Визуальное проектирование в системе Flowcode

Что такое Flowcode в двух словах? Это конструктор, состоящий из набора «деталей» для сборки готового изделия. Готовым изделием будет программа, записываемая в МК (микроконтроллер) и исполняемая в микроконтроллерной системе. «Деталями» являются разнообразные программные конструкции, позволяющие организовать тот или иной вычислительный процесс, и компьютерные модели (симуляторы) различных внешних устройств, подключаемых к МК. Поскольку все «детали» созданы разработчиками Flowcode, пользователю остается только правильно собрать из них действующую схему. Для этого, вообще говоря, не требуется даже знать языков программирования, хотя это все же желательно.

Несмотря на то, что первоначальный созданный проект работает именно в режиме симуляции, система создает и конечный, вполне работоспособный код, который может благополучно выполняться на реальном МК. За простоту разработки приходится расплачиваться, возможно, излишне расточительным использованием ресурсов МК и более низким по сравнению с другими системами быстродействием, однако для многих задач это не критично. Кроме того, не будем забывать, что данная статья – лишь первый учебный шаг к освоению микроконтроллерного программирования.

Среда проектирования Flowcode

В этом вводном разделе мы познакомимся с организацией визуальной оболочки пакета Flowcode (рассматривается Flowcode V4 for PICmicros русскоязычный вариант). Прежде всего, представим типичный вид экрана программы с открытым проектом. Основное окно будем называть рабочей областью. Ее вид показан на рис. 1.

Как во всех обычных Windows-приложениях, в верхней части окна расположено главное меню, пункты которого мы рассмотрим чуть позже. Ниже находится панель инструментов с кнопками, дублирующими вызов основных команд меню. Слева располагается инструментарий, который мы будем называть вертикальной линейкой. Отсюда мы будем брать «кубики конструктора», из которых будет состоять проект. Инструменты размещаются на так называемой диаграмме: вертикальной структуре между автоматически создаваемыми началом BEGIN и концом END.

В представленном варианте в верхней части окна расположена панель компонентов (Обычные, Входы, Выходы и т.д., может быть размещена по другому). С ее помощью к проекту подключаются эмуляции различных

внешних устройств. В частности, в примере на рис. 1 подключен эмулятор светодиода (внизу на панели отображения подключенных компонентов).

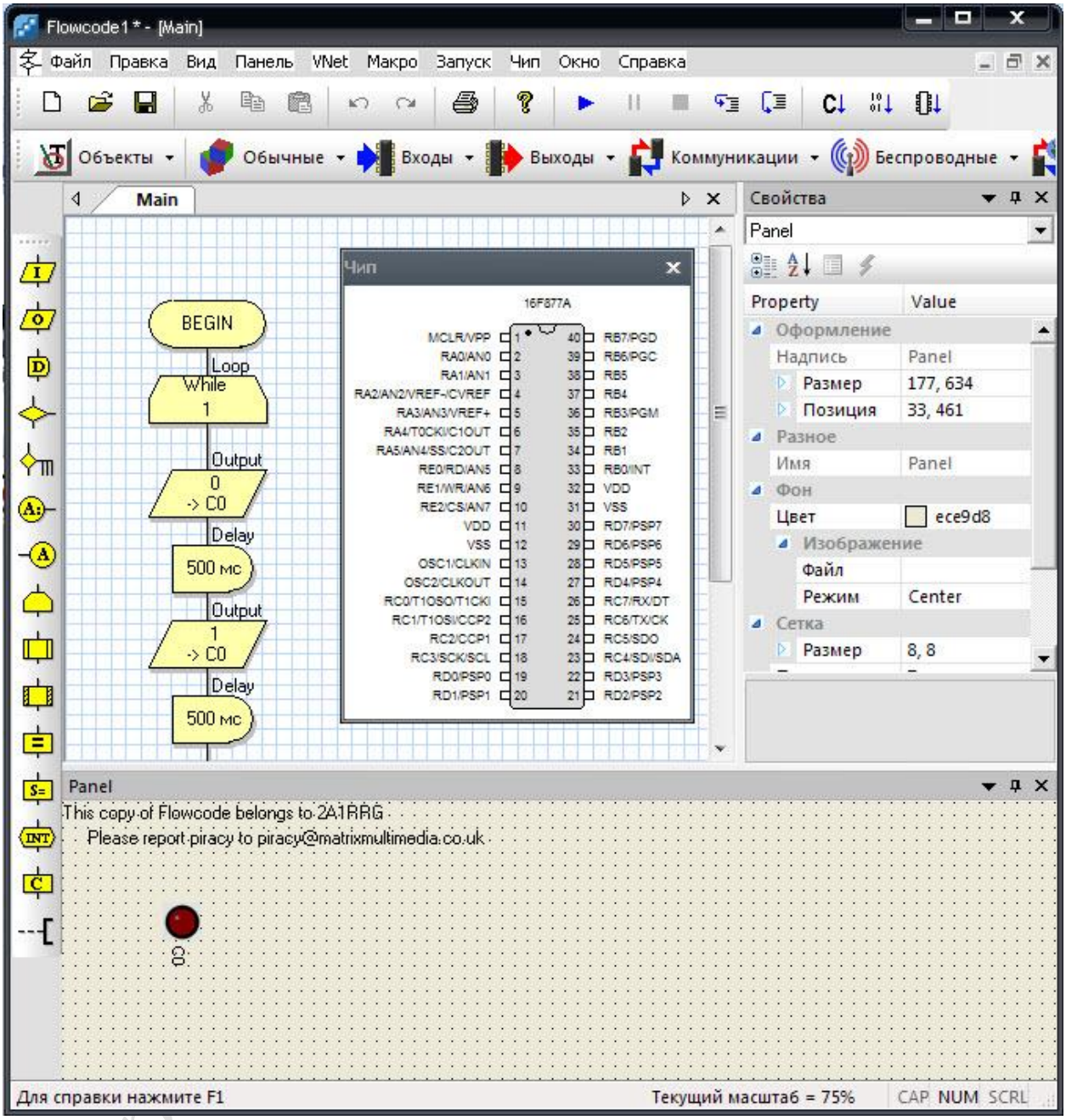

Рис. 1. Рабочая область Flowcode

Так же в рабочей области показано схематическое изображение самого микроконтроллера со всеми его выводами. Оно не представляется существенно информативным и во всех описываемых ниже проектах будет отключено.

Итак, переходим к рассмотрению главного меню программы. Пункт **Файл** совершенно типичен. Это меню позволяет открывать, закрывать и создавать проекты Flowcode. Проект представляет собой единственный файл с произвольным именем и расширением .fcf.

Меню **Правка** содержит стандартные команды **Копировать, Вырезать, Вставить**. Следует отметить, что эти операции распространяются и на элементы, размещенные на диаграмме. Здесь же присутствует пункт **Переменные**, с помощью которого можно создавать, удалять и переименовывать переменные, используемые в проекте.

Меню **Вид** настраивает содержимое рабочей области. Например, оно позволяет отменить показ схемы микроконтроллера. Все здесь просто и понятно. Отметим только, что с помощью пункта Масштаб можно масштабировать размер объекта. В частности, это полезно, когда в диаграмме накопилось много элементов. Если же диаграмму необходимо увидеть целиком, то это можно сделать автоматически, выбрав команду **Вписать в страницу**.

Переходим к меню **Макрос**, – это подпрограмма, реализующая какой-то набор действий. Макросы могут быть как разработанными пользователем, так и системными (компонентными). Последние представляют собой процедуры или функции, обслуживающие эмулируемые и реальные устройства. Например, компонентным макросом для ЖК-дисплея является функция печати числа или символа.

Новый пользовательский макрос создается с помощью команды **Новый**, что будет проиллюстрировано в дальнейших примерах. Команда **Показать** открывает список макросов данного проекта. Выбранный в нем макрос отображается в рабочей области. Поскольку макрос является подпрограммой, он отображается в виде вторичной диаграммы, аналогичную по структуре основной.

Меню **Макрос** содержит две важных команды: **Импорт** и **Экспорт**. Макрос, созданный в одном проекте можно использовать в другом. Для этого он экспортируется в виде файла с расширением .fcm. В целевом проекте такой файл необходимо импортировать.

Следующий пункт меню, **Запуск**, служит для запуска проекта на выполнение в режиме эмуляции. Первая его команда просто запускает проект. Для этой же цели служит клавиша <F5>. Полезен также режим пошагового исполнения, активизируемый по нажатию клавиши <F8>. Эмуляцию можно приостановить и завершить.

Пункт меню **Чип → Конфигурация** позволяет выбрать конкретный микроконтроллер, для которого создается проект, выбрать тип осциллятора. Кнопка **Switch To Expert Config Screen** дает возможность сконфигурировать МК весьма наглядным способом.

Также в меню **Чип** можно перевести блок схему в программу на языке С, для этого есть команда **Чип → Компиляция в С**. Изучать С-программы крайне полезно. Пользователям, владеющим программированием на С,

просмотр кода поможет понять алгоритм решения задачи. Между прочим, опытный программист заметит, что С-код проекта весьма примитивен. Это – расплата за удобства визуального программирования сверхвысокого уровня.

Команду **Компиляция в Chip** пропустим, поскольку для ее выполнения требуется «родной» для Flowcode программатор. Для создания прошивки нужно воспользоваться командой **Компиляция в Hex**, если в проекте все правильно, то в результате последней компиляции создается файл с именем, совпадающим с именем проекта, с расширением .hex.

Меню **Windows** традиционно. Оно позволяет манипулировать с окнами проекта. Наконец, меню **Help**, как всегда, помогает решать проблемы, возникающие при работе с Flowcode.

Теперь переходим к примерами реализации различных алгоритмов. Мы пойдем обычным путем от простого к сложному, поэтому последовательность рассмотрения примеров играет существенную роль.

Первый Blink

Слово «Blink» в нашем случае соответствует миганию светодиода. Говоря иначе, цель нашего первого проекта – программа, которая заставляет один светодиод переключаться с длительностью состояния, допустим, полсекунды.

Поскольку это, действительно, - наш «первый блин», то, чтобы он не оказался комом, процесс создания и модификации проекта опишем и проиллюстрируем графически более подробно. В последующих разделах стандартные аспекты разработки мы будем по возможности опускать.

Итак, при запуске программы Flowcode открывается основное окно приложения, на фоне которого сразу же отображается приглашение создать новый проект или открыть существующий (рис. 1.1).

Естественно, для первого проекта следует согласиться с выбранным по умолчанию вариантом **Создать новую блок-схему FlowCode**, и нажать кнопку **ОК**.

Далее будет предложено выбрать тип микроконтроллера, с которым планируется работать (рис. 1.2).

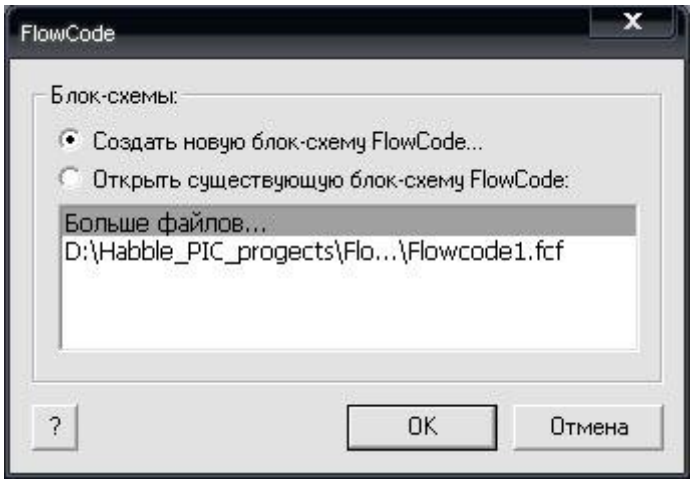

Рис.1.1. Окно, открываемое при запуске Flowcode

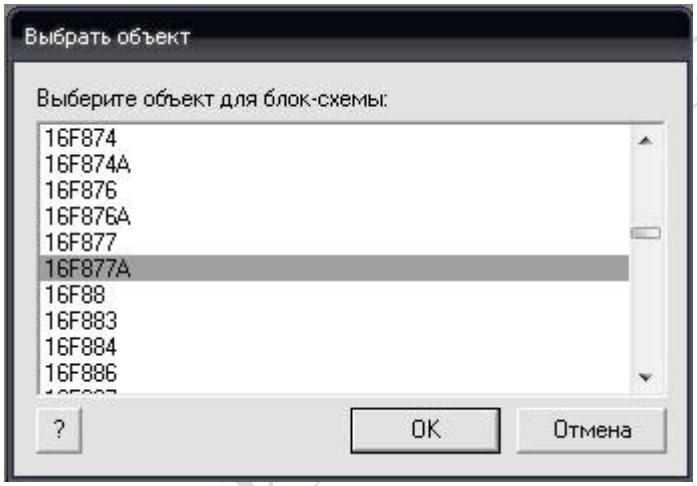

Рис. 1.2. Выбор типа микроконтроллера

В большинстве случаев мы будем программировать PIC16F877A.

