

# Современная среда разработки mikroC для программирования микроконтроллеров на языке высокого уровня Си

(часть 1)

Олег Вальпа (Челябинская обл.)

Приводится описание современной, мощной и удобной среды разработки mikroC, которая включает большую библиотеку готовых функций для работы с разнообразными интерфейсами и устройствами и позволяет быстро создавать эффективные программы на языке высокого уровня Си для микроконтроллеров семейств PIC, AVR, MCS-51 и др.

## ВВЕДЕНИЕ

При разработке программ для микроконтроллерных устройств разработчик программы встречает ряд трудностей, преодоление которых отнимает время. Программист вынужден детально вникать в структуру программируемого микроконтроллера, изучать назначение множества его регистров, вплоть до каждого разряда, систему команд и т.п. Кроме того, существуют непроизводительные затраты времени, связанные с повторением этапов, многократно пройденных другими разработчиками.

Программисты при разработке программы, как правило, создают коды, с помощью которых выполняются процедуры инициализации регистров и векторов прерываний микроконтроллера, формируют функции и обработчики прерываний для внутренних интерфейсов микроконтроллера и внешних компонентов. Тем самым разработчики программ фактически повторяют многие стандартные процедуры. При этом большая часть времени тратится на отладку создаваемых функций и обработчиков.

Если для реализации конкретного алгоритма работы устройства действительно требуется уникальный код программы, то для организации работы с внутренними интерфейсами микроконтроллера и стандартизованными внешними устройствами вполне можно обходиться готовыми и проверенными библиотеками, имеющими в своём составе набор самых разнообразных функций для конкретного типа микроконтроллера.

Здесь можно провести аналогию с популярной средой разработки программ для персональных компьютеров Microsoft Visual C++, которая комплектуется библиотекой готовых функций MFC. Такой комплект позволяет создавать сложные программы в довольно сжатые сроки, не тратя массу времени на разработку функций для работы с клавиатурой, манипулятором «мышь», портами компьютера, файлами графики, звуком и т.п. Аналогичные функции имеет среда разработки Borland C++ Builder с библиотекой VCL, а также другие мощные инструменты для разработки компьютерных программ для ПК.

Однако, в настоящее время не только разработчики программ для ПК, но и разработчики программ микроконтроллерных устройств могут воспользоваться замечательной средой, имеющей в своём составе настоящий арсенал готовых функций, позволяющих использовать всю внутреннюю архитектуру микроконтроллера с многочисленными типами интерфейсов и множеством стандартизованных внешних устройств. Одним из таких программных инструментов является среда разработки mikroC компании MikroElektronika [1].

## НАЗНАЧЕНИЕ И СОСТАВ СРЕДЫ РАЗРАБОТКИ

Данная среда разработки позволяет быстро создавать эффективные программы на весьма распространённом и популярном языке высокого уровня Си. Среда имеет удобный и эргономичный интерфейс пользователя (IDE) со встроенным редактором

и мощным отладчиком программ. Встроенный в среду разработки мастер проектов позволяет в считанные минуты создать заготовку рабочей программы для любого микроконтроллера из целого семейства микроконтроллеров. Библиотека готовых функций, входящая в состав этой среды, обеспечивает программиста мощной поддержкой для быстрого и безошибочного создания практически любой программы.

Среда mikroC включает в себя огромное количество библиотечных функций – практически на все случаи жизни. Она содержит функции, которые поддерживают следующие устройства и интерфейсы:

- встроенный аналого-цифровой преобразователь (АЦП) микроконтроллера;
- внутреннюю энергонезависимую память EEPROM микроконтроллера;
- внутренние широтно-импульсные модуляторы (PWM) микроконтроллера;
- внешние сменные карты памяти типа MMC, SD и Compact Flash;
- файловую систему FAT;
- алфавитно-цифровые жидкокристаллические индикаторы (LCD, ЖКИ);
- графические жидкокристаллические индикаторы (GLCD, ЖК-дисплей);
- интерфейсы I<sup>2</sup>C, SPI, 1-Wire, USART, RS-485, CAN, PS/2, USB (HID) и Ethernet.

Компания MikroElektronika создала среду разработки mikroC для таких популярных и известных микроконтроллеров, как семейство PIC компании Microchip, AVR компании Atmel и MCS-51. Ведётся разработка среды и для других типов МК, в том числе, для самых современных 32-разрядных ARM-контроллеров.

Удачным подходом в создании данных сред является преимущество интерфейса для пользователя, что позволяет сэкономить время и силы на изучение среды для нового семейства

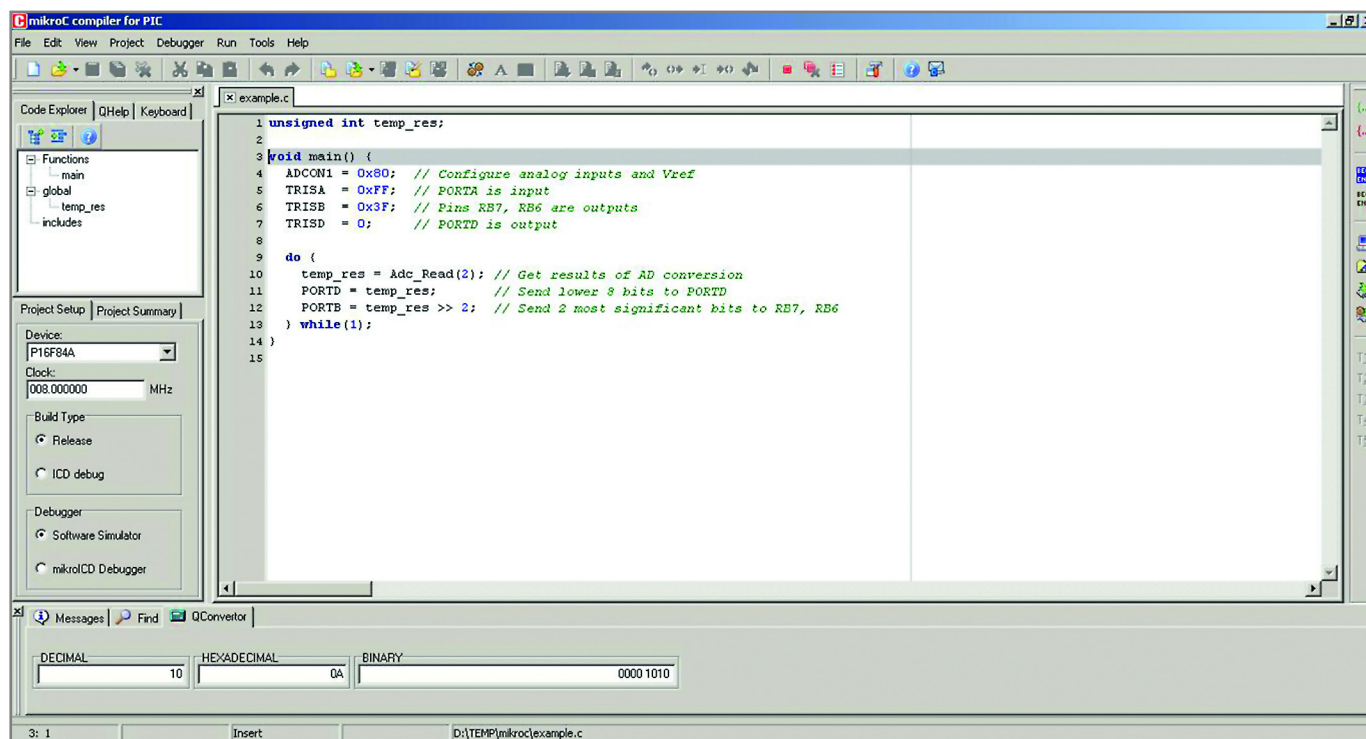


Рис. 1. Главное окно среды разработки mikroC

микроконтроллеров и сразу же приступить к этапу программирования.

