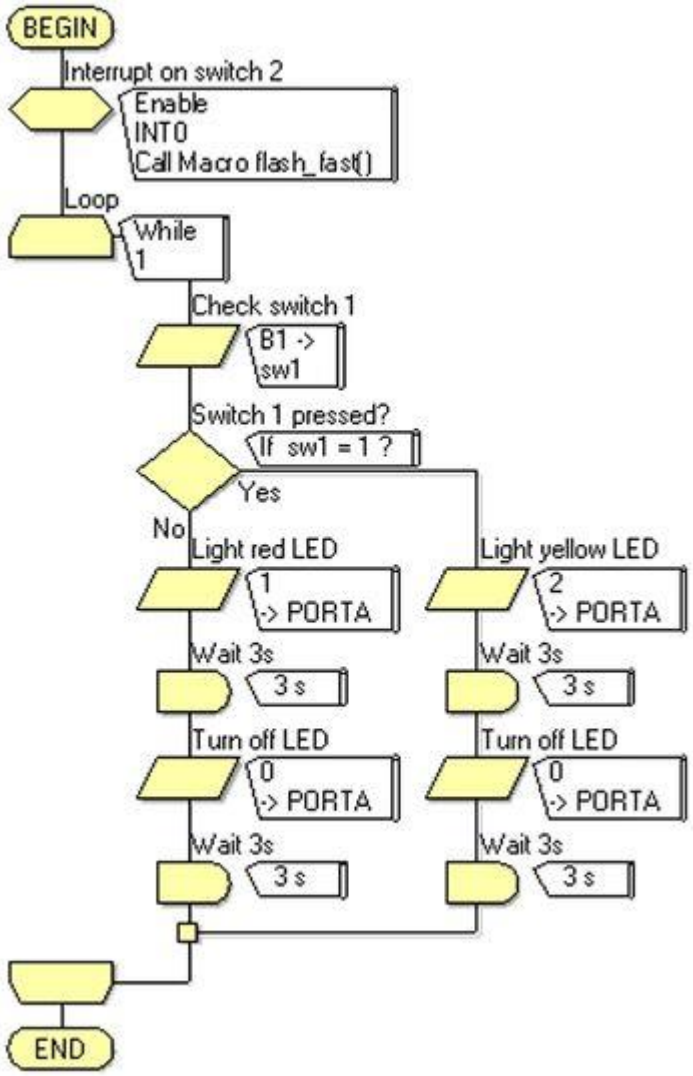
- В диалоговом окне Properties:Interrupt щёлкните по кнопке **OK & Edit Macro**.

  Создайте программу, показанную ниже.
  Отправьте значение «3» (двоичное 00011) в порт A, что включит оба светодиода.



Основная программа

Ваша программа теперь должна выглядеть похожей на ту, что ниже.



Тестирование системы

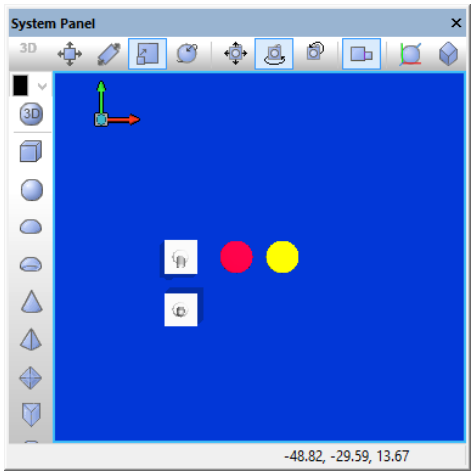
Ранее вы протестировали эффект от выключателя 1. Теперь вы можете протестировать действие выключателя 2.

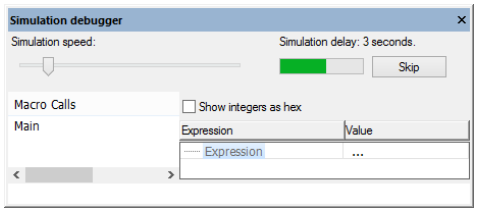
- Запустите симуляцию, и ещё раз проверьте влияние выключателя 1 на светодиоды.
- Когда Flowcode симулирует трёхсекундную паузу, кратковременно включите и выключите выключатель 2.

Трёхсекундная симуляция должна остановиться на «полпути», а оба светодиода должны быстро замигать десять раз, как задано в ISR.

После этого микроконтроллер возвратится к основной программе с того места, где был остановлен, на трёхсекундную паузу.

- Повторите тестирование, пока не убедитесь, что система работает правильно.





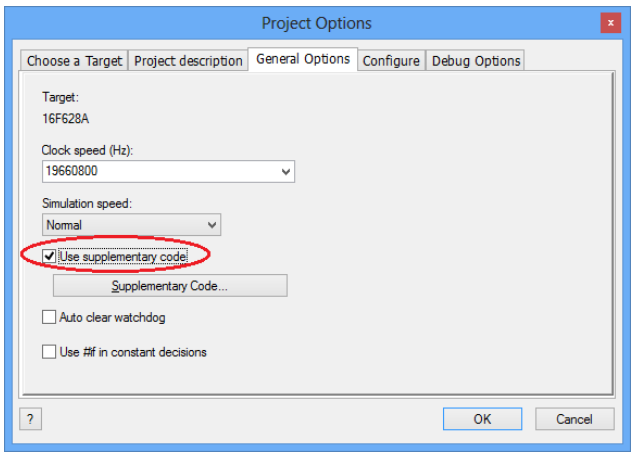
20. Вставка кода в Flowcode

Код Си – это язык высокого уровня, широко используемый в индустрии. Он породил ряд других языков программирования, таких как C#, C++, Java и Python.

Программа Flowcode вначале компилируется в Си, перед тем как «рухнуть вниз» непосредственно в hex код.

Есть два механизма добавления вашего собственного Си кода в программу в Flowcode:

- использовать иконку вставки C Code  ;
- использовать Use supplementary code в Project Options, как показано ниже.



Почему бы вам этого захотелось?

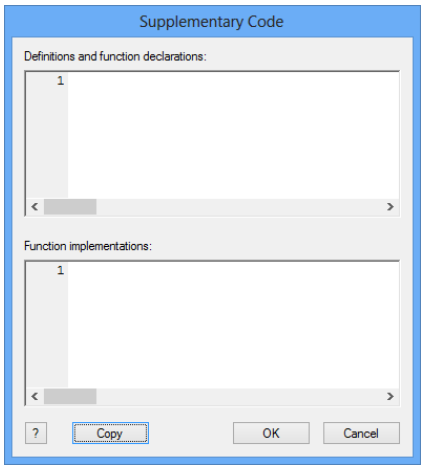
- Программа Flowcode компилируется в Си, но производит много строк кода.

  Так сделано потому, что нужно скопировать много функций или структур, которые разработчики хотели использовать.
  Программистам, знакомым с Си, может захотеться использовать код более эффективно, включив несколько строк кода, заменяющих множество строк, генерируемых Flowcode.

- Опытные программисты могут вставить небольшие секции кода Си в порядке уменьшения длины или сложности программы Flowcode.
- Студенты, изучающие программирование на Си, могут протестировать короткие секции, вставляя их в программу Flowcode. Это позволяет избежать написания полной программы на Си.

Добавление дополнительного кода

Эта возможность используется, когда вы имеете блоки кода, содержащие подпрограммы, определения, таблицы преобразования и т.п.



Определения и объявление функции (Definitions and function declarations)

Это первая секция. Она позволяет пользователю инициализировать программу, добавляя структуры подобные defines, includes и другие объявления функций. Эта секция должна быть расположена в начале файла Си, так чтобы код был «виден» всем частям программы.

Реализация функции (Function implementations)

Это вторая секция, позволяющая пользователю добавить основной код функции.

Так структурированный код доступен для любого макроса Flowcode и наоборот.

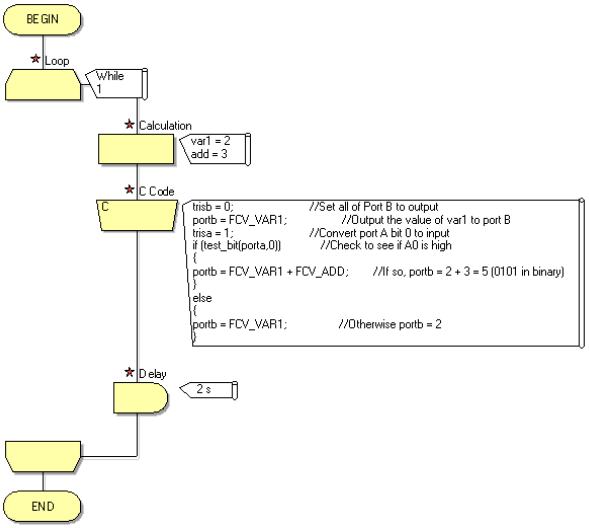
Предупреждение

Удостоверьтесь, что добавляемый код Си корректен!

Flowcode не симулирует код Си, так что программа, которая использует добавленный код, может симулироваться не вполне корректно. Если компиляция неудачна, причиной может быть то, что добавленный код имеет ошибки.

Использование иконки Code

- Откройте Flowcode.
- На начальном экране щёлкните по New project, чтобы создать новую программу.
- Добавьте иконки, показанные ниже:



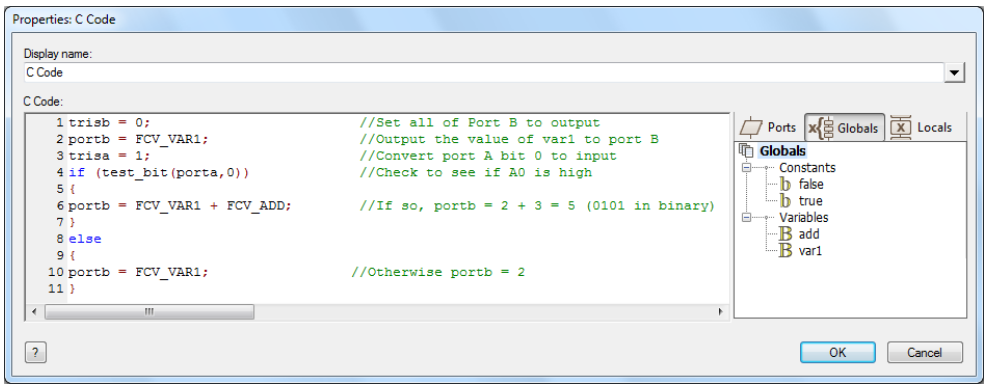
  - иконку цикла, сконфигурированную как бесконечный цикл;
  - иконку вычислений;
  - используйте её, чтобы создать две байтовые переменные var1 и add;
  - задайте им следующие значения: var1 = 2, add = 3;
  - иконку Code  – детали ниже;
  - иконку паузы, сконфигурированную на две секунды.

Код Си

Сконфигурируйте иконку Code, вставив следующий код, в точности как написано здесь:

```
1  trisb = 0;                          //Set all of Port B to output
2  portb = FCV_VAR1;                   //Output the value of var1 to port B
3  trisa = 1;                          //Convert port A bit 0 to input
4  if (test_bit(porta,0))              //Check to see if A0 is high
5  {
6  portb = FCV_VAR1 + FCV_ADD;     //If so, portb = 2 + 3 = 5 (0101 in binary)
7  }
8  else
9  {
10 portb = FCV_VAR1;                  //Otherwise portb = 2
11 }
```

Иконка Code должна быть похожа на ту, что ниже:



Программа распознаёт инструкции и комментарии, и раскрашивает их соответственно.

Некоторые замечания по коду:

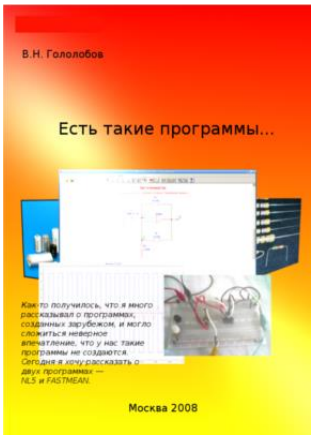
- Порты PIC-контроллеров двунаправленные – они могут работать и как входные, и как выходные.
- Их поведение определяется двумя регистрами: TRISA (для Port A) и TRISB (для Port B).
    - Значение «0» в регистре управления делает соответствующий бит порта работающим на выход.
    - Значение «1» в регистре управления делает соответствующий бит порта работающим на вход.
- Команды завершаются добавлением точки с запятой (;).
- Символы «//» используются для обозначения комментария – они помогают понять программу. Комментарии игнорируются при компиляции.
- Когда переменные определяются в программе Flowcode, к ним должны быть добавлены префиксы FCV_ (к имени переменной), переменные пишутся в верхнем регистре.
- Конструкция if-else проверяет условие - Is Port A bit 0 high?
    - Если условие истинно, значение двух переменных складывается вместе и отправляется в порт B.
    - Если условие ложно, только значение переменной var1 отправляется в порт B.
- Всё это должно было бы быть сделано, используя «чистый» Flowcode, но это потребовало бы добавления ещё пяти иконок в программу.

Тестирование

Как сказано ранее, Flowcode не симулирует добавленный код Си, так что нет смысла в симуляции.

Вместо этого оттранслируйте программу, загрузите в микросхему PIC-контроллера, который вы определили в качестве целевого.

Вы можете использовать модуль из состава E-Blocks: EB-008 Multiprogrammer, который позволит вам проверить работу программы.



Редакция в дополнение рассказа о шестой версии программы Flowcode сочла полезным включить в этот выпуск предыдущие рассказы автора о программе. Вот первый фрагмент из книги «Есть такие программы…»

Хотя речь идёт о третьей версии, многое вполне можно отнести и к следующим версиям программы Flowcode.

Полностью книга доступна для свободной загрузки с авторского сайта http://vgololobov.narod.ru/

Микроконтроллеры

Я часто просматриваю форумы. Не только радиолюбительские. Когда несколько лет назад я начал работать в операционной системе Linux, а вначале это были даже BeOS и QNX, мне очень хотелось все «потрогать», все «попробовать». Такие пробы приводили к постоянным проблемам, отчего очень часто появлялось искушение написать на форуме: «Помогите!» или «Все рухнуло, что делать?». Ответ на такие просьбы о помощи очевиден — переустановить систему заново. И я старался избежать вопросов, пытаясь найти, и не поверите — находил, ответы в уже заданных кем-то вопросах, просматривал архивные разделы, просматривал другие форумы.

Но иногда я все-таки не могу избежать искушения и пишу что-то на форумах. Общение оказывается весьма полезным. Каждый раз я открываю нечто новое для себя. Я уже говорил, что мое внимание к программам NL5 и FASTMEAN привлек мой Интернет-знакомый Александр. Другой Александр привлек мое внимание к программам для расчета акустических систем. Идею написать предыдущую главу мне подсказал администратор сайта «Радиотехник». А недавно, общаясь с администратором сайта «Технический форум в мастерской Левши», я открыл существования очень полезной, как мне кажется, при освоении микроконтроллеров программы FlowCode.

Не знаю, прав ли я, но... программа FlowCode, как самый простой и легкий путь к применению микроконтроллеров в своей практике для начинающих «летать».

Полет первый

Современные микроконтроллеры — это хорошо продуманные устройства, позволяющие существенно упростить построение схем. Очень часто в своем корпусе они имеют встроенные компараторы, АЦП, модули работы с сетью USART или радиоканалом RF. То, как настроить работу с этими устройствами, как использовать эти устройства, лучше прочитать в документации к конкретному типу микроконтроллера — никто лучше производителя не знает этого.

Но, если не пытаться при первом знакомстве с микроконтроллером использовать в полной мере все, что в нем заложено, то можно рассматривать контроллер, как обычную цифровую микросхему. У нее есть выводы, объединенные в группы, называемые портами. Выводы, в целом, могут как-то при программировании устанавливаться в свойствах быть входами или выходами микросхемы. Выходы микроконтроллера, как любой цифровой микросхемы, устанавливаются в 0 или 1. При программировании микроконтроллера в определенном месте памяти устанавливаются режимы работы: будет ли контроллер использовать встроенный тактовый генератор или последний будет внешним, с кварцем или RC цепью; будет ли контроллер использован для перехода в режим ожидания; на какой скорости будет работать USART и т.д. Часто эти параметры устанавливаются в «слове конфигурации» контроллера, как биты 0 или 1. Эта конфигурация записывается при вводе программы в контроллер с программатора и остается неизменной в дальнейшем.

Все, что относится к слову конфигурации, мы можем рассмотреть тогда, когда заговорим о

программировании и программаторах, и о программе, работающей с программатором. Как программаторов, так и программ работающих с программаторами много. Если вы не планируете профессионально заниматься микроконтроллерами, «прошивая» по сотне микросхем в день, то совсем не обязательно выбирать программатор, работающий с LPT или USB портами компьютера, вполне подойдет и простенький программатор, присоединяемый к COM-порту. То же можно сказать и о выборе контроллера — если у вас нет сложной задачи, для которой запланировано время на выбор контроллера, используйте тот, что подешевле, что можно купить в ближайшем магазине или получить по почте. Позже, когда вы наберетесь опыта, вам не сложно будет сменить PIC на AVR или любой другой микроконтроллер. Среды разработки, такие как MPLAB или AVRStudio, могут работать напрямую с рядом программаторов. Но это потребует от вас больше работы по поиску схемы и сборке устройства. Порой проще выполнить всю работу по написанию кода и его отладке в одной программе, а выполнить программирование микросхемы в другой. Но это, как говорят, дело вкуса.

Рассмотрим самый простой случай, когда все выводы микроконтроллера используются «на выход». Для конкретизации используем PIC16F628A. А для реализации простых задач программу FlowCode. Такой «самый простой» случай позволит использовать контроллер в качестве управляющего устройства, скажем для переключения елочных гирлянд. Или для генерации меандра. Или как индикатор включения. Или...

Запускаем программу FlowCode.

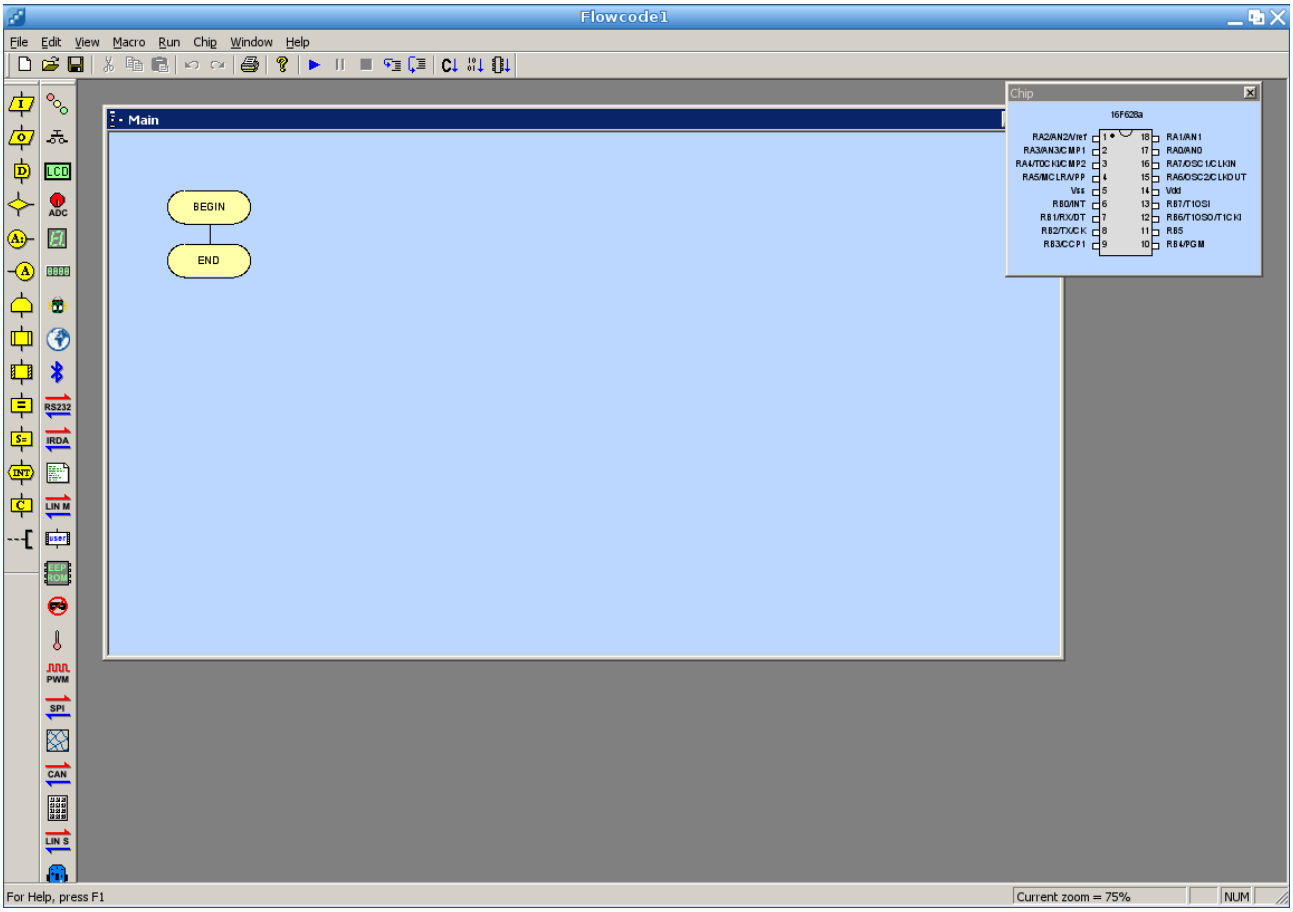


Рис. 1.1. Первый запуск программы FlowCode

Программа предназначена для операционной среды Windows, но я использую ее в Linux, поэтому могут иметь место незначительные отличия во внешнем виде и поведении программы. Так у меня при запуске программы основное рабочее поле необходимо открыть, используя стандартную кнопку (**Распахнуть)** в правом верхнем углу. Если вы уже работали с программой, то при запуске в окне диалога открытия файла можно выбрать, предстоит ли работа с новым файлом (*Create a new*

FlowCode flowchart...), или будет продолжена работа со старым (*Open an existing FlowCode flowchart...*), который можно выбрать из предложенного ниже списка.

Я не борец за «крутость», по причине чего хочу заставить микроконтроллер управлять светодиодом, который на макетной плате припаяю к выводу, скажем, нулевому порта А. Пусть мигает раз в секунду.

Я подозреваю, что в программе это можно сделать несколькими способами, но использую самый очевидный — на правой инструментальной панели есть кнопочка с буквой «О». Думаю, это от слова *output*-выход (если навести курсор мышки на эту кнопочку, то высвечивается подсказка *Output*). Цепляю этот выход (нажимаю левую клавишу мышки, когда курсор указывает на иконку, и, не отпуская клавиши, перемещаю курсор в рабочее поле схемы) и тащу его к линии между овалами Begin-начало и End-конец. При этом курсор выглядит как стилизованная иконка, а слева появляется стрелочка-указатель.

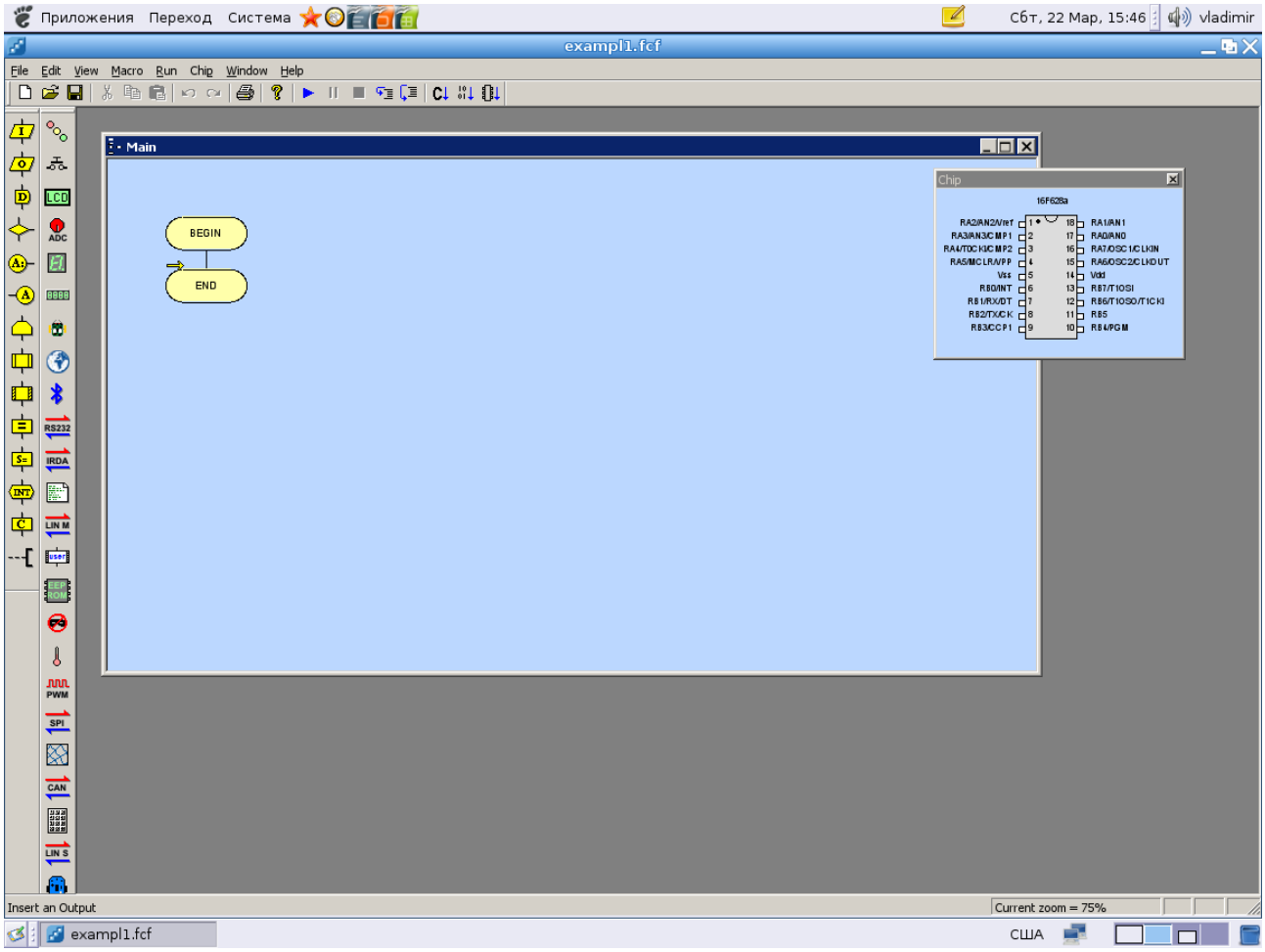


Рис. 1.2. Добавление элемента программы к диаграмме

После добавления элемента Output диаграмма принимает следующий вид:

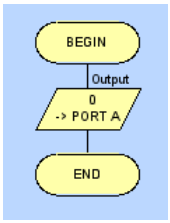


Рис. 1.3. Начальный вид диаграммы первой программы

Нуль в заголовке состояния порта А подразумевает, что все выводы находятся в состоянии логического нуля (с низким уровнем напряжения) или что они будут установлены в 0.

Мигать чем-то — это менять состояние, но чтобы мигание было заметно, понадобится пауза. Такой элемент (иконка на левой инструментальной панели с литерой «D») *Delay*-задержка есть. Перетаскиваем его и вставляем ниже первого выхода *Output*. Но, если состояние выводов порта А меня устраивало, то время паузы 1 мС не то, что мне хотелось бы. Двойной щелчок левой клавишей мышки по этому элементу на рабочем поле открывает диалоговое окно свойств элемента.

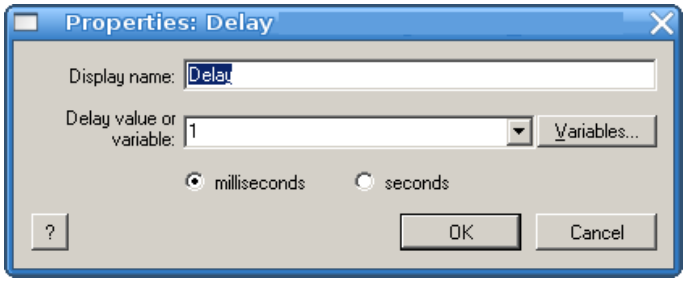


Рис. 1.4. Диалоговое окно свойств элемента *Delay*

Теперь достаточно выбрать опцию *seconds*, чтобы превратить миллисекунду в секунду. Следом за задержкой в 1 секунду я добавляю еще один *Output*, как и в первый раз, но теперь, двойным щелчком левой клавиши мышки по нему на рабочем поле схемы, открываю диалоговое окно его свойств:

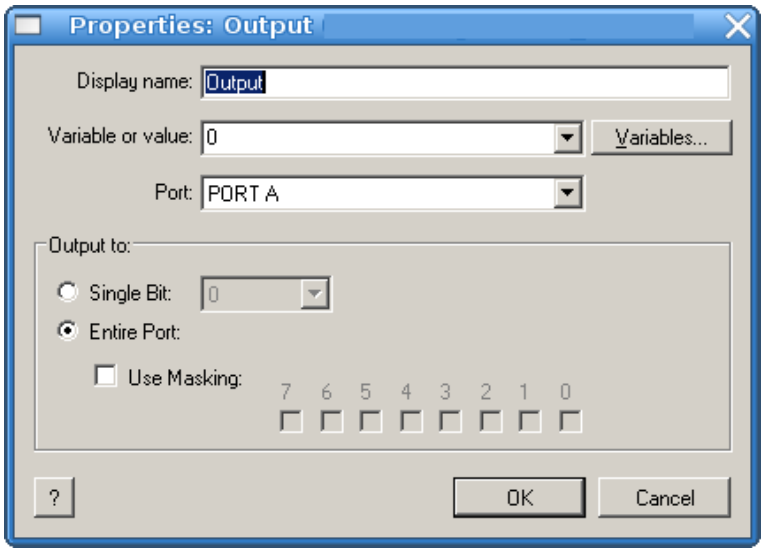


Рис. 1.5. Изменение свойств выхода порта А в диалоговом окне

Заменив 0 в окне *Variable or Value:* единицей, добавив еще одну паузу, я почти достиг желаемого, если не... учитывать, что я хотел бы, чтобы светодиод мигал непрерывно. Такое непрерывное выполнение фрагмента программы, если не ошибаюсь в программировании называется циклом-*Loop*. И такой элемент на левой инструментальной панели есть, седьмая кнопка сверху. Добавление его к концу столь «долго» выстраиваемой программы, конечно, не приводит к цели. Но, нажав левую клавишу мышки, когда курсор находится на пустом месте над моей программой, я, удерживая клавишу, отрисовываю прямоугольник, включающий всю мою программу, кроме цикла. Все, что теперь выделено, можно перетащить к линии, соединяющей начало цикла *While* и конец, отмеченный как *Loop*. В программировании часто используются циклы, и бывают они разного вида, например, выполняемые заданное количество раз (**For...**) или условные, выполняемые до тех пор, пока не будет (или будет) выполнено некое условие, которое может, в свой черед, проверяться до

выполнения очередного прохода программы, заключенной в цикл, или после прохода и т.д., но это уже имеет отношение к кодированию программы, к языку программирования, а не к нашему первому опыту.

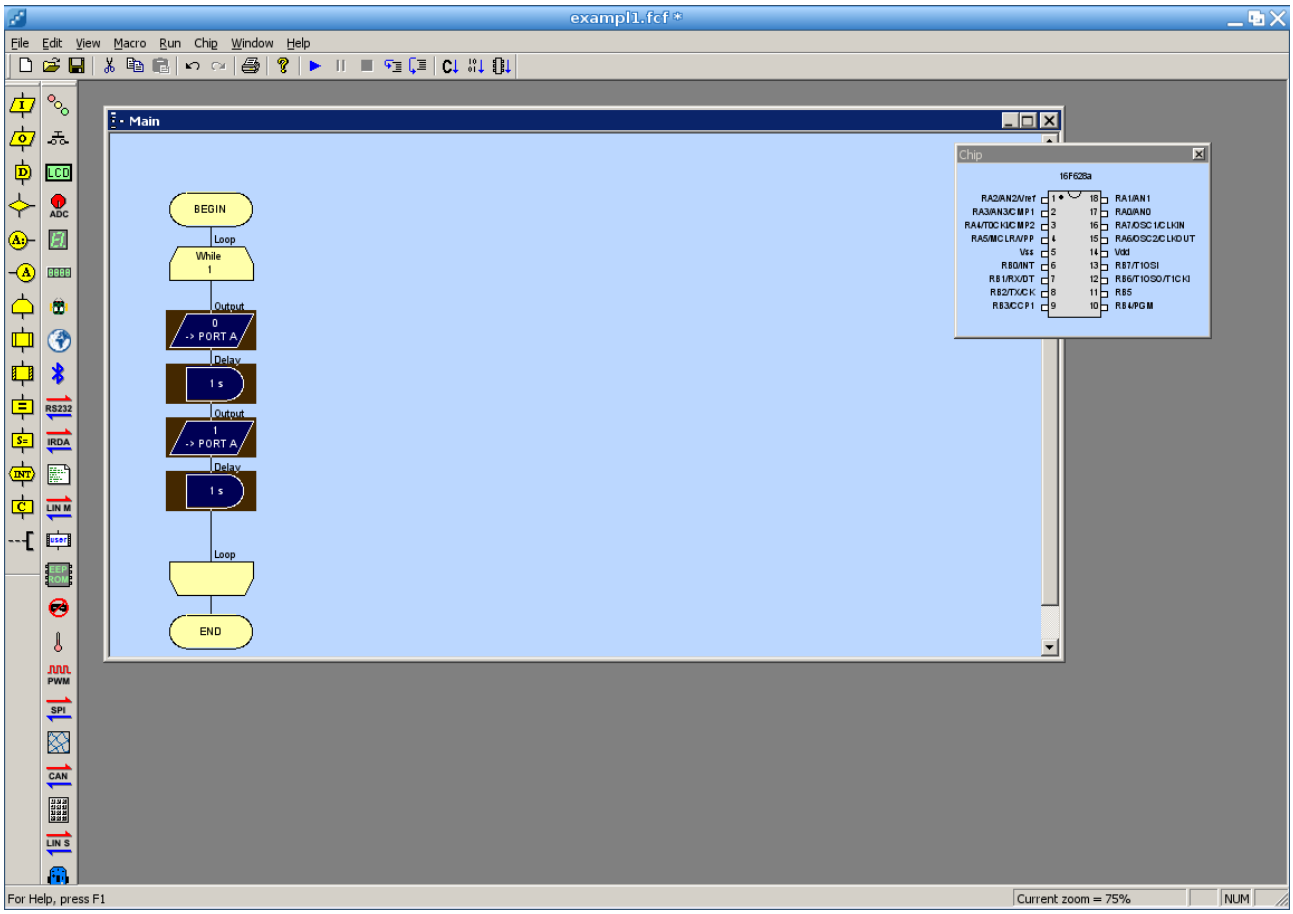


Рис. 1.6. Первая программа в FlowCode

А мы написали первую программу, и пора бы проверить, работает ли она. Программа FlowCode имеет отладочные средства. И я глубоко убежден, что их должно хватать для создания достаточного количества устройств без необходимости покупать и программировать контроллеры.

Чтобы запустить отладку, достаточно в основном меню выбрать пункт *Run* и раздел *Go/Continue* или на основном инструментальном меню нажать кнопочку с иконкой, как у любого плеера обозначающей воспроизведение. Однако прежде, чем это сделать, полезно (или весьма полезно) на левой инструментальной панели (второй) нажать первую кнопочку с изображением ряда индикаторов (светодиодов), появляющаяся подсказка к ним *LEDs*. Вот теперь можно и запустить отладку. Мигающий светодиод, обозначенный как A0, в точности повторит то, что вы увидите, собрав макетную плату.

Для первой программы полезно будет попробовать менять состояние порта А во втором *Output*, вписывая разные числа. Они все будут отображаться состояниями выводов порта А в виде, который можно называть кодом 1-2-4-8, или в двоичном виде.

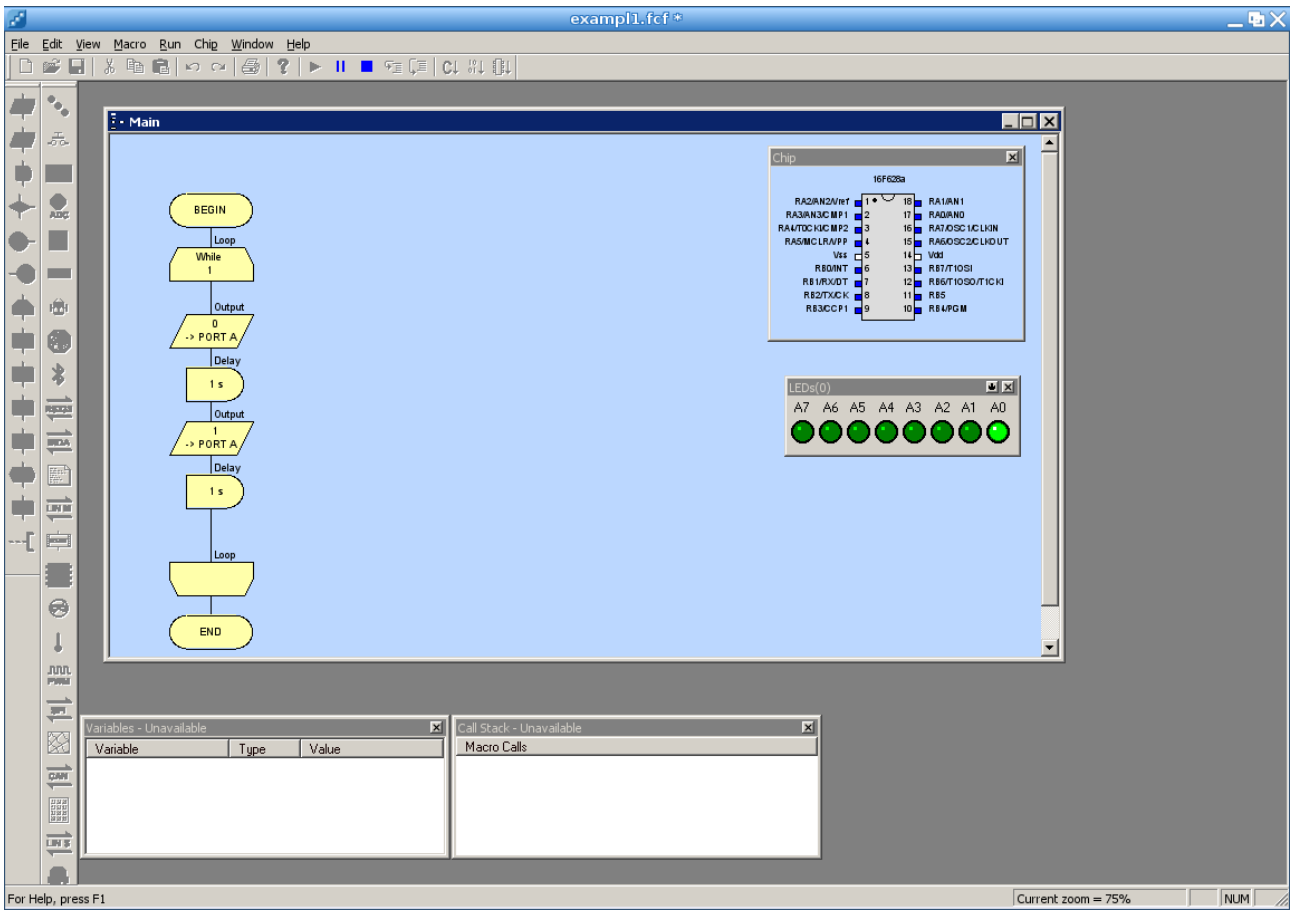


Рис. 1.7. Первый запуск первой программы

Как вы могли убедиться, написать небольшую программу для микроконтроллера гораздо проще, чем собрать схему на аналоговых ли, цифровых ли элементах, которая заставила бы светодиод мигать. А ведь эту программу легко поправить, чтобы она выполняла более сложные операции.

Полет второй

Когда пишешь прикладную компьютерную программу, проще — оттранслировал и можно запустить, посмотреть, что из этого получается, даже если ничего не получается.

С микропроцессорами и микроконтроллерами хуже. Я помню, как много лет назад, при написании программы испытывал нестерпимый зуд в руках, в кончиках пальцев, хотелось как можно скорее схватить паяльник и что-нибудь к чему-нибудь обязательно припаять. Уж так устроен «нормальный железячник».

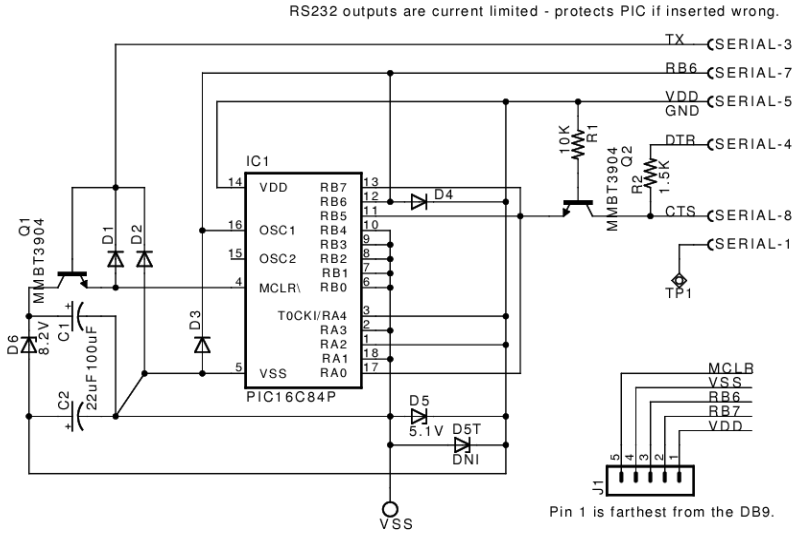
Поэтому прежде, чем вернуться к программе FlowCode, поговорим немного о железе.

Здесь у меня есть определенные трудности. Несколько лет назад, когда мне понадобилось в среде разработки PIC-контроллеров MPLAB что-то сделать, я не отважился ни на покупку программатора, который работал бы с этой программой, ни к пайке такого программатора. Из программ, работающих с простыми программаторами, я обнаружил только PonyProg для Windows, и, спаяв программатор, работал с этой парой. Позже, рассказывая о программах Piklab и KTechlab для Linux, я спаял еще один простой программатор, который прекрасно работал с ними. Оба программатора я оформил в самом простом виде, поставив панельку только для микроконтроллера PIC16F628A. А в промежутке между пайкой программаторов, чтобы не обидеть Windows, купил недорогой, но более универсальный программатор EXTRA-PIC.

Я не готов к тому, чтобы «хаять» программу PonyProg и программатор, работающий с ней. Я не готов к тому, чтобы утверждать, что это и есть то, что нужно, и больше ничего не надо. И, увы, не

готов к тому, чтобы найти и спаять схему программатора, который, возможно, будет работать с программой FlowCode. Но в любом случае, работа программ для программаторов в паре с программатором отличается друг от друга не столь разительно, хотя может иметь много особенностей, чтобы считать это принципиальным моментом. С другой стороны, было бы непростительно не проверить «живьем» результаты, полученные в программе FlowCode. Я предлагаю компромисс. Я опишу процесс программирование в Piklab, сделав вид, что это некоторая специально для программатора существующая программа, а вы или сделаете вид, что верите мне, или подумаете, а не использовать ли, как это делаю я, Linux в качестве второй операционной системы. Под Linux есть достаточно удобная среда работы с PIC-контроллерами Piklab, есть бесплатные и полнофункциональные компиляторы Си, а об ассемблере я и не говорю. Или, если понадобится, посмотрите описание программатора EXTRA-PIC. Программа для него тоже распространяется бесплатно.

Итак. Схема программатора с которым я буду работать в программе Piklab для Linux. Кстати, это классическая схема, с которой, мне кажется, будут работать Windows-программы. Нужно их только



поискать.

Рис. 2.1. Схема программатора для программы Piklab

Программатор у меня подключается шлейфовым кабелем в метр длиной, чтобы удобно было работать возле компьютера, к COM-порту. Выглядит он так:

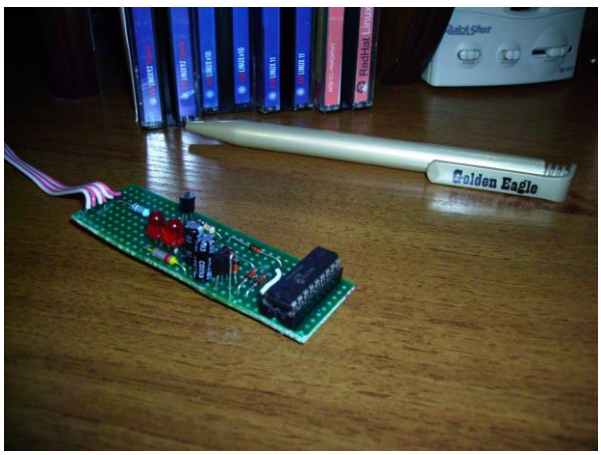


Рис. 2.2. Внешний вид программатора

Внешний вид и описание программатора EXTRA-PIC можно найти в Интернете, там же есть схема. Можно, думаю, заказать его по почте в агентстве «Десси». Словом, это, мне кажется, не столь принципиально, и больше зависит от ваших вкусов и привычек, чем от чего-либо еще.

Сделав вид, что программатор я только что спаял, я хочу программу, написанную в FlowCode, загрузить в микросхему; микросхему после этого перенести на макетную плату, где у меня кроме панельки для микросхемы есть несколько светодиодов, оставшихся от предыдущих экспериментов, и еще что-то, что не имеет отношения к данному случаю. Если светодиод будет мигать, я помещу фотографию макетной платы, если не будет мигать, я напишу пару абзацев о том, как здорово он мигает. Вот такие планы на этот «ночной» полет.

Первое, что нам потребуется, это программу из предыдущей части рассказа превратить в файл с hex-кодом. Такой файл загружается в программу для программатора, которая и отправит работающий код в микроконтроллер.

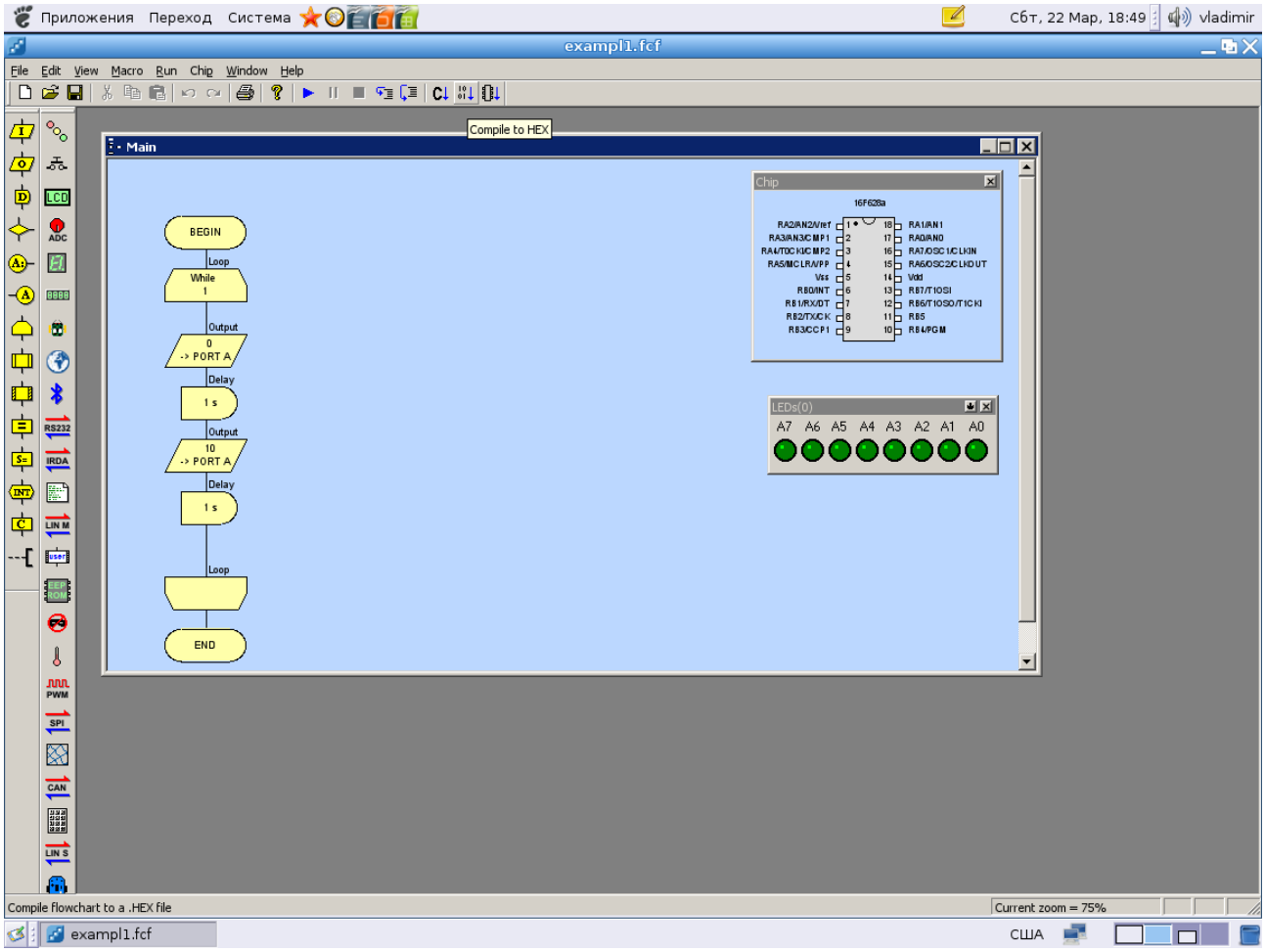


Рис. 2.3. Трансляция программы в hex-код

Предпоследняя клавиша основного инструментального меню с подсказкой *Compile to HEX*, как видно из рисунка, должна выполнить это... или, положим, должна была бы... или могла бы... но в Linux эта процедура явно не проходит.

В Windows все заканчивается благополучно, а в папке с программой появляется искомый hex-файл. Если бы в моем распоряжении был программатор, подключенный по USB интерфейсу к компьютеру, программатор, умеющий разговаривать с FlowCode, то следующая кнопка основного инструментального меню с иконкой микросхемы загрузила бы программу.

Я использую Piklab. Не то, чтобы это говорило о моей пристрастности к Linux, я просто работаю в Linux и несколько лет кряду пытаюсь понять, что имеют в виду люди, когда говорят о трудностях

освоения Linux. Работа в Linux ничем не отличается от работы в Windows, разве только удобнее. Этот текст я пишу в OpenOffice Writer'e, Piklab работает на другом рабочем столе графического менеджера Gnome, а на третьем я при необходимости поправляю иллюстрации в графическом редакторе Gimp. В Windows мне не хватает этих удобств.

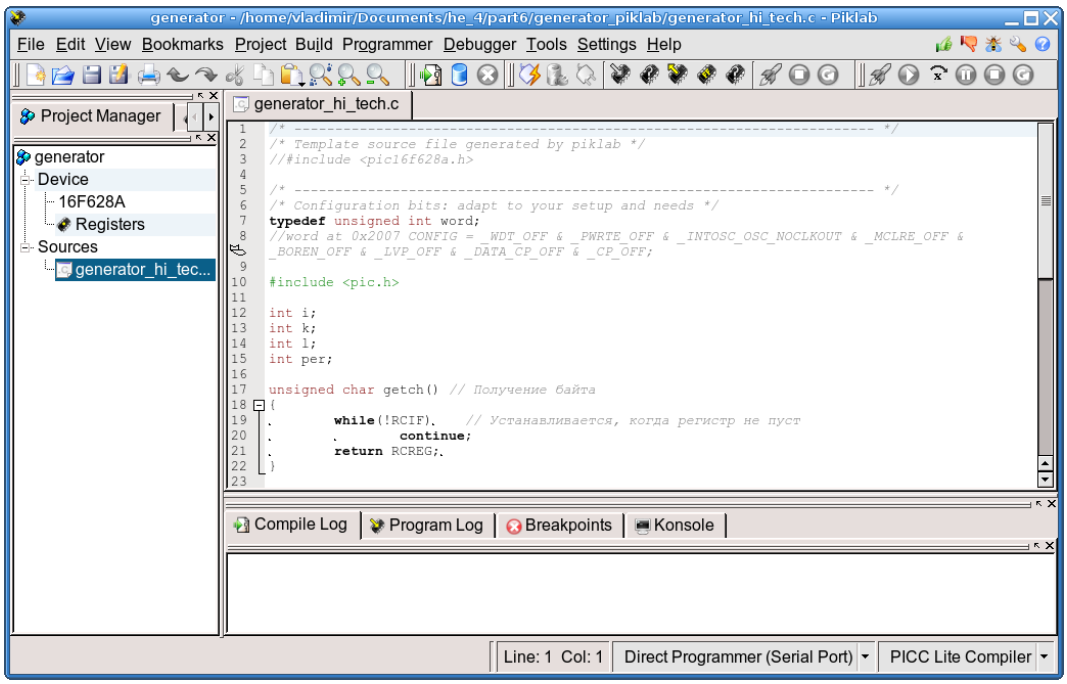


Рис. 2.4. Среда программирования PIC-контроллеров Piklab

После загрузки я закрываю предыдущий проект и открываю hex-файл, созданный в программе FlowCode — в основном меню File-Open.

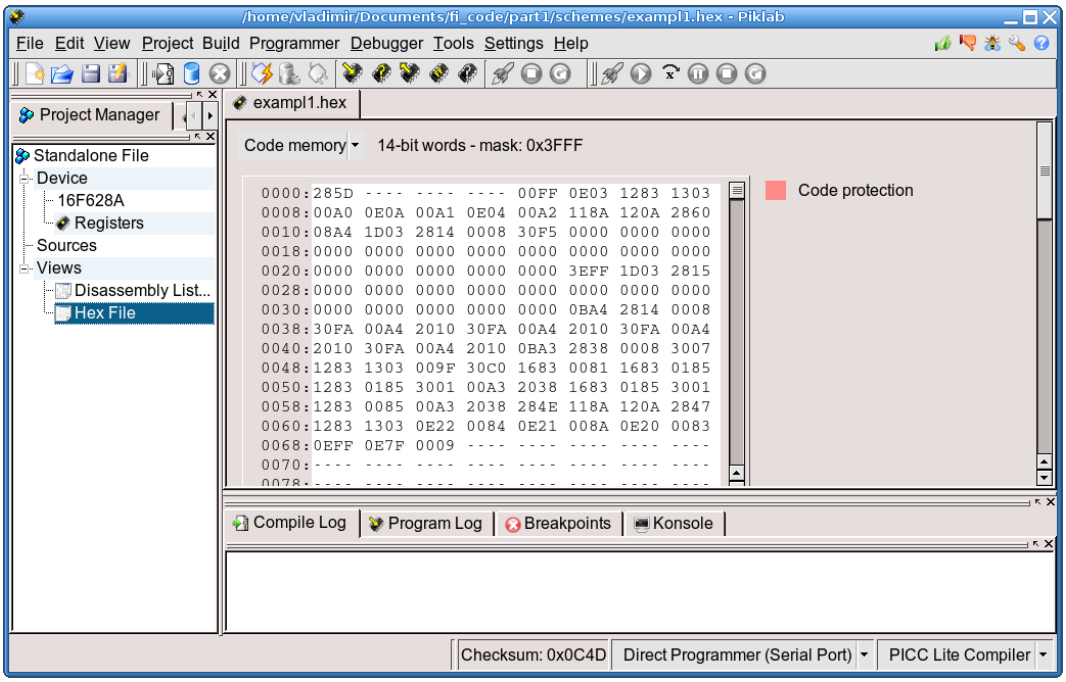


Рис. 2.5. Hex-файл, открытый в Piklab

Файл можно было бы загрузить в микросхему, но перед этим нужно записать слово конфигурации. В разных программах, работающих с программаторами, эта процедура может выглядеть различно, но

в этой достаточно спуститься чуть ниже в окне кода, об этом же говорят полосы прокрутки. В окне с адресом 2007 перед отправкой кода в микроконтроллер я запишу 2118 (в действительности 3F18h), что означает отказ от блокировок, использование внутреннего генератора и еще ряд особенностей, которые меня пока не интересуют.

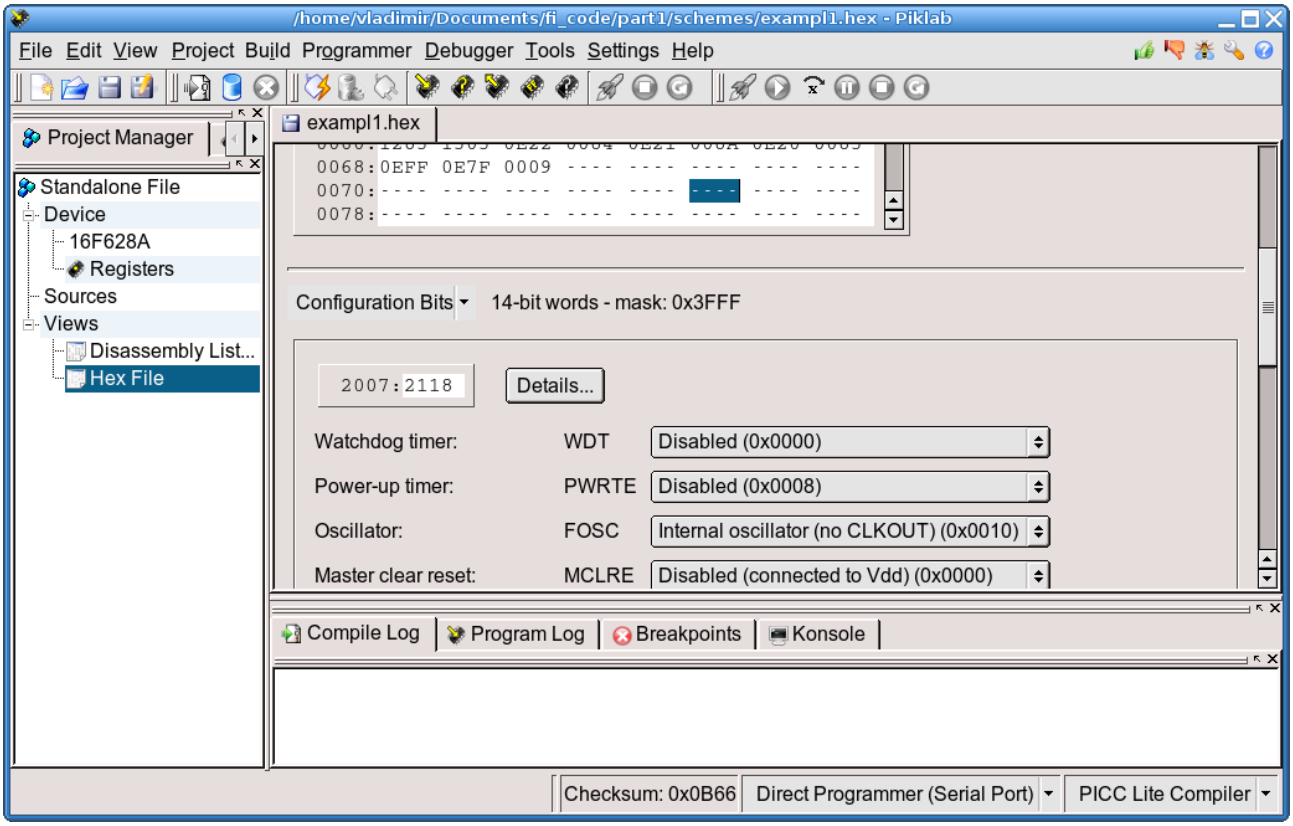


Рис. 2.6. Запись слова конфигурации в Piklab

Теперь все готово к записи, осталось нажать на инструментальной панели кнопку с иконкой микросхемы и стрелкой, направленной внутрь. Через несколько секунд в окне сообщений появляется сообщение о завершении процесса записи.

На этом можно закончить повествование, но я обещал проверить, все ли работает. Поэтому после программирования микроконтроллера я переношу микросхему на старенькую макетную плату, оставшуюся от предыдущих опытов, включаю блок питания и наблюдаю, как зажигается и гаснет светодиод. Делает он это не совсем так, как я заказывал, не раз в секунду, а медленнее. Я даже могу сказать, что раз в 5 медленнее. Происходит это потому, что, я думаю, программа FlowCode рассчитывает на работу с кварцем 20 МГц, а внутренний генератор, который я использую, работает на частоте 4 МГц. Если бы мне нужны были именно секундные импульсы, я за 10-15 секунд подправил бы времена задержки, и переписал бы микроконтроллер еще раз.

Если бы я хотел этот контроллер использовать для работы с елочной гирляндой, я добавил бы вместо светодиода транзистор (на всякий случай) с реле, имеющим контакты на 220 В, и подключил гирлянду через эти контакты.

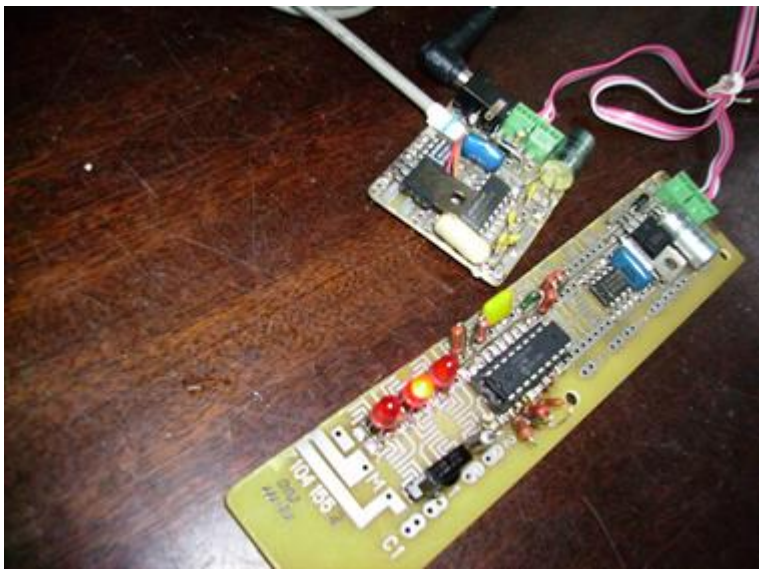


Рис. 2.7. Работающий микроконтроллер на макетной плате

Программа FlowCode может генерировать файл на языке Си.

```
//********************************************************************************
//**
//**  File name: C:\Documents and Settings\vladimir\Рабочий
стол\fi_code\part1\schemes\exampl1.c
//**  Generated by:  Flowcode v3.2.2.40
//**  Date:          Saturday, March 22, 2008 20:37:58
//**  Licence:       Demo
//**
//**        ***DEMO VERSION***
//**
//**
//**     NOT FOR COMMERCIAL USE
//**
//**  http://www.matrixmultimedia.com
//********************************************************************************

#define MX_PIC

//Defines for microcontroller
#define P16F628A
#define MX_EE
#define MX_EE_TYPE1
#define MX_EE_SIZE 128
#define MX_UART
#define MX_UART_B
#define MX_UART_TX 2
#define MX_UART_RX 1
#define MX_PWM
#define MX_PWM_CNT 1
#define MX_PWM_TRIS1 trisb
#define MX_PWM_1 3

//Functions
#include <system.h>
#pragma CLOCK_FREQ 19660800

//Configuration data
//Internal functions
#include "C:\Program Files\Matrix Multimedia\Flowcode V3\FCD\internals.h"
```

```
//Macro function declarations
//Variable declarations
//Supplementary defines
//Macro implementations
//Supplementary implementations

void main()
{
	//Initialisation
	cmcon = 0x07;

	//Interrupt initialisation code
	option_reg = 0xC0;

	//Loop
	//Loop: While 1
	while( 1 )
	{
		//Output
		//Output: 0 -> PORT A
		trisa = 0x00;
		porta = 0;

		//Delay
		//Delay: 1 s
		delay_s(1);

		//Output
		//Output: 1 -> PORT A
		trisa = 0x00;
		porta = 1;

		//Delay
		//Delay: 1 s
		delay_s(1);
	}
	mainendloop: goto mainendloop;
}

void interrupt(void)
{
}
```

Это удобно, если предполагать работу с языком Си. Хотя запись на языке Си может потребовать правки, зависит от используемого компилятора, она достаточно универсальна. Ту же программу можно использовать для других микроконтроллеров.

Но это тема другого полета.

Полет третий

Программа FlowCode позволяет быстро создавать программы достаточно интересные.

На инструментальной панели, где в прошлый раз обнаружились светодиоды, чуть ниже есть кнопка с иконкой семисегментного индикатора. Если ее нажать, то на рабочем поле появится этот самый индикатор.

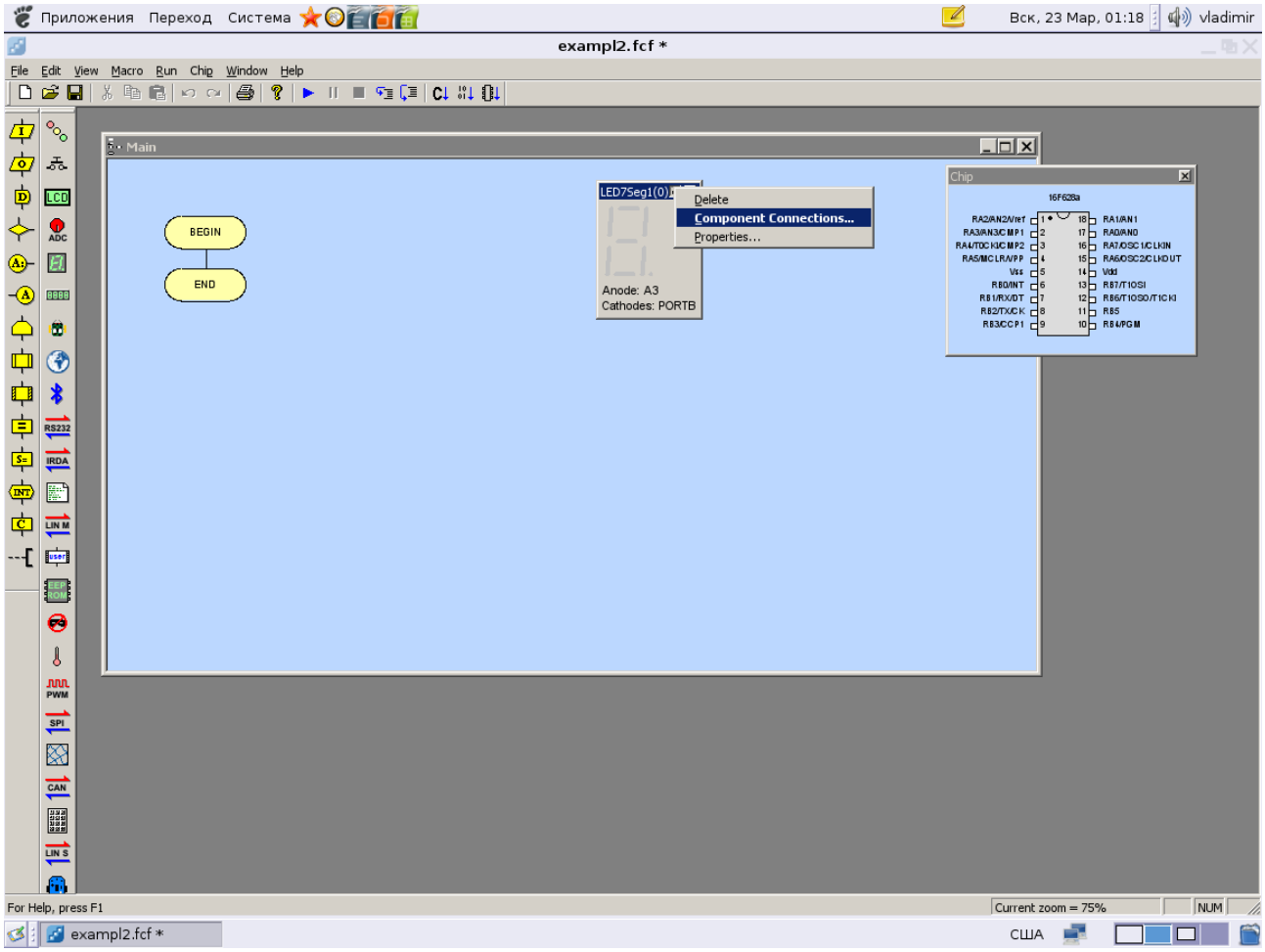


Рис. 3.1. Начало работы с программой обслуживания семисегментного индикатора

Как мне помнится, есть светодиодные индикаторы с общим анодом и общим катодом. Если так, то при практической реализации легко можно сменить состояние общего вывода и инвертировать состояние выводов управления, если не так, то и менять ничего не придется.

Выходы микроконтроллера PIC16F628A рассчитаны на достаточно большие токи, но, если нет особой нужды в минимизации количества деталей схемы, можно в реальной схеме добавить токоограничительные резисторы. Будут или нет добавлены «резисторы безопасности», это никак не отразится на работе программы, поэтому при написании программы можно отложить решение этого вопроса до момента реализации схемы на макете.

Если на панельке индикатора нажать на кнопочку со стрелкой в правом верхнем углу, то появится выпадающее меню. Выбор раздела *Component Connections...* приводит нас в диалог настройки подключения индикатора.

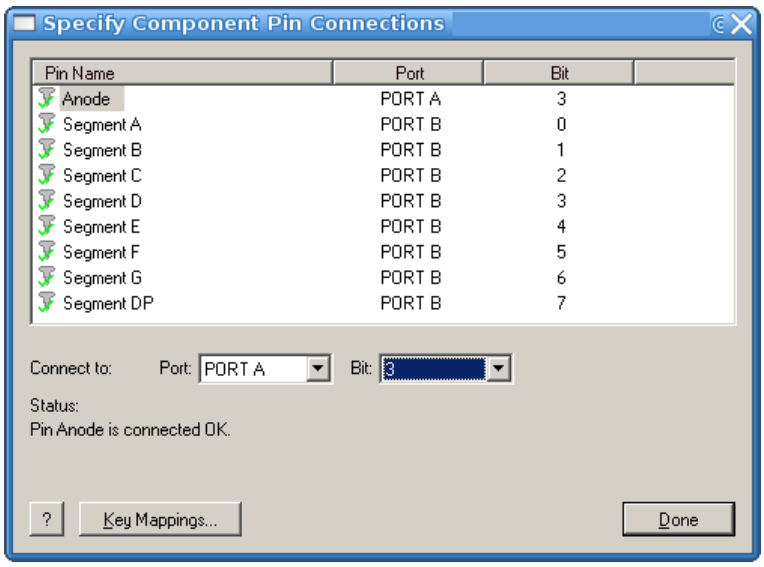


Рис. 3.2. Диалог настройки подключения индикатора

При первом запуске вывод *Anode* оказывается не подключен (*Unconnected*). Для его подключения достаточно выбрать в окне *Port:*, скажем, порт A и Bit 3. Сегменты индикатора можно оставить подключенными так, как это сделано по умолчанию.

Теперь, подключив анод к выходу 3 порта A, мы можем установить вывод в «1», чтобы засветить все сегменты. Для этого используем элемент *Output*, как делали это прежде, и изменим свойства этого элемента.

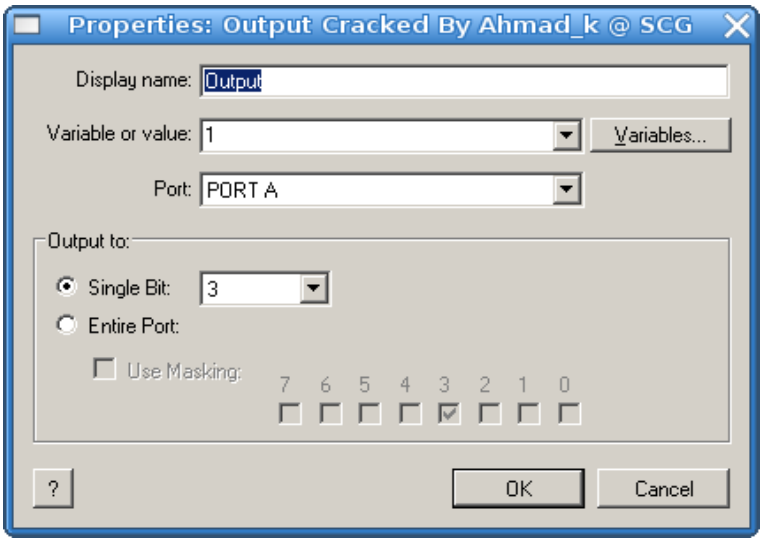


Рис. 3.3. Диалог настройки элемента Output

В этот раз я хочу использовать только бит 3, который установлю в «1». Если такое подключение семисегментного индикатора засветит все сегменты, то мы должны получить цифру 8.

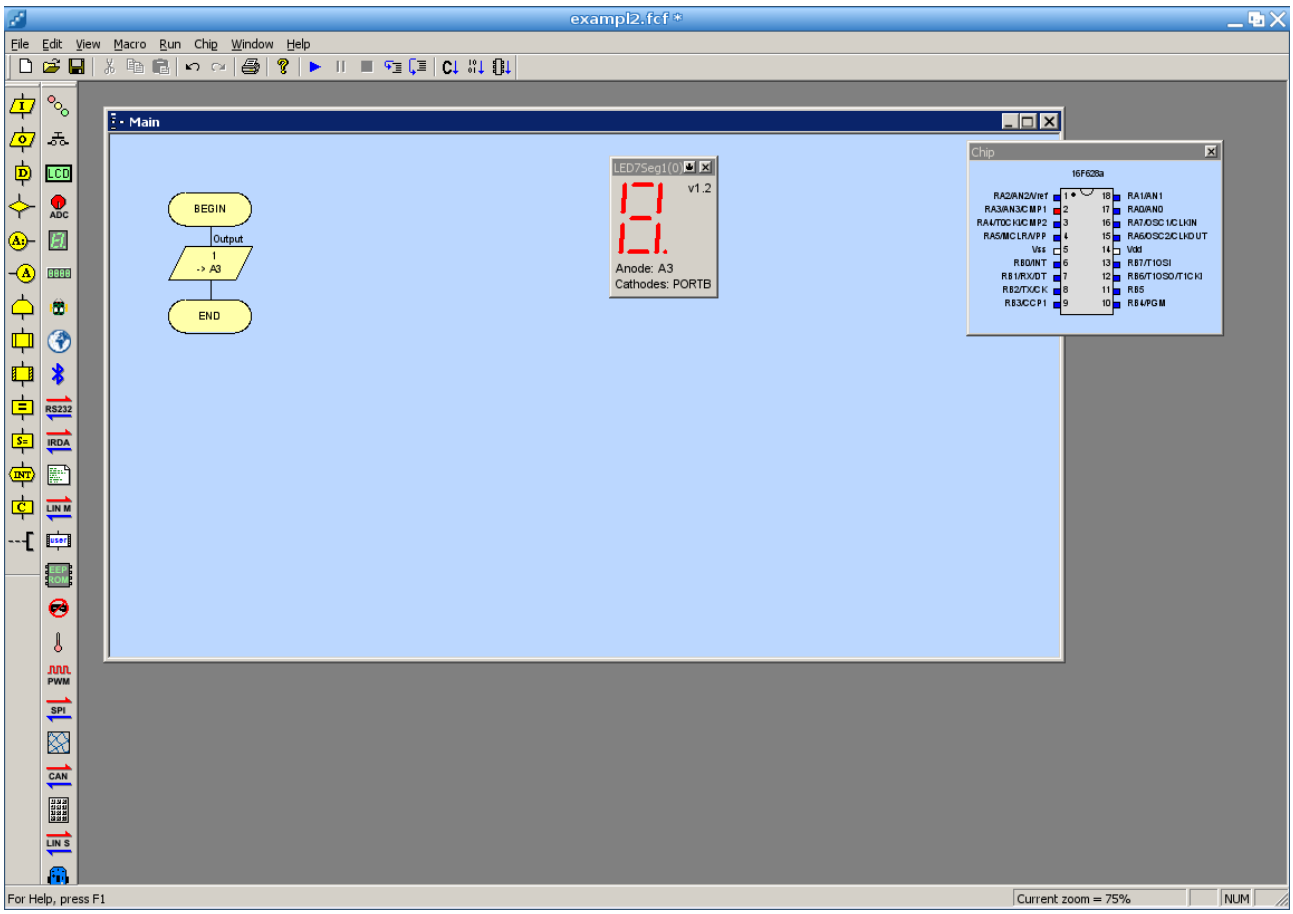


Рис. 3.4. Запуск симуляции программы

Теперь постараемся превратить цифру 8 в цифру 0 (самая простая трансформация). За «перекладинку» отвечает сегмент G, подключенный к биту 6 порта B, который установим в «1».

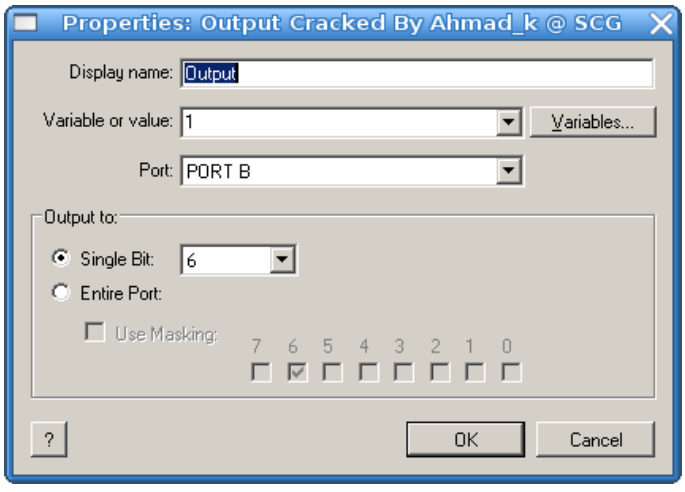


Рис. 3.5. Изменение состояния вывода порта B.6

Запустив симуляцию можно проверить, что 8 превращается в 0, как мы и ожидали. Таким образом, манипулируя состоянием выводов порта B, мы можем высвечивать любую цифру на семисегментном индикаторе.