На следующем этапе необходимо задать тактовую частоту осциллятора микроконтроллера. Для этого в главном меню программы выбираем команду **Вид → Настройки проекта** (4000000 Hz, рис. 1.3).

Далее необходимо выбрать тип используемого генератора. Для этого в главном меню выбираем команду **Чип → Конфигурация,** выбираем тип осциллятора – кварц (**XTAL)**). Нажав **Switch To Expert Config Screen** задаем слово конфигурации (пропишем 0x3F71 в правом нижнем углу (рис.1.4)), после этого можно сохранить проект под названием *Blink.*

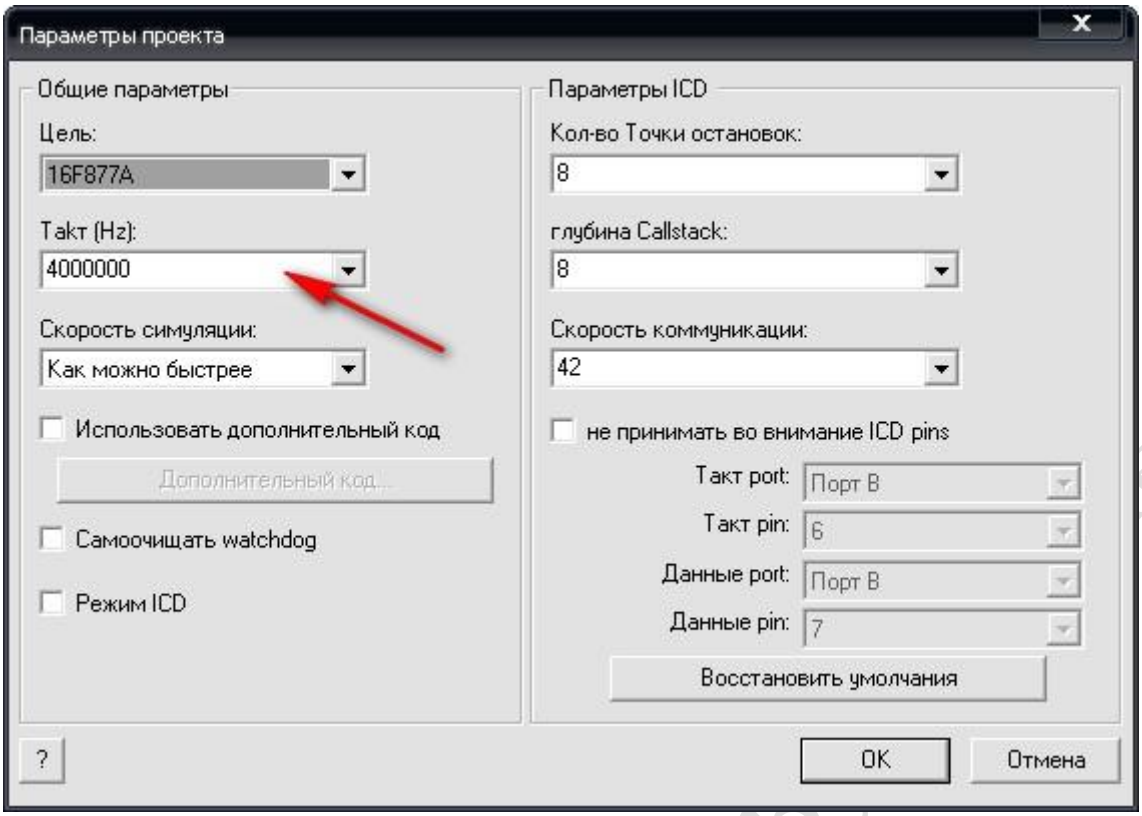

Рис. 1.3. Диалоговое окно **Параметры проекта**

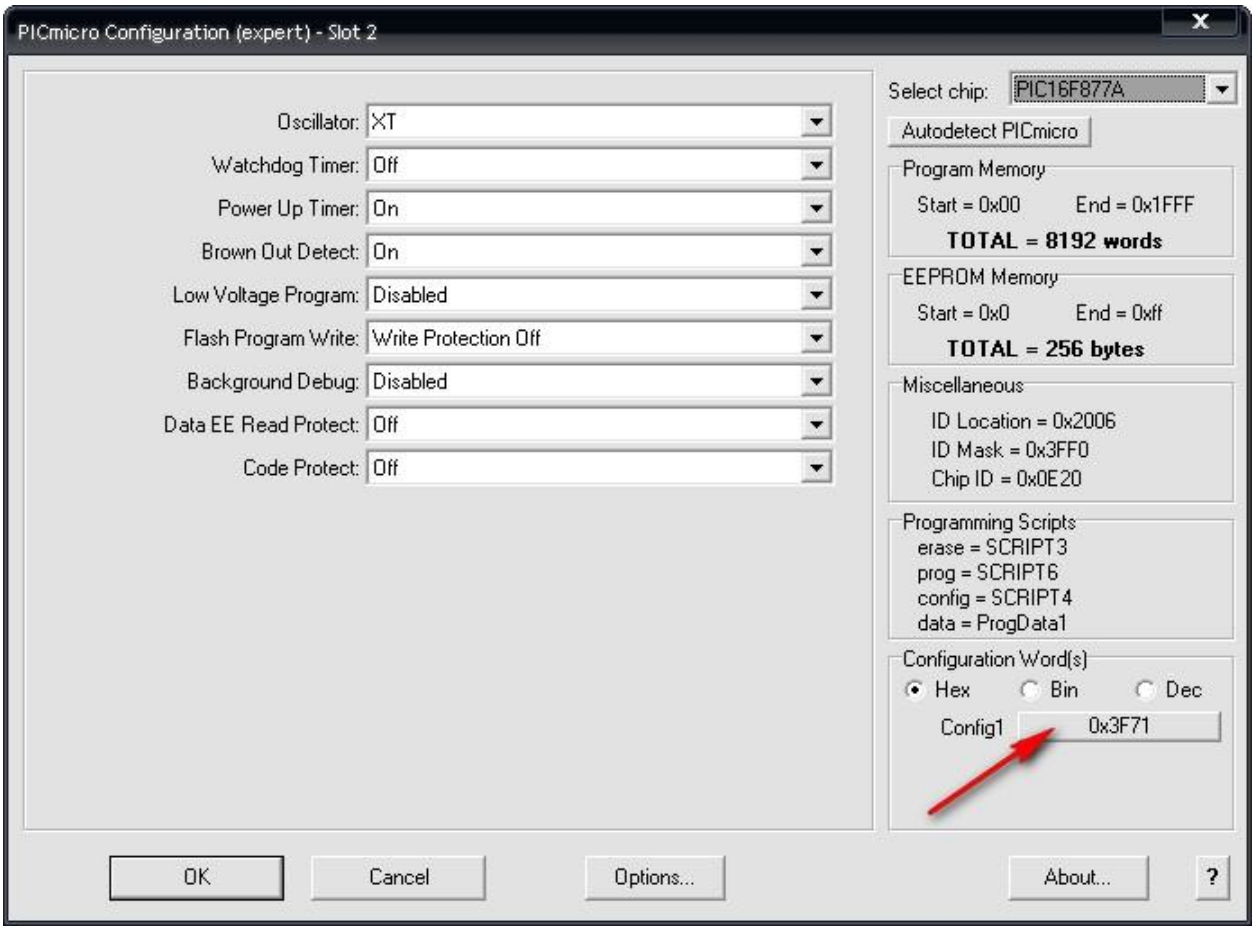

Рис.1.4. Диалоговое окно конфигурации МК

Теперь для проверки работы будущего проекта в режиме эмуляции мы должны разместить в рабочем окне светодиод, которому предстоит мигать. Для этого в Flowcode существует стандартный компонент LED. Выбрать его можно на панели инструментов **Обычные → LED** либо **Выходы → LED** (Рис.1.5 и 1.6).

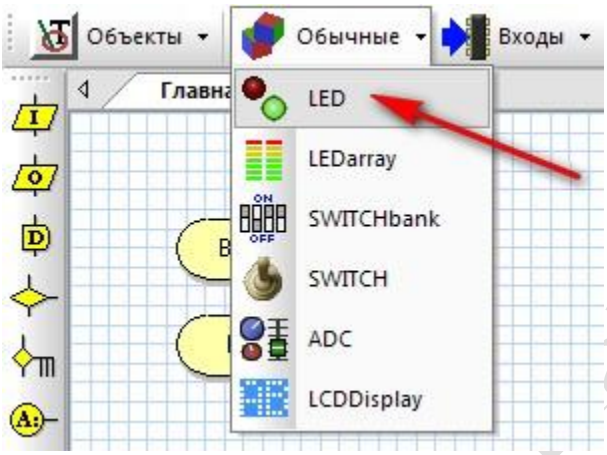

Рис.1.5. Выбор компонента LED

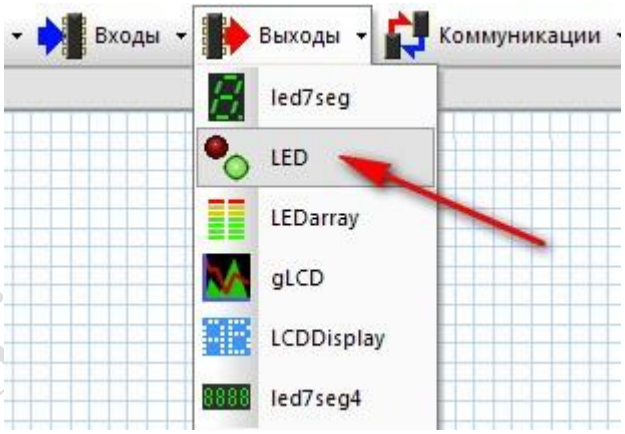

Рис.1.6. Выбор компонента LED

Светодиод появляется внизу на панели отображения компонентов. Теперь нам нужно его подключить к какому либо выводу МК, например к нулевому биту порта С, для этого выделим светодиод, нажав по нему левой кнопкой мышки (рис.1.7).

