ГОСУДАРСТВЕННЫЙ КОМИТЕТ СВЯЗИ, ИНФОРМАТИЗАЦИИ И ТЕЛЕКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ФАКУЛЬТЕТ ТЕЛЕКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

МИКРОКОНТРОЛЛЕР В ПРОГРАММЕ FLOWCODE

Методические указания по выполнению лабораторных работ предмету
Цифровая техника и микропроцессоры

Ташкент 2014

Авторы: Х. Ю. Абасханова, С. В. Ваулина

«Микроконтроллер в программе FlowCode»/ ТУИТ. Ташкент, 2014.

Методическое пособие «Микроконтроллер в программе FlowCode» состоит в изучении принципов программирования микроконтроллеров AVR и PIC с помощью программы FlowCode (Windows), а также даёт возможность создавать сложные электронные системы в считанные минуты, практически не имея опыта программирования.

В этом методическом пособии будут приведены примеры создания различных программ, с увеличивающимся уровнем сложности, которые позволяют просматривать и комментировать С и Ассемблер код, генерируемый из блок-схем, с последующей возможностью рисовать переднюю панель для разрабатываемого прибора, которая в свою очередь является электронным симулятором реально существующих электронных систем.

Методическое пособие рассмотрено и одобрено на научно-методическом Совете факультета «Телекоммуникационных технологий» (протокол № 8 от 29.04.2014 г.)

Рецензенты:
Доцент кафедры сети и системы передачи данных — Джаббаров Ш.Ю.

Корректор — Хамдам-Заде Л.Х

Ташкентский Университет Информационных Технологий, 2014

ВВЕДЕНИЕ

ПРЕИМУЩЕСТВА ИСПОЛЬЗОВАНИЯ СРЕДЫ РАЗРАБОТКИ FLOWCODE ПРИ СОЗДАНИИ ПРОГРАММ ДЛЯ МИКРОКОНТРОЛЛЕРОВ

В современном мире скорость разработки программного обеспечения является одним из главных факторов успешности продукта на рынке. С появлением и обширным внедрением в повседневную жизнь устройств на базе микроконтроллеров (**MCU**) появилась проблема ускорения процесса написания ПО для таких систем. Одним из способов решения данной задачи является применение сред визуального программирования. Данный подход позволяет даже начинающему программисту создавать большие и сложные программы для МК, затрачивая при этом значительно меньше времени.

Ярким представителем подобных средств разработки является программный комплекс **FlowCode** от компании **Matrix Multimedia**.

FlowCode - это среда разработки с понятным графическим интерфейсом, использующая язык программирования на основе объектов и блок-схем. Реализация технологии **Drag and drop** позволяет с лёгкостью создавать программы простым перетаскиванием необходимых иконок, а блок-схемный подход к написанию программы повышает её наглядность и структурированность.

Данная среда программирования позволяет создавать код для микроконтроллеров **AVR, ARM и PIC,** которые являются на сегодня самыми распространёнными. В ней имеются готовые библиотеки программного кода для различных периферийных модулей, таких как **USART, SPI, ADC**, а также различных компонентов, которые, как правило, входят в состав устройств на основе микропроцессоров (светодиодные индикаторы, LCD шаговые двигатели и т.д.). Данная возможность позволяет представить данные блоки как отдельные элементы, имеющие соответствующие входы и выходы, что существенно сокращает время на реализацию кода. Достаточно просто добавить обращение к нужному модулю из программы.

Ещё одной полезной функцией данной программы является открытая архитектура, которая даёт возможность получить листинг на языке Ассемблер и С. При необходимости его можно отредактировать, что особенно актуально в приложениях, где требуется максимальная производительность. Также можно проверить код на наличие логических ошибок с помощью встроенного

отладчика, который позволяет визуализировать такие процессы как вывод информации на ЖК дисплей, вращение шагового двигателя и т.д.

И так подведём итог:

Flowcode является одним из самых передовых графических языков программирования для микроконтроллеров в мире. Большое преимущество **Flowcode** в том, что практически не имея опыта программирования можно создавать сложные электронные системы в считанные минуты. **Flowcode** доступен более, чем на двадцати языках и в настоящее время поддерживает серии микроконтроллеров **AVR**.

Основные характеристики:

- Простой в использовании интерфейс, просто перетащите значки на экране;
- Обширная подпрограмма высокого уровня компонентов;
- Открытая архитектура, позволяющая просматривать и прокомментировать С и Ассемблер код, генерируемый из блок-схем;
- Полностью поддерживаются спектр материалов для обучения и развития встраиваемых систем.

Преимущества:

- Позволяет тем, кто имеет мало опыта - быстро и без ошибок создавать электронные системы;
- Быстрая разработка простых и сложных встраиваемых систем.

Возможности:

- Поддерживаются интерфейсы **I2C, SPI, RS232, Bluetooth, Zigbee, IrDA, CAN, LIN, TCP/IP, Webserver, USB, RFID, GPS**;
- Имеются компоненты для **LED**, кнопок, выключателей, клавиатур, **LCD, Graphical colour LCD, Graphical mono LCD**, сенсоров, 7-сегментных дисплеев, внутренней **EEPROM**;
- Можно рисовать переднюю панель для разрабатываемого прибора.

Вывод:

Вы можете иметь лишь поверхностные навыки в программировании, которые сводятся к составлению нужного алгоритма и, соответственно, блок-схемы. **Flowcode** генерирует С код, а также компилирует его в **.hex** файл, который можно сразу же прошить в контроллер или, к примеру, смоделировать в **Proteus'e**.

И так подведём итог:

Flowcode является одним из самых передовых графических языков программирования для микроконтроллеров в мире. Большое преимущество **Flowcode** в том, что практически не имея опыта программирования можно создавать сложные электронные системы в считанные минуты. **Flowcode** доступен более, чем на двадцати языках и в настоящее время поддерживает серии микроконтроллеров **AVR**.

Основные характеристики:

- Простой в использовании интерфейс, просто перетащите значки на экране;
- Обширная подпрограмма высокого уровня компонентов;
- Открытая архитектура, позволяющая просматривать и прокомментировать C и Ассемблер код, генерируемый из блок-схем;
- Полностью поддерживаются спектр материалов для обучения и развития встраиваемых систем.

Преимущества:

- Позволяет тем, кто имеет мало опыта - быстро и без ошибок создавать электронные системы;
- Быстрая разработка простых и сложных встраиваемых систем.

Возможности:

- Поддерживаются интерфейсы **I2C, SPI, RS232, Bluetooth, Zigbee, IrDA, CAN, LIN, TCP/IP, Webserver, USB, RFID, GPS;**
- Имеются компоненты для **LED**, кнопок, выключателей, клавиатур, **LCD, Graphical colour LCD, Graphical mono LCD**, сенсоров, 7-сегментных дисплеев, внутренней **EEPROM;**
- Можно рисовать переднюю панель для разрабатываемого прибора.

Вывод:

Мы можем иметь лишь поверхностные навыки в программировании, которые сводятся к составлению нужного алгоритма и, соответственно, блок-схемы. **Flowcode** генерирует C код, а также компилирует его в **.hex** файл, который можно сразу же прошить в контроллер или, к примеру, смоделировать в **Proteus'e**.

ЛАБОРАТОРНАЯ РАБОТА №1.

ЗНАКОМСТВО С ИНТЕРФЕЙСОМ ПРОГРАММЫ FLOWCODE. ЗАПИСЬ И ОТЛАДКА ПРОСТЫХ ПРОГРАММ.

Цель работы: Изучение инструментальных панелей команд и добавочных элементов, а также ознакомление с созданием простых программ и их отладка.

Основные теоретические сведения.

Программа FlowCode как самый простой и легкий путь к применению микроконтроллеров в своей практике.

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 на конкретный тип 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 при программировании могут 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. Позже, когда вы наберетесь опыта, вам не сложно будет сменить PIC на AVR или любой другой. 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 созданию кода и его отладке в одной программе, а выполнить программирование микросхемы в другой.

Программа FlowCode существует в двух версиях — для контроллеров AVR и PIC. Удобно, что можно импортировать решения из одной версии в другую. Для начинающих удобна простота отладки, поскольку есть много внешних элементов устройств, обычно используемых вместе с микроконтроллерами.

Первая левая инструментальная панель — это панель команд.

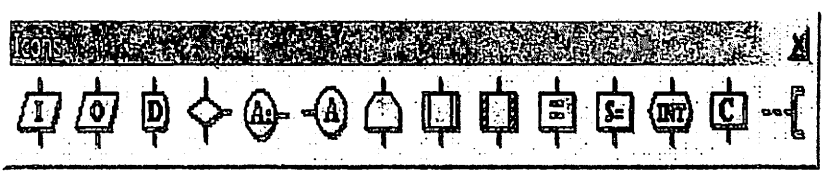


Рис. 1.1. Инструментальная панель команд.

Перечень представленных команд (слева-направо на рисунке, сверху-вниз при запуске):

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).

Вторая инструментальная панель для добавочных внешних элементов.

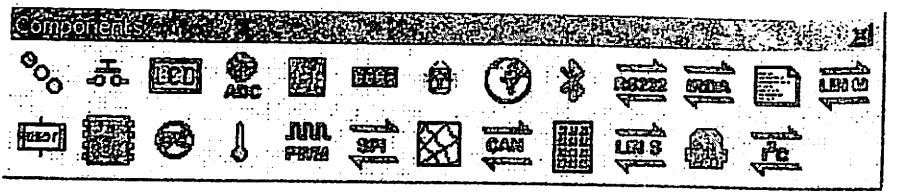


Рис. 1.2. Инструментальная панель добавочных элементов.

Компоненты (слева-направо):

LEDs (светодиоды), ***Switches*** (переключатели), ***LCDDisplay*** (жидкокристаллический дисплей), ***ADC*** (АЦП, если есть порт АЦП), ***LED7Seg1*** (семисегментный индикатор), **LED7Seg4** (блок из 4х семисегментных индикаторов), ***Buggy*** (компонент игрушки), далее несколько стандартных интерфейсов ***TCP_IP, Bluetooth, RS232, IrDA, AddDefines*** (добавить определения), ***LinMaster*** (ведущий в локальной сети), ***Custom*** (заказной компонент), ***EEPROM*** (перепрограммируемая память), ***Alarm*** (охранное устройство), ***Thermometer*** (термометр), ***PWM*** (широтно-импульсный модулятор), ***SPI*** (последовательный внешний интерфейс), ***WebServer*** (web-сервер), ***CAN*** (сеть абонентского доступа), ***KeyPad*** (клавиатура), ***LinSlave*** (ведомый в локальной сети), ***FormulaFlowCode*** (компонент игры), ***I2C*** (шина связи между ИС).

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.

Перед началом создания программы, выбираем, будем ли мы работать с новой программой или с уже существующей, а также выбираем тип используемого микроконтроллера:

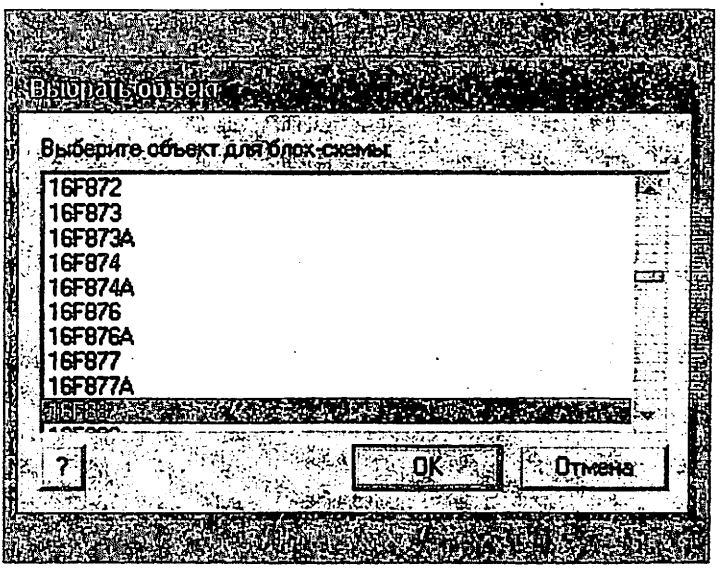


Рис. 1.3. Выбор программы и микроконтроллера.

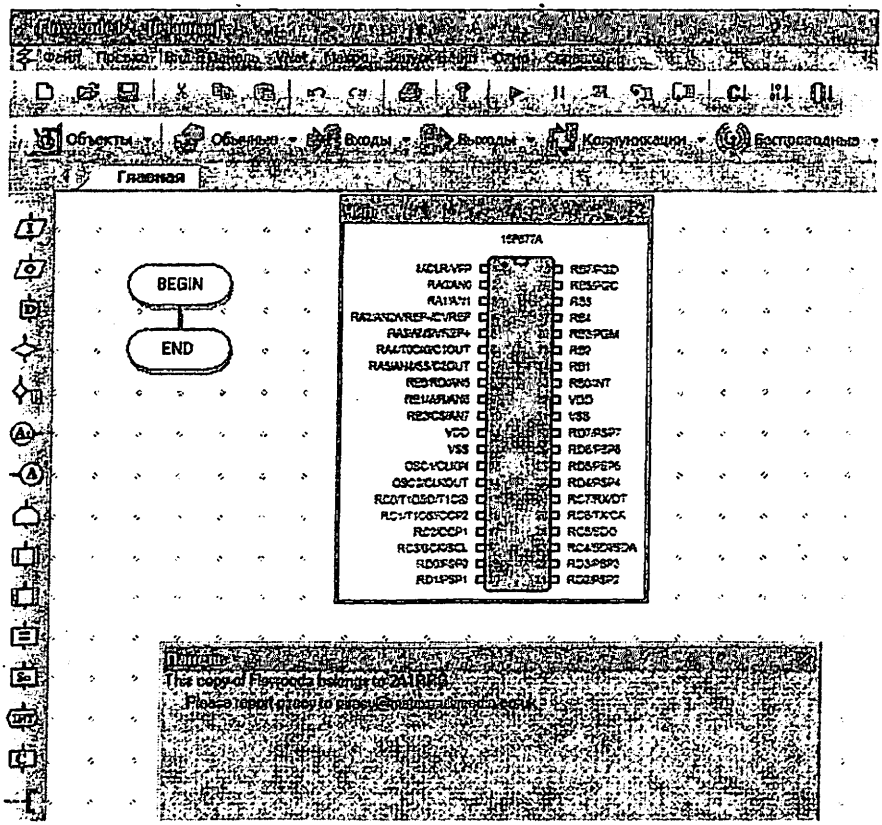


Рис. 1.4. Первый запуск программы FlowCode.