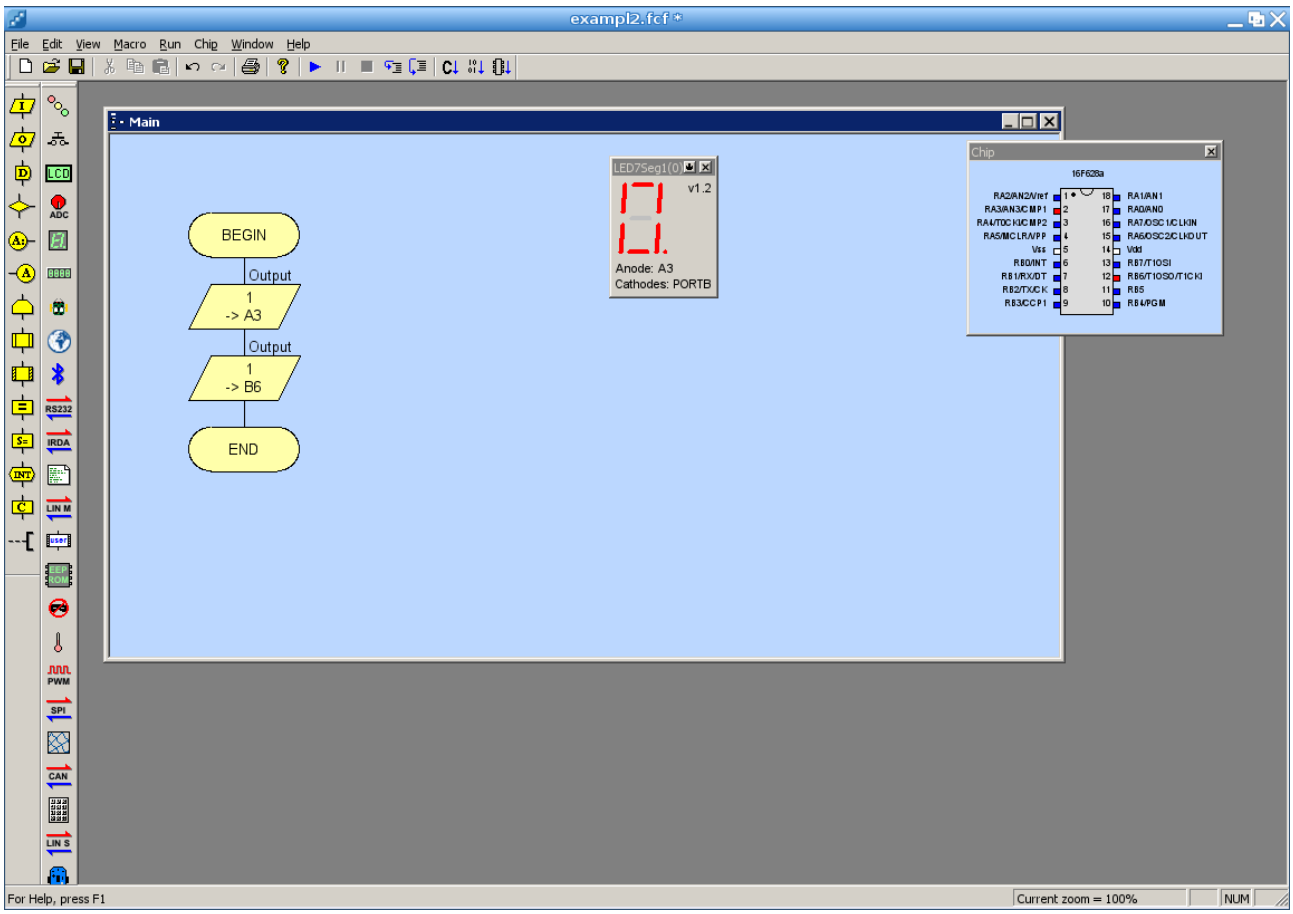


Рис. 3.6. Превращение 8 в 0 на индикаторе

Программа FlowCode, если заглянуть в папку, где она установлена, имеет много обучающих примеров. Если при первом знакомстве, например, с программированием микроконтроллера можно не заботиться о виде программы, довольствуясь полученными «по умолчанию» названиями, то в дальнейшем это может мешать. При взгляде на предыдущий рисунок трудно понять назначение первого элемента. Но это легко исправить. Двойной щелчок по нему, в окне *Display name:* пишем то, что нам понятно. Лучше, если это будут названия на английском, но можно на латинице сделать для себя понятное название элемента.

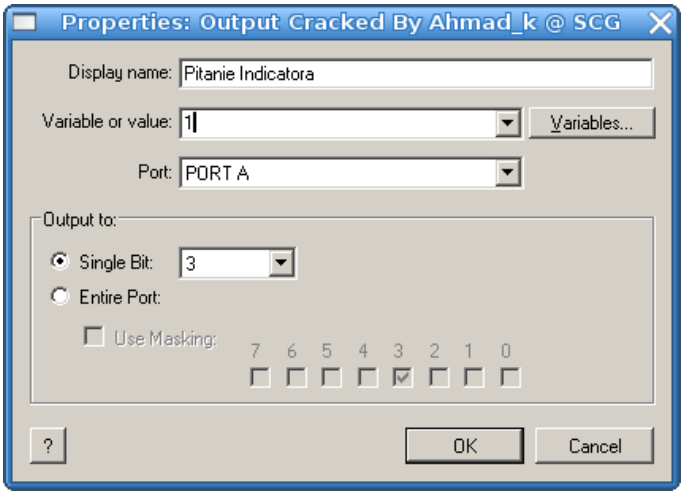


Рис. 3.7. Смена названия элементов программы

Несколько примеров в папке «Examples» показывают, как удобно для себя использовать возможности программы. Профессиональные программисты используют разные приемы оформления программы, но главное — обеспечить понятность для себя и других. Даже когда пишешь

программу для себя, а программы имеют свойство повторного использования готовых модулей, важно сделать программу понятной, используя ясные названия переменных, комментарии. Увлекаясь работой, жалеешь время на обдумывание этих, в сущности нейтральных по отношению к коду программы атрибутов, но возвращение к уже написанной и отлаженной программе показывает, что это время было бы потрачено не зря. Переменные вида a1, a2, a3 и т.д. экономят время при написании программы, но их назначение трудно понять по прошествии месяца после завершения работы. Но это дело вкуса и личных предпочтений, а возможностям понятного описания программы посвящено несколько примеров в папке «Examples».

На одном из сайтов, кажется «Паяльник», я видел просьбу помочь с устройством, на первый взгляд простым — несколько кнопок, при нажимании которых счетчик должен отображать количество нажатий. Все просто. Счетчик, кнопки, индикатор. Небольшая проблема — первое нажатие любой кнопки должно обнулять счетчик. И еще одна небольшая проблема — дребезг контактов кнопок. Две небольшие проблемы (а может и не две) существенно усложняют схемное решение. И совет, данный автору темы, использовать микроконтроллер, совершенно, как мне кажется, правилен.

Подключение индикатора к контроллеру ясно из предыдущего. Как подключаются кнопки, можно увидеть в примерах.

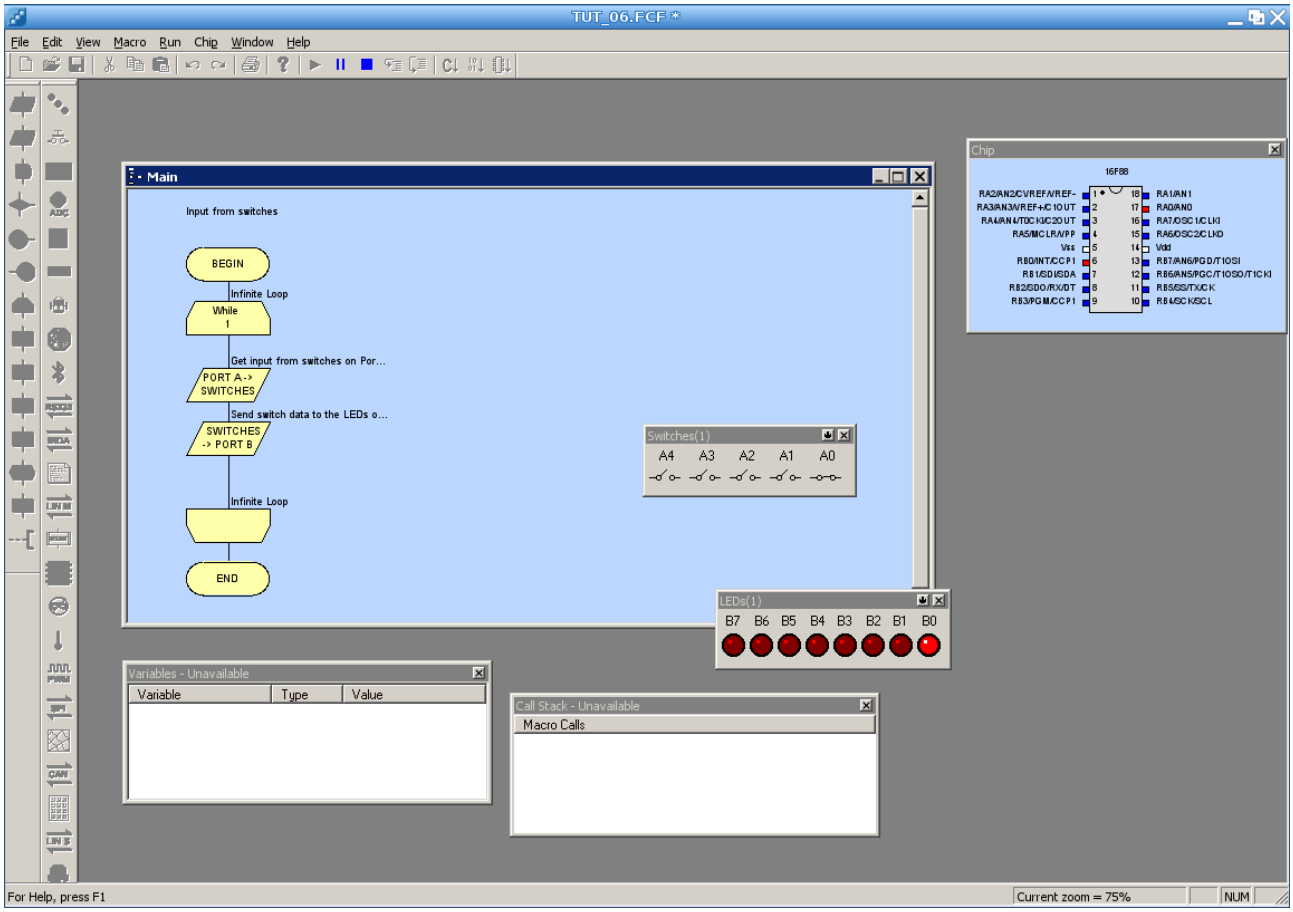


Рис. 3.8. Пример использования кнопок в программе FlowCode

Пример взят из программ, приходящих с FlowCode. А предыдущий пример показывает, как использовать элемент вычислений *Calculation*.

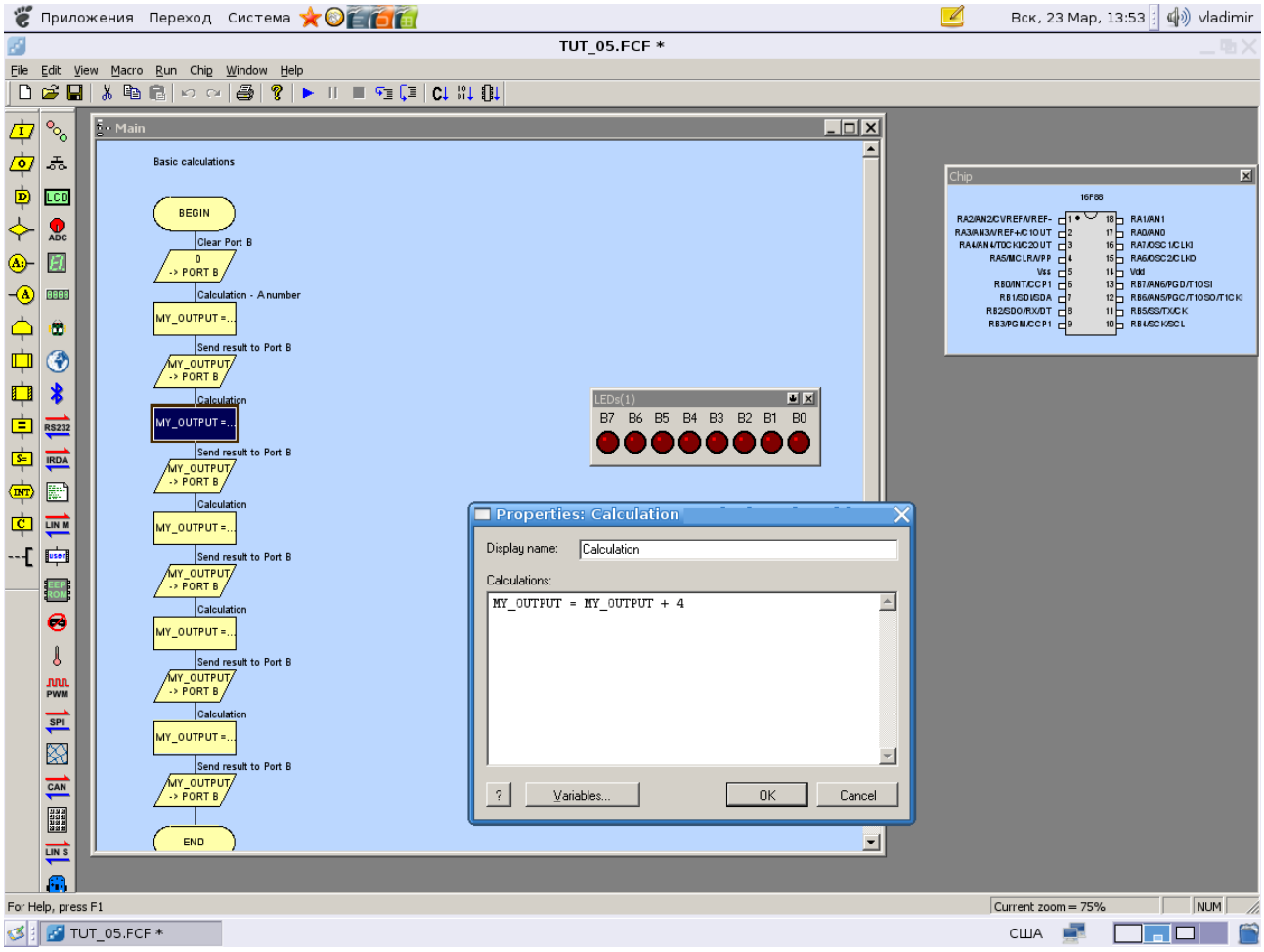


Рис. 3.9. Пример использования вычислений

Здесь переменная MY_OUTPUT увеличивается на 4. Собственно, есть все подсказки для того, чтобы начать работу над проектом: несколько кнопок, подсчет нажатий каждой кнопки в переменной, отображение результата на индикаторе. Если добавить условие такого рода, а элемент *Decistion*-условие в программе есть, чтобы при нажатии кнопки проверялось, нажата она впервые или нет, если впервые, то обнулять переменную подсчета отображаемую индикатором, если нет, то продолжать подсчет, если добавить это, то, практически, почти все требования к схеме будут соблюдены. Для программного устранения дребезга контактов можно вставлять паузы *Delay* длительностью 0.1-0.2 секунды в то место программы, где проверяется состояние контактов.

Реализацию первого проекта такого рода можно разбить на несколько этапов.

На первом этапе, например, можно использовать переменную, значение которой должно отобразиться на семисегментном индикаторе. Переменная должна меняться от 0 до 9. И на первом этапе это можно делать «ручками»: остановили отладку, изменили значение переменной, отобразили. Или сделать цикл (с паузами), в котором переменная наращивается от 0 по 1 до 9. Условие остановки цикла — переменная стала равна 10 (или 9).

На втором этапе можно использовать кнопку для увеличения значения переменной.

На следующем этапе можно обработать две кнопки, сразу добавив обнуление счета при первом нажатии любой из них, или сделать это на следующем этапе.

И, наконец, можно проверить работу «антидребезга», сделав паузу, например, длительностью в несколько секунд и нажимая кнопку несколько раз.

Думаю, и в этом случае, как подойти к созданию программы, зависит от личных вкусов и предпочтений. Полезно, если даже вы в первый раз приступили к работе, потратить некоторое время,

чтобы разобраться в собственных вкусах и предпочтениях. Дело в том, что все люди разные. Что одному хорошо и удобно, как бы оно ни было правильно, другому может доставлять больше неудобств, чем комфорта. И если у него есть собственные удобные приемы, или подходы, или даже привычки — если это не сказывается на конечных результатах, то отчего бы ни использовать их?

Если в процессе работы появится необходимость обратиться за помощью, то в первую очередь можно заглянуть все в ту же папку «Examples».

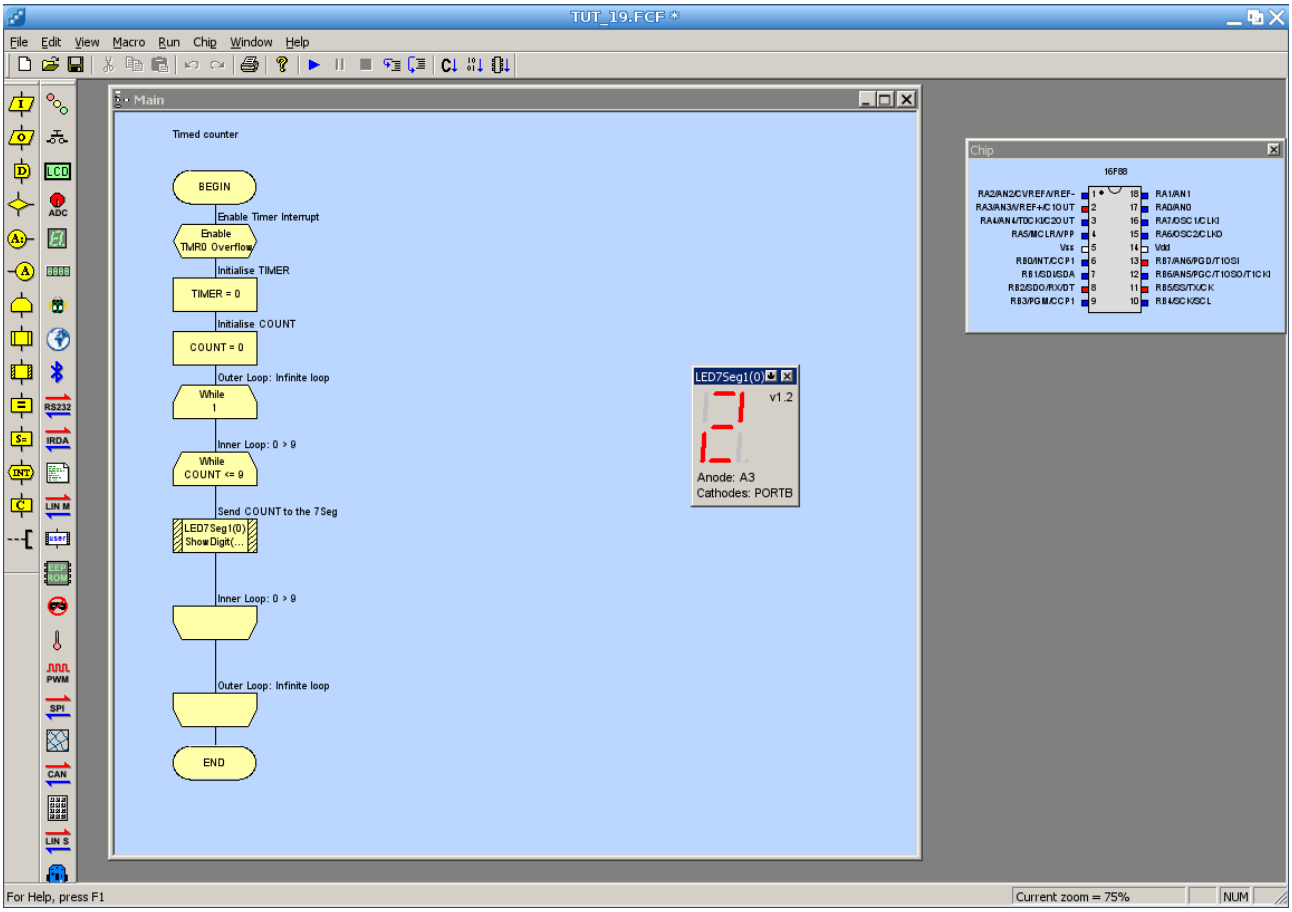


Рис. 3.10. Пример работы с семисегментным индикатором из примеров FlowCode

Программа FlowCode имеет еще ряд очень полезных качеств. Освоив работу с микроконтроллером на уровне алгоритмического построения программы, а многие профессиональные программисты небезосновательно считают, что создание алгоритма, это и есть создание программы, при желании освоить кодирование на языке Си можно видеть, как простые операции, использованные прежде, выглядят на этом языке. Не знаю, насколько удобно использовать FlowCode для работы с языком Си, но, заглянув в руководство, знаю, что компилятор языка может быть использован в программе MPLAB. То есть, все, что вы сделаете, можно перенести в MPLAB и использовать эту среду программирования с компилятором FlowCode без значительных изменений кода программы. Аналогично, мне кажется, можно поступить и ассемблером, если вас заинтересует такой вариант.

Но рассказ об MPLAB я уже написал, и нет смысла это делать вновь.

В настоящее время производитель программы Matrix отказался от поддержки третьей версии Flowcode. Если они, как некоторые из производителей ПО, сделают эту версию бесплатной для студентов и радиолюбителей, то, используя полную версию программы, последние смогут оценить, с чего им удобнее начать.

Стоит ли начать с изучения архитектуры существующих моделей, с программирования на ассемблере; использовать ли им язык программирования высокого уровня, например, Си или начинать программирование в программе с графическим языком, сосредоточив внимание не на том, как создать код программы, а на том, для чего нужен микроконтроллер, как должно работать устройство.

Вот ещё один фрагмент из другой книги автора «Qucs и Flowcode».

Полностью книга доступна для свободной загрузки с авторского сайта http://vgololobov.narod.ru/

Микроконтроллер и FlowCode

Микроконтроллер – это контроллер, регулирующее или управляющее устройство в электронике, в миниатюрном исполнении в виде одной микросхемы. Основу работы микроконтроллера составляет взаимодополнение процессора и программы. Поэтому многие книги о микроконтроллерах начинаются с описания устройства процессора и продолжаются описанием языка ассемблера или языка высокого уровня, чаще языка Си, как наиболее употребительного в профессиональной практике.

По всей видимости, такой подход мешает начинающим осваивать программирование микроконтроллеров в программе FlowCode. Вольно или невольно, они пытаются найти в программе FlowCode те регистры, о которых идет речь в книгах, или те банки памяти, куда они могли бы записать операторы. Отдельный предмет их беспокойства и первых действий в освоении микроконтроллеров – создание программатора и поиск программы для обслуживания этого программатора.

Бесспорно, и знание устройства микроконтроллера, и программатор – все это нужно. Но не следует забывать, что применение микроконтроллеров – это в первую очередь умение создавать программы для них, а овладение искусством программирования, как и любым другим, требует достаточно много времени; в перерывах всегда найдется несколько свободных часов и для пайки программатора, и для чтения описания конкретной модели микроконтроллера. Найдется время и для изучения тех составляющих микроконтроллера, которые называются модулями ШИМ, АЦП или USART.

Программа FlowCode – это среда разработки программ для микроконтроллеров нескольких популярных видов PIC, AVR и ARM. В этом смысле есть программы, которые выглядят одинаково, но работают с выбранными типами микроконтроллеров, хотя есть возможность, например, программу, написанную для PIC-контроллера, импортировать в программу для работы с AVR-контроллерами и наоборот.

Как среда разработки программы, FlowCode в качестве основных компонентов предлагает наиболее употребительные языковые конструкции, которые можно найти, практически, в любом из языков высокого уровня: ветвление программы, цикл и т.п. Именно языковые конструкции и есть то, что следует искать в программе, из чего, кирпич к кирпичу, возводится здание программы. В отличие от других сред программирования, как MPLAB для PIC-контроллеров и AVRStudio для AVR, программа FlowCode в качестве основного языка программирования использует графический язык. И, как в объектно-ориентированном программировании, объекты FlowCode выполняют ряд операций и наделены набором свойств. Графическое программирование – отличительная черта и главное достоинство программы FlowCode, особенно для начинающих.

Демонстрационная версия программы, распространяемая бесплатно, имеет ряд серьезных ограничений, что может создать неверное представление о программе, как только о предназначенной для обучения. Это не так, даже в отношении демо-версии, в рамках которой вполне можно создать устройства весьма полезные.

Но, если говорить об обучении, то достоинство программы FlowCode еще и в том, что в качестве промежуточных результатов она записывает программу на языках Си и ассемблере. Что позволяет получить наиболее часто используемые коды фрагментов программы для этих языков, и наверняка облегчит их освоение.

План этого рассказа: краткое описание программы FlowCode, краткое введение в программирование, пример создания простых программ в FlowCode, получение в программе FlowCode программных фрагментов, относящихся к наиболее употребительным операциям, на языке Си, пример переноса этих фрагментов в другую среду разработки (на языке Си).

Под кратким описанием программы FlowCode подразумевается не то, что следует изучать, но только то, что полезно иметь под рукой, когда работаешь в FlowCode. Многим начинающим не нравится, что первые примеры создания программы для микроконтроллера слишком примитивны: «Подумаешь, помигать светодиодом!». Но не следует забывать, что основное, что делает микроконтроллер – это «мигает» своими выводами, используете вы USART или ШИМ, работаете с индикатором или дисплеем. Да, можно написать программу решения тензорного уравнения, которая будет вызывать почтение, но будет ли она вам нужна? Да и каким образом это поможет вам разобраться и в без того насыщенном новыми элементами мире программирования? Не мудрствуя лукаво, используйте то учебное пособие, что есть в пакете программы FlowCode.

В этом рассказе будут использованы версия FlowCode для PIC-контроллеров и среда разработки MPLAB с компилятором PICC Lite (HI-TECH). Вы можете использовать аналогичную пару для AVR-контроллера.

Найти все необходимое можно на следующих сайтах:

http://www.matrixmultimedia.com/ (программа FlowCode)

http://www.microchip.com/ (среда программирования MPLAB)

http://www.htsoft.com/ (компилятор Си HI-TECH)

http://sdcc.sourceforge.net/ (компилятор Си SDCC)

Последние версии программы MPLAB при установке позволяют загрузить и установить компилятор PICC Lite (бесплатная версия имеет ряд ограничений). Предыдущие версии достаточно легко позволяют использовать полнофункциональный и бесплатный компилятор SDCC.

Если у вас есть возможность приобрести (или собрать) программатор, работающий в среде FlowCode (или MPLAB, или AVRStudio), вы избавитесь от необходимости применять программу для обслуживания программатора.

Не ищите в этом рассказе решение вашей задачи. Здесь будет то, что необходимо вам в начале пути к той цели, которую можно было бы обозначить, как освоение работы с микроконтроллерами. Весь путь вам придется проделать самому. И если вы не хотите, чтобы он превратился в пытку, в изнуряющую необходимость идти, превозмогая себя, то не пытайтесь сразу найти самое таинственное место программы, которое превратит невозможное в возможное. Не пытайтесь сразу найти в одном месте ответы на все вопросы, которые у вас возникают, а наберитесь терпения и находите удовольствие в тех небольших победах, которые у вас обязательно будут, когда из непонимания вырастает знание, а из ошибок и неполадок работающее устройство.

Знакомство с интерфейсом программы FlowCode

Свободно распространяемая версия при первом запуске покажет следующее:

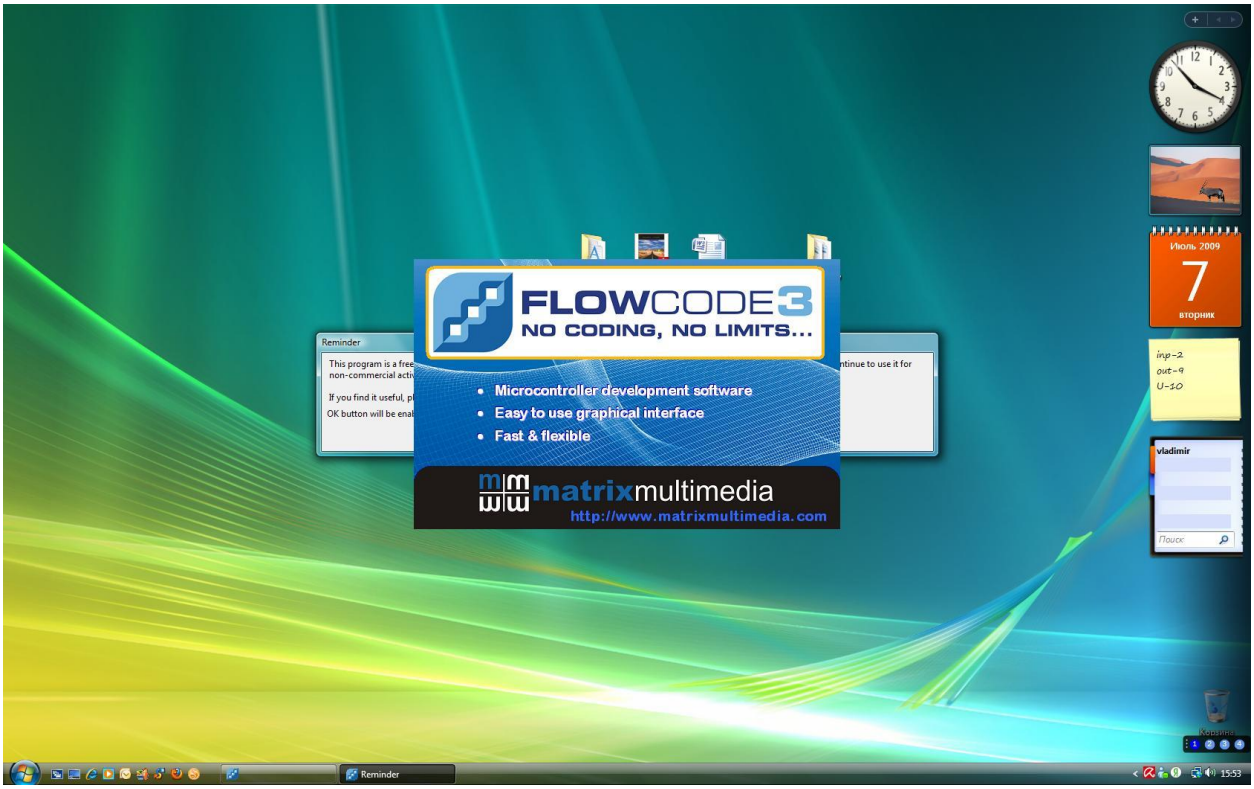


Рис. 1.1. Запуск демо-версии FlowCode

О том, что это демонстрационная версия постоянно напоминает заставка.

Рис. 1.2. Заставка демо-версии

Она же появляется и тогда, когда вы выходите из программы. Если не обращать внимания на эти мелочи и на то, что есть ряд существенных ограничений, то можно продолжить работу с программой. Следующий диалог предлагает вам создать новый проект или открыть уже существующий. Существующий проект можно открыть двойным щелчком в этом окне.

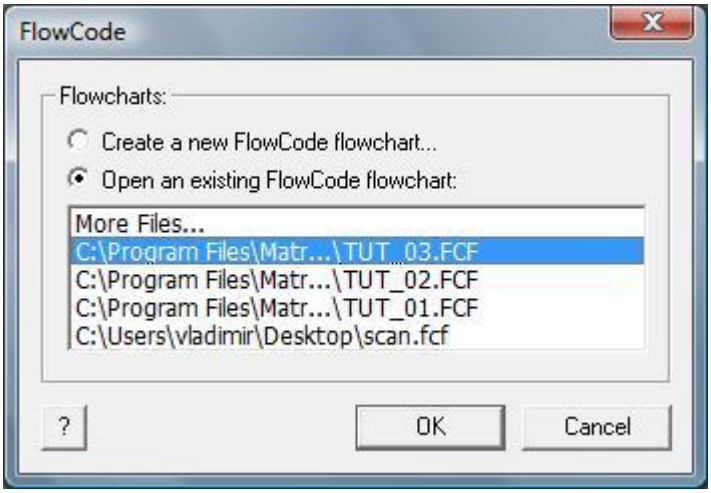


Рис. 1.3. Выбор продолжения работы

Если вы выбираете создание новой программы (опция *Create a new FlowCode flowchart*), то в следующем диалоговом окне будет предложено выбрать модель микроконтроллера (в демо-версии Flowcode_AVR только одна модель).

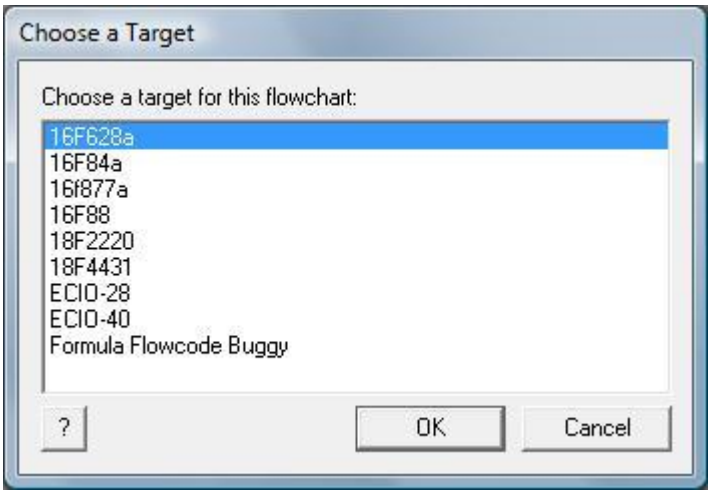


Рис. 1.4. Выбор модели контроллера

В полной версии программы этот список гораздо длиннее. Но этого более чем достаточно, если вы начинаете осваивать программирование микроконтроллеров. Конечно, вам захочется не только посидеть за компьютером, устанавливая программы и разбираясь в работе с ними, вам захочется запрограммировать микросхему, собрать что-то на макетной плате и посмотреть, как «оно живьем». Очень советую, по этой же причине и рассказываю, как правило, об одной единственной модели PIC16F628A, купить эту микросхему. В ней нет ничего, что так бы ее выделяло среди других, но она есть в демоверсии программы FlowCode, она достаточно хорошо (как PIC16F627A) поддерживается компилятором PICC Lite, программатор для нее можно быстро спаять и саму микросхему, если ее нет в ближайшем магазине радиотоваров, купить в почтовом агентстве «Десси». Стоит она не дорого. Позже, когда вы вполне освоитесь, у вас появятся критерии выбора микроконтроллера. Тогда и список поддерживаемых моделей вам понадобится длинный. А, может быть, и не понадобится. Микроконтроллер PIC16F628A не столь плох, чтобы не использовать его и дальше.

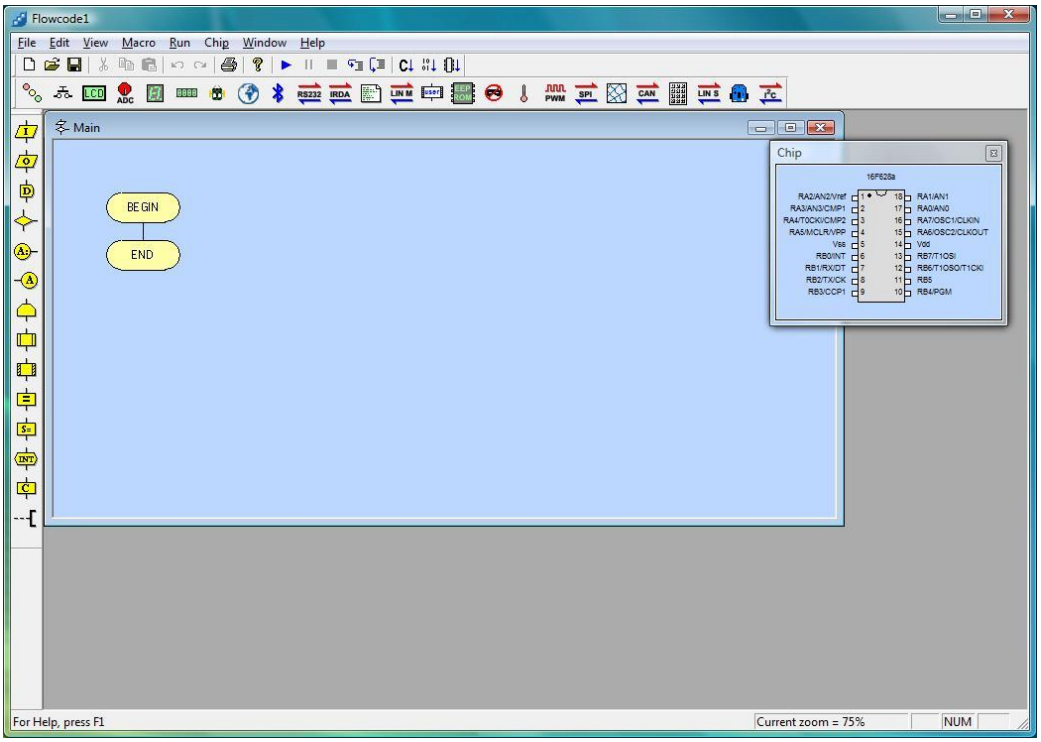


Рис. 1.5. Вид программы после выбора модели контроллера

Итак. Окно с надписью *Main* – это рабочее поле программы. Маленькое окошко с надписью *Chip* – это информационное окошко, где показана цоколевка выбранной модели. Программа, как любая другая, имеет основное меню и несколько инструментальных панелей. Каждая из них может быть перенесена в любое удобное для вас место – подцепите ее мышкой и перетащите.

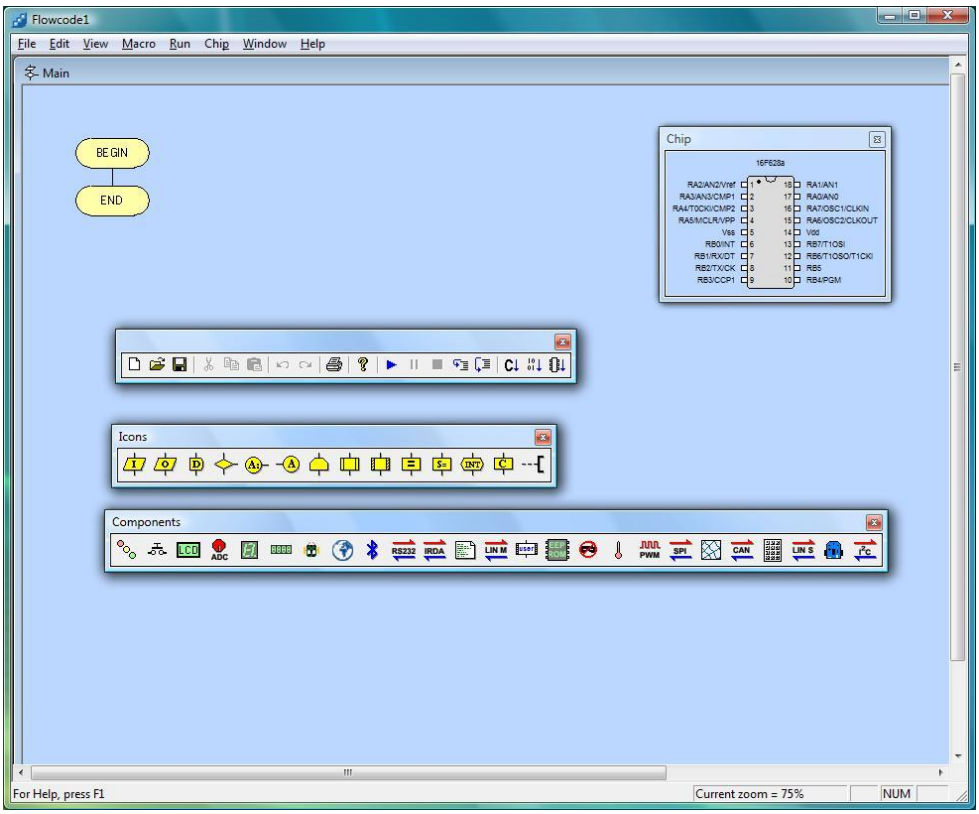
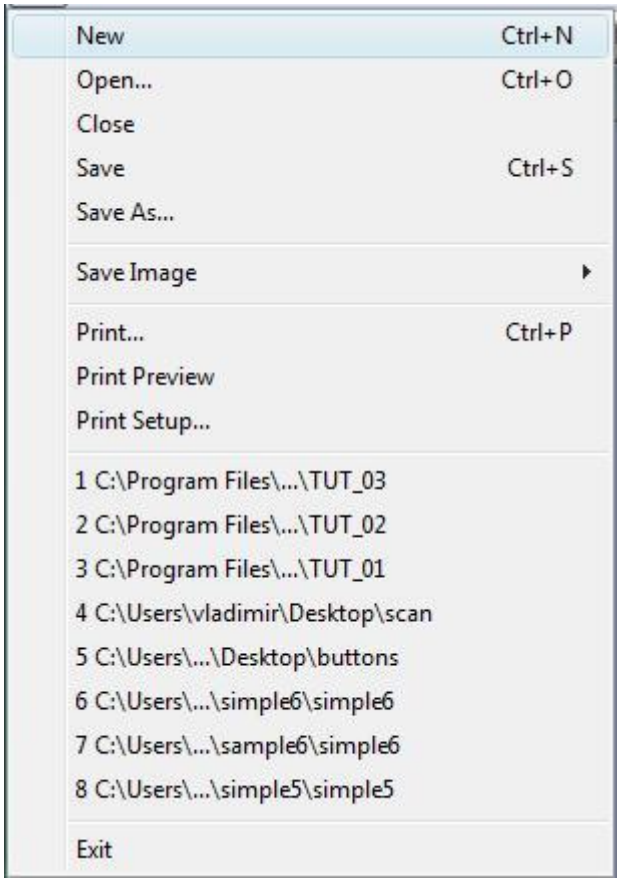


Рис. 1.6. Возможное расположение инструментальных панелей в FlowCode

Основное меню программы

Основное меню начинается, редко встречается другой вариант, с команд работы с файлами.



В полновесной версии в этом меню есть еще один полезный раздел – **Import**.

Его использование позволяет импортировать программу, написанную для AVR-контроллера, чтобы использовать ее с PIC-контроллером.

Рис. 1.7. Меню **File** основного меню программы

Меню содержит стандартный набор операций: ***New*** — создать новый файл; ***Open***... — открыть уже существующий; ***Close*** — закрыть файл; ***Save*** — сохранить файл; ***Save As***... - сохранить как..., удобно на тот случай, если вы хотите сохранить файл под другим именем или в другой папке; следующий пункт выпадающего меню ***Save Image***, позволяет сохранить вам программу в виде картинки в двух графических форматах, которые вы выбираете в открывающемся меню, если курсор наведен на этот пункт; пункты меню, начиная с ***Print***..., относятся к выводу на печать, последний из них позволяет настроить принтер, а ***Print Preview*** предварительно увидеть, как будет выглядеть вывод на печать. Завершают этот список перечень недавно открывавшихся файлов и выход из программы ***Exit***.

Почти все выпадающие подменю основного меню программы контекстно-зависимые: если нет объекта для выполнения операции, раздел меню не активен, надпись блеклая. Только тогда, когда есть, что делать конкретной команде, раздел меню активизируется, надпись становится четкой.

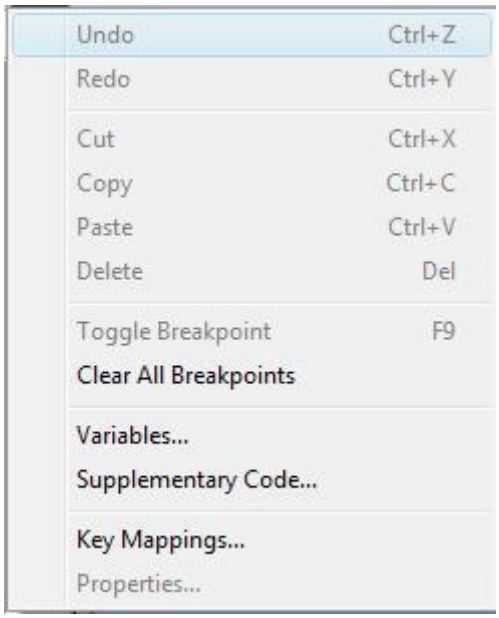


Рис. 1.8. Меню **Edit** основного меню

Я хочу подчеркнуть, что это вид выпадающего меню из списка основного меню, поскольку, щелкнув правой клавишей мышки в окне графического редактора, в зависимости от места, по которому вы щелкните, вы получите выпадающие меню для работы с элементами графики.

В этом меню первая половина относится к обычным операциям с объектами любого редактора: ***Undo*** и ***Redo*** — две взаимно противоположные команды по отмене последнего действия и возвращения его; ***Cut*** — вырезать; ***Copy*** — копировать; ***Paste*** — вставить; ***Delete*** — удалить. Два следующих пункта относятся к режиму отладки программы: ***Toggle Breakpoint*** — переключение точки останова (точка останова позволяет вам остановить выполнение программы при отладке, чтобы, например, посмотреть значения переменных), если точка останова была включена (задана), то она выключится, и наоборот; ***Clear All Breakpoints*** — очистит вашу программу от всех точек останова, которые вы задавали.

Следующий пункт в меню ***Variables***... открывает диалоговое окно, в котором вы можете определить переменные вашей программы, если они вам нужны. Программы всегда полны переменных, с которыми они и работают.

Следующий пункт ***Supplementary Code***... (дополнительный код) позволяет при желании (или необходимости) добавить код к графической программе, открывая окно диалога, где есть поле для объявления функции и поле для реализации этой дополнительной функции. Это соответствует языковым конструкциям с разделением ***Definitions and function declarations*** (определение и объявление функции) и ***Function implementations*** (реализация функции).

В соответствующих окнах мини-редакторов можно ввести текст, который после нажатия клавиши **OK** позволит использовать эту дополнительную функцию. Возможность добавить нужные коды в программу значительно расширяет функциональность FlowCode для профессионалов, но и любителям, уже освоившимся в программе, достаточно хорошо владеющим языком программирования, это дополнение не покажется лишним.

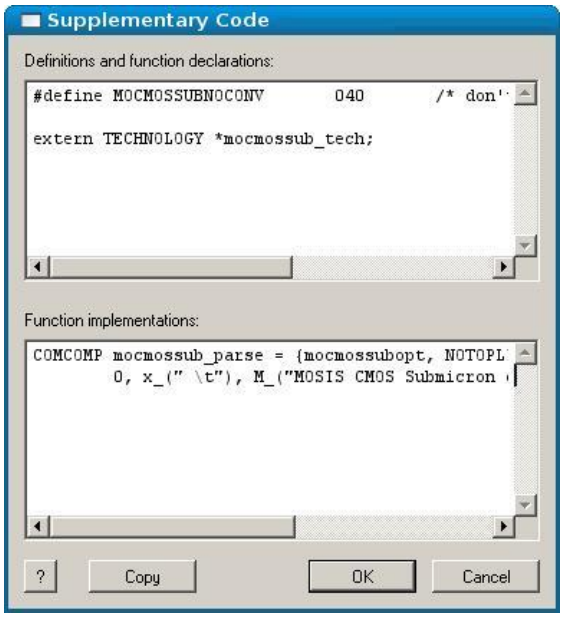


Рис. 1.9. Диалоговое окно добавления функции

Как я и говорил, все выпадающие меню в программе «чувствительны» к происходящему в активном окне редактирования. И пока вы не добавите для отладки в программу клавиатуру (**KeyPad**) из набора дополнительных элементов, следующий пункт ***Key Mappings***... (карта соответствий) не будет активен. Но с появлением клавиатуры в проекте вы можете изменить свойства этого дополнительного элемента.

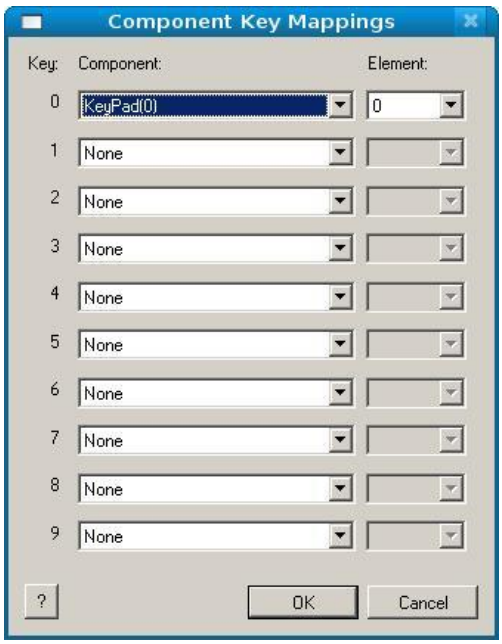


Рис. 1.10. Окно ***Key Mappings***

Аналогично поведение и последнего элемента списка из набора возможностей редактирования — ***Properties***... (свойства). Чтобы «оживить» его, вам следует выделить любой элемент программы. Тогда, щелкнув по этому пункту меню, вы попадете в окно диалога свойств выбранного элемента. Вид этого диалога и предоставляемые пользователю возможности зависят от выделенного элемента. Так для элемента **Input** (вход) можно задать привязку к переменной и к порту, выбрать работу с одним битом или всем портом, задать маску, поставив галочку в поле *Use Masking*.

Каждый элемент программы в FlowCode имеет свой набор свойств, назначение и смысл которых яснее всего проявляются тогда, когда вы начинаете работать над программой. У вас обязательно появятся вопросы вида — как сделать так, чтобы в конкретном месте программы проверить состояние только одного бита порта, а не всех вводов? Например, когда вы хотите использовать ветвление программы по условию, зависящему от состояния одного бита. Очень часто это проверка состояния «флага» (одного бита переменной или регистра, меняющего свое значение по окончании процесса или по результату операции).

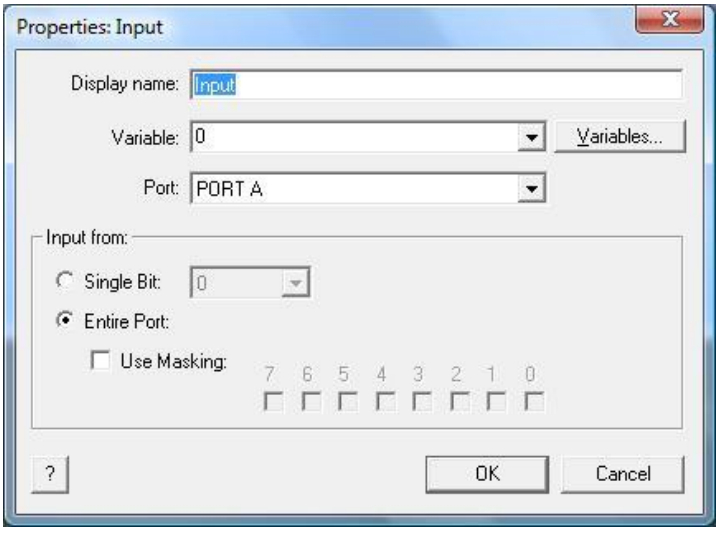


Рис. 1.11. Диалоговое окно свойств программного элемента **Input**

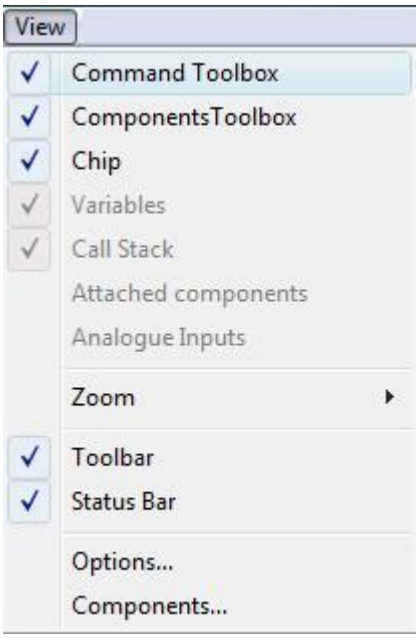


Рис. 1.12. Подменю **View** (вид)

«Галочки» слева показывают, какие компоненты вы желаете видеть при работе: ***Command Toolbox*** (инструментальное меню команд), ***Components Toolbox*** (инструментальное меню компонент), ***Chip*** (контроллер, с которым вы работаете), ***Variables*** (переменные, если они есть), ***Call Stack*** (стек вызовов подпрограмм), *Attached components* (присоединенные компоненты, если вы добавили клавиатуру или светодиоды для отладки), ***Analogue Inputs*** (аналоговые входы).

Присоединенные компоненты появляются в виде дополнительного списка, на что указывает стрелка справа от пункта меню. Как и дополнительный список масштаба изображения ***Zoom***, который вы можете выбрать, увеличить или уменьшить, привести к заполнению экрана или ширине окна редактирования.

Вы можете также включить или выключить ***Toolbar*** (основное инструментальное меню) и ***Status Bar*** (строку состояния). Вы можете изменить свойства проекта, такие как цвет рисунка и фона, шрифт, используя раздел ***Options...,*** или выбрать из предлагаемого списка, какие компоненты вам нужны в настоящее время для работы — ***Components...*** Все это позволяет вам выбрать комфортный в вашей работе вид программы: только нужные компоненты, цветовую гамму, возможность легко читать все надписи в элементах программы и т.д. Не забывайте только, что, выключив из списка компонентов что-то, вы не увидите этот компонент при следующем запуске программы, и, если сегодня он вам не нужен, то завтра может понадобиться.

Следующие пункты основного меню я опишу кратко. Тому несколько причин — многое из этого вам понадобится не сразу, а, когда понадобится, вы будете разбираться в среде программирования микроконтроллеров FlowCode не хуже меня; рассказ о перечне пунктов меню к тому времени, когда вам понадобится воспользоваться чем-то, забудется, и легче самому сообразить, чем найти в тексте описания; да и не интересно начинать работу с программой, если ты о ней все уже знаешь, гораздо приятнее делать «свои маленькие открытия».

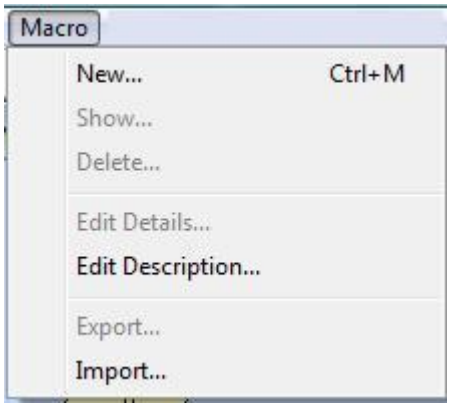
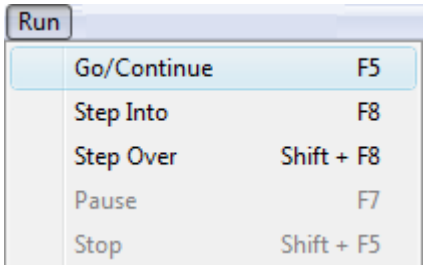


Рис. 1.13. Подменю **Macro** (макрос)

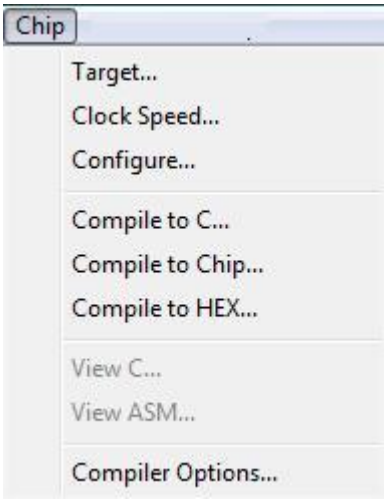
New... (новый макрос), ***Show...*** (показать макрос), ***Delete...*** (удалить), ***Edit Details...*** (редактировать детали), ***Edit Description...*** (редактировать описание, которое очень полезно делать), ***Export...*** (экспортировать), ***Import...*** (импортировать) предварительно экспортированную подпрограмму (макрос).



Go/Continue (запуск/продолжение), ***Step Into*** (шаг внутрь, например, функции), ***Step Over*** (шаг через, например, условие), ***Pause*** (пауза), ***Stop*** (стоп).

Рис. 1.14. Подменю работы с отладчиком программы

Отладка написанной программы – это очень важный компонент любой среды разработки программы. Этому больше внимания будет уделено в следующих главах.



Target... (выбор модели), ***Clock Speed***... (частота тактового генератора), ***Configure***... (слово конфигурации), ***Compile to C***... (транслировать на Си), ***Compile to Chip***... (транслировать все и загрузить), ***Compile to Hex***... (получить hex-файл загрузки), ***View C***... (просмотр Си кода), ***View ASM***... (просмотр ассемблерного кода), ***Compiler Options***... (опции компиляции).

При установке программы есть один момент, имеющий отношение к пункту ***Configure***.

Рис. 1.15. Подменю раздела **Chip**

При установке программы появляется следующее окно диалога.

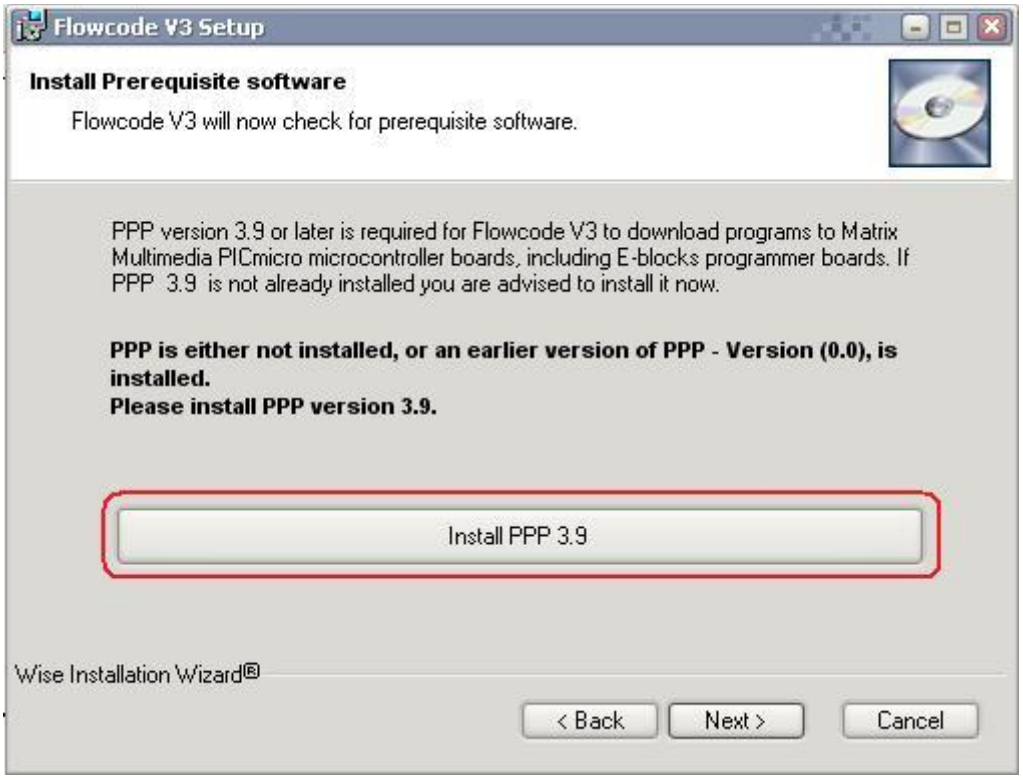
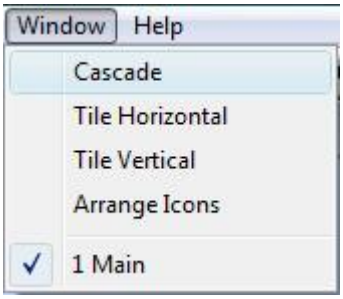


Рис. 1.16. Один из моментов установки программы FlowCode

Обязательно нажмите отмеченную на рисунке кнопку **Install PPP 3.9** до того, как нажмете кнопку **Next >**. При установке этой части программы в папке, где расположена программа FlowCode появляется папка с именем *Common*. В ней располагается то, что позволит вам ввести слово конфигурации для микроконтроллера.



Черепицей, горизонтально, вертикально, упорядочить иконки — вот, что можно сделать с окнами.

Рис. 1.17. Подменю **Window**

Последний пункт основного меню **Help**, на тот случай, если есть желание или необходимость обратиться к руководству по работе с программой.

Инструментальная панель программных компонентов

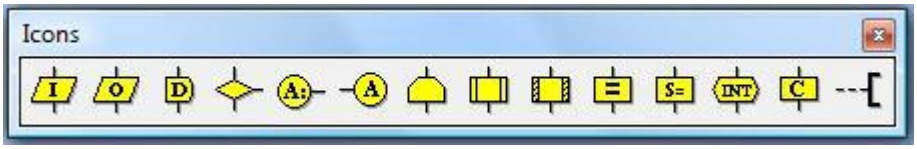


Рис. 1.18. Программные компоненты

Перечень представленных компонентов (слева-направо на рисунке, сверху-вниз, когда панель справа):

Input (ввод), ***Output*** (вывод), ***Delay*** (пауза), ***Decision*** (ветвление), ***Connection Point*** (две точки соединения), ***Loop*** (цикл), ***Macro*** (макрос), ***Component Macro*** (макрос компонента, добавленного в программу), ***Calculation*** (вычисление), ***String Manipulation*** (строковые операции), ***Interrupt*** (прерывание), ***C Code*** (блок кода на языке Си), ***Comment*** (комментарий).

Все эти названия появляются в виде подсказки, когда курсор мышки наведен на компонент.

Рассмотрим некоторые из программных компонентов подробнее. Так «взаимодополняющие» или «взаимоисключающие» компоненты **Input** и **Output** имеют очень похожие диалоговые окна свойств. Известно, что, как правило, любой из выводов порта микроконтроллера можно назначить для входа или для выхода. Можно это сделать и для всего порта. Тогда весь порт будет работать, как входной или выходной. Эти свойства выбираются в диалоговом окне (после добавления компонента к программе достаточно дважды щелкнуть по нему левой клавишей мышки или выделить одинарным щелчком и воспользоваться основным меню ***Edit->Properties***).

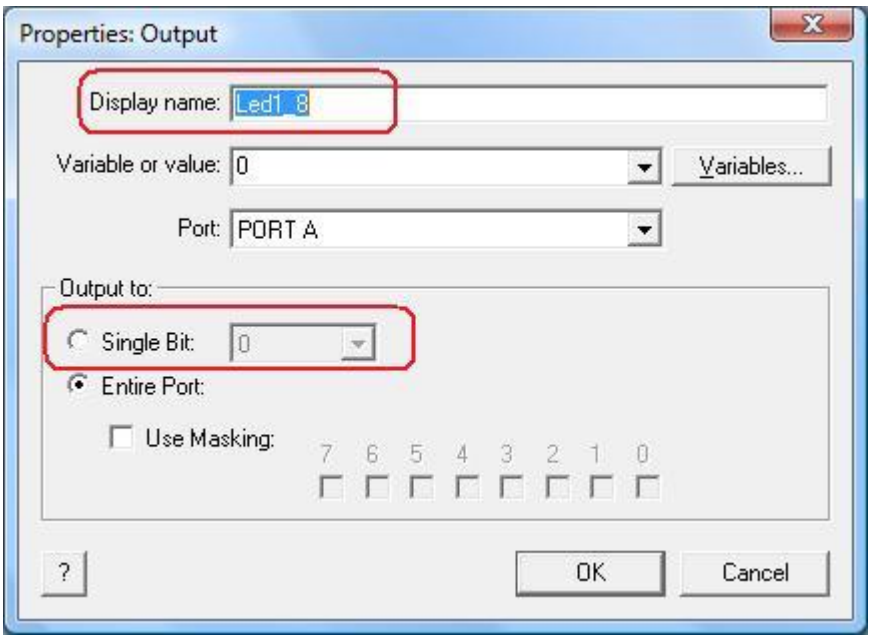


Рис. 1.19. Диалоговое окно программного компонента **Output**

По умолчанию в окне *Display name:* (отображаемое имя) появляется *Output*. Его можно оставить, но можно заменить более понятным, скажем, для вас. Работая над программой, легче понять ее, а «разрастается» она быстро, когда вы описываете назначение всех элементов программы. FlowCode позволяет вписать название кириллицей. Но я не советую это делать. Лучше написать что-то в роде: svetodiody, – если вы, как и я, не слишком дружите с английским языком.

На рисунке отмечено, что вы можете использовать единственный бит (единственный вывод) порта. По умолчанию используется весь порт. Кнопка справа от окошка со стрелкой вниз позволяет выбрать из выпадающего списка нужный бит. А кнопка справа от окошка с именем порта предлагает список портов.

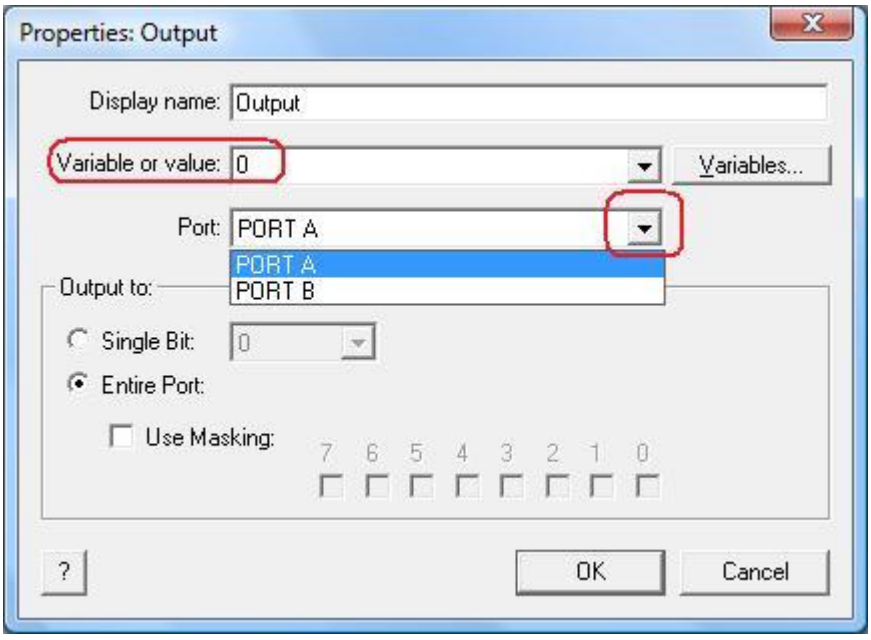


Рис. 1.20. Выбора нужного порта для выхода

Чуть выше вы задаете значение (*Variable or value:*), которое нужно вывести в порт или задать биту порта. Используя кнопку ***Variables*** можно выбрать существующую переменную для вывода в порт. Закончив настройку выхода, вы можете подтвердить сделанные изменения, нажав на кнопку ***OK***, или можете отменить все с помощью кнопки ***Cancel***.

Следующий программный компонент – **Delay** (пауза). Нужный компонент, особенно когда вы формируете выходной импульс, а формирование импульсов очень часто используется. После добавление компонента в его диалоговом окне задается значение.

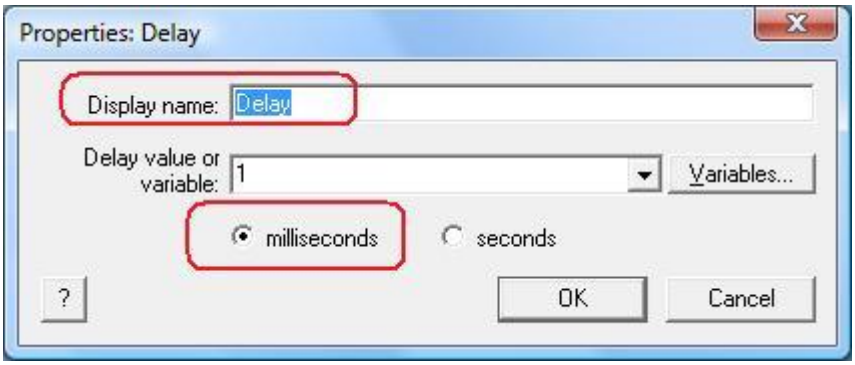


Рис. 1.21. Диалоговое окно компонента **Delay**

Как и в других окнах диалога, можно этому компоненту дать имя, которое отражало бы его назначение. По умолчанию пауза задается числом миллисекунд. Изменив опцию на *seconds*, можно задать время в секундах. Кроме того, используя клавишу ***Variables***, можно выбрать переменную, если она была создана для этого случая, которая задаст значение для паузы.

Есть некоторое неудобство в том, что программа FlowCode оперирует с целыми числами, и нельзя задать время паузы, скажем, 0.1 мс. Эту проблему можно обойти, если использовать вставку на языке Си, ассемблере или задавая счетный цикл с количеством нужных проходов. Но об этом позже.

Следующий программный компонент – ***Decision*** (ветвление). Только очень простые программы обходятся линейным написанием, когда выполняется какое-то количество операций подряд. Да и такие программы, как правило, выполняются в цикле. Обычно контроллер отслеживает события, происходящие на его входах (или входе), по результатам опроса входов программа проходит по одной или другой ветке программы. Например, такой фрагмент программы:

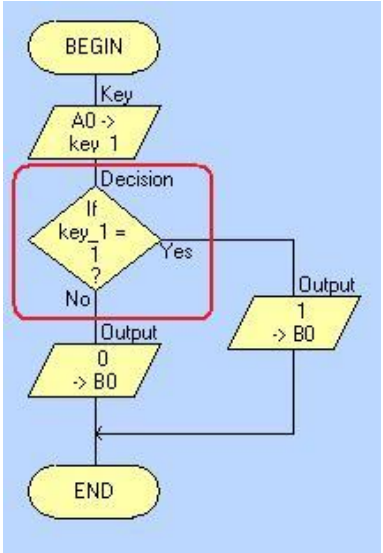


Рис. 1.22. Фрагмент программы с компонентом **Decision**

В этом фрагменте нулевой вывод порта A назначен на вход. Если его состояние (он связан с переменной *key_1*) равно единице, то программа проходит по ветке ***Yes***, если нулю, то по ветке ***No***. В первом случает нулевой вывод порта B (назначенный на выход) принимает высокий уровень, то есть, единицу, во втором низкий, то есть, ноль. Если этот фрагмент заключить в бесконечный цикл, то можно посмотреть в отладчике FlowCode, как программа реагирует на нажатие кнопки.

Условие ветвления вводится в диалоговом окне.

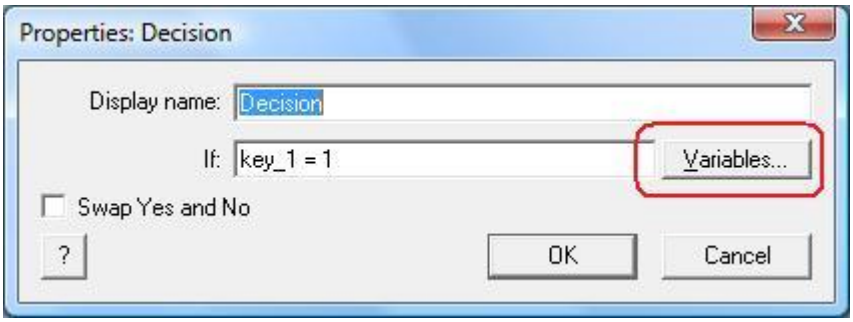
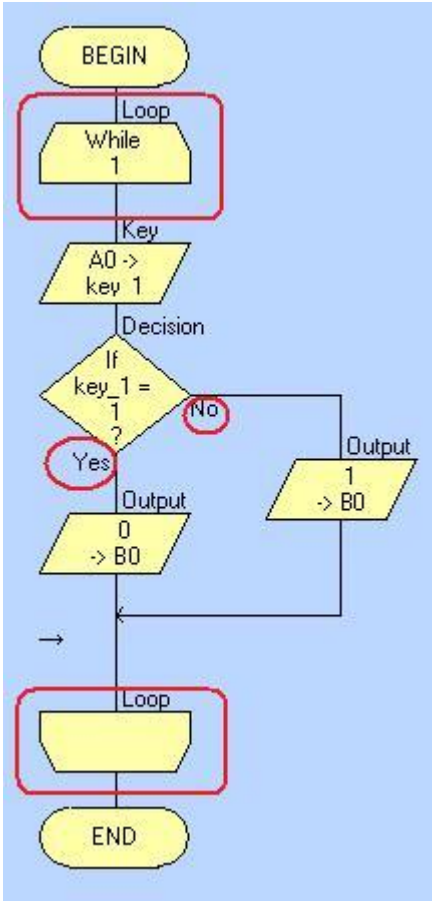


Рис. 1.23. Диалоговое окно компонента **Decision**

С помощью кнопки ***Variables***, отмеченной на рисунке, я выбрал ранее созданную (когда оформлялся ввод) переменную, продолжив запись условия ветвления. Это условие может быть простым, как выше, но может быть и сложным, использующим, например, логические операции и переменные.

Опция *Swap Yes and No* меняет местами эти ветки. Иногда так удобнее.

О следующих двух (они используются парой) компонентах мы поговорим позже, а сейчас остановим внимание на очень важном элементе программы – **Loop** (цикл). Приведенный выше фрагмент программы вполне «правильный». Логически. Но «нежизненный». Даже проверить его работу с помощью отладчика можно только в пошаговом режиме. А, если скомпилировать программу и загрузить в микросхему, то даже проверить ее не получится. Программа выполнится так быстро, что нажать кнопку не успеешь; правда, можно один раз включить схему при отжатой кнопке, а второй раз при нажатой. Но ценность такой программы равна нулю. Другое дело, если мы добавим бесконечный цикл. Заодно я использую опцию *Swap Yes and No*, о которой говорил выше.



Обратите внимание на ветки (отмеченные) *Yes* и *No*. И сравните с предыдущим фрагментом.

Цикл «окаймляет» программу, начинаясь с условия выхода из цикла. В данном случае *While 1*, поскольку никаких условий выхода из цикла я не задавал. Это не заданное условие, как и заведомо не выполнимое условие, приведут к тому, что программа, заключенная в цикл, будет повторяться бесконечно.

Такие бесконечные циклы опасны, если созданы по ошибке, ошибка при написании программы или сбой программы, и опасны тем, что программа «повисает». Она перестает реагировать на внешний мир или реагирует только на часть нужных команд.

С другой стороны, когда это сделано осознанно, так работает почти каждый контроллер: он «крутится» в бесконечном цикле.

Рис. 1.24. Фрагмент вышеприведенной программы с циклом

Свойства цикла, как и других программных компонентов, задаются в его диалоговом окне.

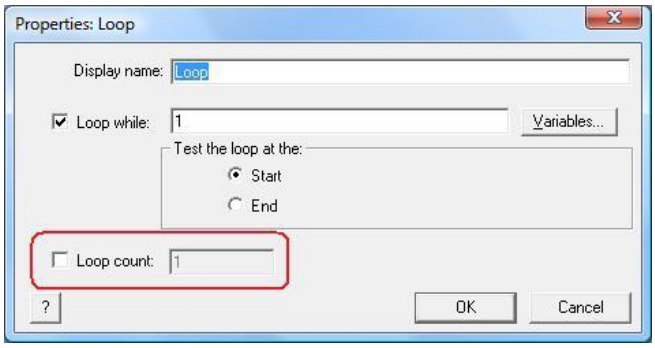


Рис. 1.25. Диалоговое окно компонента **Loop**

С помощью этого диалога можно задать условие выхода из цикла, используя кнопку ***Variables*** для выбора или создания переменной. Как правило, эта переменная должна меняться внутри цикла. Если в вышеприведенном фрагменте использовать переменную, связанную с кнопкой входа порта А, то можно при нажатии, скажем, кнопки выходить из цикла. Иногда переменную создают внутри цикла, которая меняется при работе фрагмента программы, заключенного в цикл, а когда достигает заданного значения, цикл прекращает работу.

Проверка условия завершения цикла может происходить в начале цикла, если установлена опция *Start*. При этом если условие выполняется до выполнения программы, заключенной в цикл, то

программа не будет выполнена ни разу. Если установить опцию *End*, то проверка выполнения условия будет происходить в конце программы, заключенной в цикл. При этом программа будет выполнена хотя бы один раз.

В этом же диалоге можно сменить вид цикла. Ранее мы говорили об условном цикле, то есть, таком, когда программа внутри цикла выполняется до тех пор, пока не будет выполнено условие выхода из цикла. Если установить опцию *Loop count*, задав в окне число, то цикл будет выполняться до того момента, когда количество проходов программы в цикле станет равным заданному числу.

Следующий программный компонент – ***Macro*** (макрос). Или подпрограмма. Такой же смысл имеют процедуры и функции в языках программирования высокого уровня. Обычно нужда в этом элементе программы обусловлена тем, что к нему обращаются много раз за время работы программы, а выполняемые им операции одни и те же. В языке Паскаль, например, различают процедуры – фрагменты программы, в которых выполняются операции – и функции, когда производятся вычисления. В языке Си оба вида объединены под именем «функция». В FlowCode для этого введен компонент **Macro**.

После добавления компонента в программу мы можем перенести в него ранее созданную процедуру. Для этого дважды щелкнем по компоненту **Macro**, открывая диалоговое окно.

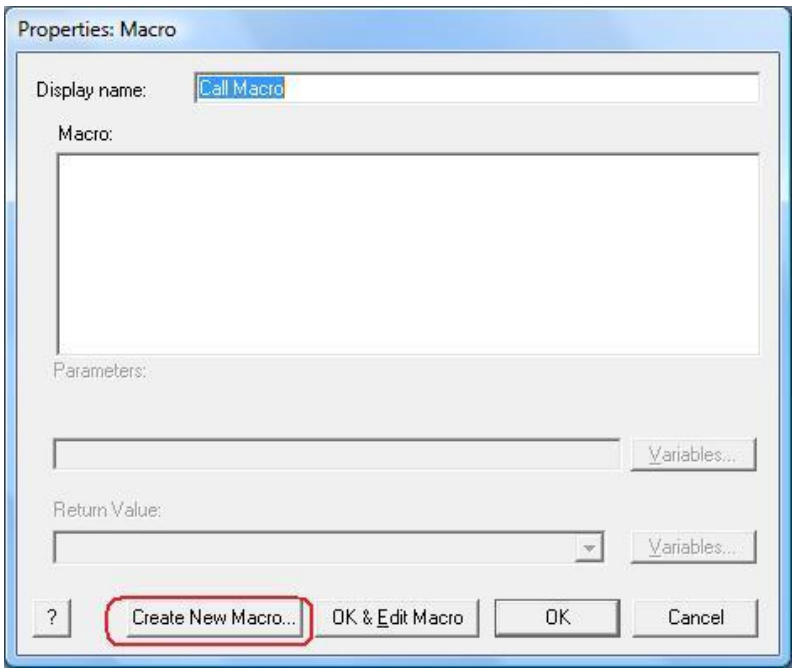


Рис. 1.26. Диалоговое окно компонента **Macro**

И вновь, имя компонента можно изменить. Если подпрограмма еще не создана, а мы только что добавили компонент, следует использовать кнопку ***Create New Macro***.