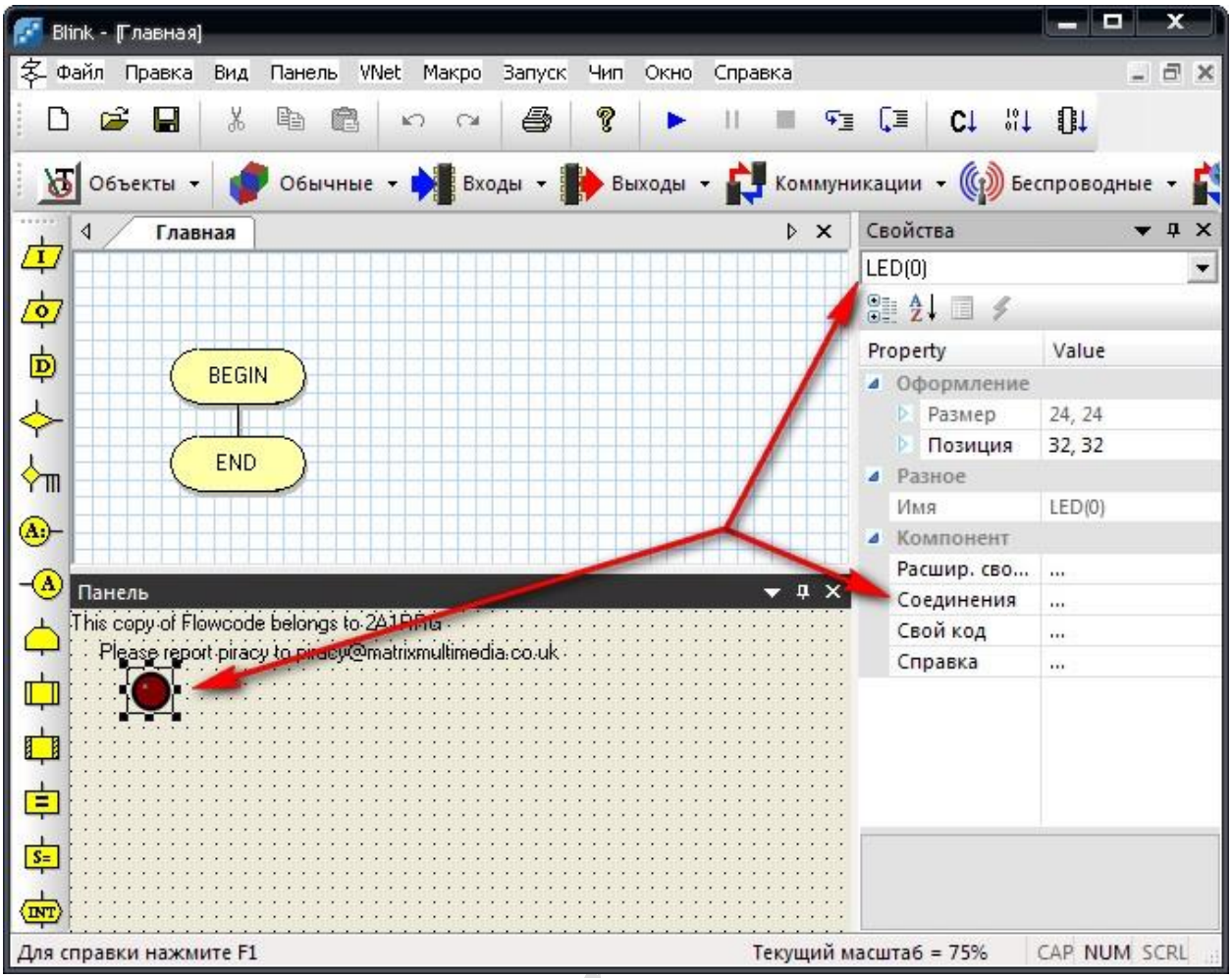

Рис.1.7. Выделение компонента и его свойства

В окне свойств можно задавать цвет, форму, количество светодиодов и т.д. (рис.1.8).

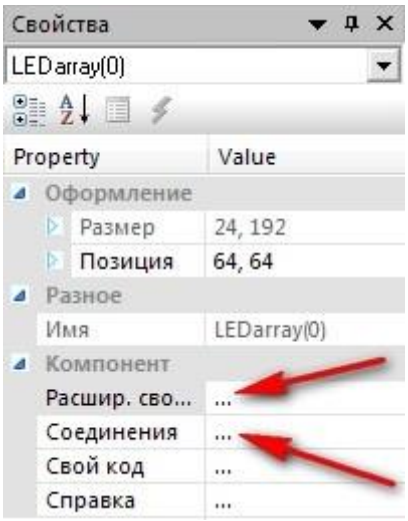

Рис.1.8. Свойства LED

В первую очередь нас интересует соединение с МК, поэтому для вызова этих свойств щелкаем левой кнопкой мышки по троеточию напротив

Соединения (рис.1.8 нижняя стрелка). В результате у нас появится окно (рис.1.9) где можно настраивать подключение к МК.

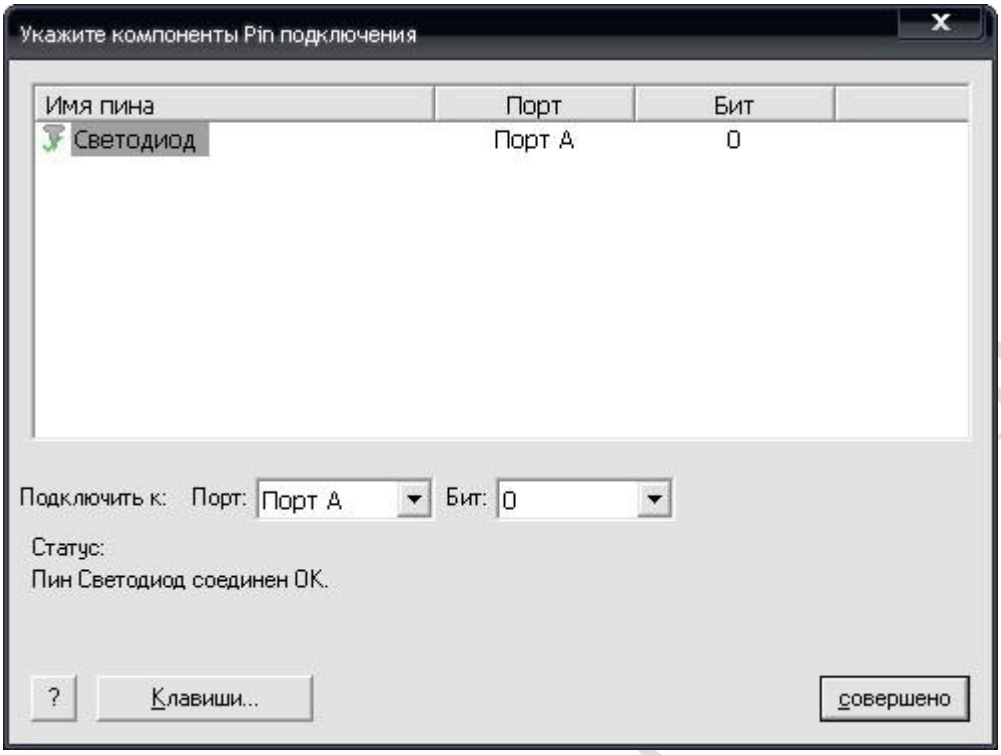

Рис.1.9. Конфигурация соединения для LED

Внизу окна мы видим строку – **Подключить к**. Далее порт и номер бита. Как мы договаривались, подключим данный светодиод к нулевому биту порта С (рис.1.10.).

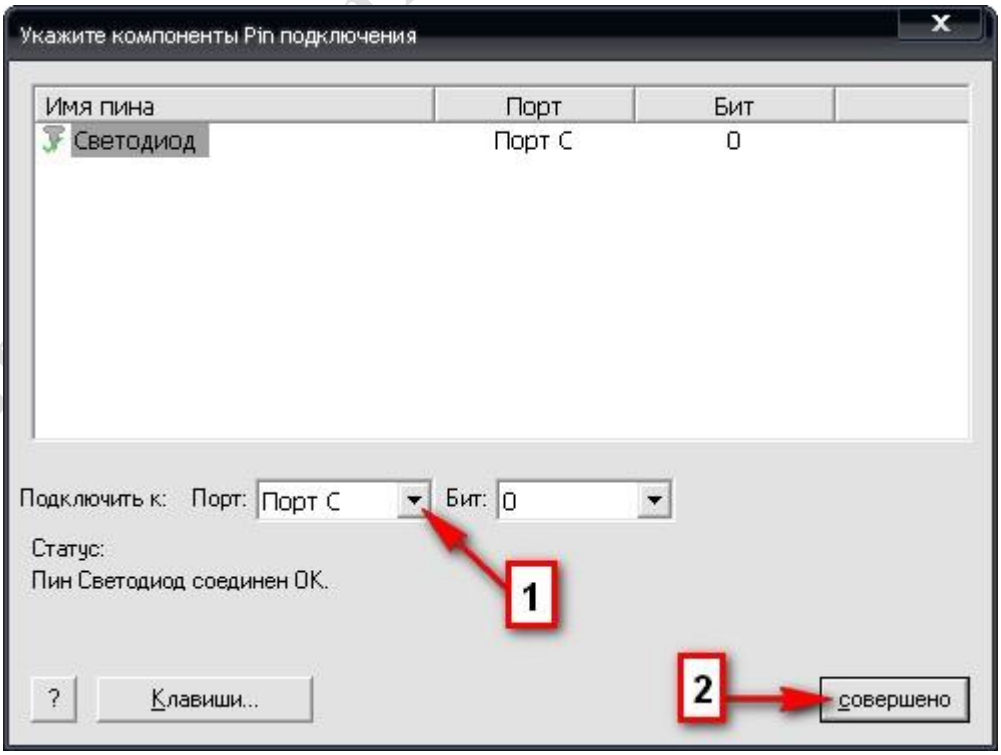

Рис. 1.10. Выполнение необходимых действий для подключения

На этом предварительная работа по организации проекта завершена. Мы получили рабочую область, соответствующую рис.1.11. Далее мы будем, в основном, иметь дело с инструментами, размещенными в вертикальной линейке вдоль левого края окна Flowcode.