## ОБЗОР СРЕДЫ

Рассмотрим среду разработки mikroC для весьма популярных и распространённых микроконтроллеров семейства PIC компании MicroChip [2] на примере версии 8.2.0.0. Другие версии данной среды разработки аналогичны описываемой здесь и отличаются несущественно.

Поскольку компания MikroElektronika использует принцип наследования для своих продуктов, переход к среде разработки для других семейств микроконтроллеров происходит довольно просто. Интерфейс пользователя и даже названия подавляющей части библиотечных функций в среде разработки mikroC для различных семейств микроконтроллеров остаются практически неизменными. Поэтому, изучив и освоив на практике описываемую здесь среду mikroC для микроконтроллеров семейства PIC, в дальнейшем можно будет легко создавать программы для микроконтроллеров других семейств в соответствующей для них среде mikroC.

Среда MikroC является дружелюбной и интуитивно понятной. На рисунке 1 представлено главное окно среды разработки. В центре этого окна располагается редактор кода (Code Editor) с исходным текстом программы на языке программирования Си. Слева от ре-

дактора кода находится окно с закладками проводника кода (Code Explorer), быстрой помощи (QHelp) и назначением клавиш (Keyboard).

Под окном редактора кода находится окно с закладками настройки проекта (Project Setup) и содержимого проекта (Project Summary). Внизу располагается окно с закладками сообщений (Messages), поиска (Find) и быстрого конвертора (QConvertor). Справа находится панель инструментов, а сверху – главное меню среды разработки.

Среда MikroC имеет следующие возможности:

- текст программы вводится с помощью встроенного редактора исходного кода;
- все строки программы имеют нумерацию;
- имеется встроенная помощь кода и параметров, контекстная подсветка, автоматическая коррекция кода, кодовые шаблоны и т.п.;
- проводник кода (Code Explorer) позволяет оперативно контролировать структуру программы, переменные и функции проекта;
- после компиляции проекта создаются комментированный файл на ассемблере и стандартный HEX-файл для использования разными типами программаторов;
- встроенный отладчик позволяет проверять ход и логику исполнения программы;

- после компиляции предоставляется полная статистика использования памяти, ассемблерный листинг, дерево вызовов функций и т.п.;
- включено большое количество примеров, которые можно расширять и использовать как составные части разрабатываемых проектов.

## СОЗДАНИЕ ПРОГРАММЫ

Программы в mikroC организованы в виде проектов, состоящих из файла проекта с расширением .prc, одного или нескольких файлов исходного кода, имеющих расширение .c, а также создаваемых в процессе трансляции вспомогательных файлов. Файл с исходным кодом автоматически компилируется, если он включен в состав проекта.

Файл проекта несёт следующую информацию:

- имя проекта и необязательное описание;
- тип микроконтроллера;
- конфигурацию;
- тактовую частоту микроконтроллера;
- список исходных файлов проекта с указанием путей к ним.

Самый простой способ создания проекта – использовать для этих целей мастер новых проектов, который запускается из главного меню с помощью команды Project → New Project. В открывшемся окне нового проекта (см. рис. 2) следует заполнить поля

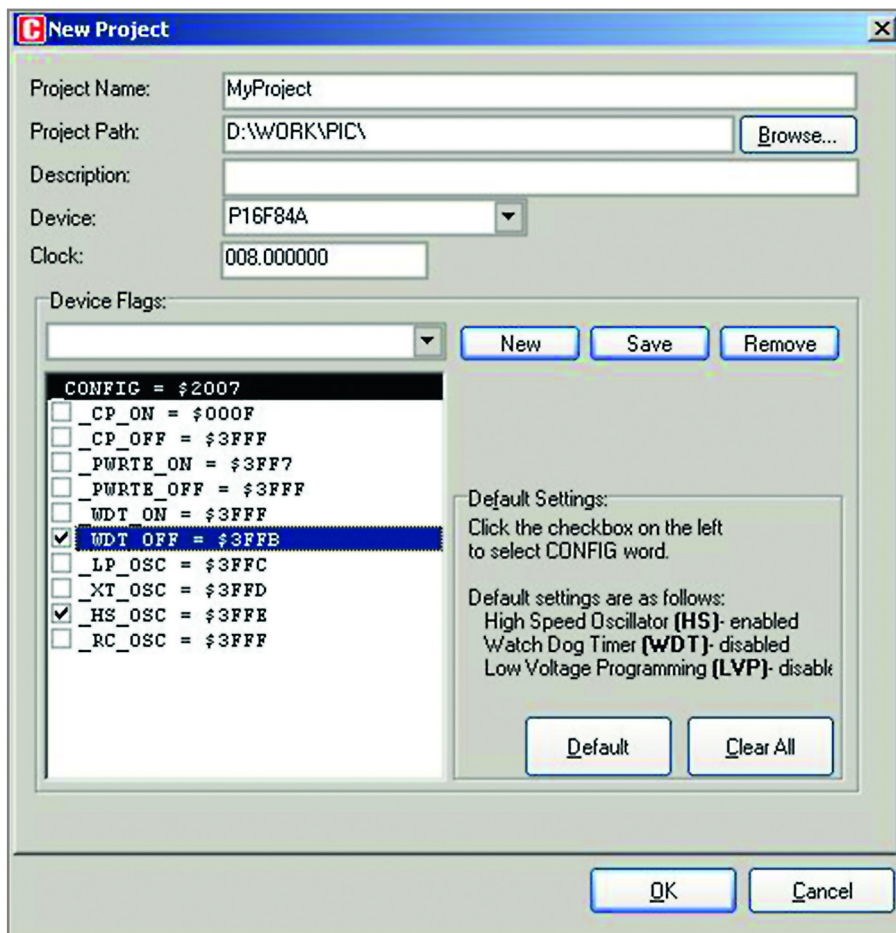


Рис. 2. Окно нового проекта

описания проекта необходимыми значениями (название и описание проекта, расположение, устройство, частота, слово конфигурации). После щелчка по программной кнопке ОК mikroC создаст соответствующий файл проекта.

Кроме этого, после создания проекта будет создан пустой исходный файл с расширением .c для текста программы, имеющий название проекта. Среда MikroC не требует, чтобы исходный файл имел такое же название, как проект, поэтому его можно переименовать.

В любое время можно изменить настройки проекта, используя главное меню Project → Edit Project. Проект можно переименовать, изменить его описание, используемый микроконтроллер, частоту, конфигурацию и т.п.

Для удаления проекта достаточно удалить папку, в которой располагается файл проекта с расширением .prc. Проект может содержать произвольное количество исходных файлов с расширением .c. Список имеющих отношение к проекту файлов сохраняется в файле самого проекта. Чтобы добавить исходный файл в проект, надо

выбрать команду Project → Add to Project из выпадающего меню или кликнуть мышкой по пиктограмме добавления файла к проекту.

Для удаления файла из проекта следует выбрать команду Project → Remove from Project из выпадающего меню или кликнуть по пиктограмме удаления файла из проекта.

Для включения в проект заголовочного файла с расширением .h, следует использовать в коде программы директиву препроцессора #include, например:

```
#include <header_name>
```

или

```
#include "header_name"
```

Чтобы создать новый исходный файл, следует выбрать команду File → New из выпадающего меню, или нажать комбинацию клавиш Ctrl+N, или кликнуть мышкой по пиктограмме создания нового файла. После этого открывается вкладка с новым названием. Это и есть новый исходный файл. Для задания названия следует выбрать команду File → Save As из выпадающего

меню и сохранить файл с новым названием.

Чтобы отрыть уже существующий файл, надо выбрать команду File → Open из выпадающего меню, или нажать комбинацию клавиш Ctrl+O, или кликнуть мышкой по иконке открытия файла. После этого появляется окно выбора файла, в котором и следует произвести обычные действия по выбору и открытию. Открытый файл выводится в собственной вкладке. Если выбранный файл уже открыт, его вкладка становится активной, т.е. доступной для редактирования.