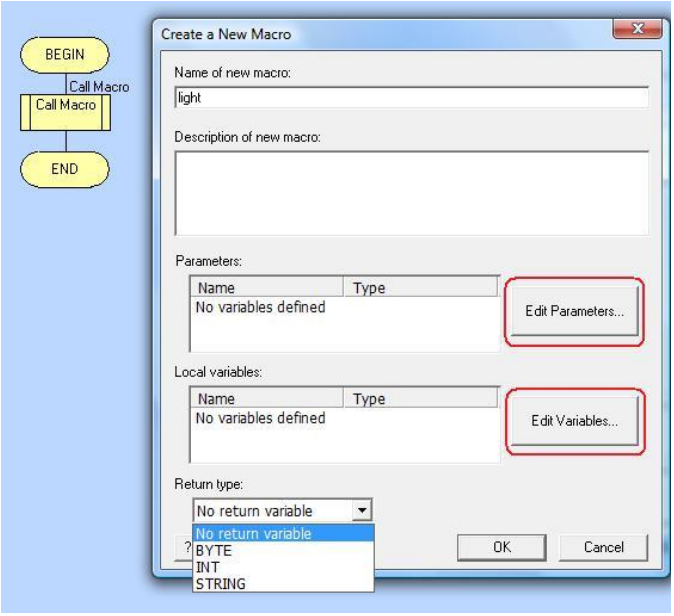


Рис. 1.27. Диалоговое окно создания нового макроса

Кроме имени новой подпрограммы, как и для функции или процедуры можно задать параметры (кнопка ***Edit Parameters***), можно создать локальные переменные (кнопка ***Edit Variables***), можно задать тип возвращаемой переменной (если подпрограмма возвращает переменную), который выбирается из выпадающего списка. Нажав кнопку ***OK***, вы возвратитесь в предыдущий диалог. Если нажать кнопку ***OK & Edit Macro***, то программа открывает второе рабочее окно для создания подпрограммы.

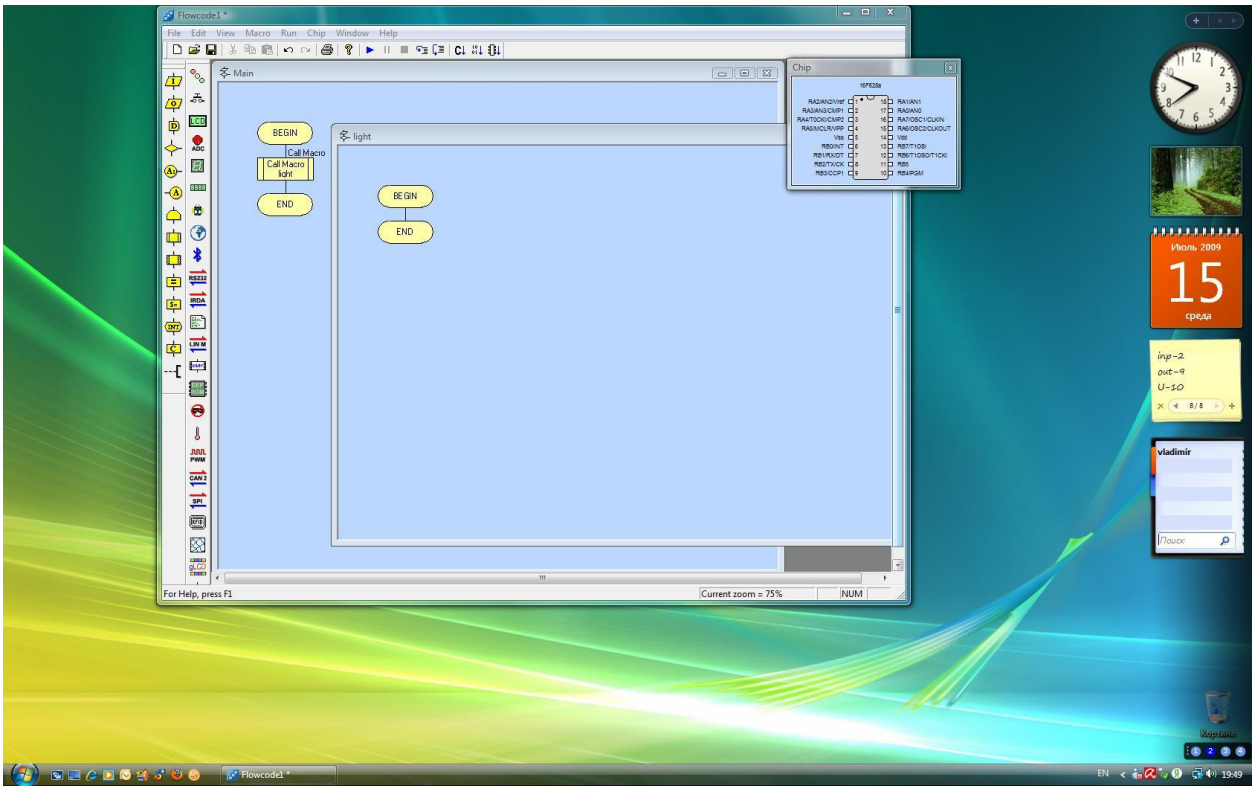


Рис. 1.28. Второе рабочее окно редактора для создания подпрограммы

К этому мы вернемся позже, а сейчас отметим, что между созданием программы и созданием подпрограммы в FlowCode в техническом плане нет разницы.

Опустим ряд очень полезных программных компонентов, назначение которых, свойства и применение удобнее рассмотреть на конкретных примерах, и рассмотрим два компонента: ***Calculation*** (вычисление) и ***Comment*** (комментарий).

Необходимость в компоненте ***Calculation*** возникает уже тогда, когда необходимо произвести присваивание значения переменной. В разных языках программирования этот оператор может иметь разный вид. В FlowCode он совпадает со знаком равенства.

Используемый для расчетов компонент может содержать одно или несколько выражений, результаты операций можно использовать дальше в программе.

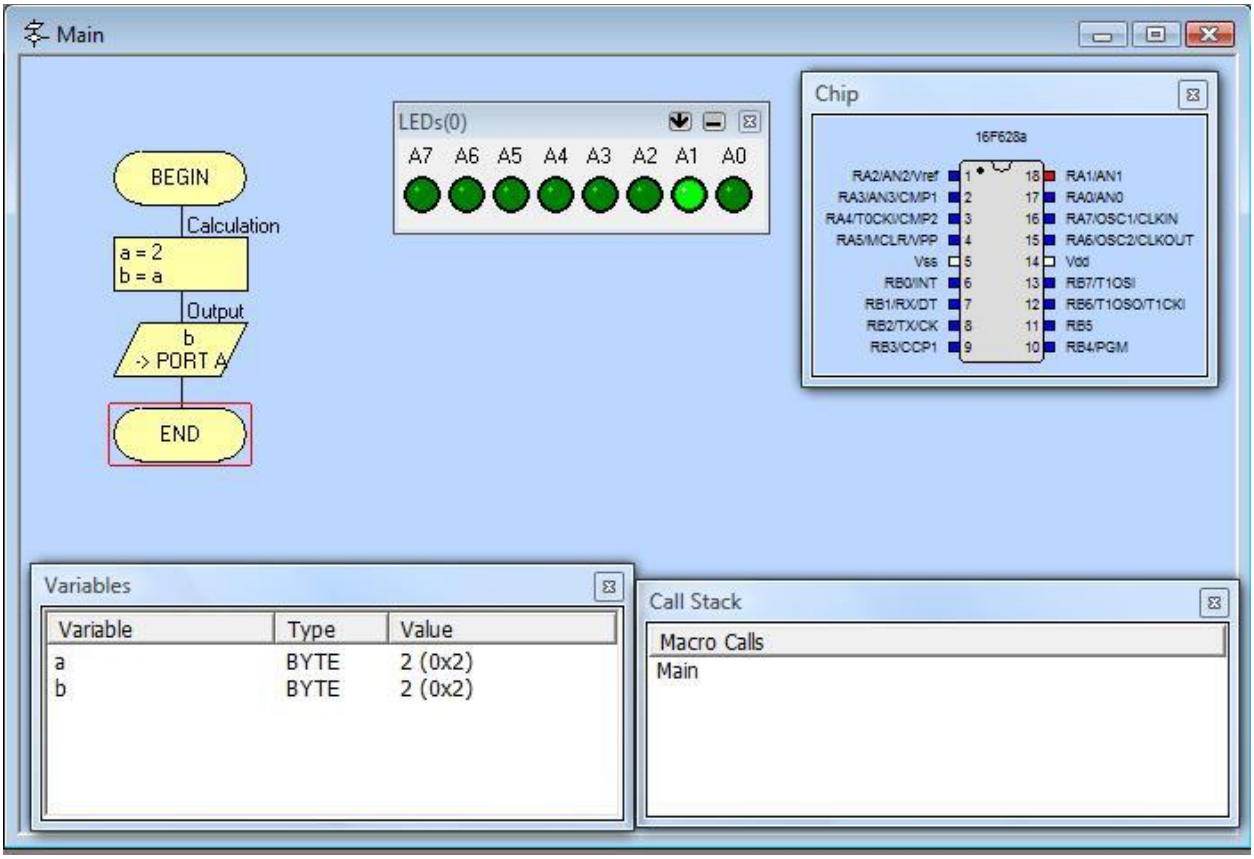


Рис. 1.29. Использование компонента ***Calculation***

Все необходимые операции записываются в окне диалога.

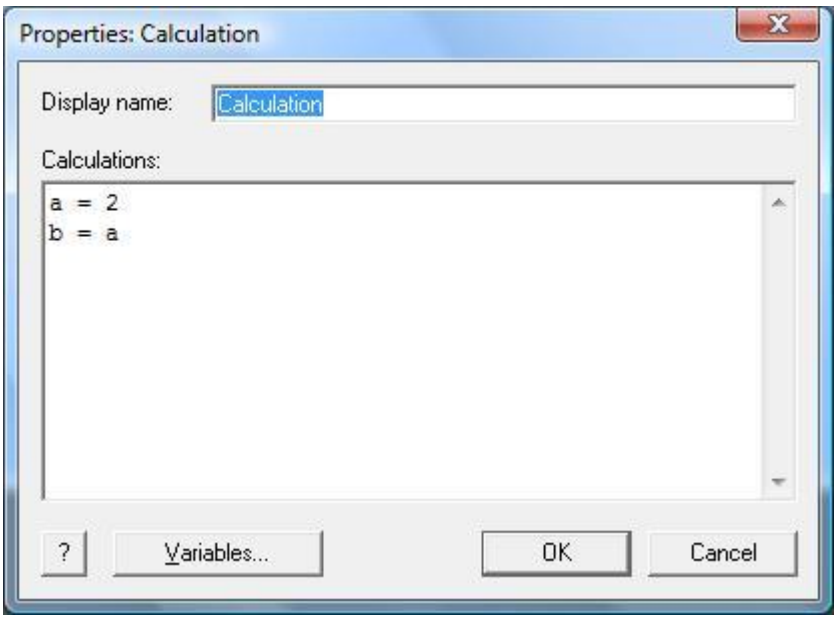


Рис. 1.30. Диалоговое окно компонента ***Calculation*** (вычисление)

Каждое новое выражение следует писать с новой строки, если в выражении допущена синтаксическая ошибка, появляется сообщение об ошибке, а строка, где допущена ошибка, подсвечивается. Кнопка ***Variables*** служит для выбора переменных из списка всех переменных программы.

Программный компонент ***Comment*** (комментарий) используется для добавления к программе пояснений. Хотя формально все, что стоит за знаком комментария (или внутри скобок комментария), компилятор должен игнорировать, лучше писать комментарии латиницей, иначе могут возникать проблемы.

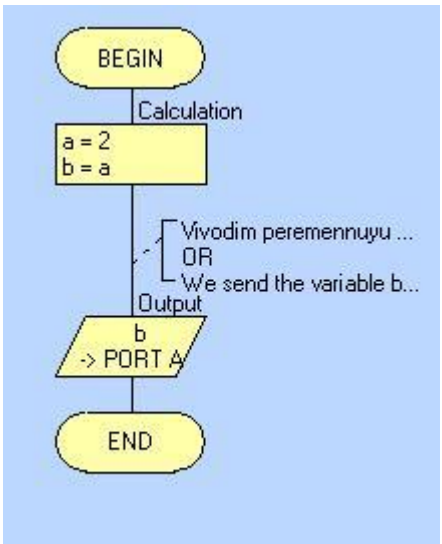


Рис. 1.31. Использование компонента ***Comment***

Для ввода комментария служит диалоговое окно свойств компонента ***Comment***.

Не менее важна для создания программ другая инструментальная панель – панель дополнительных компонентов. Она содержит элементы, необходимые при отладке программы. Набор кнопок, используемый для изменения состояния входов, или линейка светодиодов могут

использоваться не только тогда, когда устройство рассчитано на подключение, скажем, светодиодов. Светодиоды линейки можно использовать в качестве программных «заглушек», на месте которых позже появятся подпрограммы или прерывания; их можно использовать для отслеживания событий при тестировании программы.

Кроме внешних составляющих схемы, как семисегментный индикатор или линейка таких индикаторов, на панели есть модули, часто встраиваемые в микроконтроллеры. Если выбранная вами модель контроллера имеет, скажем, встроенный модуль USART, вы сможете проверить работу контроллера с помощью компонента RS232.

О назначении дополнительных модулей и их применении удобнее рассказать тогда, когда будут рассматриваться примеры программ.

Основная инструментальная панель

Рис. 1.32. Группа работы с файлами основной инструментальной панели

Основная инструментальная панель содержит несколько групп кнопок, повторяющих операции основного меню, начинаясь с группы работы с файлами: **New** (создание нового файла), **Open** (открытие существующего файла), **Save** (сохранение файла).

Диалоговые окна всех операций типовые. И здесь можно обратить внимание на то, что в папке Examples есть много примеров – обучающие программы.

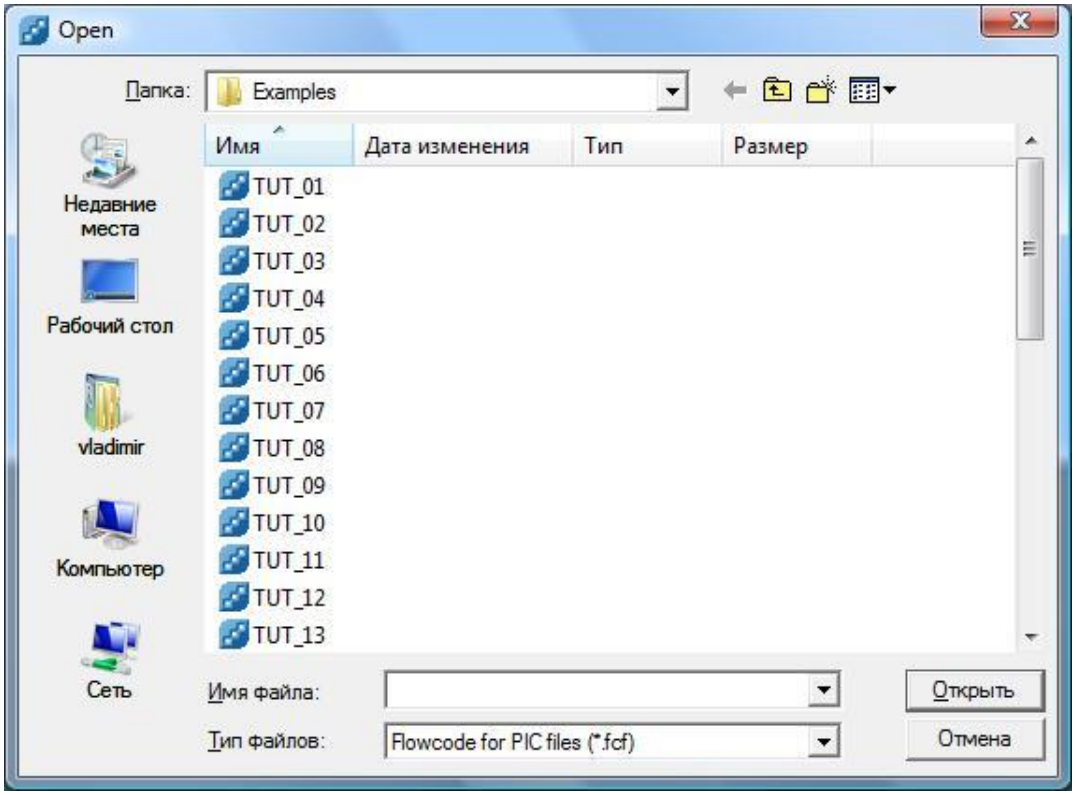


Рис. 1.33. Файлы примеров программы FlowCode

Следующая группа кнопок используется при редактировании.

Рис. 1.34. Группа редактирования основной панели

Cut (вырезать), **Copy** (копировать), **Paste** (вставить), **Undo** (отменить), **Redo** (вернуть) – обычный набор любого редактора. Пусть текст программы состоит не из букв, а из слов и фраз, но к ним применимы все операции редактирования, как к любому тексту.

Прежде, чем применять эти операции следует выделить элемент программы – достаточно щелкнуть по нему – или выделить фрагмент программы: поместить курсор на свободном месте рабочего поля над нужным фрагментом, нажать левую клавишу мышки и, удерживая ее, обвести прямоугольником выделения фрагмент программы.

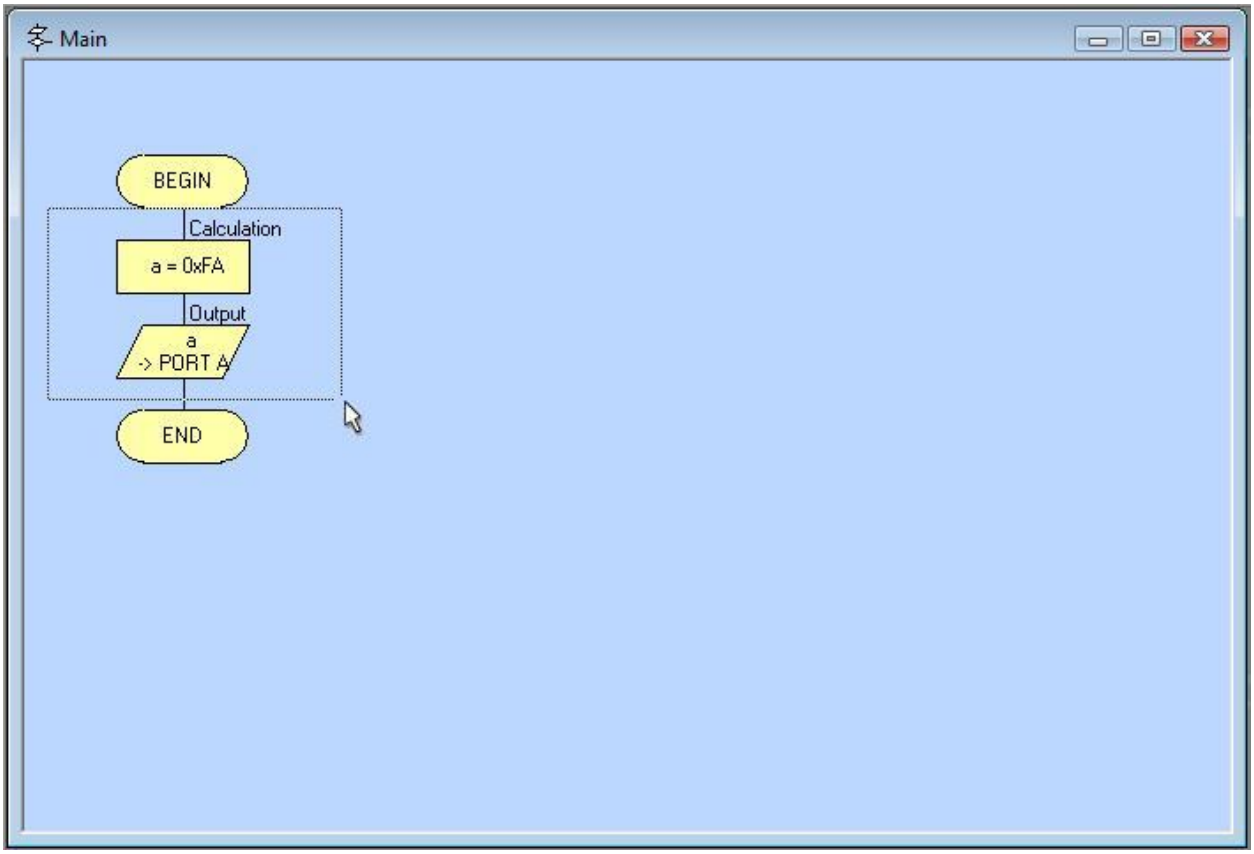


Рис. 1.35. Выделение фрагмента программы

После того, как вы отпустите клавишу, выделенный фрагмент меняет цвет. Теперь его можно вырезать, копировать, вставлять. Создав подпрограмму, можно перенести уже отлаженный фрагмент программы в подпрограмму, просто скопировав его.

Следом за командами редактирования на инструментальной панели расположены команды печати, команда распечатает программу на принтере, и команда вывода информации о версии программы.

Следующая группа команд относится к отладке программы.

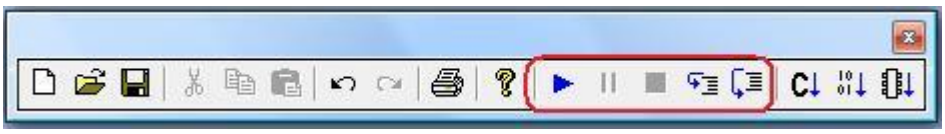


Рис. 1.36. Группа команд отладки

Первая кнопка «запускает» программу. Если для отладки программы добавлены светодиоды, группа выключателей, то, «нажимая» мышкой выключатели, можно увидеть, как реагируют светодиоды на эти нажатия (если программа поддерживает это). После запуска программы активизируются две следующие кнопки – пауза и стоп.

Отлаживая программу, бывает полезно посмотреть, как меняются переменные. Когда вы «запускаете» программу кнопкой ***Run***, вы можете и не увидеть изменения переменных. Есть два способа посмотреть значения переменных – установить точки останова. Для этого достаточно выделить элемент программы, щелкнуть правой клавишей мышки и выбрать из выпадающего меню **Toggle Breakpoint**.

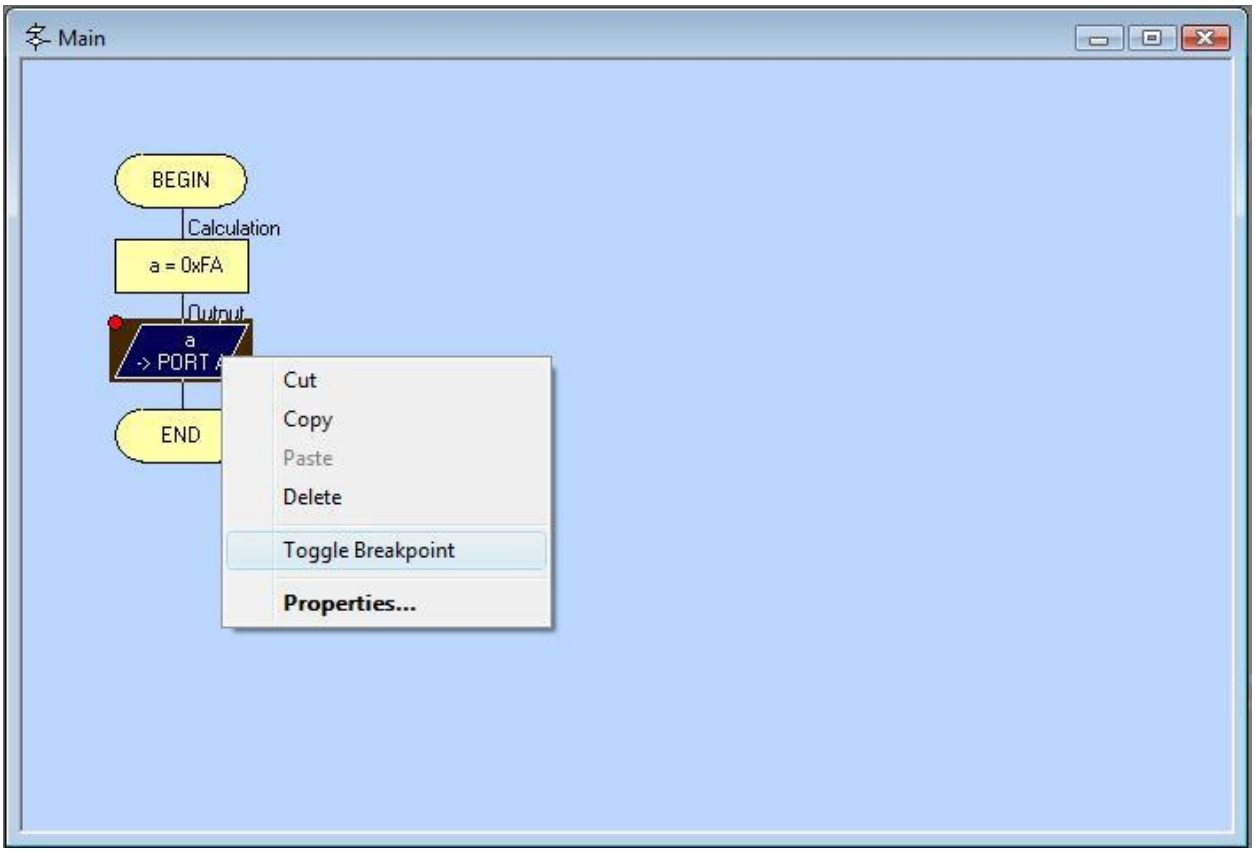


Рис. 1.37. Задание точки останова

Слева от элемента появится точка, точка останова программы, и, когда программа дойдет до этого элемента, отладчик остановит выполнение программы.

Точек останова можно задать несколько в тех местах программы, где вас интересуют значения переменных. Они отображены в окне переменных. Второе окно показывает состояние стека, если в вашей программе есть подпрограммы.

Чтобы удалить точку останова можно повторить операцию задания точки останова. Повторное нажатие на команду **Toggle Breakpoint** снимет точку останова. Чтобы снять все точки останова, достаточно воспользоваться разделом **Edit** основного меню, выбрав пункт **Clear All Breakpoints**.

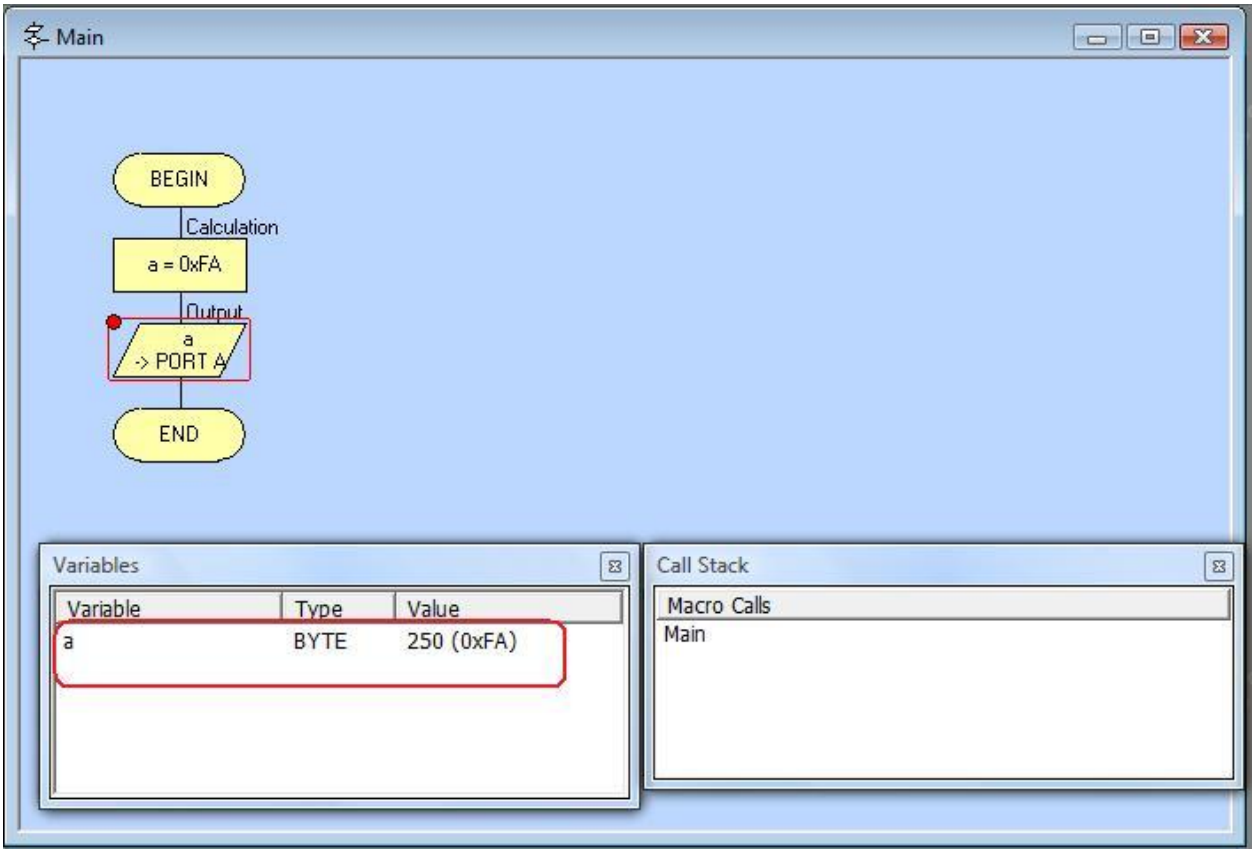


Рис. 1.38. Значение переменной *a*, отображаемое в точке останова

Другой способ увидеть, как меняются переменные, это использовать пошаговый режим отладки. Следом за кнопкой ***Stop*** на инструментальной панели расположена кнопка ***Step Into*** (шаг внутрь). При пошаговом выполнении программы значения переменных отображаются так же, как показано выше. Кнопка ***Step Over*** позволяет «перешагнуть» через выполнение, например, долго выполняемых команд.

Следующая группа относится к трансляции программы.

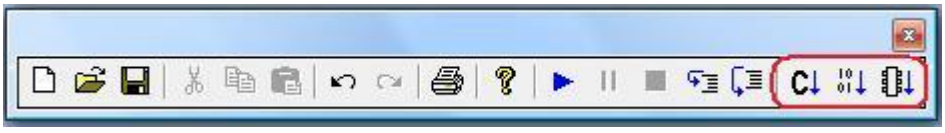


Рис. 1.39. Группа команд трансляции программы

Первая кнопка позволяет транслировать программу в код на языке Си. Полученный код можно увидеть во встроенном текстовом редакторе. Для этого достаточно использовать раздел основного меню **Chip**, где есть пункт **View C**. Если трансляция прошла удачно, то появится сообщение:

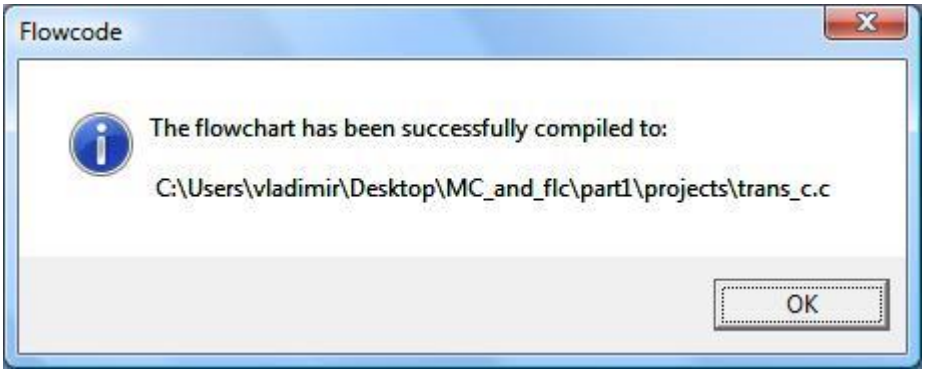


Рис. 1.40. Сообщение об удачной трансляции программы

И полученный код можно открыть, можно копировать и использовать как код на языке Си, о чем мы поговорим позже.

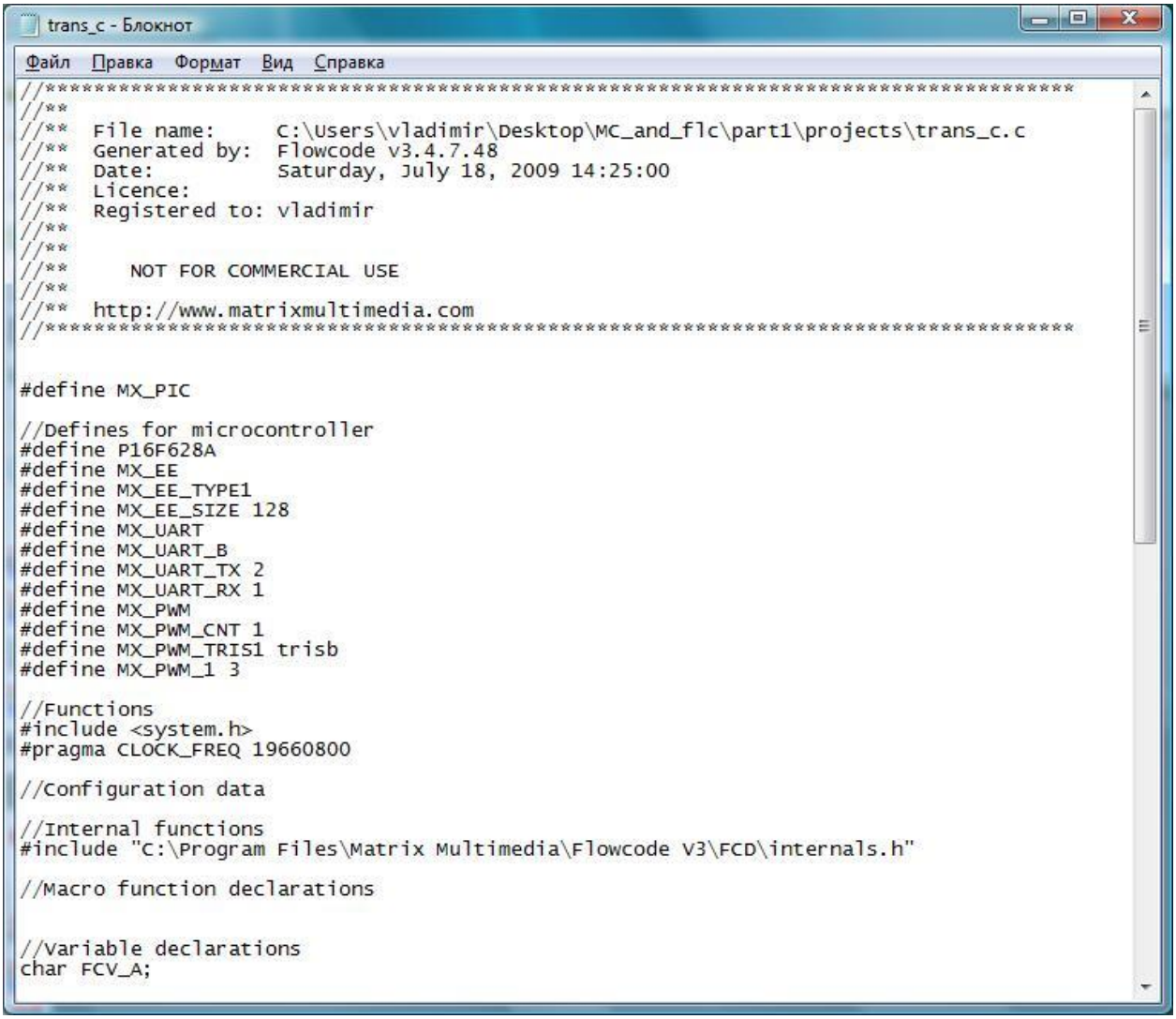


Рис. 1.41. Полученный после трансляции код

Следующая кнопка позволяет транслировать программу в hex-файл для загрузки в микросхему. Если сразу нажать эту кнопку, то программа будет сначала оттранслирована на Си, затем на

ассемблер (вы можете посмотреть и этот код, используя раздел **View ASM** пункта **Chip** основного меню), и только после этого будет получен hex-файл.

Последняя кнопка позволяет вам загрузить полученный hex-файл с помощью программатора, который работает с программой, в микросхему. Если у вас нет такого программатора, то вы включите программу обслуживания программатора, который у вас есть, откроете полученный hex-файл, настроите, если это нужно, слово конфигурации (фузы) и загрузите программу в микроконтроллер.

Тот факт, что программа FlowCode создает код на языке Си и ассемблере, может оказать поддержку при изучении этих языков, о чем мы поговорим позже. Все файлы, на языке Си, ассемблере и hex-файл вы найдете в папке, где сохранили свой проект.

Использование программного компонента Calculation

К использованию этого программного компонента приходится прибегать достаточно часто. Вот некоторые операции, поддерживаемые этим компонентом:

( )	- Скобки
= <>	- Равно, НЕ равно
+ - * / MOD	- Прибавить, вычесть, умножить, разделить, модуль
< <= > >=	- Меньше, чем; меньше или равно; больше, чем; больше или равно
>> <<	- Сдвиг вправо, сдвиг влево
NOT AND OR XOR	- NOT(инверсия), И, ИЛИ, Исключающее ИЛИ

и математические функции:

abs(), round(), floor(), ceil()	- абсолютное значение и округление
fround(,)	- округление числа с плавающей точкой
mod(), fmod()	- модуль целого и числа с плавающей точкой
sqrt(), cbrt()	- квадратный и кубический корни
log(), log10()	- логарифмы (с основанием e и 10)
exp(), pow(,)	- функции экспоненты и степенная
sin(), cos(), tan()	- тригонометрические функции
asin(), acos(), atan(), atan2()	- инверсия тригонометрических функций
sinh(), cosh(), tanh()	- гиперболические функции
asinh(), acosh(), atanh()	- инверсия гиперболических функций
fact()	- факториал
rand()	- случайные числа

isnan(), isinf() - проверка, число ли это, и бесконечности

Увидеть это и многое другое можно в разделе подсказки **Help** основного меню программы FlowCode.

Знакомство с программированием в FlowCode

Простые обучающие программы

Рассмотрим обучающие примеры, которые были получены вместе с программой.

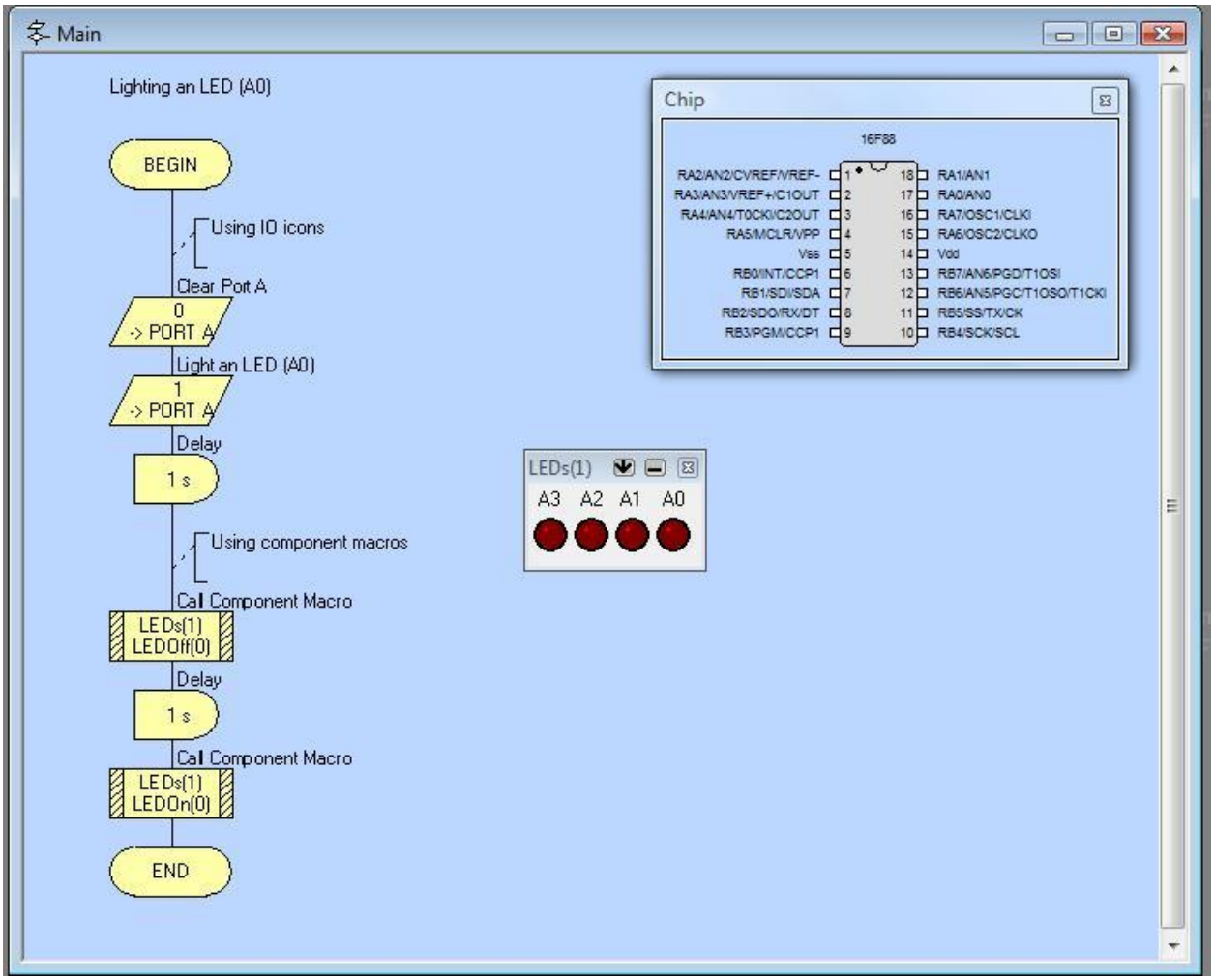


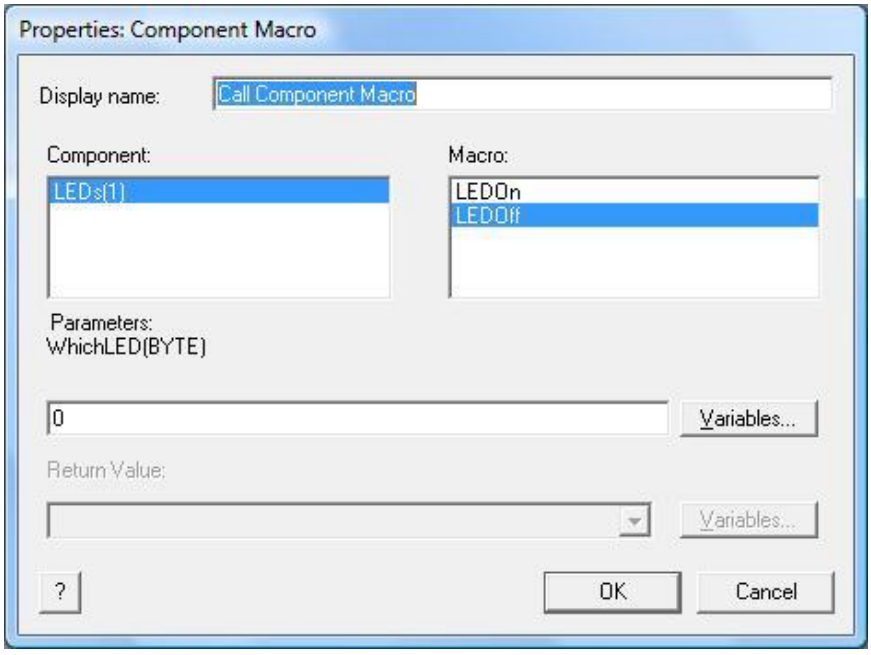
Рис. 2.1. Первый пример из папки Examples

В этом примере для отладки использована линейка светодиодов. Если программные компоненты нужно «подцепить» мышкой и перенести в рабочее поле в нужное место, то для дополнительных компонентов достаточно нажать соответствующую кнопку на инструментальной панели дополнительных компонентов.

Рис. 2.2. Инструментальная панель дополнительных компонентов, отмечены светодиоды

Как в обычном разговорном языке можно написать заявление, а можно написать сонет, так в программе FlowCode можно создать простую программу, а можно очень сложную. При создании программы можно использовать для одной и той же цели разные программные компоненты, что и показывает первый урок по созданию программы, зажигающей светодиод.

Верхняя часть программы использует для этой цели программный компонент **Output**. А нижняя использует компонент, о котором мы не говорили, когда шла речь об инструментальной панели программных компонентов. Этот элемент называется **Component Macro**. Все элементы панели дополнительных компонентов имеют соответствующие подпрограммы, которые уже написаны, имеют необходимый набор функций и свойств. Связь между дополнительными компонентами, как линейкой светодиодов, и их свойствами осуществляет **Component Macro**. Откройте двойным щелчком по этому компоненту в первом примере диалоговое окно его свойств.



В левом окне отображено, какой компонент, а их может быть несколько, используется.

В правом окне выбор доступных функций. В данном случае *LEDOn* (включить) и *LEDOff* (выключить).

Рис. 2.3. Диалоговое окно свойств линейки светодиодов

В качестве параметра (*Parameters*) вы можете указать, какой из светодиодов следует использовать (на рисунке нулевой).

Обратите внимание, что комментарий (как программный компонент) добавляется к «пути следования» программы. Когда вы, подцепив мышкой программный компонент, перемещаете курсор на рабочее поле, он, курсор, принимает вид того программного компонента, который вы добавляете, а слева появляется стрелка, указывающая в то место программы, куда будет вставлен компонент.

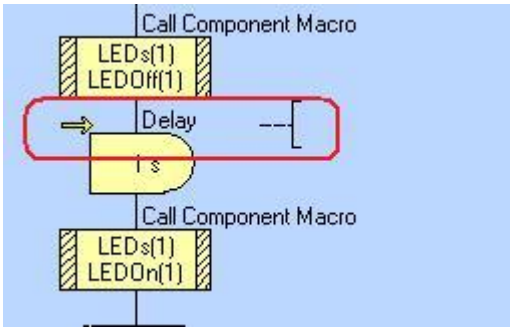
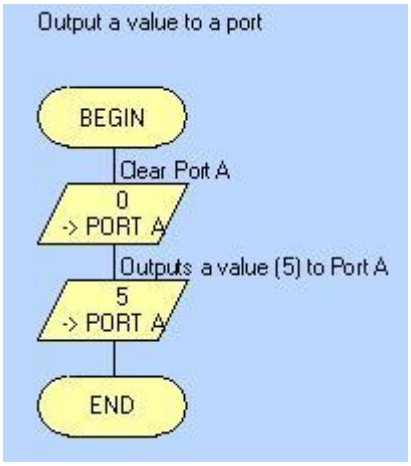


Рис. 2.4. Добавление комментария к программе

Следующий пример (TUT_02) показывает, как вывести число в порт.



В следующем примере, TUT_03, показано, как можно управлять выводами порта «побитово».

Очень полезно открыть диалоговые окна и посмотреть, как организован побитовый вывод, и по шагам пройти всю программу.

Рис. 2.5. Вывод числа 5 в порт A

Пример TUT_05 показывает основные операции: присваивание, сложение, вычитание, деление и умножение, выполняемые с помощью программного компонента **Calculation**.

А пример TUT_06 показывает, как можно «напрямую» связать входы и выходы портов, используя программные компоненты **Input** и **Output** и дополнительные компоненты.

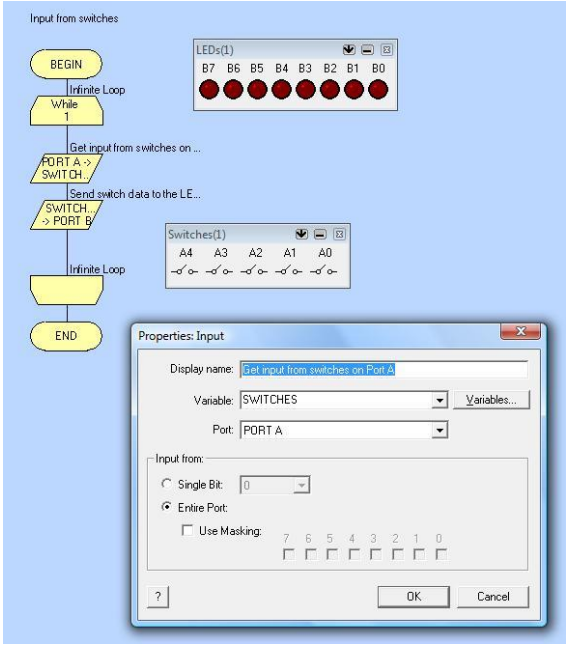


Рис. 2.6. Использование «прямой» связи выключателей и светодиодов

Рис. 2.7. Выключатели на инструментальной панели дополнительных компонентов

Создав одну переменную *SWITCHES*, к которой будет «привязан» компонент **Input**, можно вывести эту переменную, с помощью компонента **Output** в порт.

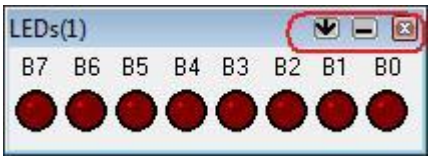


Рис. 2.8. Кнопки управления на панели дополнительных компонентов

Попутно отметим, что компоненты инструментальной панели дополнительных компонентов, имеют три кнопки, первая из которых открывает выпадающее меню, вторая сворачивает компонент, а третья «убирает» его с рабочего поля. Выпадающее меню линейки светодиодов выглядит следующим образом:

Рис. 2.9. Выпадающее меню управления свойствами дополнительных компонентов

И очень интересно диалоговое окно свойств (*Properties*).

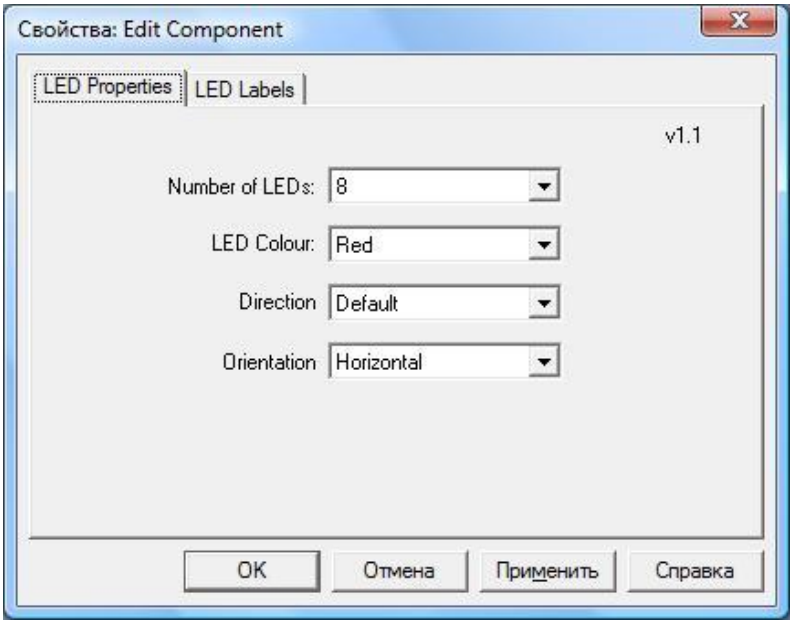


Рис. 2.10. Диалоговое окно свойств линейки светодиодов

В первом окне можно выбрать количество светодиодов, в следующем цвет, дальше можно изменить направление отсчета слева-направо или наоборот, и, наконец, последнее в свойствах, что можно изменить, это ориентация, горизонтальная или вертикальная.

Вторая закладка LED Labels позволяет добавить подписи к элементам.

Если, скажем, вы хотите, чтобы микроконтроллер «покомандовал» двумя светофорами на перекрестке, то можно добавить нужное число линеек светодиодов, выбрать цвет и подпись для каждого и проверить работу светофора.

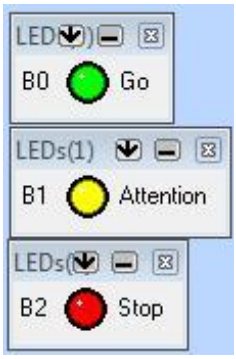


Рис. 2.11. Создание светофора из нескольких линеек светодиодов

Выключатели управляются (нажимаются) мышкой. Иногда, запустив программу, достаточно «пощелкать» выключателем, но бывает так, что мышкой нужно управлять выключателем и еще чем-то одновременно. В этом случае можно изменить свойства выключателей в их диалоговом окне.

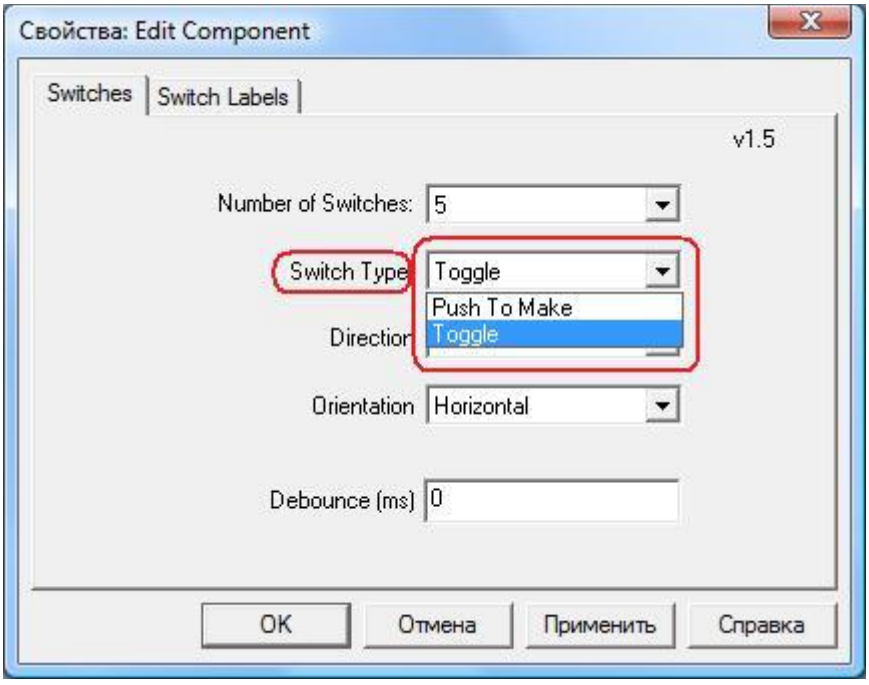


Рис. 2.12. Выбор характера включения выключателей

Пример TUT_08 показывает, как применить маску для выделения нужных входных битов.

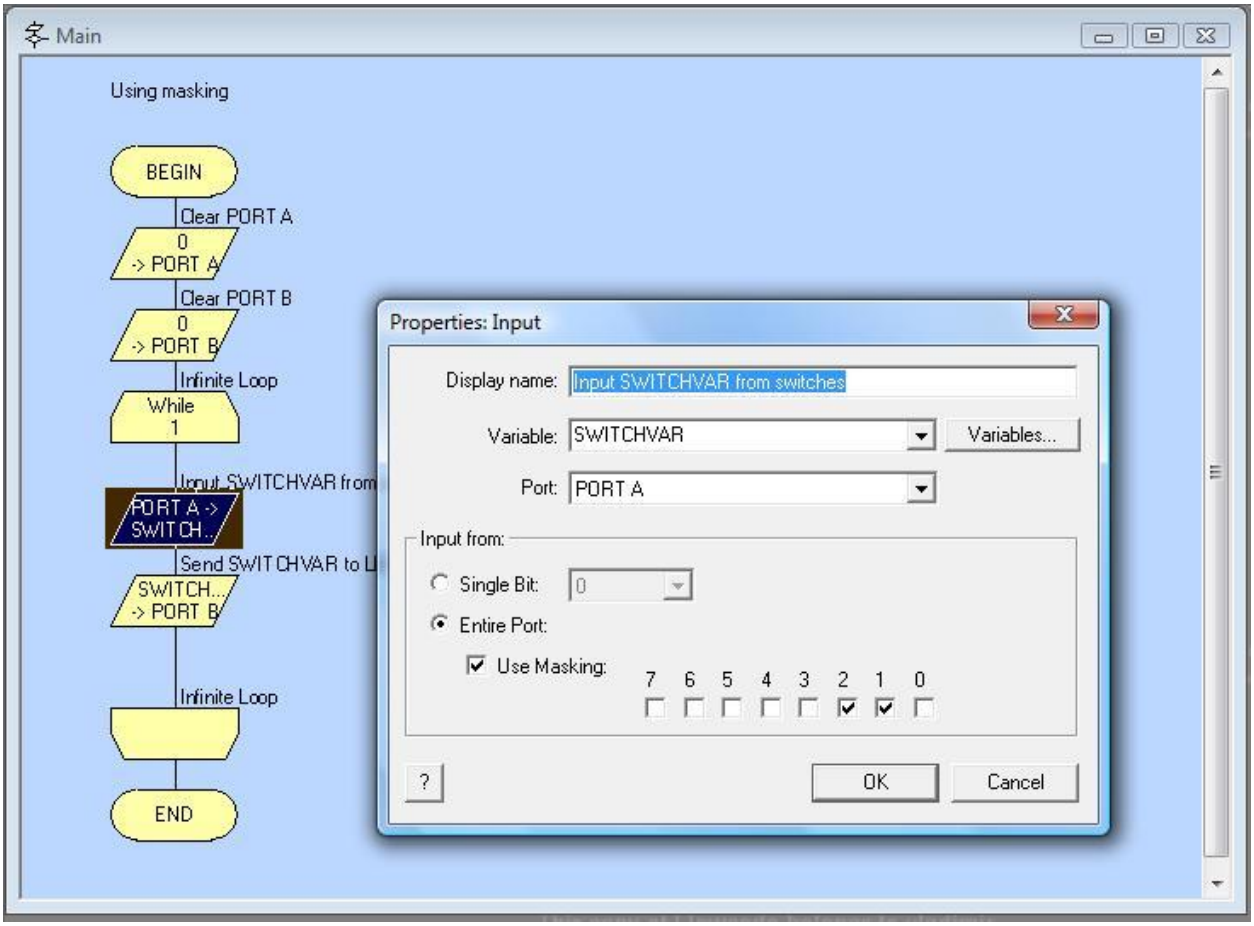
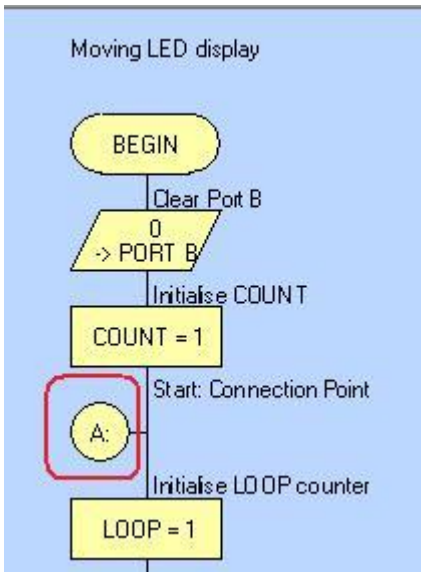


Рис. 2.13. Применение маски

Ряд примеров, показывающих, как можно создать счетчик, как использовать циклы, как применить компонент **Calculation** для сдвига содержимого регистра вправо или влево, дают возможность приобрести навыки программирования, значительно расширяющие кругозор, вдобавок эти программы могут впоследствии использоваться в качестве подпрограмм.

Пример TUT_15 показывает, зачем нужен такой программный компонент, как пара точек **Connection Point**.

Метки в программе, особенно на ассемблере, применяются достаточно часто. Первая из пары точек **Connection Point** и есть метка, а вторая – переход к метке.



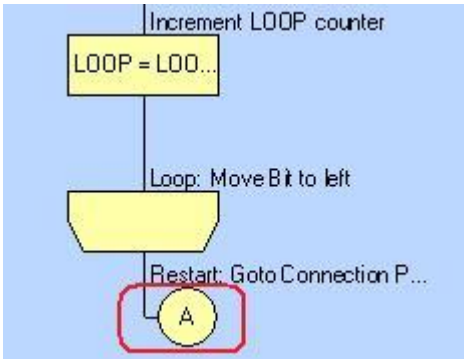


Рис. 2.15. Точки соединения

В диалоговом окне свойств второй точки соединения указывается переход:

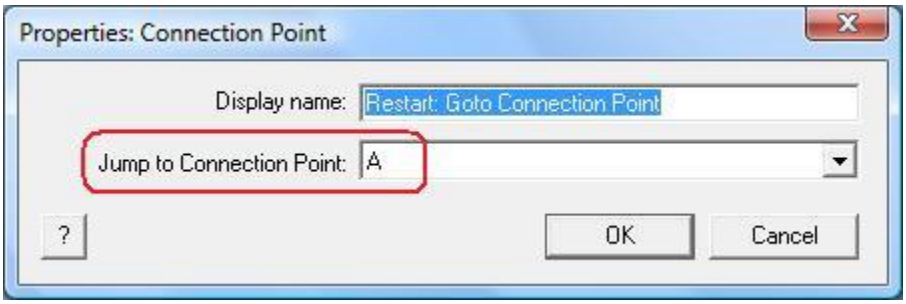


Рис. 2.16. Диалоговое окно свойств точки соединения

Еще один пример полезного использования программного компонента **Component Macro** приведен в программе TUT_18.

Этот компонент используется тогда, когда в программе есть соответствующий дополнительный компонент.

Рис. 2.17. Семисегментный индикатор на инструментальной панели

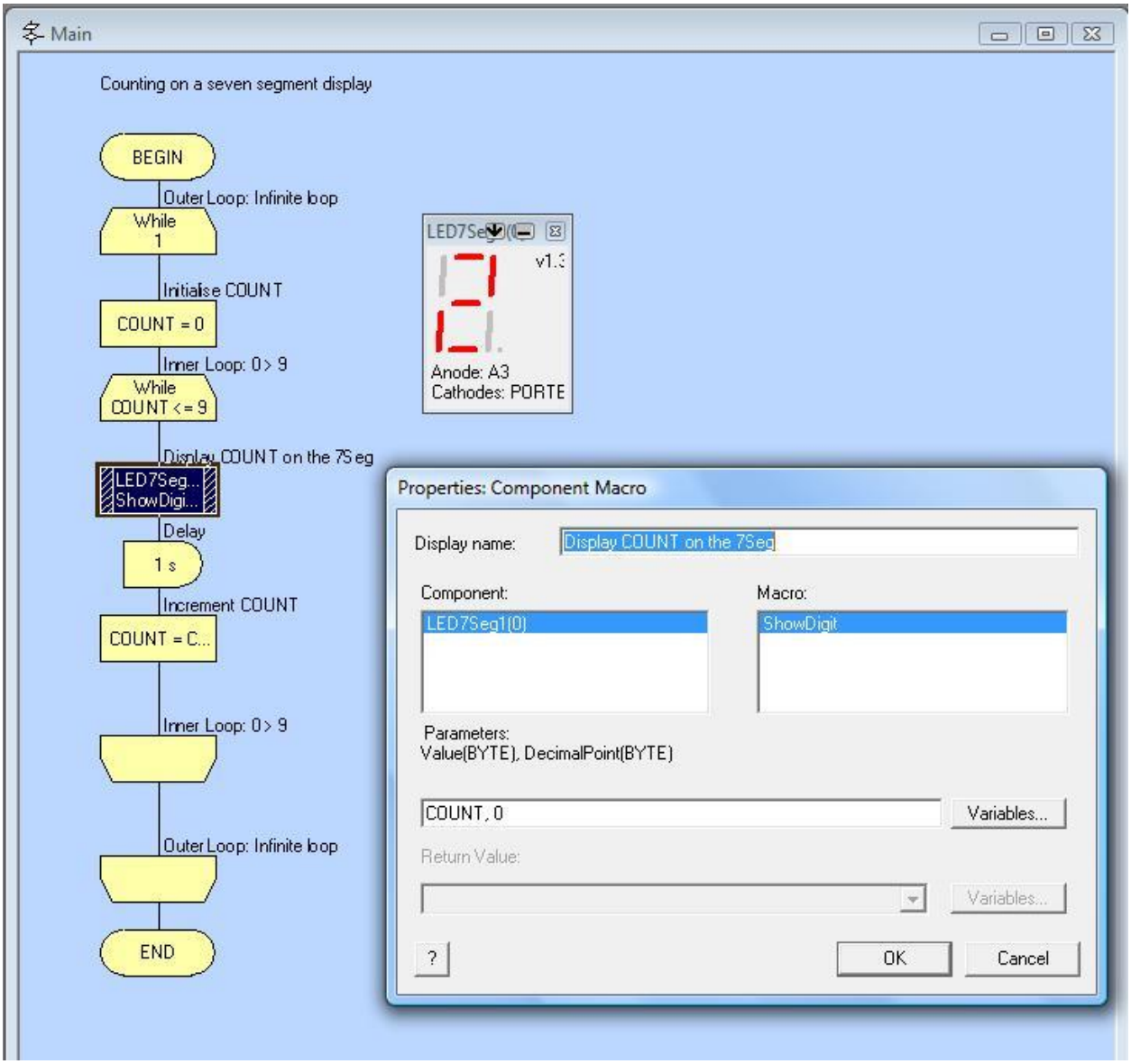


Рис. 2.18. Использование индикатора

В качестве параметра (*Parameters*) используется переменная COUNT. Это позволяет вывести на индикатор любую цифру (от 0 до 9), которое присвоено переменной.

Примеры более сложных приемов программирования

Прерывание используется при программировании достаточно часто. Необходимость в этом элементе программирования возникла в связи с обслуживанием процессором внешних устройств, работающих гораздо медленнее процессора. Обслуживание таких внешних устройств возможно по опросу, когда процессор, передав часть задания, постоянно опрашивает внешнее устройство, готово ли оно продолжить работу, и ждет. Ожидание не позволяет процессору заняться чем-то полезным, в результате бесцельно пропадает процессорное время.

Ситуацию исправляет механизм прерывания. Передав часть задания внешнему устройству, процессор разрешает ему прервать свою работу, когда устройство будет готово продолжить выполнение задания. Пока внешнее устройство «трудится», процессор может заняться выполнением другой задачи. А, получив запрос на прерывание, поместить в стек, специально отведенную для этой цели область памяти, адрес программы, где его застал запрос на прерывание, и данные, с которыми процессор работал. После этого процессор может перейти к подпрограмме обслуживания прерывания, взяв из стека необходимые данные для продолжения работы с внешним устройством.

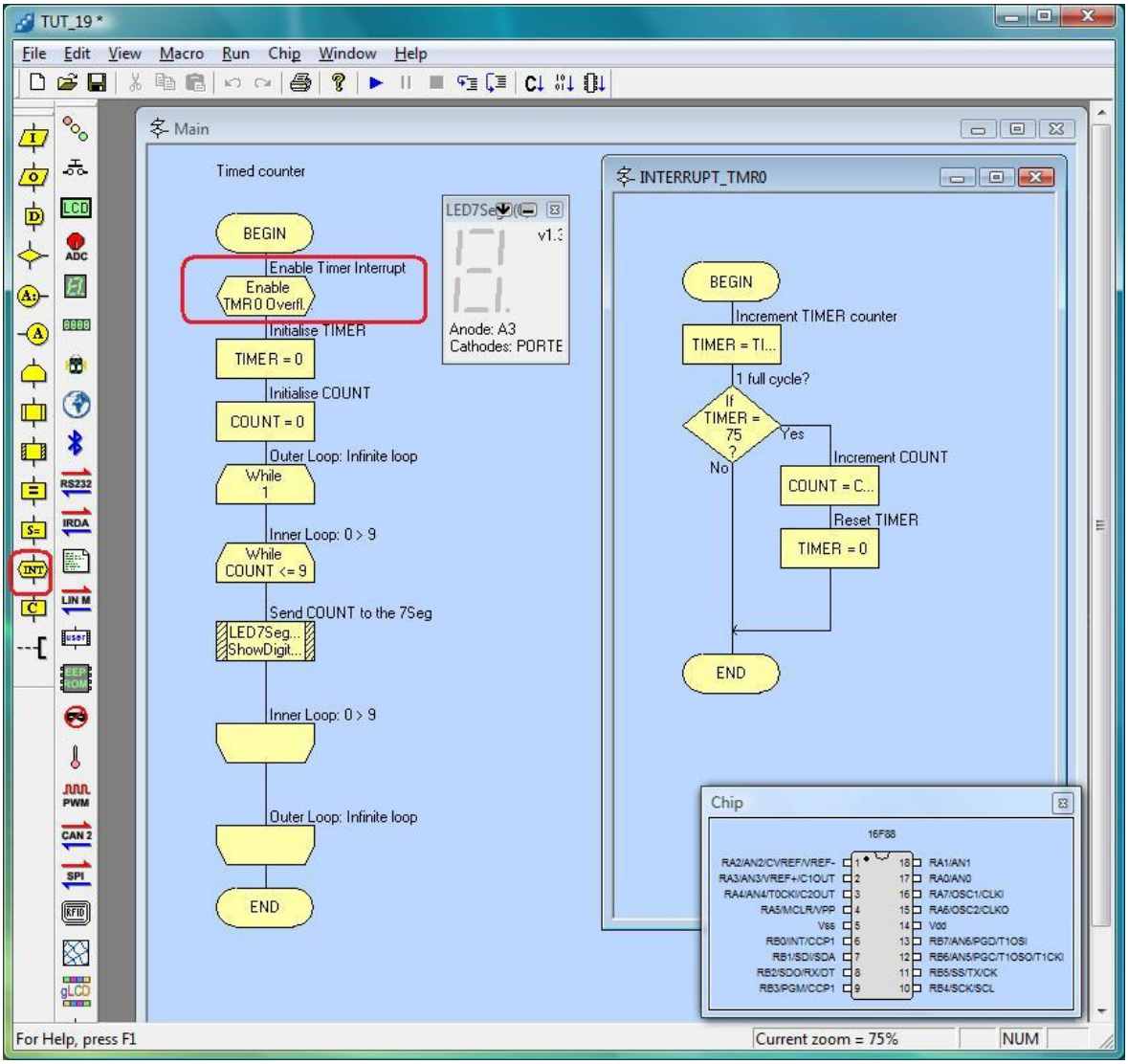


Рис. 2.19. Использование программного компонента **Interrupt** (прерывание)

На рисунке кроме основного окна программы (Main) видно второе окно подпрограммы INTERRUPT_TMR0. Кроме прерываний по таймеру, как в показанной выше программе, диалоговое окно свойств программного компонента **Interrupt** дает возможность выбрать из выпадающего списка еще несколько видов прерываний. Используемые прерывания могут зависеть от модели микроконтроллера. Так вместо опроса в цикле входных выводов, к которым может быть подключена клавиатура, можно использовать прерывание по изменению этих входов (RB Port Change).

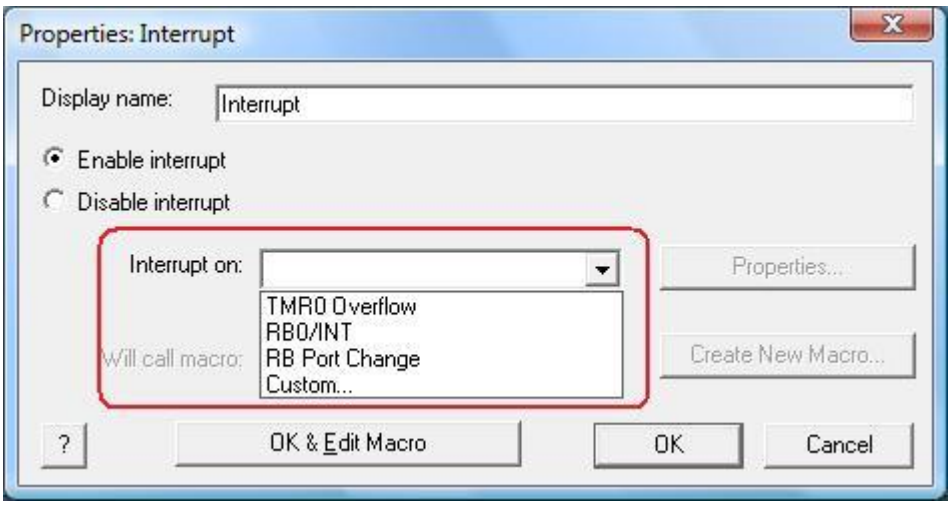


Рис. 2.20. Допустимые виды прерывания для модели PIC16F88

На рисунке выше выбрано прерывание по таймеру *TMR0 Overflow* (переполнение таймера 0). Если нажать кнопку ***Properties*** рядом с окном выбора вида прерывания, то можно задать ряд свойств прерывания. Все свойства выбираются из выпадающих списков.

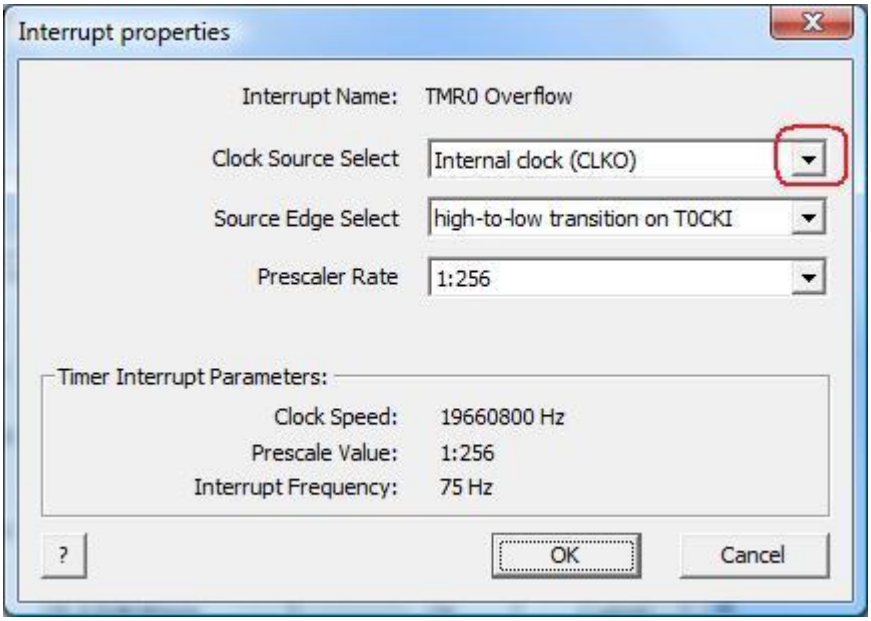


Рис. 2.21. Свойства прерывания, заданные для программы выше

Можно выбрать источник тактового сигнала (Clock Source Select), выбрать каким из двух фронтов, передним или задним, импульса будет инициировано прерывание, выбрать предварительное деление тактовой частоты для отсчета времени. Ниже показана полученная частота при заданной частоте тактового генератора и выбранного коэффициента деления.

После задания свойств прерывания необходимо написать подпрограмму обслуживания прерывания. Это обычная, в сущности, подпрограмма, и вначале нужно создать макрос с помощью кнопки ***Create New Macro*** (рис. 2.20), затем с помощью кнопки ***OK & Edit Macro*** открыть новое рабочее поле для подпрограммы и начать писать подпрограмму.

Рассмотрим подпрограмму вышеприведенного примера.

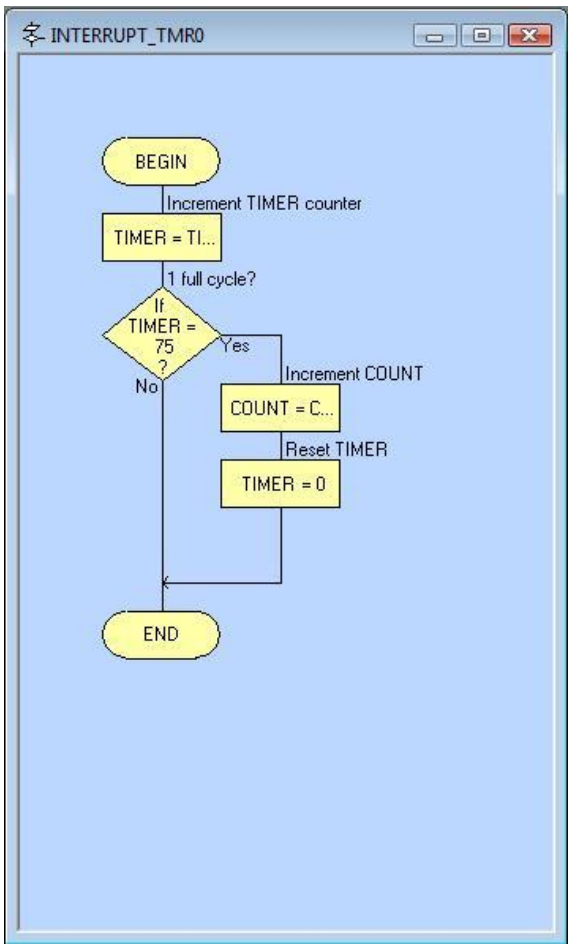


Рис. 2.22. Подпрограмма обслуживания прерывания по таймеру

Откроем диалоговое окно первого элемента программы *Increment TIMER counter* (наращивание счетчика таймера).

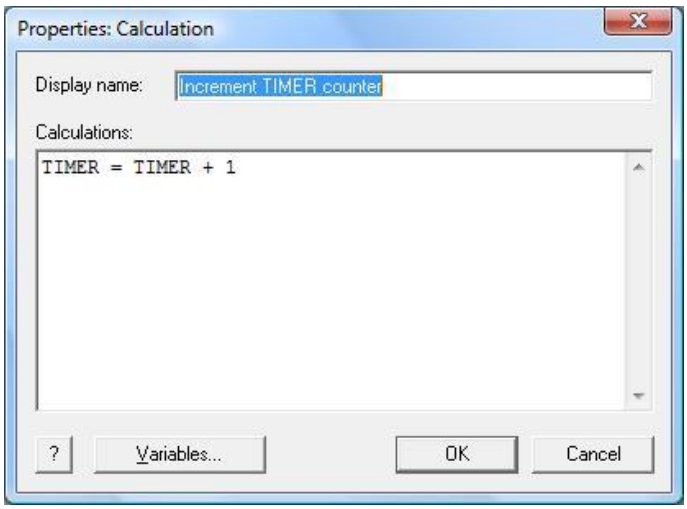


Рис. 2.23. Первый элемент подпрограммы

Как видно, это программный компонент **Calculation**, где значение переменной *TIMER* увеличивается на единицу. Затем следует проверка значения переменной (программный компонент **Decision** – ветвление).