Программа предназначена для операционной среды **Windows**, но можно использовать её в **Linux**, поэтому могут иметь место незначительные отличия во внешнем виде и поведении программы. Так 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.

Заставим микроконтроллер управлять светодиодом, который на макетной плате припаяем к выводу, скажем, нулевому порта А. Пусть мигает раз в секунду. На правой инструментальной панели есть кнопочка с буквой «О». Это от слова **output**-выход (если навести курсор мышки на эту кнопочку, то высвечивается подсказка **Output**). Цепляем этот выход (нажимаем левую клавишу мышки, когда курсор указывает на иконку, и, не отпуская клавиши, перемещаем курсор в рабочее поле схемы) и тащим его к линии между овалами **Begin**-начало и **End**-конец. При этом курсор выглядит как стилизованная иконка, а слева появляется стрелочка-указатель.

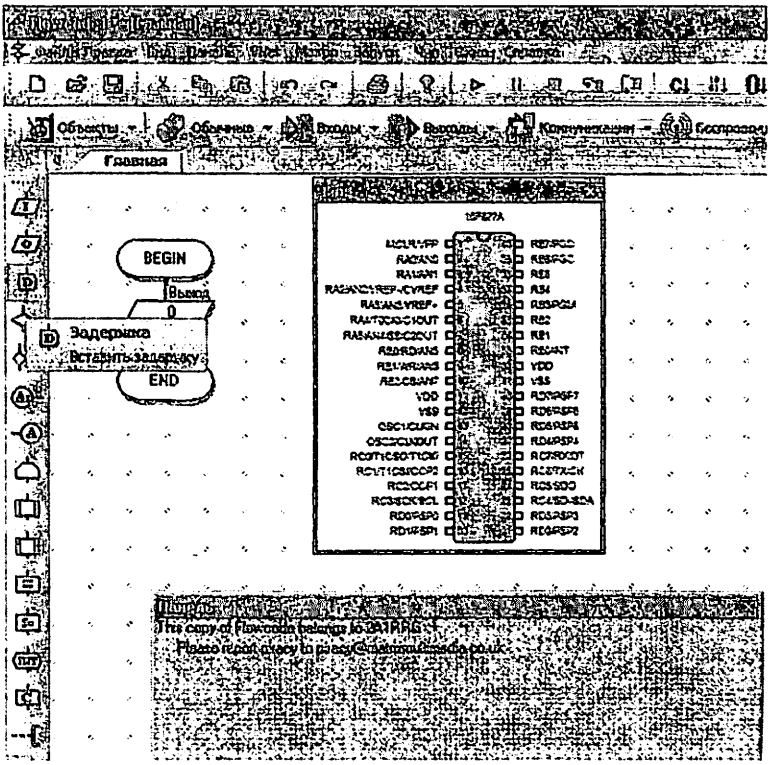


Рис. 1.5. Добавление элемента программы к диаграмме.

После добавления элемента **Output** диаграмма принимает следующий вид:

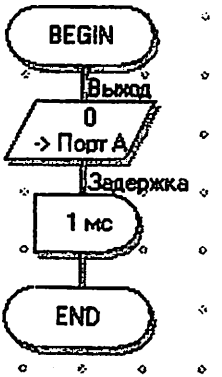


Рис. 1.6. Начальный вид диаграммы первой программы.

Нуль в заголовке состояния порта А подразумевает, что все выводы находятся в состоянии логического нуля (в низком уровне напряжения) или что они будут установлены в 0.

Мигать чем-то — это менять состояние, но чтобы мигание имело место, понадобится пауза. Такой элемент (иконка на левой инструментальной панели с литерой «D») **Delay**-задержка есть. Перетаскиваем его и вставляем ниже первого выхода **Output**. Чтобы изменить время паузы 1 мС нужно произвести двойной щелчок левой клавишей мышки по этому элементу, после чего на рабочем поле открывается диалоговое окно свойств элемента.

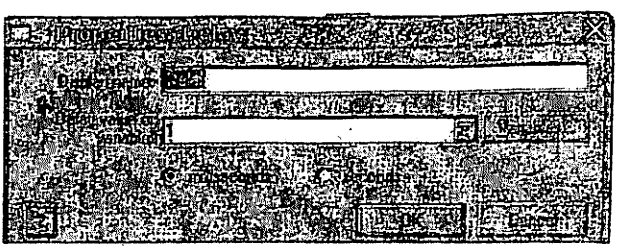


Рис. 1.7 Диалоговое окно свойств элемента Delay.

Теперь достаточно выбрать опцию **seconds**, чтобы превратить миллисекунду в секунду. Следом за задержкой в 1 секунду добавляем еще один **Output**, как и в первый раз, но теперь, двойным щелчком левой клавиши мышки по нему на рабочем поле схемы, открываем диалоговое окно его свойств:

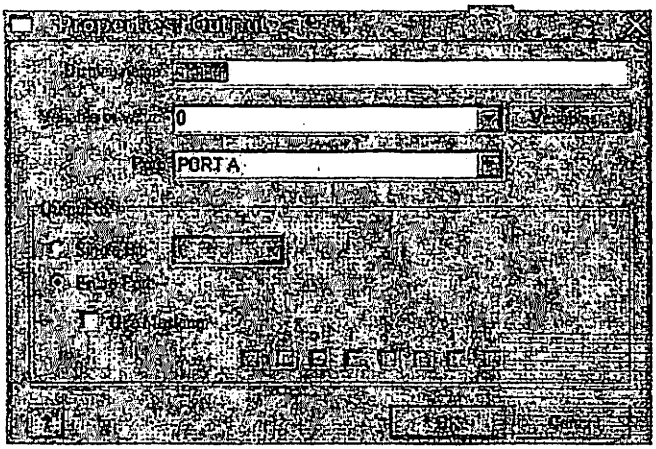


Рис. 1.8 Изменение свойств выхода порта А в диалоговом окне.

Заменим 0 в окне **Variable or Value:** единицей и добавим еще одну паузу. Но нам необходимо, чтобы светодиод мигал непрерывно. Такое непрерывное выполнение фрагмента программы 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 программой, удерживая клавишу, отрисовываем прямоугольник, включающий всю 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.

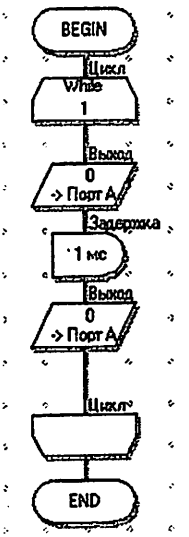


Рис. 1.9. Первая программа в FlowCode.

Вы написали первую программу, и пора бы проверить, работает ли она. Программа FlowCode имеет отладочные средства. 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 двоичном виде.

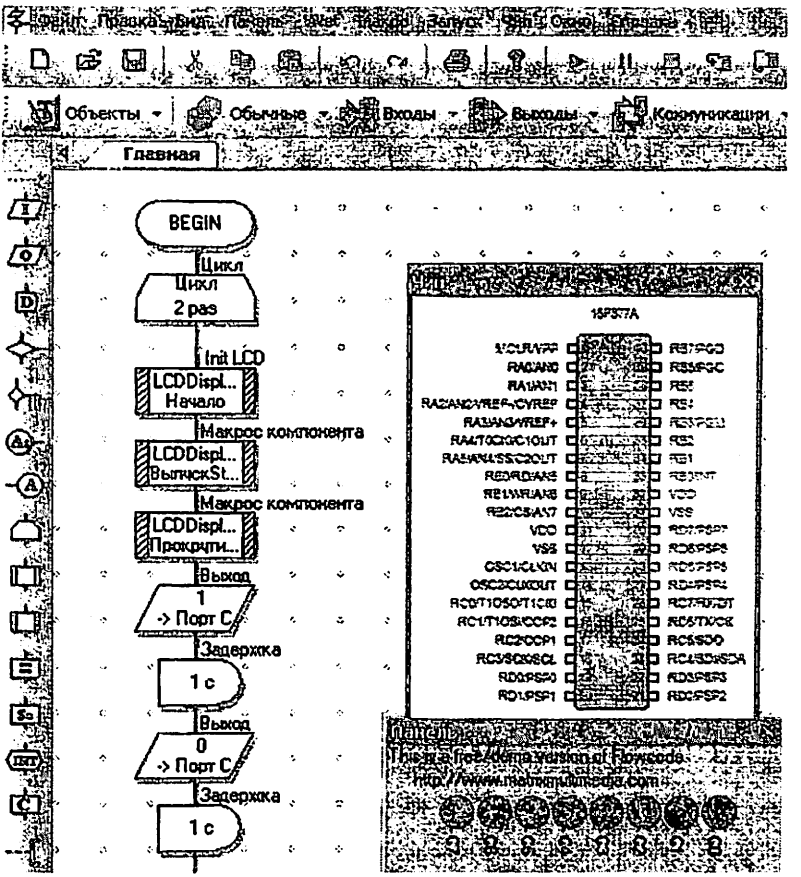


Рис. 1.10. Первый запуск первой программы.

Варианты заданий для лабораторной работы № 1:

1) Отправка значения (5) в порт А.
2) Зажигание светодиодов по одному в 4-битах порта В.
3) Использование переменной и отправка значения переменной в порт.
4) Создание игрушки светофор.
5) Создание игрушки «ёлочная гирлянда».
6) Написать номер группы с помощью мигающих светодиодов.

Форма и содержание отчета:

1) Титульный лист;
2) Формулировка варианта задания;
3) Блок-схема алгоритма решения задачи;
4) Визуальный отчет и результаты выполнения программы.

Контрольные вопросы:

1) Дать определение, что такое программа Flowcode?
2) Для каких типов микроконтроллеров она используется?
3) Основные характеристики программы и её преимущества?
4) Какие возможности имеет данная программа?

ЛАБОРАТОРНАЯ РАБОТА №2.

СОЗДАНИЕ РАЗВЕТВЛЁНЫХ ПРОГРАММ.

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 выходе), по результатам опроса входов

программа проходит по одной или другой ветке программы. Например, такой фрагмент программы:

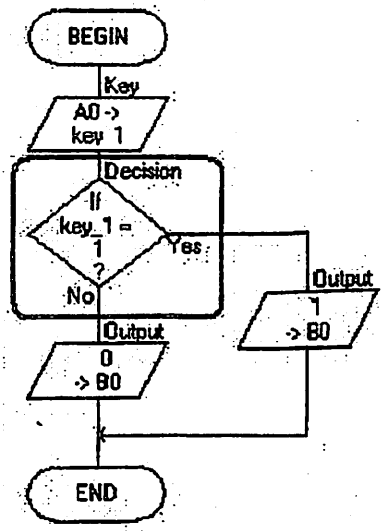


Рис. 2.1. Фрагмент работы с компонентом программы **Decision.**

В этом фрагменте нулевой вывод порта А 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 случае нулевой вывод порта В (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.

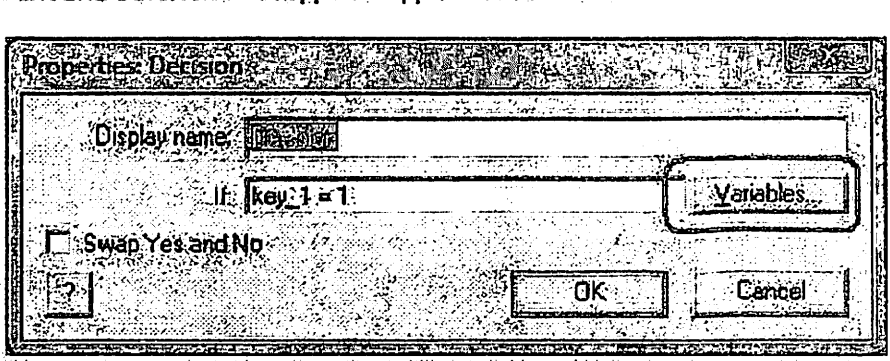


Рис.2.2. Диалоговое окно компонента Decision.

С помощью кнопки Variables, отмеченной на рисунке, выбираем ранее созданную переменную, продолжив запись условия ветвления. Это условие может быть простым, как выше, но может быть и сложным, использующим, например, логические операции и переменные.

В данной лабораторной работе рассмотрим программу по созданию двоичного счетчика, инкрементируемого по нажатию кнопки. При достижении 8 счетчик обнуляется.

Для начала создадим элементы, которые нам нужны для работы: это кнопка и 4 светодиода.

Для создания кнопки щелкаем Входы, выбираем SWITCH.

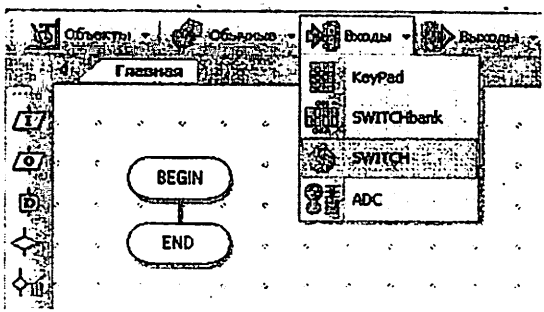


Рис. 2.3. Создание элемента SWITCH.

В нижней панели появится тумблер, что не страшно. Если щелкнуть правой кнопкой по тумблеру, то можно попасть в контекстное меню и выбрать Расширенные свойства и там настроить как нужно.

