Использование инструментария LPCXpresso для разработки приложений на базе 32-разрядных микроконтроллеров NXP с ядрами ARM Cortex-M0 и Cortex-M3

Часть 2

Павел Редькин (г. Ульяновск)

Статья посвящена программно-аппаратному обеспечению LPCXpresso, предназначенному для разработки и отладки приложений для 32-разрядных микроконтроллеров семейств LPC11xx/13xx/17xx производства NXP с ядрами ARM Cortex-M0, Cortex-M3.

Работа с проектами в LPCXpresso IDE

После завершения загрузки ПО GNU toolchain пользователь может перейти непосредственно к работе в LPCXpresso IDE, т.е. к созданию собственных проектов.

Каждой рабочей области в LPCXpresso IDE сопоставляется отдельный каталог на диске ПК. При этом ресурсы всех создаваемых пользователем в LPCXpresso IDE проектов по умолчанию будут храниться в каталоге той рабочей области, в которой они создавались. Указанный каталог может создаваться LPCXpresso IDE, например, в папке Documents and Settings\User\My Documents\lpcxpresso_3.6\workspace.

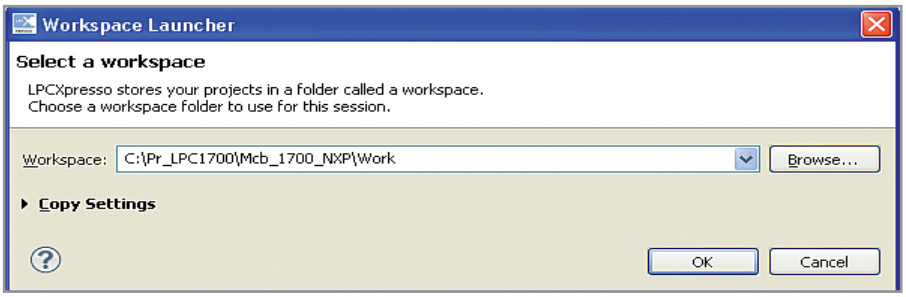

Рис. 4. Диалоговое окно выбора рабочей области *Workspace Launcher*

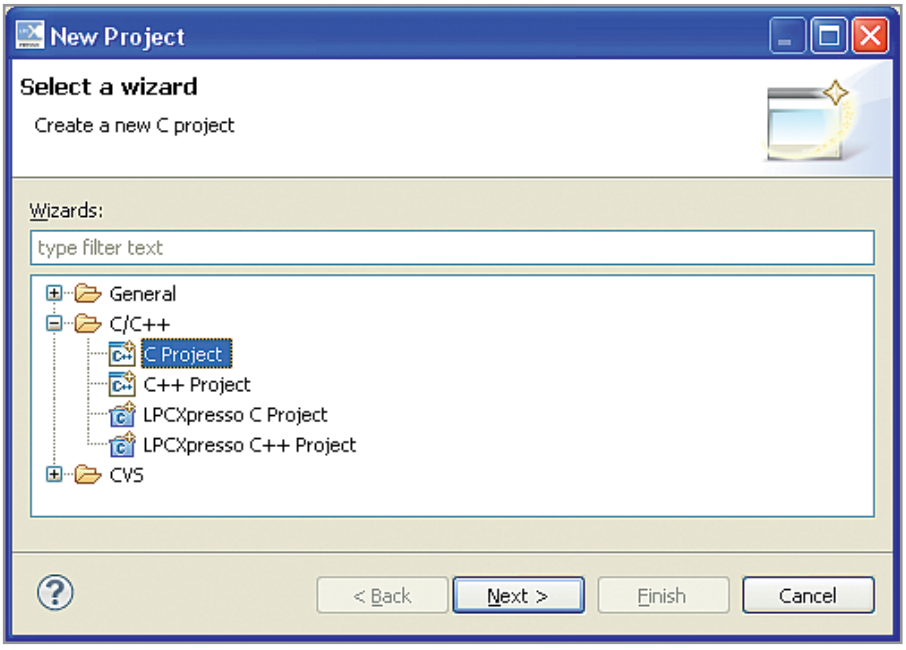

Рис. 5. Диалоговое окно выбора типа нового проекта *New Project*

Сохранение ресурсов в каталоге их рабочей области автоматически производится при выходе из LPCXpresso IDE.

При необходимости задания каталога рабочей области на диске ПК следует до начала создания проектов выбрать в меню перспективы *File > Switch Workspace > (Other...) >*, после чего откроется диалоговое окно *Workspace Launcher*, как показано на рисунке 4. В этом окне следует указать название каталога рабочей области и путь к нему. При каждом запуске LPCXpresso IDE окно *Workspace Launcher* будет автоматически открываться, предоставляя возможность выбора той рабочей области, которая будет загружена в LPCXpresso IDE.

Первое, что рекомендуется сделать в начале разработки приложения LPCXpresso IDE, – создать новый проект (project). Под проектом здесь понимается группа связанных между собой исходных и заголовочных файлов (для проекта на языке C – файлов с расширениями *.c и *.h), содержащих исходный текст встроенной управляющей программы для целевого МК.