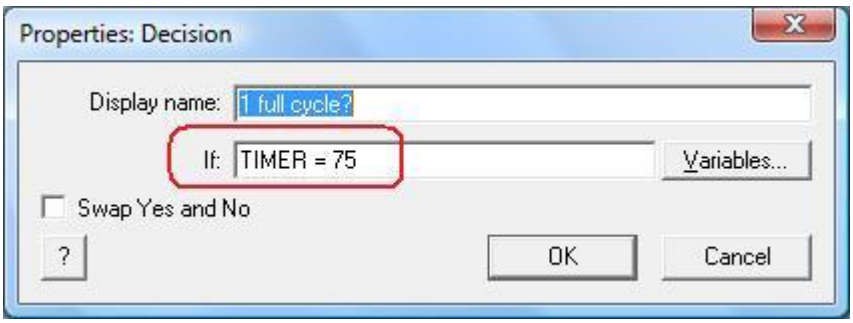


Рис. 2.24. Проверка значения переменной

Если значение достигло заданной величины, то с помощью двух программных компонентов **Calculation** увеличивается на единицу значение переменной *COUNT* и обнуляется переменная *TIMER*, обуславливая завершение одного полного цикла, как и обозначено выше в имени ветвления *1 full cycle*.

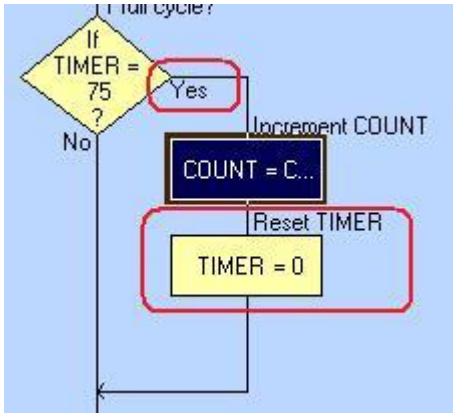


Рис. 2.25. Ветвление программы

Если запустить эту программу, используя кнопку ***Run*** инструментальной панели или команду **Go/Continue** раздела **Run** основного меню, то… можно ничего интересного и не увидеть. Достаточно долго ничего не меняется. Это достаточно часто встречающаяся ситуация при отладке программы. Внесем некоторые изменения (только для отладки): уменьшим значение, скажем, до 5 в условии ветвления подпрограммы. И поставим точку останова в подпрограмме на элемент *Increment TIMER counter*.

Если теперь запустить программу, то она будет останавливаться каждый раз, когда начинается работа подпрограммы. Перезапуская программу (без остановки кнопкой ***Stop*** при каждом прогоне) несколько раз, можно увидеть изменения.

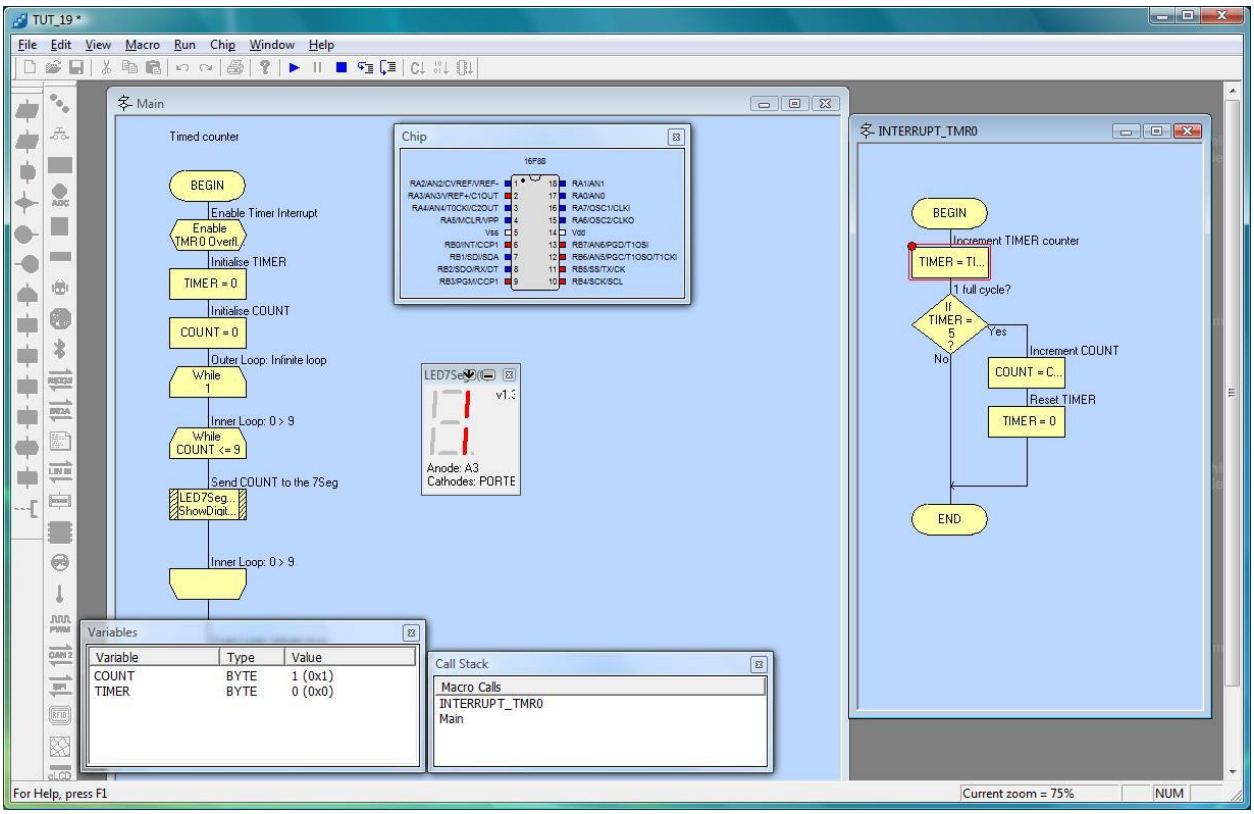


Рис. 2.26. Отладочный запуск программы

Прерывание полезный прием, но не следует забывать, что каждое прерывание использует стек, а стек имеет ограниченное пространство в памяти – обилие не оправданных прерываний может приводить к переполнению стека, вызывая сбой программы.

В примере TUT_20 используется семисегментный индикатор с четырьмя секциями. Действительно, если вам захочется сделать часы или таймер на базе микроконтроллера, то вам нужно будет отображать четыре цифры. Понятно, что когда мы используем одну секцию семисегментного индикатора, выводов порта достаточно для обслуживания индикатора. Но как быть, когда нам нужно в четыре раза больше выводов. Не менять же микроконтроллер.

В первую очередь, где на инструментальной панели дополнительных компонентов находится подобный индикатор?

Рис. 2.27. Семисегментный индикатор с четырьмя секциями

Открыв пример TUT_20 обратите внимание на то, как подключен индикатор. Для этого используйте кнопку, открывающую выпадающее меню.

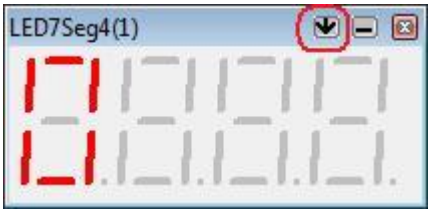


Рис. 2.28. Кнопка выпадающего меню дополнительных компонентов

А в выпадающем меню выберите пункт **Component Connections**.

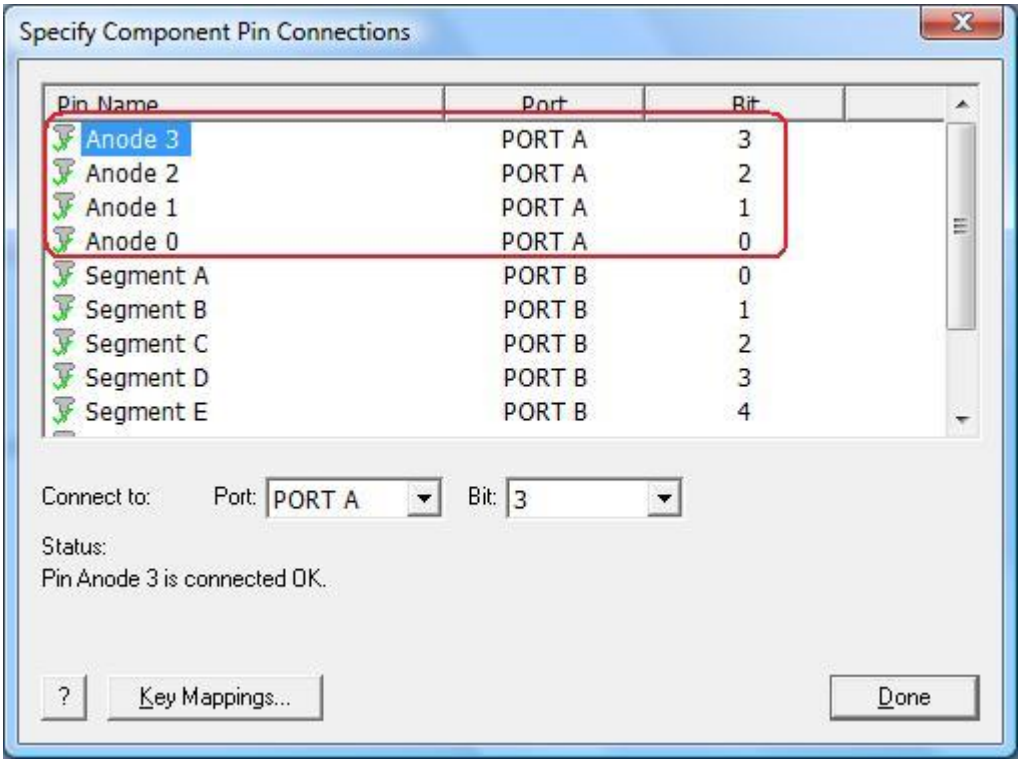


Рис. 2.29. Подключение индикатора к портам контроллера

Таким образом, сегменты всех секций подключены к выводам одного порта, а аноды этих сегментов подключены к выводам другого порта. Пример TUT_20 показывает, как использовать динамическую индикацию.

Развитие индикаторов со временем привело к появлению не слишком дорогих даже для любителей жидкокристаллических дисплеев. Теперь микроконтроллер, снабженный подобным

дисплеем, может стать основой целой серии полезных устройств. На дисплей можно вывести, например, фразу. Как это сделать, показано в примере TUT_21.

Найти дисплей можно на инструментальной панели дополнительных компонентов.

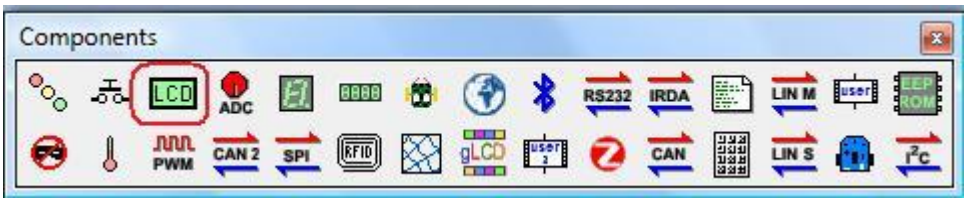


Рис. 2.30. Дисплей на инструментальной панели

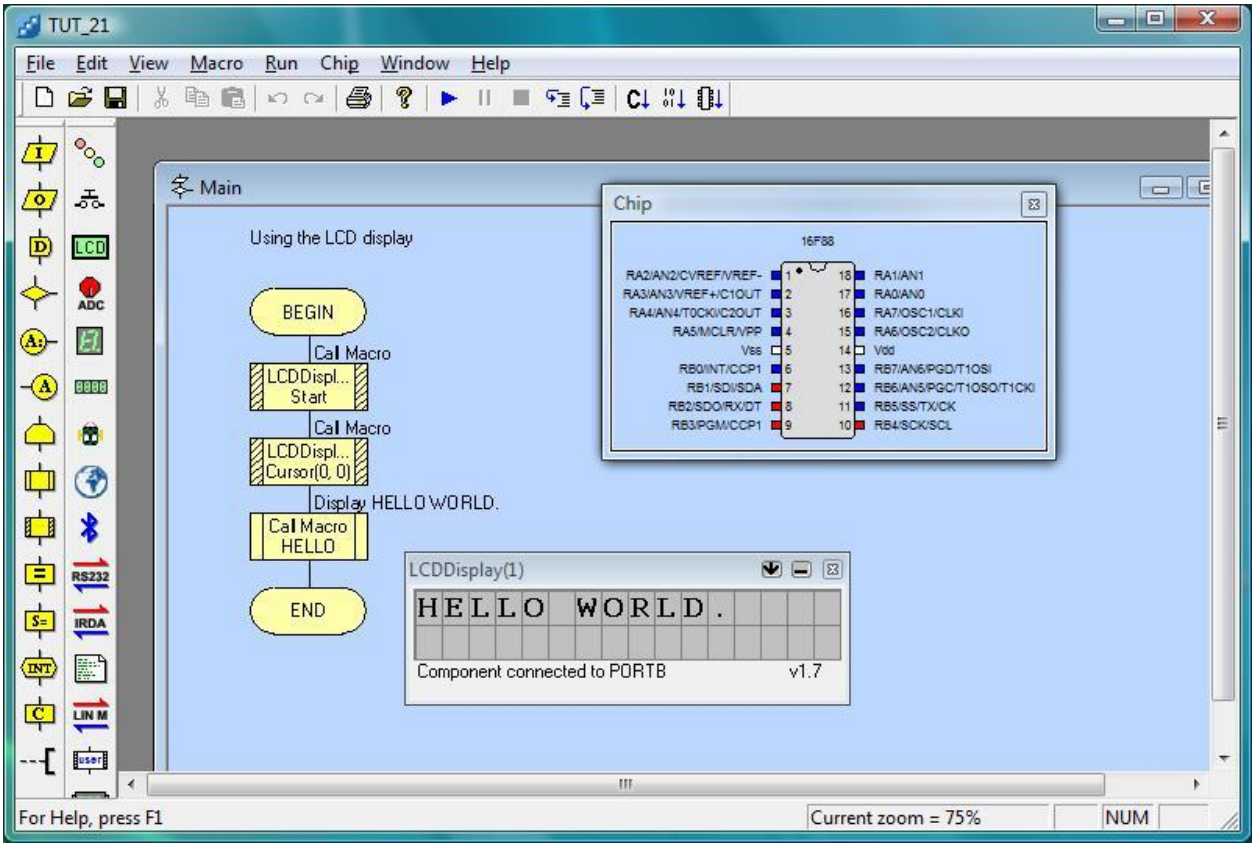


Рис. 2.31. Вывод фразы на дисплей

Если вы откроете свойства дисплея, то можете убедиться, что это не единственный дисплей, доступный для ваших экспериментов. Это семейство дисплеев. И обратите внимание на подключение дисплея.

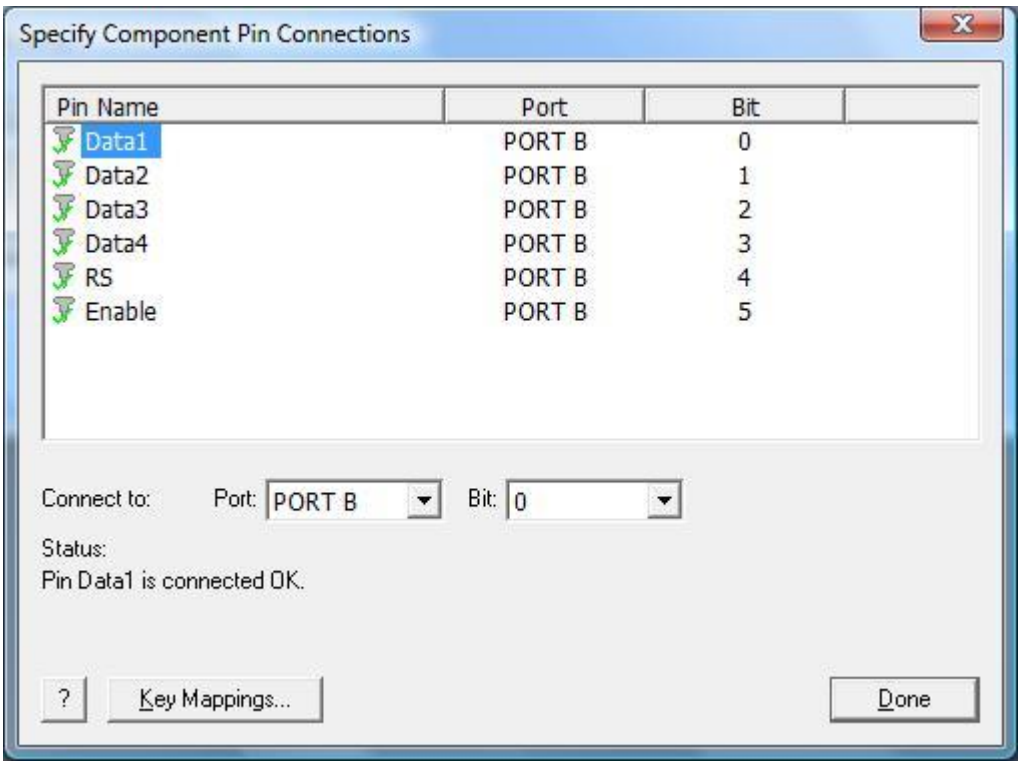


Рис. 2.32. Подключение дисплея к порту контроллера

В программе несколько раз используется макрос, связанный с этим элементом – **Component Macro**. Если вы откроете любой из них, то увидите, сколько встроенных в эту подпрограмму функций есть в вашем распоряжении (для этого нужно щелчком выделить название компонента).

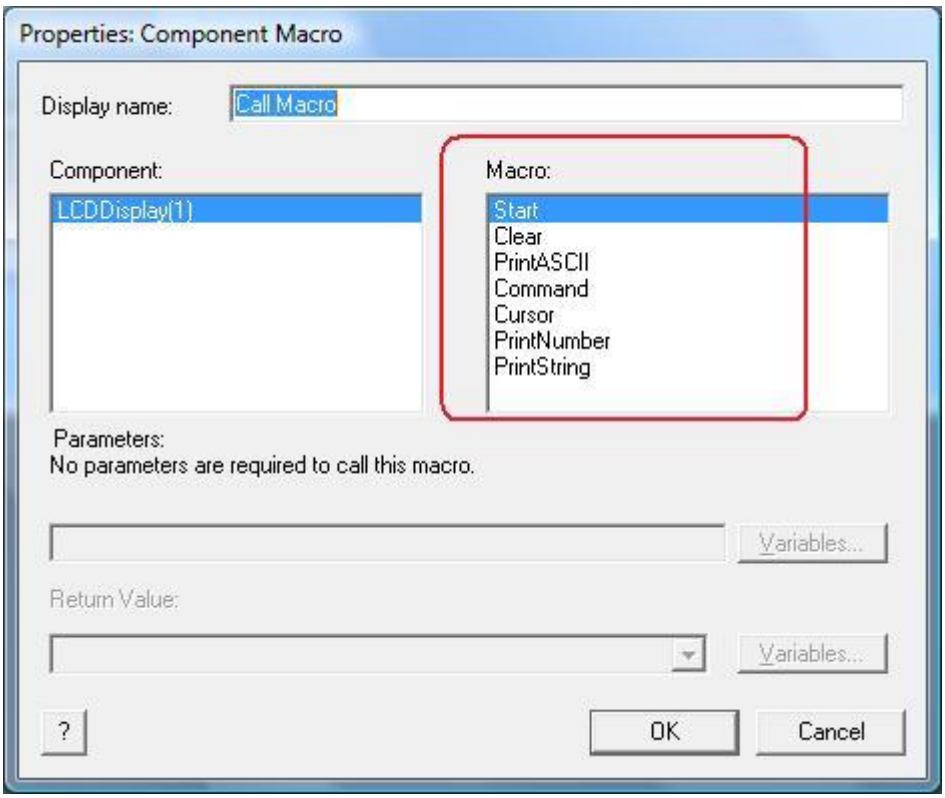


Рис. 2.33. Доступные функции компонента LCDDisplay

Если ваш микроконтроллер имеет встроенный аналого-цифровой преобразователь, АЦП, то вы можете использовать его, совместно с дисплеем, для создания целого ряда автоматических станций, которые могут следить за температурой или влажностью почвы.

Рис. 2.34. АЦП на инструментальной панели

Дополнительный компонент ACD позволяет проверить работу встроенного АЦП, при этом ручка, изображенная на иконке, вращается, если ее «подцепить» мышкой.

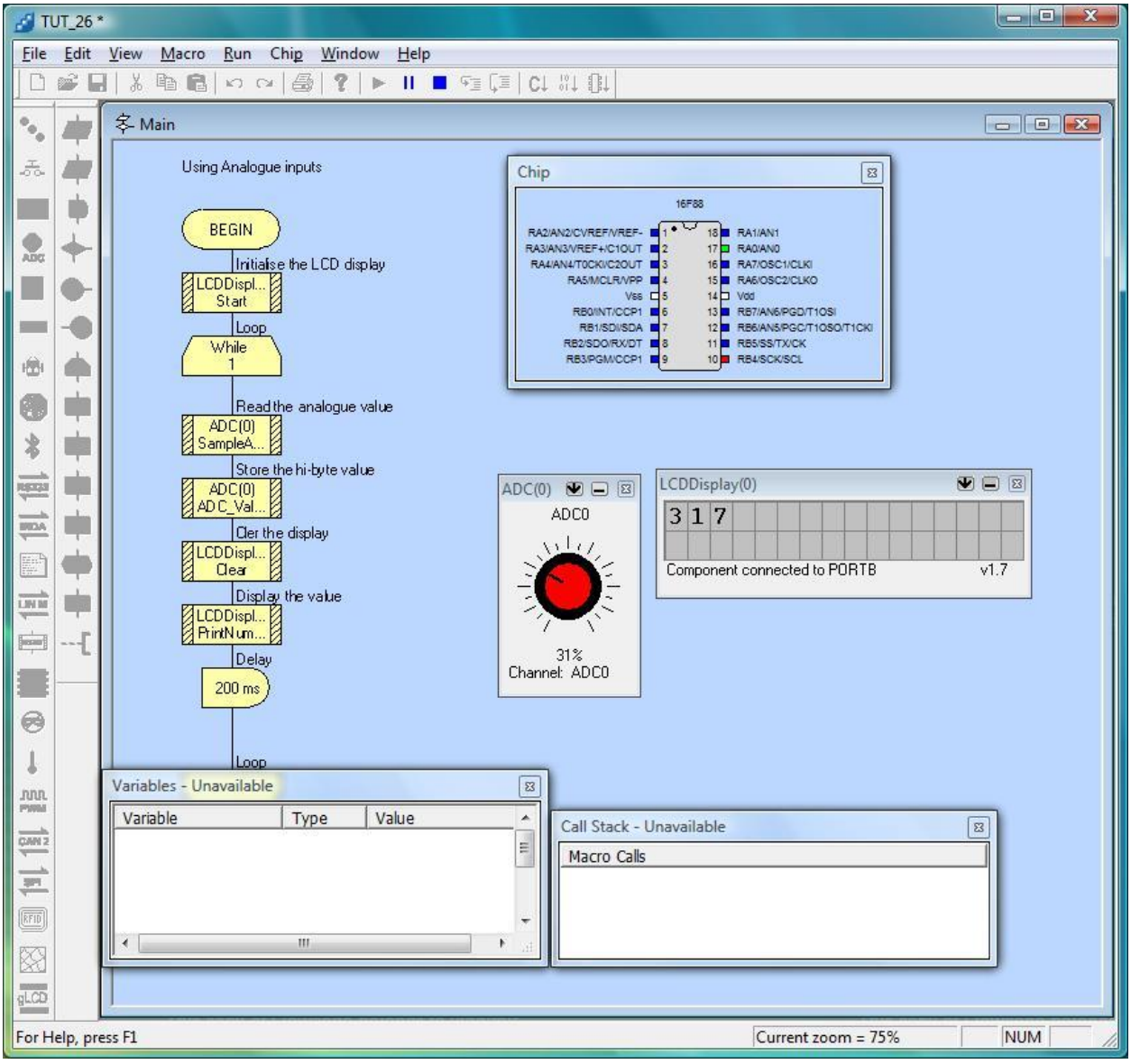


Рис. 2.35. TUT_26 – пример использования АЦП

При этом вы можете использовать программный компонент String Manipulation для создания строк, которые можно вывести на дисплей.

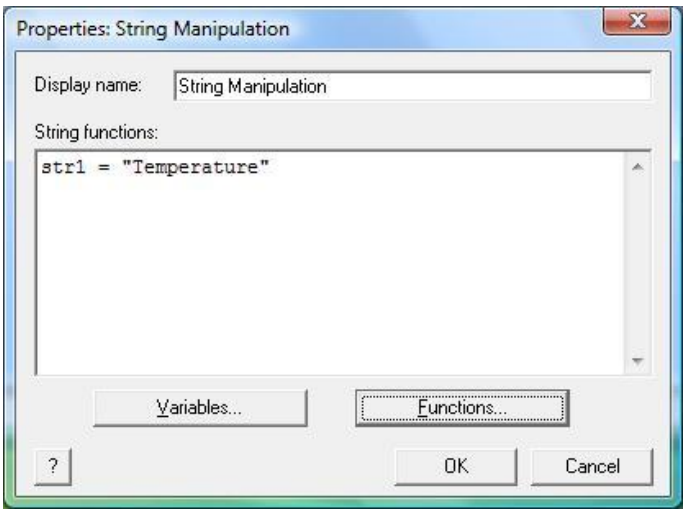


Рис. 2.36. Использование программного компонента String Manipulation

Этот программный компонент имеет ряд встроенных функций для работы со строковыми переменными, которые отображаются, если нажать кнопку ***Functions***. Добавив к программе показанной выше еще несколько программных компонентов, вы можете получить, например, следующее:

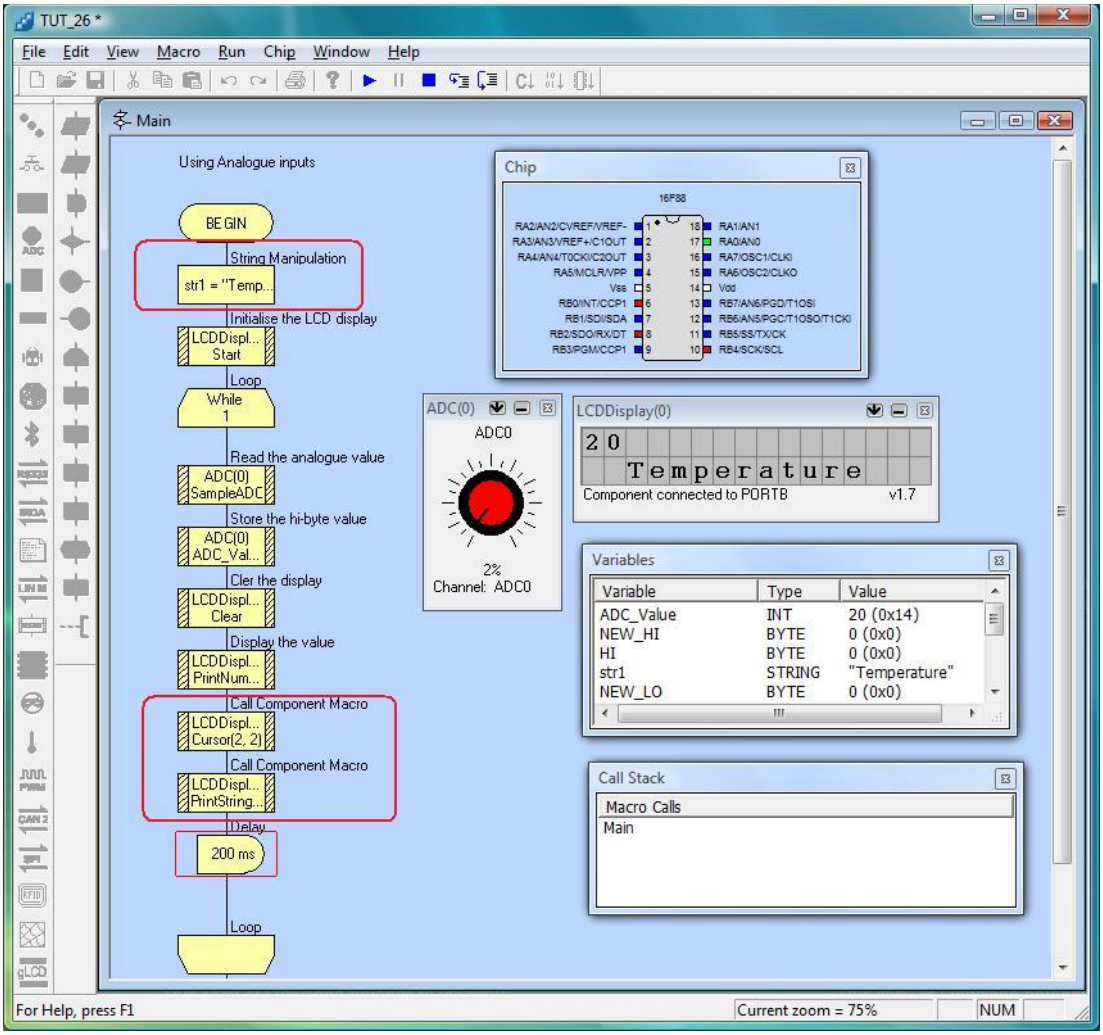


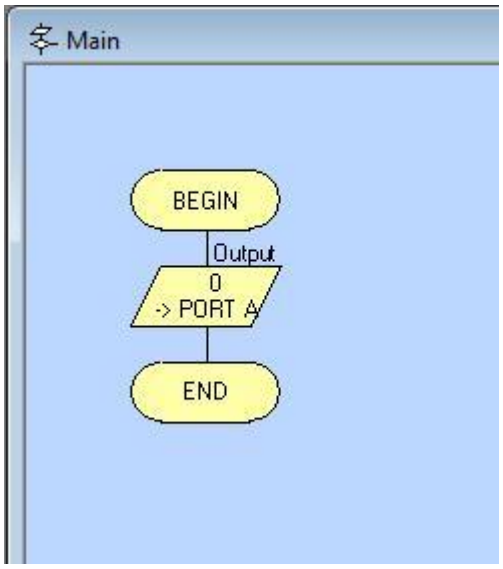
Рис. 2.37. Модификация программы TUT_26

Переход к программированию на языке Си

Программа, если вы ее купите, возможно, полностью удовлетворит все ваши нужды в работе с микроконтроллерами. Но, используя демо-версию, вы очень быстро столкнетесь с ограничениями. Да и ощущение, что не все в ваших руках, не все возможности вы используете, это можно отнести к таким программным компонентам FlowCode, как вставки на Си и ассемблере, должно подвигнуть вас на изучение языка Си и ассемблера. И программа FlowCode хороший помощник в этом.

Первые шаги

Сейчас нам нужен только один программный элемент – **Output**. «Подцепив» его мышкой, перенесем между элементами *BEGIN-END*. И ничего с ним делать не будем. Таким образом, мы все выводы порта А включаем «на выход» и в низкое состояние, то есть в «0».



Оттранслируем программу на язык Си: для этого в программе FlowCode в основном меню выбираем раздел **Chip** и пункт **Compile to C**.

Мы можем посмотреть полученный результат: **Chip-> View C**.

Рис. 3.1. Простейшая программа в FlowCode

Программа, которую мы открыли во встроенном редакторе, уже достаточно длинная (для первого шага). Но нас интересует только основная ее часть:

```
void main()
{
//Initialisation
cmcon = 0x07;
//Output: 0 -> PORT A
trisa = 0x00;
porta = 0;
}
```

main – это начало программы на языке Си. Сама программа находится в фигурных скобках {...}. Каждый оператор завершается точкой с запятой. *void* перед *main* означает, что функция не возвращает значения. Я привел только основную часть программы, удалив все директивы компилятору и часть, относящуюся к прерываниям, их отложим «в долгий ящик».

Мы получили код на языке Си для программы, которая устанавливает выводы порта в «0». Модифицируем эту программу так, чтобы: устанавливать не весь порт, а один вывод порта, и не в «0», а в «1».

Для этого двойным щелчком по программному компоненту **Output** откроем его диалоговое окно:

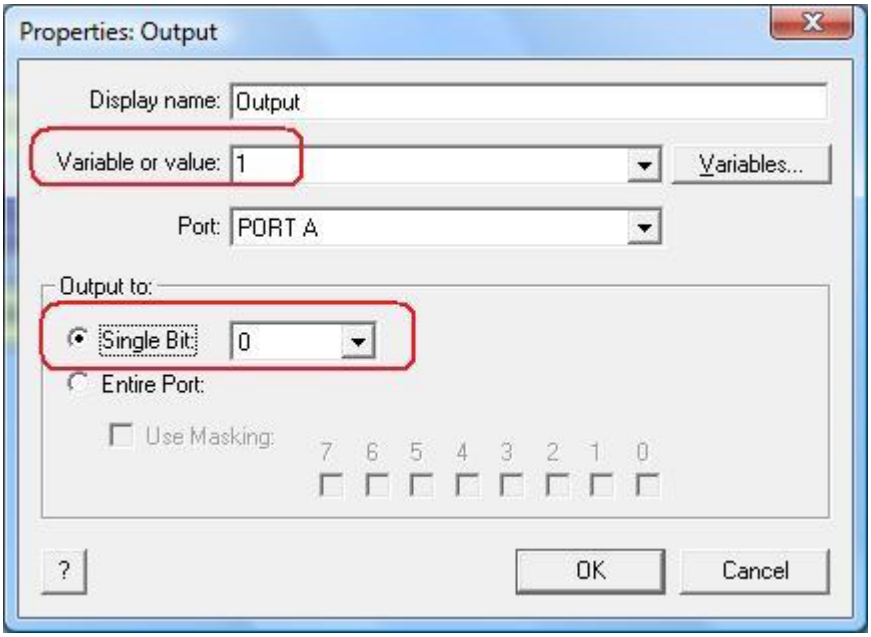


Рис. 3.2. Диалоговое окно программного компонента **Output**

На рисунке отмечены внесенные изменения: вместо 0 в окошке *Variable or value* (переменная или значение) вписана 1; вместо *Entire Port* (весь порт) выбрана опция *Single Bit* (единственный бит) и оставлен вывод 0 порта А (хотя можно было изменить и порт, чуть выше). Нажмем кнопку ***OK*** и сохраним проект в новой папке с новым именем, чтобы не путать с прежним.

Повторим операцию трансляции на язык Си: **Chip->Compile to C...**

Так же выделим только нужную часть кода:

```
void main()
{
//Initialisation
cmcon = 0x07;

//Output: 1 -> A0
trisa = trisa & 0xfe;
if (1)
porta = (porta & 0xfe) | 0x01;
else
porta = porta & 0xfe;
}
```

Вот так выглядит установка одного вывода порта А в «1». Если повторить все установки для одного вывода, но с установкой его в «0» (а не всего порта, как в самом начале), то текст программы будет выглядеть следующим образом:

```
void main()
{
//Initialisation
cmcon = 0x07;

//Output: 0 -> A0
trisa = trisa & 0xfe;
if (0)
porta = (porta & 0xfe) | 0x01;
else
porta = porta & 0xfe;
}
```

В итоге мы располагаем тремя фрагментами текста на языке Си, которые делают базовые для микроконтроллера операции – устанавливают выбранный нами вывод порта в «0» и в «1» или устанавливают весь порт в «0».

Проверим еще один вариант: установим в «1» не нулевой вывод порта А, а первый.

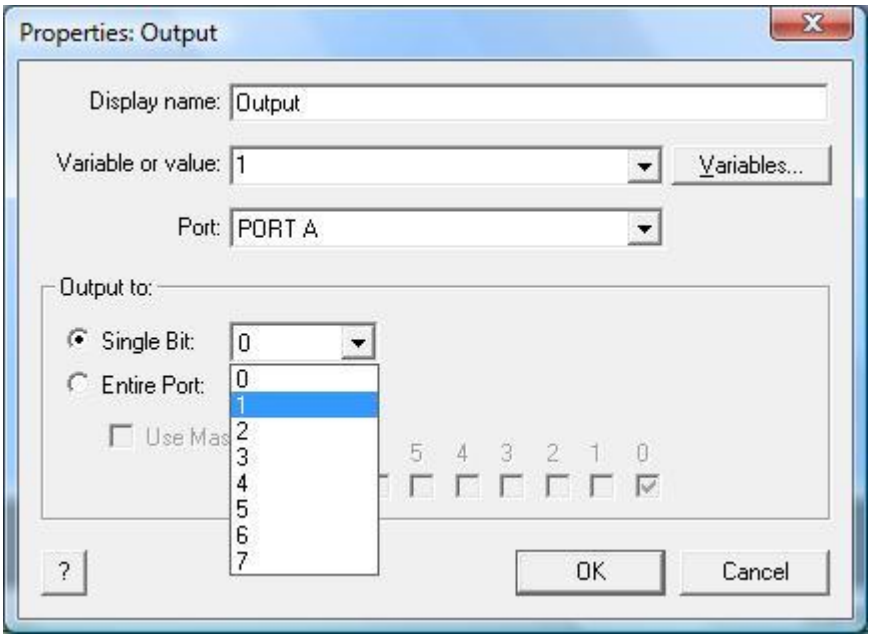
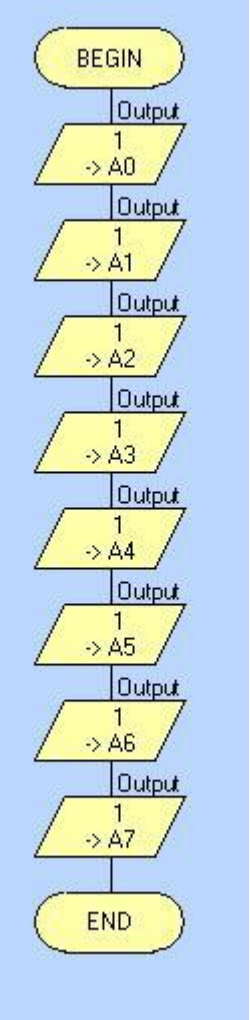


Рис. 3.3. Установка в «1» первого вывода порта А

Посмотрим, как изменится текст программы на языке Си (лишнее опять отбросим):

```
void main()
{
//Initialisation
cmcon = 0x07;

//Output: 1 -> A1
trisa = trisa & 0xfd;
if (1)
porta = (porta & 0xfd) | 0x02;
else
porta = porta & 0xfd;
}
```

На некоторое время оставим «заготовки» на языке Си и вернемся к программе FlowCode. Создадим такую программу для микроконтроллера: последовательно все выводы порта А устанавливаются в «1». В FlowCode программа выглядит так:



Заметьте, что выводы в программных элементах Output меняют свой номер: A0, A1 и т.д.

Эту программу можно проверить в отладчике программы FlowCode, используя линейку светодиодов.

Если теперь нажать кнопку Run основной инструментальной панели, то программа запустится на выполнение. Но ничего интересного мы не увидим – все светодиоды загорятся, и все. Микроконтроллер работает очень быстро, не успеешь глазом моргнуть. И чтобы посмотреть, что происходит в программе, используем кнопку пошагового управления отладкой.

Нажимая эту кнопку, мы увидим, как последовательно зажигаются светодиоды.

Рис. 3.4. Программа последовательной установки всех выводов порта

Но нас сейчас интересует не проверка работы программы (или микроконтроллера), а работа с кодом на языке Си. У нас есть нужные фрагменты программы, постараемся разобраться в них с тем, чтобы воспроизвести последовательное включение выводов порта А на языке Си. Мы можем использовать самый первый фрагмент текста, напомню его:

```
void main()
{
//Initialisation
cmcon = 0x07;
//Output: 0 -> PORT A
trisa = 0x00;
porta = 0;
}
```

В этом фрагменте мы использовали весь порта А для вывода, что, похоже, отображено оператором `trisa = 0x00;` поскольку в следующих фрагментах этот оператор выглядит иначе: с регистром, в котором записывается назначение выводов, проделывается логическая операция `trisa = trisa & 0xfe;`

Если заглянуть в описание микроконтроллера (datasheet), то выяснится, что запись всех нулей в регистр TRISA означает, что все выводы порта А предназначаются на выход. То есть, тот вывод, что для выхода, должен здесь прописываться, как «0», тот, что для входа, как «1». В нашей программе мы можем записать в регистр TRISA все нули, как в первом случае.

Следующий оператор записывает 0 в порт, что переводит все выходы в низкое состояние. Но, если мы изменим оператор, написав `porta = 1;`, то можно предположить, что нулевой бит (вывод RA0) порта А будет «поднят». Какие числа нужно вписывать в порт, чтобы последовательно установить все выводы в высокое («1») состояние? Восемь выводов порта соответствуют восьми битам регистра порта. А для работы с битами есть хороший инструмент – шпаргалка (калькулятор в любой из ОС).

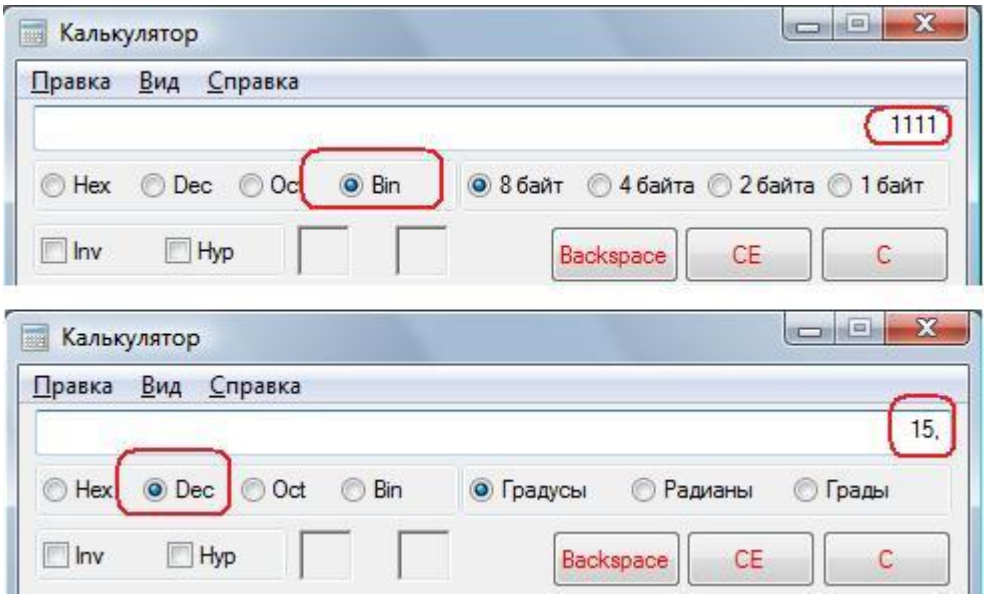


Рис. 3.5. Калькулятор, как шпаргалка для работы с битами

Выяснив, какие числа вписывать в порт А, мы можем записать программу на языке Си:

```
void main()
{
//Initialisation
cmcon = 0x07;
trisa = 0x00;
//Output
porta = 0;
porta = 1;
porta = 3;
porta = 7;
porta = 15;
porta = 31;
porta = 63;
porta = 127;
porta = 255;
}
```

Бумага, как известно, стерпит все. Что ни напишешь, все хорошо. Но понравится ли наша запись кода на языке Си самому языку Си? Вот, что интересно.

Мы можем проверить и это. Изготовители микроконтроллеров предлагают разработчикам бесплатно версии среды разработки. Так производитель PIC-контроллеров предлагает среду разработки MPLAB IDE: http://www.microchip.com/

Эта среда разработки контроллеров прекрасно работает с языком ассемблер, но для использования языка Си нужно добавить компилятор, производимый другим разработчиком. Есть облегченная, но бесплатная версия, HI-TECH (PICC Lite). Основное ограничение – не все контроллеры поддерживаются, и использовать можно только половину памяти. Чтобы скачать компилятор, нужно зарегистрироваться на сайте:

http://www.htsoft.com/products/compilers/PICClite.php

Кроме этого компилятора есть еще SDCC, бесплатный и полный, но его поддерживаются не все версии MPLAB, хотя можно попробовать предыдущие версии на предмет поддержки компилятора SDCC: http://sdcc.sourceforge.net/

И еще, последние версии MPLAB устанавливают компилятор при установке программы.

Вернемся к нашей программе, написанной на языке Си. Запустим установленную на компьютере среду разработки программ MPLAB для микроконтроллеров серии PIC.

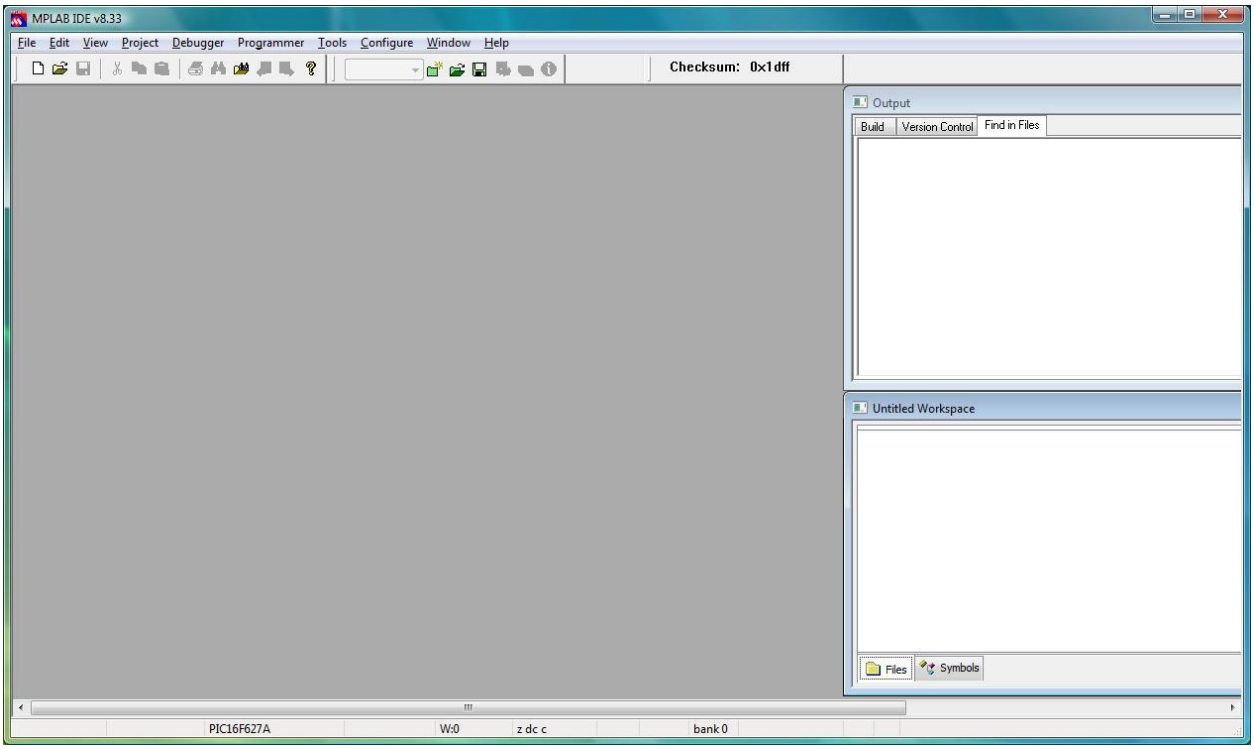


Рис. 3.6. Первый запуск программы MPLAB

Самое разумное при первом знакомстве с программой воспользоваться услугами мастера создания нового проекта: **Project->Project Wizard**.

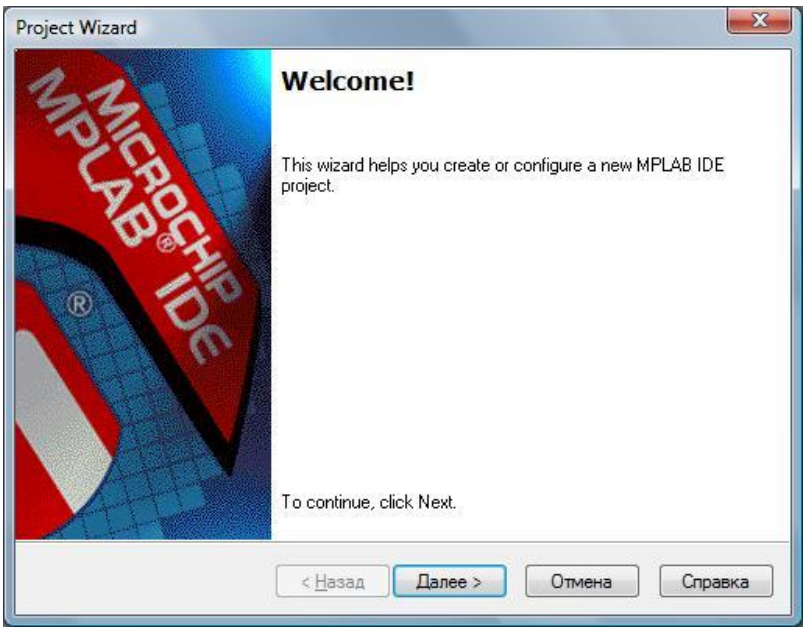


Рис. 3.7. Мастер создания нового проекта

Кнопка ***Далее>*** открывает первое диалоговое окно:

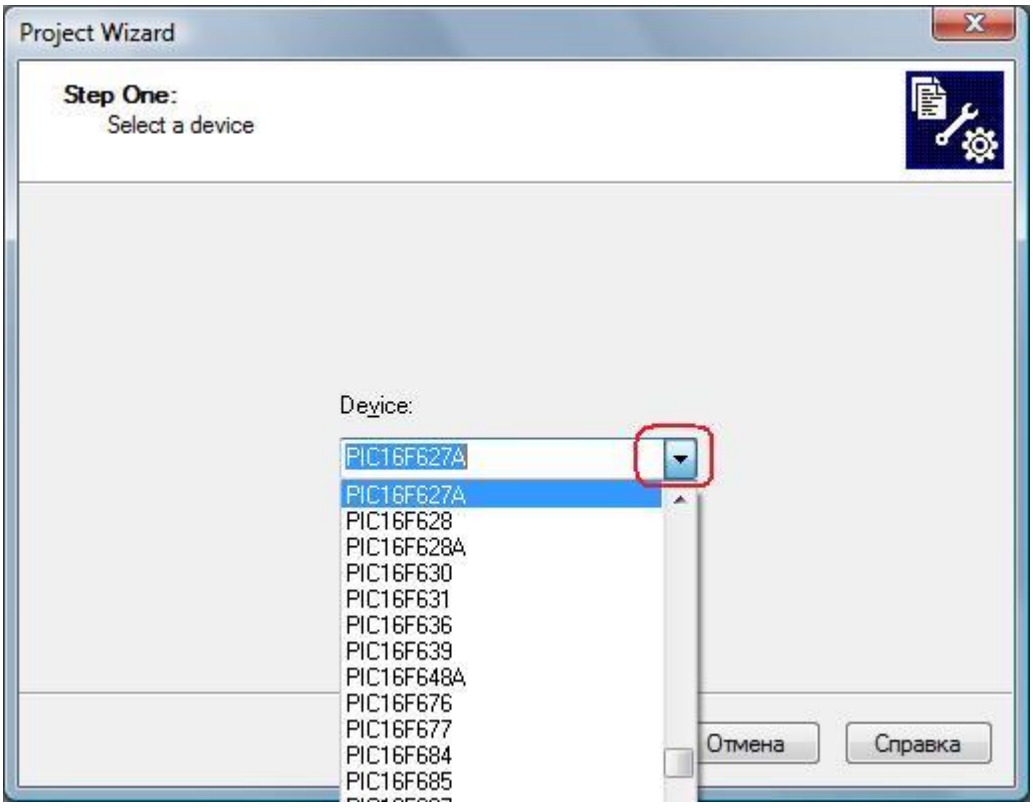


Рис. 3.8. Диалоговое окно выбора модели микроконтроллера

Бесплатная версия компилятора может не работать с PIC16F628A. На всякий случай, пока это не проверено, остановим свой выбор на PIC16F627A. Для выбора служит отмеченная на рисунке кнопка. В следующем диалоге определяется компилятор. Возможно, установленный на этапе установки MPLAB компилятор PICC Lite будет обнаружен сразу, но, если этого не произойдет, можно указать его, используя клавишу ***Browse***. Если установить опцию *Show all installed toolsuites*, то список доступных компиляторов пополнится.

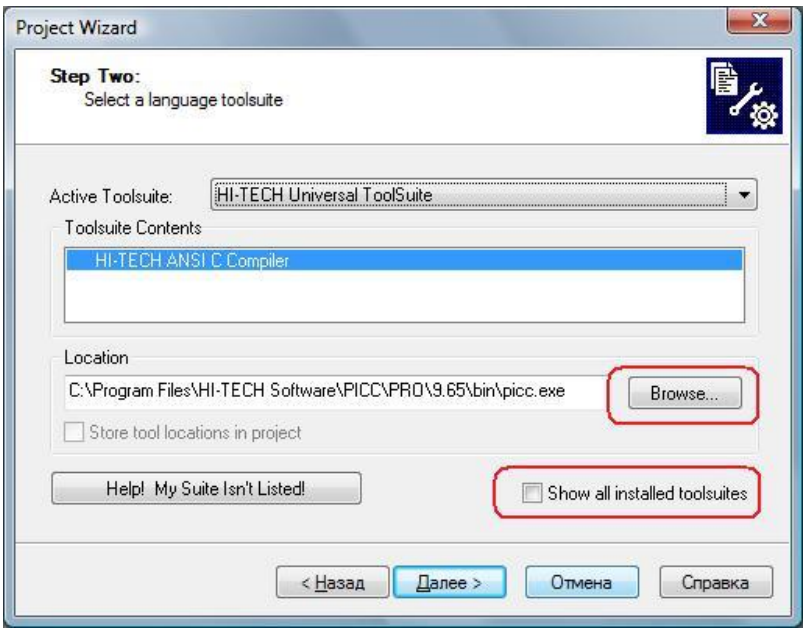


Рис. 3.9. Выбор компилятора для работы с проектом

А в следующем диалоге назовем проект *test* и укажем путь к папке, которую предварительно создали, для этого проекта.

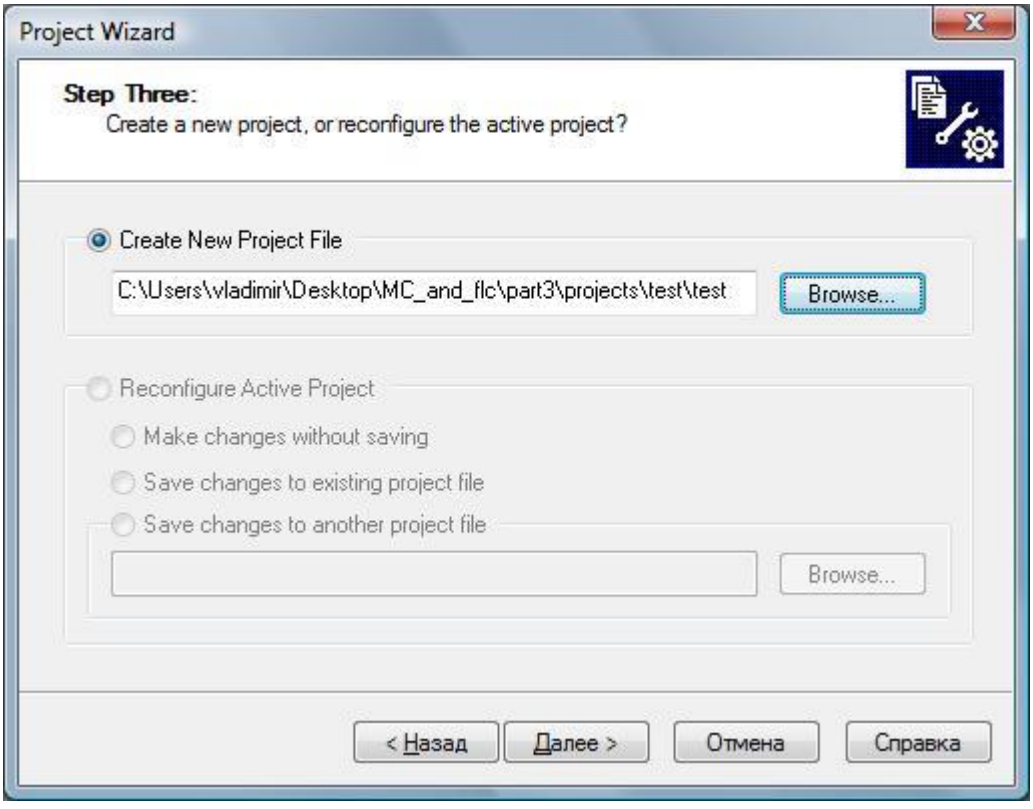


Рис. 3.10. Путь к проекту и имя проекта

По опыту работы с разными программами я знаю, что лучше всего создавать папку, которую можно назвать *work*, в корневой директории основного диска. Но сейчас меня это не очень волнует, и, пока не возникнут трудности, я создаю новый проект на рабочем столе.

На следующем шаге мастер предлагает включить в проект ранее созданные файлы, но их нет, и этот шаг можно пропустить, нажав кнопку ***Далее>***.

Закончив подготовку и нажав на кнопку ***Готово***, вы можете увидеть следующее:

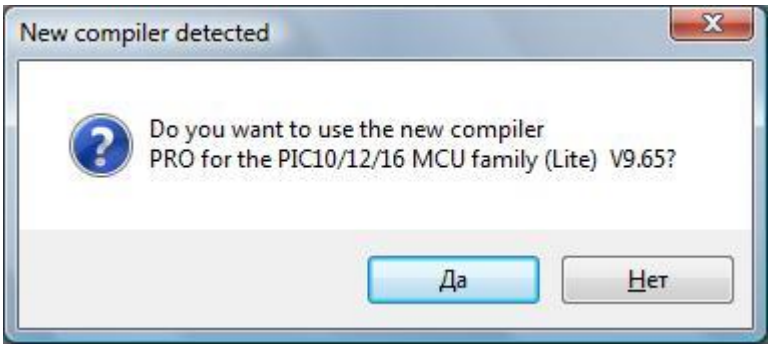


Рис. 3.11. Сообщение о компиляторе

Нажмите кнопку ***Да***.

Прежде пустые окна основного окна программы начинают заполняться.

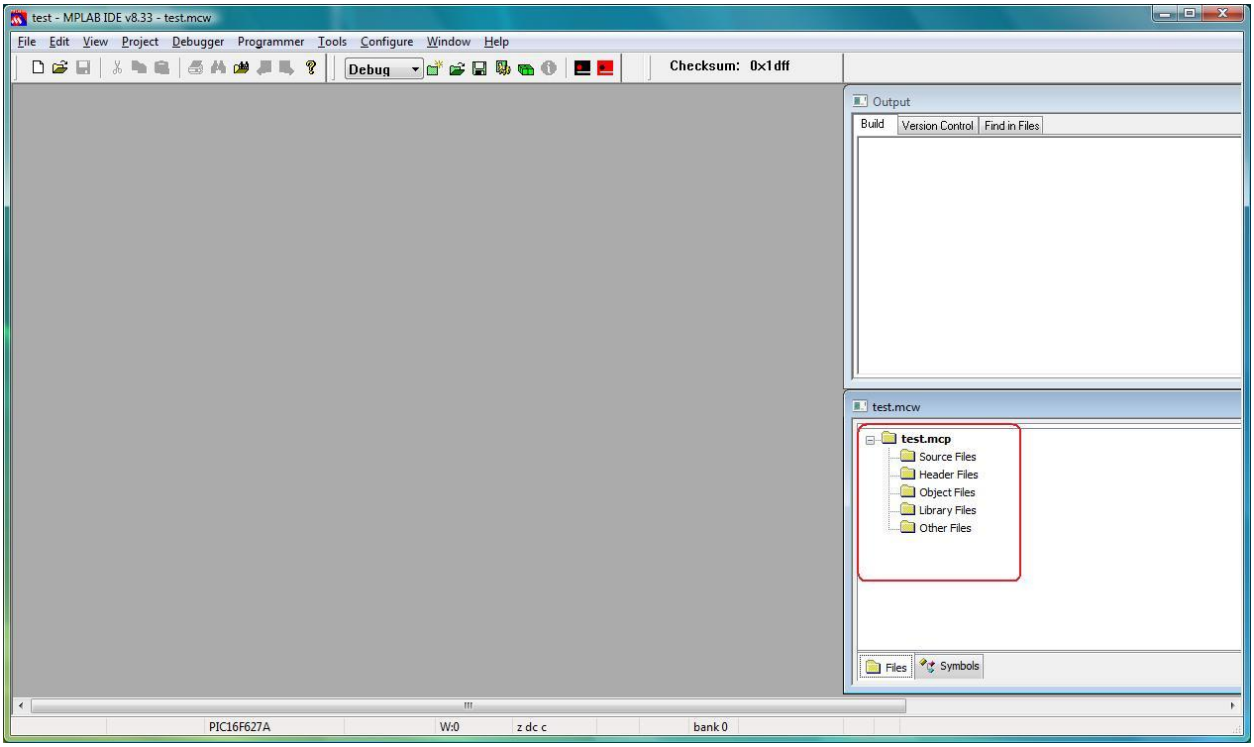


Рис. 3.12. Изменения в основном окне программы после создания проекта

Создадим файл (**Project->Add New File to Project)**, укажем имя нового файла *test.c*, а файл сохраним в той же папке, где лежит проект. Скопируем нашу программу в открытое окно встроенного текстового редактора. Сохраним файл, сохраним проект. И попробуем компилировать файл. В окне Output программы MPLAB на закладке Build появляются сообщения об ошибках.

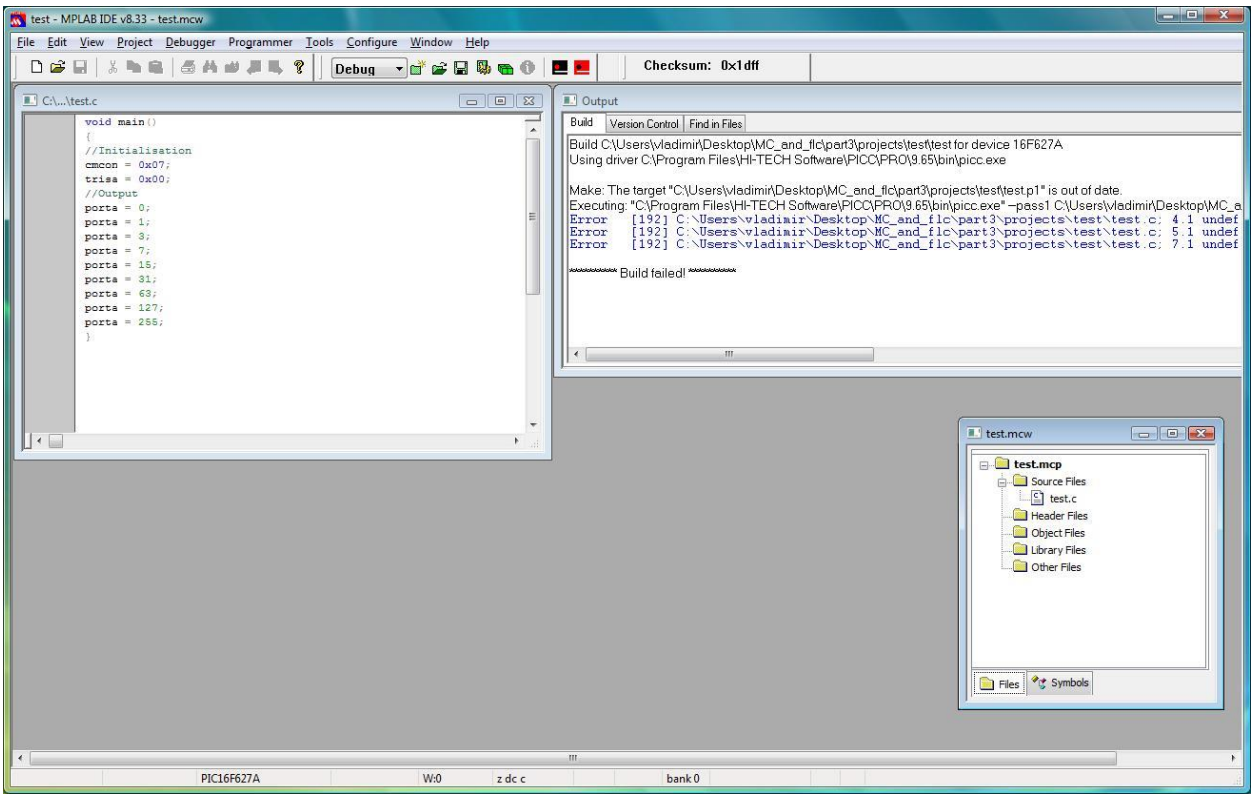


Рис. 3.13. Первая неудачная попытка компиляции

Возможная причина неудачи заключена в следующей фразе:

```
test.c; 4.1 undefined identifier "cmcon"
```

Компилятор сообщает, что идентификатор `cmcon` не определен. В предыдущих версиях компилятора и программы MPLAB (сейчас версия MPLAB 8.33) нужно было включить файл заголовка соответствующего микроконтроллера и изменить названия регистров – заменить маленькие буквы заглавными. Включение файла заголовка дает сообщение о том, что этот файл уже включен, вместо этого предлагается включить другой файл. Для этого достаточно скопировать в программу предложенную строку. После замены букв, перезагрузки программы и выбора ранее созданного проекта компиляция проходит успешно.

В MPLAB есть свой отладчик. Можно проверить, что происходит с портом А. Выберем: **View->Spicial Function Registers**. Этим открывается окно наблюдения за регистрами. Выбираем отладчик:

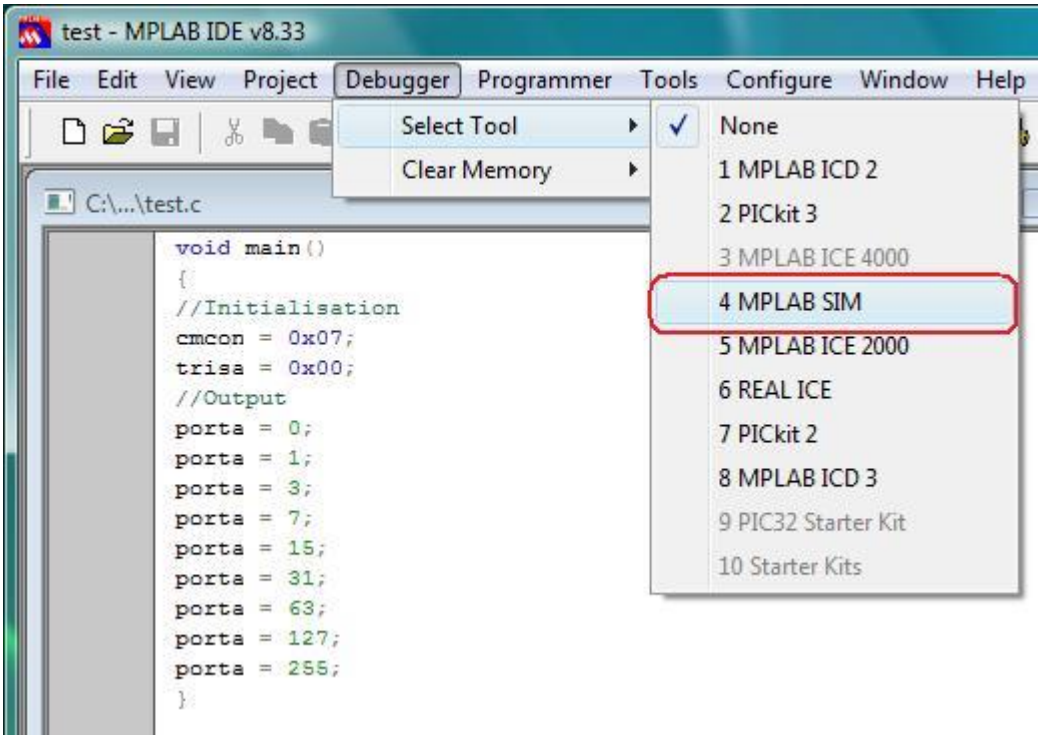


Рис. 3.14. Выбор отладчика в программе MPLAB

Теперь можно запустить отладку: **Debugger->Animate**.

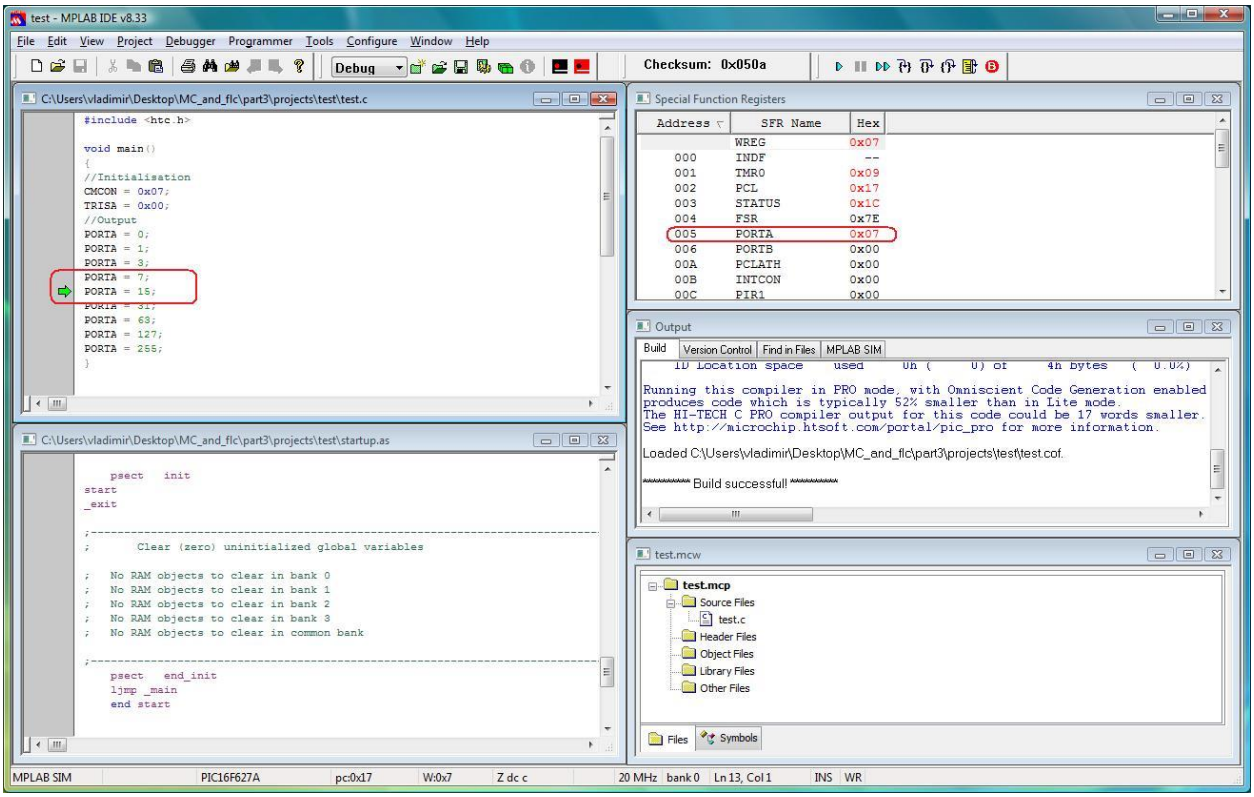


Рис. 3.15. Наблюдение за портом в отладчике программы MPLAB

Состояние порта А меняется согласно тому, что написано в программе.

Мы можем, как и в программе FlowCode, получить загрузочный hex-файл, загрузить его в микросхему, если есть программатор, но, как сказано выше, ничего интересного мы пока на

макетной плате не увидим – все светодиоды, связанные с портом А, загорятся разом. Это можно наблюдать в отладчике программы FlowCode. Чтобы создать некоторый «наблюдаемый» эффект для макетирования, нам предстоит еще поработать над программой.

Подведем первые итоги.

Мы сделали несколько «заготовок» на языке Си в программе FlowCode. Использовали одну из них, чтобы получить программу на языке Си, которую можно использовать в среде разработки MPLAB. Мы проверили, что полученная нами программа вполне «устраивает» программу MPLAB, потребовались лишь незначительные поправки. Мы использовали тот фрагмент, который был удобнее для создания программы – запись числа в порт, но могли бы использовать и другие «заготовки».

При этом мы еще даже не приступили к изучению языка Си. Мы руководствовались, скорее, догадками и соображениями.

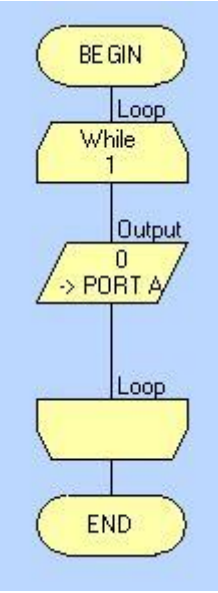
Первые шаги (продолжение)

Вернемся в FlowCode и подумаем над улучшением вида нашей программы. Сейчас программа очень маленькая. Она работает так, как нам хотелось бы, но, когда мы начнем ее «расширять», она разрастется так, что станет «неудобочитаема». Это, как правило, происходит очень быстро. Да и записывать (даже в FlowCode) множество однотипных операций, согласитесь, утомительное занятие. На этот счет программисты давно придумали разные операции, которые помогают сократить число записей.

Заметим, что мы повторяем несколько раз одну и ту же операцию – записываем в порт А некоторое число. Число меняется, но операция нет. Для повторения одной и той же операции (или нескольких операций) давно есть два механизма: цикл и подпрограмма.

Цикл – это повторение одной и той же операции множество раз, вплоть до бесконечного повторения одного и того же.

Есть разные виды циклов, но мы остановимся на двух видах: счетный цикл и условный цикл. В первом случае операция в цикле повторяется заданное число раз, то есть, «цикл считает» количество проходов, а, достигнув нужного, прекращает свою работу. Обращаясь к рисунку 3.4, можно сказать, что программе требуется восемь проходов в цикле. При каждом из них в порт А отправляется число. Если не задумываться над тем, как менять это число, то программа выглядит следующим образом:



Программный элемент **Loop** (цикл), в программе FlowCode находится на инструментальной панели программных компонентов. Как и другие программные компоненты, он «перетаскивается» мышкой в рабочее поле и помещается на линию, соединяющую «программные скобки» BEGIN-END.

В данном случае показан условный цикл *While*. Но в его свойствах (двойной щелчок по элементу открывает диалоговое окно свойств) можно заменить условный цикл счетным.

Рис. 3.16. Цикл в программе FlowCode

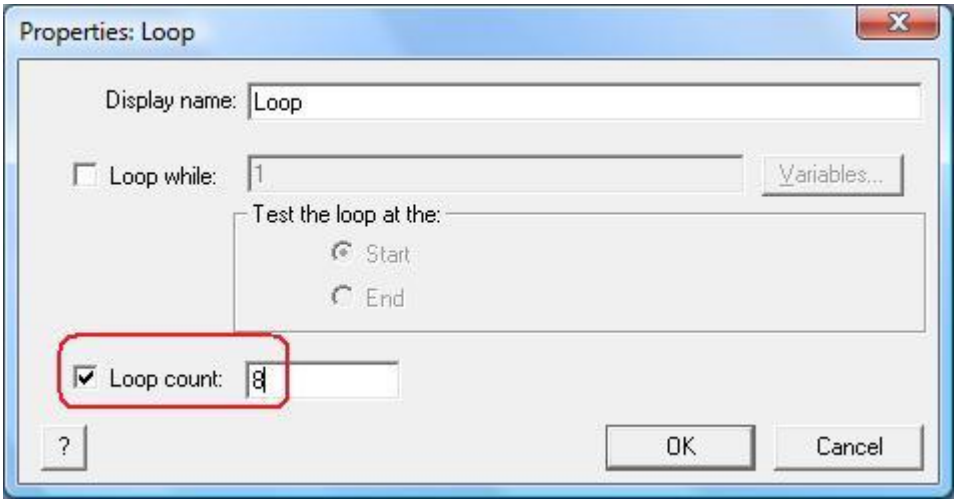


Рис. 3.17. Замена условного цикла счетным

В окошке *Loop count:* количества проходов можно вписать требуемое число. Если вписать число восемь, то цикл будет выполняться восемь раз. И опять, как и прежде, посмотрим, как эта программа выглядит на языке Си.

```
//Variable declarations (объявление переменных)
char FCLV_LOOP1;

void main()
{
      //Initialisation
      cmcon = 0x07;

      //Loop: Loop 8 times
      for (FCLV_LOOP1=0; FCLV_LOOP1<8; FCLV_LOOP1++)
      {
      //Output: 1 -> PORT A
      trisa = 0x00;
      porta = 1;
      }
}
```

Как и прежде, из текста удалено все лишнее. Есть и знакомое место, его мы видели раньше, и оно выделено. Выпишем новые строки кода, поясняя их значение.

```
//Variable declarations (объявление переменных)

char FCLV_LOOP1;
```

Для выполнения цикла восемь раз нужно где-то вести счет. Этой цели служит переменная `FCLV_LOOP1`, которая может иметь любое, в рамках допускаемых компилятором, имя. Кроме того, любая переменная должна иметь заданный тип данных. В данном случае это тип `char`, символьная переменная. Хотя она не используется для хранения символов, но восемь бит этого типа данных вполне устраивают нас.

И дальше новое в коде:

```
void main()
{
      //Loop: Loop 8 times
      for (FCLV_LOOP1=0; FCLV_LOOP1<8; FCLV_LOOP1++)
      {
      что-то делается в этом месте, но не важно, что
      }
}
```

Кстати, все, что следует за двойной косой чертой, это комментарии: записи для программиста, которые поясняют, что это такое или как это понимать. Компилятор игнорирует эти записи. А счетный цикл начинается со слова `for`. Далее в скобках следует запись, что переменная от значения 0 до значения меньшего 8 будет увеличиваться на единицу. Любую переменную можно увеличивать на единицу записью:

```
FCLV_LOOP1 = FCLV_LOOP1 + 1;
```

Но в языке Си часто употребляют запись `FCLV_LOOP1++`, что сокращает время написания оператора, а смысл остается прежним – увеличить значение переменной на единицу. Попутно отметим, что знак «=» в записи выше, отнюдь не знак равенства, а оператор присваивания в языке Си. Для языка Паскаль он выглядит иначе «:=».