Например, так:

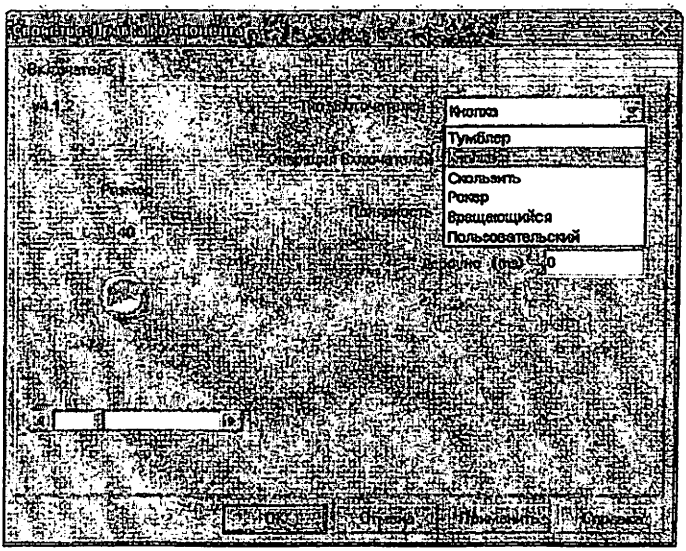


Рис. 2.4. Настройка элемента SWITCH.

Также, в контекстном меню кнопки выбираем Соединения и приделываем кнопку к порту С и выводу 0.

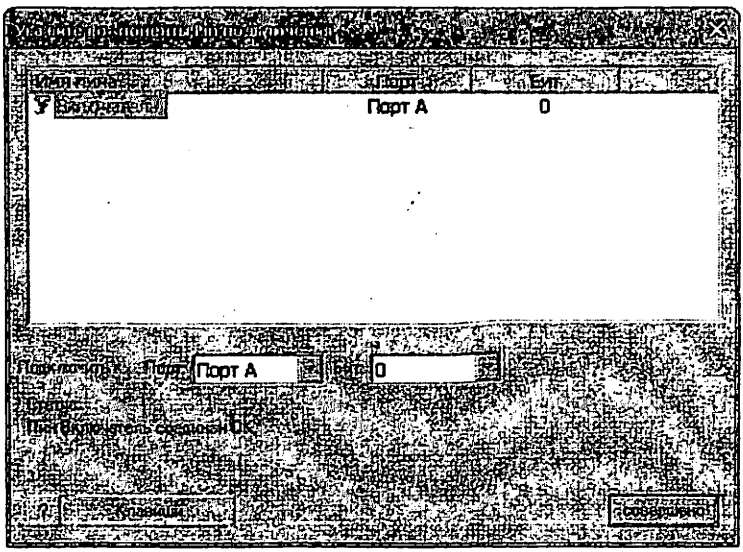


Рис. 2.5. Подключение элемента SWITCH

Аналогично подключаем каждый светодиод, но уже к портам B0, B1, B2, B3. Светодиод ищем в пункте Выходы.

Окончательный результат выглядит аналогично этой картинке:

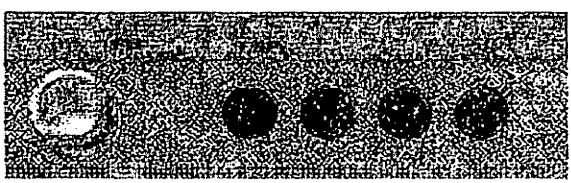


Рис. 2.6. Панель с четырьмя светодиодами и SWITCH

Также необходимо создать всего две переменные: **count** и **button**. Обе типа **BYTE**.

Первая - это сам счетчик, она будет меняться от 0 до 8 в процессе счета.

Вторая - может быть 0 или 1, она показывает, нажата ли кнопка.

Создаются следующим образом:

Правка - Переменные, в открывшемся окне создаем новую переменную.

Далее собираем следующую блок-схему:

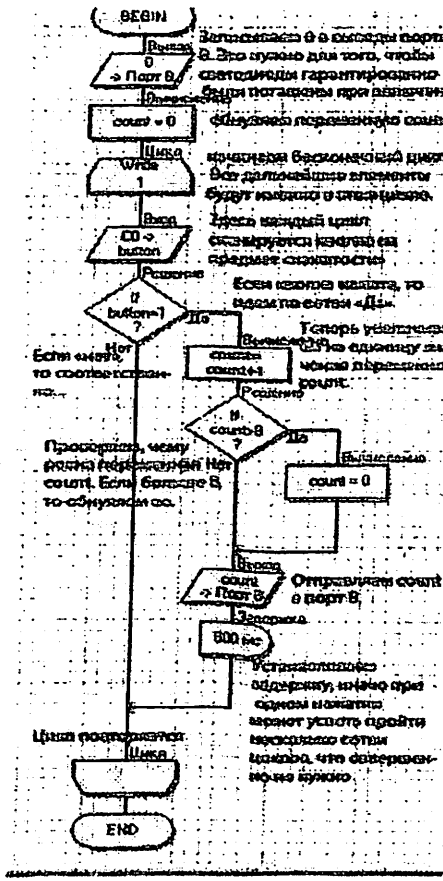


Рис. 2.7. Блок-схема программы 4-х разрядного двоичного счетчика до 8, инкрементируемого по нажатию кнопки

Все используемые элементы берутся из левой панели программы. После сборки запускаем. Светодиоды по нажатию кнопки должны выдавать двоичное 4-битное число. При достижении 8 счетчик обнуляется, все светодиоды гаснут.

Варианты заданий для лабораторной работы № 2:

1) С помощью светодиодов создать 8-разрядный двоичный счетчик прямого счета.
2) С помощью светодиодов создать 8-разрядный двоичный счетчик обратного счета.
3) С помощью светодиодов создать 8-разрядный двоичный счетчик с контролем переполнения.

Форма и содержание отчета:

1) Титульный лист;
2) Формулировка варианта задания;
3) Блок-схема алгоритма решения задачи;
4) Визуальный отчет и результаты выполнения программы.

Контрольные вопросы:

1) Дать определение, программного компонента **Decision**, в каких случаях он используется?
2) Приведите свои примеры разветвлённых программ.

ЛАБОРАТОРНАЯ РАБОТА №3.

СОЗДАНИЕ ЦИКЛИЧЕСКИХ ПРОГРАММ И ПОДПРОГРАММ.

Рассмотрим новый важный компонент программы - **Loop** (цикл).

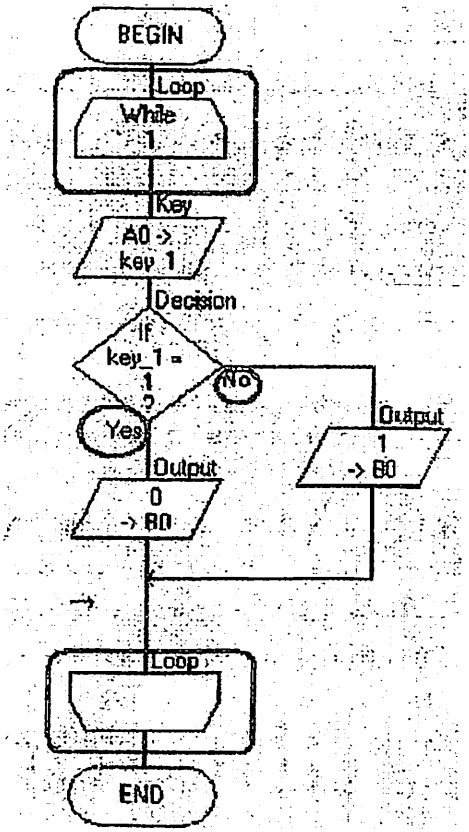


Рис 3.1. Фрагмент программы с циклом

Цикл «окаймляет» программу, начиная с условия выхода из цикла. В данном случае **While 1**, поскольку никаких условий выхода из цикла не задано. 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 перестаёт реагировать на команды программиста или реагирует только на часть команд. С другой стороны, если это сделано осознанно, так работает почти каждый контроллер: он «крутится» в бесконечном цикле.
Свойства цикла, как и других программных компонентов, задаются в диалоговом окне.

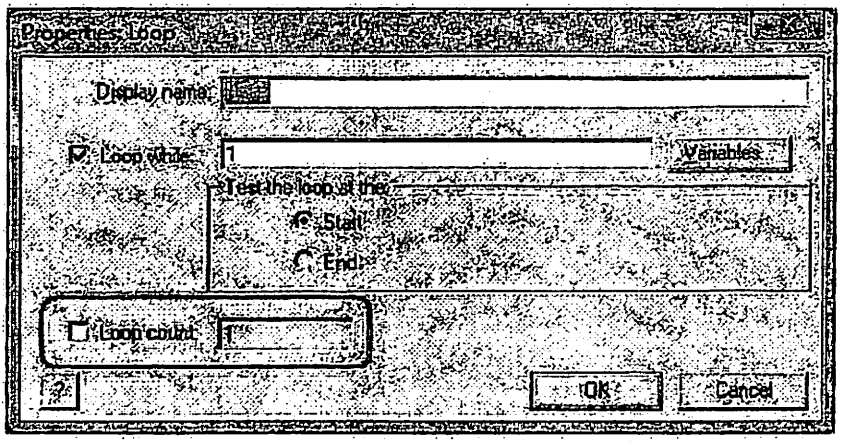


Рис. 3.2. Диалоговое окно компонента Loop

С помощью этого диалога можно задать условие выхода из цикла, используя кнопку **Variables** для выбора и создания переменной. Как правило, эта переменная должна меняться внутри цикла. Если в вышеприведенном фрагменте использовать переменную, связанную с кнопкой входа порта A, 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.

После добавления компонента **Macro** в программу мы можем перенести в него ранее созданную процедуру. Для этого дважды щелкнем по компоненту **Macro**, открывая диалоговое окно.

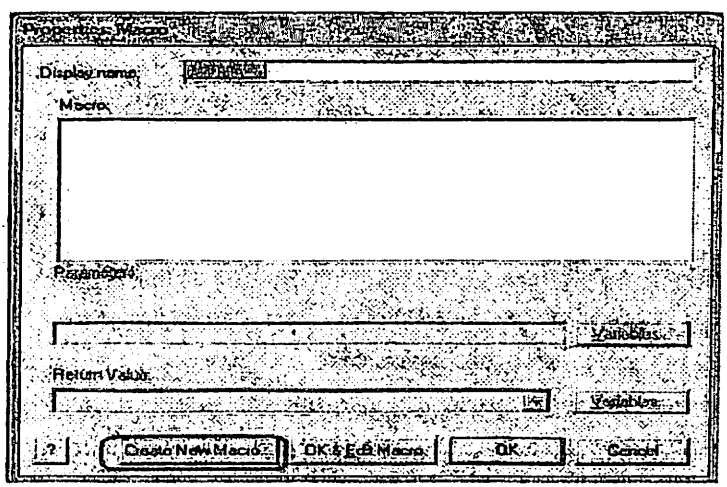


Рис. 3.3. Диалоговое окно компонента **Macro**

И вновь, имя компонента можно изменить. Если подпрограмма ещё не создана, а мы только что добавили компонент, следует использовать кнопку **Create New Macro**.

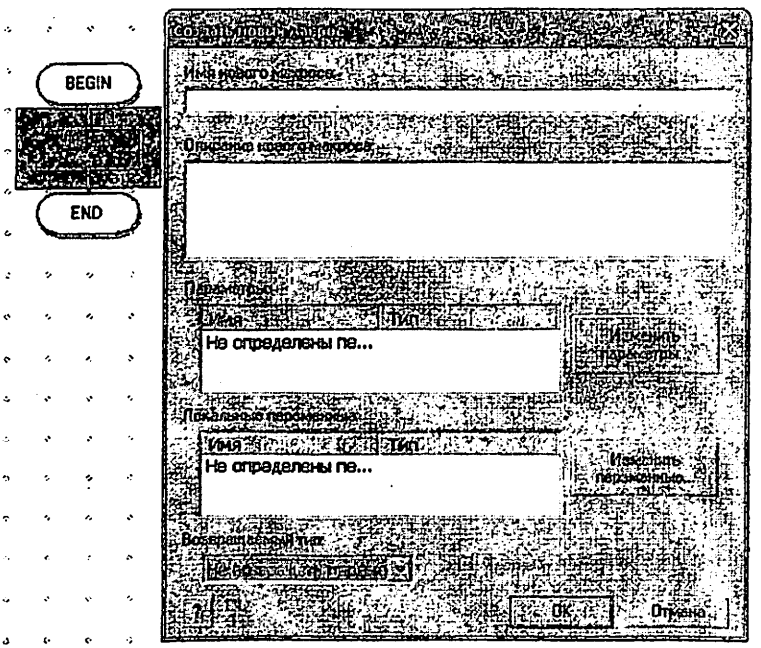


Рис. 3.4. Диалоговое окно создания нового макроса

Кроме имени новой подпрограммы, как и для функции или процедуры можно задать параметры (кнопка **Edit Parametrs**), 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 Makro**, то программа открывает второе рабочее окно создания подпрограммы.

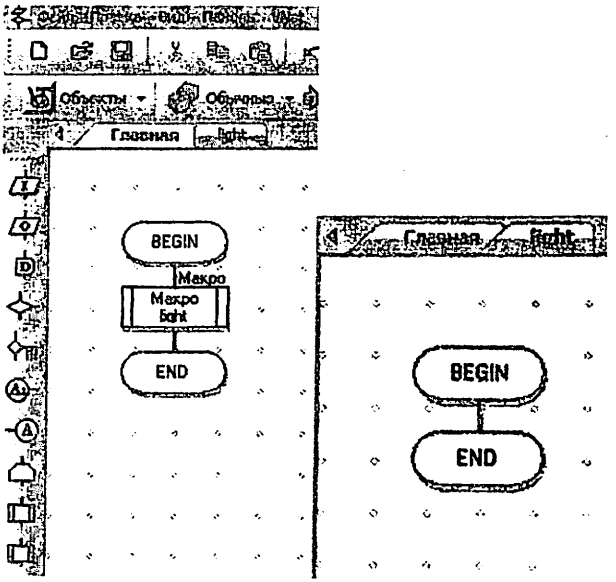


Рис. 3.5. Второе рабочее окно редактора для создания подпрограммы

К этому мы вернёмся позже, а сейчас отметим, что между созданием программы и созданием подпрограммы в **FlowCode** в техническом плане нет разницы. Метки в программе, особенно на ассемблере, применяются достаточно часто. Первая из пары точек **Connection Point** и есть метка, а вторая – переход к метке.