Если потребуется распечатать какой-либо файл проекта, то вкладка файла, который предполагается вывести на печать, должна быть активной. Затем следует выбрать команду File → Print из выпадающего меню, или нажать клавиши Ctrl+P, или кликнуть мышкой по пиктограмме печати. В предварительном окне печати следует установить необходимые настройки печати и кликнуть по кнопке ОК. После этого файл будет отправлен на выбранный принтер.

Для сохранения файла надо выбрать команду File → Save из выпадающего меню, или нажать клавиши Ctrl+S, или кликнуть мышкой по пиктограмме сохранения файла. Файл будет сохранён под старым названием. Вкладка файла, который предполагается сохранять, должна быть активной.

Для сохранения файла под другим названием следует выбрать File → Save As из выпадающего меню или нажать клавиши Ctrl+Shift+S. После этого выводится диалоговое окно сохранения файла под другим названием. В этом окне надо выбрать каталог, где будет сохранён файл. В поле имени файла надо задать новое название и кликнуть мышкой по пиктограмме сохранения.

Заккрытие файла выполняется выбором команды File → Close из выпадающего меню или кликом по кнопке закрытия вкладки. Если файл был изменён со времени последнего сохранения, будет предложено сохранить эти изменения.


### РЕДАКТИРОВАНИЕ КОДА ПРОГРАММЫ

Среда разработки программ mikroC имеет встроенный современный текстовый редактор, способный удовлетворить запросы даже искушённого

профессионала. Редактирование кода программы практически не отличается от работы со стандартным текстовым редактором, включая возможности копирования, вставки и отката, используемые в операционной системе Windows.

Но, кроме основных операций, этот редактор поддерживает специальные сервисные возможности, значительно облегчающие труд программиста. К этим возможностям относятся:

- настраиваемая контекстная подсветка (Syntax Highlighting);
- кодовый ассистент (Code Assistant);
- ассистент параметров (Parameter Assistant);
- кодовые шаблоны;
- автокоррекция кода программы (Auto Correct);
- закладки и переходы по номеру строки.

Настройка параметров этих возможностей редактора осуществляется в окне диалога Preferences (см. рис. 3). Данный диалог запускается с помощью команды Tools → Options из выпадающего меню или щелчком мыши по пиктограмме .

Рассмотрим подробнее назначение и использование этих возможностей. Контекстная подсветка значительно улучшает читаемость программы и обнаружение в ней ошибок за счёт автоматического окрашивания фрагментов программы в соответствии с их функциональным назначением. Например, как это сделано в окне с кодом программы диалога Preferences.

Диалог Preferences редактора кода программы уже содержит три готовые схемы раскраски с названиями: mikroDream (с белым фоном), MrGreen (с зелёным фоном) и Zedar (с чёрным фоном). В диалоге Preferences пользователь среды может выбрать любую из имеющихся схем подсветки либо создать свою собственную.

Кодовый ассистент (Code Assistant) подсказывает правильное название и написание существующего кода программы. Например, если в редакторе кода программы напечатать один или несколько символов слова и нажать клавиши Ctrl+Space, все разрешённые идентификаторы, соответствующие напечатанным символам, будут предложены во всплывающем окне (см. рис. 4). Теперь можно продолжить ввод символов для сужения предлагаемого

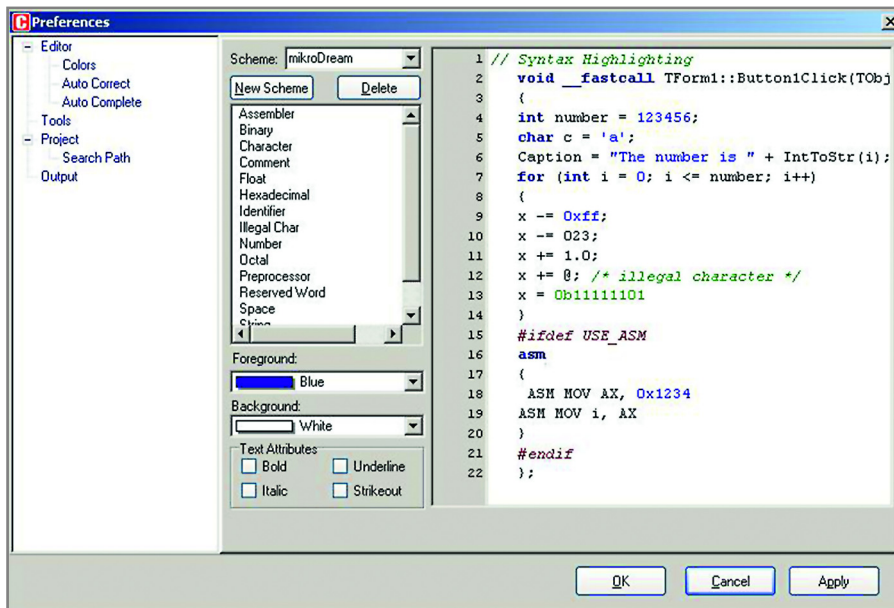


Рис. 3. Окно диалога Preferences

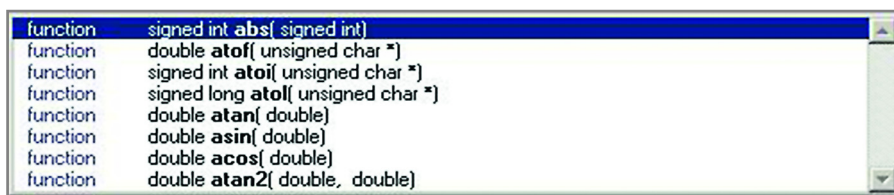



Рис. 4. Окно Code Assistant

списка или с помощью стрелок на клавиатуре выбрать подходящий вариант кода из предложенных строк и нажать клавишу Enter.

Ассистент параметров (Parameter Assistant) помогает правильно ввести параметр или аргумент функции. Он вызывается автоматически после ввода символа открывающейся скобки «(» или после нажатия комбинации клавиш Shift+Ctrl+Space. Если имя разрешённой функции предшествует скобкам, предполагаемые аргументы будут выведены на всплывающую панель. По мере печати очередного фактического аргумента, следующий предлагаемый аргумент будет выделен жирным шрифтом.

Кодовые шаблоны помогают ускорить ввод стандартных операторов и конструкций языка программирования без ошибок. Шаблон можно вставлять в текст программы, вводя название шаблона (например, for) и нажав комбинацию клавиш Ctrl+J, после чего редактор автоматически сгенерирует код шаблона и вставит его в текст программы. Другой способ вставки шаблона заключается в вызове диалога с помощью клавиши F12 и последующем выборе шаблона из списка в закладке Auto Complete, с переносом его в код программы копированием.

Кроме того, можно добавить собственные шаблоны с помощью диалога Preferences на вкладке Auto Complete. Для этого следует ввести соответствующее ключевое слово, описание и свой кодовый шаблон. Автокоррекция позволяет исправлять наиболее часто встречающиеся опечатки. Для доступа к списку распознаваемых опечаток следует выбрать команду Tools → Options из выпадающего меню или кликнуть по пиктограмме инструментов , а затем выбрать вкладку Auto Correct. Здесь можно добавить в список свои собственные предпочтения.

Закладки облегчают навигацию в коде большого размера. Для установки закладки используется комбинация клавиш Ctrl+Shift+number. Для перехода к закладке используется Ctrl+number.

Переход на заданную строку облегчает навигацию в коде большого размера. Для этого достаточно нажать комбинацию клавиш Ctrl+G и в появившемся окне (см. рис. 5) ввести номер искомой строки.



Кроме этого, редактор исходного кода имеет возможность показывать и скрывать комментарии к выбранному коду одним кликом мышки, используя пиктограммы Comment Code  и Re-




Рис. 5. Окно поиска строки по номеру


move Comments  на панели инструментов (Code Toolbar).

С появлением новых версий среды разработки появляются и новые возможности редактора кода. Их описание всегда можно найти во встроенной справочной системе среды разработки.