Я назвал цикл счетным, но он и в этом случае остается условным, поскольку есть условие:

```
FCLV_LOOP1<8;
```

Все-таки будем называть такой цикл счетным, чтобы легче было различать разные виды циклов. Итак, покинем программу FlowCode, вернемся к программе MPLAB и используем новый программный элемент языка Си для проверки того, как к нему относится компилятор языка Си программы MPLAB. Чтобы не вносить путаницу, я вновь создаю папку с именем test3. И начинаю работу в MPLAB с мастера создания проектов, создавая новый проект под именем test3. Как и в прошлый раз, я копирую текст с записью цикла и вставляю его в новый файл встроенного редактора текста программы MPLAB. Как и в прошлый раз, я меняю маленькие буквы в названиях регистров большими, чтобы не было ошибок на этот счет.

Полученная программа компилируется без ошибок, можно запустить отладку, хотя отлаживать, вернее, смотреть не на что. И можно было бы вернуться к FlowCode, чтобы решить, как нам менять число, которое мы будем записывать в порт, но я хочу поступить иначе.

Мы знаем, как можно менять переменную. Знаем, как записать объявление переменной. Поэтому исправим текст сразу на языке Си:

```
char leds; //объявление новой переменной
```

Внутрь цикла добавим запись:

```
leds = 1 + leds*2;
```

А в порт будем записывать не число, а переменную:

```
PORTA = leds;
```

Программа выглядит следующим образом:

```
#include <htc.h>
//Variable declarations (объявление переменных)
char FCLV_LOOP1;
char leds;

void main()
{
	//Initialisation
	CMCON = 0x07;

	//Loop: Loop 8 times
	for (FCLV_LOOP1=0; FCLV_LOOP1<8; FCLV_LOOP1++)
	{
	//Output: leds -> PORT A
	TRISA = 0x00;
	leds = 1 + leds*2;
	PORTA = leds;
	}
}
```

По совету компилятора, как и в прошлый раз, добавлено включение файла заголовка:

```
#include <htc.h>
```

После компиляции программы и запуска отладчика (**Debugger->Animate**):

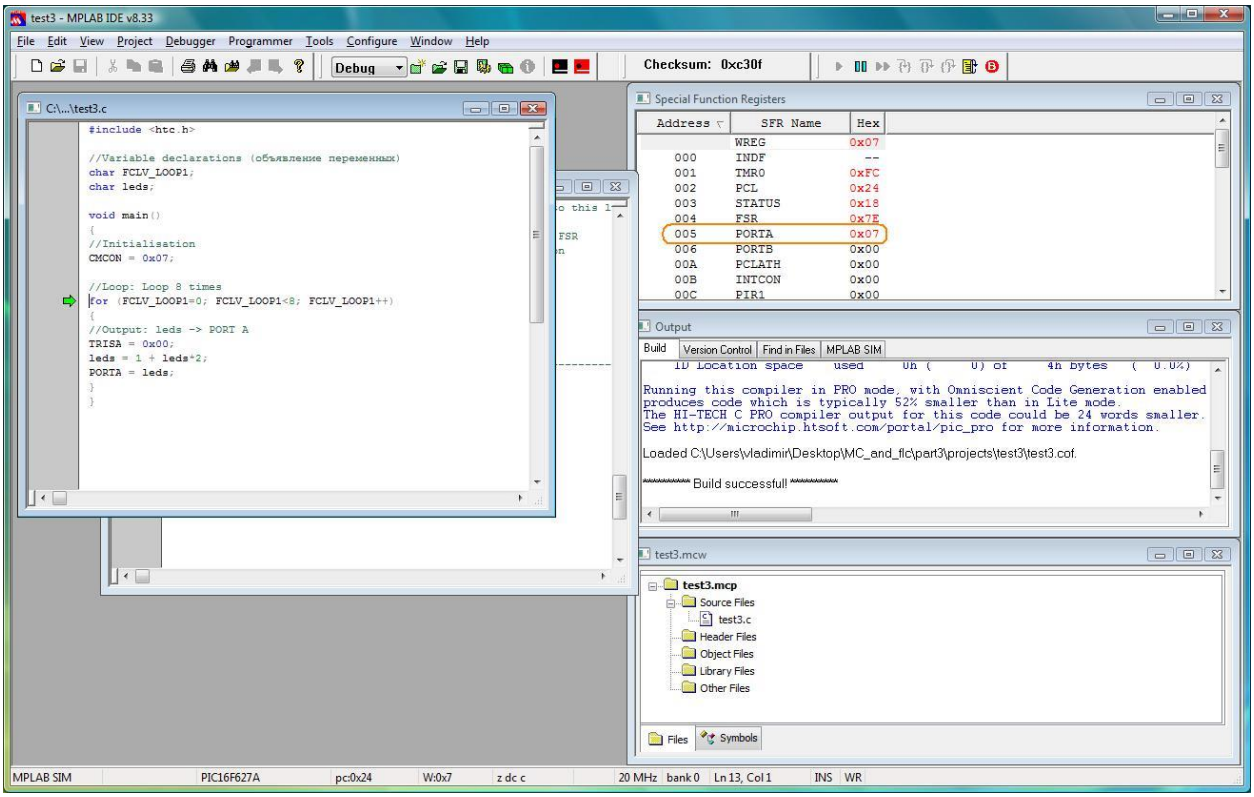


Рис. 3.18. Запуск в отладчике MPLAB модифицированной программы

Если теперь получить hex-файлы двух программ: последней с циклом и первой без него, – то, загрузив эти файлы в две микросхемы контроллеров, можно убедиться, что обе программы работают одинаково (и, кстати, до сих пор ничего интересно увидеть в их работе нельзя).

Как в разговорном языке, так и в языке программирования, можно описать одно и то же различным образом. И, если в начале работы мы переходили от программы FlowCode к программе MPLAB, то сейчас сделаем обратный переход: используем код программы на языке Си для преобразования программы в графический язык.

Как и программа на языке Си позволяет нам вводить переменные, так и программа на графическом языке позволяет сделать это.

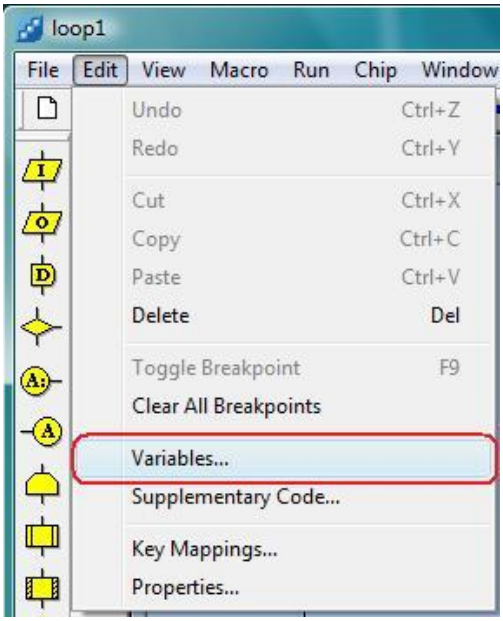


Рис. 3.19. Вызов менеджера обслуживания переменных

Если мы не создали ни одной переменной, то в диалоговом окне увидим следующее:

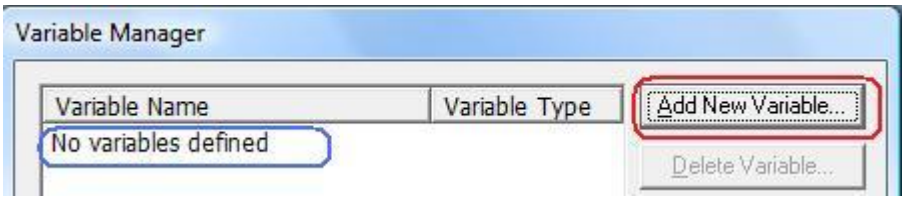


Рис. 3.20. Диалоговое окно менеджера переменных

Воспользуемся кнопкой ***Add New Variable***, чтобы создать новую переменную. После нажатия кнопки открывается новый диалог.

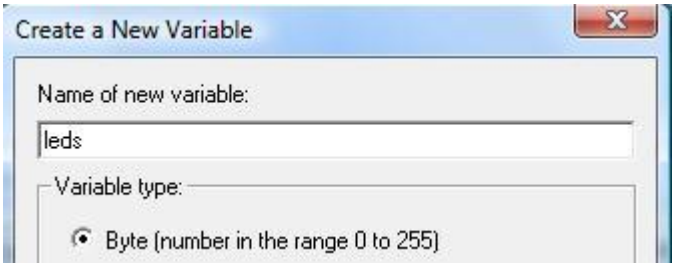


Рис. 3.21. Диалоговое окно создания переменной

Впишем имя переменной, как на языке Си, *leds*, ее тип оставим *Byte*.

Теперь мы можем в порта А писать не число, а переменную.

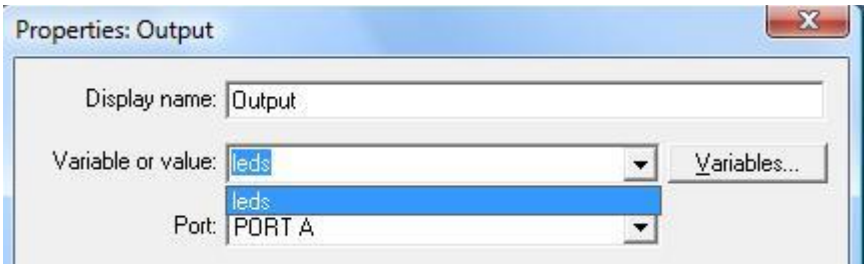
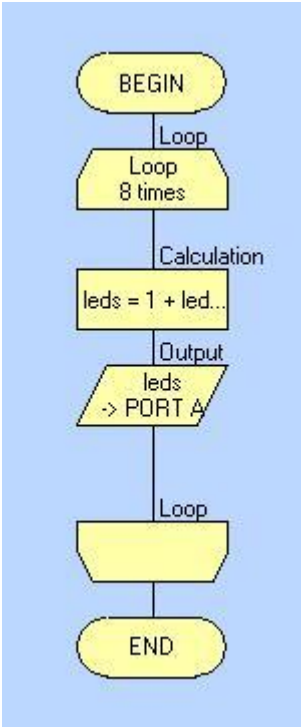


Рис. 3.22. Исправление в диалоге компонента **Output**

Осталось, используя программный компонент **Calculation** программы FlowCode, записать необходимые преобразования в переменную *leds*:

Рис. 3.23. Управление переменной внутри цикла

Программа в FlowCode принимает следующий вид:



Обратите внимание на схожесть записей программы. Если на языке Си для выделения программы используются фигурные скобки, то в FlowCode программа выделяется скобками BEGIN-END.

Если на языке Си цикл начинается с его задания, начинаясь служебным словом `for`, то в FlowCode используется элемент **Loop**. В Си используются фигурные скобки для выделения «тела» цикла, в FlowCode используются графические элементы.

Как в программе на языке Си, так и в программе FlowCode мы меняем переменную *leds* внутри цикла.

Как в программе на языке Си, так и в программе FlowCode мы отправляем переменную в порт А.

Очень похоже, не правда ли?

Рис. 3.24. Запись программы в FlowCode

Может возникнуть вопрос, откуда взялась запись `leds = 1 + leds*2` и в той, и в другой программе? Что означает эта запись, понятно – мы меняем переменную, присваивая ей новое значение. Но почему так, а не иначе? Здесь готового ответа нет. И решение, мне кажется, может быть не одно. Я использовал тот факт, что ряд чисел 1, 3, 7, 15 и т.д., можно представить такой формулой с нулевым начальным значением. Кстати, многие компиляторы при задании переменной дают ей нулевое значение по умолчанию, но неплохо было бы записать это явным образом (что я не сделал!). Подобная небрежность в программировании может приводить к трудно обнаруживаемым ошибкам.

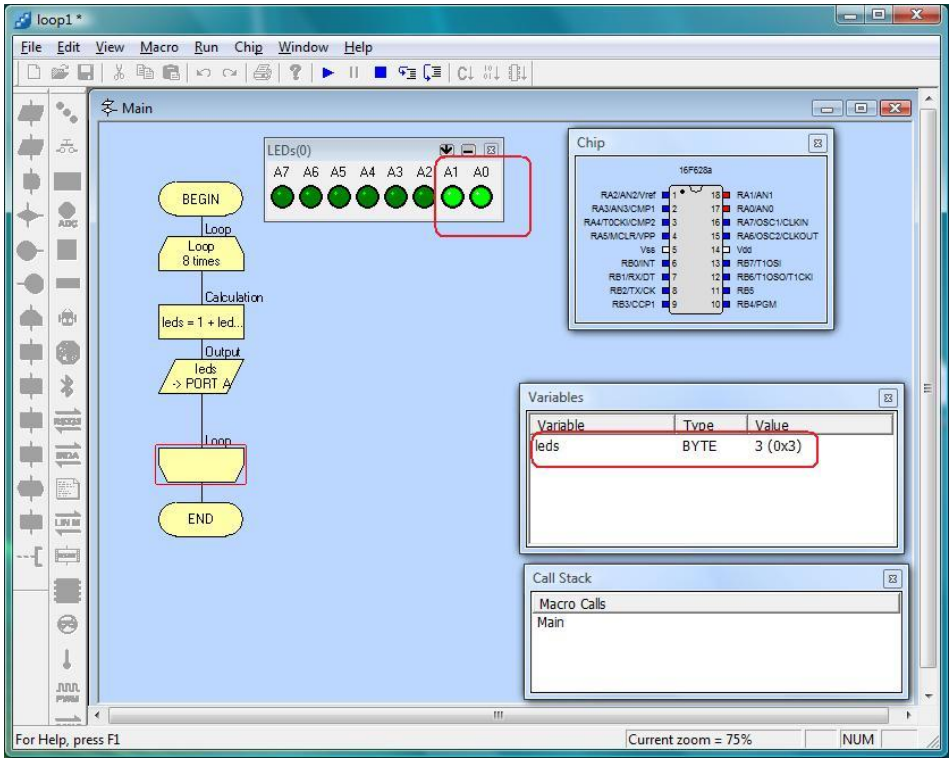


Рис. 3.25. Работы программы в пошаговом режиме в FlowCode

И все-таки хотелось бы увидеть результат работы программы не в отладчике, а на макетной плате. Чтобы это получилось достаточно добавить в программу (на рисунке выше) еще одни программный элемент – **Delay** (пауза). Его можно добавить ниже элемента **Calculation**, в котором меняется число для записи в регистр порта А. По умолчанию длительность паузы одна миллисекунда. Достаточно изменить это значение на 300 мс, скажем, чтобы, запустив «прогон» программы в отладчике (кнопка на инструментальной панели или **Run->Go/Continue** в основном меню) FlowCode увидеть то, что будет показано и на макетной плате.

Однако цель – получить код программы на языке Си – еще не достигнута. Конечно, можно и в этот раз скомпилировать программу в FlowCode и посмотреть код программы на Си. Но ничего интересного, в конечном счете, мы не увидим, поскольку вот такой код:

```
//Delay: 300 ms

delay_ms(255);

delay_ms(45);
```

для любого другого, отличного от компилятора FlowCode, компилятора будет непонятен. Компилятор FlowCode имеет добавляемый файл, где функция `delay_ms()` полностью определена и описана. А компилятор MPLAB просто покажет ошибку. Важно еще, что функция дает интервал времени, выраженный в долях секунды, но определяемый в зависимости от тактовой частоты микроконтроллера. А тактовая частота может в широких пределах меняться. Чтобы получить нужный интервал времени, придется использовать какие-то другие механизмы в программе на языке Си. В данном конкретном случае нам не нужна строго определенная по времени пауза, что позволяет использовать в коде программы пустой счетный цикл, с которым мы знакомы.

```
#include <htc.h>
//Variable declarations (объявление переменных)
char FCLV_LOOP1;
char leds;
int dl;

void main()
{
      CMCON = 0x07;
      for (FCLV_LOOP1=0; FCLV_LOOP1<8; FCLV_LOOP1++)
      {
      TRISA = 0x00;
      leds = 1 + leds*2;
      PORTA = leds;
      for (dl=0; dl<=10000; dl++){}
      }
}
```

Для добавленного цикла используется еще одна переменная *dl* типа *int* (целое). В остальном все нам знакомо.

Длительность паузы, «изготовленной» подобным образом, можно проверить на макетной плате, подбирая подходящее значение. Можно использовать отладчик, чтобы «оценить» длительность паузы. Но самое время заглянуть в папку примеров, которые установлены вместе с компилятором HI-TECH для программы MPLAB. Среди примеров есть и пример использования функции паузы, которая поддерживается файлом *htc.h*.

Все, что необходимо добавить в программу, оказывается парой строк, а программа приобретает вид:

```
#include <htc.h>
#define _XTAL_FREQ 4000000

//Variable declarations (объявление переменных)
char FCLV_LOOP1;
char leds;

void main()
{
      //Initialisation
      CMCON = 0x07;

      //Loop: Loop 8 times
      for (FCLV_LOOP1=0; FCLV_LOOP1<8; FCLV_LOOP1++)
```

```
        {
        //Output: leds -> PORT A
        TRISA = 0x00;
        leds = 1 + leds*2;
        PORTA = leds;
        __delay_ms(100);
        }
}
```

Длительность в 100 мс получается, как определено инструкцией:

```
#define _XTAL_FREQ 4000000
```

при частоте 4 МГц. Если использовать в МК PIC16F628A внутренний тактовый генератор, то частота, как раз, получится равной 4 МГц.

Эту задержку, если нужно ее увеличить до 300 мс, можно повторить три раза с помощью того же счетного цикла:

```
char i;

for (i=0; i<3; i++) __delay_ms(100);
```

С паузами в FlowCode есть одна небольшая неприятность – можно использовать только целые значения секунд или миллисекунд и нельзя получить паузы длительностью в несколько микросекунд. В программе MPLAB эта проблема решается гораздо проще.

Итак.

С помощью программы FlowCode мы получили текст программы на языке Си, который можно компилировать в среде разработки MPLAB, предназначенной для работы с языками Си и ассемблером. Результат полной компиляции, hex-файл, можно загрузить в микросхему, создав почти полезное устройство, если всю программу поместить в бесконечный цикл while. Микроконтроллер будет, например, включать гирлянды или оформление витрины. При этом мы использовали только три программных компонента программы FlowCode.

Шире шаг – ветвление, подпрограмма, ввод, прерывание, программатор

Очень часто есть необходимость в ветвлении программы. То есть, если произошло какое-то событие или изменение, скажем, переменной, то соответственно этому программа должна выполнить разные действия. Это условное ветвление программы. Например, если переменная, которую мы назовем *SOME*, принимает значение равное единице, то выполняется одно действие, иначе другое.

Программный компонент **Decision** в FlowCode, если его добавить в программу, а затем получить код на языке Си, даст следующую запись:

```
if (FCV_SOME == 1)
        {
                // Некое действие

        } else {
                // Другое действие
        }
```

Это буквальная запись: если (в скобках условие), то… иначе другое. С подобной записью мы сталкивались, когда переводили в код Си вывод одного бита порта А.

Условия могут при необходимости следовать одно за другим, выстраиваясь в цепочку. В языке Си, чтобы не загромождать текст программы подобными конструкциями, предусмотрен оператор, который называется «программным переключателем».

В программе FlowCode необходимость в использовании условного ветвления возникает чаще всего тогда, когда используется ввод (программный компонент **Input**). К микроконтроллеру может быть подключена клавиатура, и программа ждет нажатия клавиши, и когда клавиша нажата, программа меняет свой ход. Или к микроконтроллеру подключен датчик. Пока на входе одно значение, получаемое от датчика, программа, скажем, подключает обогреватель, а при изменении состояния датчика, когда меняется значение на входе микроконтроллера, программа отключает обогреватель.

Ветвления программы возникают и при использовании циклов, используются при программировании и безусловные переходы (оператор *GOTO*). Но, пожалуй, условный переход используется наиболее часто. Особенно это характерно для микроконтроллеров, основное назначение которых обеспечить отслеживание событий во внешнем мире и выбрать реакцию на эти события, в зависимости от порядка или сочетания событий, отражающихся на входах микроконтроллера.

Кроме явных ветвлений программы есть ряд ветвлений, которые, собственно, ветвлениями программы и не считаются, хотя программа, дойдя до этого элемента, не спешит выполнить следующий.

В программе FlowCode есть программный компонент **Macro**.

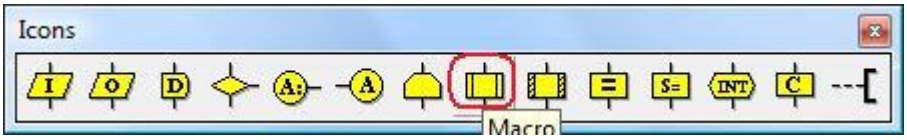
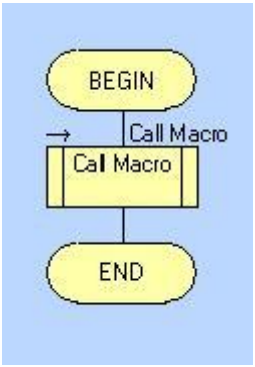


Рис. 3.26. Подпрограмма в FlowCode

Мы можем добавить этот компонент в программу.



Добавление компонента, как это видно из рисунка, не более чем вызов подпрограммы (*Call Macro*). Саму подпрограмму следует написать.

Рис. 3.27. Создание программы

Открыв диалоговое окно этого компонента двойным щелчком левой клавиши мышки по нему, дадим ему имя. Но перед этим следует, используя кнопку ***Create New Macro***, создать подпрограмму.

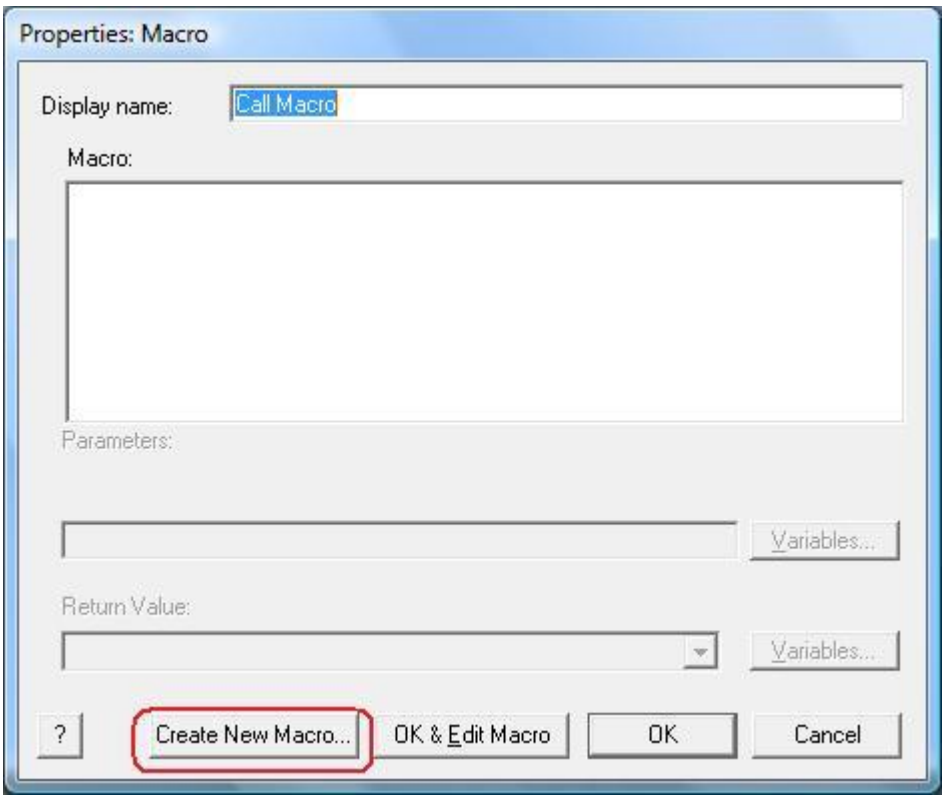


Рис. 3.28. Диалоговое окно **Macro**

В новом диалоговом окне можно вписать имя подпрограммы.

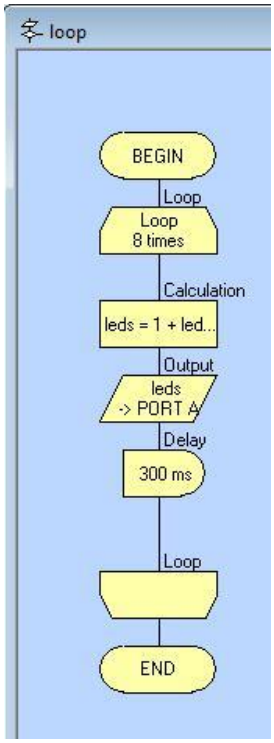
Рис. 3.29. Задание имени подпрограмме

Нажимая кнопки ***OK***, мы возвращаемся в окно основной программы. Мы еще не написали подпрограмму, но можем компилировать результат на язык Си (**Chip->Compile to C**). Открыв полученный код на языке Си во встроенном текстовом редакторе, можно посмотреть, что из текста, касающегося подпрограммы, было добавлено.

```
//Macro function declarations (объявление)
void FCM_loop();

//Macro implementations (реализация)
void FCM_loop()
{
}

void main()
{
    //Initialisation
    cmcon = 0x07;

    //Call Macro
```

```
    //Call Macro: loop (вызов подпрограммы)
    FCM_loop();
}
```

Хотя нам это и не нужно, мы можем ранее написанную программу вставить в подпрограмму.



Нам это не нужно, поскольку программа уже есть. Но, с другой стороны, чтобы не писать новую программу для этого примера, отчего бы не использовать уже готовый вариант?

Чтобы создать подпрограмму можно воспользоваться кнопкой ***OK & Edit Macro*** в диалоге (рис. 3ю28). В результате появится второе рабочее поле с именем *loop*.

Все элементы программы, благо их немного, можно быстро разместить так же, как это было сделано в основной программе ранее. Но, можно было в предыдущей программе использовать функцию экспорта подпрограммы, а сейчас импортировать подпрограмму.

Для экспорта подпрограммы нужно было добавить компонент Macro, дать ему имя, перенести всю программу в подпрограмму. После этого активизируется команда экспорта.

Рис. 3.30. Создание подпрограммы

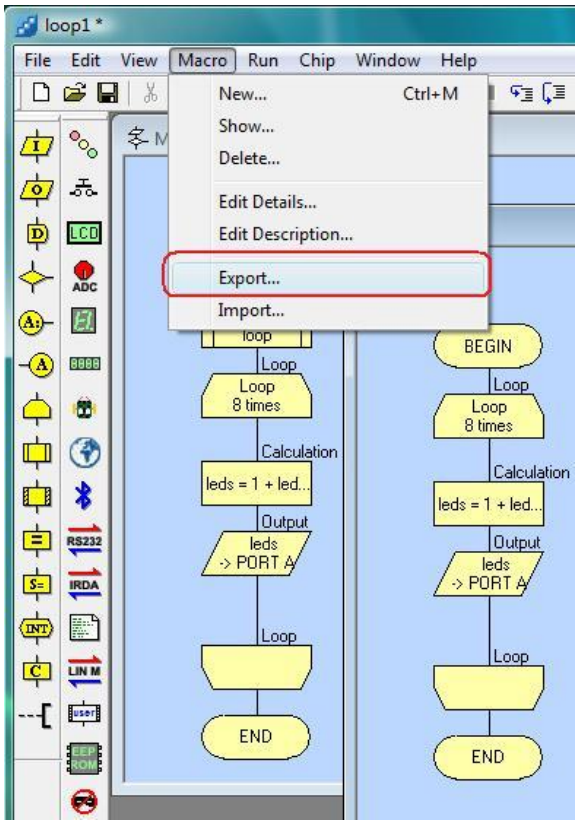


Рис. 3.31. Подготовка программы к экспорту подпрограммы

Экспортированная подпрограмма сохраняется с расширением, которое FlowCode понимает как подпрограмму, и именем, которое мы дали подпрограмме.

Экспортированную подпрограмму можно импортировать в новой программе (если сделать это до того, как мы дали имя подпрограмме).

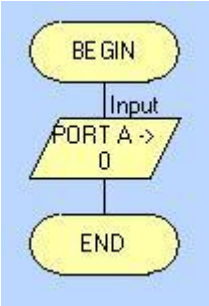
Но вернемся к подпрограмме на языке Си. Написав подпрограмму, можно получить код на языке Си в FlowCode. И посмотреть, что изменилось с прошлого раза?

```
//Macro implementations (реализация)
void FCM_loop()
{
    //Loop
    //Loop: Loop 8 times
    for (FCLV_LOOP1=0; FCLV_LOOP1<8; FCLV_LOOP1++)
    {
          //Calculation
          //Calculation:
          //  leds = 1 + leds * 2
          FCV_LEDS = 1 + FCV_LEDS * 2;

          //Output
          //Output: leds -> PORT A
          trisa = 0x00;
          porta = FCV_LEDS;


          //Delay
          //Delay: 300 ms
          delay_ms(255);
          delay_ms(45);
    }
```

В первую очередь изменилось содержание реализации подпрограммы. У нас есть опыт использования полученного в программе FlowCode кода на языке Си, поэтому, поменяв паузу, изменив написание регистров, мы можем использовать подпрограмму в среде разработки MPLAB.

Начиная рассказ о программных компонентах FlowCode, мы добавили единственный элемент – **Output**. Повторим это для **Input** – программного компонента, обеспечивающего ввод информации в контроллер. В самом простом случае эта информация определяет состояние одного или нескольких выводов: высокий уровень или низкий. Например, к этому выводу подключена кнопка. Если вывод «подтянут» резистором к плюсу источника питания, то кнопка «вторым концом», видимо, должна быть соединена с общим проводом. Тогда при нажатой кнопке на входе будет низкий уровень. Изменение уровня на входе можно зафиксировать как событие, на которое микроконтроллер должен как-то отреагировать.

Но вначале проделаем все операции, то есть, создадим новый файл, «подцепим» мышкой программный компонент **Input** на инструментальной панели программных компонентов и перенесем его на линию, соединяющую программные скобки BEGIN-END.



Пока мы не меняем свойств компонента, не возникает никаких вопросов. Но, если открыть диалоговое окно, чтобы убедиться в использовании всех выводов порта для ввода, то при попытке закрыть окно, нажав на кнопку ***OK***, мы увидим сообщение…

Рис. 3.32. Добавление в программу компонента **Input**

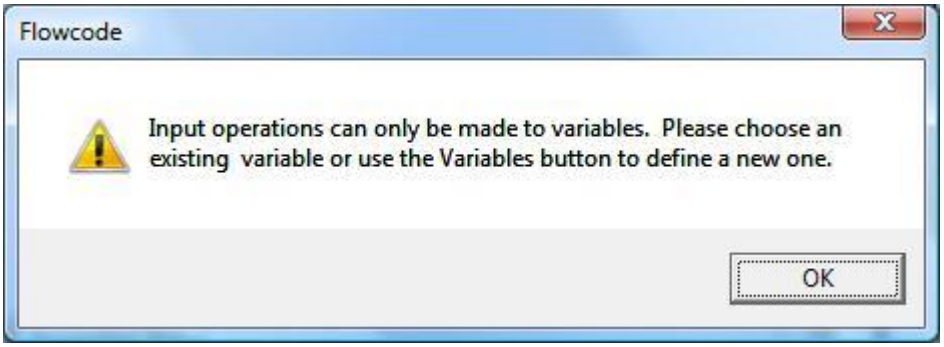


Рис. 3.33. Сообщение по нажатию на кнопку ***OK*** диалога

В этом сообщении сказано, что операция ввода может быть осуществлена только для переменной. Предлагается выбрать либо уже существующую переменную, либо использовать кнопку ***Variables*** для создания новой переменной. Воспользуемся последним предложением для создания переменной *key*, которая будет «привязана» к порту для отображения его состояния. Закрыв диалоговое окно, мы можем оттранслировать программу на Си (**Chip->Compile to C**). Убрав, как обычно, все «лишнее», получим код на языке Си:

```
//Variable declarations
char FCV_KEY;

void main()
{
    //Initialisation
    cmcon = 0x07;

    //Input: PORT A -> key
    trisa = trisa | 0xff;
    FCV_KEY = porta;
}
```

Запись: `trisa = trisa | 0xff;` означает логическую операцию ИЛИ между значением регистра `trisa` и шестнадцатеричным числом `ff`, в результате которой в регистр будут записаны единицы, что, в свою очередь, означает – порт предназначен для ввода. А мы получили на языке Си фрагмент программы, позволяющий отслеживать состояние порта, при изменении которого, мы можем что-то сделать, исходя из наших намерений и нужд.

И еще одно, если использовать не порт А, а порт В, то… для отслеживания состояния порта можно использовать прерывание. Иногда вполне можно обходиться без прерываний, но не всегда. Посмотрим, как выглядит прерывание, записанное на языке Си.

Создадим новый файл, в программу добавим программный элемент **Interrupt**.

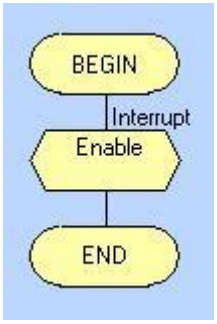


Рис. 3.34. Программа с компонентом **Interrupt**

Пока мы не задали свойств прерывания, программа показывает только то, что разрешено «какое-то» прерывание. Откроем диалоговое окно свойств прерывания. Из возможных для микроконтроллера PIC16F628A прерываний выберем одно, отмеченное на рисунке.

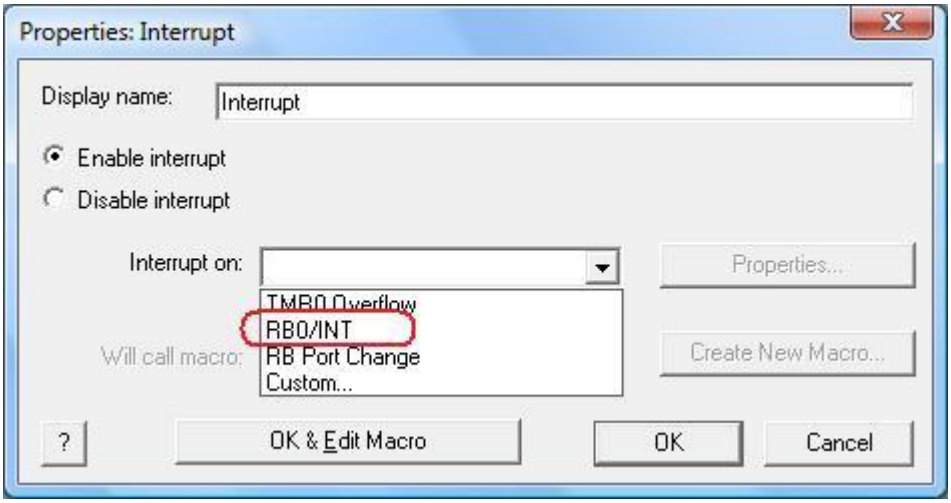


Рис. 3.35. Диалоговое окно программного компонента прерывание

Выбор этого вида прерывания означает, что прерывание разрешено при изменении состояния одного вывода порта В, нулевого бита регистра порта. Но, если мы разрешили прерывание, мы должны создать подпрограмму обслуживания прерывания, что можно сделать, используя кнопку *Create New Macro* (после выбора вида прерывания она активизируется). Подпрограмму назовем key_intr, впрочем, вы вольны выбрать другое название.

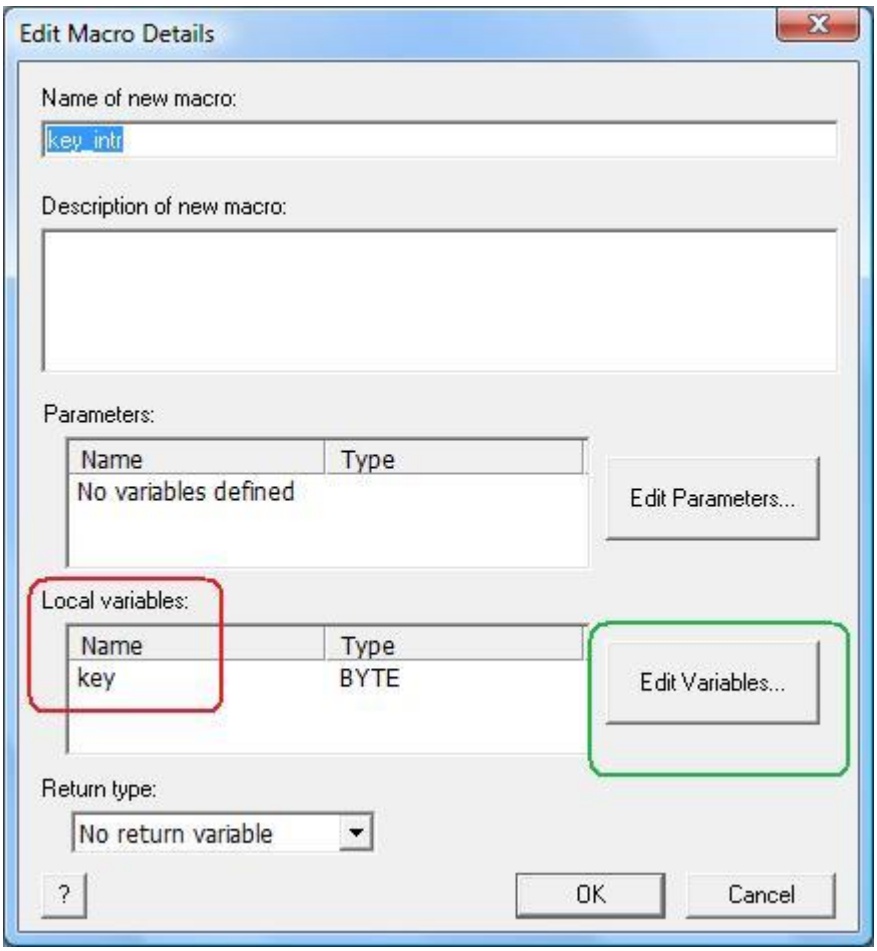


Рис. 3.36. Диалоговое окно создания подпрограммы обслуживания прерывания

Для подпрограммы я создал (с помощью кнопки ***Edit Variables***) локальную, то есть, существующую только в пределах подпрограммы, переменную *key*. И создал только для того, чтобы показать разницу между локальной и глобальной, действительной для всей программы, переменными. В самой подпрограмме обслуживания прерывания я добавлю только один программный компонент **Calculation**.

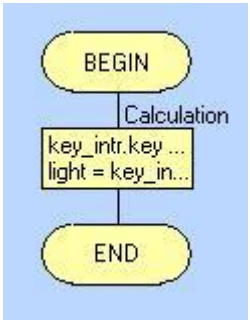


Рис. 3.37. Вид подпрограммы обслуживания прерывания

А сам элемент **Calculation** заполнен:

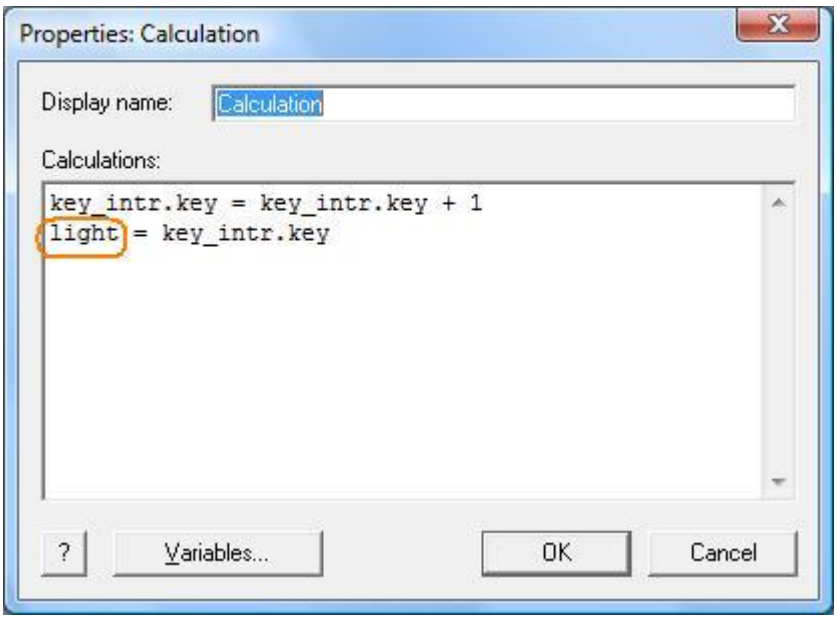
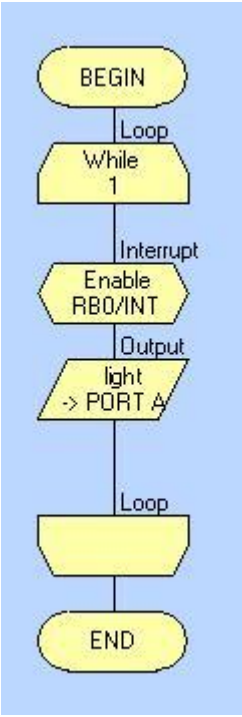


Рис. 3.38. Записи в **Calculation**

Вторую переменную, *light*, нужно создать в программе, как обычно, чтобы она стала доступна в любом месте программы, в частности, и в подпрограмме. Кстати, если сейчас заглянуть в менеджер переменных (**Edit->Variables**), то мы увидим только переменную *light*.

Добавим в программу бесконечный цикл, мы умеем это, добавим вывод в порт А, и это мы умеем, чтобы программа выглядела так:



Для отладки программы, вернее, чтобы увидеть, что делает программа, если она что-то будет делать, используем линейку светодиодов с панели дополнительных компонентов и выключатели. Можно оставить выключатели, изменив только характер включения, как есть, но можно убрать все лишние, оставив только один.

Посмотреть, что происходит в программе, удобнее в режиме пошаговой отладки.

Проходя цикл While несколько раз, вы не увидите изменений в состоянии порта А – ни один из светодиодов не загорается. Но, если вы включите **Switches**, то попадете в подпрограмму обработки прерывания, где изменится значение переменной *light* через локальную переменную *key*.

Рис. 3.39. Программа с прерыванием

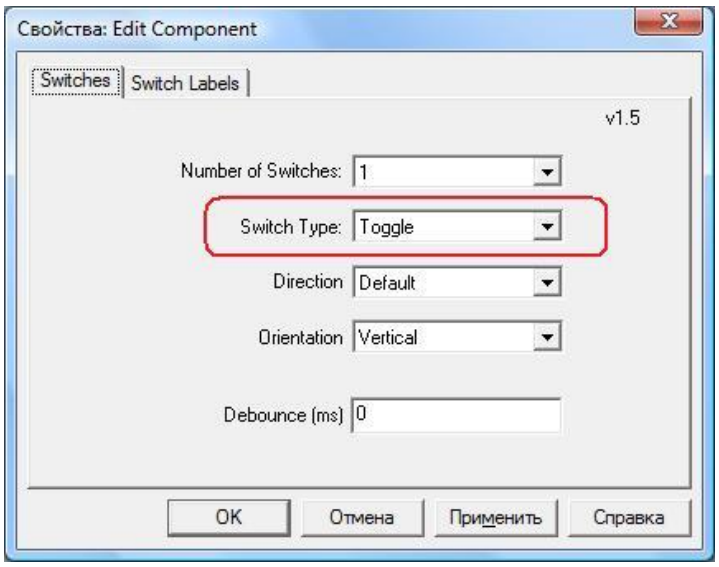


Рис. 3.40. Диалоговое окно свойств дополнительного компонента **Switches**

Разница между локальной и глобальной переменными отражается в том, что сколько бы раз вы ни попали в подпрограмму, значение глобальной переменной *light* остается прежним. Каждый раз, когда возникает прерывание, локальная переменная *key* «начинает новую жизнь», принимая значение по умолчанию равное нулю. В компоненте подпрограммы **Calculation** она увеличивает это значение на единицу и передает его глобальной переменной *light*. Все. В следующий раз, при следующем прерывании, все начинается сначала.

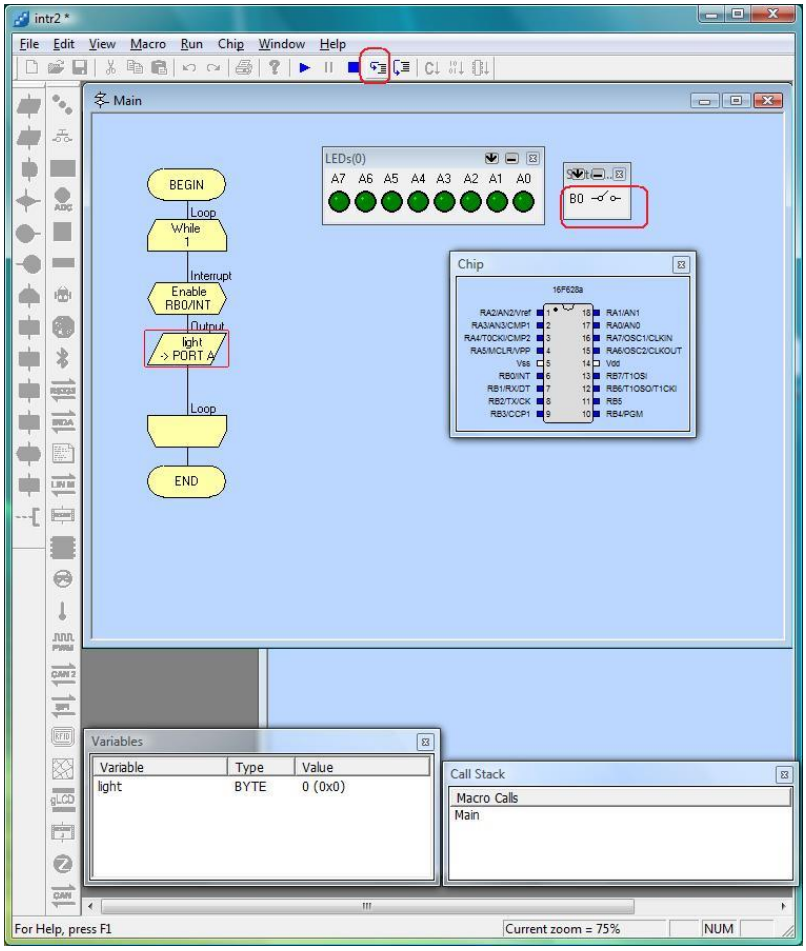


Рис. 3.41. Пошаговое прохождение программы до изменения состояния выключателя

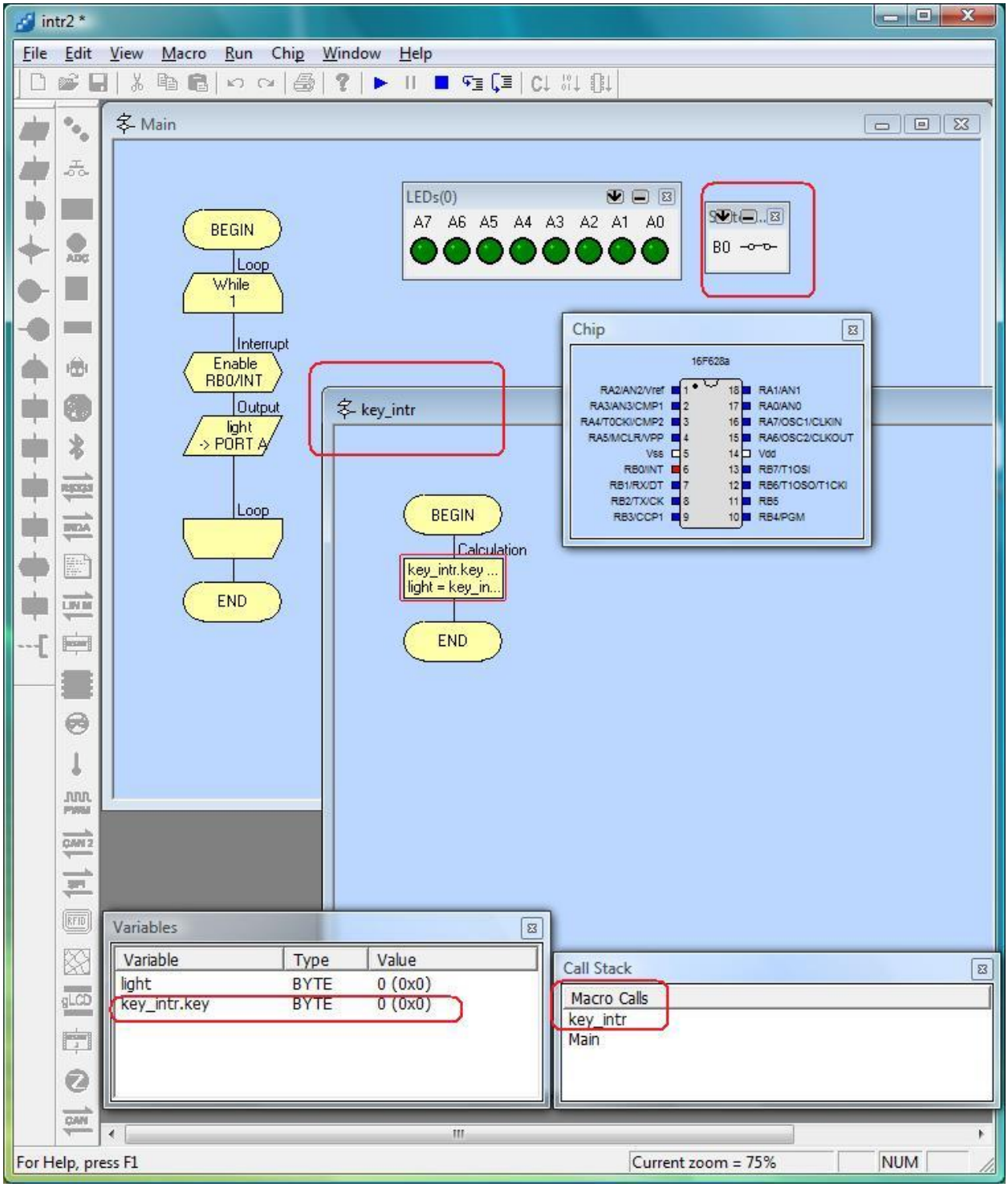


Рис. 3.42. Прерывание при изменении состояния выключателя

И, оттранслировав полученную программу, мы получим код на языке Си:

```
//Macro function declarations
void FCM_key_intr();

//Variable declarations
char FCV_LIGHT;

//Macro implementations
void FCM_key_intr()
{
 //Local variable definitions
 char FCL_KEY;

 //Calculation:
 //  key_intr.key = key_intr.key + 1
 //  light = key_intr.key
 FCL_KEY = FCL_KEY + 1;
 FCV_LIGHT = FCL_KEY;
}

void main()
{

 //Initialisation
 cmcon = 0x07;

 //Interrupt initialisation code
 option_reg = 0xC0;

 //Loop
 //Loop: While 1
 while (1)
 {
       //Interrupt: Enable RB0/INT
       option_reg.INTEDG=1;
       intcon.GIE=1;
       intcon.INTE=1;

       //Output: light -> PORT A
       trisa = 0x00;
       porta = FCV_LIGHT;
 }
}
```

Проверить, как отнесется к этому коду на языке Си компилятор программы MPLAB (или AVRStudio), я предоставлю вам, а сейчас хочу упомянуть о программаторе. С программой MPLAB работает программатор PICkit 2 (появился и PICkit 3), подключаемый к порту USB компьютера. Есть подозрение, что он будет работать и с программой FlowCode. Стоит программатор не слишком дорого, вдобавок его можно собрать и самостоятельно – в Интернете можно найти и схему, и прошивку. А выбор программатора осуществляется в диалоговом окне конфигурации (**Chip->Configure**).

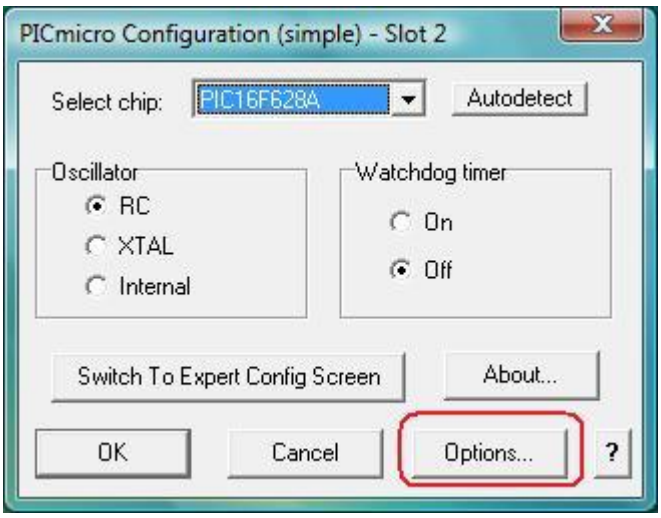


Рис. 3.43. Диалоговое окно конфигурации микроконтроллера

Кнопка ***Options*** открывает следующий диалог:

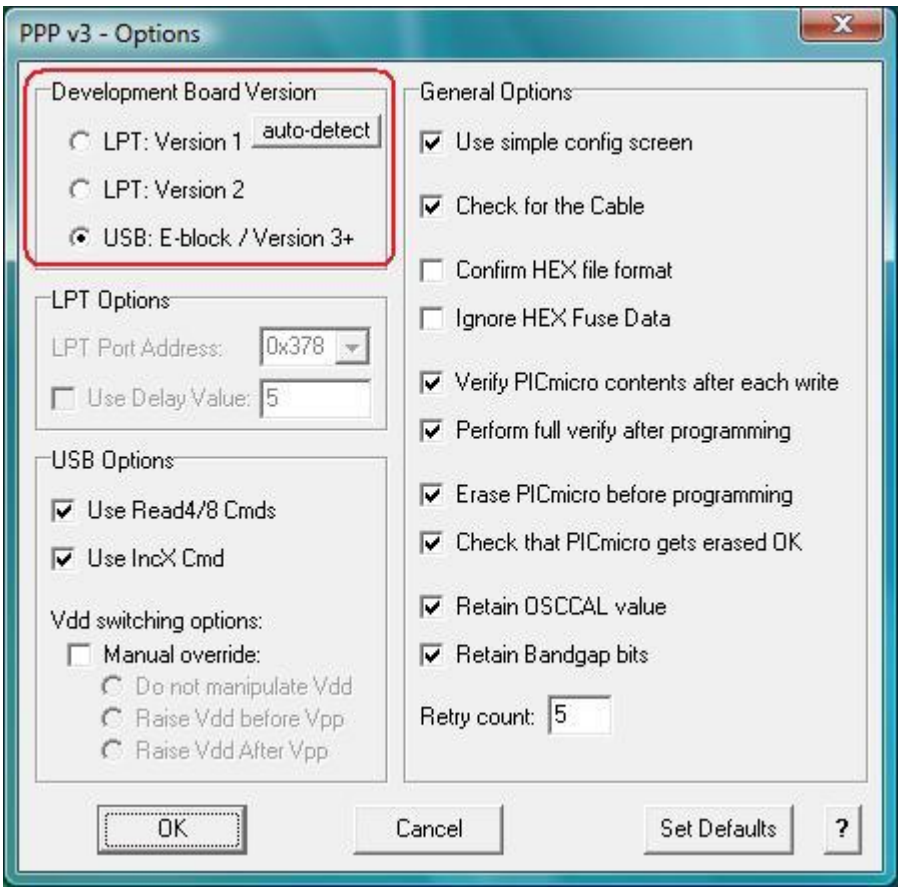


Рис. 3.44. Диалоговое окно настроек программатора

Кроме подключения программатора к порту USB компьютера, есть возможность использовать программатор, работающий с портом LPT. Для тех, кого интересует программатор, работающий с FlowCode, я советую заглянуть на страницу одного из сайтов:

http://microsin.ru/content/view/820/1/

А рассказ будет продолжен о…

Встроенные модули USART и PWM

Микроконтроллер PIC16F628A,как, впрочем, и другие, имеет встроенный модуль для поддержки сетевой работы – USART.

Используем компонент **RS232** панели дополнительных компонентов для программы, поддерживающей работу с USART.

Рис. 3.45. Компонент RS232 для работы с USART

После того, как RS232 добавлен (достаточно «щелкнуть» по нему мышкой), можно добавить в программу программный компонент **Component Macro**. К этому моменту все выглядит как-то так:

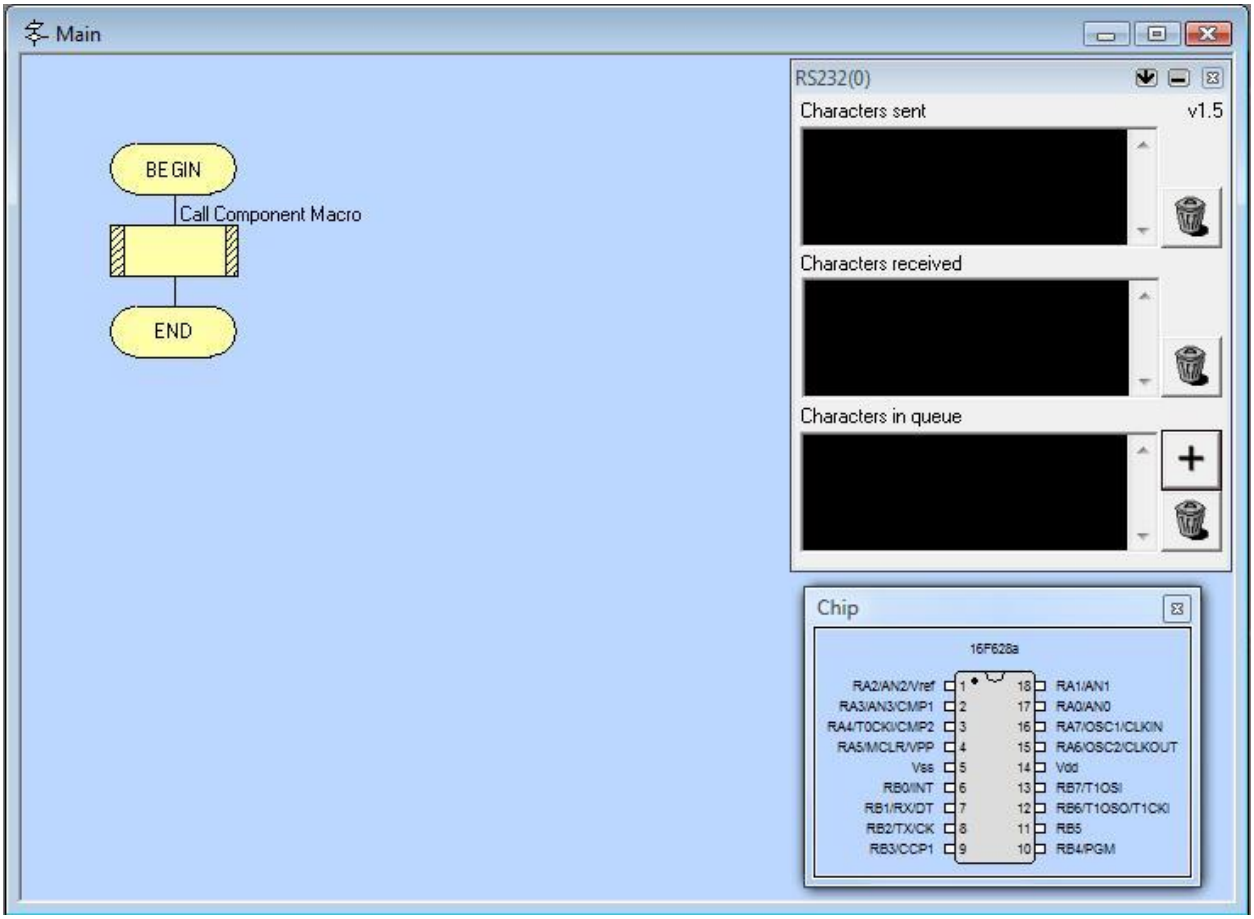


Рис. 3.46. Начало работы с программой, использующей USART

Перед тем, как продолжить создание программы, есть резон решить, а что мы хотим от программы? Пусть будет так: мы получаем по сети литеру N, что устанавливает (в высокое состояние) вывод 0 порта A; мы получаем по сети литеру F, что сбрасывает (устанавливает в состояние низкого уровня) вывод 0 порта A.

Двойной щелчок по **Component Macro** открывает диалоговое окно свойств этого компонента. Выбрав (выделив) получение символа, мы с помощью кнопки ***Variables*** создадим переменную *inp* и зададим параметр 0.

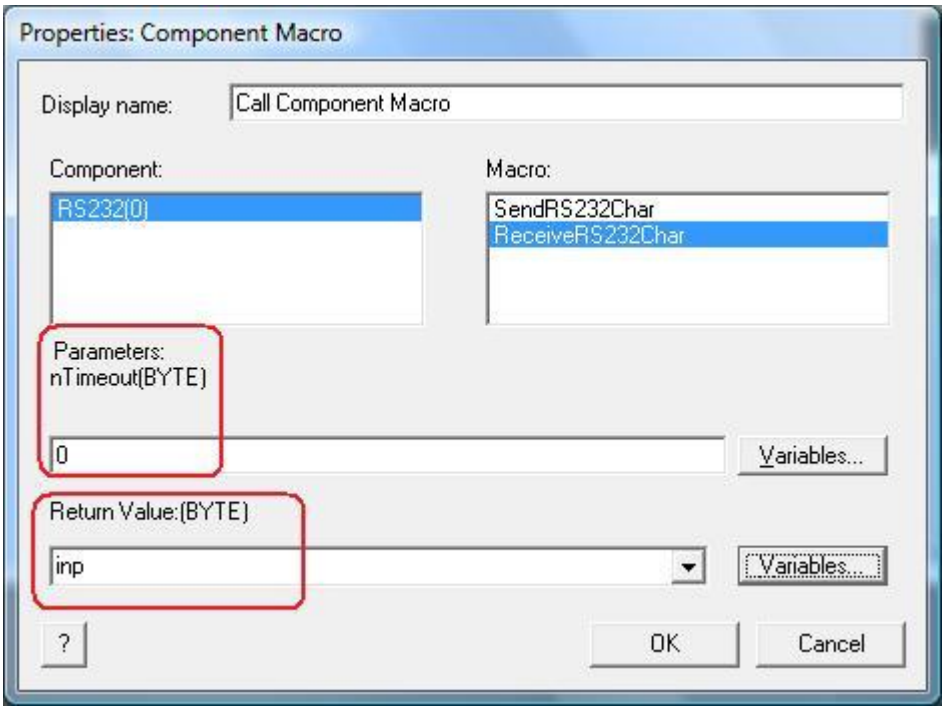
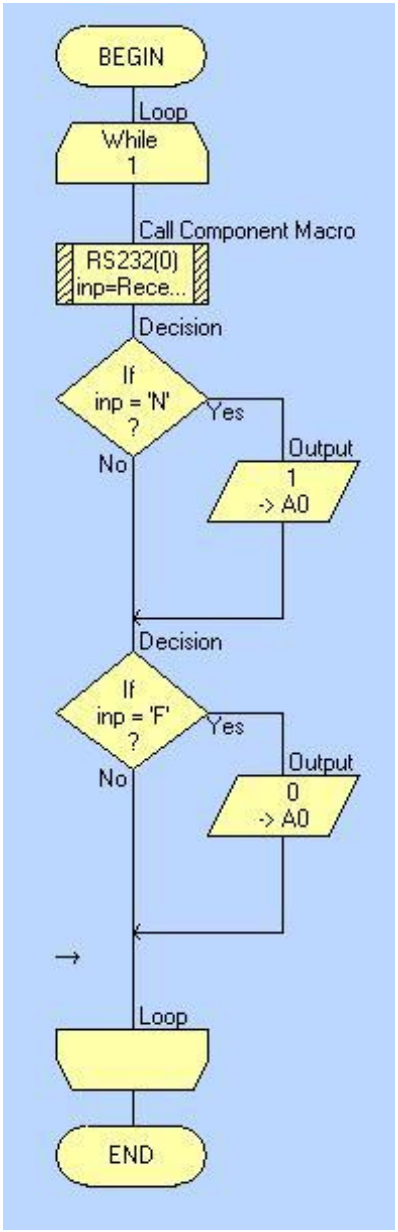


Рис. 3.47. Диалоговое окно **Component Macro** для **RS232**

Задав свойства, «погрузим» всю программу в бесконечный цикл и используем условное ветвление для выполнения ответных действий на приход управляющих символов.

Напомню, что с приходом символа «N» микроконтроллер должен установить высокий уровень на выводе 0 порта A. К этому выводу можно подключить, например, реле, которое будет своими контактами включать свет; можно подключить, например, светодиод оптопары для управления триаком, и в этом случае обойтись без реле. С приходом же символа «F» микроконтроллер должен установить низкий уровень на выводе, выключая нагрузку.

Об оптопаре я упомянул по той причине, что для отладки удобно использовать линейку светодиодов, а начинающим, привыкшим понимать все буквально, бывает трудно отрешиться от мысли, что они что-то недопонимают: написано, что микроконтроллер должен включить свет, а весь свет от светодиода. Кстати, с появлением новых типов светодиодов и это не исключено.



Подпрограмма **Component Macro**, принимающая данные по протоколу RS232, возвращает переменную *inp*.

Первое ветвление программы происходит в зависимости от того, какое значение принимает эта переменная. Если она равна символу «N», то в регистр порта A записывается 1, иначе программа проходит дальше.

Второе ветвление происходит тогда, когда переменная *inp* принимает значение «F». В этом случае в регистр порта A записывается 0.

Цикл *While* в данном случае не имеет определенного условия выхода, то есть, становится бесконечно выполняемым.

Рис. 3.48. Программа отклика на приход символов по сети

Чтобы проверить работу программы, добавив с панели дополнительных компонентов линейку светодиодов, следует использовать кнопку «**+**», отмеченную на рисунке ниже, после запуска программы (**Run->Go/Continue** основного меню, или функциональная клавиша F5 клавиатуры, или кнопка ***Run*** инструментальной панели).

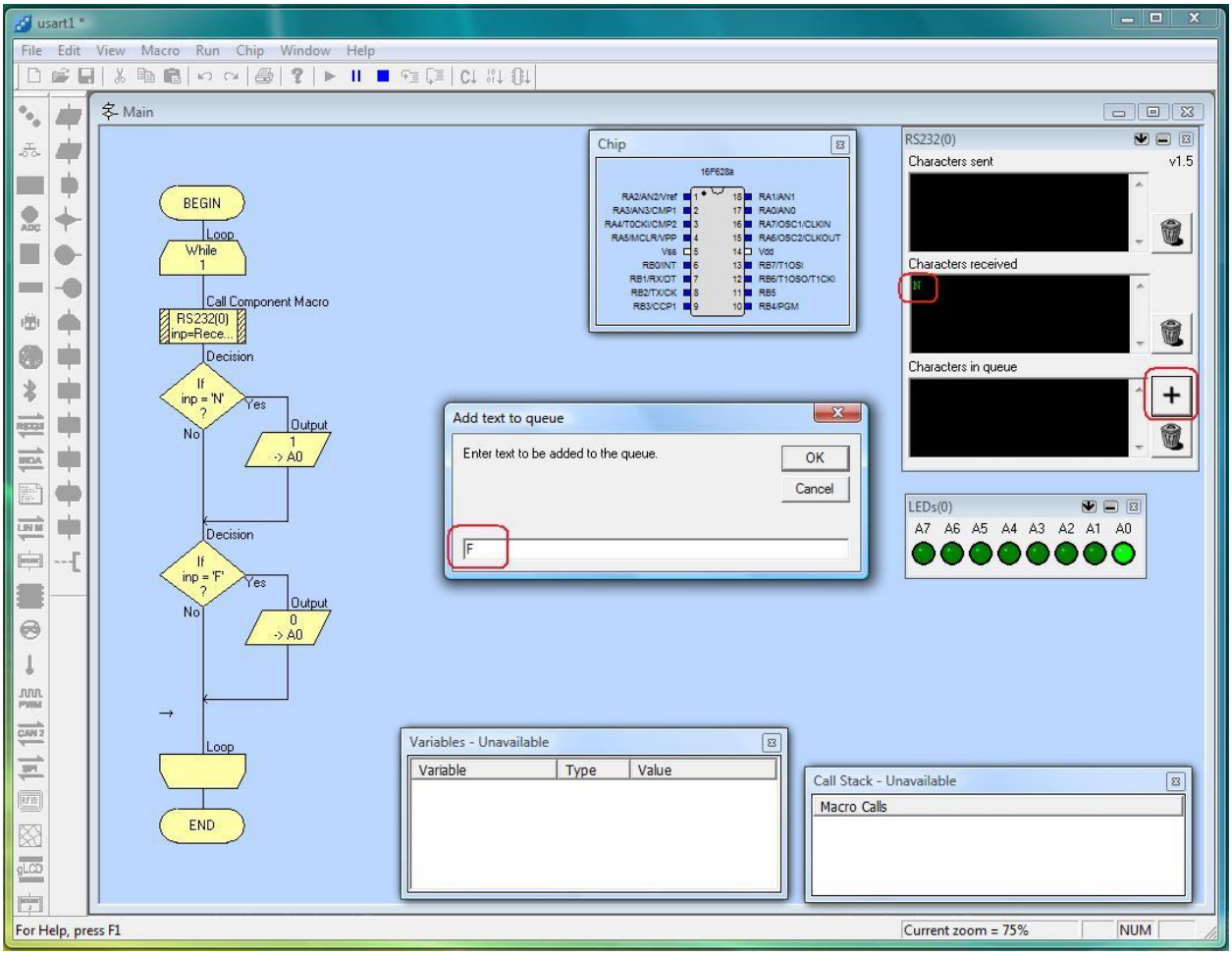


Рис. 3.49. Отладка программы

Нажав кнопку «*+*» панели *Characters in queue*, вы попадаете в диалог ввода символа. После ввода с клавиатуры нужного символа, что подтверждается кнопкой ***OK*** диалога ввода, этот символ появится в окне панели *Characters Received*, а светодиод A0 загорится. Если же повторить эту операцию, введя символ «F», то светодиод погаснет.

Создадим еще одну программу, которая не получает символы по USART, а отправляет их.

Программа простая: при каждом нажатии на кнопку микроконтроллер попеременно отправляет в сеть символы «N» и «F».

Мы знаем, что для обслуживания кнопки, соединенной с выводом порта, назначенным на вход, следует использовать переменную. Выберем переменную *inp*, которая будет обслуживать нулевой вывод порта B. Чтобы организовать переключение, используем переменную *flag*, которая будет менять свое значение с 0 на 1 при каждом нажатии на кнопку. И для отправки символа используем переменную *out*. Эти переменные можно сразу ввести в программу, но можно создавать их по мере продвижения в написании программы.

Итак. Создав новую программу, сразу добавим в нее цикл (программный компонент **Loop**). Но перед циклом, используя программный компонент **Calculation**, зададим переменной *flag* нулевое значение. Внутри бесконечного цикла будем (с помощью программного компонента **Input**)

опрашивать вход B0 и разветвим программу: ничего не будем делать, если состояние кнопки не изменилось, и будем делать «все остальное», если кнопка была нажата.

Что «остальное»? В первую очередь будем менять значение переменной *flag*: используем такую конструкцию из двух логических операций `flag = ( NOT flag ) AND 1`, которую запишем с помощью программного компонента **Calculation**. Если бы в программе FlowCode был булев тип данных, было бы достаточно операции инверсии (отрицания `NOT`), но нам не важно, с помощью скольких операций мы выполним задуманное. Тем более что можно было бы использовать и запись `flag = NOT flag`, только в этом случае переменная приобретала бы два значения: 0 и 255.

И нам остается только отправлять символ «N», если переменная *flag* равна 1, и символ «F», когда она становится равна нулю.

Как и в предыдущей программе, мы используем программный элемент **Component Macro** для RS232. Но на этот раз в его свойствах выберем отправку символа и используем переменную *out*.

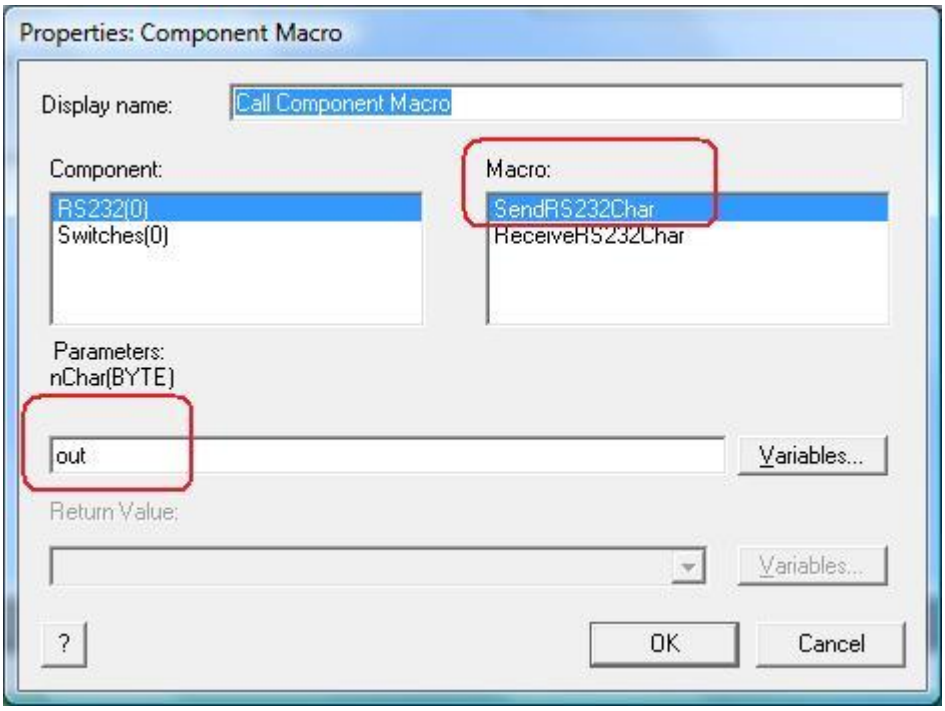


Рис. 3.50. Настройка компонента RS232 в диалоговом окне его свойств

Если запустить отладку программы в том виде, который она сейчас приняла, то… попробуйте, сами увидите, что произойдет. А, чтобы избежать этого, добавим паузу после каждой отправки символа.

Программа приобретает такой вид:

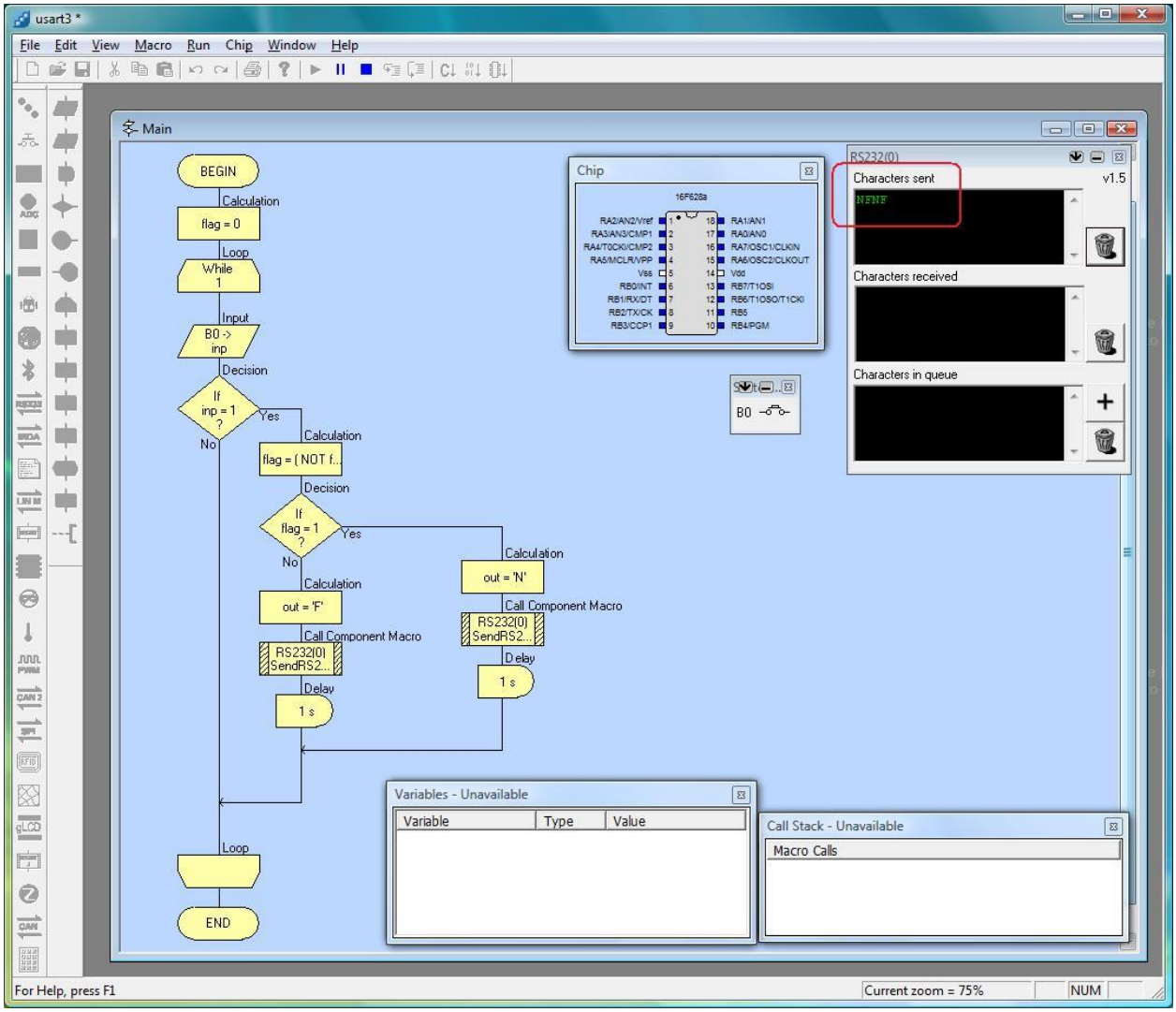


Рис. 3.51. Вторая программа, использующая USART

На рисунке отмечено, какие символы отправляются в сеть при каждом нажатии на кнопку B0.

Я полагаю, что, прочитав предыдущие главы, вы уже вполне можете справиться с написанием программы на языке Си и без использования программы FlowCode. Хотя согласитесь, что использовать ее очень удобно.

Оставим пока эти две простенькие программы, о которых хотелось бы добавить еще несколько слов, но позже. А сейчас несколько слов о модуле PWM, который есть в PIC16F628A. Модуль может использоваться для разных целей. Но я хочу привести пример программы, где меняется скважность импульсов на выходе 3 порта B, с тем, чтобы показать, как просто это можно сделать в программе FlowCode.