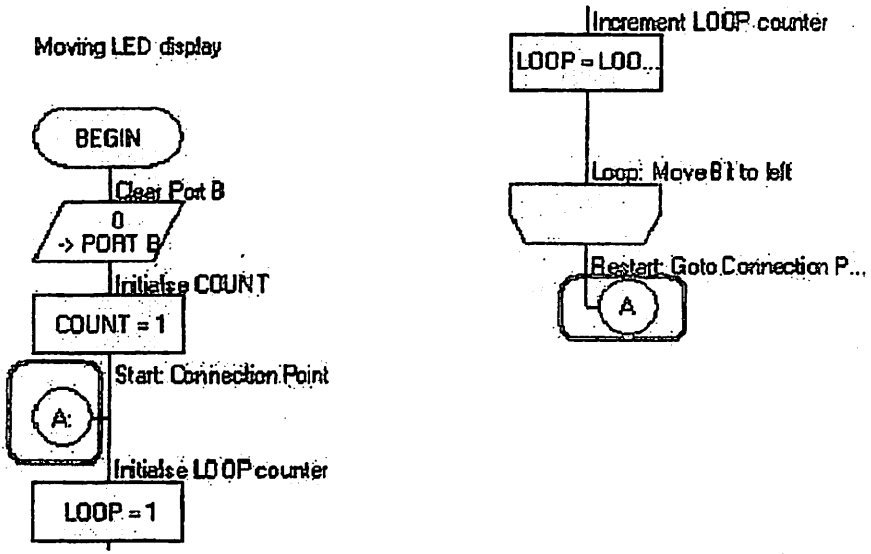


Рис. 3.6. Фрагмент программы с компонентом Connection Point

В диалоговом окне свойств второй точки соединения указываем переход:

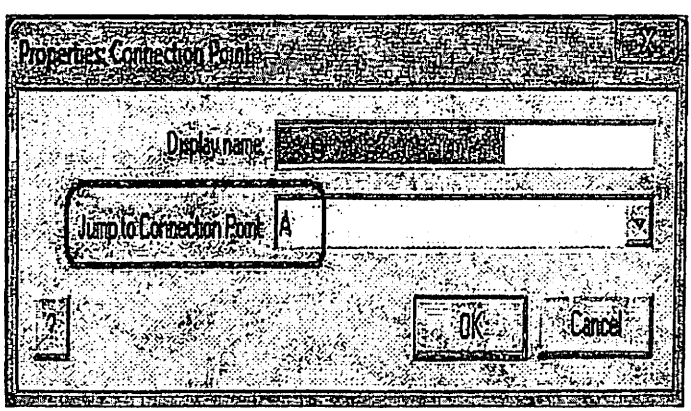


Рис. 3.7. Диалоговое окно свойств точки соединения

Варианты заданий для лабораторной работы № 3:

1) С помощью линейки светодиодов создать эффект бегущей строки (слева направо и справа налево), используя метки.

Форма и содержание отчета:

1) Титульный лист;
2) Формулировка варианта задания;
3) Блок-схема алгоритма решения задачи;
4) Визуальный отчет и результаты выполнения программы.

Контрольные вопросы:

1) Какая опция используется для проверки условия завершения цикла в начале цикла.
2) Для чего следует использовать кнопку **Create New Macro.**
3) Дать определение, программного компонента **Macro**, в каких случаях он используется?
4) Приведите свои примеры программ с **Connection Point.**

ЛАБОРАТОРНАЯ РАБОТА №4.

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 справо, 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 **Help** основного меню программы **FlowCode.**

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.

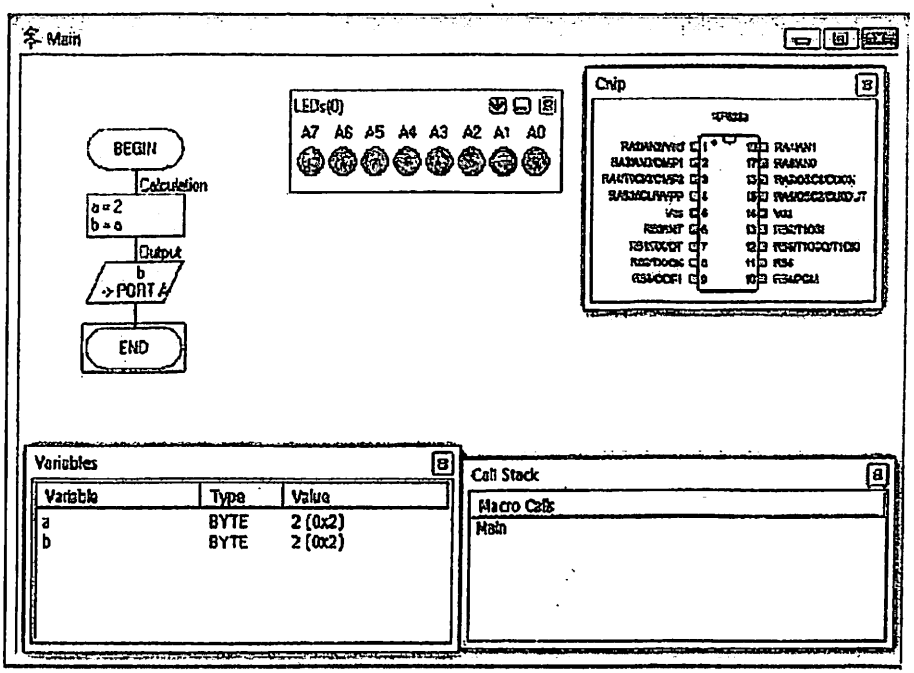


Рис. 4.1. Использование компонента **Calculation**

Все необходимые операции записываются в окне диалога.

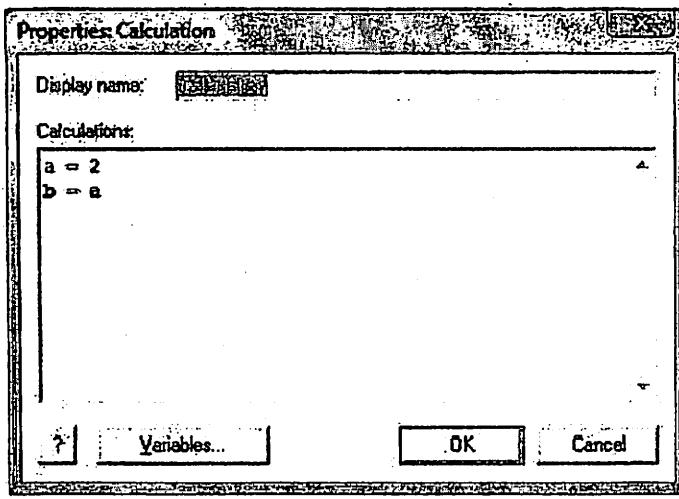


Рис. 4.2. Диалоговое окно компонента **Calculation**

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.

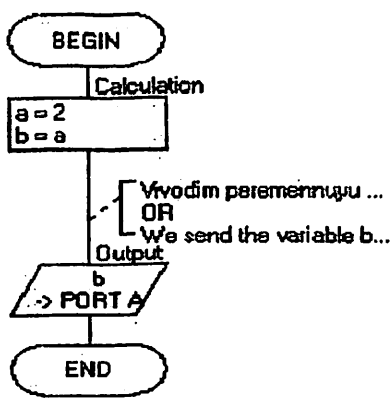


Рис. 4.3. Использование компонента **Comment**

Программный компонент **Comment** (комментарий) 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 возникнуть проблемы.

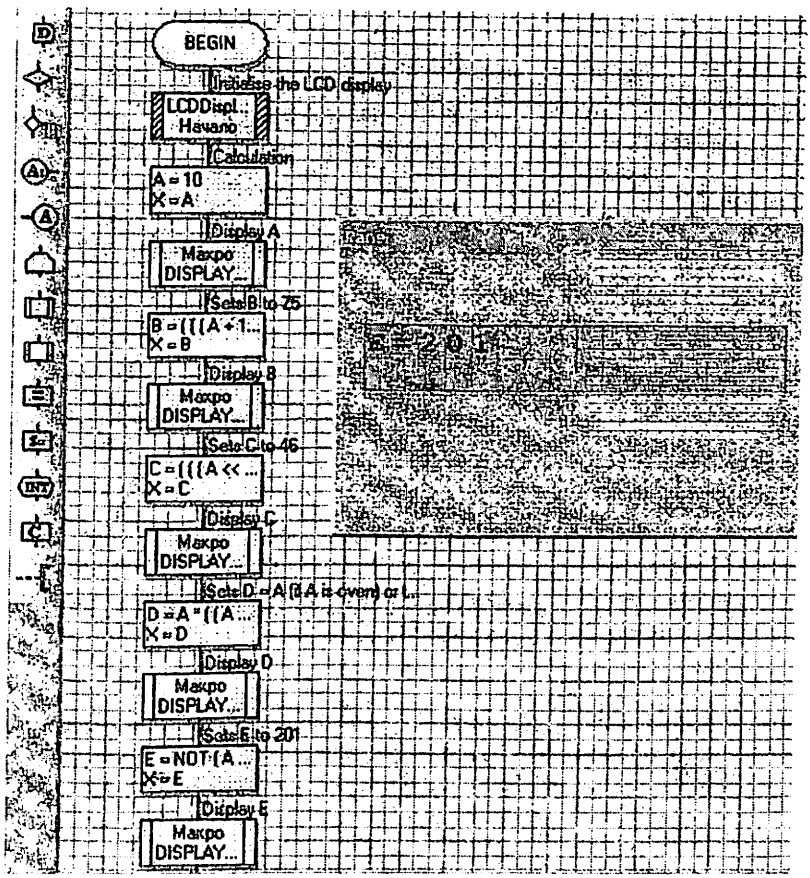


Рис 4.4. Фрагмент программы с использованием операций **AND, OR, MOD, NOT, XOR**

Варианты заданий для лабораторной работы № 4:

1) Напишите две программы, для вычисления функций используя операции **AND, OR, MOD, NOT, XOR.**

2) Напишите две программы, для вычисления математических функций.

Форма и содержание отчета:

1) Титульный лист;

2) Формулировка варианта задания;

3) Блок-схема алгоритма решения задачи;

4) Визуальный отчет и результаты выполнения программы.

Контрольные вопросы:

1) В каких случаях применяется программный компонент **Calculation.**

2) Для чего необходима кнопка **Variables.**

3) Приведите свои примеры программ с программным компонентом **Calculation.**

4) В чем состоит необходимость использования компонента **Comment.**

ЛАБОРАТОРНАЯ РАБОТА №5.

СОЗДАНИЯ ПРОСТЫХ ПРОГРАММ С ИСПОЛЬЗОВАНИЕМ LCD DISPLAY И LED КОМПОНЕНТОВ.

В программном пакете **Flow Code** есть инструмент для рисования передней панели создаваемого прибора и составные элементы для кнопок, клавиатур, выключателей, **LED, LCD,** сенсоров, внутренней **EEPROM,** 7-сегментных дисплеев.

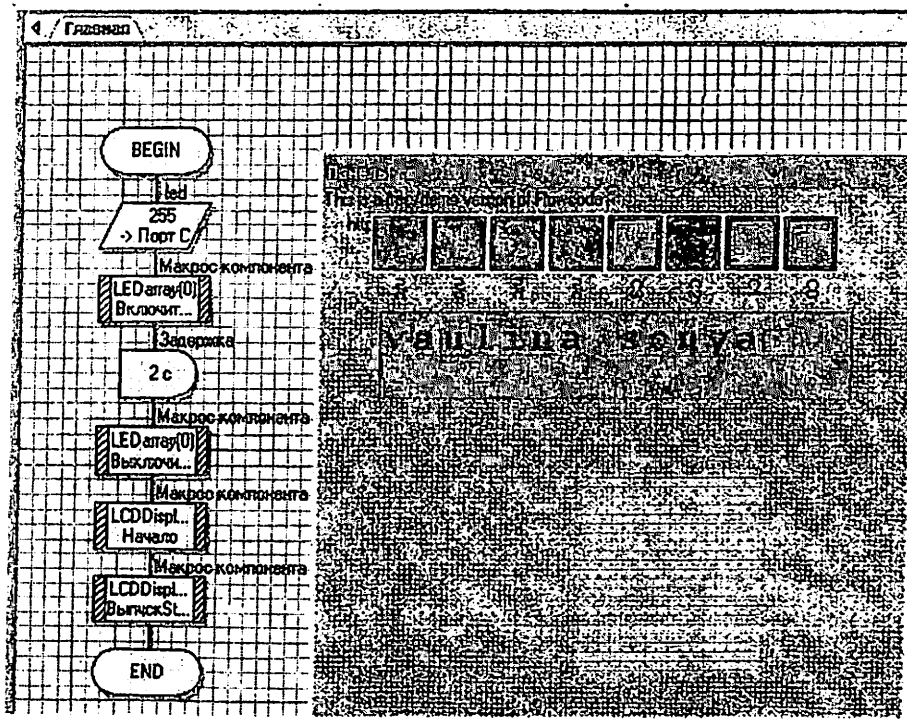


Рис.5.1. Фрагмент простой программы с циклом

Для создания этой программы вначале производим настройку соединений LED индикаторов и LSD display:

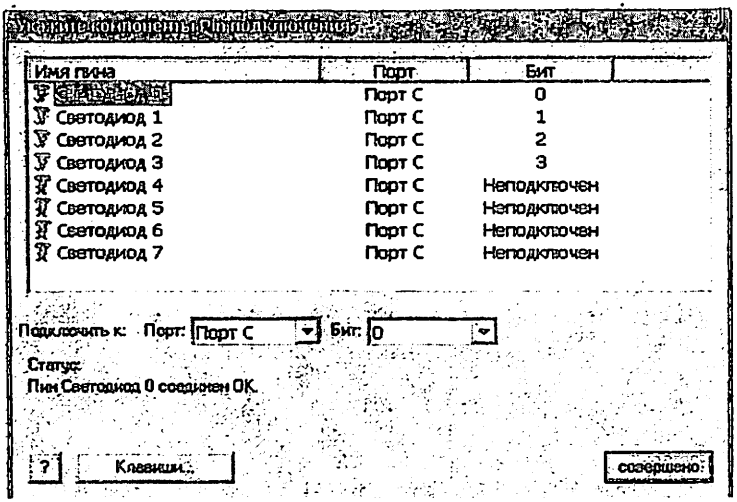


Рис. 5.2. Настройка соединения LED индикаторов

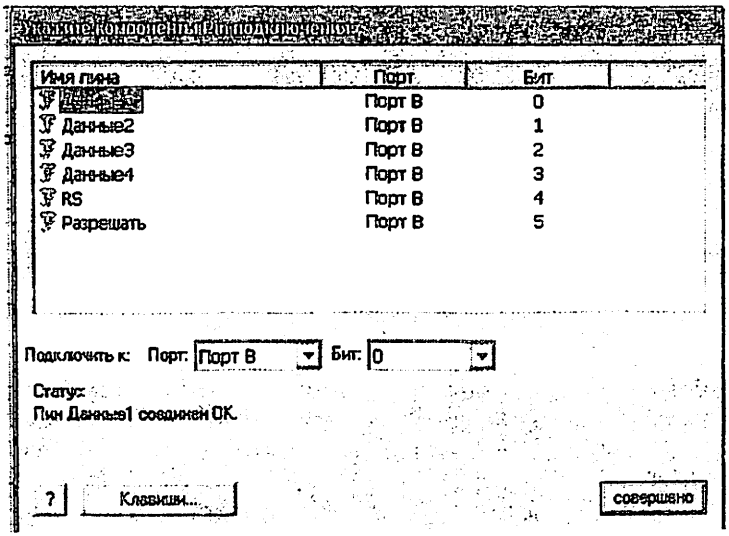


Рис.5.3. Настройку соединения LSD display

После овала **Begin** – начало, добавляем элемент **Output** и двойным щелчком левой клавиши мышки по нему, открываем диалоговое окно его свойств:

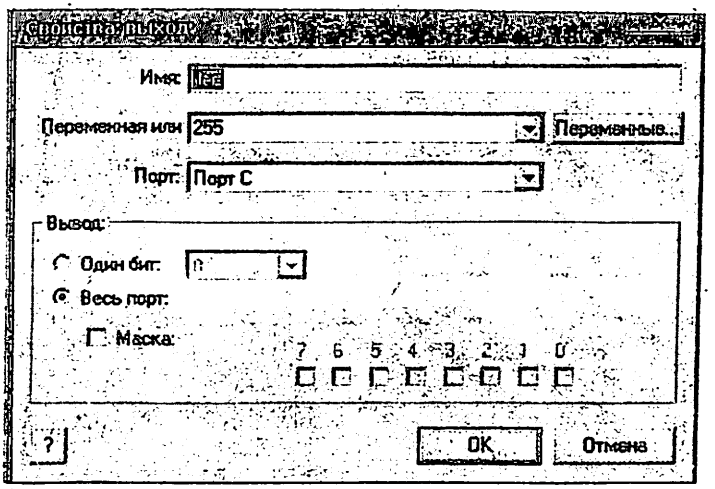
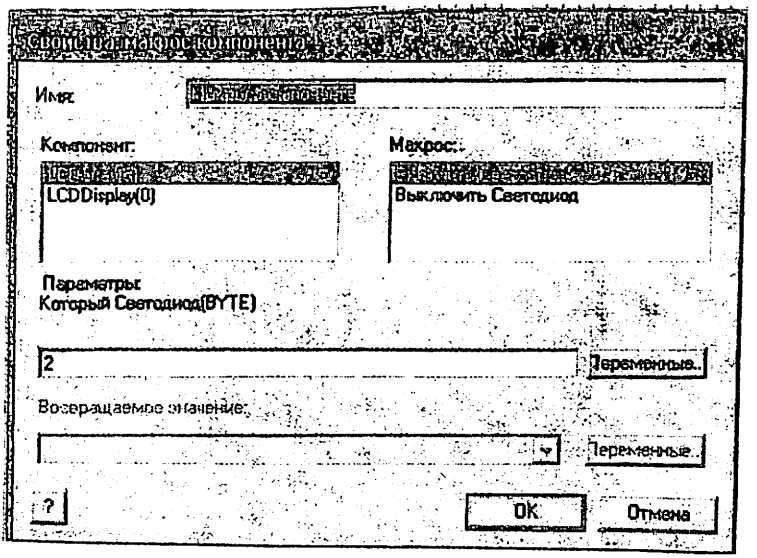


Рис 5.4. Изменение свойств порта С в диалоговом окне

Далее необходимо добавить в программу компонент **Component Macro**. Включаем линейку светодиодов, вводим задержку 2с и снова, добавив в программу **Component Macro**, выключаем второй светодиод.



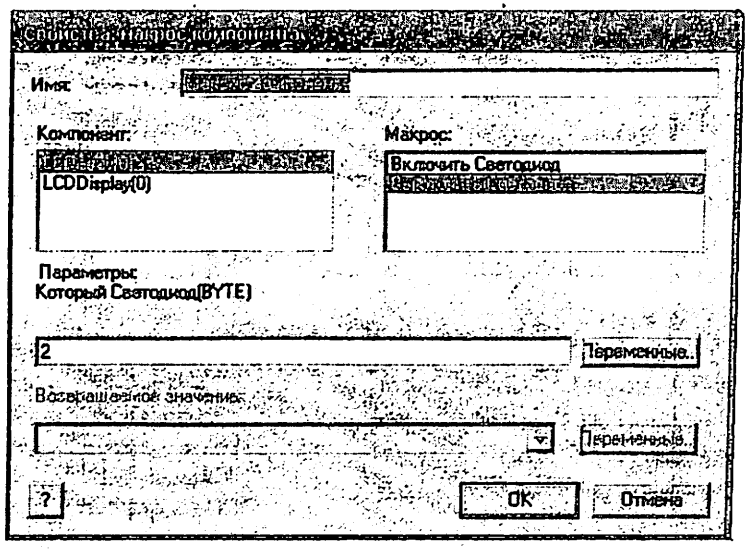
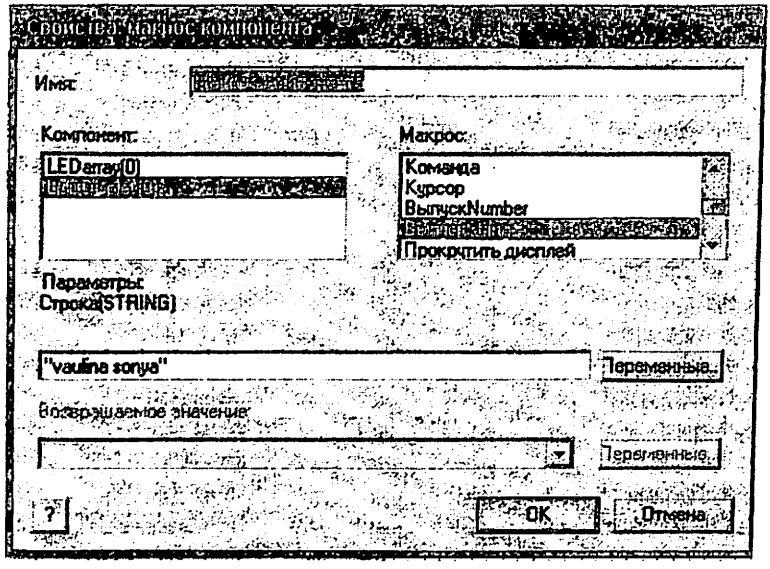


Рис 5.5. Настройка работы LED индикаторов

Теперь начинаем настройку работы LSD display, запускаем его и набираем необходимый текст в графу строка String.



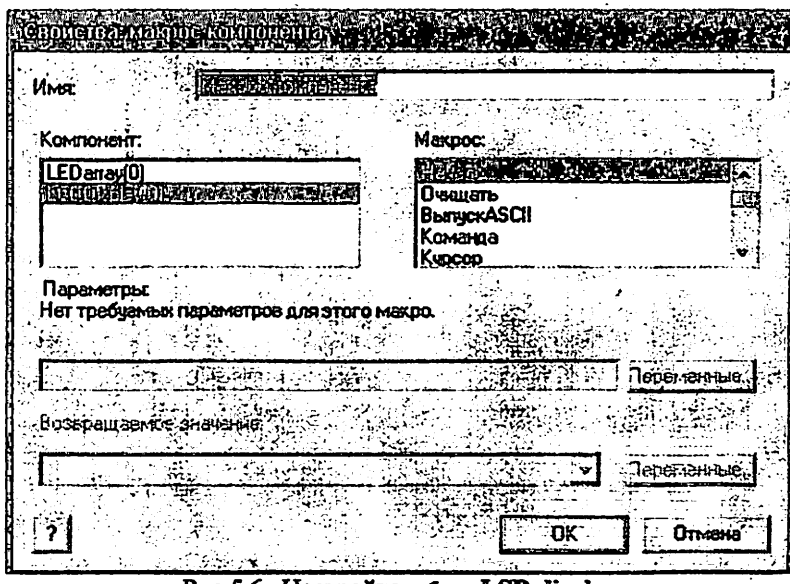
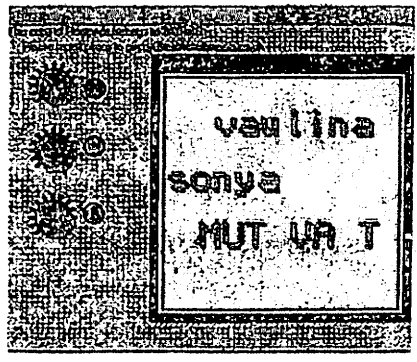


Рис 5.6. Настройка работы LSD display

Ниже приведён фрагмент простой программы с использованием элемента Graphical LCD.



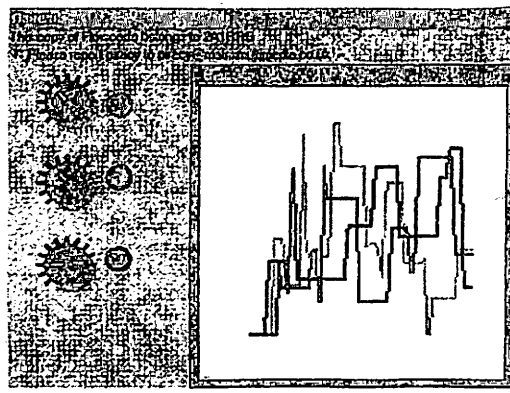


Рис 5.7. Фрагмент программы с элементом Graphical LCD.

Варианты заданий для лабораторной работы № 5:

1) Напишите простые программы, используя элемент Graphical LCD, для функций с арифметическими и логическими операциями.

Форма и содержание отчета:

1) Титульный лист;
2) Формулировка варианта задания;
3) Блок-схема алгоритма решения задачи;
4) Визуальный отчет и результаты выполнения программы.

Контрольные вопросы:

1) Опишите порядок написания программы для подключения и настройки LED индикаторов, LSD display.
2) Расскажите, как осуществить изменение текста на LSD display.
3) Где на практике находит применение использование программ с элементом Graphical LCD.

ЛАБОРАТОРНАЯ РАБОТА №6.

СЛОЖНЫЕ ПРОГРАММЫ, ИСПОЛЬЗУЮЩИЕ ФУНКЦИЮ ПРЕРЫВАНИЯ.

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 выполнения задания. Пока внешнее устройство «трудится», процессор может заняться выполнением другой задачи. А получив запрос на прерывание, поместить в стек, специально отведённую 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.

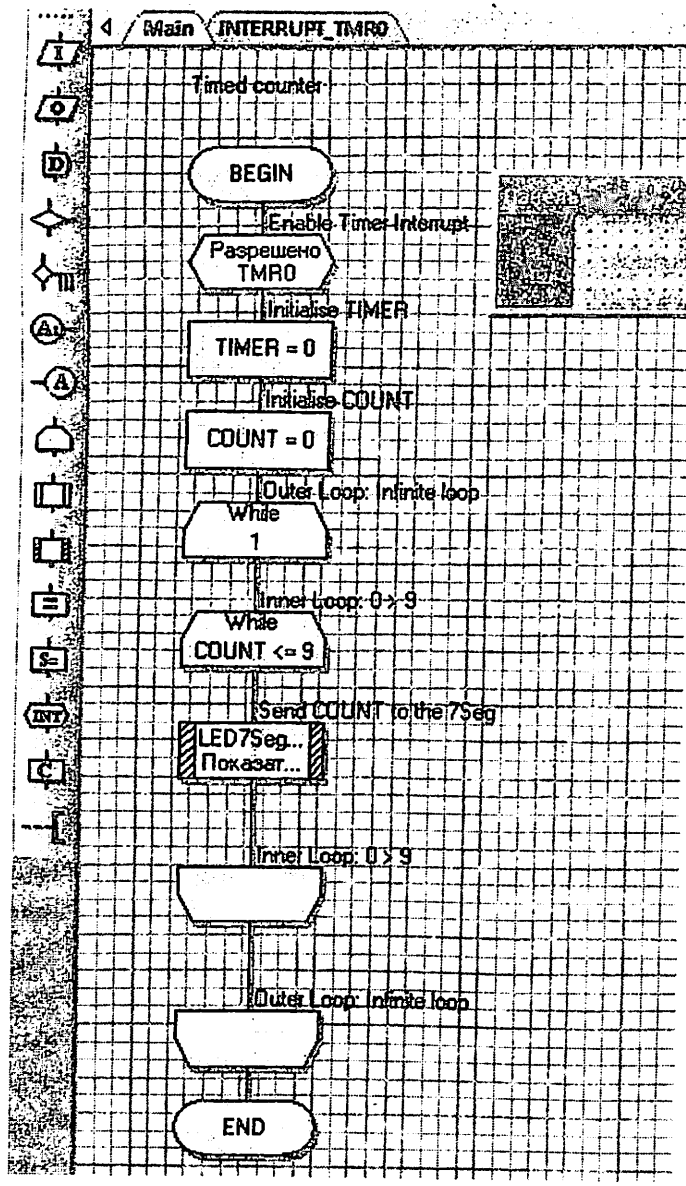


Рис. 6.1. Использование программного компонента **Interrupt** (прерывание).

На рисунке кроме основного окна программы (Main)видно второе окно подпрограммы **INTERRUPT_TMR0**. Кроме прерывания по таймеру, как в показанной выше программе, диалоговое окно свойств программного компонента **Interrupt** дает возможность выбрать из выпадающего списка ещё 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)**.