### КОМПИЛЯЦИЯ ПРОЕКТА

После создания проекта и написания исходного кода необходимо выполнить его компиляцию. Для этого следует выбрать команду Run → Compile из выпадающего меню или кликнуть по пиктограмме компилятора , расположенную на панели компилятора (Compiler Toolbar).


Появившийся индикатор процесса компиляции будет информировать о ходе компиляции. В случае обнаружения ошибок появятся соответствующие сообщения в окне ошибок (Error Window). Если ошибок нет, в окне Messages появятся сообщения об успешной компиляции и объёме использованной памяти микроконтроллера под программу. В случае успешной компиляции среда mikroC создаёт выходные файлы в каталоге проекта (каталог, где содержится файл проекта с расширением .prc). Описание выходных файлов приведено в таблице 1.

После компиляции программы в mikroC, можно кликнуть по иконке просмотра ассемблера (View Assembly)  или выбрать команду View → View Assembly из выпадающего меню, чтобы проверить сгенерированный ассемблерный код (файл с расширением .asm) в новом окне. Ассемблерный код представляет собой текст с символическими именами.

ческими именами. Все физические адреса и прочая информация может быть найдена в статистике или файле листинга.

Если программа не компилировалась и ассемблерный файл не создавался, использование этой возможности приводит к компиляции кода и получению ассемблерного файла для просмотра.

### СТАТИСТИКА


После успешной компиляции можно посмотреть статистические сведения о коде. Для этого следует выполнить команду View → View Statistics из выпадающего меню или кликнуть по пиктограмме статистики . При этом появится окно с шестью вкладками (см. рис. 6):

- окно использования памяти (Memory usage) обеспечивает просмотр использования RAM и ROM в форме гистограмм;
- окно размера функций (Procedures sizes) выдаёт в форме гистограммы размеры памяти, занимаемой каждой функцией;
- окно расположения функций (Procedures locations) выдаёт в форме гистограммы положение в памяти, занимаемой каждой функцией;
- окно детальной информации по каждой функции (Procedures Details) позволяет посмотреть полное дерево вызова вместе с остальной информацией: размер, начальный и конечный адреса, частота обращений, тип возвращаемого значения и т.п.;
- окно RAM позволяет посмотреть распределение всех регистров и их адреса. Также выводятся символьные имена переменных с адресами;
- окно ROM выдаёт список всех кодов инструкций с адресами в читабельном шестнадцатеричном представлении.

### ОТЛАДКА ПРОГРАММЫ

Отладчик в исходных кодах является встроенной компонентой среды mikroC.

Он предназначен для симуляции работы МК и для облегчения пользователю процесса отладки кодов программы на языке программирования Си, написанных для этих микроконтроллеров.

После успешной компиляции проекта можно запустить отладчик с помощью команды Run → Start debugger из выпадающего меню, или кликнуть по пиктограмме отладчика , или нажать клавишу F9. При этом появится окно отладчика программы (см. рис. 7).

Отладчик позволяет выполнять пошаговое исполнение программы (Step Into), пошаговое исполнение с «перешагиванием» функций (Step Over), исполнение до текущей позиции курсора (Run to Cursor) и т.д. Строка программы, которая будет исполняться, подсвечивается (по умолчанию – синим цветом).

В отладчике можно включать и выключать точки останова в текущей позиции курсора с помощью клавиши F5. Чтобы посмотреть все точки останова, используется команда Run → View Breakpoints из выпадающего меню. Щелчок левой кнопкой мыши по номеру строки в окне исходного текста программы также включает и выключает точку останова в этой строке.

Команда Run to cursor (клавиша F4) выполняет все операторы программы от текущего и до позиции курсора.

Команда Step Into (клавиша F7) выполняет текущий оператор программы с заходом в вызываемую функцию.

Команда Step Over (клавиша F8) выполняет текущий оператор программы без захода в вызываемую функцию.

Команда Step Out (клавиши Ctrl+F8) последовательно выполняет все операторы программы, пока не встретится точка останова.

Переключение между окном программы на языке Си и окном с дизассемблированным кодом осуществляется с помощью комбинации клавиш Alt+D.

Окно наблюдения отладчика (Watch) – это основное окно отладки, которое позволяет контролировать элементы программы во время её отладки. Чтобы открыть окно наблюдения, надо выбрать команду View → Debug Windows → View Watch из выпадающего меню во время отладки. Окно наблюдения показывает переменные и регистры микроконтроллера, их адреса и значе-

Таблица 1. Описание выходных файлов

| Формат файла        | Описание содержимого файла   | Расширение файла |
|---------------------|--|------------------|
| Intel HEX           | Шестнадцатеричный код в формате Intel. Используется для программирования                                   | hex              |
| Двоичный            | Библиотека компилятора mikroC. Двоичные дистрибутивы подпрограмм, пригодные для включения в другие проекты | mcl              |
| Файл листинга       | Общая картина распределения памяти микроконтроллера: адреса инструкций, регистры, программы и метки        | lst              |
| Текст на ассемблере | Читаемый ассемблерный файл с символическими именами, полученный из файла листинга                          | asm              |

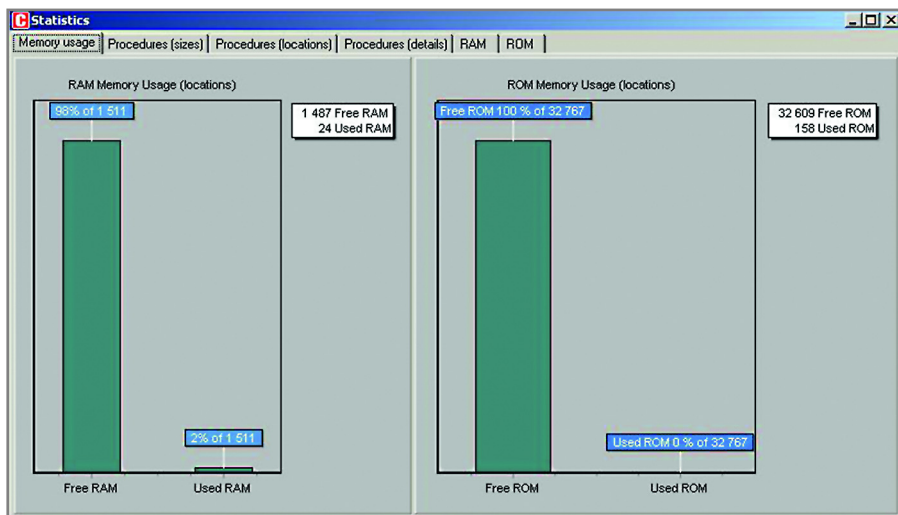


Рис. 6. Окно статистики

ния. Значения обновляются в процессе симуляции работы программы в отладчике. Последние изменённые элементы в окне выделяются красным цветом. Для добавления или удаления наблюдаемых элементов в окно следует использовать соответствующее выпадающее меню.

Двойной щелчок на элементе в окне наблюдения открывает окно редактирования, где можно присвоить выбранной переменной или регистру новое значение. Также можно выбрать двоичное, шестнадцатеричное, десятичное или символьное представление этого значения в окне наблюдения.

Окно хронометража (Stopwatch) отладчика (см. рис. 8) доступно из выпадающего меню View → Debug Windows → View Clock. В окне хронометража отображается текущий счётчик (Current Count) периодов тактовой частоты и секунд от момента запуска отладчика. Секундомер (Stopwatch) измеряет время исполнения в периодах тактовой частоты и секундах от момента запуска отладчика и может быть обнулён в любое время. Разность (Delta) представляет фактически время выполнения участка программы от предыдущей точки останова до текущей в периодах и секундах (при пошаговом

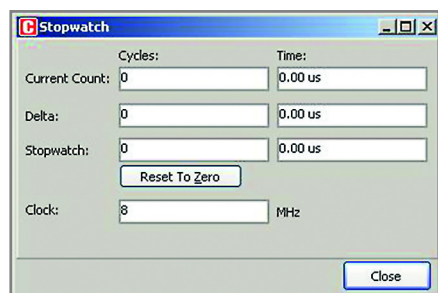


Рис. 8. Окно хронометража отладчика

исполнении отображается время выполнения одной строки кода программы на Си). Также в окне отображается текущая тактовая частота микроконтроллера (Clock).