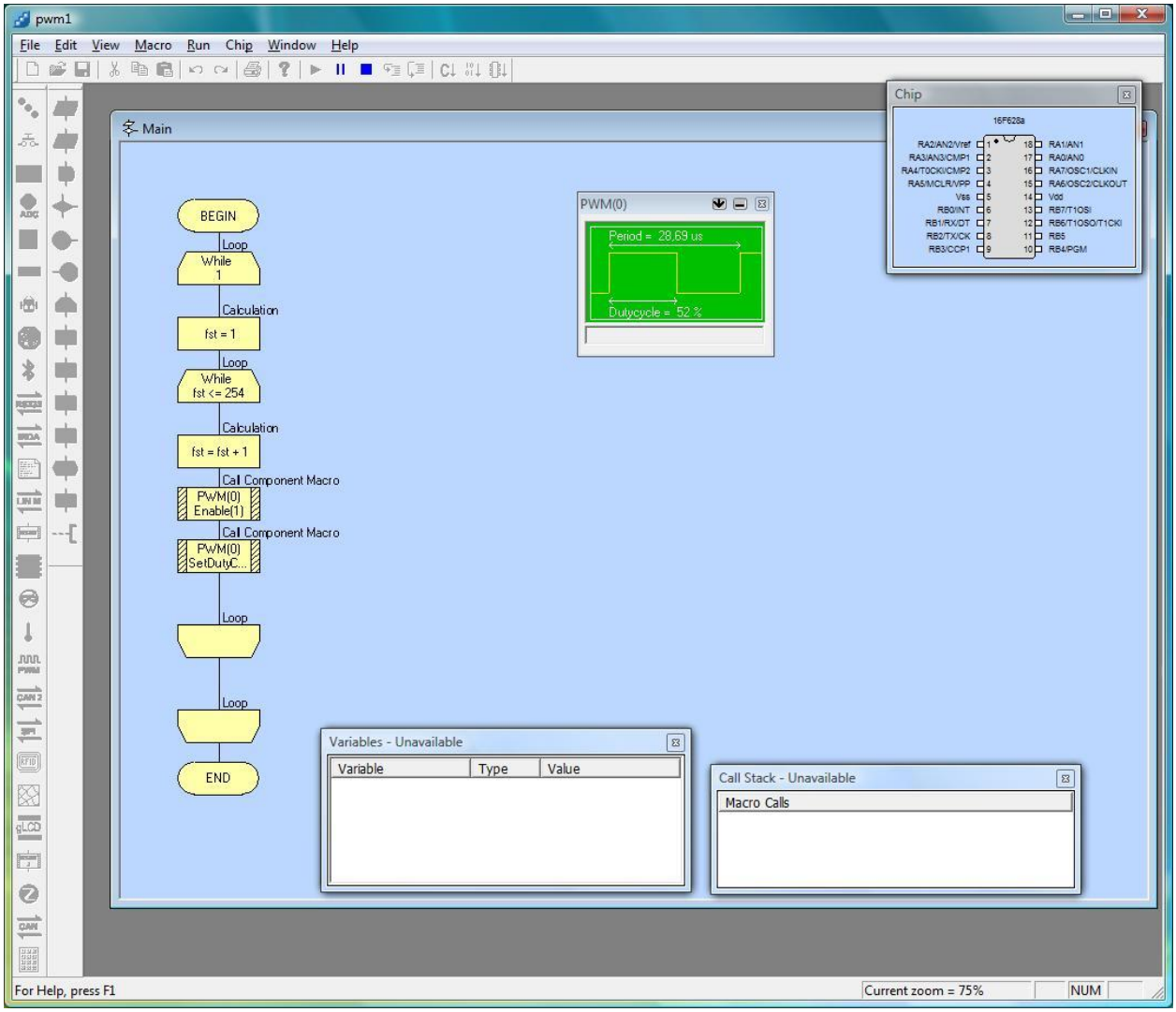
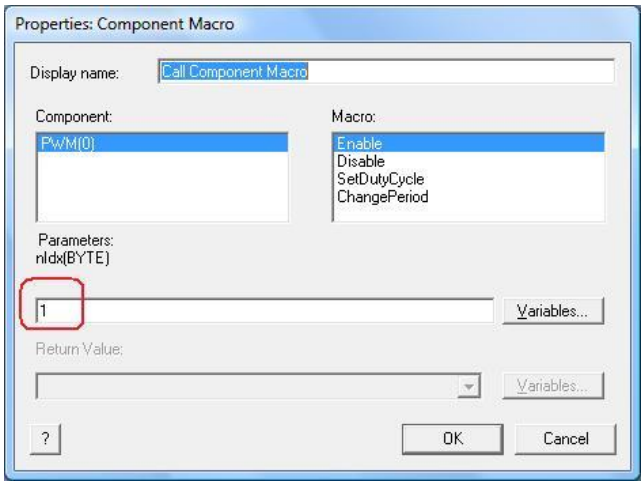


Рис. 3.52. Программа работы с ШИМ (широтно-импульсной модуляцией)

Вся программа, кроме двух циклов: бесконечного внешнего и условного внутреннего, – сводится к добавлению двух программных элементов **Component Macro** и компонента **Calculation**, где переменная *fst* увеличивается на единицу при каждом проходе внутреннего цикла. Первый **Component Macro** разрешает использование ШИМ.



Отмеченная на рисунке единица относится к номеру канала, поскольку есть модели, имеющие несколько каналов ШИМ.

Рис. 3.53. Диалоговое окно дополнительного компонента PWM

В следующем диалоговом окне (второго **Component Macro**) задается скважность импульсов.

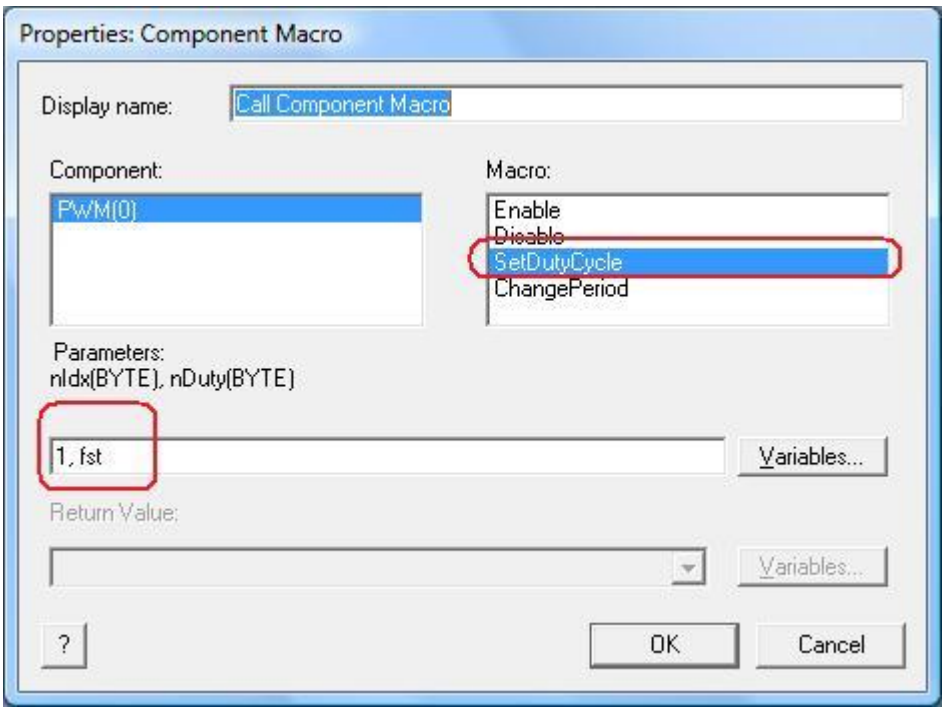


Рис. 3.54. Задание скважности импульсов через переменную *fst*

После запуска программы в окне компонента появится сигнал, плавно меняющий скважность импульса.

Таким образом, полная версия программы FlowCode позволяет быстро и эффективно разрабатывать устройства, базирующиеся на микроконтроллерах. Помимо этого можно проекты, разработанные для PIC-контроллера, импортировать в версию, предназначенную для работы с AVR-контроллерами, что еще больше расширяет возможности применения программы.

Однако это несколько уводит нас от основной темы рассказа – как использовать демо-версию FlowCode, чтобы научиться программированию на языке Си (или ассемблере). Вернемся к теме рассказа и рассмотрим несколько конкретных примеров.

Некоторые примеры программирования в среде FlowCode

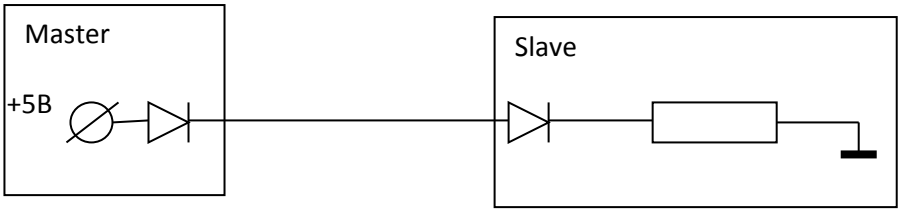
Первый пример

Первый пример прост, но полученное устройство вполне может оказаться полезным, когда покупка готового прибора по каким-то причинам не целесообразна.

Сегодня часто используют кабель из нескольких витых пар, который заканчивается разъемом RJ45. Разъем «обжимается» быстро, но всегда полезно проверить результат. Конечно, для устройства «прозвонки» проводов не обязательно использовать микроконтроллер. Но посмотрите, как просто сделать устройство, используя микроконтроллер.

Итак, сформулируем задачу: необходимо «прозвонить» кабель, в котором восемь проводов. Для индикации используем светодиоды, выбрав модель с падением напряжения 1.5-2 В на диоде. Конструктивно устройство будет состоять из двух блоков: основной блок (Master) и вспомогательный (Slave). Оба блока снабжены гнездами для разъема RJ45 и линейкой светодиодов. При включении прибора светодиоды последовательно зажигаются на обоих блоках при исправных проводах и соединении. Если провода перепутаны, то это должно отобразиться в последовательности зажигания светодиодов или в отсутствии свечения.

Немного о схеме. На первый взгляд схема крайне проста:



Но есть один «подводный камень»: для самого простого решения по «прозвонке» восьми проводов не хватает девятого, общего провода. В этом случае микроконтроллер окажет неоценимую услугу, поскольку можно модифицировать включение светодиодов следующим образом:

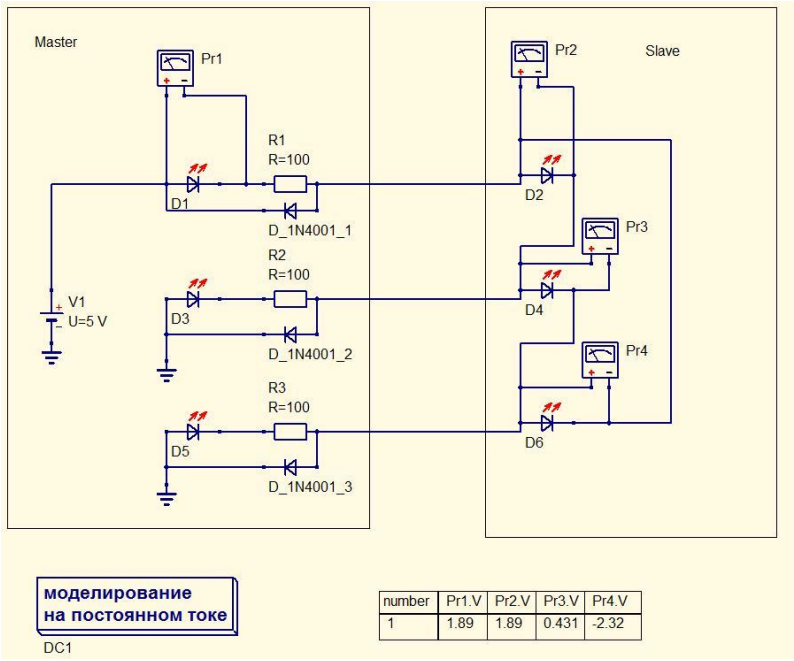
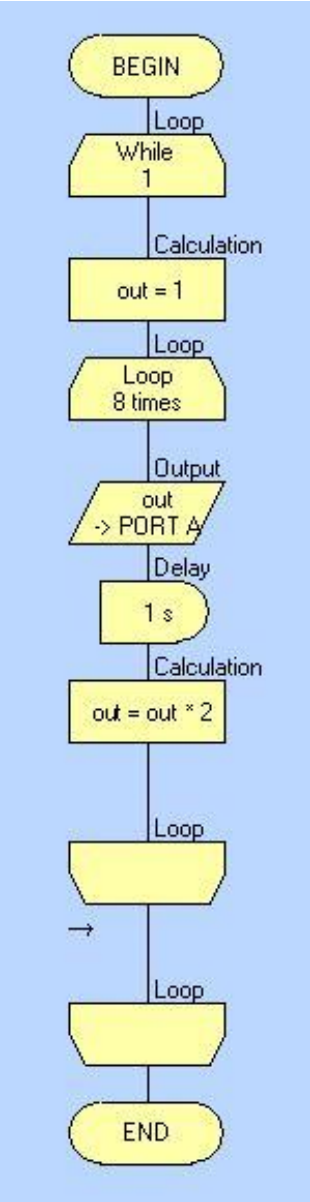


Рис. 4.1. Модификация схемы для трех проводов

Если исключить из схемы выше (для трех проводов) приборы Pr1-4, то смысл схемы ясен, а роль источника питания V1 будет играть микроконтроллер. И программа, обслуживающая работу микроконтроллера, тоже проста: на один из выходов микроконтроллера подается высокий уровень, а на остальные низкий. Вот вид программы в FlowCode:



В бесконечном цикле (первый компонент **Loop**) мы восемь раз (второй компонент **Loop**) отправляем в порт А число, которое начинается с 1 (первый компонент **Calculation**), а после паузы в 1 секунду умножается на 2 (второй компонент **Calculation**).

Для перевода программы на язык Си у нас все есть: вывод числа, цикл. Даже с паузой при использовании компилятора HI-TECH мы разобрались. Единственное отличие от алгоритма, записанного в FlowCode – двухступенчатое получение нужной длительности паузы.

Напишем так:

```
#include <htc.h>
#define _XTAL_FREQ 4000000
void main()
{
char out;
char i;
char k;
CMCON = 0x07;
      while (1) {
            out = 1;
            for (i=0;i<8;i++) {
                  TRISA = 0x00;
                  PORTA = out;
                  for(k=0;k<10;k++) {
                  __delay_ms(100);
                  }
                  out = out*2;
            }
      }
}
```

Рис. 4.2. Программа обслуживания «прозвонки»

Теперь эту программу можно использовать в среде разработки MPLAB: проверить, оттранслировать.

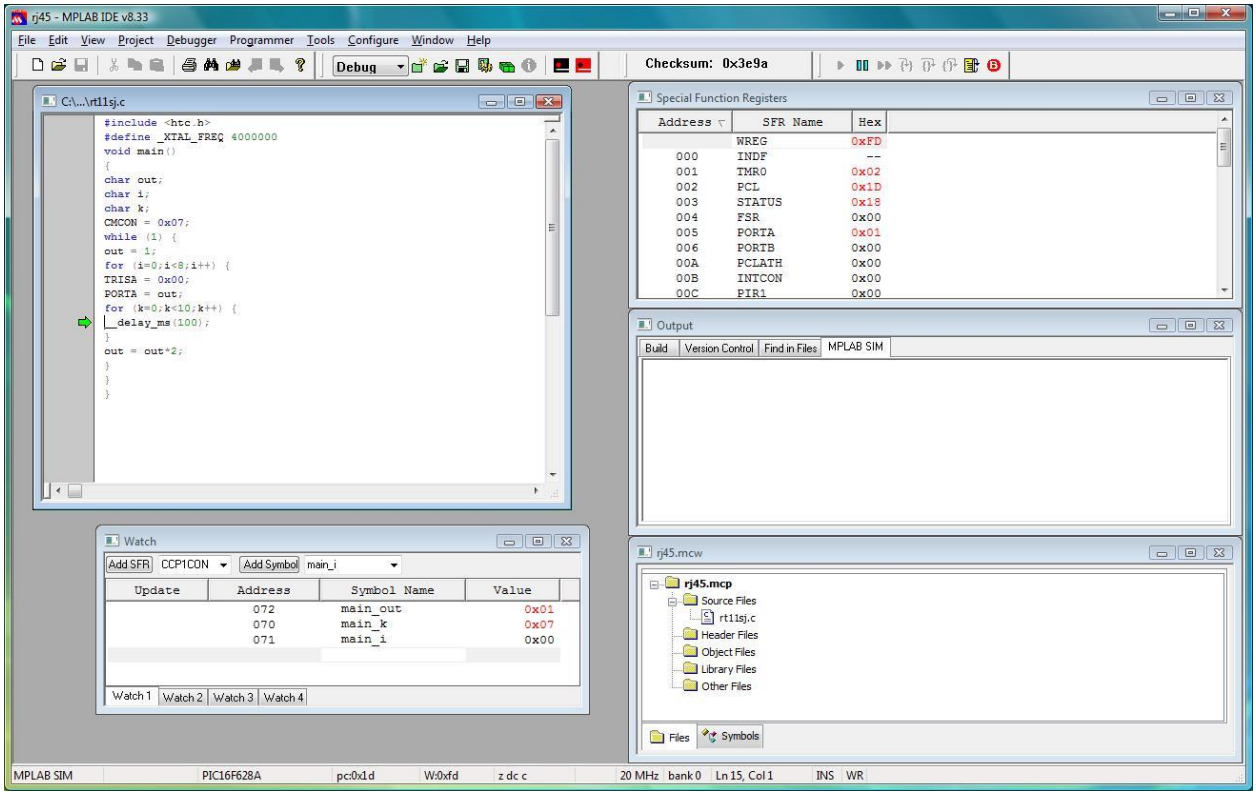


Рис. 4.3. Поверка программы в среде MPLAB

Или можно использовать для проверки работы схемы другую программу.

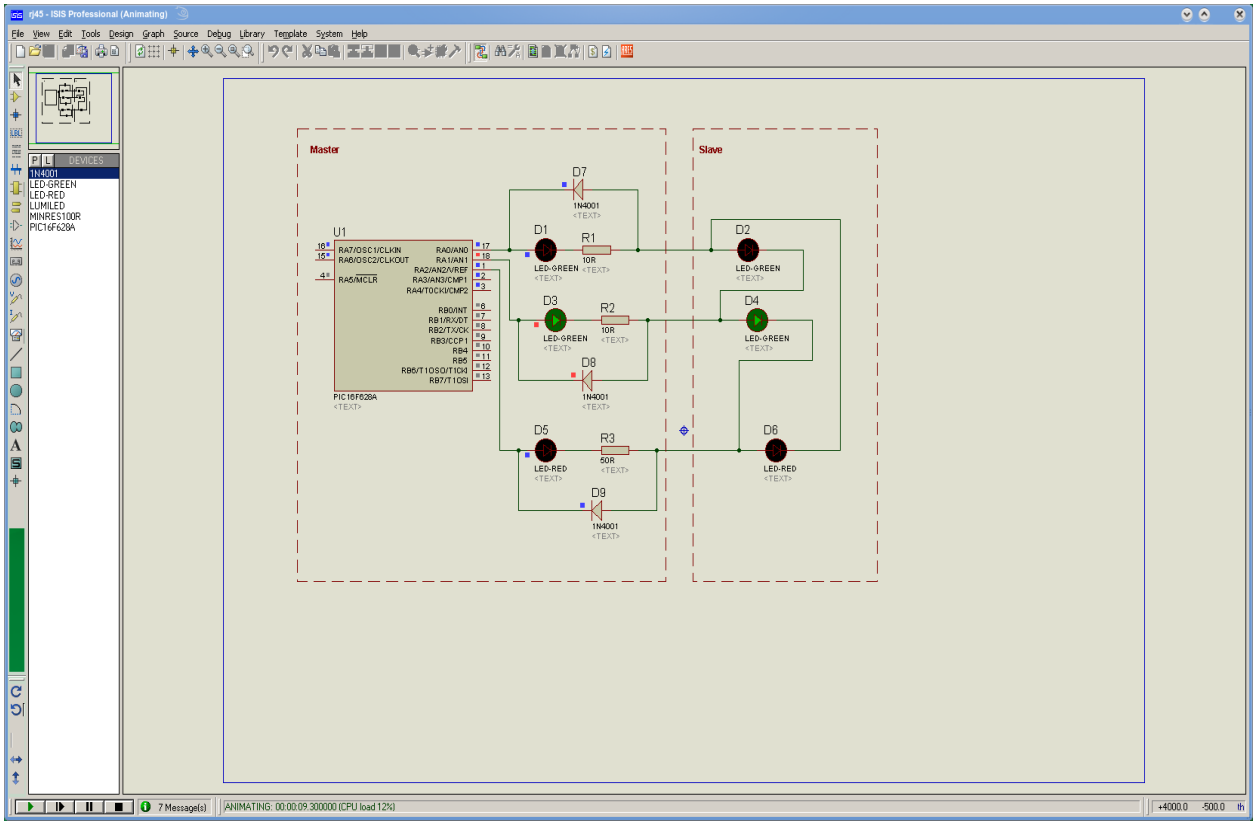


Рис. 4.4. Проверка схемы в Proteus

Или проверить работу схемы на макетной плате, что даст самый убедительный результат. Нет, не в проверке работы программы, а всего устройства в целом. Например, даст ответ на вопрос, удобно ли работать с устройством, если один или несколько проводов в обрыве, если один или несколько проводов перепутаны и т.п.

Многие любители, особенно «бывалые», пренебрежительно относятся к простым программам и инструментам, облегчающим создание устройств на базе микроконтроллеров. Между тем, сложность конструкции плохо соотносится с ее полезностью, и имеет смысл только тогда, когда без этой сложности не обойтись.

Не могу сказать, нужно ли кому-нибудь устройство, описанное выше. Но мне хотелось показать, что, зная несколько простых фрагментов на языке Си, полученных с помощью демо-версии программы FlowCode, можно создавать вполне реальные, жизнеспособные устройства. И это главное.

Второй пример

Этот пример построим на основе первого: добавим к устройству одну кнопку, нажатием которой ускорим обход всех проводов. Если прежде каждый светодиод был включен в течение секунды, то при нажатой кнопке он будет включаться только на одну миллисекунду. Что при этом изменится в работе устройства? Внешне это выразится в том, что визуально все светодиоды будут гореть одновременно, но слабее, чем при отжатой кнопке.

Вспомним, как выглядит фрагмент кода на языке Си для ввода:

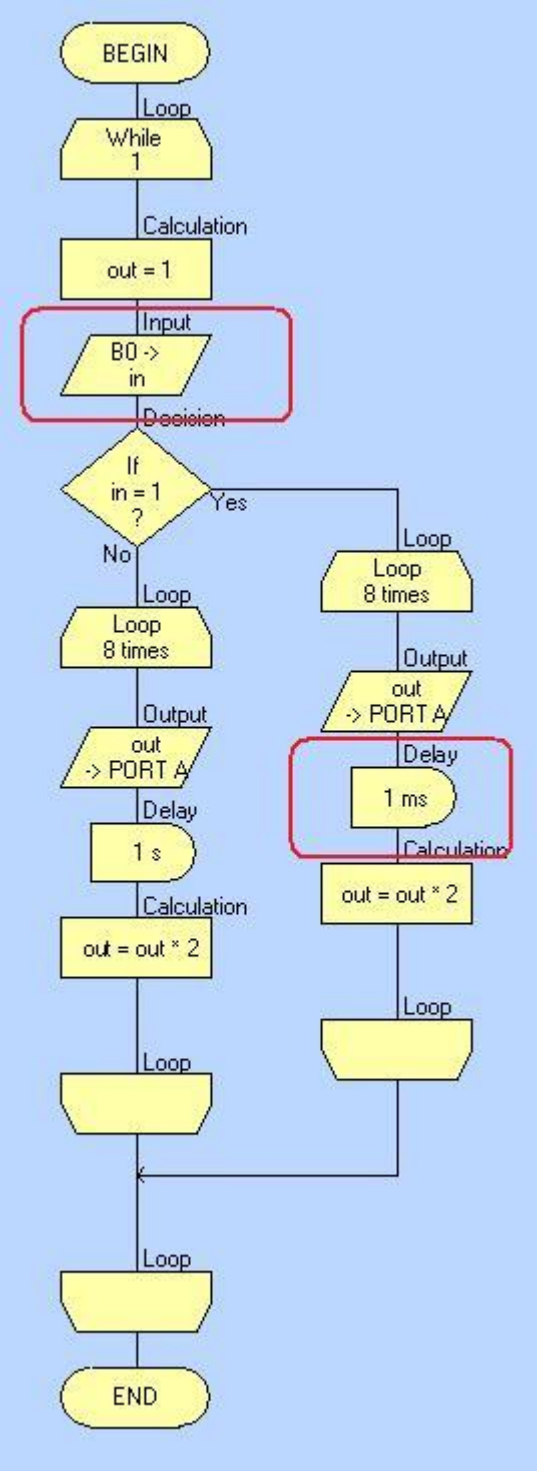
```
char FCV_IN;
void main()
{
        //Input: B0 -> in
        trisb = trisb | 0x01;
        FCV_IN = (portb & 0x01);
}
```

Для ввода (программный компонент Input) понадобится переменная, в данном случае `char FCV_IN`. Для ввода используется единственный вывод RB0 порта B, что отражено в строке `trisb = trisb | 0x01`. Переменной, обслуживающей ввод, присваивается значение, прочитанное в RB0.

И вспомним, как выглядит условное ветвление:

```
if (FCV_IN == 1)
        {
                // Некое действие

        } else {
                // Другое действие
        }
```

Вся программа в среде разработки FlowCode имеет вид:



Кроме добавления компонента **Input**, свойства которого соответствующим образом изменены, и компонента ветвления **Decision**, все остальное сводится к копированию предыдущей программы во вторую ветку и изменению длительности паузы.

Думаю, вам интереснее самостоятельно написать эту программу на языке Си и отладить ее в программе MPLAB.

Но несколько слов о смысле, который можно придать новой программе, вернее, модификации старой. Когда светодиоды последовательно зажигались и горели секунду, легко заметить обрыв одного из проводов. Но, если разъем обжат плохо: соединение есть, а переходное сопротивление достаточно большое, – то при использовании первого режима работы устройства это не будет бросаться в глаза. А когда все светодиоды горят, они горят с одинаковой яркостью, если все в порядке. Если же один (или несколько) проводов имеют плохой контакт с выводом разъема, соответствующий светодиод (при выбранной конструкции два светодиода) будет светиться слабее остальных. Что сразу бросится в глаза.

Конечно, справедливость этой идеи следует проверить на макетной плате, но мне хотелось показать совсем другое – то, как легко, практически без переделок схемы, можно расширить функциональность устройства.

И что для этого можно использовать простые фрагменты языка Си, полученные в FlowCode.

Рис. 4.5. Модификация предыдущей программы

Пример третий

Рассказывая о встроенных в микроконтроллер модулях, я предложил рассмотреть простое устройство, принимающее по сети команду и выполняющее простое действие – включение реле, например. И позже я собирался добавить несколько слов. Вот это позже и настало.

Полнофункциональная версия FlowCode позволяет быстро добавить работу с такими модулями, как USART или PWM. Не так сложно, посмотрев пример, который устанавливается с компилятором HI-TECH, понять, как следует на языке Си написать эту часть работы контроллера. Программа и в этом случае получается «простенькой».

Однако, вспоминая как «подвисали» компьютерные концентраторы в нашей сети, когда достаточно было выключить его и включить вновь, я думаю, что применение устройства, созданного из «простенькой» программы для выполнения этого «обряда» не на месте, а нажатием одной кнопки в региональном офисе, было бы весьма полезно. Или, скажем, свет в подъезде, который горит всю ночь. Программа, о которой идет речь, выполняло простое действие – нажатием кнопки в сеть отправлялась команда, которая включала, положим, свет. Любой электрик нарисует вам эту схему без применения контроллеров. Но реализуйте эту схему не для одной лампочки, а для лампочек всего подъезда, да еще так, чтобы можно было выключить свет на своем этаже. Здесь преимущества «простенькой» программы очевидны.

А говорю я об этом по той причине, что хочу привести пример еще одной «простенькой», но полезной программы для микроконтроллера.

Очень часто устройство на базе микроконтроллера имеет клавиатуру. С целью экономии выводов кнопки можно включить «матрицей». При четырех кнопках, как на рисунке ниже, мы не получаем существенного выигрыша, но, используя восемь выводов, можно получить клавиатуру с шестнадцатью клавишами. И здесь выигрыш очевиден.

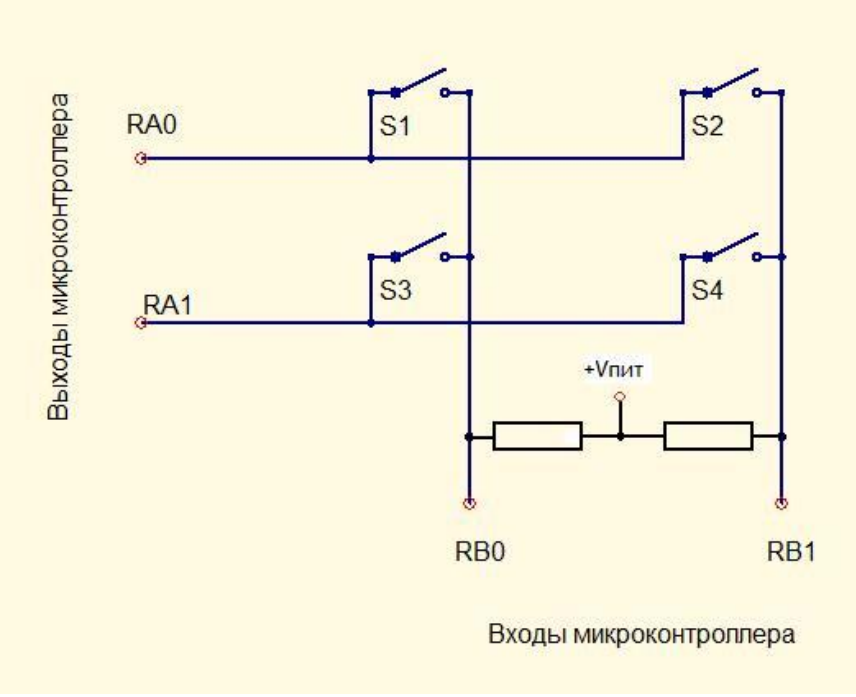
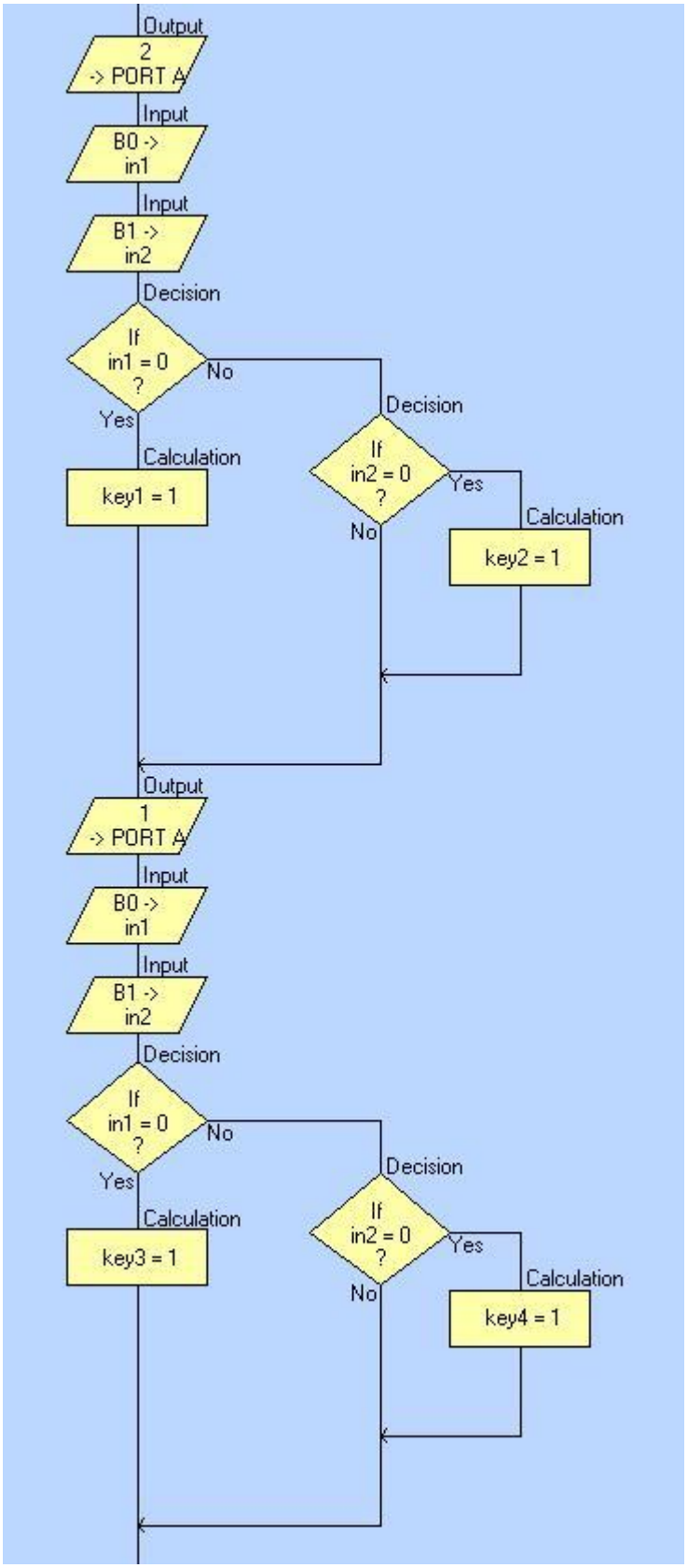


Рис. 4.6. «Матричное» включение четырех клавиш

Попеременно подавая низкий уровень на выходы микроконтроллера RA0 и RA, определяя состояние входов RB0 и RB1, можно определить состояние четырех кнопок S1-S4. Для реализации программы достаточно уже известных фрагментов на языке Си. «Блок-схема» программы может выглядеть так:



«За кадром» остался бесконечный цикл, в котором работает опрос клавиатуры. Программа может быть построена и иначе. Например, с использованием подпрограмм.

Результат работы не заметен – переменные программы key1 - key4 приобретают некоторое значение, которое невозможно проверить на макетной плате.

Более того, программу трудно проверить в FlowCode из-за особенностей свойств компонента выключатели.

Но если заменить присваивание значений переменным на изменение состояния выходов, то работу программы можно проверить на макетной плате.

Реальная программа, конечно, выполняет какие-то действия по нажатию кнопок, что позволяет проверить ее работу.

Тем не менее, будет полезно записать программу на языке Си и постараться проверить ее работу в MPLAB или Proteus.

В полной версии FlowCode работает дополнительный компонент KeyPad. Для работы с клавиатурой можно использовать этот компонент.

Рис. 4.7. «Блок-схема» программы обслуживания клавиатуры

Как и другие компоненты, **KeyPad** добавляется в программу щелчком по кнопке инструментальной панели дополнительных компонентов. После этого достаточно добавить программный компонент **Component Macro**, в диалоговом окне свойств которого выбрать, например, в качестве возвращаемого значения номер клавиши.

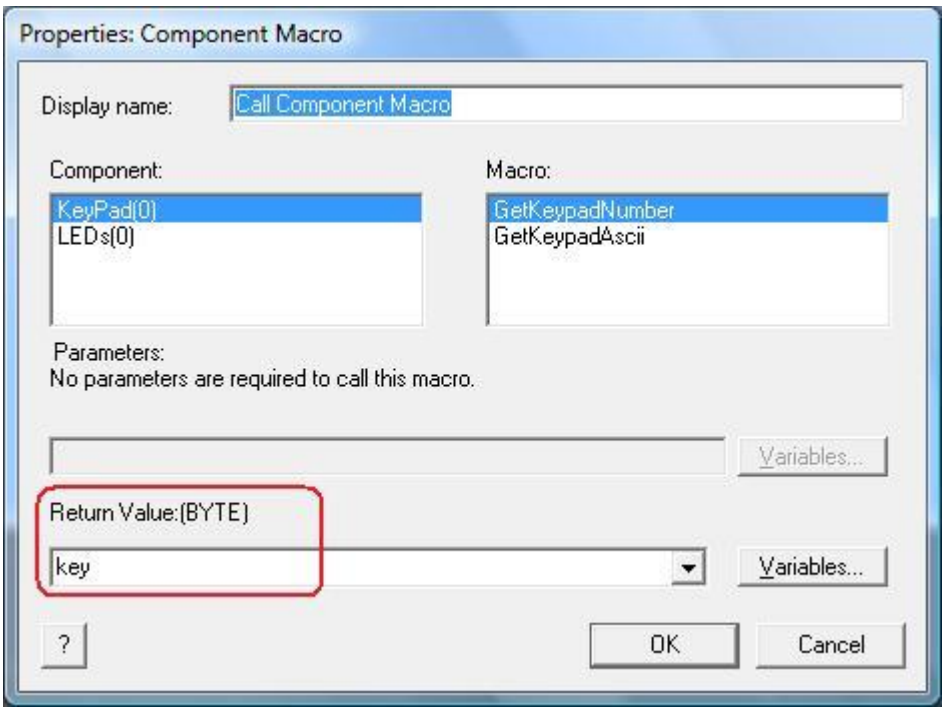


Рис. 4.8. Диалоговое окно программного компонента для **KeyPad**

Этот номер будет присваиваться созданной нами переменной *key*. Дальнейшие действия зависят от наших намерений. Простейшая программа проверки клавиатуры может иметь следующий вид:

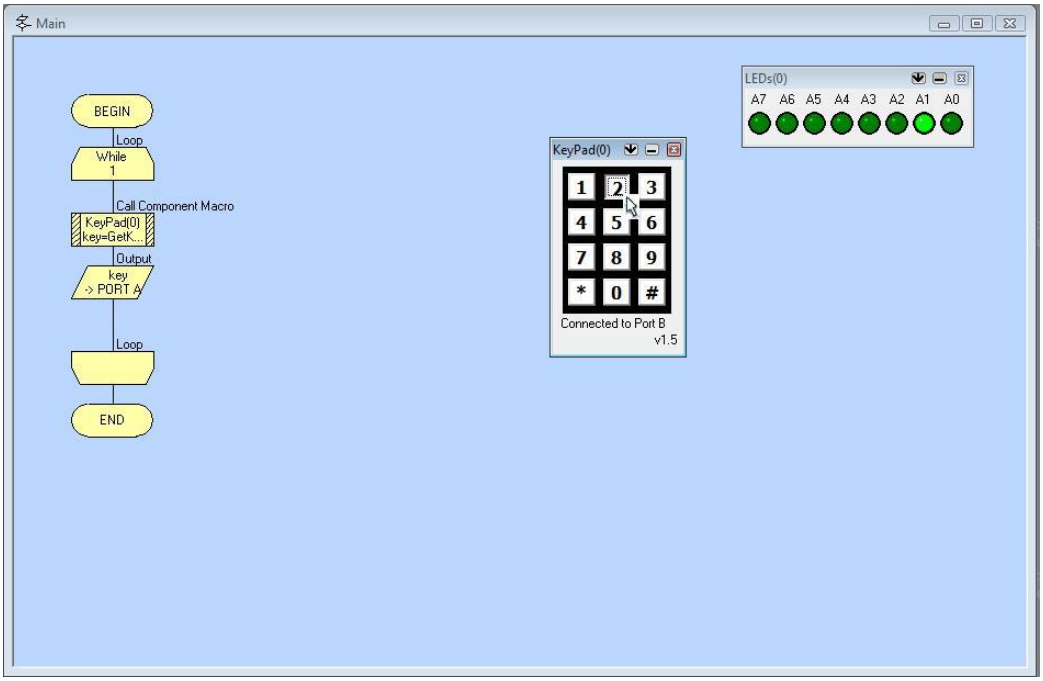


Рис. 4.9. Работа клавиатуры в программе FlowCode

Пример четвертый

Много лет назад мой знакомый спросил, его тоже кто-то попросил об этом, как удобнее сосчитать количество посетителей. По тем временам микроконтроллеры были достаточно дороги, да и требовали много из того, что не у всякого любителя найдется. Но сегодня я точно порекомендовал бы использовать микроконтроллер для создания подобного счетчика.

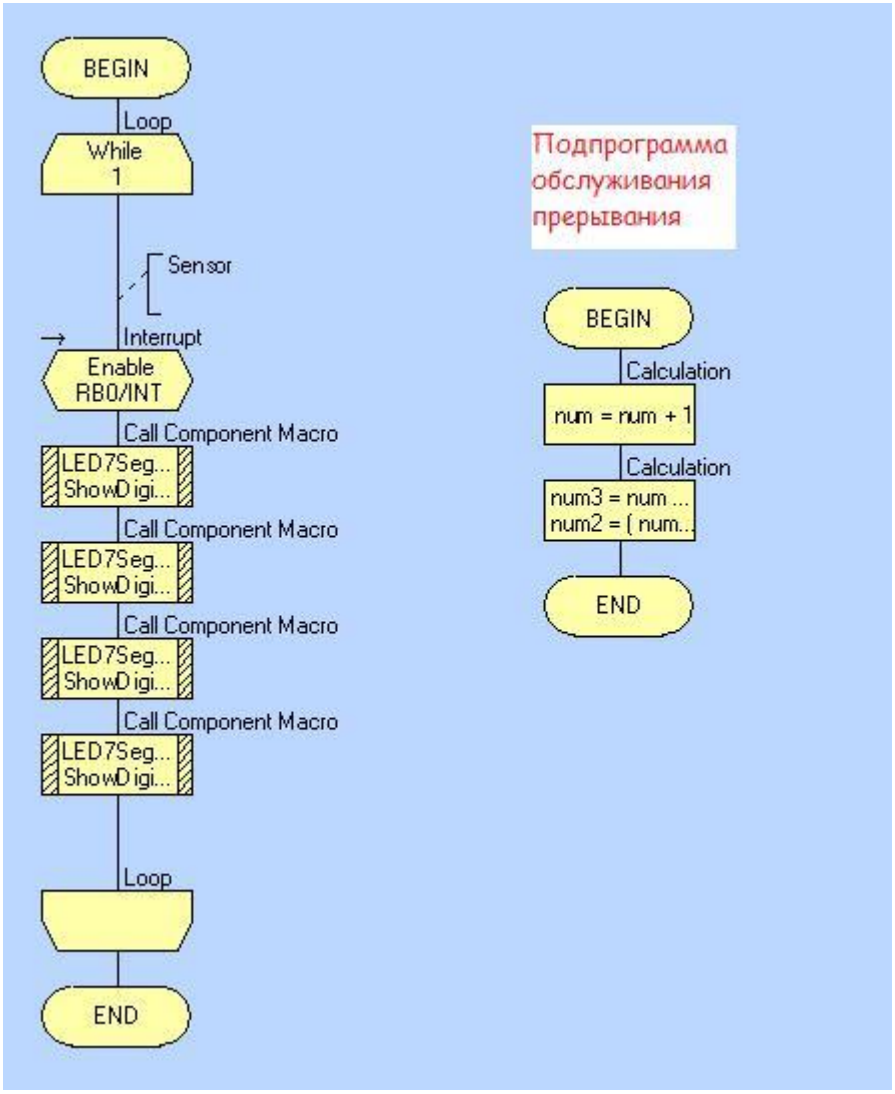


Рис. 4.10. Программа счета срабатывания датчика на входе RB0

Программа FlowCode предлагает четырехразрядный семисегментный индикатор в качестве одного из дополнительных компонентов. Открыв свойства вызова подпрограммы обслуживания этого индикатора, можно увидеть:

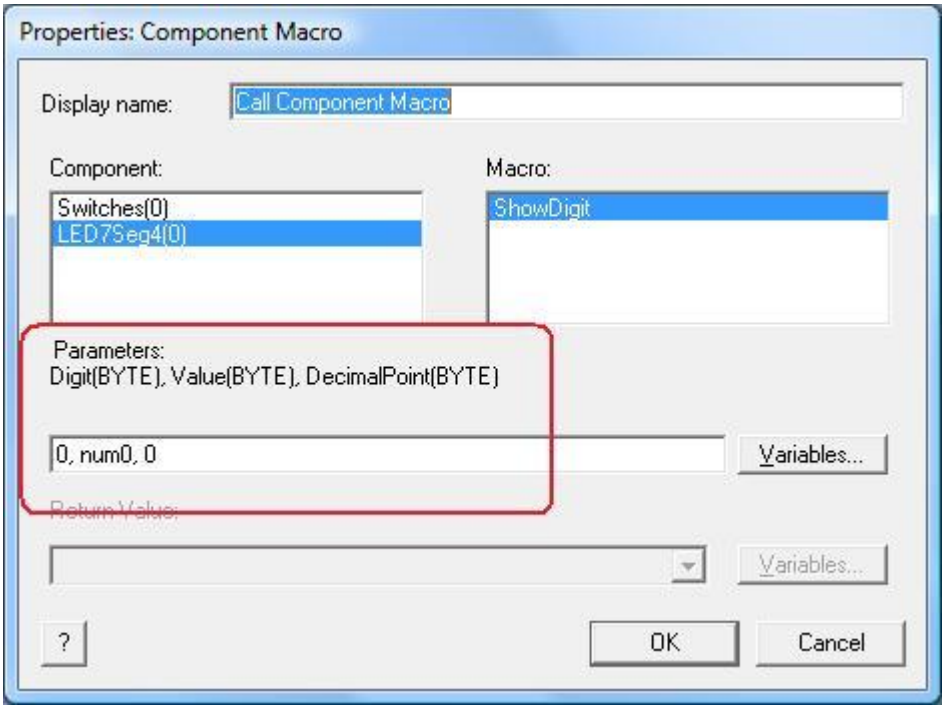


Рис. 4.11. Диалоговое окно свойств подпрограммы индикатора

Первый параметр – номер разряда, второй – значение, третий – положение точки.

При использовании простейшего герконового датчика, установленного на двери, вся тяжесть работы ляжет на создание подходящего корпуса устройства. Но, индикаторы бывают двух типов: с общим анодом, что представлено в программе FlowCode, и с общим катодом. Программа FlowCode, похоже, не позволит применить такой индикатор. В этом случае знания, полученные при изучении языка Си с помощью программы FlowCode, позволят вам быстро преобразовать готовый код к нужному виду – заменить высокий уровень на общих анодах на низкий, а низкий уровень сегментов высоким.

Фрагмент, разрешающий прерывание, выглядит так:

```
 //Interrupt initialisation code
 option_reg = 0xC0;
//Interrupt: Enable RB0/INT
       option_reg.INTEDG=1;
       intcon.GIE=1;
       intcon.INTE=1;
```

Прерывание:

```
void interrupt(void)
{
 if (intcon & (1 << INTF))
 {
       FCM_display();
       clear_bit(intcon, INTF);
 }
}
```

А функция `FCM_display()` в обслуживании прерывания:

```
void FCM_display()
{
    //Calculation:  num = num + 1
    FCV_NUM = FCV_NUM + 1;

    //Calculation:
    //  num3 = num / 1000
    //  num2 = (num - num3 * 1000 ) / 100
    //  num1 = (num - num3 * 1000 - num2 * 100 ) / 10
    //  num0 = (num - num3 * 1000 - num2 * 100 - num1 * 10 )
    FCV_NUM3 = FCV_NUM / 1000;
    FCV_NUM2 = (FCV_NUM - FCV_NUM3 * 1000 ) / 100;
    FCV_NUM1 = (FCV_NUM - FCV_NUM3 * 1000 - FCV_NUM2 * 100 ) / 10;
    FCV_NUM0 = (FCV_NUM - FCV_NUM3 * 1000 - FCV_NUM2 * 100 - FCV_NUM1 * 10);
}
```

И заметьте, если вы поспешили и сделали печатную плату, то вам не понадобится вносить изменения в монтаж, достаточно изменить программу. А если вы еще раз поспешили и сделали вторую печатную плату, то немного переделав программу, вы можете использовать устройство, например, для велосипеда. Конечно, датчик, закрепленный на колесе, потребуется, скорее всего, другого типа, но посчитывая количество оборотов колеса, зная длину его окружности, вы можете отображать на индикаторе пройденный путь. Или, еще раз изменив программу, можете отображать скорость движения. И ничто не помешает вам добавить одну кнопку для переключения этих режимов работы. Само устройство остается прежним. В этом проявляется одно из самых важных свойств микроконтроллера – меняя программу, устройство можно превратить в другое, не менее полезное.

В этом примере можно отметить еще две важные детали – полная версия программы FlowCode позволяет работать быстрее, чем другие среды разработки; а знание языка программирования Си помогает быстрее и легче справляться с возникающими осложнениями.

В этом смысле трудно переоценить возможности, предоставляемые программой FlowCode – увидеть необходимые действия, которые образуют достаточно понятную конструкцию, а затем уже обратиться к коду на языке программирования. Думаю, любой специалист согласится, что рисунок схемы, использующий «прямоугольник» микросхемы с заданными свойствами, гораздо понятнее, чем схема, где все микросхемы нарисованы с полным внутренним содержанием.

Пятый пример

Или пример, который должен показать, как легко можно создать устройство, используя FlowCode, но как важно, не останавливаясь на достигнутом, потратить некоторое время и приложить усилия к освоению, скажем, языка Си.

Сегодня в каждом доме есть хотя бы один пульт управления, использующий инфракрасное излучение. Разработчики телевизоров, музыкальных центров, домашних кинотеатров и т.п. прилагают немалые усилия к тому, чтобы их ИК-пульт обладал уникальностью. Иначе, переключая телевизионные каналы, вы можете заставить кондиционер «нагнать» такого холода, что «заморозите» картинку на экране.

Однако в любительской практике, где управление с помощью инфракрасного излучения достаточно привлекательно, нет нужды излишне усложнять код, излучаемый пультом. Создадим такой пульт на базе МК, максимально упростив задачу: она должна быть легко обозримой, поскольку усложнить ее всегда можно простым повторением уже полученных фрагментов программы. Но вначале посмотрим на один из вариантов организации управляющего ИК кода.

У начинающих порой вызывает недоумение наличие несущей частоты, которая может меняться от нескольких килогерц до нескольких сотен килогерц. Нужно ли это усложнение?

Это усложнение в любом случае очень полезно: без несущей частоты код управления будет подвержен воздействию конкурирующих тепловых излучений, воздействующих на приемник. Кроме того, для повышения яркости свечения излучателя приходится увеличивать ток через него, а последний ограничен для обычных светодиодов несколькими десятками миллиампер. Тогда как через излучающий светодиод можно пропустить импульс тока в несколько сотен миллиампер, но при условии, что этот импульс будет коротким, а скважность достаточно большая.

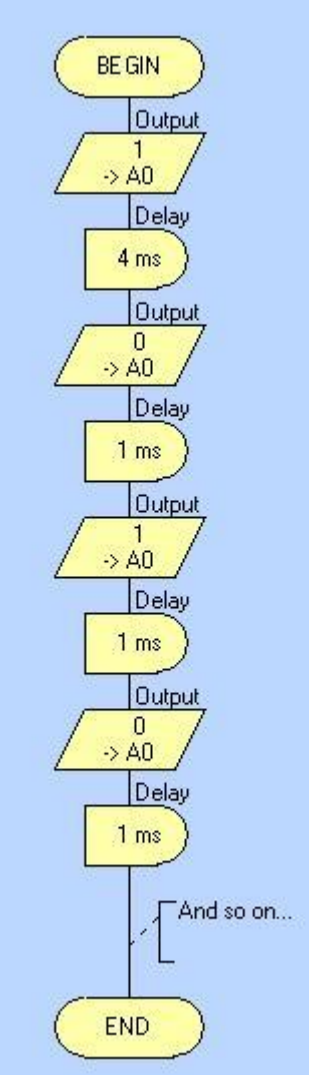
Но оставим пока вопрос о несущей и вернемся к организации управляющего кода. Выберем базовый период *T* для нашего кода. Длительность обязательного заголовка примем равной *4T*. Паузу, разделяющую информационные биты, примем равной по длительности *T*. Для единицы примем длительность импульса равную 2T, а для нуля, как и для паузы, длительность пусть будет равна *T*. Какой быть длительности, выраженной в единицах времени, для базового периода, мы определим позже, а пока примем ее равной одной миллисекунде. Примем, что мы используем самый простой вариант – после заголовка будем передавать один байт, как он есть. То есть, код для команды номер 1, выражаемой числом 1, будет выглядеть (в терминах базового периода) так:

Импульсный код для байта 00000001

Рис. 4.12. Структура импульсов кода

Пауза между командами должна быть не менее 25 мс.

Таким образом, программа должна воспроизводить примерно такую последовательность команд:



Я отнюдь не фанат «изящного кода», но даже мне такое лобовое решение не по нраву.

Но мы уже знакомы со счетным циклом, который можно выполнить восемь раз (по количеству отправляемых бит).

При этом каждый бит может принимать только два вида, либо единицы, либо нуля.

Выше мы знакомились с использованием клавиатуры, которая по нажатию кнопки может задавать значение (номер кнопки) переменной. Назовем ее *key*. Эту переменную можно использовать в программе для генерации соответствующего кода.

Чтобы определять значения каждого бита, можно использовать маску и программную конструкцию вида *key AND mask*, где *mask* – это маска, выделяющая (нашими усилиями) значение нужного бита.

Ниже приведена программа для пошаговой проверки и отладки генерации кода. Значение кнопки, якобы нажатой, вводится вручную. При необходимости эту часть можно заменить программой опроса клавиатуры.

Рис. 4.13. Программа, генерирующая управляющий код

В подпрограмме *cod_out* ниже используются два фрагмента **Calculation**. Но их можно объединить в один.

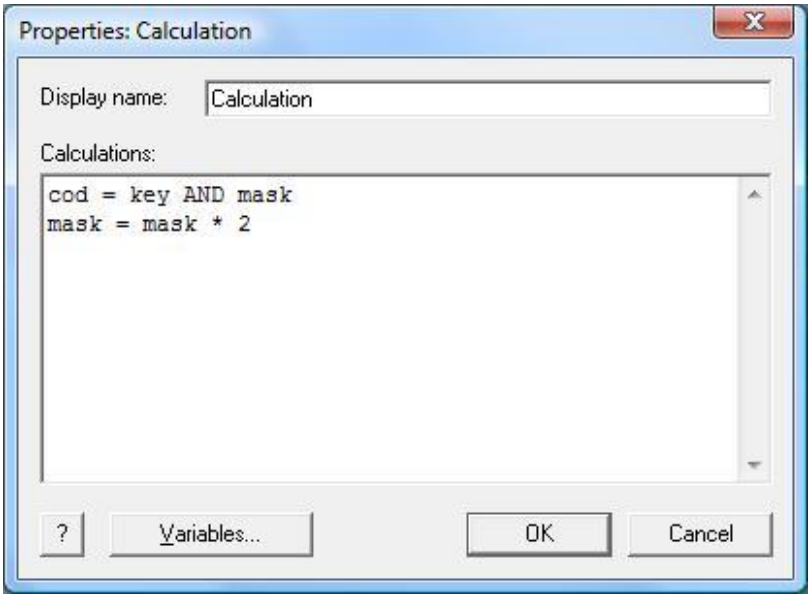


Рис. 4.14. Манипуляции с битами номера нажатой на пульте кнопки

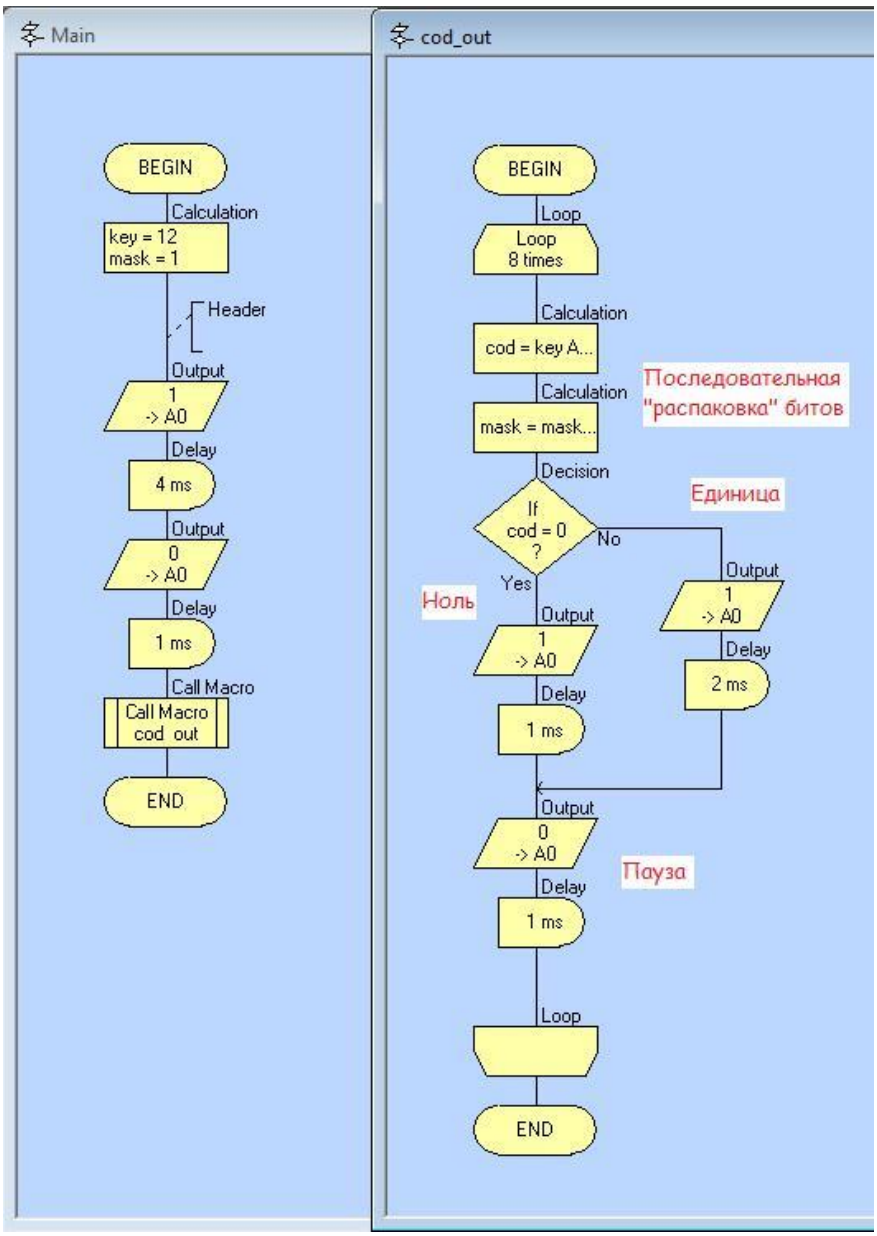


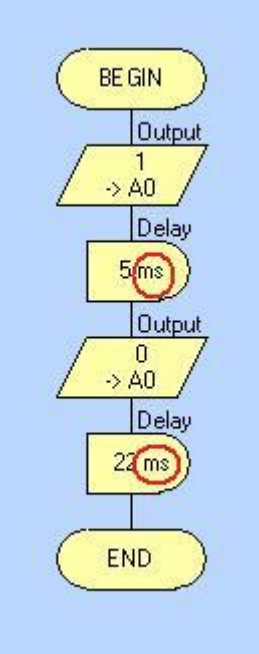
Рис. 4.15. Программа и подпрограмма подготовки номера кнопки к генерации кода

После заголовка, который остался без изменений, первым «в путь» отправится младший бит, а последним старший.

И вновь – мы используем только несколько уже знакомых фрагментов кода на языке Си. Для пущей надежности генерируемый код можно повторить несколько раз, не забывая вставлять паузу в 25 мс между повторениями команды. И еще одно: для своих экспериментов с ИК управлением можно использовать готовый пульт. Но, как мне кажется, разумнее создать свое устройство. В качестве излучающего светодиода можно использовать красный индикаторный диод АЛ307А. Если его «дальнобойности» не хватит, тогда поискать более мощный ИК светодиод. АЛ307А допускает импульсный ток до 100 мА при длительности импульса менее 10 мс и среднем токе порядка 20 мА. Для обеспечения нужного среднего тока мы примем меры, однако прежде все те фрагменты программы, где вывод RA0 принимает значение равное единице, следует заменить модулирующими импульсами.

Как видно из вышеизложенного, создание программы в FlowCode занимает немного времени и не требует больших усилий. Не намного больше усилий потребуется для создания кода программы на языке Си с использованием фрагментов, полученных в FlowCode. Но мне хотелось подчеркнуть, я об этом упоминал, что освоение языка Си будет полезно и в том случае, если вы намерены использовать полную версию программы.

Рассмотрим вопрос о замене состояний RA0 равных единице модулирующим кодом, или какими должны быть модулирующие импульсы? Если в приемнике использовать, что мне кажется очень разумным, приемный модуль типа TSOP с частотой фильтра 36-38 кГц, то период модулирующих импульсов окажется близким к 27 мкс. Если мы хотим использовать ток в 100 мА, то длительность включения 5 мкс должна обеспечить средний (за период) ток порядка 20 мА. Итак, импульс модуляции в программе может выглядеть таким:



Эта конструкция должна заменить и «единицу», и «ноль», и «заголовок» в предыдущей программе.

И все было бы хорошо, если бы ни одно «НО»!.. В программе FlowCode нет пауз меньше одной миллисекунды.

Вот здесь и самое время использовать знание языка Си и возможности, например, компилятора PICC Lite. Повторяя модулирующий импульс столько раз, сколько требует каждая из перечисленных «заготовок», мы сформируем полный ИК управляющий код.

И последнее – если использовать это устройство с автономным питанием, то следует подумать о режиме «сна» для МК. Это будет экономить энергию батареек.

Рис. 4.16. Программное формирование импульса модуляции

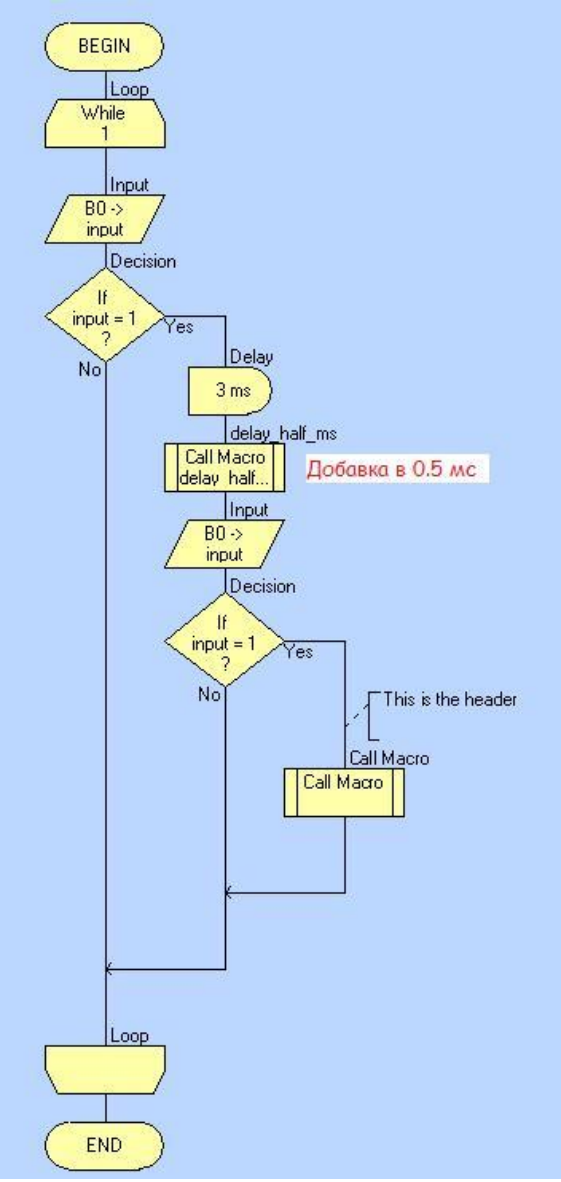
Если заменить ИК излучатель радио-модулем в разрешенной для, например, управления моделями частотной области с дальностью действия в несколько метров, то можно использовать пульт для передачи команд в соседнюю комнату, где может располагаться приемник спутникового телевидения. А в качестве приемного устройства использовать аналогичный пульту модуль, который по полученным командам будет воспроизводить ИК коды пульта управления приемником спутникового телевидения. Для реальной работы вам потребуется не так много команд, а на создание программ для МК вы потратите несколько часов. Да и те уйдут, скорее всего, на прочитывание кодов пульта управления приемником спутникового ТВ.

А в следующем примере мне хотелось бы пояснить, почему я предпочел бы пульт ИК управления, описанный выше, использовать собственного «изготовления».

Шестой пример

Сегодня, когда в магазине можно купить готовый пульта управления, когда, как правило, есть старые, уже не нужные пульты, когда относительно недорого можно купить пульт, способный запоминать нужные ИК коды, сегодня изготовление пульта управления может показаться не нужным. Собственно, так оно и есть.

Однако прочитать коды, отправляемые с пульта управления, гораздо легче, если вы сами создали удобные для прочитывания коды. Как прочитать управляющие коды пульта из предыдущего примера, покажем в этом примере. Подразумевается, что в качестве фотоприемника используется микросхема типа TSOP, на выходе которой мы получим код, изображенный на рисунке 4.12. Только порядок битов будет обратным (если мы не изменим этого в программе пульта).



В данном случае программа ожидает появления сигнала на входе RB0.

Если приходит импульс от фотоприемника, то, учитывая, что задуманный нами заголовок имеет длительность 4 мс, программа делает паузу на 3.5 мс и вновь проверяет состояние входа. При приходе заголовка команды входной импульс не изменит своего состояния, что можно фиксировать, как начало команды.

И в этой программе «довесок» в 0.5 мс потребуется «изготовить» на языке Си.

После подтверждения прихода заголовка программа вызывает подпрограмму обработки команды. Сканирование команды можно организовать по тому же алгоритму, что был использован для чтения заголовка. Сдвиг на 0.5 мс существенно облегчит работу по сканированию кода.

Рис. 4.17. Один из возможных вариантов программы прочитывания кода

Но можно обойтись в программе приемника без вставки паузы в 0.5 мс. Вернее, полностью использовать другой алгоритм считывания кода.

Программа будет начинаться тоже с ожидания изменения состояния входа RB0 (кстати, можно использовать для этой цели и прерывание по изменению состояния RB0).

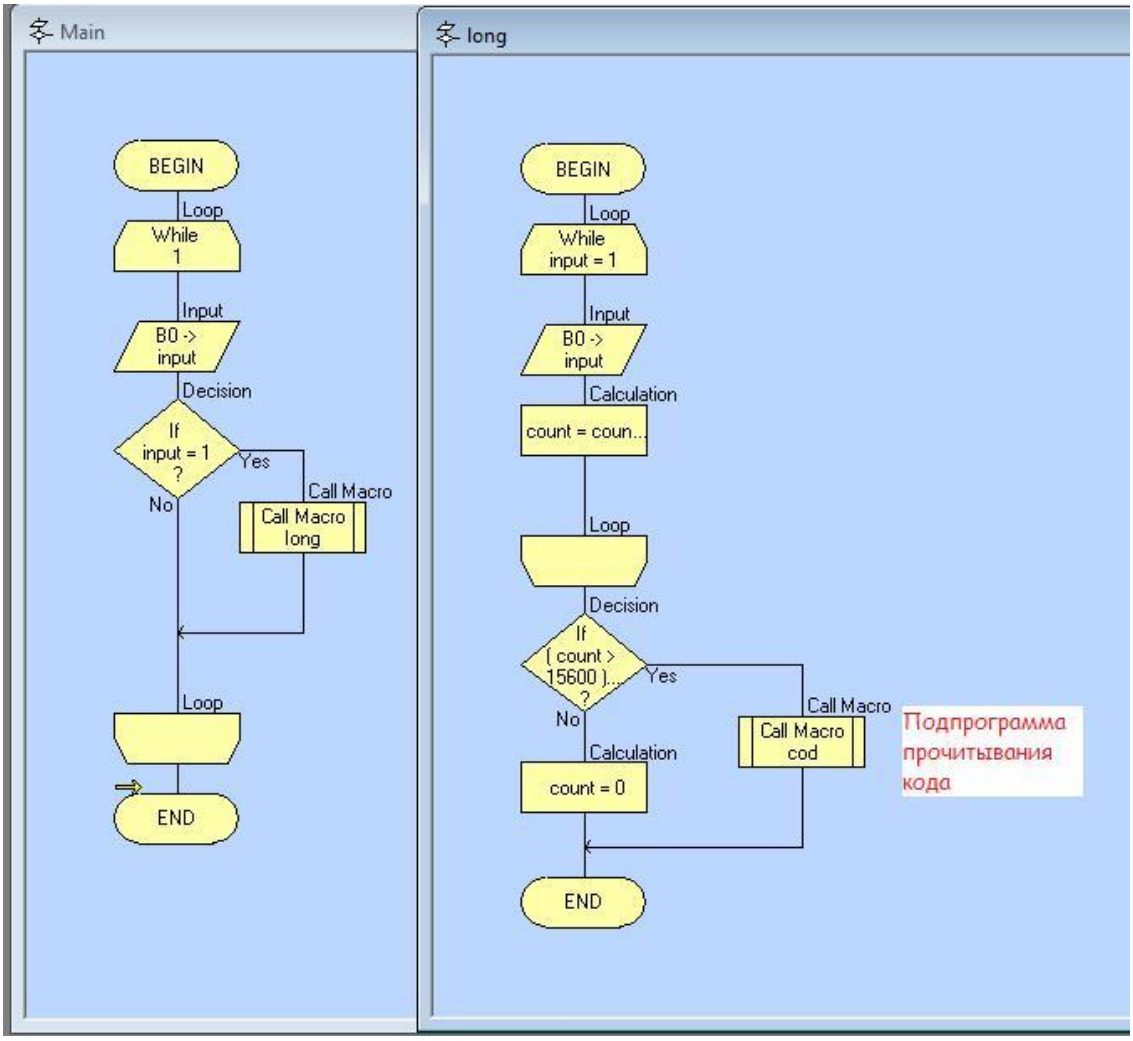


Рис. 4.18. Другой вариант программы прочитывания кода ИК команды

В этом варианте с началом прихода положительного импульса запускается счетчик. Грубая оценка полученного числа (без учета того, что цикл While input = 1 не будет выполняться за один машинный цикл) дает значение 16000. Заголовок длится 4 мс, а один такт МК занимает 0.25 мкс. Можно скорректировать это значение, если учитывать «неоднородность» приходящих модулированных импульсов, например, используя условие ветвления в подпрограмме:

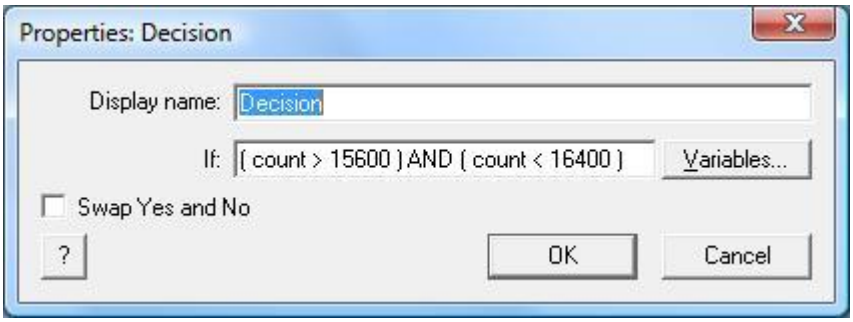


Рис. 4.19. Задание возможного разброса значений импульса

Аналогично можно использовать подсчет длительности импульсов и в подпрограмме «расшифровки» управляющего кода.

Кроме того, можно построить алгоритм прочитывания кода, используя встроенный таймер. Словом, здесь и таится самое интересное в работе с микроконтроллером – придумать и реализовать свой алгоритм работы устройства.

Но то, что мне хотелось показать в этом примере, надеюсь, мне удалось показать: если вы точно знаете формат принимаемого кода, у вас появляется много вариантов того, как создать программу для приемника.

Приложение

Немного о четвертой версии FlowCode

Четвертая версия, которая появилась в настоящее время, даже при первом запуске явно показывает изменения, произошедшие с программой.

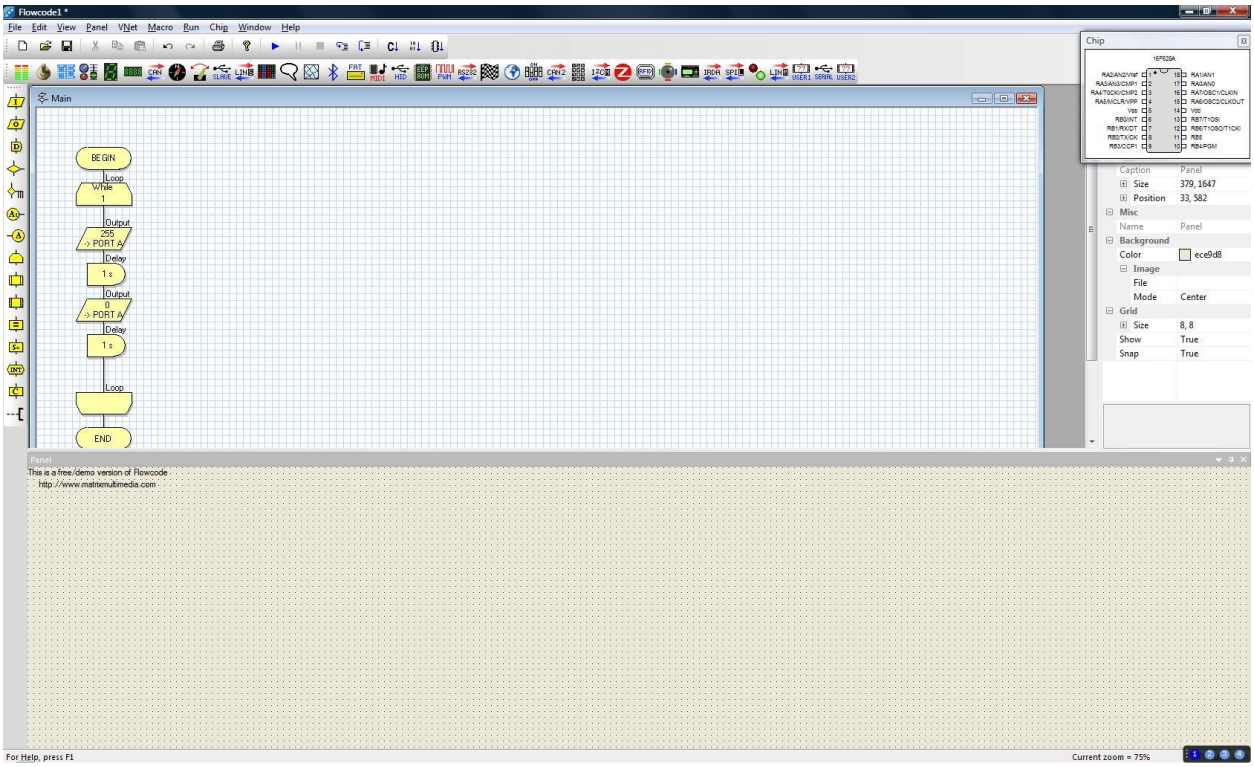


Рис. 5.1. Новое окно FlowCode

Кроме рабочего поля программы появилось новое поле – Панель для размещения дополнительных компонентов, список которых тоже существенно пополнился.

Для профессионалов существенным покажутся изменения в компиляторе. Например, появление возможности работать с числами с плавающей точкой.

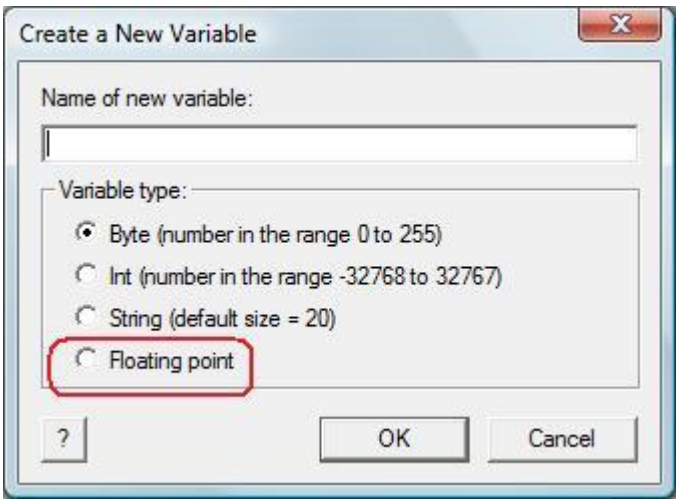


Рис. 5.2. Новое диалоговое окно переменной

Для всех удобным будет появление новой единицы в программном компоненте **Delay**.

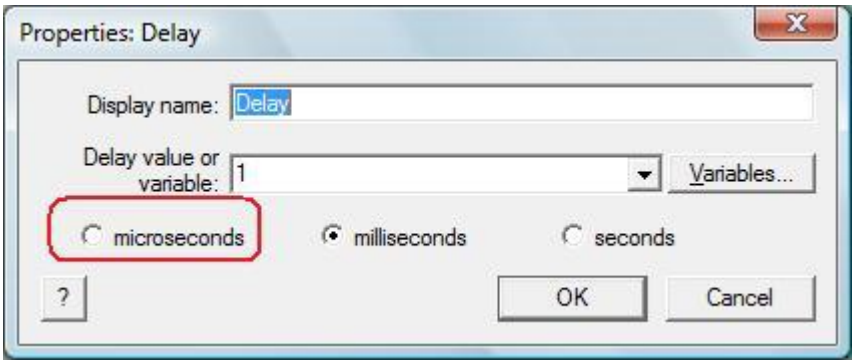


Рис. 5.3. Диалоговое окно компонента **Delay**

Не менее удобно появление нового программного компонента **Switch**.