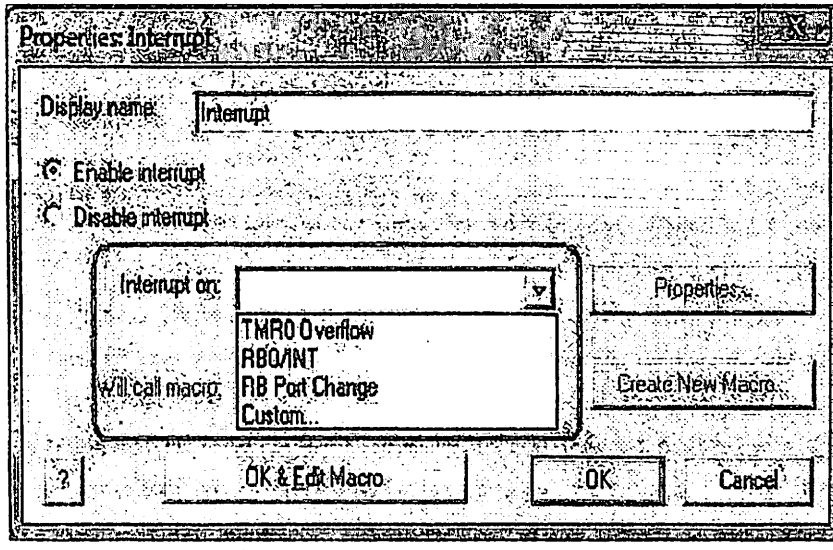


Рис. 6.2. Допустимые виды прерывания для модели **PIC16F88**

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.

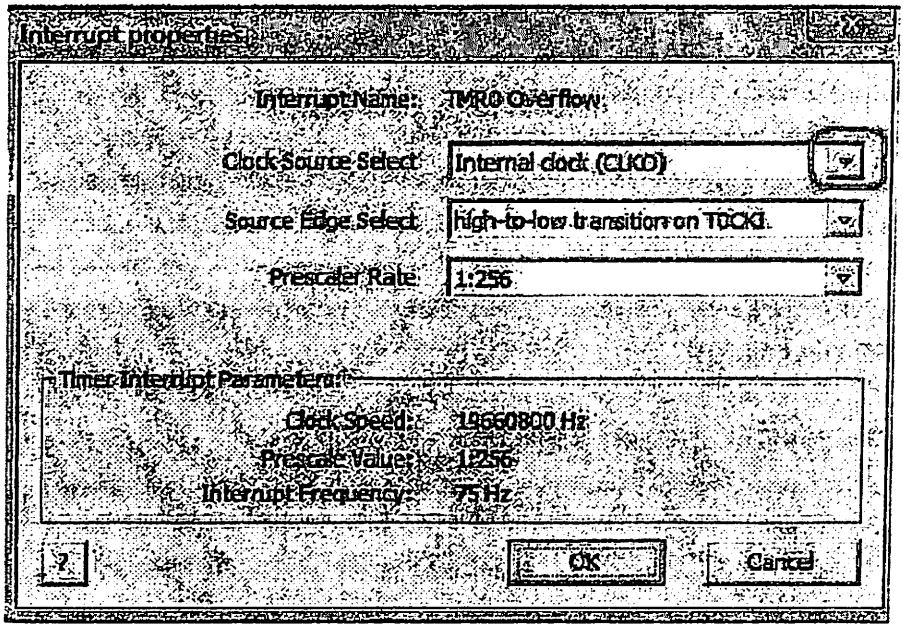


Рис. 6.3. Свойства прерывания, заданные для программы выше

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. 6.2.), затем с помощью кнопки **OK & Edit Makro** открыть новое рабочее поле для подпрограммы и начать писать подпрограмму.

Рассмотрим подпрограмму выше приведённого примера.

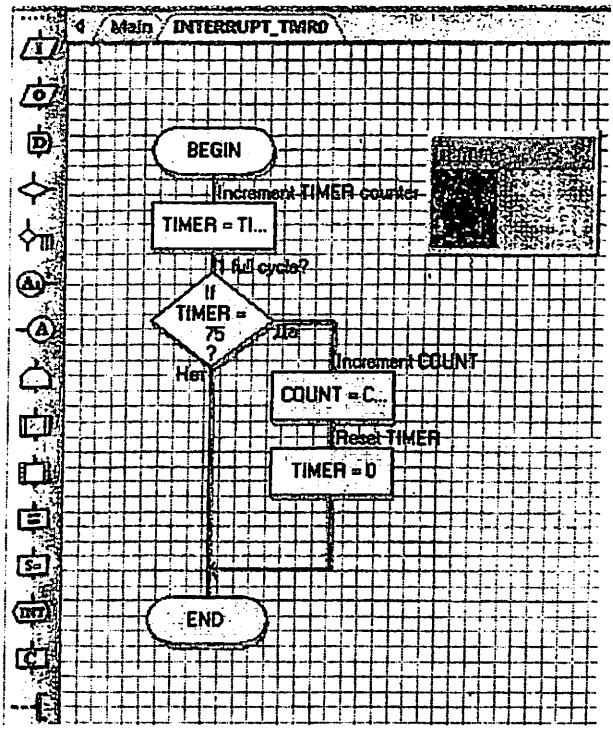


Рис. 6.4. Подпрограмма обслуживания прерывания по таймеру

Откроем диалоговое окно первого элемента программы **Increment TIMER counter**(наращивание счетчика таймера).

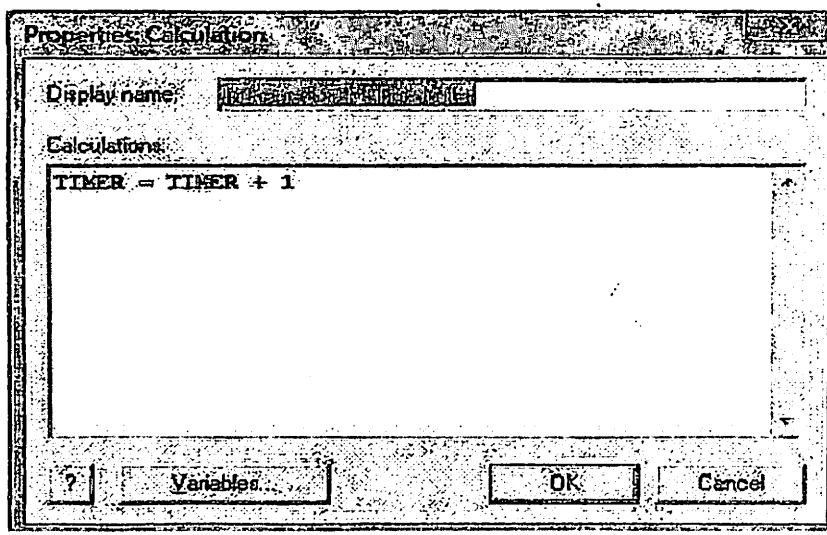


Рис. 6.5. Первый элемент подпрограммы

Как видно, это программный компонент **Calculation**, где значение переменной **TIMER** увеличивается на единицу. Затем следует проверка значения переменной (программный компонент **Decision** ветвление).

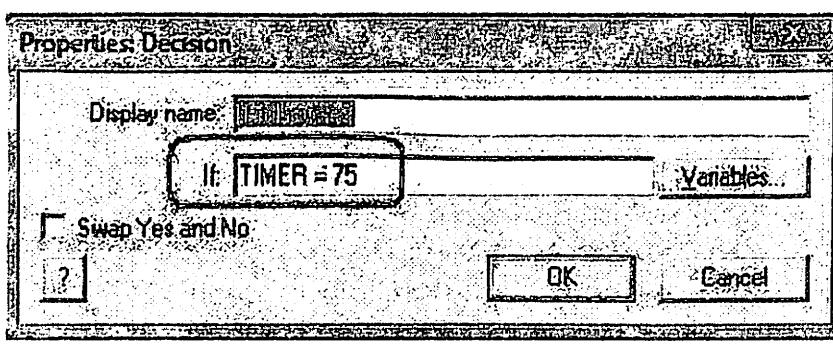


Рис. 6.6. Проверка значения переменной

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 ***1full cycle.***

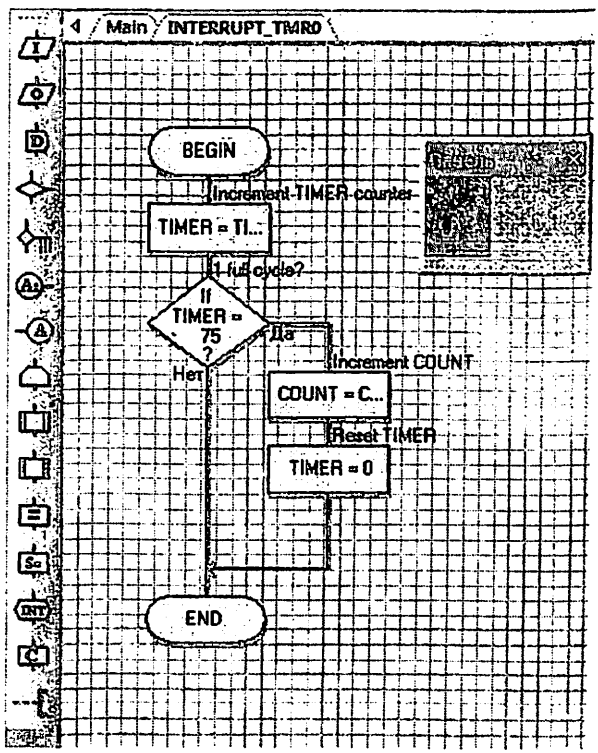


Рис. 6.7. Ветвление программы

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. Внесём 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.

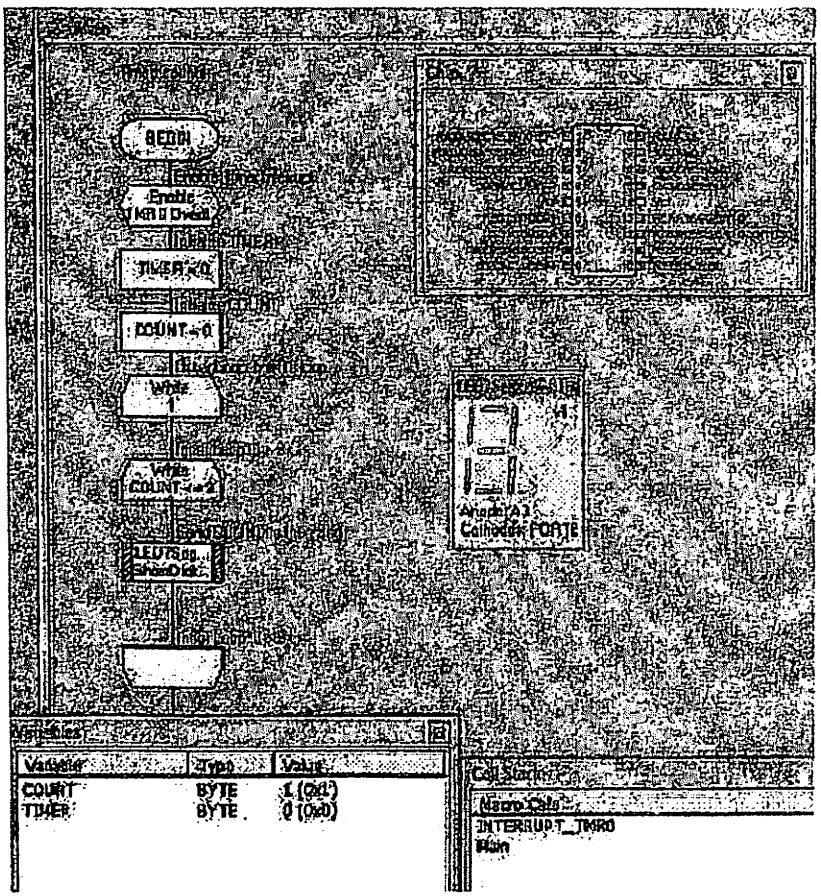


Рис. 6.8. Отладочный запуск программы

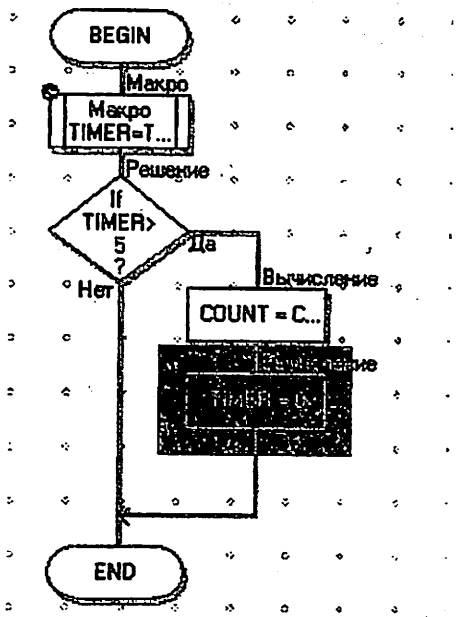


Рис. 6.9. Отладочный запуск подпрограммы

Прерывание полезный приём, но 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.

Варианты заданий для лабораторной работы № 6:

1) Используя семисегментный индикатор с четырьмя секциями, собрать электронные часы или таймер на базе микроконтроллера.

Форма и содержание отчета:

1) Титульный лист;

2) Формулировка варианта задания;

3) Блок-схема алгоритма решения задачи;

4) Визуальный отчет и результаты выполнения программы.

Контрольные вопросы:

1) Как работает механизм прерывания, почему возникает необходимость в прерывании.

2) Опишите механизм создания подпрограммы обслуживания прерывания.

3) Какую возможность даёт программный компонент **Interrupt**.

4) С помощью, какой кнопки можно задать ряд свойств прерывания.

5) Для чего служит кнопка **OK & Edit Makro**.

ЛАБОРАТОРНАЯ РАБОТА №7.

ПРОГРАММЫ С ВСТРОЕННЫМ МОДУЛЕМ USART.

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**.

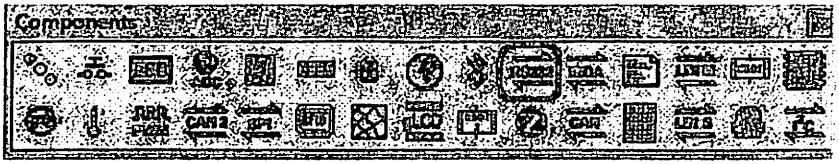


Рис. 7.1. Компонент **RS232** для работы с **USART**.