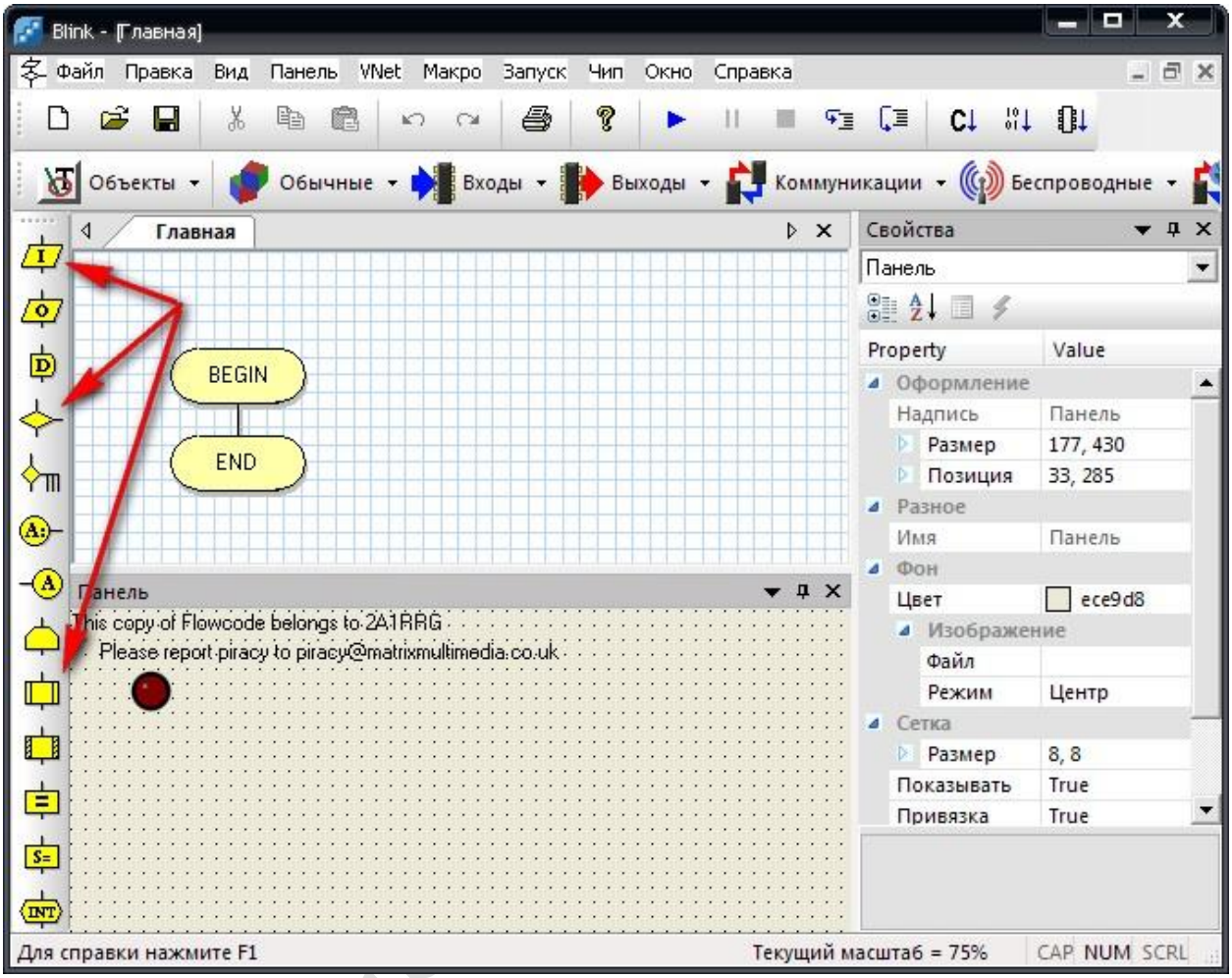

Рис.1.11. Рабочая область и ее инструменты для графического программирования

Прежде всего, необходимо создать бесконечный цикл работы МК. В отличие от программ для ПК, микроконтроллерные системы, как правило, работают в бесконечном цикле, который прерывается только при аварийных ситуациях или по принудительному выключению. Для организации цикла необходимо перетащить мышью с линейки инструментов на диаграмму BEGIN-END элемент **Цикл** (восьмой сверху, смотрите рис.1.12). Отпустить кнопку мыши можно в тот момент, когда появится желтая стрелка.

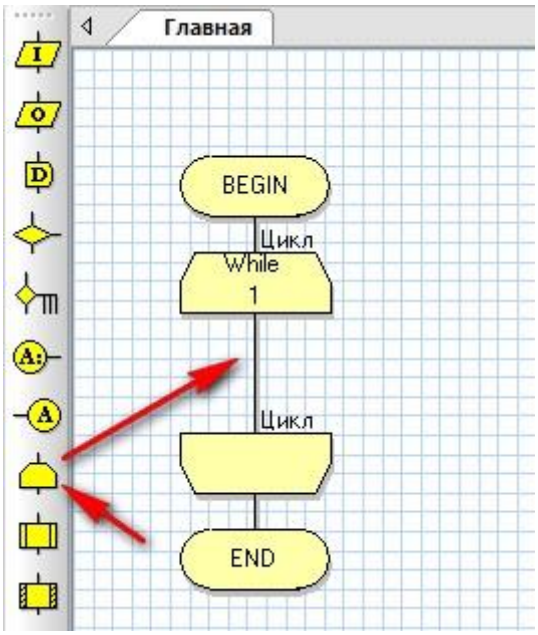

Рис. 1.12. Выбор элемента **Цикл**

Дважды щелкните мышью на первом (верхнем) элементу **Цикл**. В результате откроется диалоговое окно, показанное на рис.1.13.

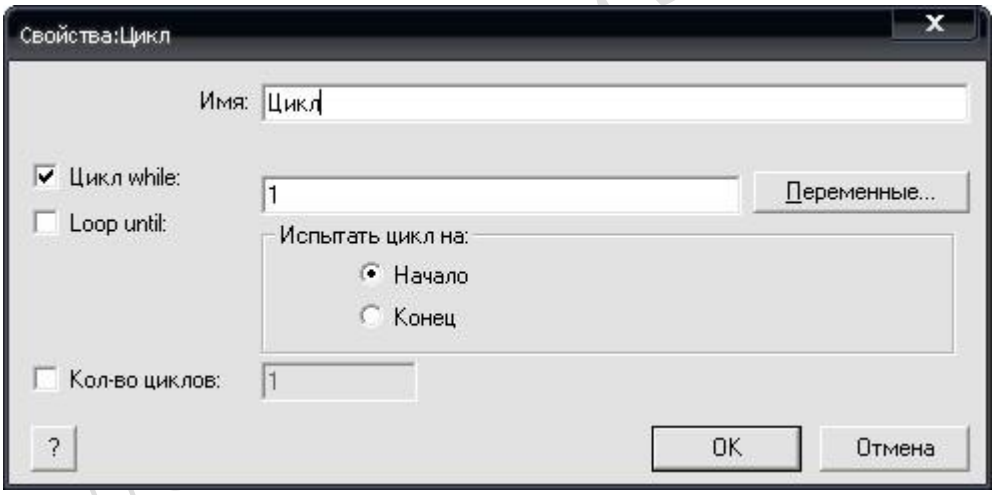

Рис.1.13. Свойства элемента **Цикл**

Здесь изменять ничего не нужно, однако следует разобраться в том, что мы видим. В поле **Имя** задается имя, которое будет отображаться на диаграмме. Если бы циклов было несколько, то имело бы смысл дать им различные имена. На втором поле остановимся поподробнее.

Установленный флажок **Цикл while** говорит о том, что цикл будет выполняться до тех пор, пока выражение в расположенном справа поле возвращает значение ИСТИНА. Здесь, например, можно указать условие **x<1.** В общем случае такое выражение возвращает значение ИСТИНА, если оно не равно нулю, или ЛОЖЬ в случае равенства нулю. Такое толкование поля **Цикл while** наиболее близко к концепциям языка программирования С. В

нашем случае здесь всегда указывается **1**, т.е. цикл выполняется всегда (бесконечно), что нам и требуется.

Переключатели **Начало** и **Конец** указывают проекту момент проверки условия: перед выполнением операций в цикле или после. В нашем случае это, естественно, значения не имеет, однако в других примерах выбор одного из переключателей может играть большую роль. Заметим, что предварительная проверка и постпроверка реализованы в языке С различными циклами: while и do-while.

Если необходимо задать конкретное количество выполнений цикла (как, например, в цикле for языка С), то флажок **Цикл while** следует сбросить, а вместо него установить флажок **Кол-во циклов** и ввести интересующее число в расположенном справа поле.

Наступил момент определить операции, которые будут выполняться в нашем бесконечном цикле. Напомним, что мы хотим, чтобы в цикле полсекунды светодиод был отключен, а следующие полсекунды – включен. Для начала перетащим на диаграмму с вертикальной линейки инструментов элемент **Выход** (второй сверху). Действие этого элемента по умолчанию заключается в подаче на порт А некоторого значения.

Нас такой вариант не устраивает. Мы хотим, чтобы выходным портом был С, а не А. Кроме того, нам необходимо работать не со всем портом, а только с его младшим (нулевым) выводом. Дважды щелкните мышью на элементе **Выход** в диаграмме. В результате откроется окно, представленное на рис. 1.14.

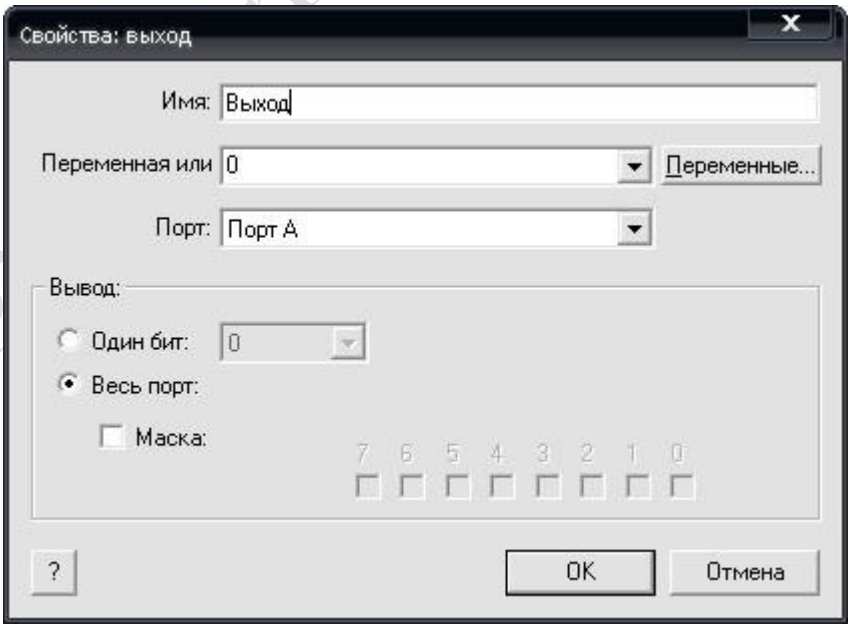

Рис. 1.14. Свойства порта А

В поле **Переменная или** значение введите **0** (выходное значение порта). Сам порт задаем в следующем поле: **Порт С**. Далее выберите

переключатель **Один бит** и укажите в расположенном справа поле значение **0** (рис.1.15).

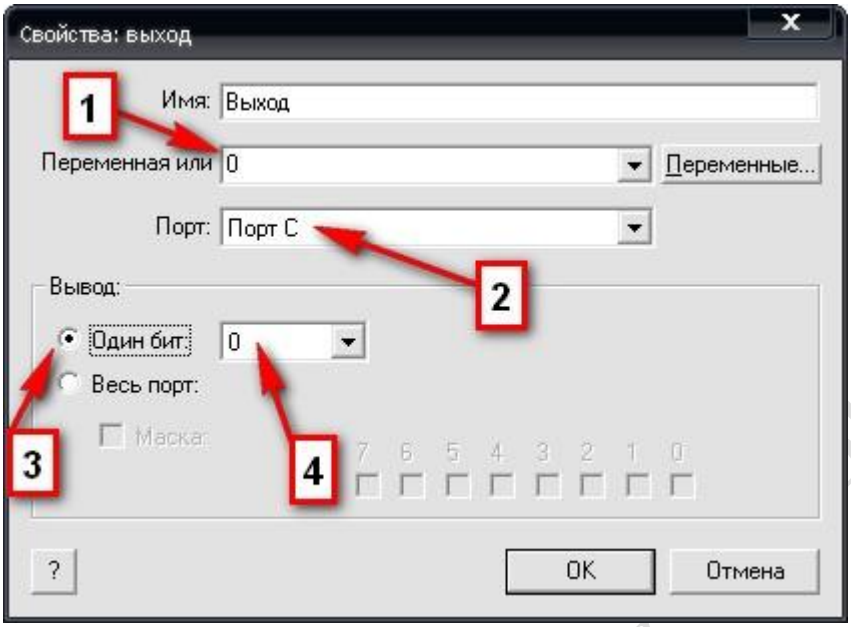

Рис.1.15. Настройка элемента **Выход**

Ниже элемента **Выход** разместите в диаграмме элемент **Задержка** (третий сверху в линейке инструментов), а после него – еще один элемент **Выход** и еще один элемент **Задержка**. Все эти элементы должны находиться внутри организованного цикла. Теперь рабочая область соответствует рис.1.16.