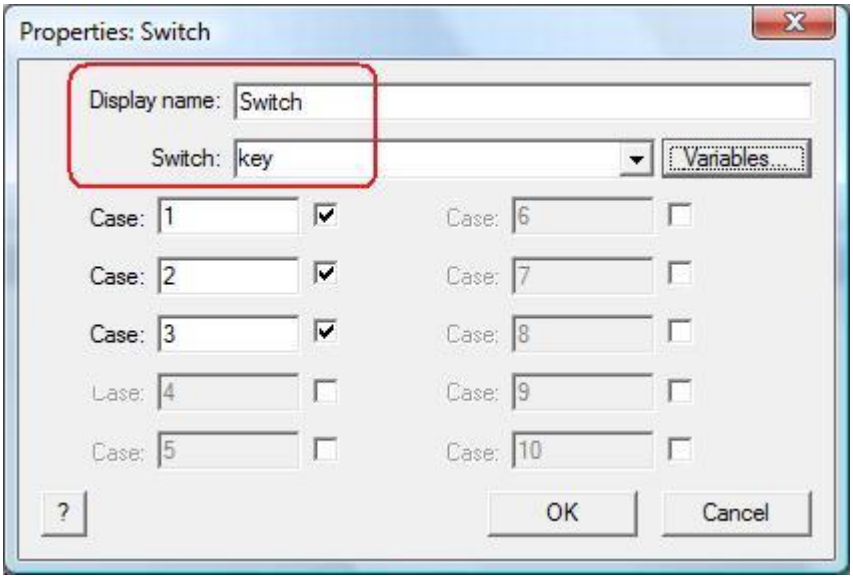


Рис. 5.4. Диалоговое окно компонента **Switch**

Использование этого компонента делает программу нагляднее.

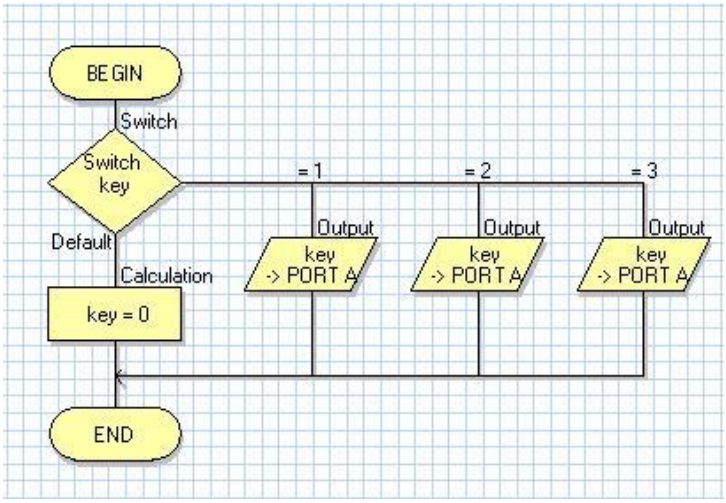


Рис. 5.5. Использование компонента **Switch** в программе

Изменения коснулись и дополнительных компонентов. Так появилось окно свойств, где двойной щелчок левой клавишей мышки в поле *Connections* открывает диалоговое окно подключения компонента.

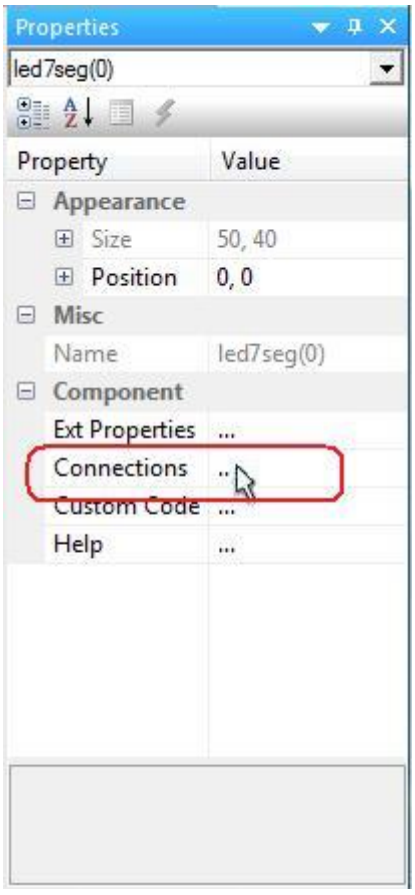


Рис. 5.6. Окно свойств компонентов

Есть изменения и в подключении компонентов. Если диалоговое окно *Connection* мало отличается от привычного ранее, то второй элемент в окне свойств *Ext Properties* открывает новый диалог.

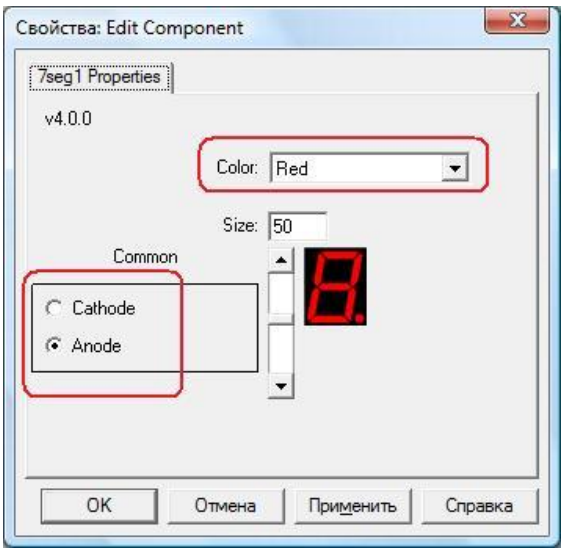


Рис. 5.7. Диалоговое окно дополнительных свойств семисегментного индикатора

Теперь возможно изменить цвет индикатора, его размер, а для многих важно то, что можно изменить общий вывод: общий катод или общий анод.

Несколько изменилось и поведение ряда дополнительных компонентов, так четырехзначный индикатор теперь будет работать при его подключении к порту в том случае, когда вы используете **Component Macro**. Что вполне разумно.

Изменился вид отдельных дополнительных компонентов, вернее, появился выбор этого вида.

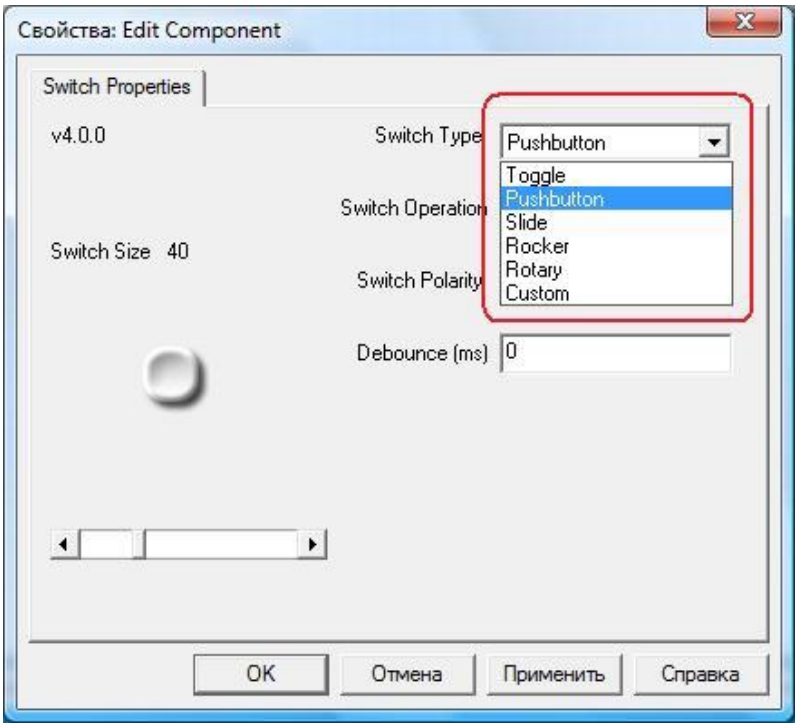


Рис. 5.8. Свойства компонента выключатель

И еще одно полезное нововведение:

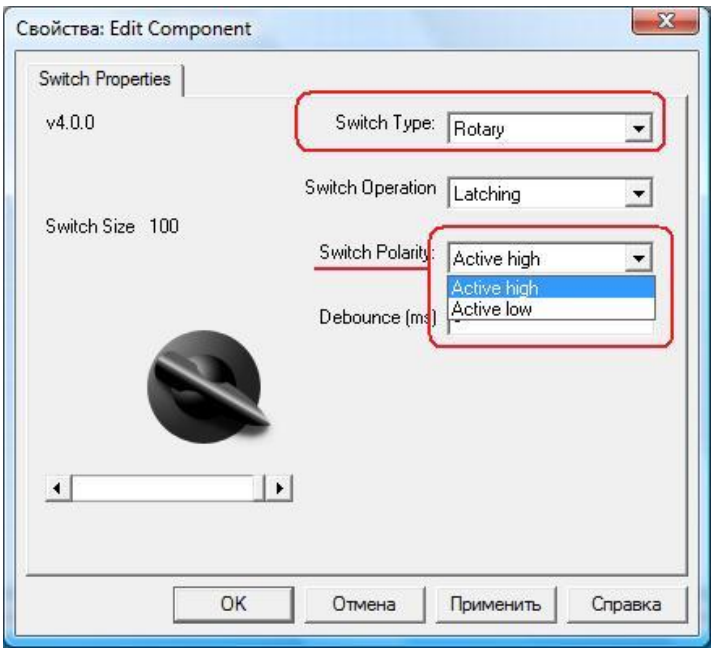


Рис. 5.9. Выбор полярности подключения выключателя

Панель для установки внешних компонентов может по вашему желанию изменить вид.

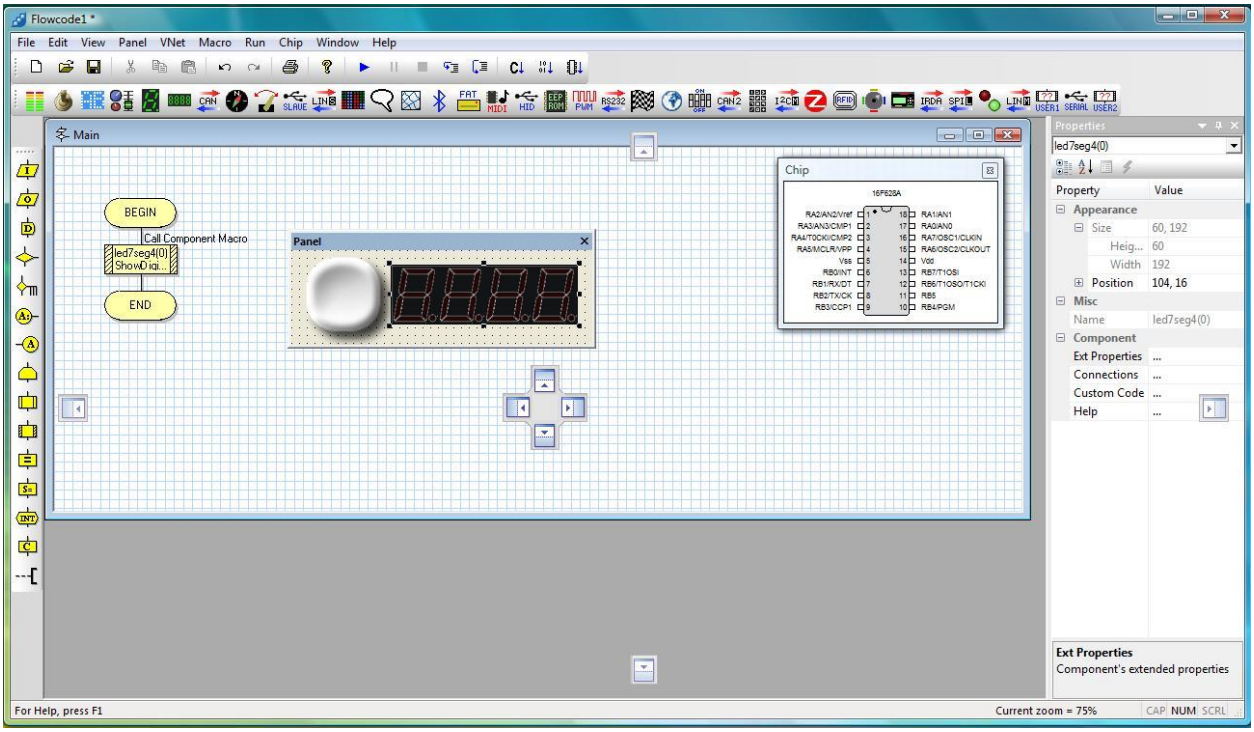


Рис. 5.10. Измененный вид панели (Floating – плавающая)

Для выбора вида панели достаточно щелкнуть правой клавишей мышки по ее титульной панели.

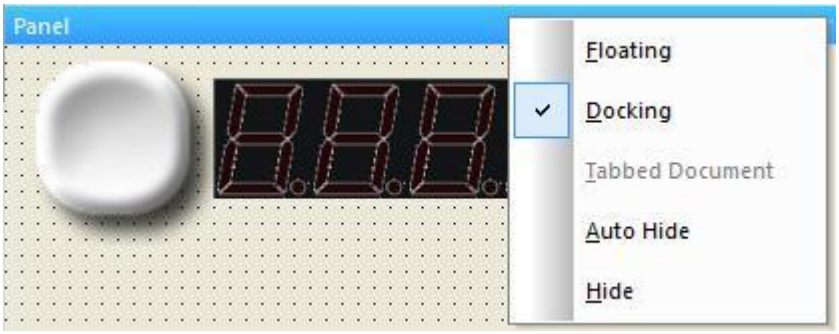


Рис. 5.11. Меню выбора вида панели

Словом, изменений много, а, став обладателем полной версии, вы найдете много для себя полезного. Вот, например, что говорит **Help** об изменениях в новой версии.

Развитие компонентов:

- Новые компоненты, включающие Stepper, Servo, Speech…
- Улучшена функциональность компонентов, включая ADC, RS232, I2C, PWM…
- Симуляция компонентов стала более гибкой и изменяемой
- Компонентные макросы теперь изменяемы
- Компоненты классифицированы

Программные возможности:

- Виртуальные сети FlowCode
- Внутрисхемная отладка (ICD)
- Окно наблюдения за переменными может отображать значения в реальном времени с использованием ICD
- Окно наблюдения за переменными может отображать значения регистров устройства в реальном времени с использованием ICD
- Поддерживается арифметика с плавающей точкой
- Пользовательские макросы только для чтения
- Паузы в микросекундах
- Сторожевой таймер теперь поддерживается паузами и компонентами
- Функции преобразования чисел с плавающей точкой и строковых данных
- Функции преобразования шестнадцатеричных чисел и ASCII
- Встроенный редактор кода с подсветкой синтаксиса

Улучшения графического интерфейса:

- Новый улучшенный графический интерфейс
- Панель симуляции
- В подсказках под иконками полный текст
- Окна с закладками
- Новый вид
- Темы фонового рисунка
- Улучшенные иконки

Улучшения иконок:

- Новая иконка программного переключения
- Новые иконки компонентов и графики симуляции
- Точкам соединения могут назначаться выразительные этикетки

Совместимость с моделями

- Внешний целевой микроконтроллер поддерживается прямо из FlowCode
- Внешние прерывания поддерживаются как стандартные для целевых устройств
- Улучшена поддержка программных средств других производителей

Изменений, действительно, достаточно много. Все ли они к лучшему, покажет время.

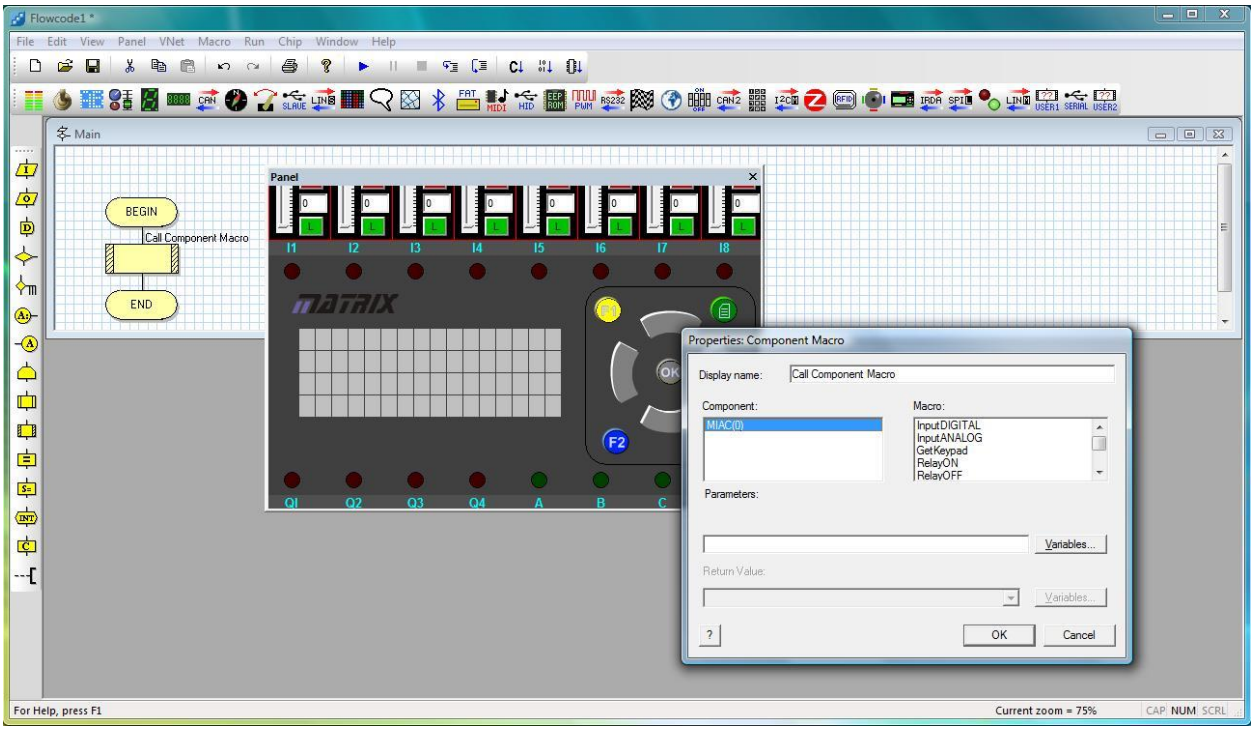


Рис. 5.12. Четвертая версия FlowCode

Немного о программе Proteus

Вернее о ее составляющей ISIS. Для работы с микроконтроллерами достаточно программы FlowCode, если вы точно знаете, что хотели получить от контроллера и получили в результате. Сама программа FlowCode предоставляет более чем исчерпывающие средства отладки. Однако, скажем, для проверки работы контроллера с внешним генератором, или при проверке сетевой связи двух микроконтроллеров можно воспользоваться программой Proteus, если она у вас есть.

Поводом использовать все имеющиеся средства может стать и появление неких чудесных явлений, причина которых, впрочем, кроется либо в неисправности микроконтроллера, либо, что бывает чаще, в собственных ошибках. Вот пример такой ошибки, которая, как я полагаю, заставит и вас работать по принципу «доверяй, но проверяй».

Все достаточно банально. Для решения задачи нужно бы «засветить» четырехразрядный семисегментный индикатор. Программа пишется и проверяется в FlowCode. Все, похоже, работает. Но при проверке «в железе» не горят два сегмента: E и F. Индикатор подключен к порту A

контроллера PIC16F628A. С сегментом E понятно, на выходе транзистор с «открытым» стоком, то есть, требуется его подтянуть к плюсу питания. А со вторым сегментом...

На помощь приходит программа Proteus.

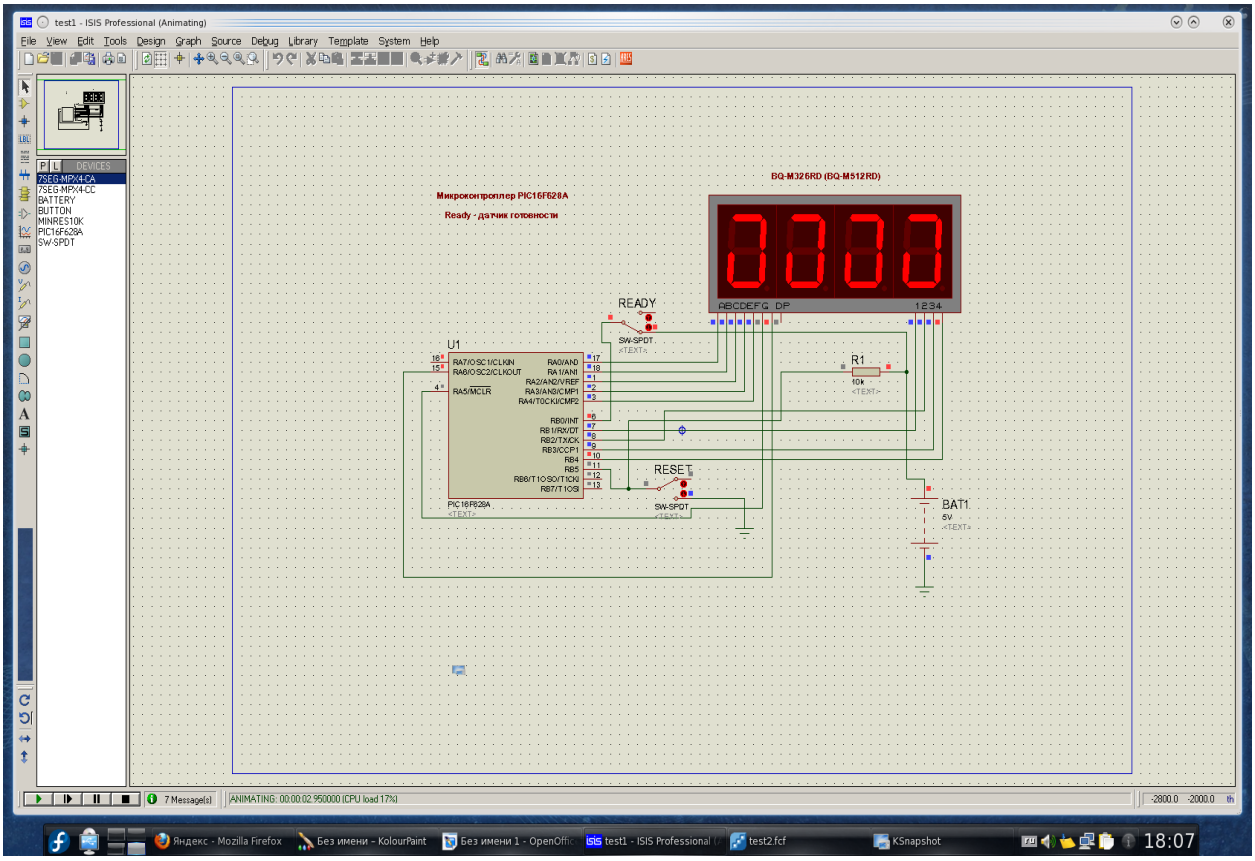
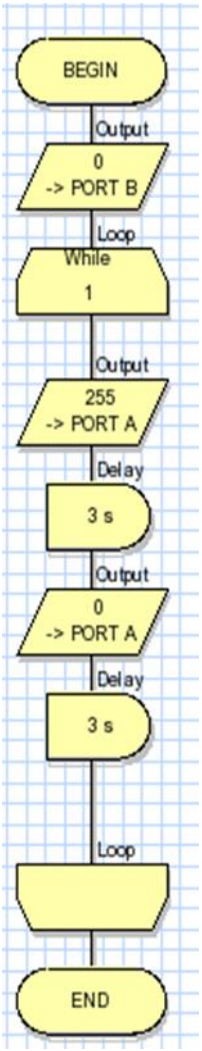


Рис. 5.13. Моделирование в программе Proteus

И в Proteus сегмент F не горит. Такое впечатление, что один из выводов порта A не желает, в отличие от моих планов, работать на выход. Можно, конечно, проверить программу на макетной плате, но не хочется покупать индикатор только для единственной проверки. Задачу можно, впрочем, упростить, вернее, упростить программу, оставив только зажигание и гашение всех сегментов для индикатора с общими катодами, которые подключены к порту B.



Программа подключает катоды всех разрядов к общему проводу (устанавливает низкий уровень на выходах порта B).

Затем в бесконечном цикле все выводы порта A устанавливаются в высокое состояние, а после паузы в 3 секунды все выводы порта устанавливаются в низкое состояние.

Программа совершенно проста и ясна, нет никаких сомнений относительно ошибок, скажем, в использовании режима динамической индикации.

Но прежде, чем начинать проверки, я еще раз читаю описание микроконтроллера PIC16F628, отрыв первое, подвернувшееся под руку, благо оно на русском языке. Получается, что если слово конфигурации задано верно (а компараторы отключены), то весь порт A должен работать как цифровой вход/выход.

Получить hex-файл в такой простой программе дело минутное.

Рис. 5.14. Проверочная программа

Полученный hex-файл загружается в программу Proteus, запускается и…

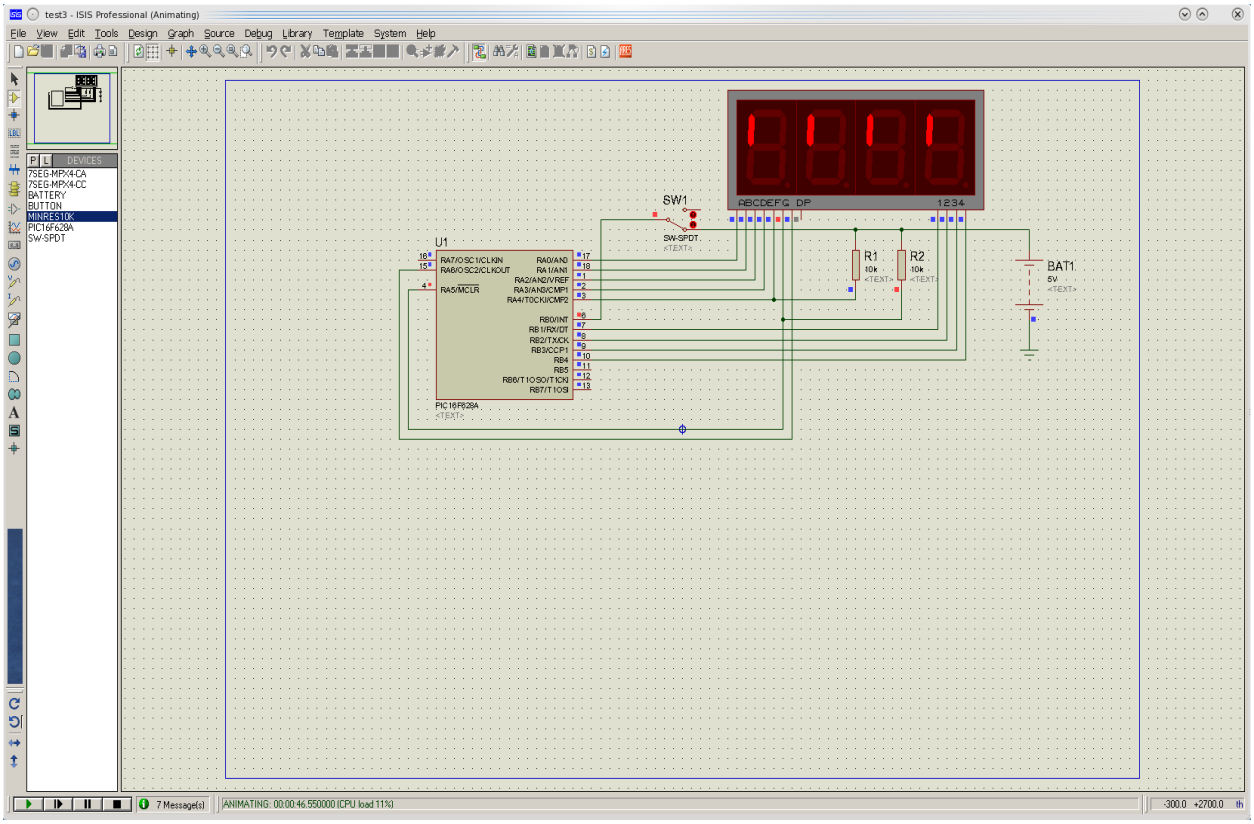


Рис. 5.15. Проверка работы предыдущей программы в Proteus

И... теперь сегмент зажигается, но не гаснет. Повторяем все тщательно еще раз. Заводим новую папку для программы ISIS (Proteus). Запускаем программу. Устанавливаем микроконтроллер. Есть разные способы достичь этого. Я привык (и когда успел?!) использовать кнопку инструментальной панели и менеджер библиотеки.

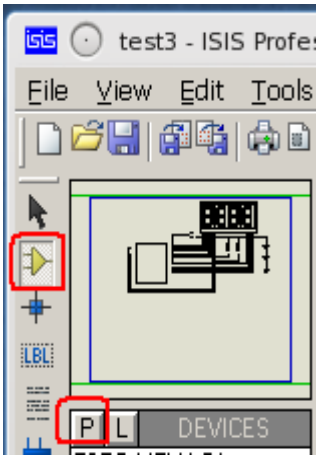


Рис. 5.16. Доступ к компонентам в Proteus

Proteus я использую в Linux, там есть небольшие проблемы с выбором из списка. Но есть поиск по названию, который я использую:

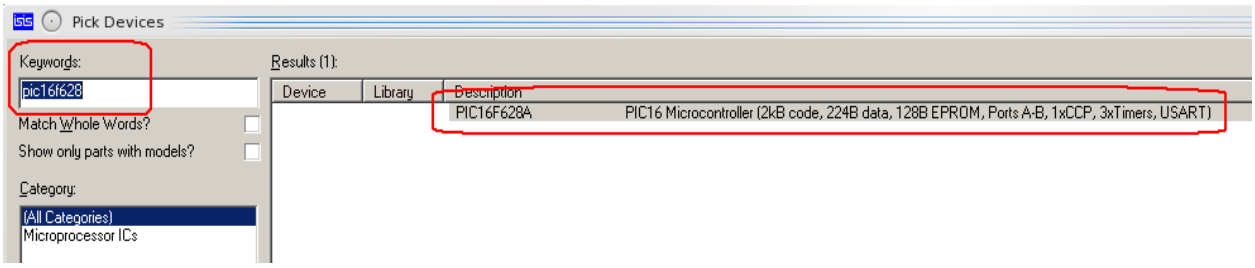


Рис. 5.17. Поиск в менеджере компонентов Proteus

Установив (в менеджере компонентов щелчок по кнопке ОК, затем в нужном месте рабочего поля щелчок левой клавиши мышки) контроллер на схему, в разделе Optoelectronics менеджера компонентов я нахожу две модели индикаторов: 7SEG-MPX4-CA (общий анод) и 7SEG-MPX4-CC (общий катод). Последний и выбираю, вводя его в строку поиска. Теперь осталось соединить все выводы согласно программе и рисунку индикатора.

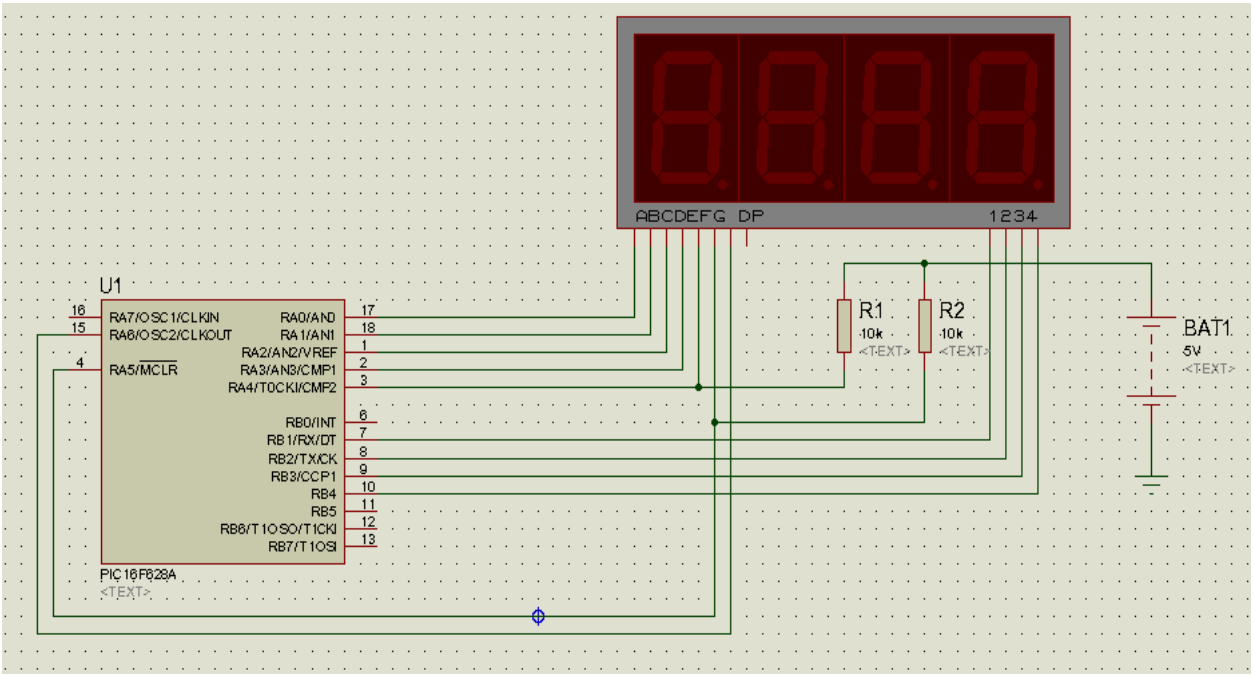


Рис. 5.18. Схема проверки

Двойной щелчок по микроконтроллеру открывает диалоговое окно его свойств.

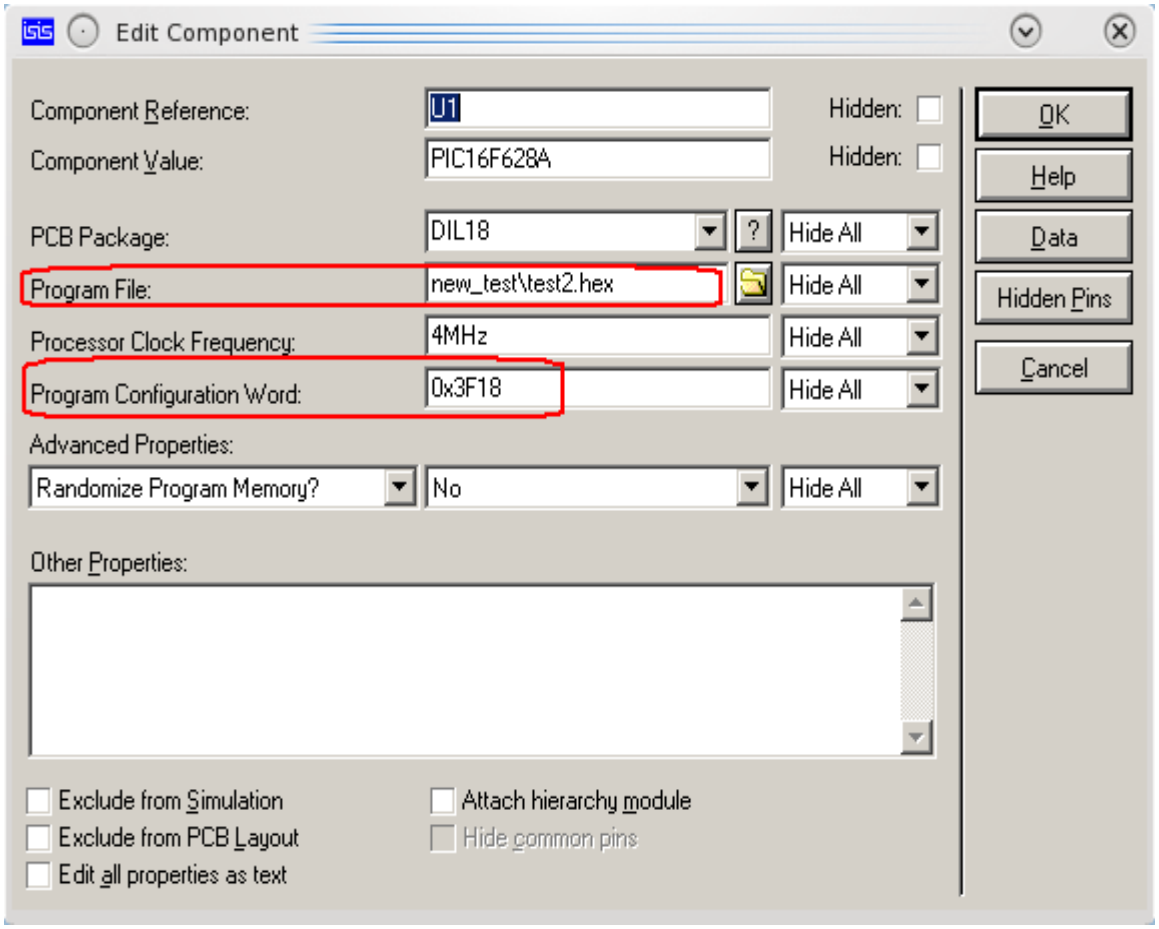


Рис. 5.19. Диалоговое окно свойств микроконтроллера

В окне диалога следует указать путь к hex-файлу, который лучше перенести в папку с проектом Proteus, указать слово конфигурации. Проделав все это, я запускаю программу, и... получаю тот же результат, что изображен выше. Похоже, что бит, связанный с сегментом F не желает работать. Но программа Proteus имеет свои возможности отладки, которые можно запустить, например, из основного меню (пункт Debug).

Конечно, сам запуск программы уже операция по отладке схемы. Можно использовать графики. Эта возможность, присущая всем программам, в которых разрабатываются электрические схемы, эквивалентна применению осциллографа при работе с макетной платой. Здесь вполне справедлив подход: лучше один раз увидеть, чем сто раз услышать.

Программа Proteus, к слову, имеет виртуальный осциллограф. Но в данном случае удобнее было бы использовать график, с которым можно работать, детально разбирая все аспекты полученной осциллограммы. Однако при работе с микроконтроллерами, все-таки основой является программа, которую в первую очередь и следует проверить.

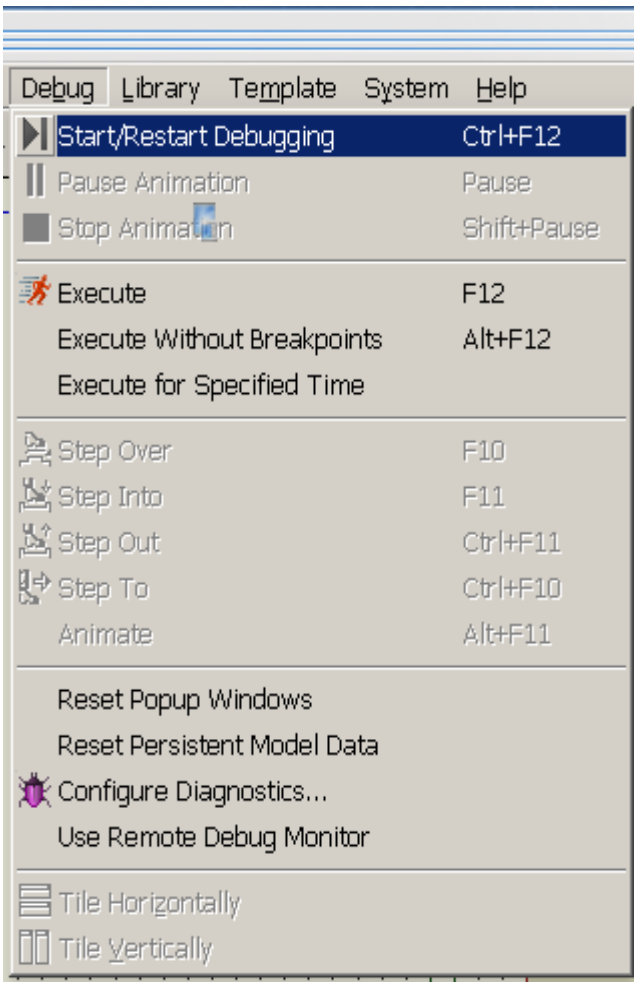


Рис. 5.20. Выпадающее меню отладчика в Proteus

Нажав ***Start/Restart Debugging***, после повторного входа в этот раздел основного меню можно обнаружить перемены, которые произошли с меню.

Такие изменения весьма часто встречаются в программах. А меню такого рода называют контекстным меню. Это относится и к тем разделам меню, которые выглядят бледно. Они активизируются только тогда, когда программа переходит к выполнению действий, затрагивающих эти команды. Иначе эти команды не используются. Поэтому они не активизируются.

Очень полезно, осваивая программу, внимательно присматриваться к тому, что происходит и с программой, и с ее компонентами: элементами схемы, например, с выпадающими меню, с всплывающими меню и подсказками.

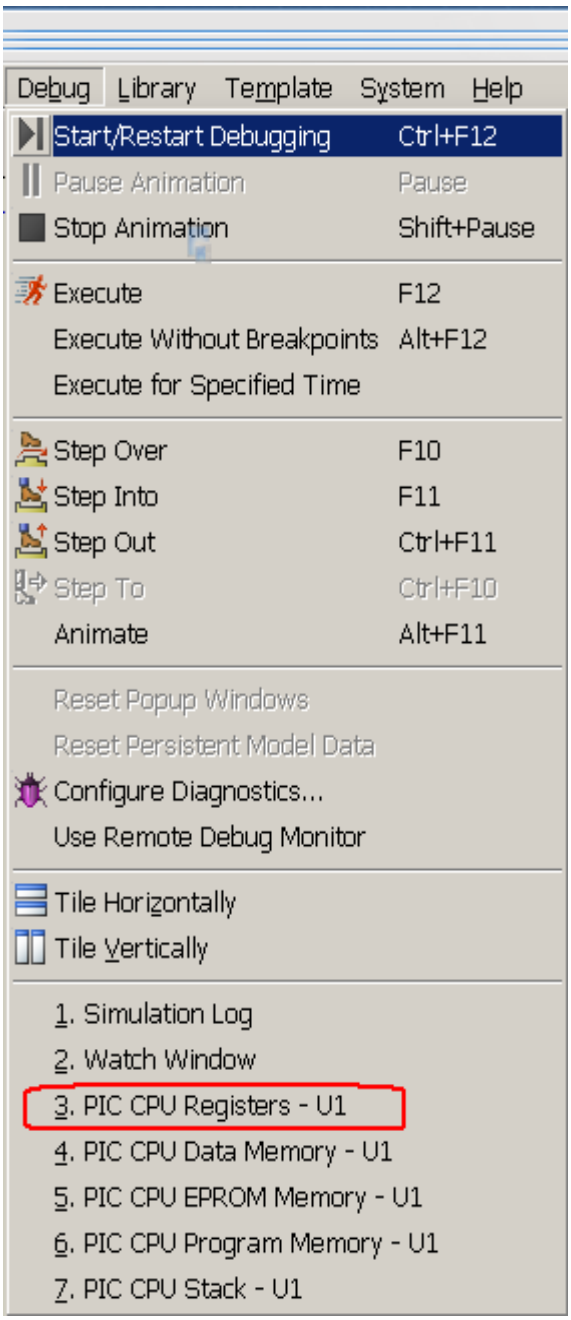


Рис. 5.21. Выпадающее меню отладчика после запуска отладки

Нажимая пункт, отмеченный на рисунке, получаем возможность наблюдать за регистрами. Это можно сделать в пошаговом режиме (например, клавиша F11 на клавиатуре), можно использовать команды ***Step Over*** (клавиша **F10**) или ***Step Out***, (клавиши **Ctrl+F11**), можно, наконец, запустить программу клавишей «▶» на панели строки состояния.

Рис. 5.22. Клавиши управления симуляцией

Но главное, что мне хотелось увидеть, это:

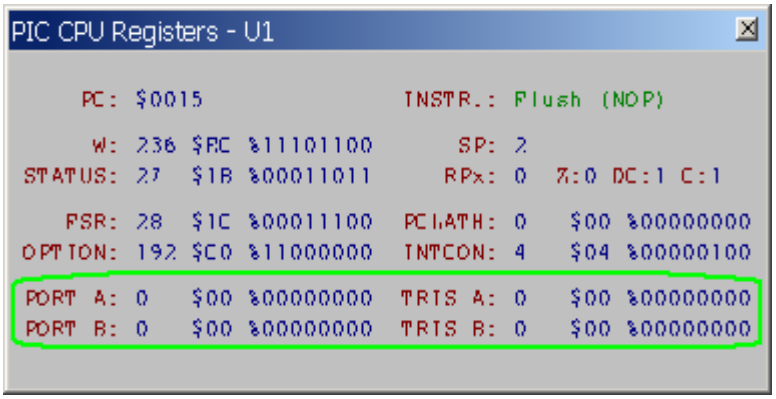


Рис. 5.23. Окно наблюдения за регистрами

Как я полагаю, порты A и B работают на выход, в портах все нули, значит... все сегменты должны погаснуть. Кстати, чтобы не возвращаться к этому: в примерах к программе Proteus есть возможность лучше посмотреть, как используются отладка программы микроконтроллера.

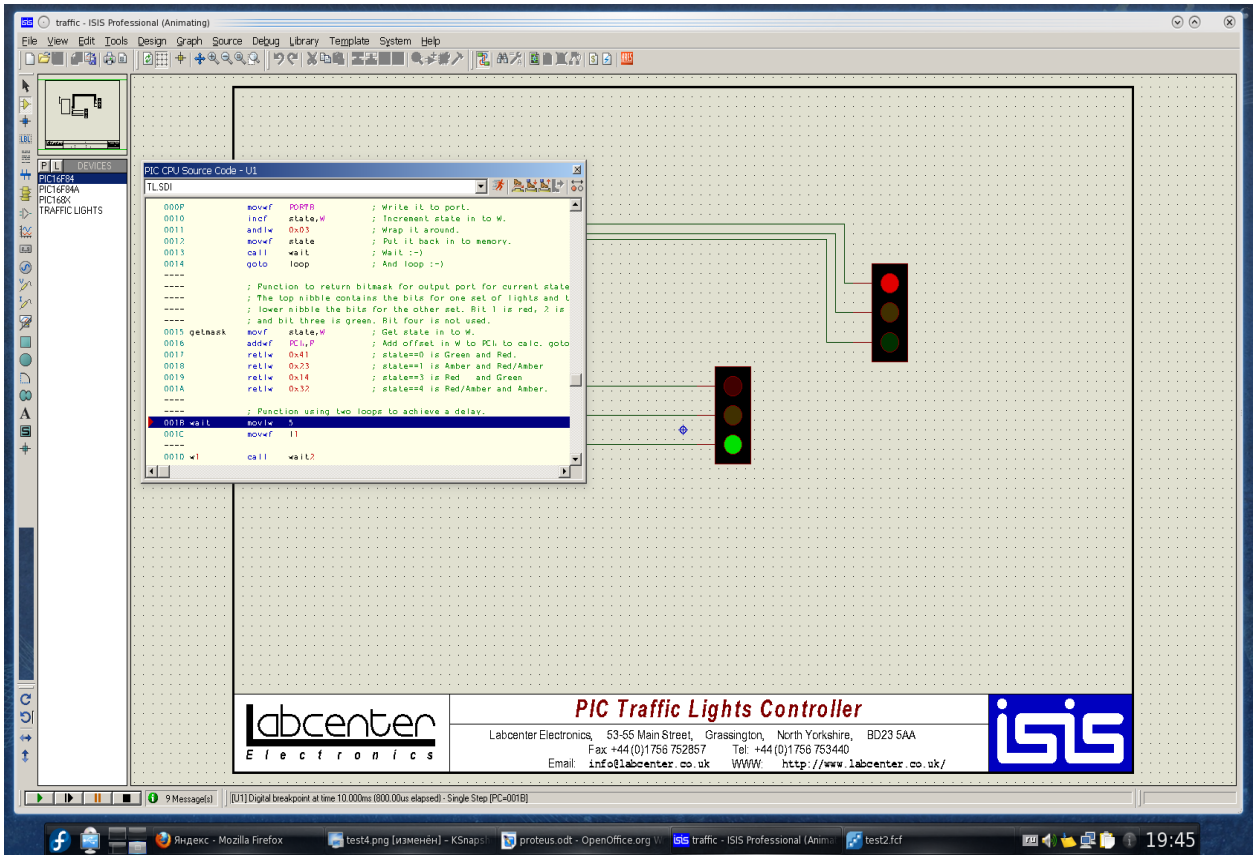


Рис. 5.24. Отладка программы МК из примеров Proteus

Когда знаешь решение задачи, когда знаешь ответ на вопрос, то удивляешься, отчего же ты не увидел то, что лежит на поверхности. Но это, когда знаешь.

Проверка на макетной плате показала, что подтянутый резистором к плюсу источника питания вывод RA5 остается в состоянии с высоким уровнем после команды – установить низкий уровень на всех выводах порта A.

Теперь самое время вспомнить о том, с чего начинался разговор о программе FlowCode. С того, что она хороший помощник в изучении языка Си. Повторить программу, изображенную на рисунке 5.14, совсем несложно.

```
#include <htc.h>
#define _XTAL_FREQ 4000000
void main()
{
    char i;

    while (1) {
          CMCON = 0x07;
          TRISA = 0x00;
          PORTA = 0xFF;
          for (i=0;i<30;i++) {
                __delay_ms(100);
          }
          TRISA = 0x00;
          PORTA = 0x00;
          for (i=0;i<30;i++) {
                __delay_ms(100);
          }
    }
}
```

И, создав проект в программе MPLAB, добавив основной файл проекта на языке Си, запустить отладку тестовой программы в этой среде работы с микроконтроллерами.

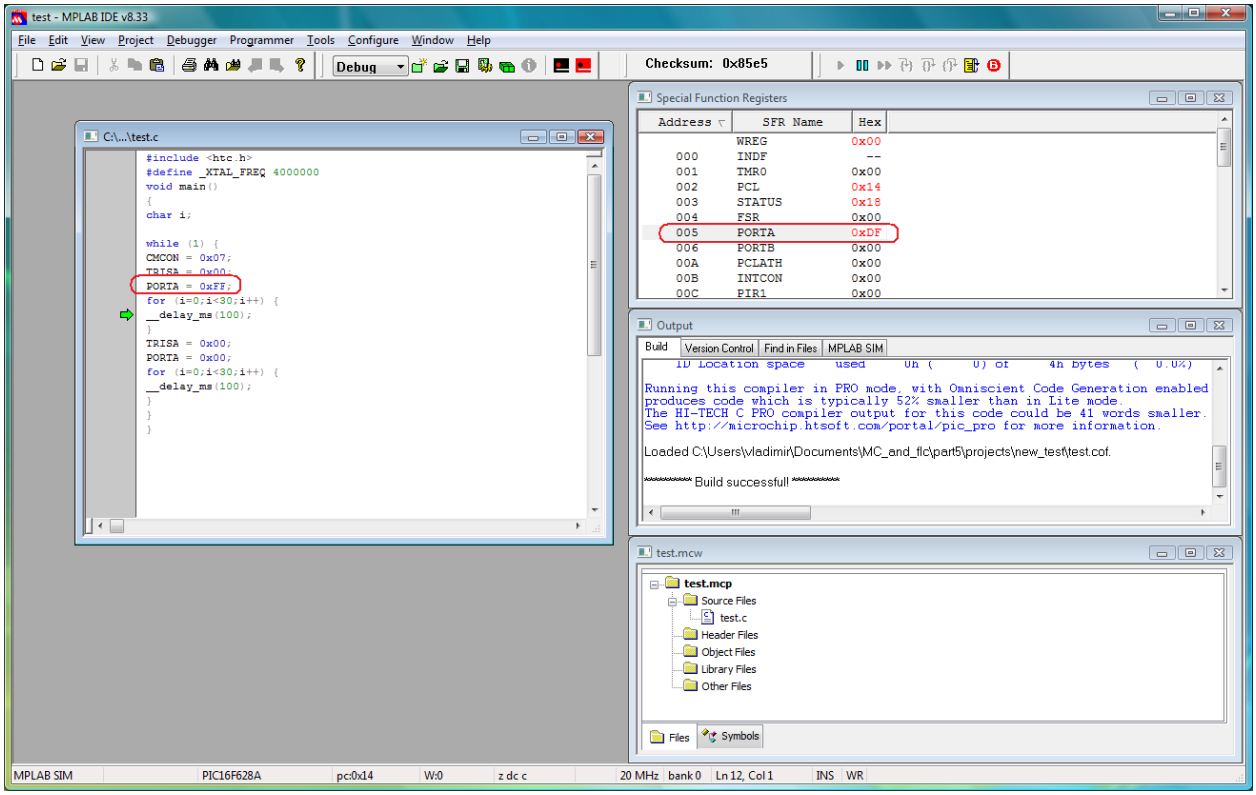


Рис. 5.25. Запуск отладки программы в среде MPLAB

В этот раз видно, что после команды установить все выводы порта А в высокое состояние, в высокое состояние устанавливаются все выводы, кроме RA5 (в окне наблюдения за регистрами видно, что значение не FF, а DF).

И, наконец, решение задачи, ответ на вопрос, что происходит с выводом RA5, обнаруживается в оригинальной версии описания микроконтроллера, где написано:

RA5 is a Schmitt Trigger input only and has no output driver (RA5 – это только вход триггера Шмитта и не имеет выходного драйвера)

и далее о слове конфигурации, касательно этого вывода

RA5/MCLR pin function is digital Input (функция вывода RA5/MCLR – это цифровой вход).

В заключение хочу сказать, что эта банальная ошибка позволила мне чуть-чуть рассказать о программе Proteus и еще раз напомнить, что как бы ни удобно было работать с программой FlowCode, очень полезно освоить работу над программой хотя бы на языке Си.

Пока все получается, пока нет вопросов – дело вкуса. Вы можете изучить ассемблер и пользоваться только им. Вы можете изучить Си и пользоваться только им. Если для вас важно быстро создать программу, вы обязательно обратите внимание на FlowCode.

Но, когда что-то идет не так, когда появляются вопросы, никакие знания не будут лишним обременением. И в этот момент изучать что-то будет гораздо труднее.

Не слушайте тех, кто говорит, что пользоваться FlowCode плохо. Они заблуждаются. Но и повторять их заблуждения, пользуясь только FlowCode, не совсем разумно. Не так ли?

Читая учебник

Аналогично тому, как удобно изучать электротехнику и электронику с программой и книжкой, не менее удобно изучать язык, повторяя программу за компьютером.

Есть прекрасная книга по использованию языка Си в работе с микроконтроллерами, автор Шпак Ю.А., «Программирование на языке С для AVR и PIC микроконтроллеров», издательство «МК-Пресс». Безусловно, самым удобным сочетанием было бы использование тех (или той) программ, о которых пишет автор. Но ничто не мешает к его рекомендациям добавить свои предпочтения. Итак.

Одной из первых встреченных программ в книге будет генератор сигнала SOS.

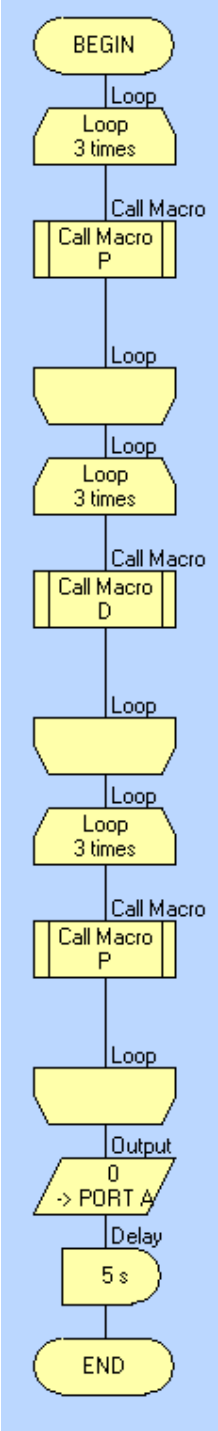
```
#include <18F458.h>
#use delay(clock=20000000)
#fuses HS,NOWDT

void Pause(int ms)
{
      output_D(0xFF);   //Все светодиоды отключены
      delay_ms(ms);     //Задержка
}

void P(void)
{
      output_D(0);      //Включаем все светодиоды
      delay_ms(5);      //Короткая задержка
```

```
      Pause(5);         //Пауза с погасшими светодиодами
}

void D(void)
{
      output_D(0);      //Включаем все светодиоды
      delay_ms(20); //Длинная задержка
      Pause(5);         //Пауза с погасшими светодиодами
}

int main (void)
{
      set_tris_D(0xFF); // Настройка порта D для вывода
      while(1)    //Бесконечный цикл
      {
        P(); P(); P(); //. . .
        D(); D(); D(); //- - -
        P(); P(); P(); //. . .
        Pause(100);
      }
   }
```

Читать код программы, повторяя его на языке FlowCode, будем с конца: трижды повторить функцию *P*, трижды повторить функцию *D*, трижды повторить функцию *P* и следом функция *Pause*.

Функциям языка Си будут соответствовать макросы FlowCode, имена для которых оставим оригинальные. Длительности сигналов увеличим, чтобы можно было увидеть работу программы в отладочном режиме, и вместо функции *Pause* используем программный компонент **Delay**. И пока не будем программу помещать в бесконечный цикл.

Вид основной программы получится таким:



Первым используем счетный цикл, внутри которого повторим трижды вызов макроса **P**, который еще предстоит написать.

Затем повторим это для макроса **D**.

И опять используем макрос P.

Завершит программу выключение всех светодиодов и пауза в 5 секунд.

Длительность свечения светодиодов для точки выберем 1 секунда, для тире 3 секунды.

При таких длительностях запуск отладки в программе FlowCode позволит удобно наблюдать за результатом.

В качестве порта вывода (подключения светодиодов) в данный момент можно использовать порт A, задаваемый по умолчанию. При необходимости порт, как и длительности, можно заменить другим.

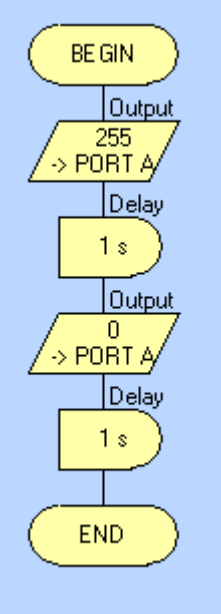
Когда программа готова, можно записать два необходимых для ее работы макроса.

Для этого двойным щелчком по программному компоненту открываем диалог, в котором выбираем кнопку ***OK&Edit Macro*** или в основном меню выбираем пункт **Macro** и команду **Show**, где выбираем по имени нужный нам макрос. Подразумевается, что сами макросы мы создавали сразу после установки программного компонента **Macro**.

И еще одно замечание: автор использует подключение светодиодов между выводом и плюсом питания. Удобнее (для отладки в FlowCode) использовать подключение светодиодов между выводом и общим проводом.

Рис. 5.26. Вид основной программы SOS в программе FlowCode

Вместе с внесенными нами изменениями подпрограмма (макрос P) для точки Морзе приобретает вид:



Все выводы порта A мы устанавливаем в высокое состояние из-за измененного нами способа подключения светодиодов.

Поскольку мы отказались от функции *Pause*, эту функцию мы повторим, устанавливая все выводы порта A в низкое состояние после выбранной нами паузы в 1 секунду, и делаем такую же паузу с погашенными светодиодами.

Рис. 5.27. Вид макроса для точки Морзе

Вид макроса D будет отличаться только длительностью первой паузы, что вытекает из вида функции, предложенной автором.

Теперь можно запустить отладку программы:

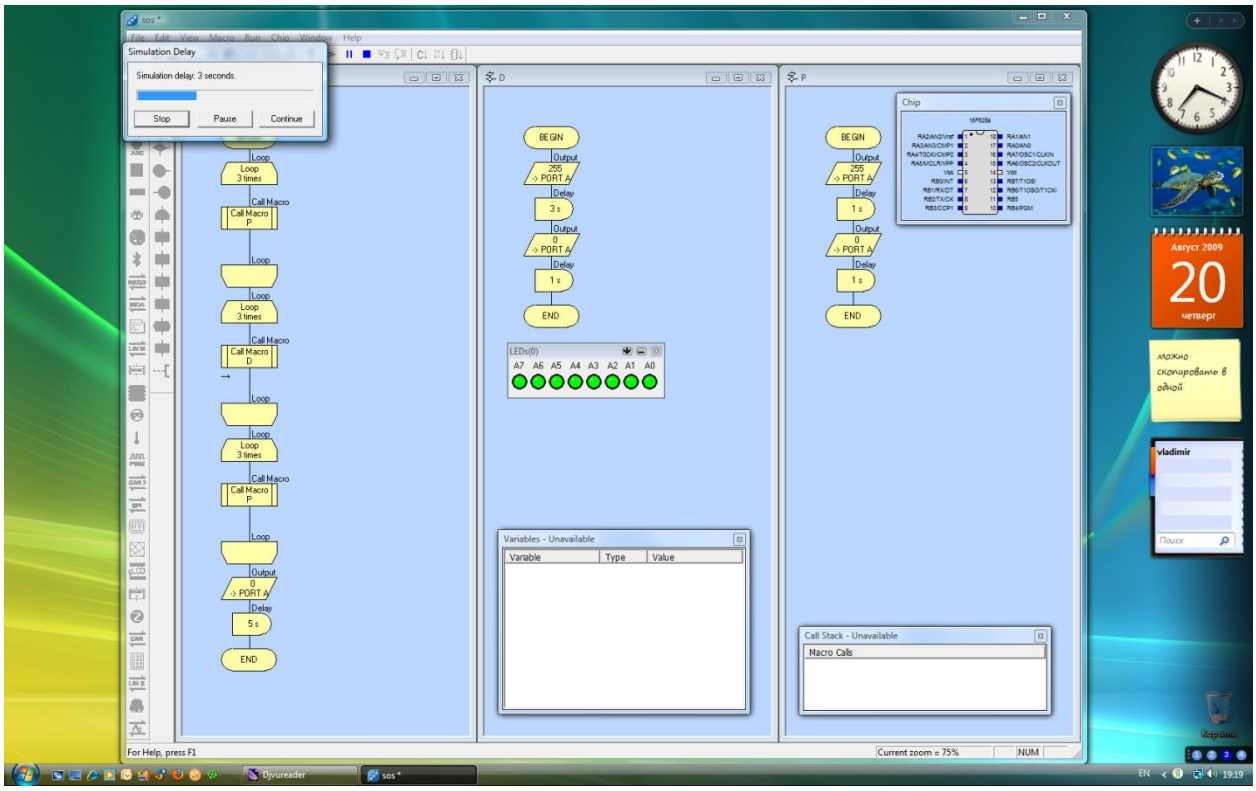


Рис. 5.28. Отладка программы SOS

Поскольку программа FlowCode имеет одинаковый вид и для PIC, и для AVR контроллеров, можно использовать программы из учебника, предназначенные для любого типа микроконтроллеров.

Например, вот программа «Бегающие глаза»:

«К выводам порта B подключены переключатели, а к выводам порта D - светодиодные индикаторы. Эффект "бегающих глаз" может быть создан путем поочередного включения двух светодиодов при выключенных остальных или, наоборот, поочередного выключения двух светодиодов при включенных остальных. Один из этих двух типов выбирается с помощью переключателя, подсоединенного к выводу 7 порта B. Остальные входы этого порта определяют скорость "бега" (коэффициент от 0 до 127)».

Листинг программы есть на CD-диске, приложении к книге:

```
#include <avr/io.h>
#include <avr/delay.h>

unsigned long DelayCount;
unsigned long Velocity = 0;
unsigned char EyeType = 0;

void ShowEyes(int i)
{
  if(EyeType) PORTD = ~i; else PORTD = i;
  _delay_loop_2(DelayCount);
}

int main (void)
{
  DDRB = 0x00;
  DDRD = 0xFF;
  while(1)
  {
    Velocity = PINB;
    if(Velocity > 127)
    {
      Velocity -= 127;
      EyeType = 1;
    }
    else EyeType = 0;
    DelayCount = 500 + (Velocity * 50);
    for(int i = 1; i <= 8; i = i*2) ShowEyes(i * 16 + i);
    for(int i = 8; i > 1; i -= i/2) ShowEyes(i * 16 + i);
  }
  }
```

Полезнее «перевести» листинг программы на графический язык, но прежде, не вижу в этом ничего плохого, можно посмотреть, например, что имеет автор в виду, когда говорит о двух вариантах.

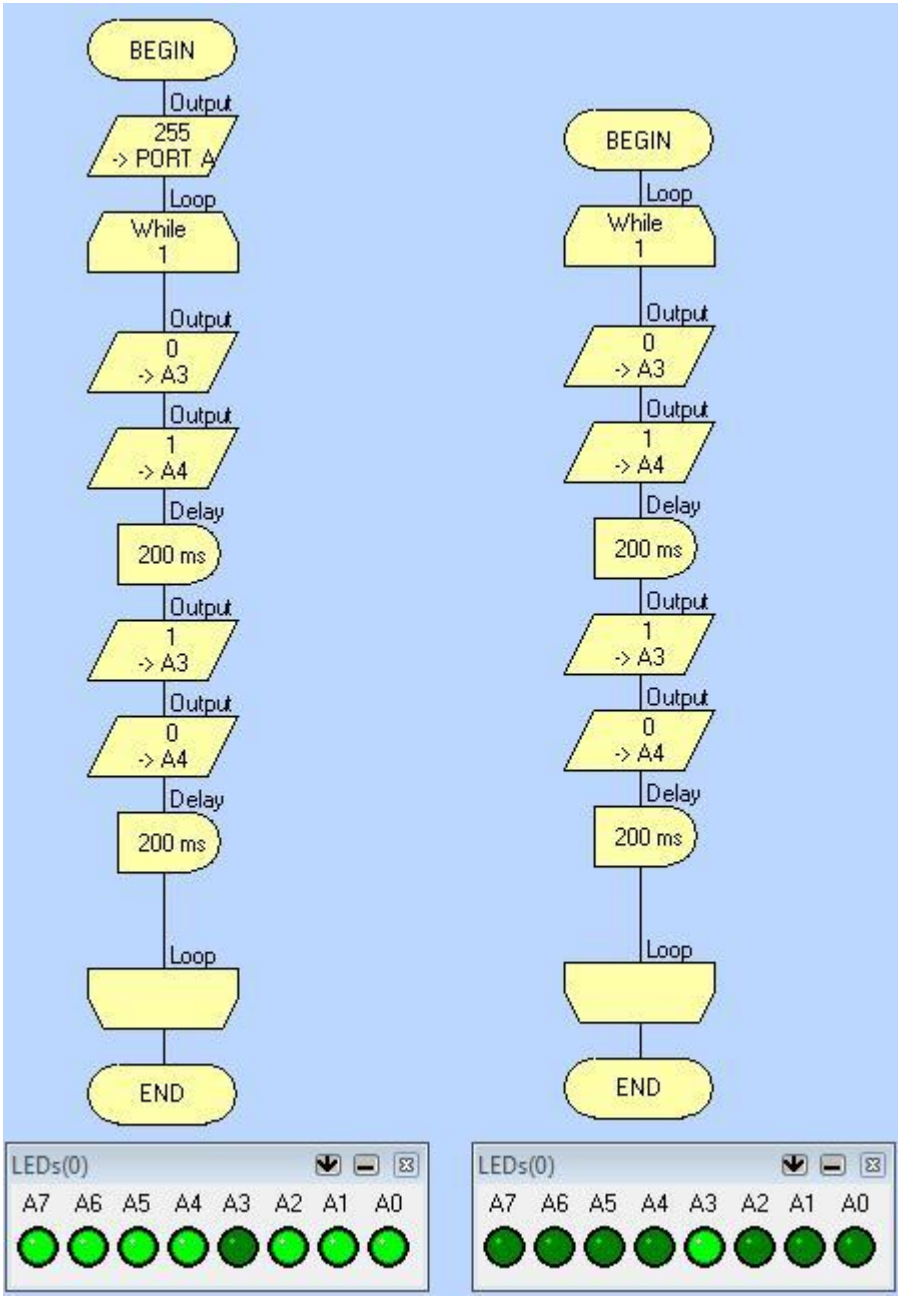


Рис. 2.29. Два варианта программы

Затем, скажем, создать программу с переключением между вариантами, оформляя варианты как две подпрограммы. На этом этапе можно убедиться, что программа, выполняющая одинаковые действия, может быть написана разными способами.

Вот один из вариантов.

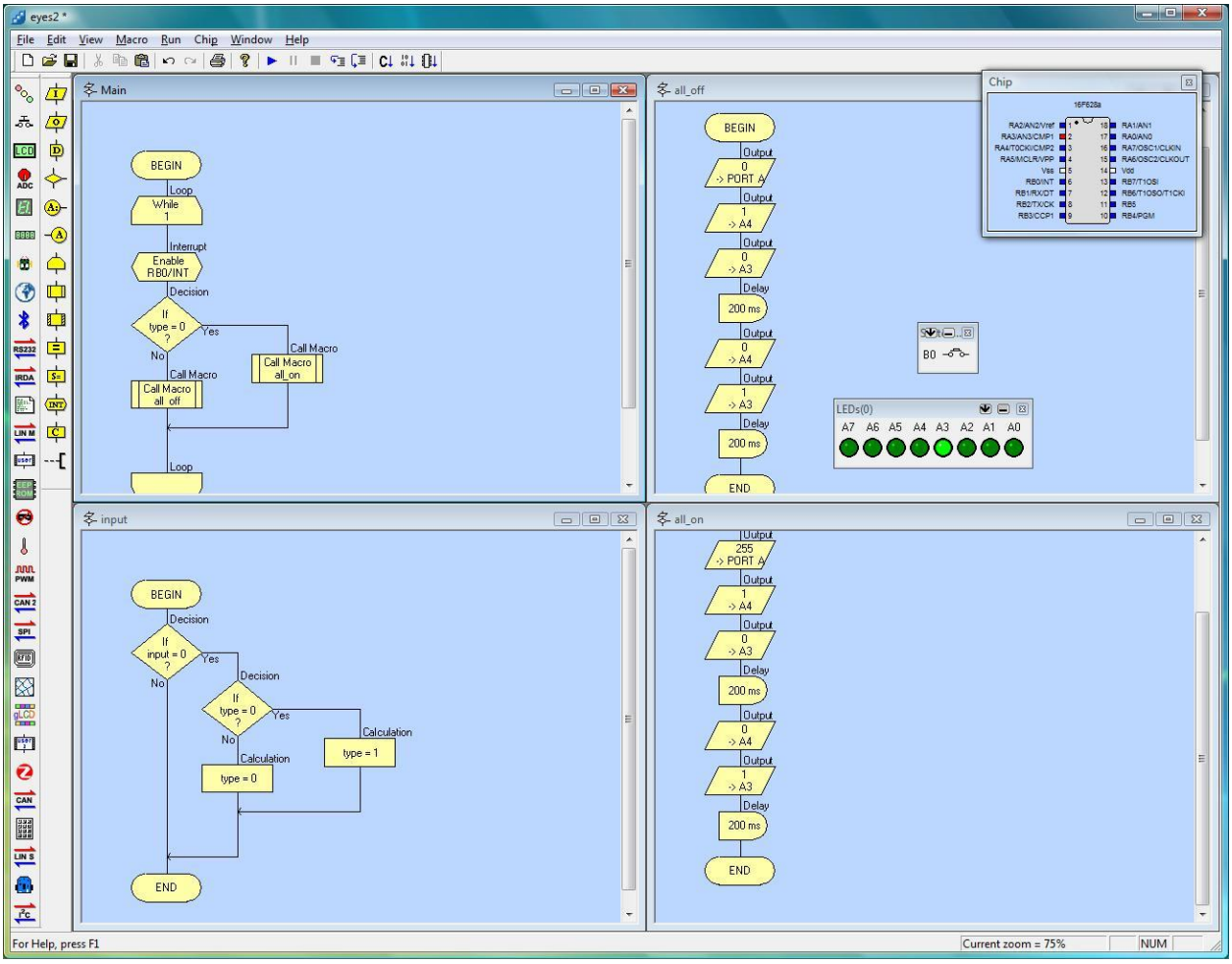


Рис. 2.30. Один из вариантов программы

Здесь используется выключатель, соединенный с выводом 0 порта B. Это позволяет применить прерывание при изменении состояния вывода RB0.

Несложно заметить, что часть программы с «мигающими» светодиодами остается одинакова в обеих подпрограммах, а меняется только «фон». Следовательно, можно убрать из подпрограмм эту часть, перенеся ее в основную программу.

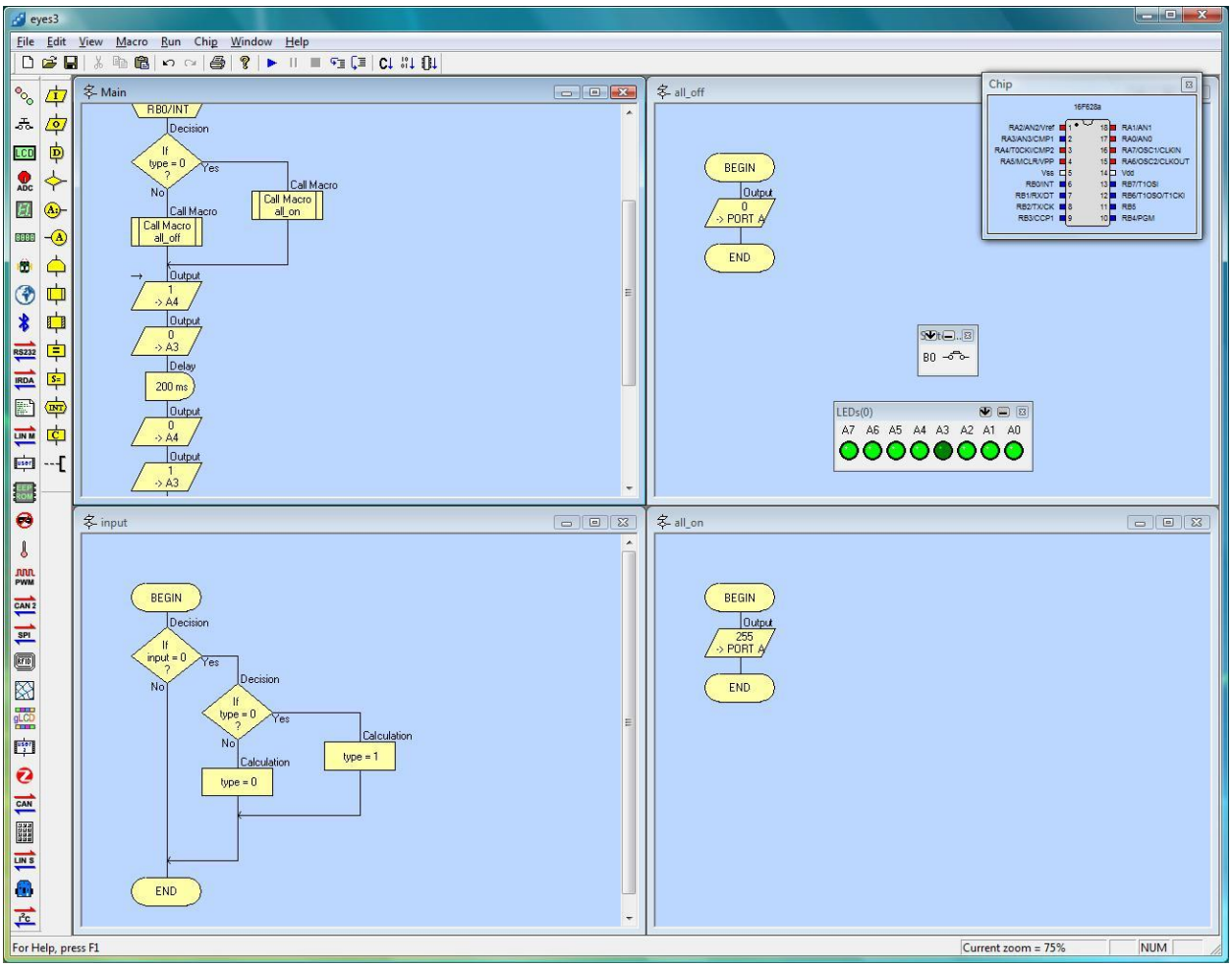


Рис. 2.31. Второй вариант программы

Варьируя разные написания программ, можно проверить, как эти вариации сказываются, например, на общем количестве кода, то есть, насколько рационально используется памяти микроконтроллера.

Затем можно перейти к той части, где меняется скорость «мигания», заменив значение в компоненте **Delay** переменной.

В конечном итоге можно постараться создать программу, которая в точности будет соответствовать авторской в книге Шпака Ю.А. При этом можно просматривать полученный результат в виде кода на языке Си и сравнивать его с листингом.

Заключение

Обязательно ли использовать учебник? Нет. Но, если вы используете учебник, то полезно будет повторить рекомендованные им программы в FlowCode. Чем привлекательна книга Шпака Ю.А., так это тем, что помимо языка Си вы узнаете много о самих контроллерах, да и язык Си в применении к контроллерам имеет особенности, которые лучше узнать от человека в этом хорошо разбирающегося.

Я не настаиваю на полной и незыблемой плодотворности того варианта изучения работы с микроконтроллером, который описан здесь, но считаю его вполне жизнеспособным. А для тех, кто пользуется операционной системой Linux, напомню, что есть полнофункциональная программа, похожая на FlowCode, с графическим языком программирования, которая называется KTechlab.

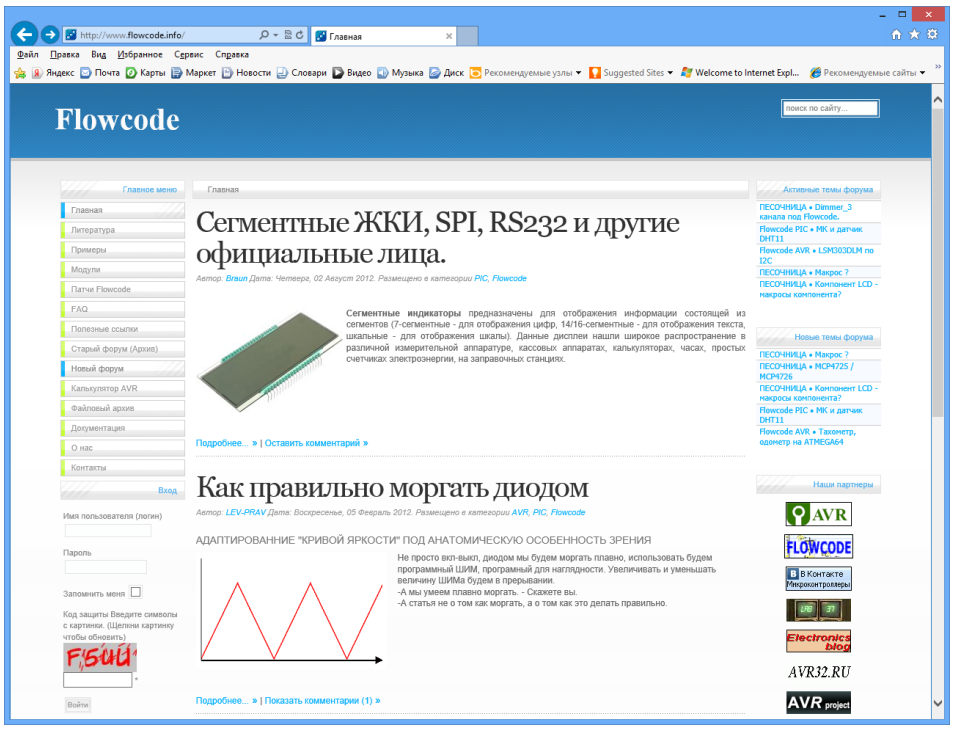
Для любителей, как мне кажется, полезно использовать все средства, включая книгу Шпака, чтобы научиться создавать свои программы для микроконтроллеров. Но еще полезнее самостоятельно придумывать устройства, где возможности микроконтроллера проявляются в полной мере, а сами устройства ближе к вашим интересам, чем приводимые учебные примеры.