После того, как **RS232** добавлен (достаточно «щелкнуть» по нему мышкой), можно добавить в программу компонент **Component Macro**.

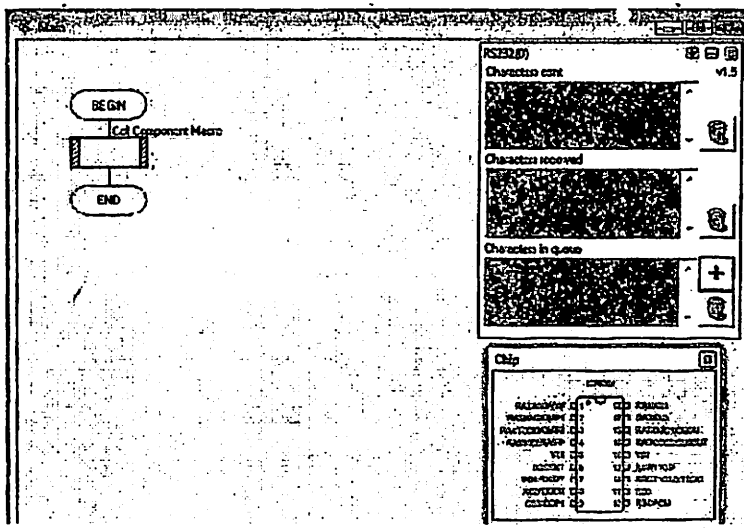


Рис 7.2. Начало работы с программой, использующей USART.

Перед тем, как продолжить создание программы, условимся: если мы получаем по сети литеру N, что устанавливает (в высокое состояние) вывод 0 порта A, а если 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.

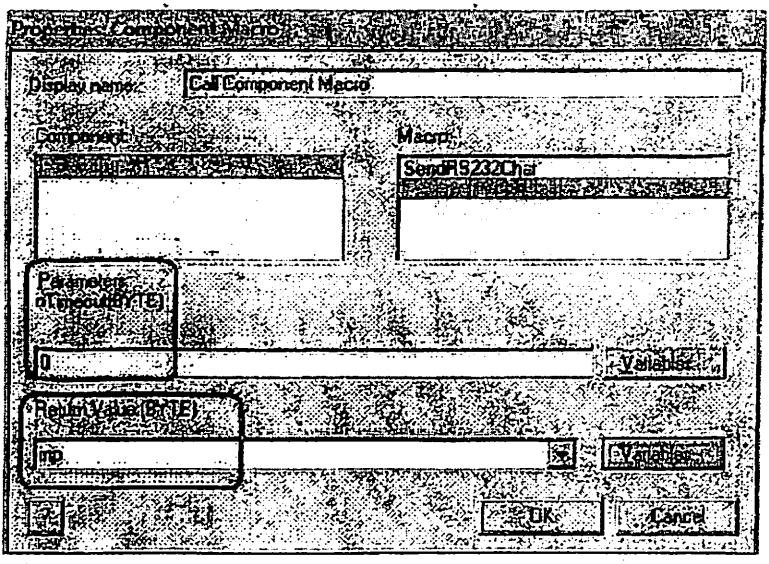


Рис. 7.3. Диалоговое окно Component Macro для RS232.

Задав свойства, погрузим всю программу в бесконечный цикл и используем условное ветвление для выполнения ответных действий на приход управляющих символов.

Напоминаем, что с приходом символа «N» микроконтроллер должен установить высокий уровень на выводе 0 порта А. к этому выводу можно подключить, например реле, которое будет своими контактами включать свет. С приходом же символа «F» микроконтроллер должен установить низкий уровень на выводе 0 порта А, выключая нагрузку.

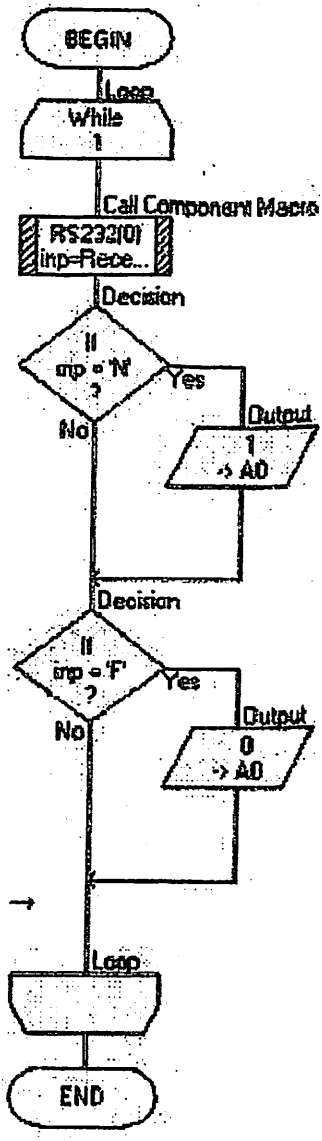


Рис.7.4. Программа отклика на приход символа по сети.

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 равное «F». В этом случае в регистр порта A записывается 0.

Цикл ***While*** в данном случае не имеет определённого условия выхода, то есть, становится бесконечно выполняемым.

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).

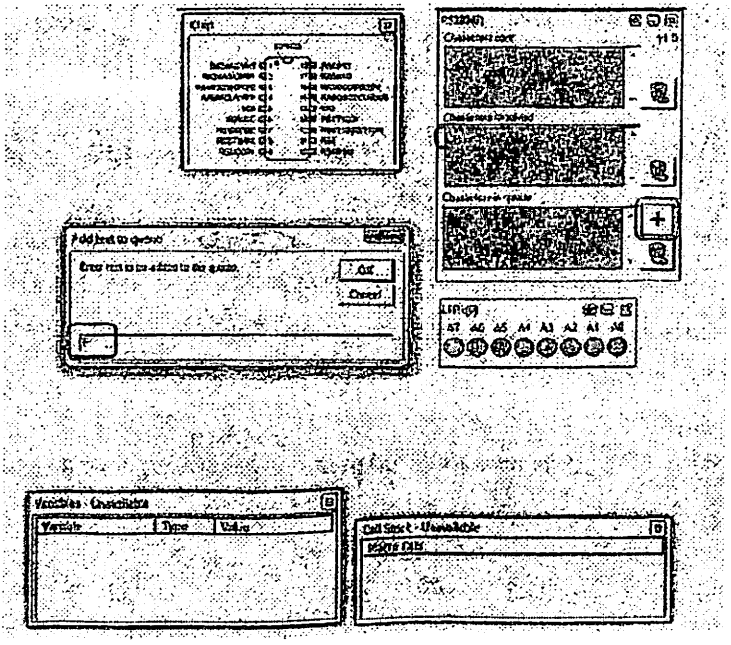


Рис 7.5. Отладка программы.

Нажав кнопку «+» панели ***Characters in queue***, вы попадёте 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 ещё 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, соединённой 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 своё 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.

Создав новую программу, сразу добавим в неё 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, а если кнопка нажата будем менять значение переменной ***flag***: используя такую конструкцию из двух логических операций ***flag = (Not flag) And 1***, которую запишем с помощью программного компонента ***Calculation***.

И нам остаётся только отправлять символ «N», если переменная ***flag*** равна 1, и символ«F» когда она становится равной 0. Как 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***.

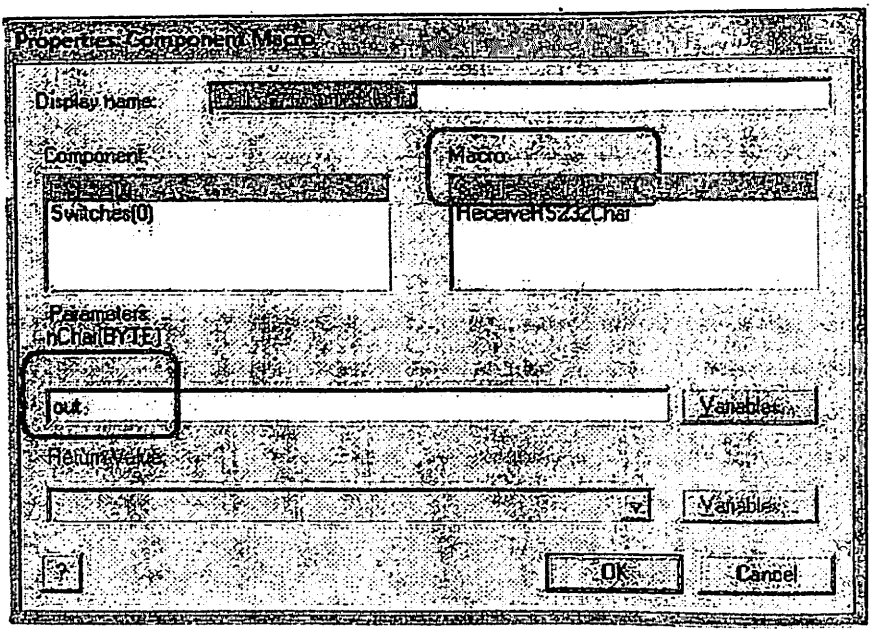


Рис. 7.6. Настройка компонента RS232 в диалоговом окне его свойств.

Для успешной работы программы добавим паузу после каждой отправки символа. Программа приобретает такой вид:

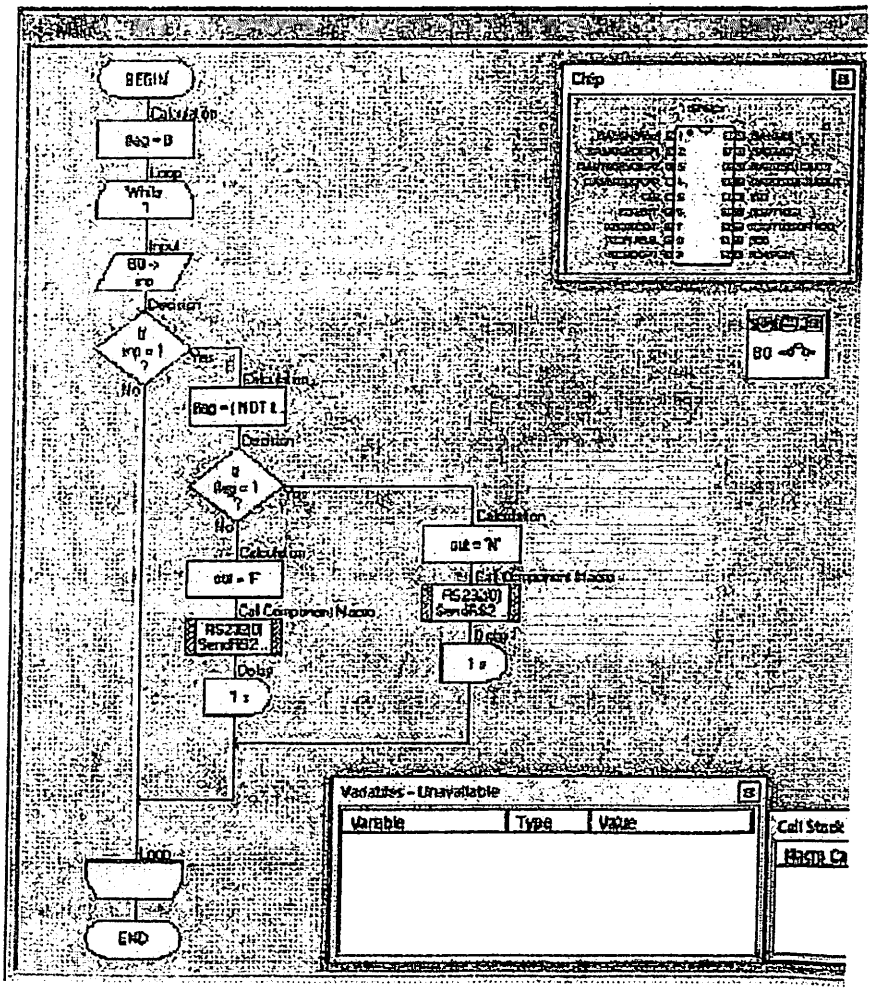


Рис. 7.7. Вторая программа, использующая USART.

Варианты заданий для лабораторной работы № 7:

1) Напишите свою программу, использующую встроенный модуль USART.

Форма и содержание отчета:

1) Титульный лист;
2) Формулировка варианта задания;
3) Блок-схема алгоритма решения задачи;
4) Визуальный отчет и результаты выполнения программы.

Контрольные вопросы:

1) Дать определение, что такое USART.
2) Как и для чего создаётся переменная *inp*.
3) Что такое управляющие символы, и какие значения им присваиваются (от каких условий это зависит).
4) Приведите пример программы, которая не получает символы по USART, а отправляет их.

ЛАБОРАТОРНАЯ РАБОТА №8.

ПРОГРАММЫ С ВСТРОЕННЫМ МОДУЛЕМ PWM.

Микроконтроллер PIC16F628A (и другие), имеет встроенный модуль PWM (широтно-импульсный модулятор). Приведём пример программы, где меняется скважность импульсов на выходе 3 порта B.