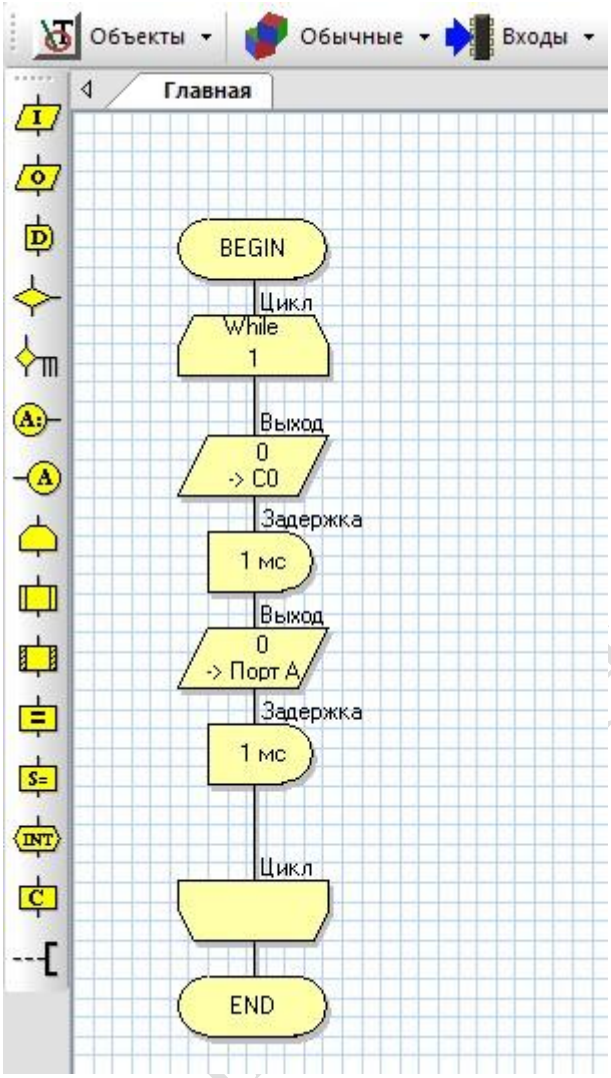

Рис.1.16. Диаграмма с двумя элементами **Выход** и двумя элементами **Задержка**

На очереди разговор о задержках. Они создают в работе программы паузу заданной длительности. Дважды щелкните мышью на первом из элементов **Задержка** в диаграмме. В результате откроется соответствующее окно свойств (рис. 1.17).

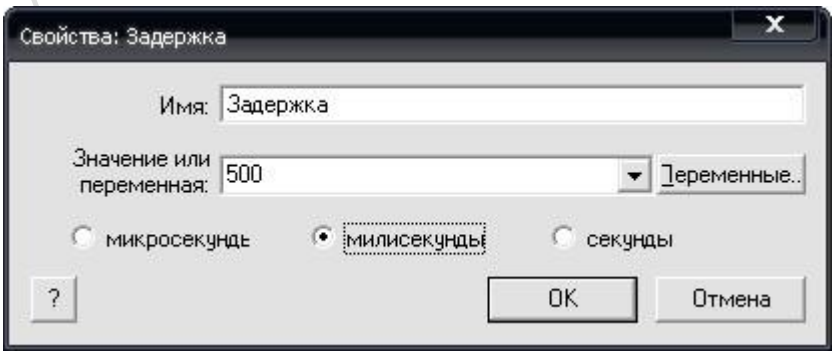

Рис.1.17. Окно свойств элемента **Задержка**

Здесь в принципе, все понятно. Задается единица измерения длительности. В нашем случае укажем 500 миллисекунд (полсекунды). Такие

же установки необходимо сделать и для второго элемента **Задержка**. Для второго элемента **Выход** следует определить те же свойства, что и для первого, однако выводимое значение на этот раз будет не **0**, а **1** (светодиод включен, смотрите рис. 1.18).

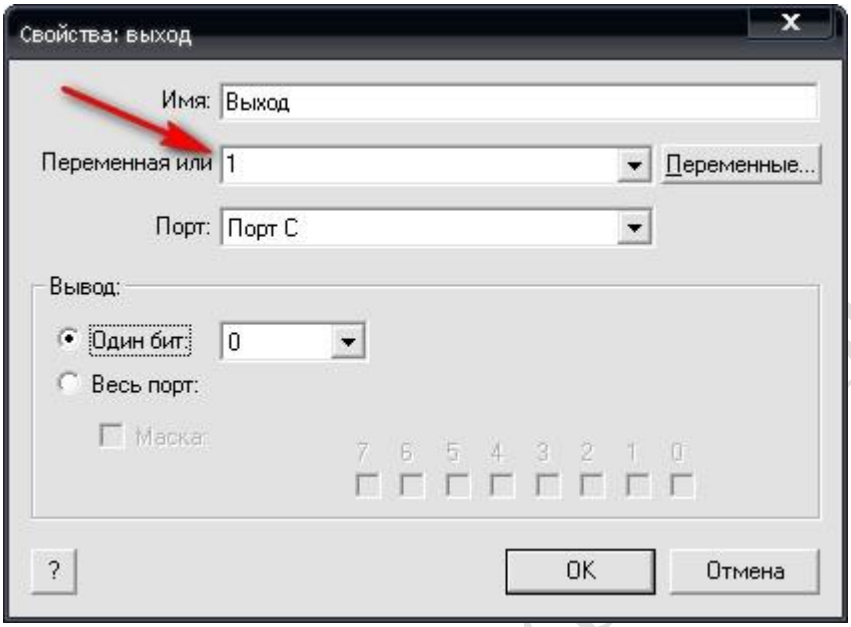

Рис.1.18. Установка нулевого бита порта С в логическую 1

Теперь наш проект готов к выполнению в режиме симуляции Flowcode. В главном меню выберите команду **Запуск → Пуск/Продолжить** или просто нажмите клавишу <F5>. Если все было сделано правильно, то светодиод начинает мигать с заданной частотой. Прекратить выполнение можно с помощью команды меню **Запуск → Стоп**, или комбинацией клавиш <Shift+F5>. Если запуск симуляции начать не с клавиши <F5>, а с <F8>, то мы получим пошаговое исполнение нашей графической конструкции, каждое нажатие <F8> будет исполнять отдельный блок нашего алгоритма.

И так, цель достигнута! Однако, напомним, что мы всего лишь реализовали эмуляцию на ПК, но можно заставить работать реальный микроконтроллер. Для этого нужен программатор, но так как их существует великое множество, то какой программатор выбрать и как им пользоваться в данной статье не рассматривается. При желании данная информация легко находится в Google. Остановимся только на процессе создания прошивки. Для этого (как мы уже говорили в начале статьи) нужно воспользоваться командой **Чип → Компиляция в Hex**, если в проекте все правильно, то в результате последней компиляции создается файл с именем, совпадающим с именем проекта, с расширением .hex. Этот файл и нужно будет «зашивать» в микроконтроллер.

Что ж, можем поздравить себя с реализацией первой, пусть и крайне простой, но работающей программой.

Двоичный счет

Действительно, предыдущий проект очень прост, как с точки зрения решаемой задачи, так и по реализации. Немного усложним задачу. Заставим МК отображать на светодиодах последовательно во времени двоичные числа от 0000 до 1111 с бесконечным повторением последовательностей. Примем, что в каждом из четырех двоичных разрядов 0 соответствует выключенному светодиоду, а 1 – включенному. Экспозиция высвечивания каждого числа, по-прежнему, пусть составляет полсекунды.

Создадите проект так же, как вы это делали для первой программы *Blink*, только без добавления светодиода, чтобы вид графика стал как на рис. 1.12 или отредактируйте старый и сохраните под названием *Bin_count*. Теперь нам нужно добавить четыре светодиода, для этого добавим «линейку» из восьми светодиодов как на рис. 1.19, потом зайдя в *Расширенные свойства* этой «линейки» выбираем количество светодиодов – 4 (рис. 1.20).

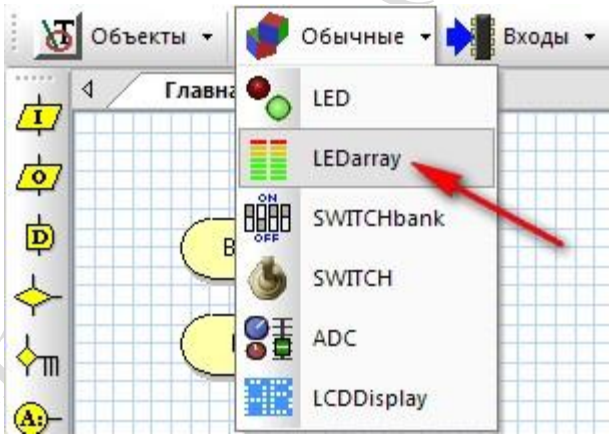

Рис.1.19. Добавление «линейки» из восьми светодиодов

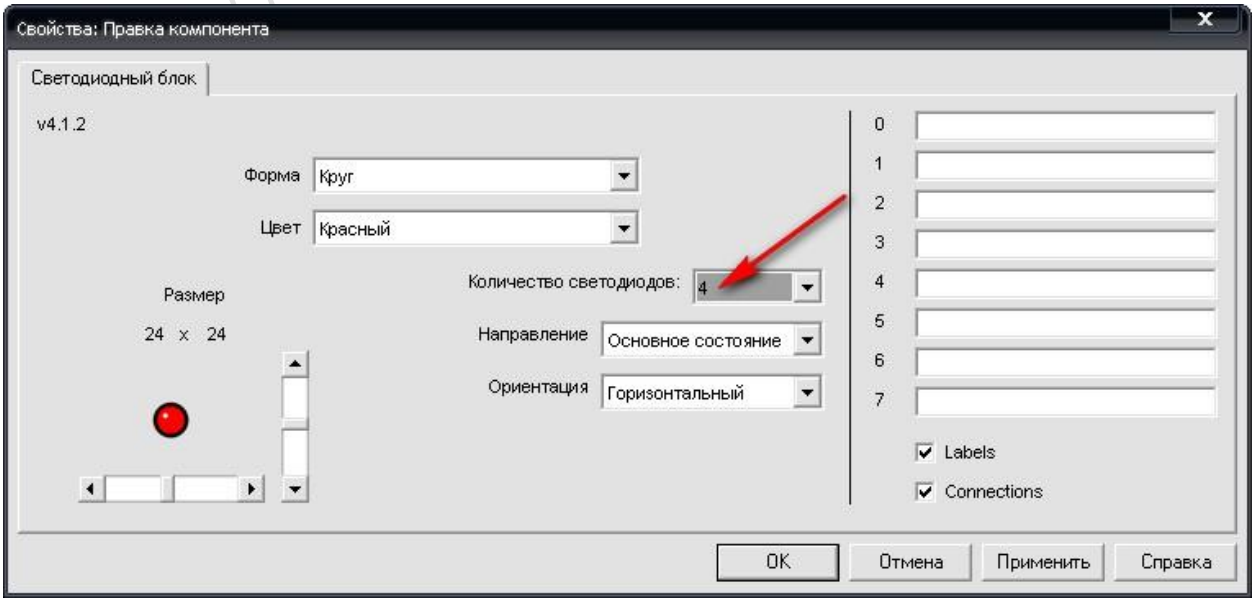

Рис.1.20. Выбор количества светодиодов

Еще осталось подключить их к порту С, делается это в свойствах «линейки» *Соединения* (рис.1.21). После этого внизу каждый светодиод будет соответственно подписан.

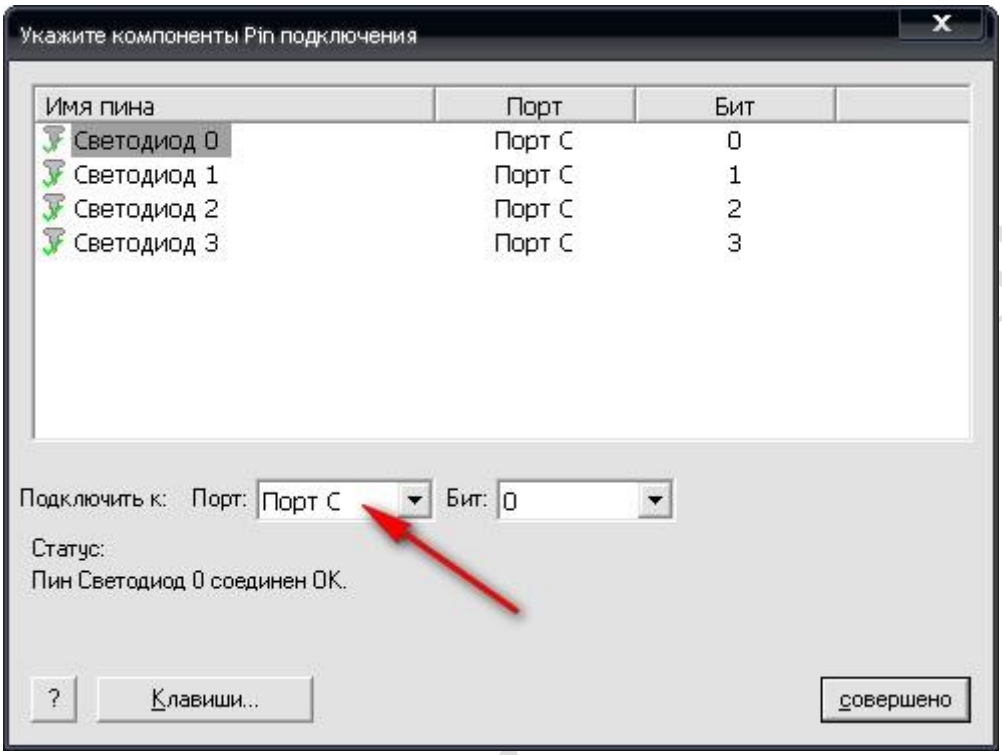

Рис.1.21. Выбор порта С

Теперь во внешний бесконечный цикл необходимо вложить еще один цикл: внутренний. Поместим его сразу после открывающего заголовка внешнего цикла. Дальше во внутренний цикл поместим элементы **Выход** и **Задержка** соответственно.