Наверное, если вы еще не начинали освоение микроконтроллеров, не следует сразу браться за сложные устройства, представляющие для вас несомненный интерес, сделайте несколько простых устройств, изучайте микроконтроллер и программирование, а параллельно думайте о том устройстве, всегда есть над чем поразмышлять, которое вы сделаете спустя немного времени, когда освоитесь с созданием устройств на базе микроконтроллеров.

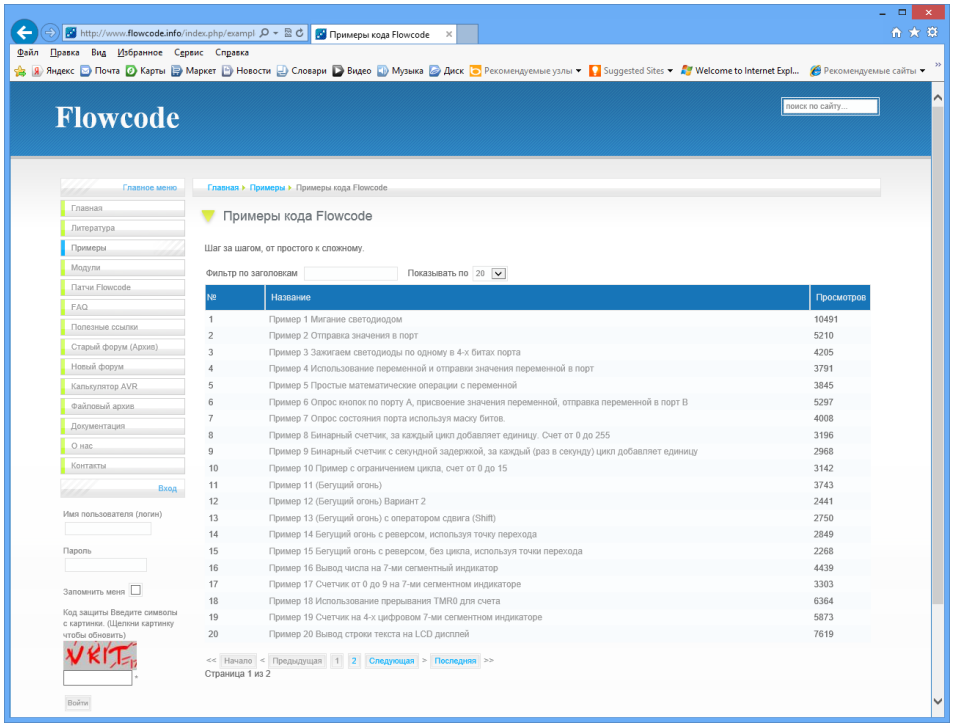
Flowcode.info – русскоязычный сайт пользователей

http://flowcode.info/

Сайт существует несколько лет, сменив «движок» и форум, следуя появлению новых версий программы, и отдавая предпочтение тем версиям, с которыми удобнее работать:

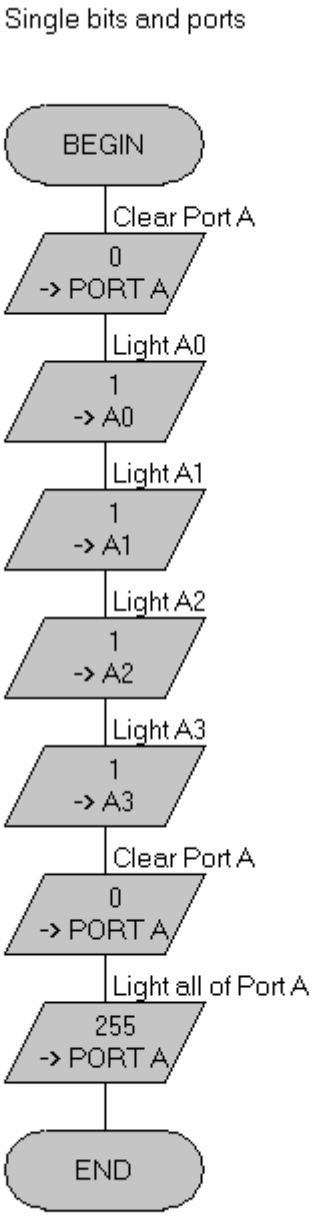


Начинающие могут ознакомиться с примерами простых программ:



Вот некоторые из примеров:

Зажигаем светодиоды по одному в 4-х битах порта



Старт программы

Отправляем "0" в порт. На всех выводах порта А - низкий уровень. Все светодиоды погашены.

Отправляем "1" в регистр "0" порта А, зажигается первый светодиод.

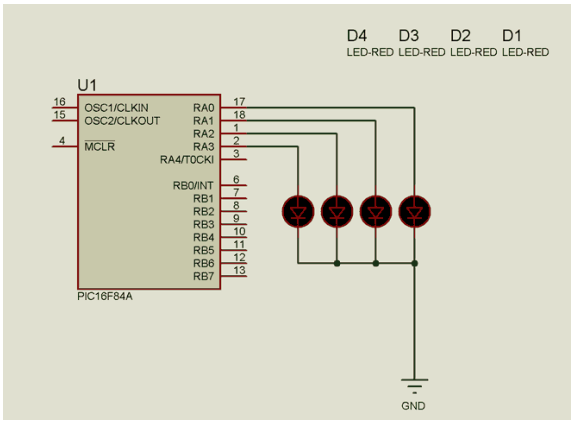
Отправляем "1" в регистр "1" порта А, зажигается второй светодиод.

Отправляем "1" в регистр "2" порта А, зажигается третий светодиод.

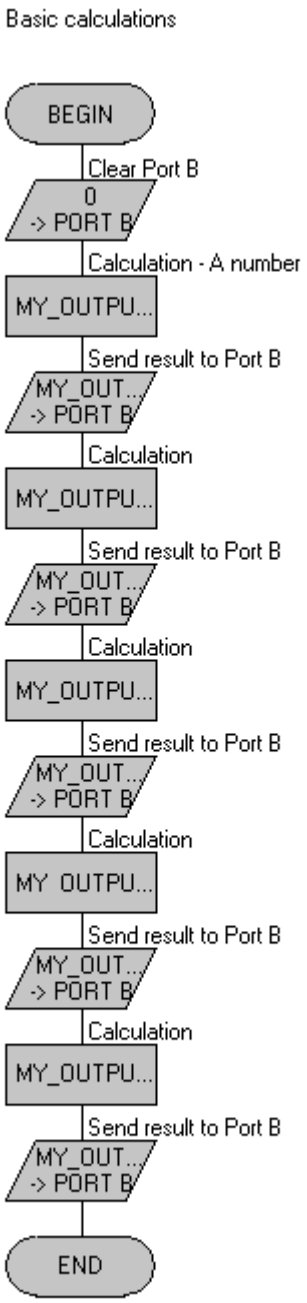
Отправляем "1" в регистр "3" порта А, зажигается четвертый светодиод.

Отправляем "0" в порт. На всех выводах порта А - низкий уровень. Все светодиоды погашены.

Отправляем "255" в порт. На всех выводах порта А - высокий уровень. Все светодиоды горят.



Простые математические операции с переменной



Отправляем "0" в порт, все светодиоды погашены.

Присваиваем переменной MY_OUTPUT значение 10

Отправляем переменную в порт, зажигается светодиод 2 и 4

Прибавляем к переменной MY_OUTPUT число 4 (MY_OUTPUT = MY_OUTPUT + 4). Теперь переменная имеет значение 14

Отправляем переменную в порт, зажигаются светодиоды 2,3 и 4

Отнимаем из переменной число 4 (MY_OUTPUT = MY_OUTPUT + 4). Значение переменной опять 10

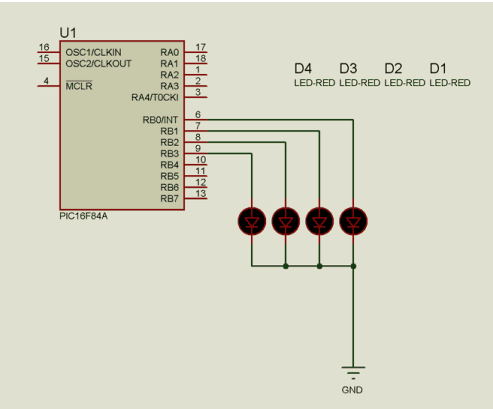
Отправляем переменную в порт, гаснет светодиод 3, горят 2 и 4

Делим переменную на два (MY_OUTPUT = MY_OUTPUT / 2). Значение переменной теперь 5

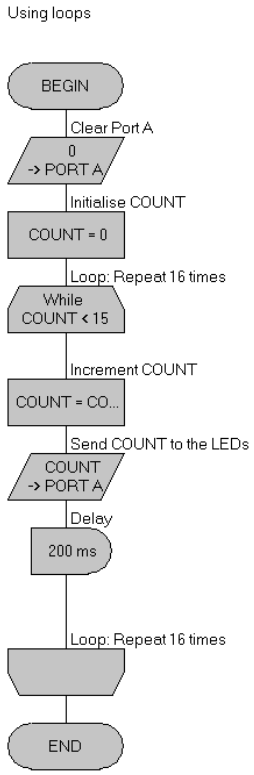
Отправляем переменную в порт, гаснут светодиоды 2 и 4, загораются 1 и 3

Умножаем переменную на 3 (MY_OUTPUT = MY_OUTPUT * 3). Значение переменной теперь 15

Отправляем переменную в порт, загораются все 4 светодиода



Пример с ограничением цикла, счет от 0 до 15. По достижении числа 15 программа заканчивает работу



Очистка порта A, все светодиоды погашены.

Присваиваем переменной COUNT значение 0

Повтор цикла, пака переменная COUNT меньше 15. При значении переменной равным 15-ти, выйти из цикла. Конец программы.

Увеличиваем переменную COUNT на единицу (COUNT = COUNT + 1).

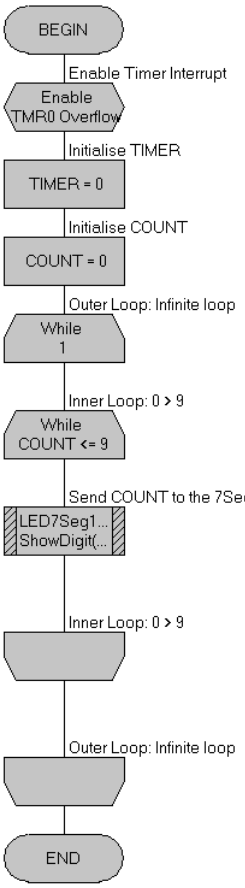
Отправляем переменную в порт A.

Задержка 200мс

Возврат к началу цикла

Использование прерывания по счетчику переполнения для отсчета точных временных интервалов

Счет индикатора от 0 до 9



Активация прерывания по таймеру. При частоте кварца 19,660800 Мгц и предделении 1:256 переполнение счетчика происходит 75 раз в секунду. Значит, макрос INTERRUPT_TMR0 тоже вызывается 75 раз в секунду.

Переменной TIMER присваивается 0

Переменной COUNT присваивается 0

Старт бесконечного цикла

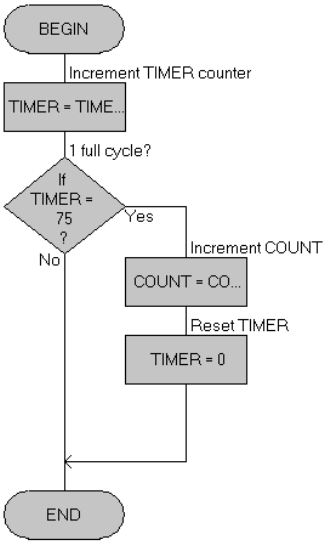
Цикл пока переменная COUNT меньше 9-ти

Отправка переменной COUNT на 7-ми сегментный индикатор

Возврат к циклу пока переменная COUNT меньше 9-ти

Возврат к бесконечному циклу

Макрос INTERRUPT_TMR0



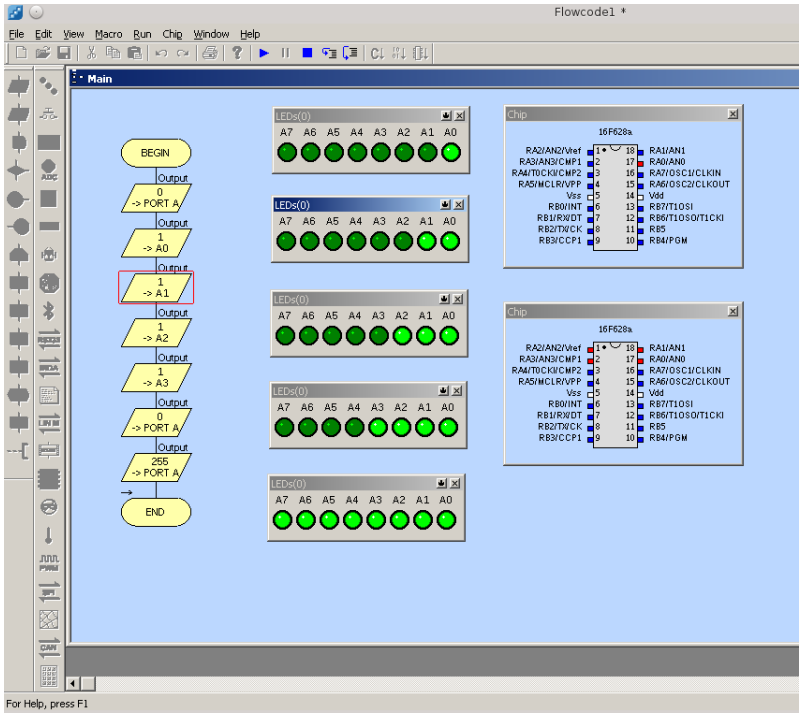
Увеличение переменной TIMER на единицу (TIMER = TIMER + 1)

Сравнение равна ли переменная TIMER 75-ти

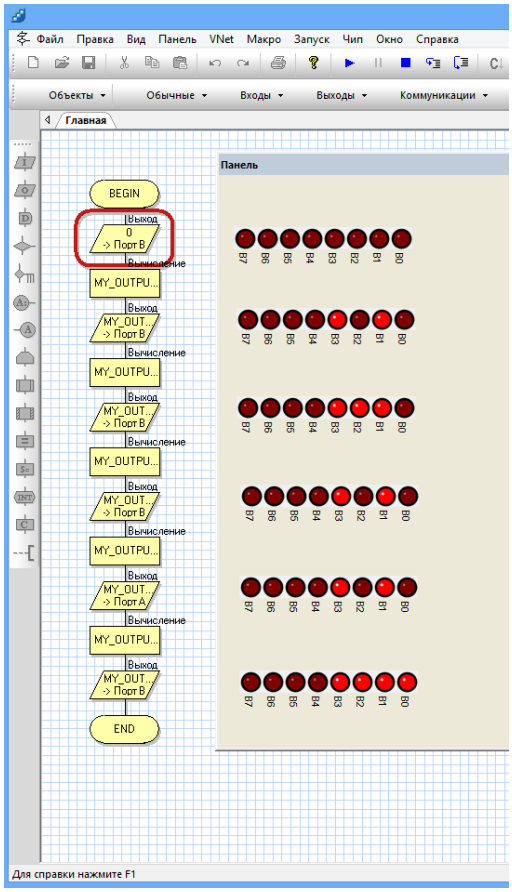
Если да, добавляем к переменной COUNT единицу (COUNT = COUNT + 1)

Обнуляем переменную TIMER

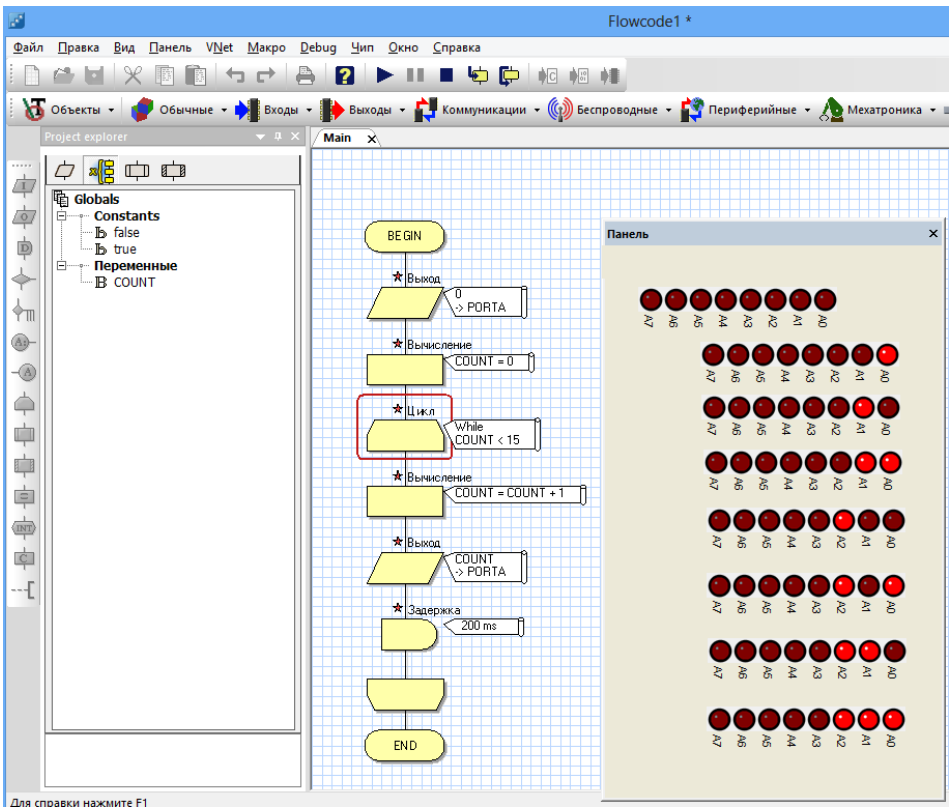
Давайте посмотрим, как работают эти примеры в Flowcode, начиная с 3 версии и следуя появлениям новых версий программы.



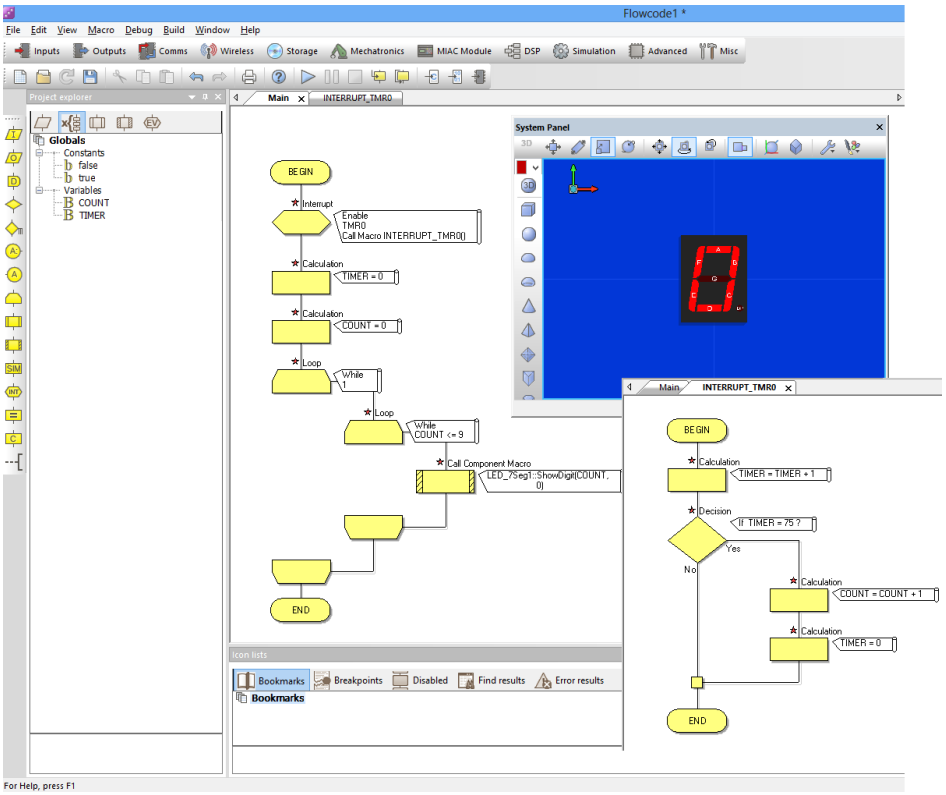
Первый пример в программе Flowcode 3



Второй пример в программе Flowcode 4 (увы, с глюком)



Третий пример в программе Flowcode 5



Четвёртый пример в программе Flowcode 6

Статьи на сайте позволяют эффективнее использовать программу с дополнительными компонентами.

Сегментные ЖКИ.

Автор: Braun

Сегментные индикаторы предназначены для отображения информации состоящей из сегментов (7-сегментные - для отображения цифр, 14/16-сегментные - для отображения текста, шкальные - для отображения шкалы). Данные дисплеи нашли широкое распространение в различной измерительной аппаратуре, кассовых аппаратах, калькуляторах, часах, простых счетчиках электроэнергии, на заправочных станциях.

Основными техническими характеристиками дисплеев данного типа являются:

• тип ЖК (TN, STN) - определяет качество изображения (STN - лучше, но и дороже);
режим управления (статический, мультиплексированный) - в случае использования мультиплексированного режима уменьшается количество выводов. Как правило, в случае необходимости отображать более 40 сегментов используют мультиплексирование (несколько общих выводов - до 4);
• напряжение питания (в основном 3,0В);
• угол зрения - это угол падения взгляда на панель ЖКИ, при котором контрастность изображения максимальна (3, 6, 9 и 12 часов, чаще встречаются 6);
• геометрические размеры, расположение крепежных отверстий (элементов);
• тип и цвет подсветки (отражение, просвет, отражение + просвет);
• диапазон рабочих температур;

Большинство индикаторов имеют **статический** тип подключения, это означает, что к каждому сегменту индикатора идет отдельный соответствующий электрический вывод, без объединения в группы и без общих контактов. Общей у них является только подложка.

Именно эти индикаторы наиболее дешевы в производстве, так как не имеют внутреннего контроллера и по подключению напоминают LED индикаторы, очень распространены в любительских конструкциях. Но есть несколько существенных отличий, как положительных, так и создающих определенные трудности.

Начнем с положительных:

1. Ток потребления. В LED семисегментниках каждый сегмент – это отдельный светодиод с потреблением около 15-20 миллиампер. Соответственно, чтобы зажечь все 7 сегментов + точку, потребуется приблизительно 150 мА. Подключив индикатор напрямую к контроллеру, мы имеем все шансы сжечь порт. Многие контроллеры ограничены током 100-150 мА на порт.

ЖКИ индикаторы потребляют в разы меньший ток, исчисляемый единицами микроампер. Например, у меня в хламе валяется системный блок компьютера с ЖКИ индикатором (температура, часы, работа винта и т.д.). Блок питается от одной батарейки типа CR2320, а часы исправно идут уже 4 года.

2. Читабельность. В отличие от LED, LCD (жки) индикаторы не «слепнут» на солнце. Они отлично и четко видны даже при прямом солнечном свете. Это делает их незаменимыми в устройствах, используемых вне помещения. А мягкая подсветка позволяет вполне комфортно использовать их при недостаточном освещении.

3. Цена. Новые ЖКИ индикаторы по стоимости на 20-80% дешевле аналогичных LED индикаторов, цена около 2-2,5 долларов за 4-х знаковый индикатор. Более того, их можно часто найти в каких-нибудь игрушках или китайских часах-будильниках. Да и на барахолке они часто в хламе валяются, можно купить вообще за копейки.

Теперь немного о минусах:

1. Питание переменкой. Специфика ЖКИ индикаторов такова, что питать их можно только переменным напряжением (меандр).

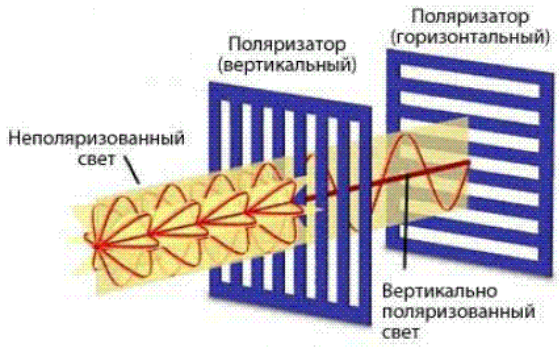
2. Стекло. Индикаторы изготовлены на стеклянной подложке, что требует определенной аккуратности в обращении с ними.

3. Статическое подключение требует большого количества выводов, напрямую к контроллеру не подключить, не хватит ног. Даже в динамике.

Немного теории:

Подробнее, что такое сегментный ЖКИ и что такое поляризация света, а так же теорию корпускулярного дуализма можно найти в интернете, заострять на этом внимание не будем.

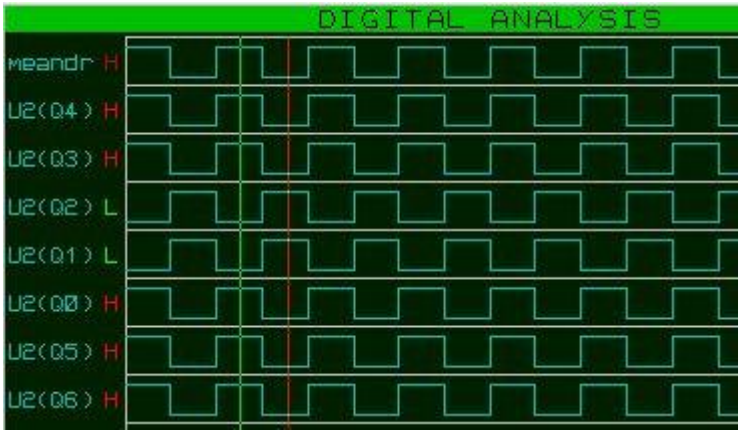
Дисплей управляется магнитным полем, которое ориентирует кристаллы, которые, в свою очередь, поляризуют проходящий через них свет.



Проходя через поляризаторы (в нашем случае плёнка на дисплее – два одинаковых поляризатора), свет не встречает препятсвий, мы видим прозрачный дисплей. Чтобы «зажечь» сегмент, свет нужно повернуть на 90 градусов, и мы увидим чёрное пятно сегмента. Тут как раз и применяют кристаллы, расположенные между двумя поляризаторами. Подаём ток, кристаллы ориентируются так, что поляризованный свет, проходя через них, меняет угол, и задерживается вторым поляризатором.

Теперь как этим управлять. Сегмент – это, по сути, конденсатор, на него нужно подать ток, ток создаст электромагнитное поле, ориентирующее кристаллы. Но просто к ноге контроллера его подключить нельзя, так как сопротивление конденсатора после заряда станет бесконечным, и ток перестанет течь, а, значит, электромагнитное поле исчезнет, и сегмент погаснет. Значит, ток нужен переменный.

Более того, ток нужно подавать в противофазе. А именно, для свечения сегментов общая подложка (COM) должна иметь положительный потенциал, а сегмент, который мы хотим зажечь, иметь отрицательный. И частота по справочным данным обычно 30-300Гц.



На рисунке видно, что в момент, когда на выводе meandr – общий, COM вывод индикатора (зеленая линия) – высокий уровень, на двух других (Q2 и Q1) низкий. На остальных тоже высокий уровень. Именно те два сегмента, с низким потенциалом, и будут «светиться», так как обеспечивается обязательное условие для «свечения» - единица на общем выводе и ноль на сегменте.

В следующую смену потенциала, на общем выводе низкий уровень (красная линия). Ни один из сегментов «светиться» не будет, так как при низком потенциале на общем выводе уже не важно, какой потенциал на сегментах. Тем не менее, концепция работы индикатора поддерживается, питается он переменным током.

Добавление русского шрифта в компонент «Графический дисплей gLCD»

В программе Flowcode есть такой компонент gLCD. Это модель цветного графического дисплея с чипом Philips 8833, который используется в более чем 30 моделях мобильных телефонов Nokia. Кстати, по заявлениям разработчиков с 4.02.2010 года, компонент поддерживает дисплей от Siemens S65. Но я лично не пробовал, сказать ничего не могу... Но кода в файле заметно прибавилось.

Но, к сожалению, не всё так хорошо для русскоязычных пользователей этой чудной программы: разработчиками не предусмотрены кириллические символы шрифтов. Правда, компонент позволяет в настройках подставлять свои собственные шрифты. Как это делается, описано уважаемым **mim** вот в этой ветке форума - **Flowcode и дисплеи, индикация**. Но и у этого способа есть парочка недостатков.

- Нужно вручную создавать готовые шрифты, используя отдельную программу.
- Количество символов ограничено 95. В которые, естественно, не поместятся сразу и русский, и латинский алфавит, цифры и спецсимволы.
- Несоответствие раскладки стандартной компьютерной клавиатуры и выводимых символов.

Но есть и неоспоримое преимущество – результат вывода на дисплей нового шрифта виден при симуляции прямо в программе.

Описание макросов вывода информации на дисплей находится в виде СИ-кода в файле компонента **PIC_gLCD.c**, находящегося в паке **Components** в директории установки программы Flowcode.

Файл можно менять по своему усмотрению. На чём и основан данный пример. Итак...

Присвоение переменным массивов со шрифтами в этом файле происходит в разделе //ASCII Pixel data, строка 412:

```
#ifdef _BOOSTC
rom char* ASCII1 =
%j
rom char* ASCII2 =
%k
rom char* ASCII3 =
%l
```

и т.д....

Данные для массивов берутся из файла **PIC_gLCD.ocx**, но его править нет смысла, можно просто подставить свои шрифты в настройках компонента gLCD.

А вот за вывод шрифта в этом файле отвечает раздел //Routine for displaying characters,

строка 1023, а именно цикл, начинающийся в строке 1053.

```
for (i=0;i<length_Str;i++) //Start at beginning of string and work along
```

Этот цикл перебирает строку пользователя и сравнивает код очередного символа с кодом в массиве, заранее загруженном в раздел //ASCII Pixel data. После чего отправляет содержимое совпавшей ячейки массива на «печать».

Все что теперь нужно, это подставить свой массив со всеми нужными шрифтами.

Как я уже упоминал, по умолчанию flowcode загружает только 95 символов. С кодом от 32 до 126 включительно. В них вписываются все спецсимволы, цифры, большие и маленькие буквы латинского алфавита. Все они разбиты на группы по 12 символов в массиве и присвоены своим переменным ASCII1 - ASCII8 (в массиве ASCII8 - 11 символов), итого 95.

Ну, с теорией покончено, начнем практику.

Перед началом обязательно сделайте копию файла **PIC_gLCD.c**. Если что-то напортачите, всегда можно будет откатиться назад.

Начнем править файл.

Во-первых, нам уже не понадобятся родные массивы символов. Для исключения их из кода просто закомментируйте строчки от 413 до 450.

Получится вот так:

```
//ASCII Pixel data
/*#ifdef _BOOSTC
rom char* ASCII1 =
%j
rom char* ASCII2 =
```

```
%k
rom char* ASCII3 =
%l
rom char* ASCII4 =
%m
rom char* ASCII5 =
%n
rom char* ASCII6 =
%o
rom char* ASCII7 =
%p
rom char* ASCII8 =
%q
#endif

#ifdef HI_TECH_C
const char ASCII1[] =
%j
const char ASCII2[] =
%k
const char ASCII3[] =
%l
const char ASCII4[] =
%m
const char ASCII5[] =
%n
const char ASCII6[] =
%o
const char ASCII7[] =
%p
const char ASCII8[] =
%q
#endif
*/
```

Теперь обратимся к разделу //Routine for displaying characters

В конце вышеупомянутого цикла, в строке 1095, поменяйте цифру 95 на 96. Мы ведь не будем останавливаться на 95 символе.

И сразу после последнего подправленного вами сравнения, после «}»:

```
else if (pos_Str < 96)
{
position = ((pos_Str - 84)*5) + count;
temp[count]=ASCII8[position];
}
```

Добавьте следующий код:

```
else if (pos_Str < 108)
{
position = ((pos_Str - 96)*5) + count;
temp[count]=ASCII9[position];
}
else if (pos_Str < 120)
{
position = ((pos_Str - 108)*5) + count;
temp[count]=ASCII10[position];
}
else if (pos_Str < 132)
{
position = ((pos_Str - 120)*5) + count;
```

```
temp[count]=ASCII11[position];
}
else if (pos_Str < 144)
{
position = ((pos_Str - 132)*5) + count;
temp[count]=ASCII12[position];
}
else if (pos_Str < 156)
{
position = ((pos_Str - 144)*5) + count;
temp[count]=ASCII13[position];
}
else if (pos_Str < 168)
{
position = ((pos_Str - 156)*5) + count;
temp[count]=ASCII14[position];
}
else if (pos_Str < 180)
{
position = ((pos_Str - 168)*5) + count;
temp[count]=ASCII15[position];
}
else if (pos_Str < 192)
{
position = ((pos_Str - 180)*5) + count;
temp[count]=ASCII16[position];
}
else if (pos_Str < 204)
{
position = ((pos_Str - 192)*5) + count;
temp[count]=ASCII17[position];
}
else if (pos_Str < 216)
{
position = ((pos_Str - 204)*5) + count;
temp[count]=ASCII18[position];
}
else if (pos_Str < 228)
{
position = ((pos_Str - 216)*5) + count;
temp[count]=ASCII19[position];
}
```

Теперь осталось указать, где находятся переменные с новыми массивами символов, поскольку старые мы закомментировали.

Поднимаемся в начало файла в строку номер 398. Тут мы видим, что директивой #include в текст программы включено содержимое файла **string.h**. Он находится в папке «**Flowcode V4\boostc\include**» компилятора BOOSTC. Положите файл из архива к этой статье **codepage.h** в указанную папку, а рядом с директивой строки 398 добавьте еще одну директиву с указанием на этот файл. Получится так:

```
#include <string.h>
#include <codepage.h>
```

Теперь при компиляции проекта содержимое файла **codepage.h** (массивы всех ASCII символов) будет добавляться к файлу **PIC_gLCD.c**.

Что теперь нужно сделать, чтобы заставить проекты Flowcode писать на дисплее русскими буквами:

- Создаем в программе строковую (String) переменную, ну, например, «hello_world».
- В основном окне программы перед основным циклом вставляем компонент C Code.
- В него вписываем следующее: char
  `FCV_HELLO_WORLD[]={207,240,232,226,229,242,160,204,232,240,0};`

Каждое число - это буква, цифра или какой-нибудь символ (см. таблицу ниже).
От 32 до 127 - латинские буквы, цифры и знаки пунктуации (нам они тут не пригодятся, они и так выводятся в FC нормально); 128 - 255 это то что нам пригодится - буквы кириллицы, всякие прикольные символы и т.д.
То есть, на Си мы создаем строковый массив в переменной hello_world, а массив через запятую заполняем числовыми кодами значений символов. В нашем случае это:
207 - П
240 - р
232 - и
226 - в
и т.д.
В конце обязательно поставьте «,0».
Все, теперь просто вызываем в нужном месте на дисплей печать этой переменной.
Конечно, при симуляции в FC этого не увидишь, вставки на языке Си не симулируются в программе.
Я делаю так:
Макросом print вывожу на экран слово латинскими буквами, это позволяет увидеть где оно расположено, размер и выводится ли вообще. Перед ним вставляю C Code, начинающийся знаком начала комментария /*, а после него знаком завершения комментария */
Теперь при компиляции программа проигнорирует этот участок, а при симуляции проигнорирует Си вставки. И не будет перебивания переменных одна другой.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL 0	SOH 1	STX 2	ETX 3	EOT 4	ENQ 5	ACK 6	BEL 7	BS 8	HT 9	LF 10	VT 11	FF 12	CR 13	SO 14	SI 15
1	DLE 16	DC1 17	DC2 18	DC3 19	DC4 20	NAK 21	SYN 22	ETB 23	CAN 24	EM 25	SUB 26	ESC 27	FS 28	GS 29	RS 30	US 31
2	SPC 32	! 33	" 34	# 35	$ 36	% 37	_ 38	' 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
3	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
4	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
5	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95
6	` 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
7	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	| 124	} 125	~ 126	DEL 127
8	Ђ 128	Ѓ 129	‚ 130	ѓ 131	„ 132	… 133	† 134	‡ 135	€ 136	‰ 137	Љ 138	‹ 139	Њ 140	Ќ 141	Ћ 142	Џ 143
9	ђ 144	‘ 145	’ 146	“ 147	” 148	• 149	– 150	— 151	□ 152	™ 153	љ 154	› 155	њ 156	ќ 157	ћ 158	џ 159
A	160	Ў 161	ў 162	Ј 163	¤ 164	Ґ 165	¦ 166	§ 167	Ё 168	© 169	Є 170	« 171	¬ 172	- 173	® 174	Ї 175
B	° 176	± 177	І 178	і 179	ґ 180	µ 181	¶ 182	· 183	ё 184	№ 185	є 186	» 187	ј 188	Ѕ 189	ѕ 190	ї 191
C	А 192	Б 193	В 194	Г 195	Д 196	Е 197	Ж 198	З 199	И 200	Й 201	К 202	Л 203	М 204	Н 205	О 206	П 207
D	Р 208	С 209	Т 210	У 211	Ф 212	Х 213	Ц 214	Ч 215	Ш 216	Щ 217	Ъ 218	Ы 219	Ь 220	Э 221	Ю 222	Я 223
E	а 224	б 225	в 226	г 227	д 228	е 229	ж 230	з 231	и 232	й 233	к 234	л 235	м 236	н 237	о 238	п 239
F	р 240	с 241	т 242	у 243	ф 244	х 245	ц 246	ч 247	ш 248	щ 249	ъ 250	ы 251	ь 252	э 253	ю 254	я 255

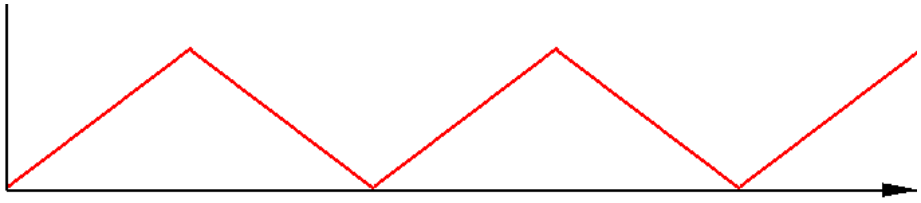
Как правильно моргать диодом

Автор: LEV-PRAV

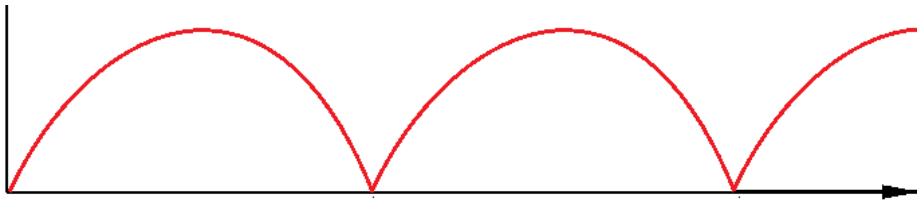
Адаптирование "кривой яркости" под анатомическую особенность зрения

Не просто вкл-выкл, диодом мы будем моргать плавно, использовать будем программный ШИМ, программный для наглядности. Увеличивать и уменьшать величину ШИМа будем в прерывании.

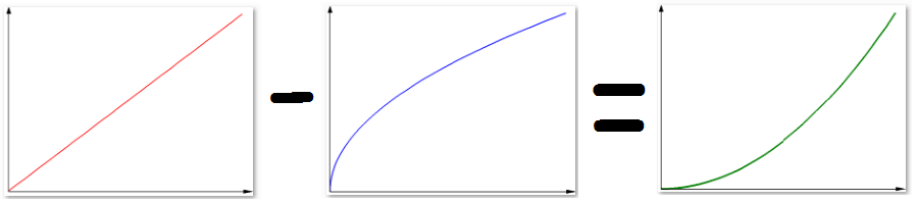
- А мы умеем плавно моргать. - Скажете вы.
- А статья не о том, как моргать, а о том, как это делать правильно.



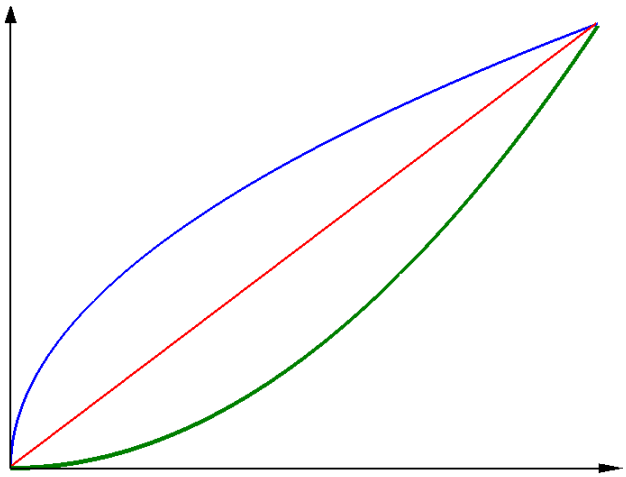
Наши глаза очень чувствительны к изменению яркости при малом количестве света. А когда света много, незначительные изменения яркости не заметны. Это нормальная функция глаза. Благодаря ей ночью в лесу можно заметить свет гнилого пенька, и при такой чувствительности не ослепнуть в зимний солнечный день.
И поэтому наш объективный график превращается вот в такой субъективный:



Надеюсь, те, кто мигал плавно диодом, замечали, что он большую часть времени горит ярко, и лишь в небольшой промежуток времени меняет яркость.
Надо от этого избавляться. А как? Давайте, выровняем кривую, точнее компенсируем за счет другой кривой. Поищем другую кривую:

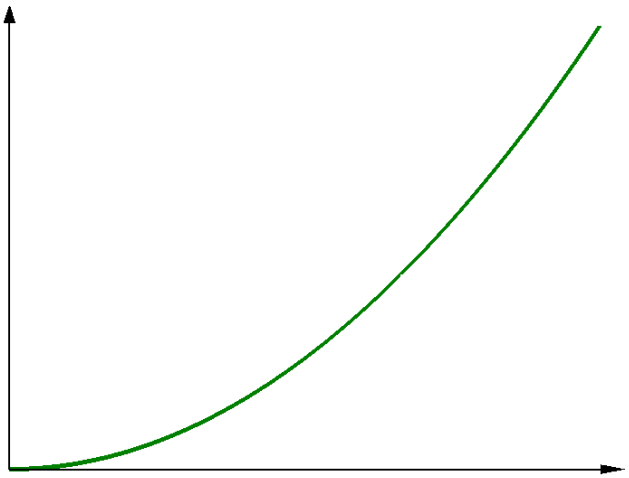


Вот такая вот арифметика – от прямой отняли кривую, получили её «зеркало».



Вам она что-нибудь напоминает?

Да, парабола. Точнее её самый кривой кусок, «пятка» . Функция этого графика: y=x*x.



Теперь подгоним всё это под нашу программу.

Есть 2 пути:

1. Табличный, занимает много памяти, но быстрый, т. к. не требует расчётов. Записывать 256 значений руками не хочется. Можете попробовать сами. Данные можно взять из «экселевского» файла, в котором записан калькулятор, созданный как раз для этой цели. В нём можно выставлять пределы кривой и её степень. А также максимальные значения счётчика, ведь не все используют 8 битный счётчик.

2. Математический. Уходит много времени на расчёты.
Адаптируем пропорции нашей параболы под наши нужды такой формулой:

PWM=STEP^2/256

Более понятный и простой эквивалент для ФК:

PWM = (STEP * STEP) >> 8

Все, что вы прочли, используется и учитывается в профессиональных устройствах управления светом.

Использование языка Си в программе Flowcode

По сути реализации программы Flowcode все макросы компонентов, иконки, модули – это блоки, написанные на языке программирования высокого уровня Си.

Это функции, к которым программа обращается, передает (или не передает) определенные параметры и получает (или не получает) результат выполнения определенного кода. Функция на Си – это завершенная по смыслу небольшая программа, которая выполняется, когда ее вызывают строкой вызова из основной программы.

Вот пример простой функции сложения:

Сначала нужно определить тип и имя глобальных переменных, которые мы будем использовать:

```
char v1;
char v2;
char v3;
```

Перед основной программой нужно поместить саму функцию:

```
void function_plus(char v1, char v2)     //идентификация функции и определение типов и имен переменных
{
v3 = v1 + v2;                            //операция сложения переменных
}
```

Теперь, если в коде необходимо произвести сложение двух чисел, нужно просто вызвать функцию function_plus и передать ей значения переменных v1 и v2:

```
function_plus(1, 2); //вызываем функцию и передаем ей значения
//кстати, вместо фиксированных значений могут быть и переменные со своими значениями
```

Результатом выполнения функции будет переменная v3, значение которой будет «3». Переменную можно использовать дальше в программе.

Но тут возникает первый справедливый вопрос - как состыковать переменные из Flowcode и переменные на Си вставках?

Для этого создатели программы Flowcode оставили возможность передавать имена и значения переменных между Си и блоками программы в обоих направлениях. Просто, при использовании переменной из Flowcode перед ее именем в блоке Си нужно написать префикс **FCV_** и имя переменной в верхнем регистре.

Например, если в программе используется переменная **button**, в блоке Си ее нужно использовать как **FCV_BUTTON.** И все значения, которые она приобретет, можно использовать в программе Flowcode.

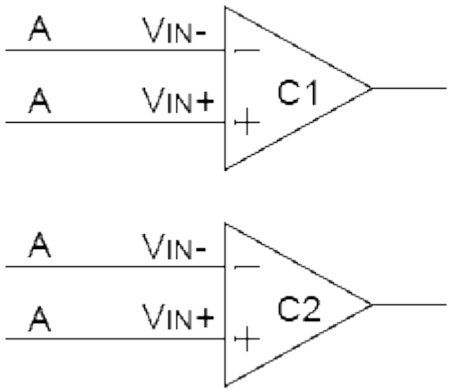
А теперь, зачем нам все это нужно и где это может пригодиться.

Программа Flowcode действительно очень удобна при создании собственной программы из «кубиков» модулей, но контроллеры очень разнообразны в своей конфигурации, и программисты Matrix Multimedia, естественно, учли самые распространенные модули контроллеров и постарались максимально разнообразить их настройку.

Однако Matrix есть еще над чем работать, и недоделанные модули приходится дописывать самим, вооружившись datasheet контроллера и вставками на Си.

Давайте, например, устраним обойденную в Flowcode функцию компараторов контроллера.

Компаратор – это аналоговый узел, сравнивающий напряжения на двух своих входах и, в зависимости от настроек, выдающий сигнал на выходе. Например, напряжение на входе 1 выше напряжения на входе 2, значит на выходе «1».



Использование компаратора очень удобно, например, при контроле напряжения батарей, где использование АЦП просто не оправдано в силу громоздкости кода последнего. Тем более, если не нужно его замерять во всем диапазоне. Просто, например, установив с помощью резистивно-стабилитронного делителя опорное напряжение в 3 вольта на одном входе, мы контролируем напряжение на втором. Если напряжение на втором опустилось ниже 3 вольт, зажегся светодиод, значит, разряжен аккумулятор, пора его заряжать.

Итак...

Общение с регистрами контроллера можно производить либо байтами, либо битами. Это означает, что если нам нужно установить все биты в единицу, нам просто достаточно в Си-блоке написать имя регистра и присвоить ему значение 255. Например:

```
cmcon = 0xFF; //установка регистра сразу одним байтом
```

Об управлении битами регистров расскажу чуть позже.

Описание всех внутренних регистров и битов контроллера находятся в файлах с именем контроллера и расширением .h, находящихся в папке компилятора Boostc (путь по умолчанию - C:\Program Files\Matrix Multimedia\Flowcode V4\boostc\include\).

Итак, нас интересует компараторы, для примера я взял распространенный микроконтроллер PIC16F628A, имеющий пару компараторов «на борту». В справке описание компараторов находится в разделе **10.0 COMPARATOR MODULE.**
Регистр, определяющий порядок функционирования компараторов, называется – **CMCON.**

Открываем файл PIC16F628A.h с описанием регистров и их битов. Ищем регистр **CMCON,** он находится в строке 44:

```
#define CMCON                   0x001F
```

Это хорошо, что он там прописан, значит, разработчики не забыли, что компараторы вообще есть в контроллере.

Дальше ищем описание битов этого регистра **CMCON**. Это важный момент, так как матриксы иногда называют биты по-своему, не так как они называются в datasheet.

Нашелся.

```
/////// CMCON Bits ////////////////////////////////////////////////////
#define C2OUT                   0x0007
```

```
#define C1OUT                   0x0006
#define C2INV                   0x0005
#define C1INV                   0x0004
#define CIS                     0x0003
#define CM2                     0x0002
#define CM1                     0x0001
#define CM0                     0x0000
```

Естественно, каждый бит может иметь значение либо «0», либо «1».

Смотрим в справочные данные, сравниваем.

R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

В первой строке указано, какие операции можно производить с этими битами.

- R - только чтение
- R/W - чтение/запись
- 0 - означает, что бит очищен, когда он принимает значение «0».

Из таблицы видно, что и в файле PIC16F628A.h, и в файле справки, есть все биты, и они называются одинаково. Значит можно в файл PIC16F628A.h больше не заглядывать, нам достаточно datasheet.

Дальше, смотрим, за что отвечает каждый бит.

bit 7 **C2OUT**: Выход компаратора 2 (это бит только на чтение, в него контроллер заносит результат сравнения компаратора 2, отсюда мы его и будем брать (считывать)).

Если C2INV = 0:
1 = C2 VIN+ > C2 VIN-
0 = C2 VIN+ < C2 VIN-

Если C2INV = 1:
1 = C2 VIN+ < C2 VIN-
0 = C2 VIN+ > C2 VIN-

bit 6 **C1OUT**: Выход компаратора 1 (это бит только на чтение, в него контроллер заносит результат сравнения компаратора 1, отсюда мы его и будем брать (считывать)).

Если C1INV = 0:
1 = C1 VIN+ > C1 VIN-
0 = C1 VIN+ < C1 VIN-

Если C1INV = 1:
1 = C1 VIN+ < C1 VIN-
0 = C1 VIN+ > C1 VIN-

bit 5 **C2INV**: Выход компаратора 2 инверсный (если выход нужно «перевернуть» 0/1).

1 = C2 Инверсный
0 = C2 Не инверсный

bit 4 **C1INV**: Выход компаратора 1 инверсный (если выход нужно «перевернуть» 0/1).

1 = C1 Инверсный
0 = C1 Не инверсный

bit 3 **CIS**: Переключатель входов компаратора (только для режимов включения компараторов 010 и 001).

Если биты CM<2:0> = 001, тогда:

1 = C1 VIN- подключен к RA3
0 = C1 VIN- подключен к RA0

Если биты CM<2:0> = 010, тогда:

1 = C1 VIN- подключен к RA3, C2 VIN- подключен к RA2
0 = C1 VIN- подключен к RA0, C2 VIN- подключен к RA1

bit 2-0 **CM**<2:0>: Режим включения компараторов

Значения битов от 000 до 111, схемы включения указаны в таблице FIGURE 10-1: на странице 62 datasheet.

Исходя из выше описанного, если мы представим в бинарном виде этот регистр, то можно получить следующее:

```
0b00000110
  ||||||||-\
  |||||||--- 110 Включение компараторов с общим опорным питанием на RA2 и
выводом результата на RA3 и RA4
  ||||||---/
  |||||----  Так как биты (2:0) не имеют значения 010 или 001, этот бит
игнорируется
  ||||-----  Выход компаратора 1 не инверсный
  |||------  Выход компаратора 2 не инверсный
  ||-------  Так как выход компаратора 1 не инверсный, то этот бит будет
иметь "1" если C1 VIN+ больше C1 VIN-
  |--------  Так как выход компаратора 2 не инверсный, то этот бит будет
иметь "1" если C2 VIN+ больше C2 VIN-
```

Теперь осталось определить, что же нам удобнее, и перевести число регистра из бинарного в шестнадцатеричный вид для удобства написания. Конфигурация, приведенная в примере выше, в Си вставке будет выглядеть так:

```
trisa = 0x07; //назначаем выводы контроллера RA<2:0> как входы
cmcon = 0x06; //по условиям разработчиков название регистра должно быть
написано маленькими буквами, а название битов большими.
```

Вставляем этот блок Си в самом начале программы сразу после BEGIN, и все. Как видите, всего парой строк можно включить и сразу настроить компаратор под свои нужды.

В данном примере общее опорное напряжение нужно подать на вход контроллера RA2. Вход компаратора 1 нужно подать на RA0, второго на RA1. Результат снимается с выводов RA3 и RA4.

Прямые операции с битами регистров.

Итак, оперировать регистром контроллера байтом мы уже умеем. Как было описано в выше, биты 6 и 7 регистра **CMCON** несут в себе результаты сравнения компараторов 1 и 2. Теперь попробуем вытащить их оттуда непосредственным обращением к ним.

Из Си общение с битами производятся с помощью команд:

- cr_bit(имя_регистра, ИМЯ_БИТА); - очищает определенный бит определенного регистра
- st_bit(имя_регистра, ИМЯ_БИТА); - устанавливает определенный бит определенного регистра
- ts_bit(имя_регистра, ИМЯ_БИТА); - считывает значение определенного бита определенного регистра

примечание: имя регистра маленькими буквами, имя бита - большими.

Создадим в программе Flowcode две переменные типа BYTE и назовем их, например, C1 и C2. А в Си коде присвоим им значение битов 6 и 7 регистра **CMCON**. Вспомним, что перенести значения из Си в программу Flowcode можно, обращаясь к ним в Си по имени с префиксом FCV_.

Внутри основного цикла добавим Си-вставку с текстом:

```
FCV_C1 = ts_bit(cmcon, C1OUT); // взять (ts_bit) значение бита C1OUT из
регистра cmcon и присвоить его переменной FCV_C1
FCV_C2 = ts_bit(cmcon, C2OUT); // взять (ts_bit) значение бита C2OUT из
регистра cmcon и присвоить его переменной FCV_C2
```

Теперь в программе Flowcode в переменных C1 и C2 находятся значения этих битов. Мы можем вывести их в порт, запустить еще какие то макросы и т. д.

Но тут возникает вопрос: цикл повторяется очень быстро, нужно ли нам так часто опрашивать результат работы компараторов? Ответ вполне логичен, конечно, нет. Нам важен сам момент, когда компаратор обнаруживает переход замеряемого значения через опорное напряжение. Зачем же напрягать контроллер ненужными действиями? Создатели контроллеров тоже об этом подумали и реализовали прерывание по изменению выходов компараторов. Теперь контроллер может не отвлекаться на опрос этих битов, прерывание само его «позовет», когда произойдет момент срабатывания компараторов. В программе Flowcode нет штатного прерывания по результату сравнения компараторов, и это не удивительно, работы с компараторами ведь тоже нет, откуда же взяться прерыванию... Зато есть возможность создания своего собственного условия прерывания, к нему и обратимся.

Установим до основного цикла программы компонент Interrupt и в списке Interrupt on: выберем пункт Custom...

Нажмем справа кнопку Properties... и введем настройки.

В самом верхнем поле введем какое-то имя этого прерывания, например, COMP_INT.

Читаем datasheet:

```
The CMIE bit (PIE1<6>) and the PEIE bit
(INTCON<6>) must be set to enable the interrupt. In
addition, the GIE bit must also be set. If any of these
```

```
bits are clear, the interrupt is not enabled, though the
CMIF bit will still be set if an interrupt condition occurs.
```

Краткий перевод:

```
Биты CMIE (регистр PIE1<6>) и бит PEIE (регистр INTCON<6>)
должны быть установлены, так же должен быть установлен
бит GIE (регистр INTCON<7>), чтобы разрешить прерывание
компаратором. Бит CMIF должен быть очищен. (Это регистр PIR1<6>).
```

Исходя из вышесказанного, в следующем поле введем разрешения на прерывание и данные инициализации.

```
cr_bit(pir1, CMIF);     // очистка бита CMIF регистра pir1 (флаг прерывания в
ноль)
st_bit(pie1, CMIE);     // установка бита CMIE регистра pie1 (разрешение
прерывания компаратором)
st_bit(intcon, PEIE);   // установка бита PEIE регистра intcon (разрешение
прерывания периферией)
st_bit(intcon, GIE);    // установка бита GIE регистра intcon (разрешение
глобальных прерываний)
```

Второе поле пока затененное, его заполним позже.

В третьем поле впишем код условия, который будет обнулять флаг прерывания и запускать макрос:

```
if (pir1 & (1 << CMIF))        //если срабатывает флаг прерывания
{
FCM_%n();                      // запустить макрос
cr_bit(pir1, CMIF);            // очистить бит флага прерывания (в ноль)
}
```

Жмем «ОК» и возвращаемся в окно настроек прерываний. Жмем кнопку «Create New Macro...», создаем новый макрос, назовем его, например, «comparator», жмем «ОК» и возвращаемся опять в окно настроек прерываний. Опять жмем «ОК» и сохраняем прерывание. Снова заходим в настройки прерываний двойным кликом по иконке Enable COMP_INT в главном окне программы. Ставим точку напротив надписи Disable Interrupt. Жмем кнопку Properties... и введем настройки отключения прерывания. Теперь нам доступно второе поле ввода текста, а первое и третье затенено. Вводим текст отключения прерывания во второе поле:

```
cr_bit(pie1, CMIE);       //очистить бит CMIE регистра pie1 (запрещаем
прерывание компаратора)
```

Жмем кнопку «ОК», переставляем точку в положение Enable Interrupt и опять жмем «ОК». Все, прерывание запрограммировано.

В макросе comparator, который мы создали на предыдущем шаге, установите Си код, который в начале статьи мы вставили в основной цикл. (Из цикла, естественно, его нужно удалить).

```
FCV_C1 = ts_bit(cmcon, C1OUT); // взять (ts_bit) значение бита C1OUT из
регистра cmcon и присвоить его переменной FCV_C1
FCV_C2 = ts_bit(cmcon, C2OUT); // взять (ts_bit) значение бита C2OUT из
регистра cmcon и присвоить его переменной FCV_C2
```

Теперь данные в переменные C1 и C2 будут записываться только тогда, когда они реально изменятся. Для эксперимента, сразу за Си-вставкой можно поставить два вывода в порт «B» Output и назначить в качестве Переменной или Значения переменные C1 и C2. При симуляции в программе Proteus или в контроллере эти выводы будут менять свои значения в зависимости от состояния выхода компараторов.

Установка внутреннего опорного напряжения компараторов

Модуль VOLTAGE REFERENCE состоит из 16-ти резисторов, соединенных последовательно. Резисторы сегментированы в два диапазона, верхний и нижний, и имеют функцию отключения питания для экономии энергии, когда они не используются. Регистр VRCON управляет включением этих резисторов.

Регистр VRCON имеет следующие биты конфигурации:

```
bit 7 VREN: Бит включения внутреннего опорного напряжения
 1 = VREF Опорное напряжение включено
 0 = VREF Опорное напряжение выключено, никакого потребления тока

bit 6 VROE: Бит вывода опорного напряжения
 1 = VREF К выводу RA2
 0 = VREF Отключен от вывода RA2

bit 5 VRR: Бит диапазона напряжений
 1 = Нижний диапазон
 0 = Верхний диапазон

bit 4 Не используется, читается как '0'

bit 3-0 VR<3:0>: Биты конфигурации напряжения 0 ? VR <3:0> ? 15
 Если VRR = 1: VREF = (VR<3:0>/ 24) * VDD
 Если VRR = 0: VREF = 1/4 * VDD + (VR<3:0>/ 32) * VDD
```

Модуль VOLTAGE REFERENCE может вывести 16 различных уровней напряжения для каждого диапазона.

Ниже приведены две таблицы напряжений при питании контроллера 5 вольт.

```
Нижний диапазон Vref/24                 Верхний диапазон Vref/32
        Vdd     5                               Vdd     5
0000            0.00000                 0000            1.25000
0001            0.20833                 0001            1.40625
0010            0.41667                 0010            1.56250
0011            0.62500                 0011            1.71875
0100            0.83333                 0100            1.87500
0101            1.04167                 0101            2.03125
0110            1.25000                 0110            2.18750
0111            1.45833                 0111            2.34375
1000            1.66667                 1000            2.50000
1001            1.87500                 1001            2.65625
1010            2.08333                 1010            2.81250
1011            2.29167                 1011            2.96875
1100            2.50000                 1100            3.12500
1101            2.70833                 1101            3.28125
1110            2.91667                 1110            3.43750
1111            3.12500                 1111            3.59375
```

Пример конфигурации регистра VRCON со следующими условиями:

```
0b11100110
  ||||||||-\
  |||||||--- Опорное напряжение
  ||||||---- 0110 равное 1,250 вольт
  |||||----/
  ||||----- Не используется
  |||------ Нижний диапазон
  ||------- Питание подключено к выводу RA2
  |-------- Опорное напряжение включено
```

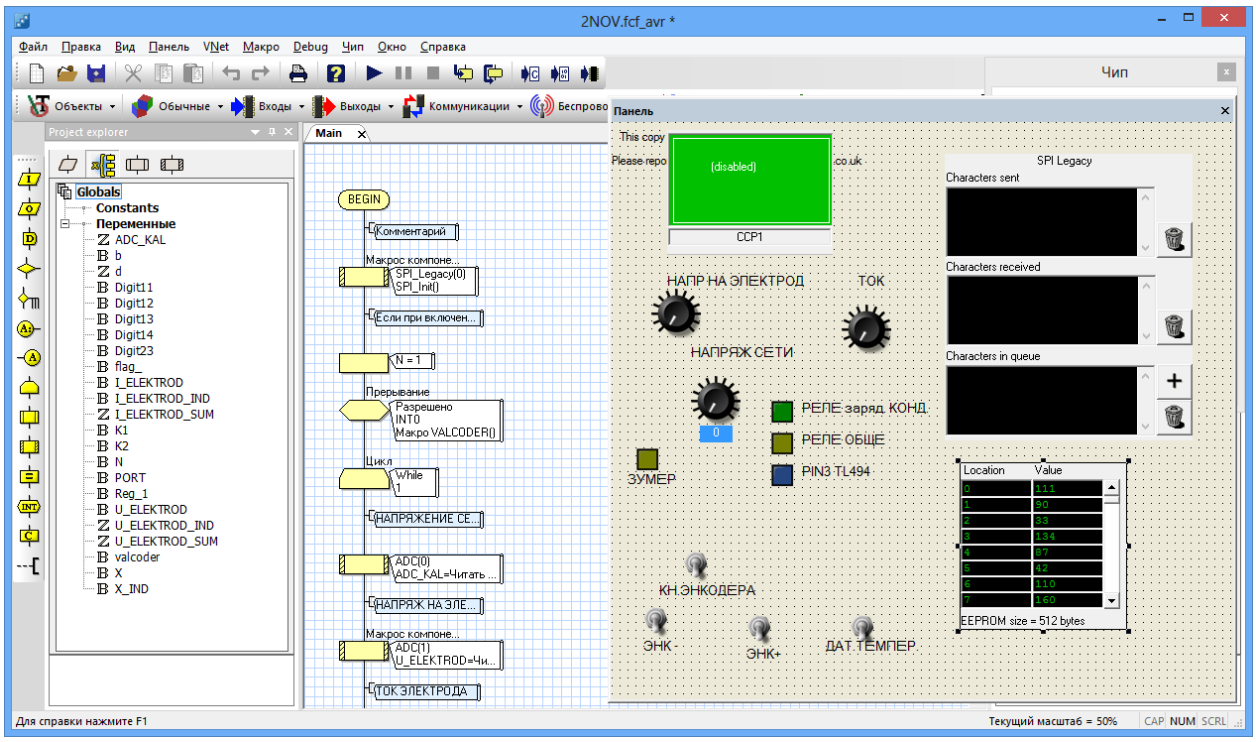
Если перевести бинарный код 0b11100110 в шестнадцатеричный, то получится 0xE6. Теперь в Си-вставку после кода cmcon = 0x06; можно добавить строку:

```
vrcon = 0xE6; //внутреннее опорное напряжение 1.250 вольт
```

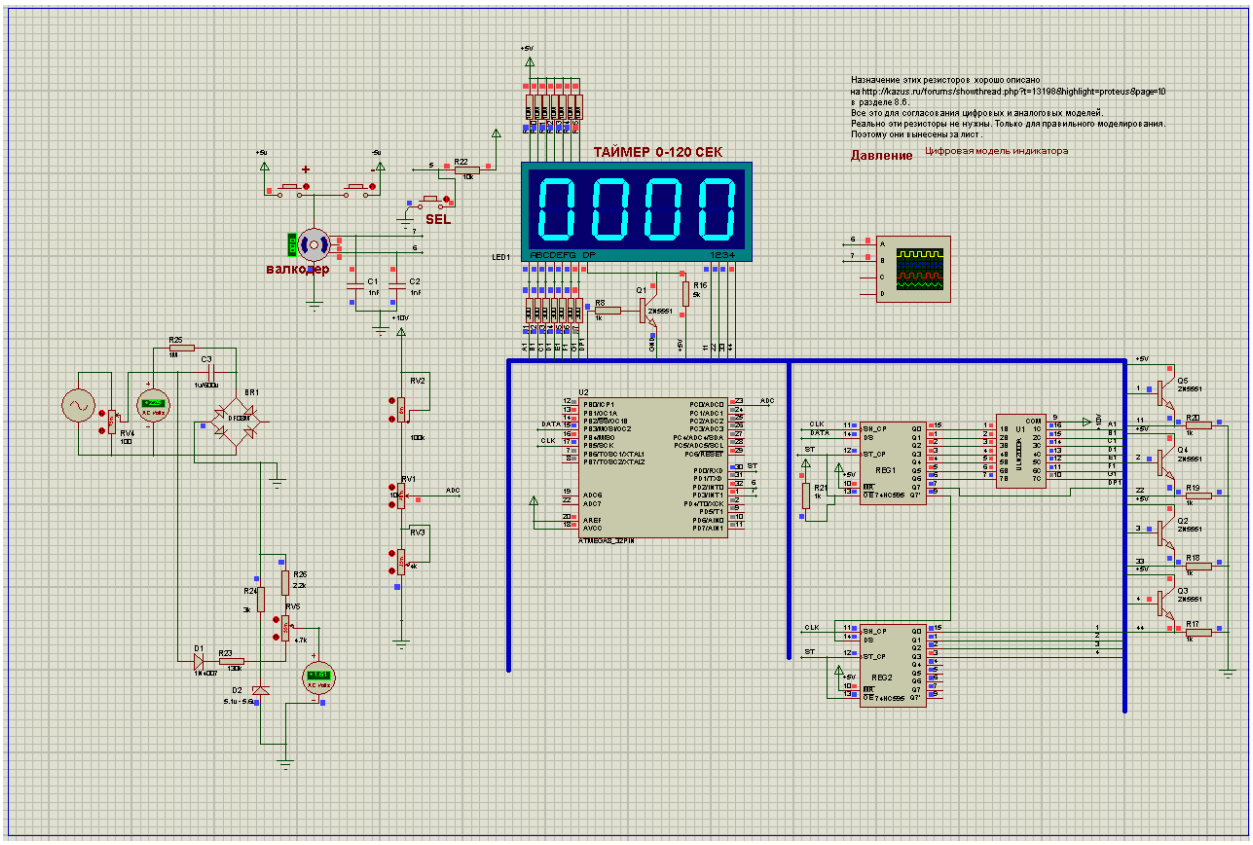
Flowcode и Proteus на форуме

Просматривая форум, можно заметить (это есть и в примерах простых программ), что очень часто окончательная проверка работы программы с внешними компонентами выполняется в программе Proteus. Отладчик Flowcode, как любой среды разработки, позволяет отлаживать программу, но работа с рядом компонентов, например, как проводные и беспроводные коммуникации, зачастую ориентирована на модули, выпускаемые производителями программы. А соединить микроконтроллер с источником нужных для проверки сигналов, добавить в схему источник питания и т.п., представляется невозможным. Поэтому использование второй программы для окончательной проверки оказывается удобной альтернативой макетной плате. Хотя окончательно расставить все точки над «и» может только опытный образец и работа устройства в реальных условиях.

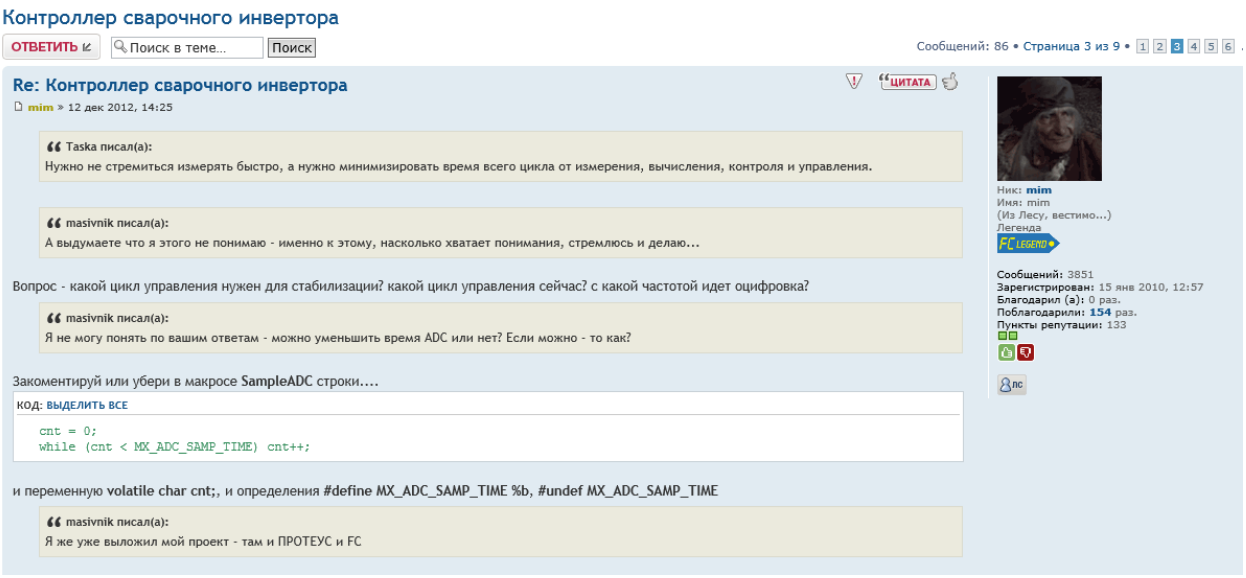
Вот один из примеров, приведённых участником форума:



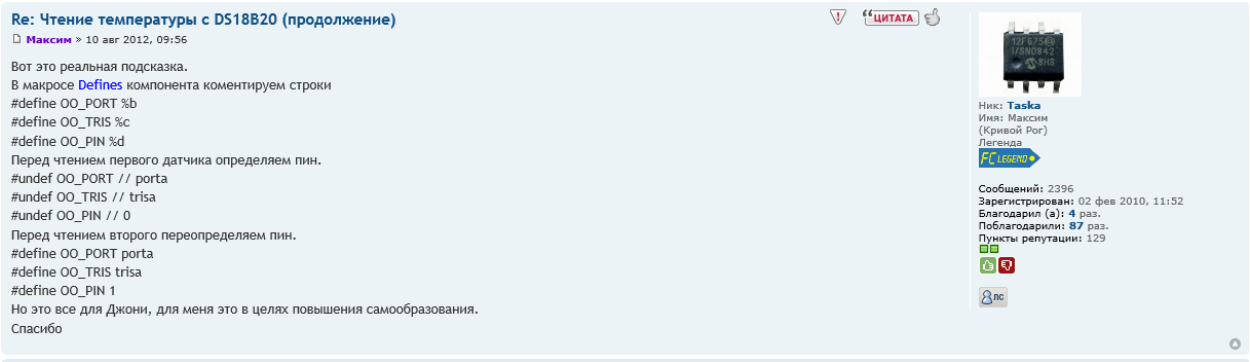
Программа устройства создана в Flowcode, но проверка в Proteus выявляет что-то, с чем предстоит разбираться:



Обсуждение на форуме, надо полагать, приведёт к желаемому результату.

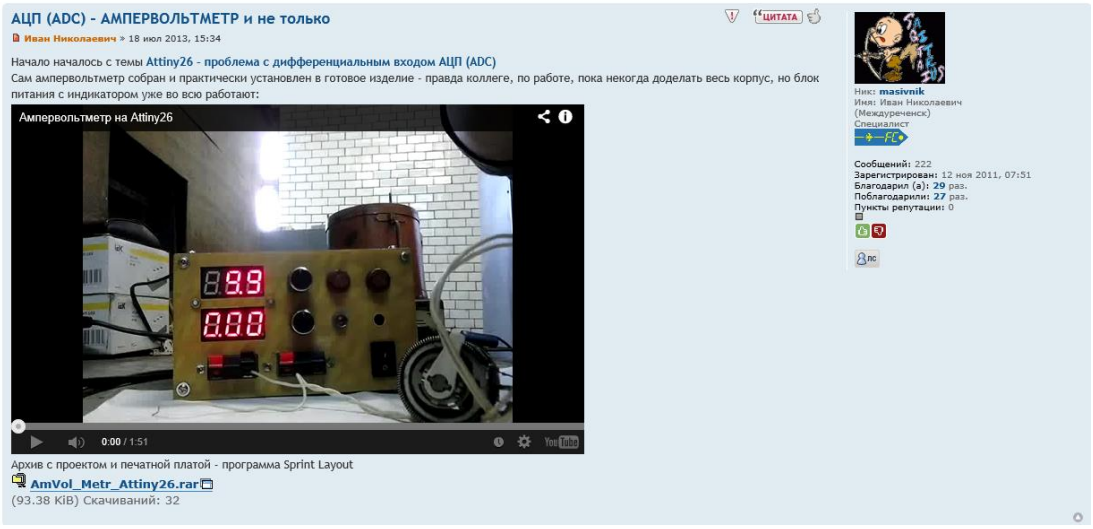


Многие из тех, кто начинал знакомство ещё с третьей версией программы Flowcode вполне освоились и с языком Си:



Flowcode.info – готовые устройства форумчан

На форуме можно посмотреть видео с работающими устройствами…

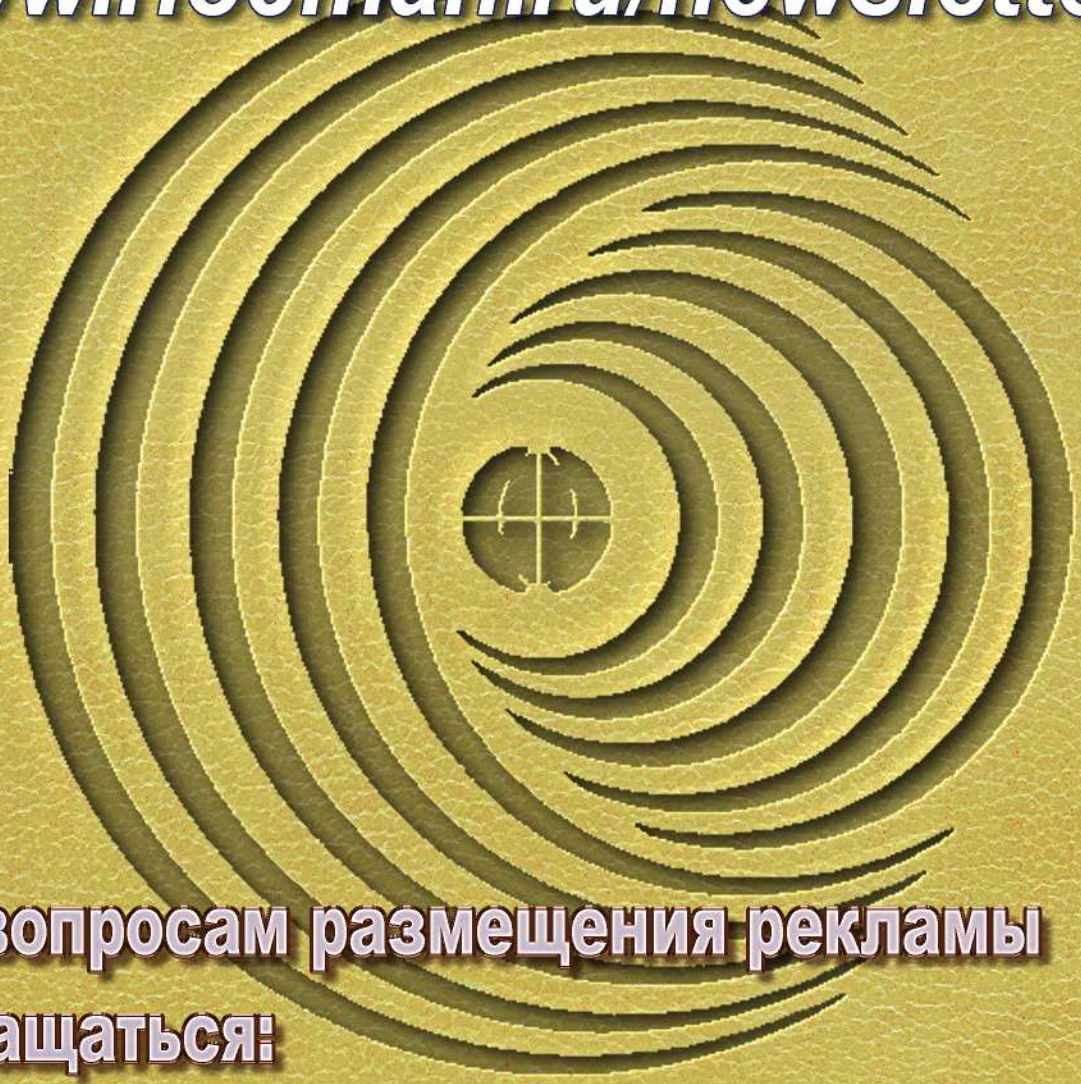


…и прочитать все пояснения к проекту, включая теорию.

Вот ещё один готовый проект:

И готовых решений много.

Уведомление о новых
выпусках "Радиоежегодника"
на основной ленте новостей
www.rlocman.ru
и выпусках почтовой рассылки
www.rlocman.ru/newsletter



По вопросам размещения рекламы
обращаться:
rlocman@rlocman.ru
radioyearbook@gmail.com