Тактовую частоту в окне хронометража можно изменять, что приведёт к пересчёту времени в секундах. Это изменение не влияет на текущие установки проекта, где тоже задана тактовая частота микроконтроллера, а влияет только на расчёт времени симуляции.

Окно просмотра памяти (View RAM Window) доступно из выпадающего меню View → Debug Windows → View RAM. Окно просмотра памяти (см. рис. 9) показывает карту памяти МК, где самые последние изменения выделены красным цветом. Можно изменять значения полей карты памяти путём двойного щелчка на нужном поле.

В случае если компилятор обнаруживает ошибку, он сообщает об этом и не создаёт выходной файл. Окно ошибок, расположенное по умолчанию внизу основного окна, напоминает об этом. Окно ошибок расположено под вкладкой сообщений и отображает место и тип ошибки, обнаруженной компилятором. Также компилятор выводит предупреждения, но они не влияют на выходной файл. Двойной щел-

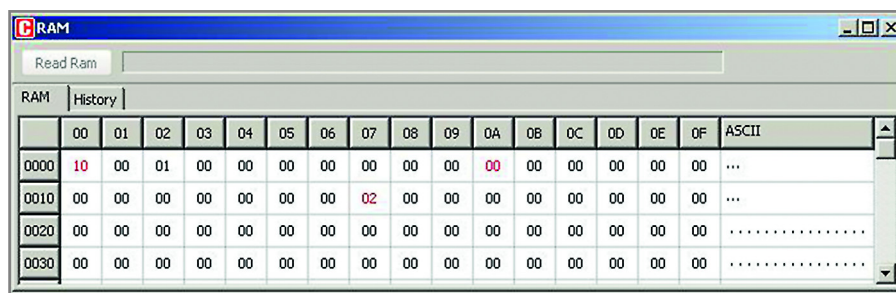


Рис. 9. Окно просмотра памяти МК

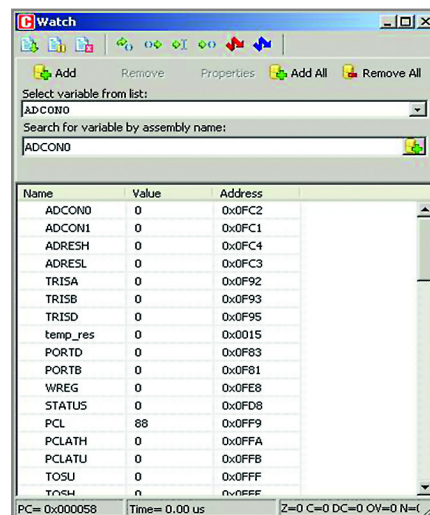


Рис. 7. Окно отладчика программы

чок по строке сообщения в окне ошибок приводит к подсветке строки исходного текста, где эта ошибка была обнаружена.

**ИНСТРУМЕНТЫ СРЕДЫ**

Встроенная карта ASCII (см. рис. 10) позволяет посмотреть код любого символа ASCII в десятичном, шестнадцатеричном и двоичном формате. Открыть карту символов ASCII можно из выпадающего меню Tools → ASCII chart, а посмотреть код символа очень удобно с помощью мыши, наводя её курсор на интересующий код в окне карты символов.

С помощью инструмента экспорта кода, вызываемого из меню Tools → Export Code To HTML, можно очень просто получить код программы в формате HTML для публикации его в Интернете.

Например, код программы, полученный из примера программы, описанного выше, будет иметь следующий вид на странице Интернет:

```
Version:0.9 StartHTML:0000000105
EndHTML:0000004659 StartFragment:0000001167 EndFragment:0000004643
void main() {
ADCON1 = 0x80; // Configure analog inputs and Vref
```

|   | 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9  | A   | B   | C  | D  | E  | F   |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|----|-----|-----|----|----|----|-----|
| 0 | NUL | SOH | STX | ETX | EOT | ENQ | ACK | BEL | BS  | HT | LF  | VT  | FF | CR | SO | SI  |
| 1 | DLE | DC1 | DC2 | DC3 | DC4 | NAK | SYN | ETB | CAN | EM | SUB | ESC | FS | GS | RS | US  |
| 2 | SPC | !   | "   | #   | \$  | %   | —   | '   | (   | )  | *   | +   | ,  | -  | .  | /   |
| 3 | 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9  | :   | ;   | <  | =  | >  | ?   |
| 4 | @   | A   | B   | C   | D   | E   | F   | G   | H   | I  | J   | K   | L  | M  | N  | O   |
| 5 | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y  | Z   | [   | \  | ]  | ^  | _   |
| 6 | `   | a   | b   | c   | d   | e   | f   | g   | h   | i  | j   | k   | l  | m  | n  | o   |
| 7 | p   | q   | r   | s   | t   | u   | v   | w   | x   | y  | z   | {   |    | }  | ~  | DEL |
| 8 | Ъ   | Г   | Г   | Г   | „   | …   | †   | ‡   | €   | %o | Ь   | <   | Ь  | Ќ  | Ў  | Ѓ   |
| 9 | ђ   | '   | "   | "   | "   | •   | —   | —   | □   | ™  | Ь   | >   | Ь  | Ќ  | Ў  | Ѓ   |
| A |     | Ў   | Ў   | Ј   | Ѡ   | Г   | Ѓ   | Ѓ   | Є   | ©  | Є   | «   | ¬  | -  | @  | Ў   |
| B | °   | ±   | I   | i   | г   | μ   | ¶   | ·   | ё   | №  | є   | »   | j  | S  | s  | i   |
| C | А   | Б   | В   | Г   | Д   | Е   | Ж   | З   | И   | Й  | К   | Л   | М  | Н  | О  | П   |
| D | Р   | С   | Т   | У   | Ф   | Х   | Ц   | Ч   | Ш   | Щ  | Ъ   | Ы   | Ь  | Э  | Ю  | Я   |
| E | а   | б   | в   | г   | д   | е   | ж   | з   | и   | й  | к   | л   | м  | н  | о  | п   |
| F | р   | с   | т   | у   | ф   | х   | ц   | ч   | ш   | щ  | ъ   | ы   | ь  | э  | ю  | я   |

Рис. 10. Встроенная карта ASCII

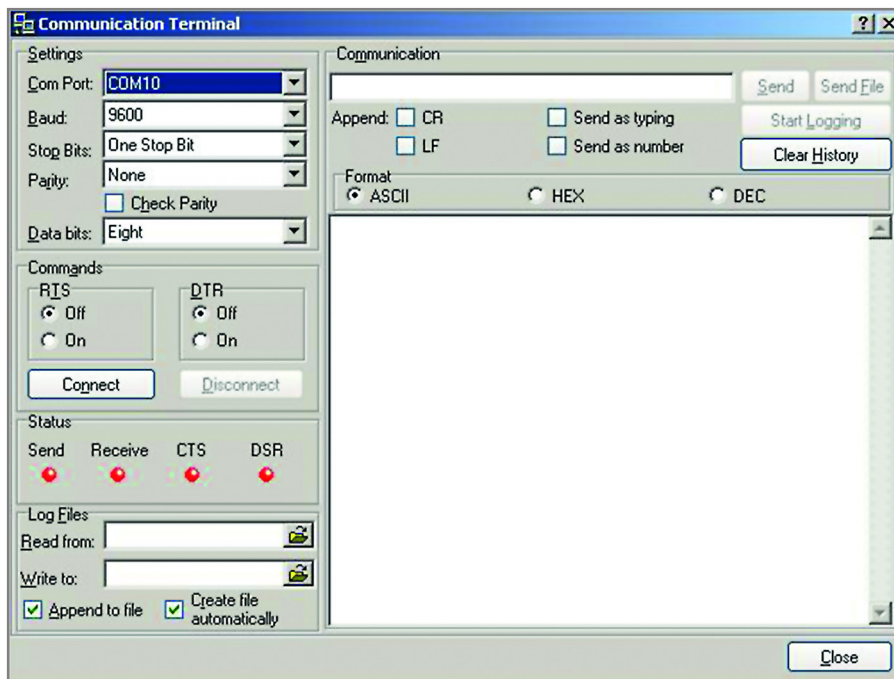


Рис. 11. Терминал связи USART

```
TRISA = 0xFF; // PORTA is input
TRISB = 0x3F; // Pins RB7, RB6
are outputs
TRISD = 0; // PORTD is output
```

```
do {
temp_res = Adc_Read(2); // Get
results of AD conversion
PORTD = temp_res; // Send lower 8
```

```
bits to PORTD
PORTB = temp_res >> 2; // Send 2
most significant bits to RB7, RB6
} while(1);
```