А сейчас наступил важный момент: создание новой (в данном случае первой) переменной. Выберите в главном меню команду **Правка → Переменные**. Как видим, пока что у нас нет никаких переменных. Нажмите кнопку **Добавить новую**. В открывшемся диалоговом окне **Создать новую переменную** введите в поле **Имя новой переменной – i** (рис.1.22).

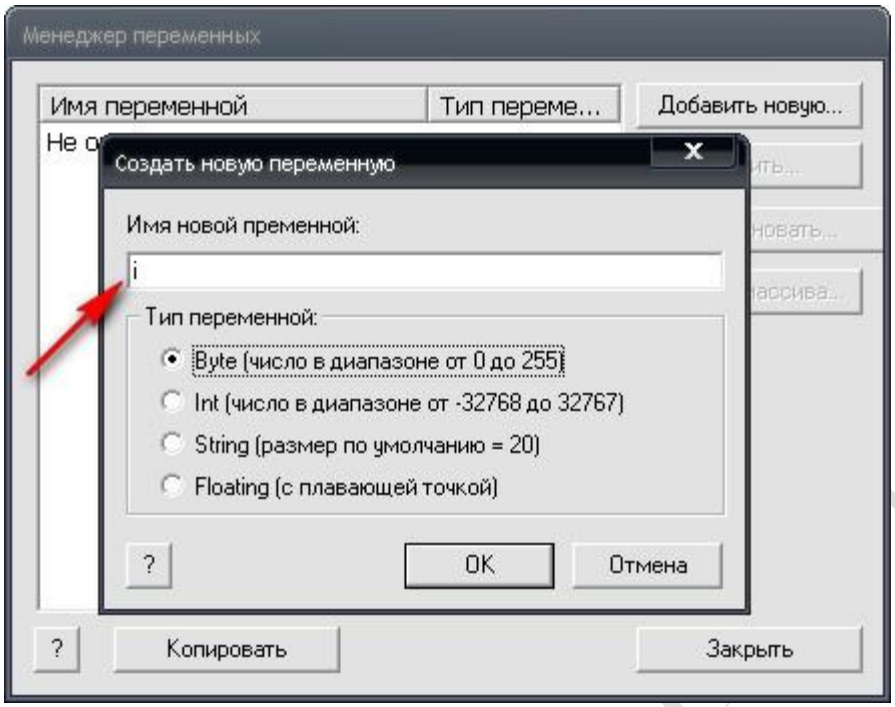

Рис.1.22. Создание новой переменной **i**

Как видим, Flowcode не блещет разнообразием типов данных. Их всего четыре.

Поскольку мы будем оперировать двоичными числами от 0 до 1111, что соответствует десятичным значениям в диапазоне от 0 до 15, нам достаточно одного байта. Таким образом, соглашаемся с предложенным по умолчанию байтовым типом.

Сразу после открытия внешнего цикла необходимо инициализировать нашу новую переменную, присвоив ей значение **0**. Для этого в указанную точку диаграммы перетащите мышью с вертикальной линейки одиннадцатый сверху элемент **Вычисление**. Это очень важный инструмент. С его помощью можно организовывать последовательность вычислений, что и отражено в его названии. Двойной щелчок мышью на элементе **Вычисление** открывает окно его свойств. В поле **Вычисления** необходимо вписать простую операцию **i = 0**, что следует понимать так: переменной **i** присвоить значение **0**. Соответствующее окно представлено на рис.1.23.

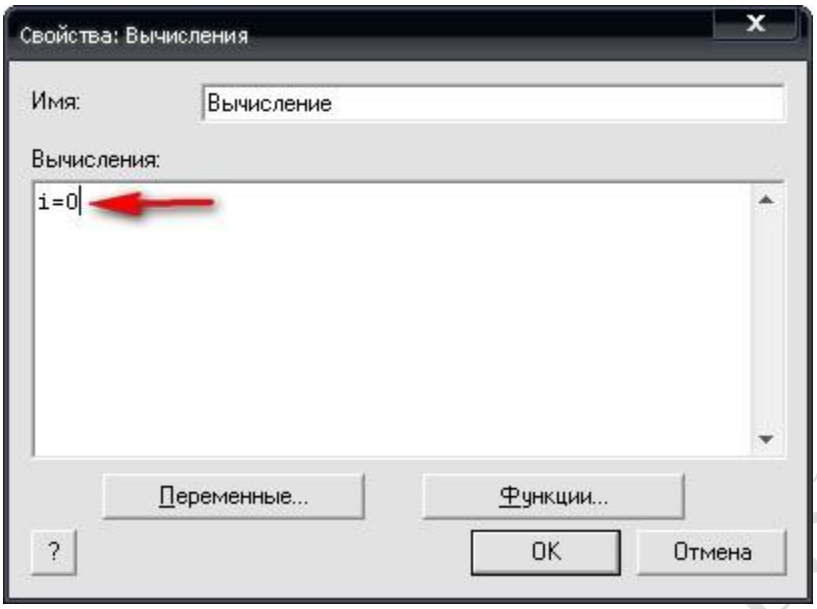

Рис.1.23. Свойства элемента **Вычисление**

Сейчас необходимо изменить заголовок вложенного цикла. Пока что он тоже выполняется бесконечно, нам же требуется, чтобы он выполнялся до тех пор, пока **i** меньше **16**. Сделать это не составит труда. Открываем окно цикла и в поле **Цикл while** вводим **i<16**.

Естественно, в цикле необходимо обеспечить инкрементирование (увеличение на единицу) переменной **i**. Это должно происходить после задержки. Вставьте в соответствующей точке еще один элемент **Вычисление** и задайте для него действие **i = i +1**. Теперь нужно настроить элементы **Выход** и **Задержку**. В элементе **Выход** нужно все сделать как на рис. 1.24.

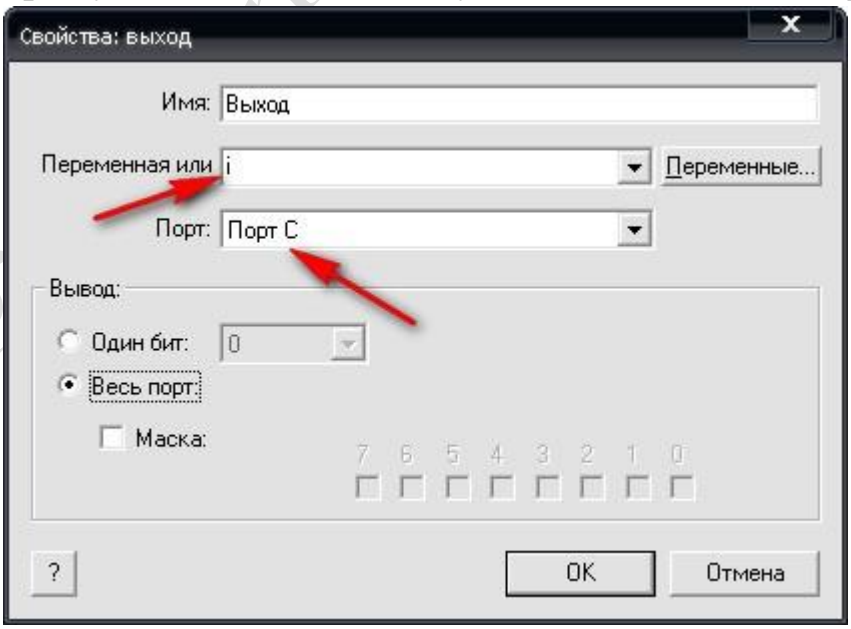

Рис.1.24. Настройка элемента **Выход**

В элементе **Задержка** так же выставить 500 миллисекунд. Вот и все, что требуется сделать в новом проекте. Теперь его вид в рабочей области должен соответствовать рис. 1.25.

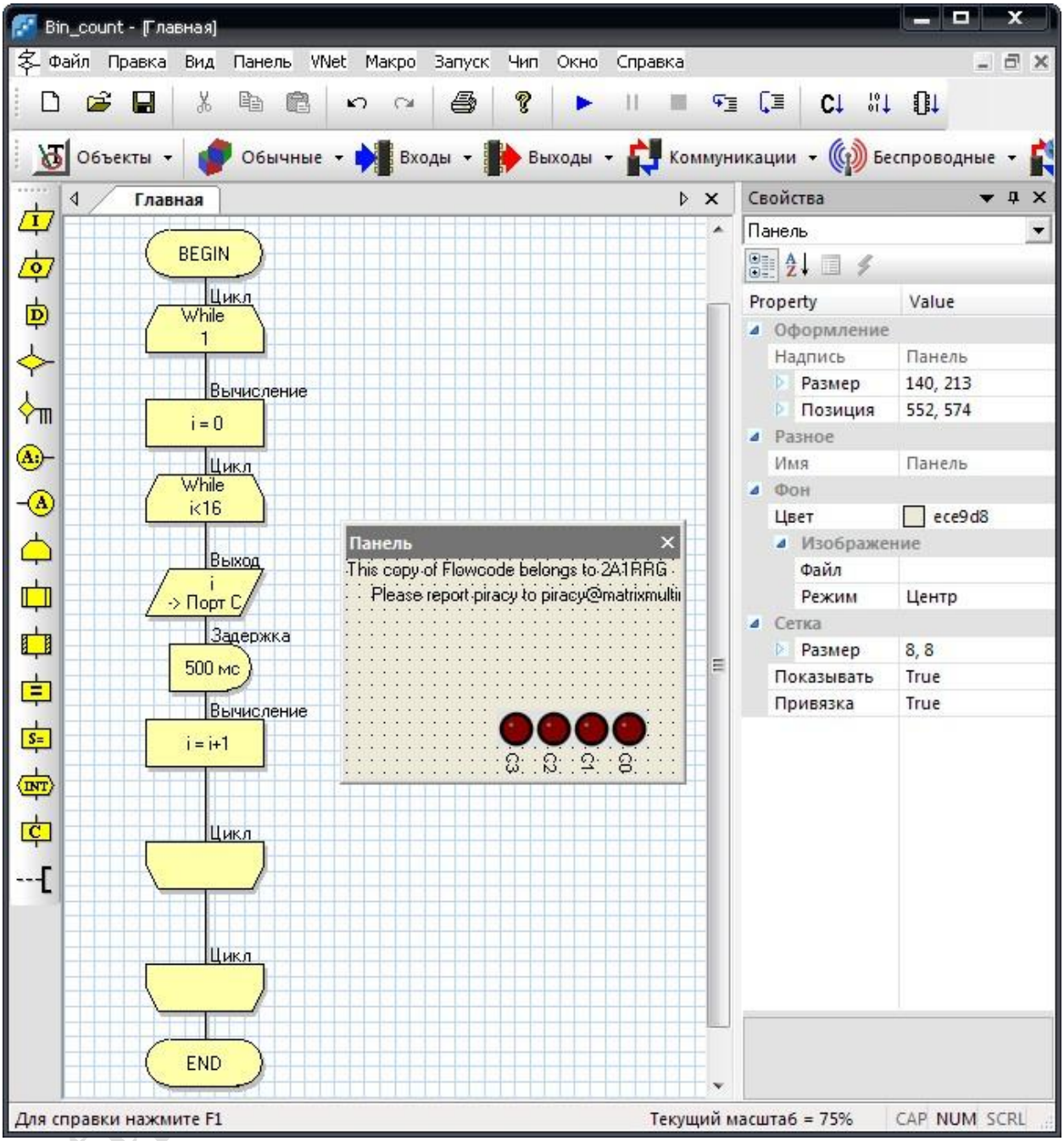

Рис.1.25. Диаграмма второго проекта

Запустив проект на симуляцию, убеждаемся, что все работает, как требовалось.