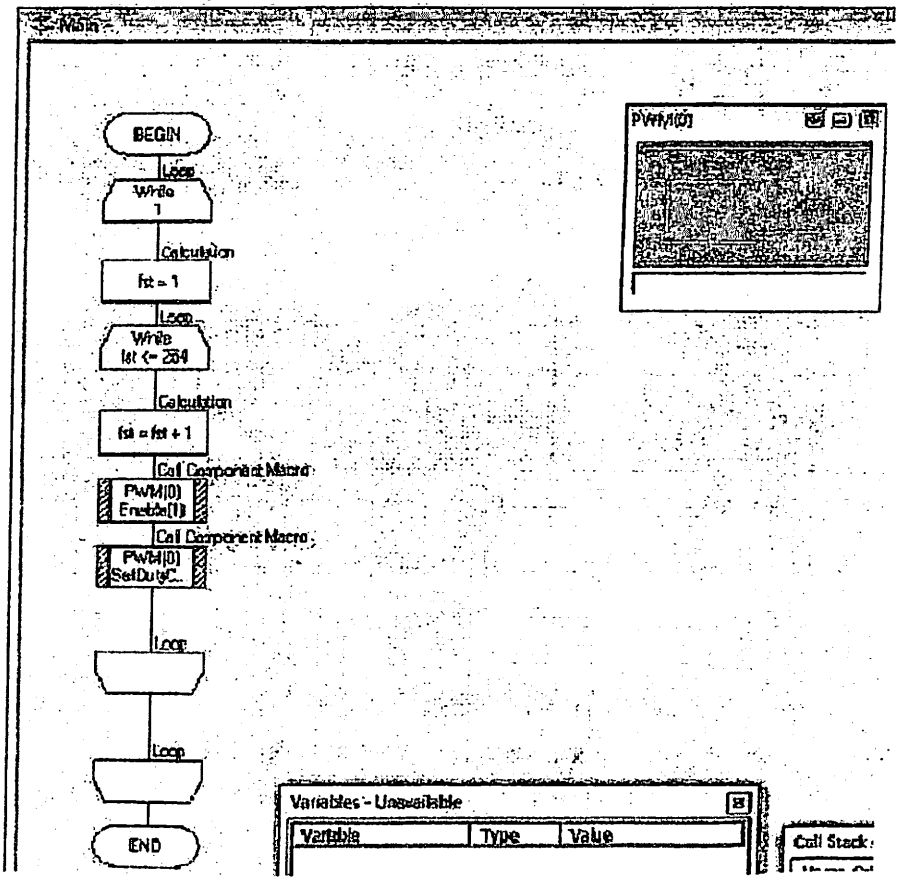


Рис 8.1. Программа работы с ШИМ (широтно-импульсной модуляцией).

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 использовать ШИМ.

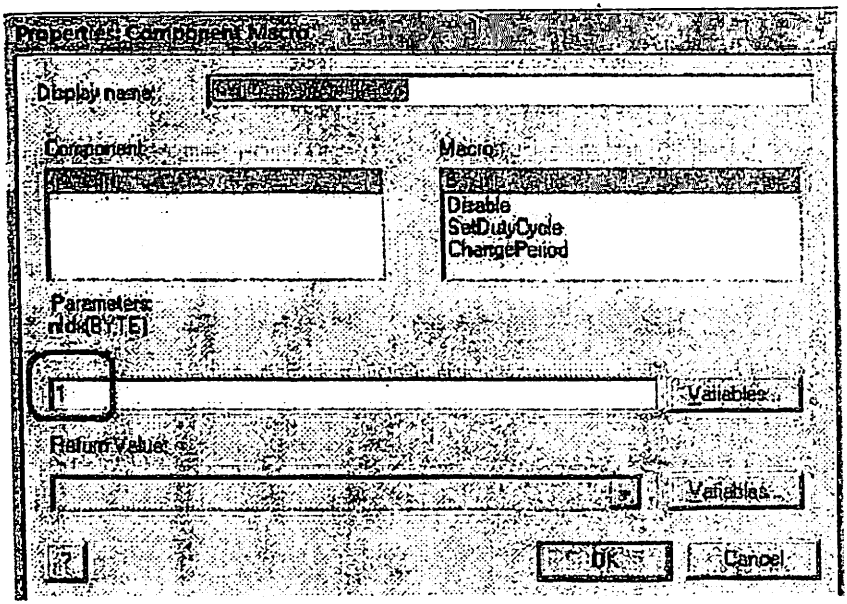


Рис.8.2. Диалоговое окно дополнительного компонента PWM.

Отмеченная на рисунке единица относится к номеру канала, поскольку есть модели, имеющие несколько каналов ШИМ.

В следующем диалоговом окне (второй **Component Macro**) задаётся скважность импульсов.

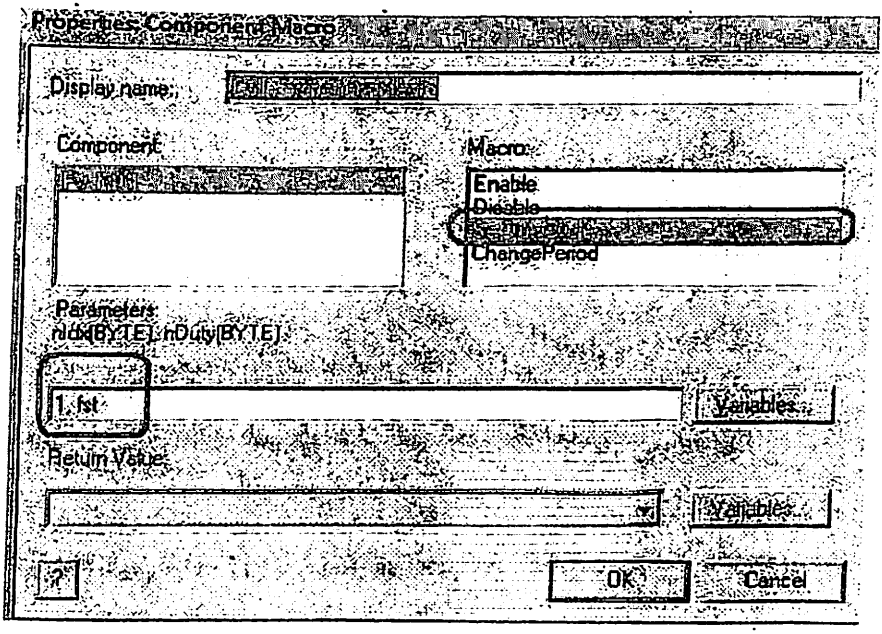


Рис.8.3. Задание скважности импульсов через переменную *fst.*

После запуска программы в окне компонента появится сигнал, плавно меняющий скважность импульса.

Варианты заданий для лабораторной работы № 8:

1)Напишите свою программу, использующую встроенный модуль PWM.

Форма и содержание отчета:

1) Титульный лист;

2) Формулировка варианта задания;

3) Блок-схема алгоритма решения задачи;

4) Визуальный отчет и результаты выполнения программы.

Контрольные вопросы:

1) Дать определение, что такое PWM и где он находит применение.
2) Как и для чего используется переменная *fst*.
3) Расскажите, что вам известно про одноканальные и многоканальные ШИМ.
4) Как задаётся скважность импульса (определение скважности) и с помощью какой переменной можно её менять.

ЛАБОРАТОРНАЯ РАБОТА №9.

ПРОГРАММЫ С ИСПОЛЬЗОВАНИЕМ КОМПОНЕНТА KeyPad.

Очень часто устройство на базе микроконтроллера имеет клавиатуру. С целью экономии выводов кнопки можно включить «матрицей». При четырёх кнопках, как на рисунке ниже, мы не получаем существенного выигрыша, но, используя восемь выводов, можно получить клавиатуру с шестнадцатью клавишами.

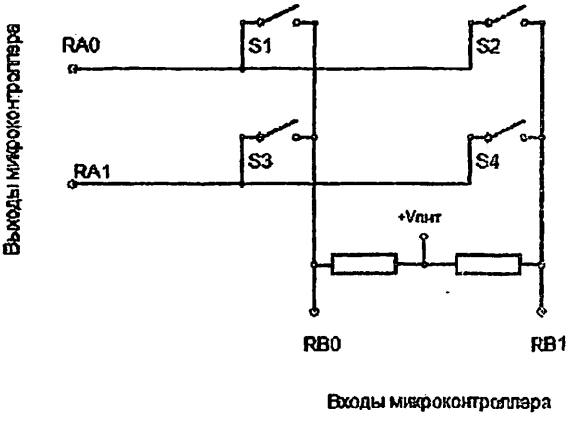


Рис.9.1. «Матричное» включение четырёх клавиш.

Попеременно передавая низкий уровень на выходы микроконтроллера **RA0** и **RA1**, определяя состояние входов **RB0** и **RB1**, можно определить состояние четырёх кнопок **S1 – S4**. Фрагмент «блок - схемы» программы может выглядеть так:

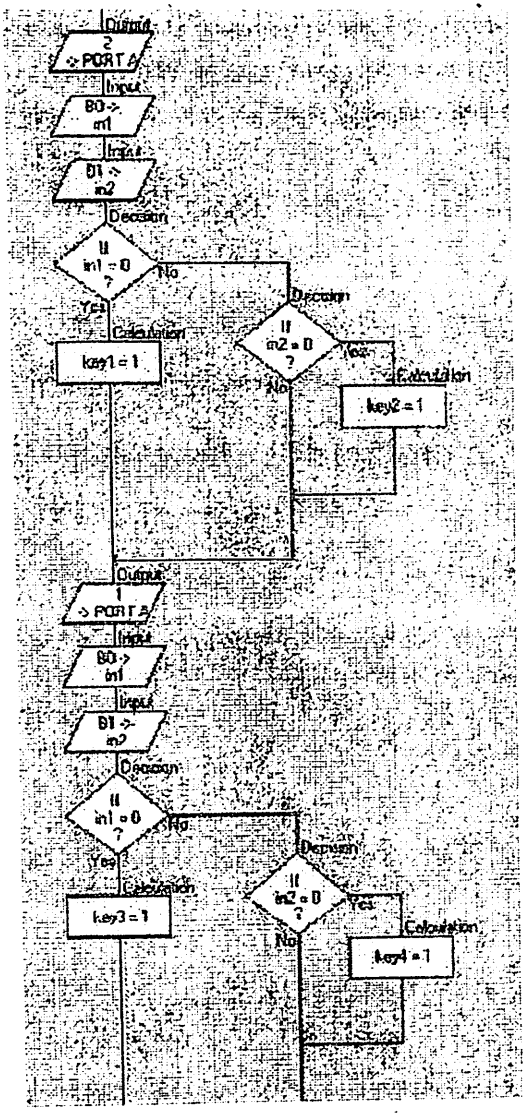


Рис.9.2. «Блок - схема» программы обслуживания клавиатуры.

«За кадром» остался бесконечный цикл, в котором работает опрос клавиатуры. Программа может быть построена и иначе, например, с использованием подпрограмм.

В полной версии FlowCode работает дополнительный компонент KeyPad. Для работы с клавиатурой можно использовать этот компонент. 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.

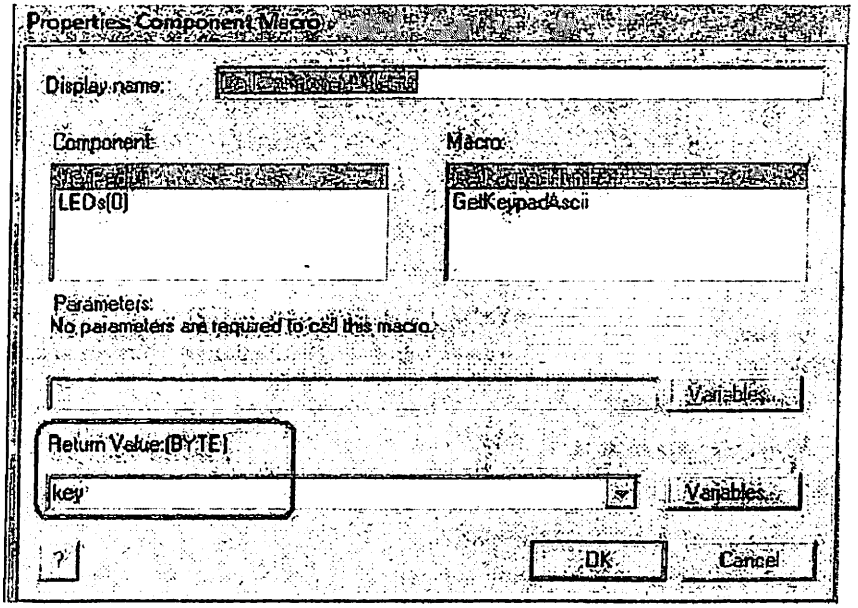


Рис.9.3. Диалоговое окно программного компонента KeyPad.

Этот номер будет присваиваться созданной нами переменной *key*. Дальнейшие действия зависят от наших намерений. Простейшая программа проверки клавиатуры может иметь следующий вид:

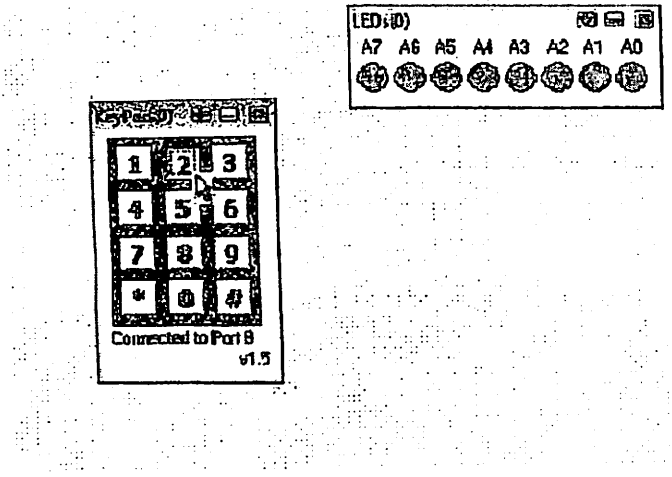


Рис.9.4. Работа клавиатуры в программе FlowCode.

Варианты заданий для лабораторной работы № 9:

1) Напишите программу для кодового замка, используя программный компонент KeyPad.

Форма и содержание отчета:

1) Титульный лист;
2) Формулировка варианта задания;
3) Блок-схема алгоритма решения задачи;
4) Визуальный отчет и результаты выполнения программы.

Контрольные вопросы:

1) Дать определение понятия «Матричное» включение клавиш.
2) Опишите, настройку программы с компонентом KeyPad.

Список литературы:

1) Гололобов В. Н. - Qucs и FlowCode. Программы для тех, кто интересуется электроникой. Интернет, 2009. – 358 с.

2) Визуальное проектирование в системе Flowcode. Радиолюбительский сайт. dxportal.ru›knigi/obuchenie…sisteme-flowcode.html.

3) Радио для всех - портфель. junradio.com›blog/mikrokontrollery_flowcode

4) ПРОГРАММА FLOWCODE - полет третий. – МикроКонтроллеры. elektron.ucoz.ru›publ/8-1-0-72.

5) Flowcode. схем.net›FlowCode.

6) Программа Flowcode и программатор Microchip PICkit 2. kit-e.ru›assets/files/pdf/2009_06_74.pdf

СОДЕРЖАНИЕ

Формат 60x84 1/16
Заказ № - 51. Тираж - 30

Отпечатано в Издательско полиграфическом
центре «ALOQACHI» при ТУИТ
Ташкент ул. Амир Темура, 108