В среде mikroC включен терминал связи USART для работы с интерфейсом RS-232 (см. рис. 11). Его можно запустить из выпадающего меню Tools → Terminal, или щелчком по иконке терминала, или с помощью клавиш Ctrl+T. Он позволяет отлаживать программы, в которых используется интерфейс USART для связи с внешними устройствами, на разных скоростях и с различными форматами данных.

Декодер семисегментных символов (см. рис. 12) является удобным инструментом получения кодов индикации по нужной комбинации активных сегментов. Активация и деактивация сегмента осуществляется простым щелчком мыши на изображении сегмента. Декодер можно запустить из выпадающего меню Tools → Seven Segment Decoder.

Встроенный генератор кода для ЖК-индикаторов (см. рис. 13) позволяет очень легко создать любой символ для матрицы ЖКИ. Данный генератор кода запускается из выпадающего меню Tools → LCD Custom Character. Он же позволяет сгенерировать программный код для отображения этого символа на языке программирования mikroPascal, mikroBasic или mikroC (см. рис. 14) простым нажатием на программную кнопку GENERATE. Ниже приведён программный код на языке программирования Си, полученный данным генератором кода.

```
const char character[] =
{0,0,16,0,2,0,8,0};
void CustomChar(char pos_row,
char pos_char) {
char i;
LCD_Cmd(64);
for (i = 0; i<=7; i++)
LCD_Chrcp(character[i]);
LCD_Cmd(LCD_RETURN_HOME);
LCD_Chrcp(pos_row, pos_char, 0);
}
```

Ещё более мощным встроенным инструментом среды является генератор кода для графических ЖК-дисплеев (см. рис. 15). Он запускается из выпадающего меню Tools → GLCD Bitmap Editor и позволяет создавать точечные рисунки для дисплеев типа KS0108,

T6963 и Nokia3110, а также генерировать программный код для этих дисплеев на трёх языках программирования.

Для отладки программ, обеспечивающих работу с картами памяти типа MMC, существует встроенный инструмент MMC Terminal (см. рис. 16). Он запускается из выпадающего меню Tools → MMC Card Terminal и позволяет осуществлять чтение и запись данных карты памяти по секторам через последовательный COM-порт компьютера. Встроенный редактор перепрограммируемой памяти микроконтроллера (см. рис. 17) запускается из выпадающего меню Tools → EEPROM Editor и позволяет модифицировать данные этой памяти.

Для отладки программ можно использовать специальный загрузчик программ mikroBootloader (см. рис. 18). Поскольку многие семейства МК имеют возможность записывать в свою собственную программную память данные, это позволяет хранить в них небольшую программу-загрузчик, которая будет принимать и записывать в программную память коды программы. В наиболее простом варианте загрузчик передаёт управление программе пользователя всегда, кроме случая, когда есть запрос на загрузку новой программы. В такой ситуации загрузчик получает данные и записывает их в программную память. Существует много усовершенствований, которые могут быть добавлены, чтобы сделать процесс загрузки более надёжным и удобным.

Естественно, что загрузчик можно использовать только с микроконтроллерами, поддерживающими запись во флэш-память программ. Резидентная часть загрузчика даёт компьютеру 5 с на то, чтобы установить с ней связь. Если этого не происходит, она запускает на исполнение имеющуюся программу пользователя. Если связь установлена, резидентная часть загрузчика принимает коды и записывает их в программную память.

Ниже описан процесс использования загрузчика:

- загрузить в микроконтроллер необходимый файл программы, используя обычную технологию программирования файла в HEX-формате. Например, для микроконтроллера PIC16F877A использовать файл p16f877a.hex;