Макросы, подпрограммы, процедуры, функции

Все эти названия, перечисленные в заголовке раздела, означают примерно одно и то же: запрограммированные алгоритмы решения каких-то задач, которые можно многократно выполнять в главной программе или даже использовать в отдельных самостоятельных программах. В наиболее обобщенном виде такие конструкции позиционируются как функции в языке С. В Flowcode употребляется термин «макросы», однако мы пока что будем пользоваться именно определением «функция».

Если функция предназначена для вычисления некоторого числового значения, то говорят, что она возвращает значение, типизированная, и ее тип определяется типом вычисляемого значения. В нашем случае функция (макрос) возвращает целое (int) или байтовое (byte) значение. Однако она может быть предназначена вовсе не для вычисления значений, т.е. не возвращает никакого числа. С другой стороны, функция может быть ориентирована на вычисление более чем одного значения. Тогда функция не является типизированной. Функция может иметь (или не иметь) один или несколько параметров (аргументов). Для реализации алгоритма может потребоваться один или несколько внутренних или локальных для данной функции переменных, которые видимы только внутри самой функции. Все эти альтернативные возможности предусмотрены в Flowcode при организации макросов.

Рассмотрим математическую функцию, называемую факториалом числа. Факториал целого числа *n* – это произведение всех натуральных чисел от 1 до *n*. Факториал целого числа *n* – это произведение всех натуральных чисел от 1 до *n*. Факториал от нуля по определению равен 1. Обозначается факториал как *n!*:

$$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$$

Не трудно убедиться, что 1! = 1; 2! = 2; 3! = 6; 4! = 24; 5! = 120 и т.д.

В нашей классификации факториал является типизированной функцией целого типа. Для $n < 6$ можно обойтись и байтовым типом. Само значение *n* является параметром (аргументом) функции, а результат произведения натуральных чисел как раз и будет возвращаемым значением.

Теперь создадим проект, вычисляющий и отображающий в двоичном виде на линейке светодиодов факториал заданного числа. Процедуру вычисления оформим в виде макроса.

Создайте новый проект Flowcode, назовите его *Macro* и сохраните в заданной папке. Настройте проект (частоту генератора, тип осциллятора) так же, как и раньше. Разместите компонент из восьми светодиодов и подключите его к порту С. В диаграмму вложите бесконечный цикл.

Создайте новую переменную **n** байтового типа. С помощью инструмента **Вычисление** перед циклом присвойте переменной **n** значение **4**. Это и будет то число, для которого рассчитывается факториал. Создайте еще одну байтовую переменную **fac**. В нее, в конце концов, будет записано вычисленное значение факториала. Все эти операции нам уже знакомы. В результате рабочее поле будет соответствовать рис. 1.26.

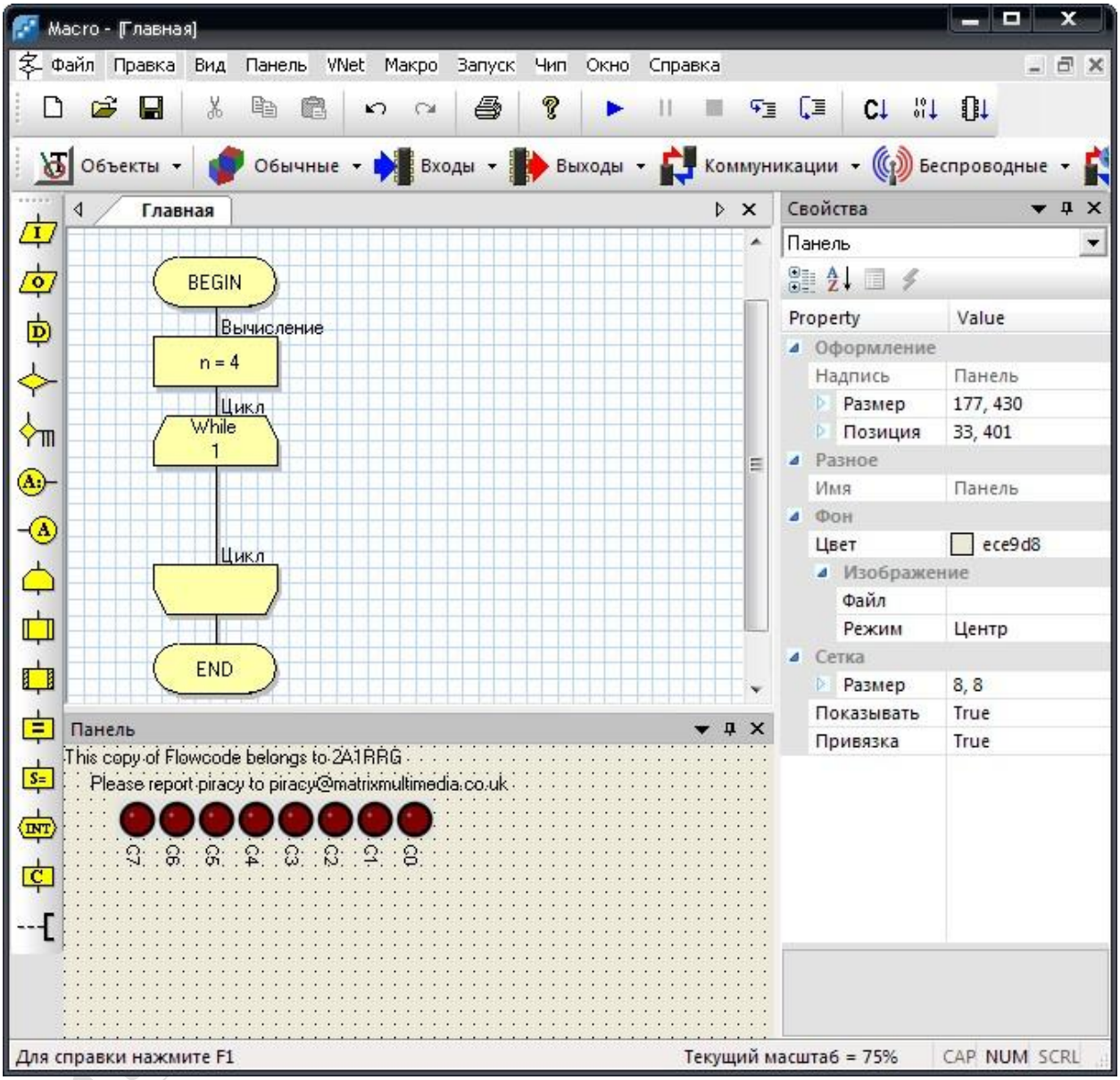

Рис.1.26. Диаграмма для вычисления факториала

Приступаем к созданию макроса. Выберите команду меню **Макро → Новый**. Открывшееся окно с введенным именем макроса в первом поле показано на рис. 1.27, а также следующие шаги, которые будут рассмотрены ниже.

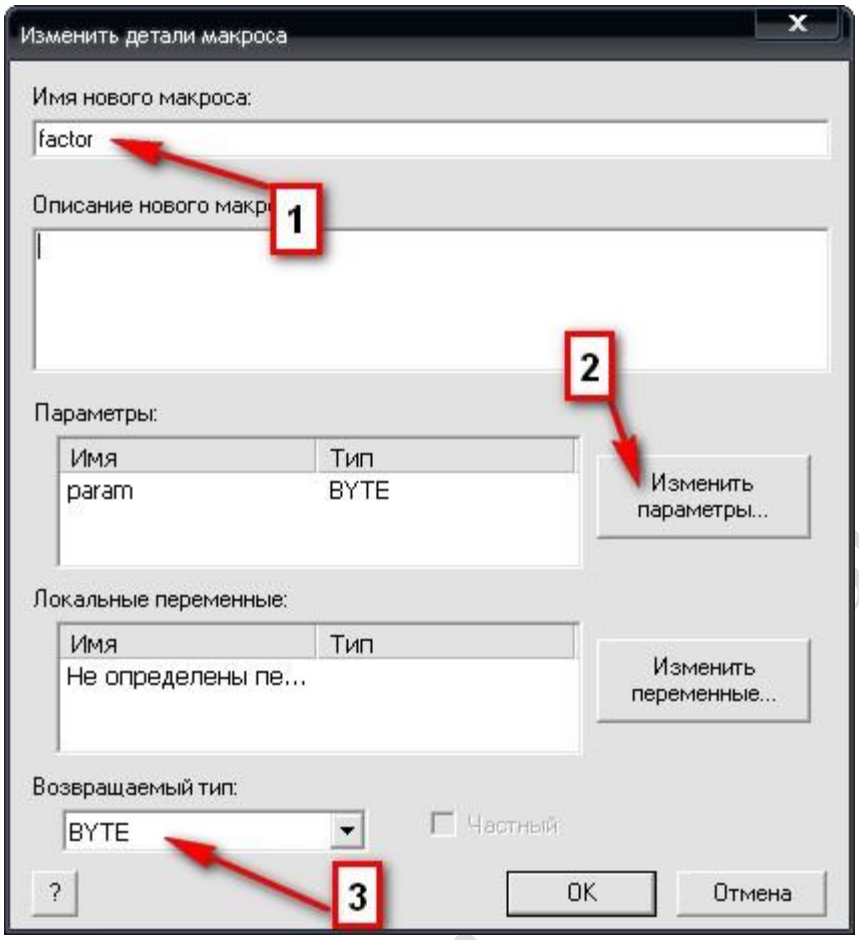

Рис.1.27. Создание нового макроса

Пока у макроса нет ни параметров, ни локальных переменных, ни возвращаемого значения. Исправим эту ситуацию. Нажмите кнопку **Изменить параметры** и добавьте один параметра байтового типа с именем **param**. Локальные переменные в нашем макросе не понадобятся, а в качестве типа возвращаемого значения выберите **BYTE**. В результате в рабочей области появится еще одна, пустая диаграмма. Это и есть подпрограмма или макрос. Перетащите мышью с вертикальной линейки инструментов внутрь цикла макрос (девятый сверху элемент). Теперь рабочая область соответствует рис.1.28.

Дважды щелкните мышью на элементе **Макрос компонента** в основной диаграмме и в открывшемся диалоговом окне (рис 1.29) укажите, что в качестве параметра (аргумента) макроса будет использоваться переменная **n**, а в качестве возвращаемого значения – переменная **fac**.