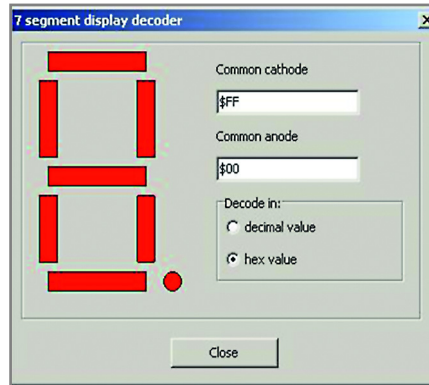


Рис. 12. Декодер семисегментных символов

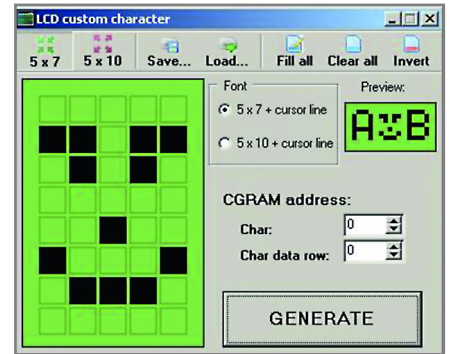


Рис. 13. Встроенный генератор кода для ЖК-индикаторов

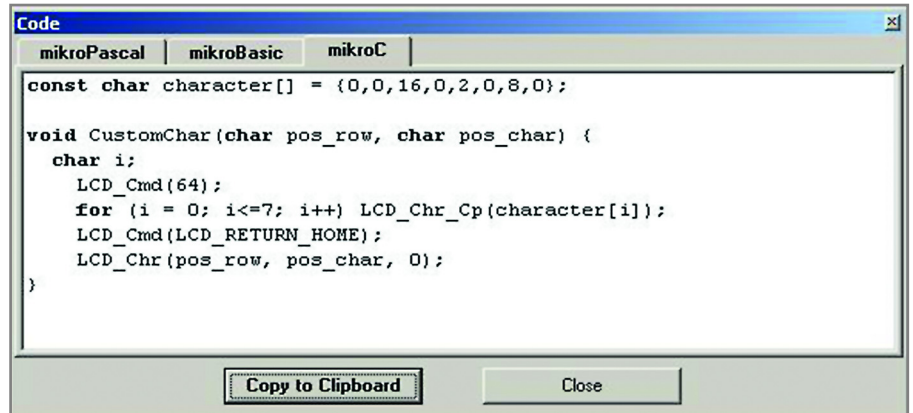


Рис. 14. Окно с примером кода для ЖК-индикаторов

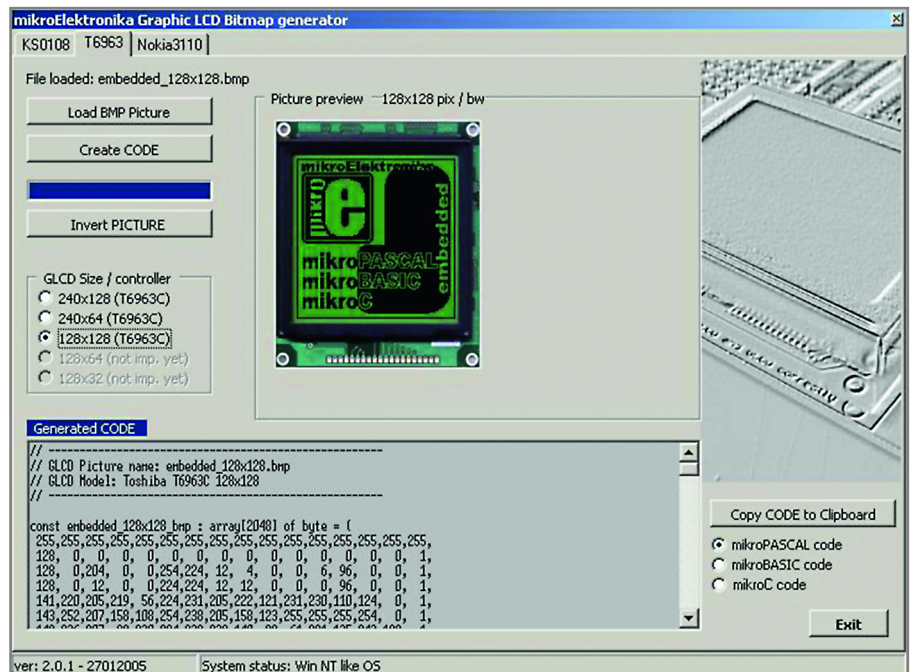


Рис. 15. Генератор кода для графических ЖК-дисплеев

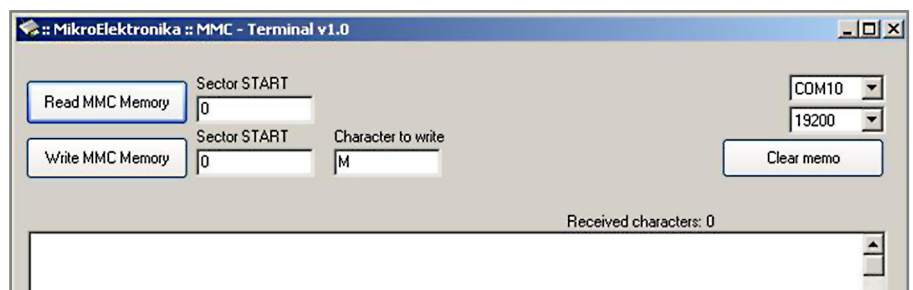


Рис. 16. Встроенный инструмент MMC Terminal



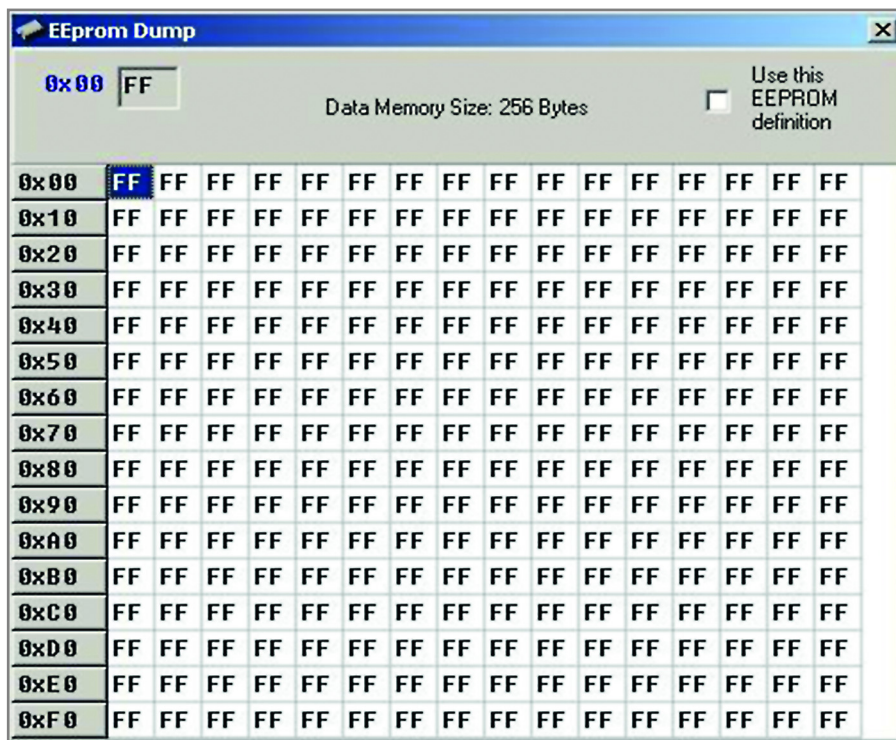


Рис. 17. Встроенный редактор перепрограммируемой памяти МК

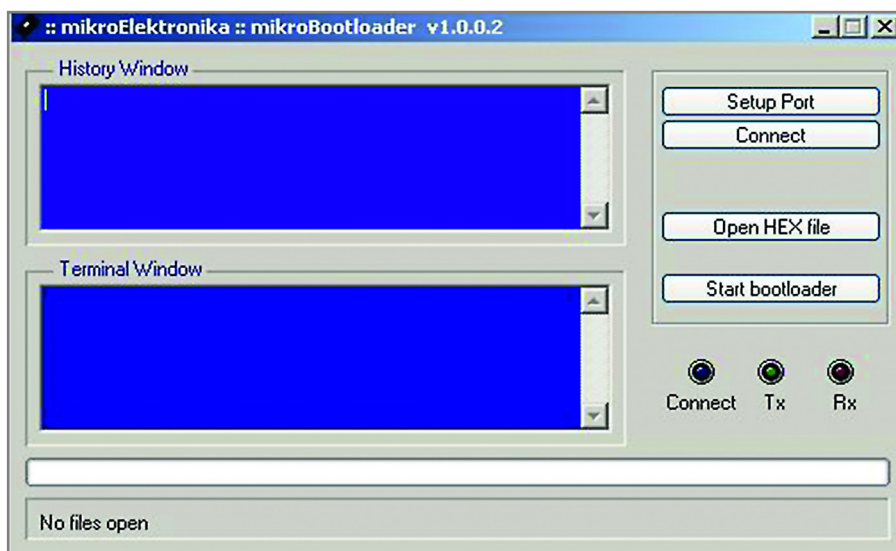


Рис. 18. Загрузчик программ mikroBootloader

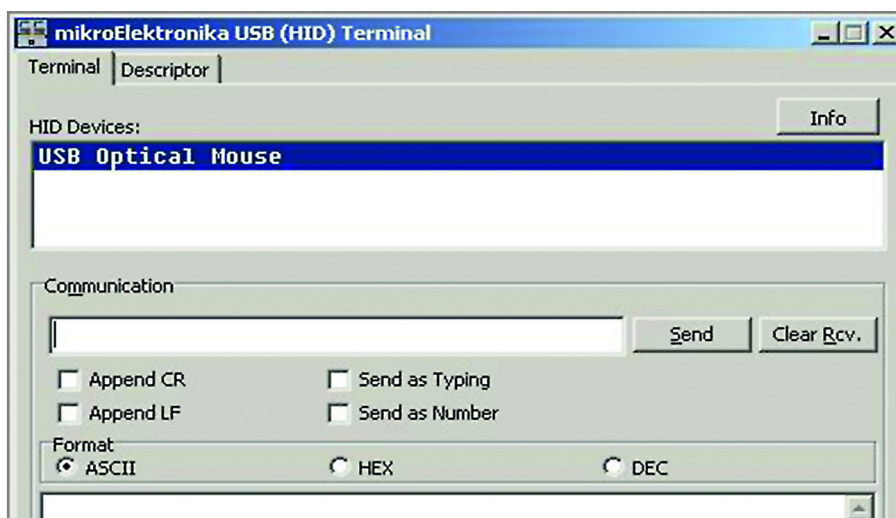


Рис. 19. Встроенный инструмент отладки программ для USB-устройств

- запустить загрузчик из выпадающего меню Tools → mikroBootloader;
- щёлкнуть на Setup Port и затем выбрать номер COM-порта, используемого для связи с микроконтроллером. Убедиться, что установлена скорость обмена 9600 кБод;
- выбрать из меню Open File и затем выбрать файл с кодами для загрузки;
- поскольку резидентный загрузчик микроконтроллера даёт компьютеру не более 5 с на установку соединения, следует сбросить микроконтроллер, а затем в течение не более 5 с жать на кнопку Connect;
- теперь последняя строчка в окне истории будет Connected;
- для запуска загрузки щёлкнуть по программной кнопке Start Bootloader;
- программа будет загружаться в память микроконтроллера. Загрузчик будет сообщать об ошибках, возникающих в процессе загрузки;
- сбросить микроконтроллер и начать исполнение загруженной программы.  
Наиболее общие функции загрузчика перечислены ниже:
- адрес аппаратного сброса находится в распоряжении резидентной программы загрузки;
- другая часть кода резидентного загрузчика занимает небольшую произвольную область памяти;
- проверяется необходимость начала загрузки нового кода;
- если нет необходимости загрузки нового кода, то запускается на исполнение имеющийся в памяти код пользователя;
- при загрузке новые данные принимаются по каналу связи;
- принятые данные записываются в программную память.  
Взаимодействие кода пользователя и резидентного загрузчика имеет некоторые особенности. Резидентная часть практически всегда использует адрес рестарта и некоторую дополнительную область памяти. Это – небольшой фрагмент кода, который не использует прерываний, таким образом, пользователю становится доступен вектор прерываний по адресу 0x0004. Резидент загрузчика должен избегать использования вектора прерывания, поэтому переход к остальной части программы загрузки должен производиться в пределах адресов от 0x0000 до 0x0003.

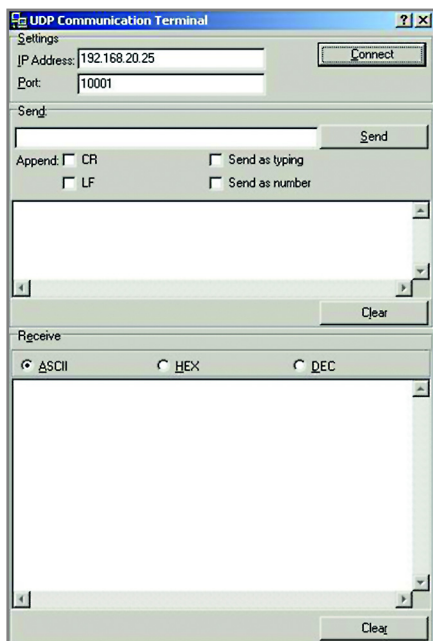


Рис. 20. Встроенный инструмент UDP Terminal

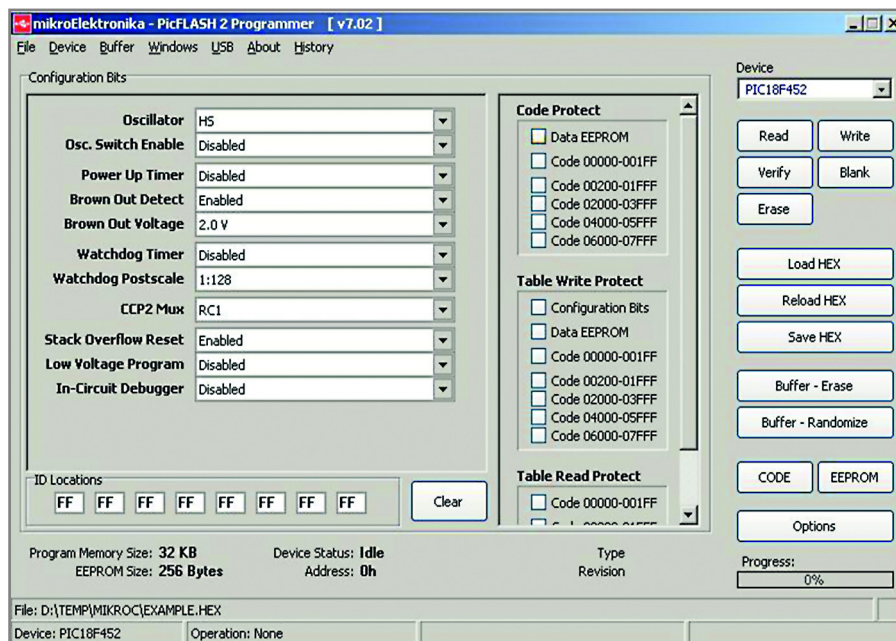


Рис. 21. Окно запуска программирования

Резидентный загрузчик должен программироваться в микроконтроллер с использованием обычного программатора, одновременно с битами конфигурации. Резидентный загрузчик не

позволяет менять биты конфигурации, поскольку они располагаются за пределами программной памяти.

Для отладки программ для USB-устройств существует встроенный

инструмент (см. рис. 19), запускаемый из выпадающего меню Tools → HID Terminal.

При создании программ для сетевых устройств с интерфейсом Ethernet очень полезным может оказаться встроенный инструмент UDP Terminal (см. рис. 20). Он запускается из выпадающего меню Tools → UDP Terminal и позволяет отлаживать программы, поддерживающие сетевой протокол UDP.

Созданную и отлаженную программу можно записать в память программ микроконтроллера непосредственно из среды разработки с помощью специального программатора, разработанного в компании MikroElektronika.

Программирование (см. рис. 21) запускается из выпадающего меню Tools → mE Programmer или с помощью клавиши F11. После программирования МК можно приступить к тестированию программы в реальном устройстве.

### Клавиатурные команды

В таблице 2 приводится список клавиатурных команд, используемых в mikroC. Данный список команд с описанием назначения на английском языке можно посмотреть в окне проводника кода (Code Explorer), на вкладке Keyboard.

*Продолжение следует*

### ЛИТЕРАТУРА

1. www.mikroe.com.
2. www.microchip.com.

Таблица 2. Список клавиатурных команд

| Клавиши                                   | Команды среды разработки                   |
|---|--|
| F1  | Помощь                                     |
| Ctrl+N                                    | Новый файл                                 |
| Ctrl+O                                    | Открыть файл                               |
| Ctrl+F9                                   | Компилировать проект                       |
| Ctrl+F11                                  | Включение/выключение проводника кода       |
| Ctrl+Shift+F5                             | Просмотр точек останова                    |
| <b>Основные команды редактирования</b>    |  |
| F3  | Найти, найти далее                         |
| Ctrl+A                                    | Выделить всё                               |
| Ctrl+C                                    | Копировать                                 |
| Ctrl+F                                    | Найти                                      |
| Ctrl+H                                    | Заменить                                   |
| Ctrl+P                                    | Печатать                                   |
| Ctrl+S                                    | Сохранить                                  |
| Ctrl+Shift+S                              | Сохранить как                              |
| Ctrl+V                                    | Вставить                                   |
| Ctrl+X                                    | Вырезать                                   |
| Ctrl+Y                                    | Повтор отменённого действия                |
| Ctrl+Z                                    | Отменить действие                          |
| <b>Расширенные команды редактирования</b> |  |
| Ctrl+Space                                | Помощь в коде (Code Assistant)             |
| Ctrl+Shift+Space                          | Помощь в параметрах (Parameters Assistant) |
| Ctrl+D                                    | Найти определение                          |
| Ctrl+G                                    | Перейти к строке номер                     |
| Ctrl+J                                    | Вставить шаблон кода                       |
| Ctrl+number                               | Перейти к закладке                         |
| Ctrl+Shift+number                         | Установить закладку                        |
| Ctrl+Shift+I                              | Отступ для выделенной части текста         |
| Ctrl+Shift+U                              | Отмена отступа для выделения части текста  |
| Alt+Select                                | Выбрать столбец                            |
| <b>Команды отладчика</b>                  |  |
| F4  | Выполнить всё до курсора                   |
| F5  | Включение/выключение точки останова        |
| F6  | Пуск/пауза программы в отладчике           |
| F7  | Шаг с заходом в функцию                    |
| F8  | Шаг без захода в функцию                   |
| Ctrl+F8                                   | Шаг с выходом из функции                   |
| F9  | Запуск отладчика                           |
| Ctrl+F2                                   | Сброс                                      |