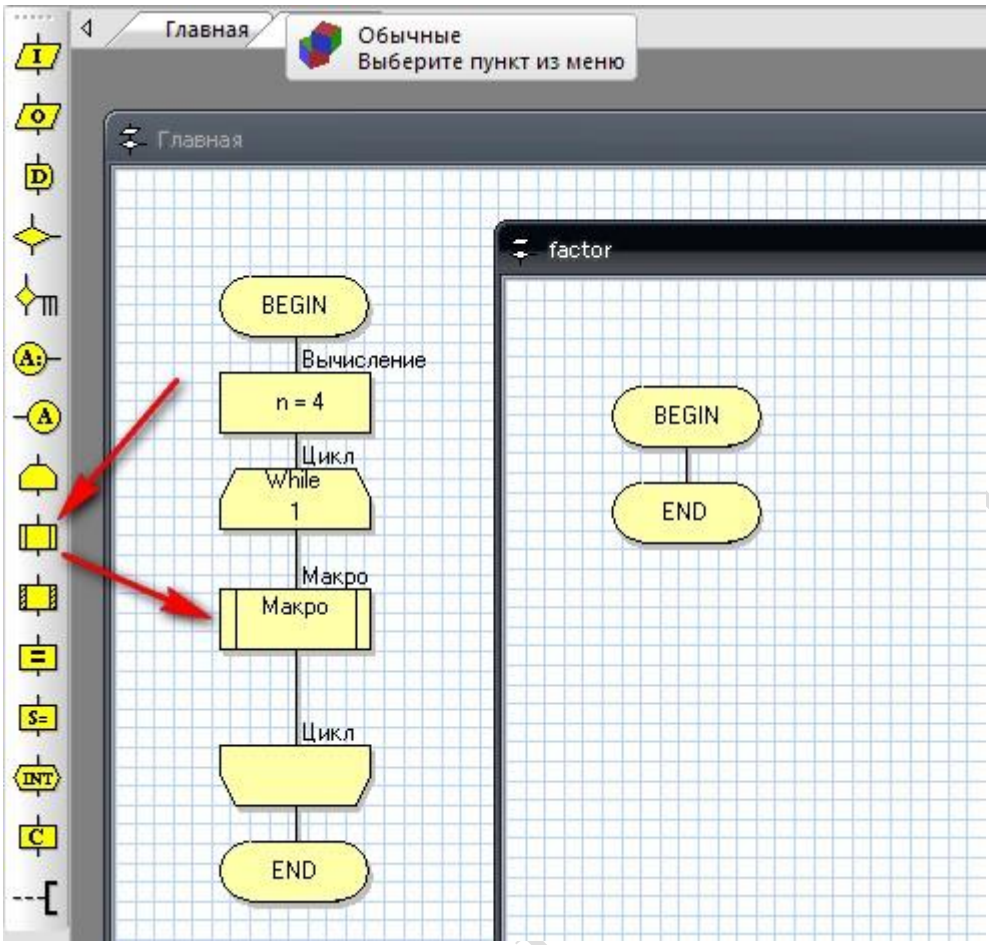

Рис.1.28. В цикл помещен макрос

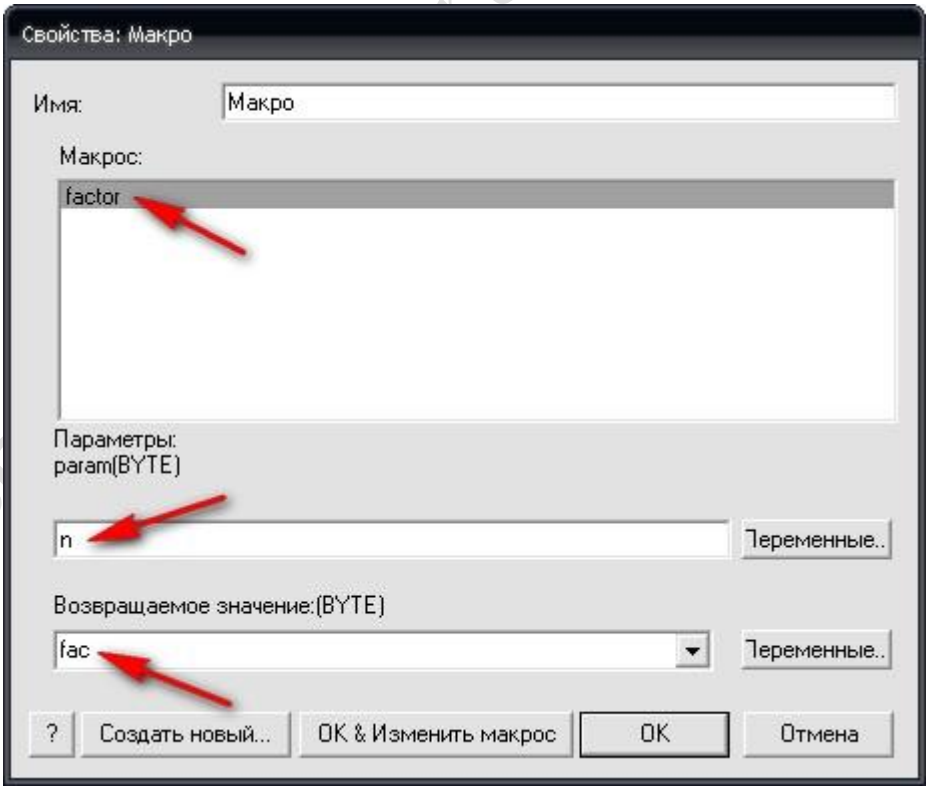

Рис.1.29. Свойства макроса

Теперь разместите после вызова макроса элемент **Выход**, свяжите его с портом С и задайте выводимое значение переменной **fac.**

В основной диаграмме все готово, однако сам алгоритм макроса пока еще не запрограммирован. Теперь будем работать с диаграммой макроса. Прежде всего отметим, что у внутренних объектов макроса (параметры и возвращаемое значение, а также локальные переменные, если они присутствуют) составные имена. Вначале идет имя макроса factor, а затем через разделитель «.» – имя. В нашем случае это выглядит как factor.param и factor.Return. Имя Return мы не задавали – оно формируется автоматически.

Для правильной работы алгоритма необходимо, чтобы начальное значение возвращаемой величины было равно 1. Разместите в области между BEGIN и END диаграммы макроса элемент **Вычисление** и определите в нем действие factor.Return = 1. Ниже разместите цикл, который выполняется до тех пор, пока параметр макроса больше нуля. В цикле разместите элемент **Вычисление**, и создайте в нем два выражения присваивания (рис. 1.30).

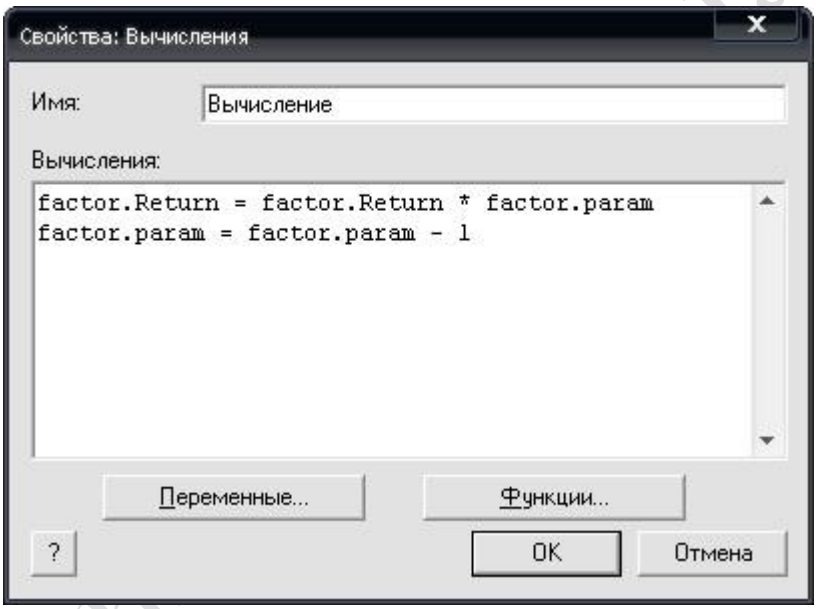

Рис.1.30. Содержимое элемента **Вычисление**

Здесь каждый раз в цикле в переменной factor.Return накапливается произведение, а параметр factor.param декрементируется, т.е. уменьшается на единицу.

Окончательный вид рабочей области проекта показан на рис. 1.31. Запустив проект на симуляцию, убеждается, что на линейке светодиодов высвечивается число 24, представленное в двоичной системе счисления. Попробуем изменить входное значение n на 0, просто отредактировав первое вычисление в элементе **Вычисление** в основной диаграмме. Все правильно. Особый случай факториала, как и оговорено, дает 1.

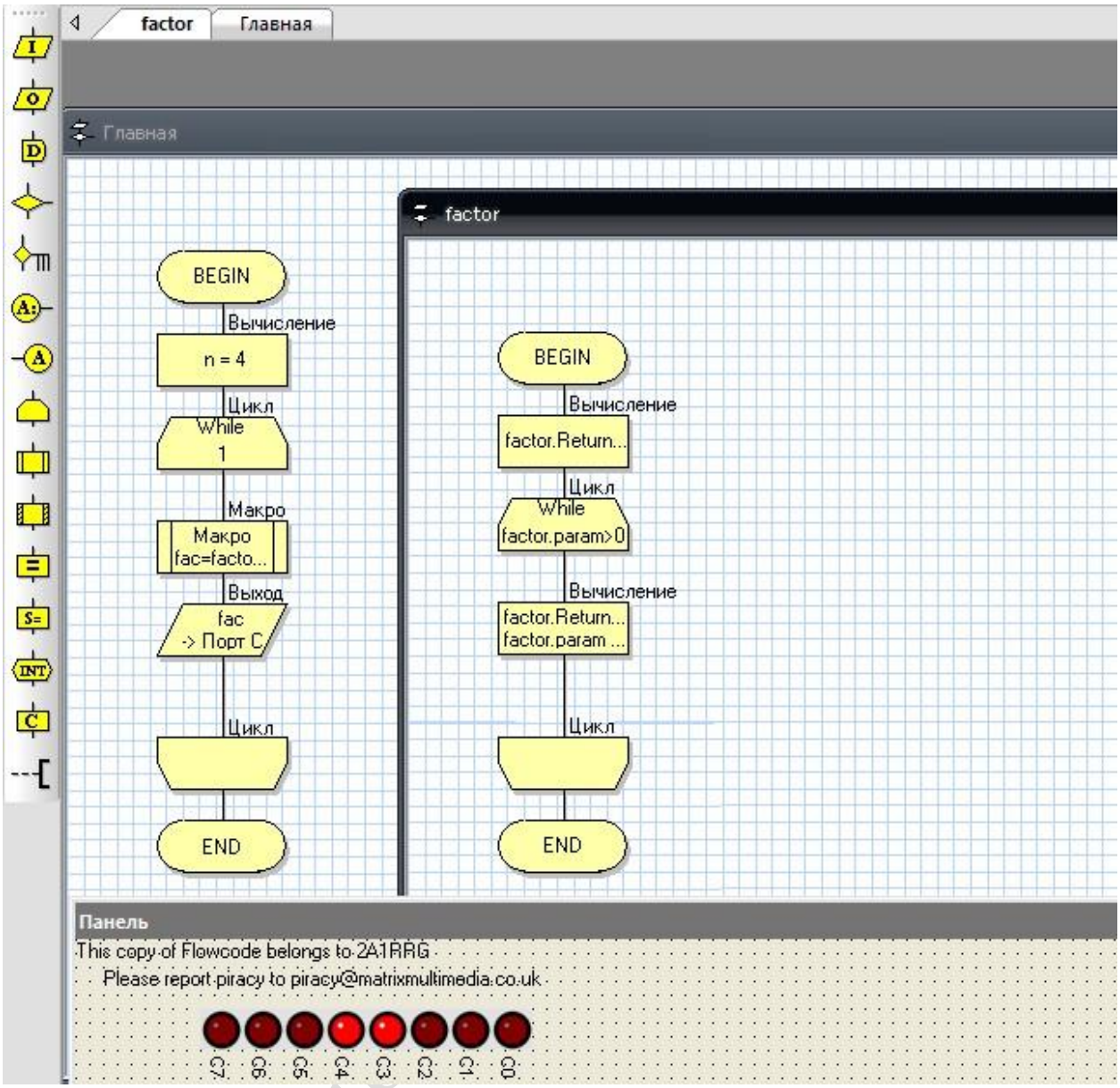

Рис.1.31. Проект вычисления факториала