Под приложением (application) в широком смысле здесь понимается совокупность всех файлов проекта, включая служебные и исполняемые файлы, генерируемые в ходе работы с проектом в LPCXpresso IDE. В более узком смысле приложение – это результат компиляции и компоновки проекта, представляющий собой встроенную управляющую программу.

Среда LPCXpresso IDE обеспечивает возможность создания проектов нескольких типов. Для создания проекта в меню перспективы следует выбрать *File > New > Project...*, после чего откроется диалоговое окно выбора типа нового проекта *New Project*, показанное на рисунке 5.

Допустим, мы создаём проект с исходным текстом на языке C, для построения которого предполагается ис-

пользовать любой из доступных в LPCXpresso IDE наборов инструментов. Выбираем в этом окне позицию папки C/C++, а в ней – позицию табуляции C *Project*, как показано на рисунке 5, и кликаем на кнопке *Next*. При этом открывается диалоговое окно задания типа, названия и местоположения нового проекта, а также выбора инструментария для его построения (Toolchains), показанное на рисунке 6. В этом окне необходимо задать название проекта (например, Pro_1) в соответствующем поле, а также указать месторасположение ресурсов проекта на диске ПК (Location) в случае, если предполагается их размещение не по умолчанию (отключена настройка *Use default location*). При включенной настройке *Use default location* ресурсы проекта будут размещаться внутри каталога рабочей области.

При создании проекта пользователь может выбрать один из следующих типов.

Executable – исполняемое приложение, папка этого типа проекта содержит следующие шаблоны проектов:

- *Empty Project* – проект, который не содержит никаких исходных файлов. Если выбрать этот шаблон, то будет создан проект, включающий только служебные файлы мета-данных, которые содержат параметры настройки проекта. Предполагается, что пользователь сам создаст исходные файлы и добавит их в проект. Для дальнейшего построения этого проекта, в поле *Toolchains* предлагается выбрать из списка доступных наборов инструментов (в нашем случае *Code Red MCU Tools* и *MinGW GCC*);
- *LPCXpresso Empty C Project* – проект, аналогичный предыдущему, однако для его построения доступен только инструментарий *Code Red MCU Tools*;
- *Hello World ANSI C Project* – простое приложение на языке C с названием *Hello World*, имеющее в своём составе главную функцию main(). Для его построения доступен только инструментарий *MinGW GCC*.

Shared Library – исполняемый модуль, который компилируется и компонуется отдельно. Когда создаётся проект, использующий *shared library* (libxx.so), CDT комбинирует объектные файлы таким образом, чтобы они были перемещаемыми и могли быть разделены многими процессами. Make-файл (makefile) проекта для этого типа генерируется автоматически. Заметим,

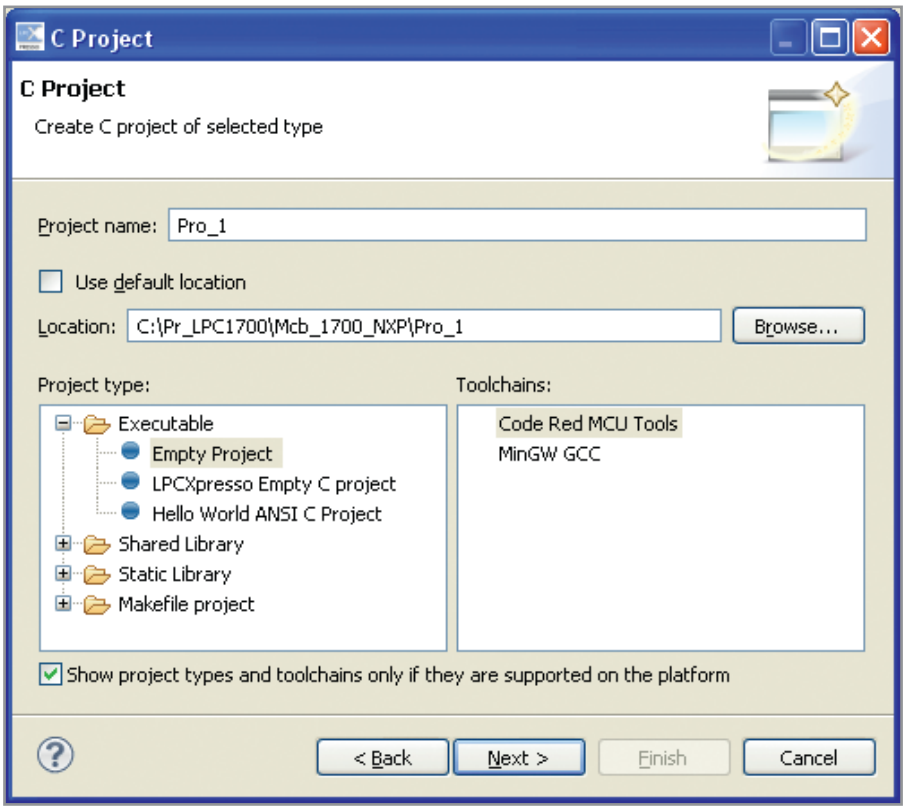

Рис. 6. Диалоговое окно задания основных параметров нового проекта на C

что make-файл любого проекта содержит набор скриптов, в которых указаны взаимозависимости файлов данного проекта и правила для их преобразования. На основе информации, содержащейся в make-файле, входящая в LPCXpresso IDE утилита make определяет и запускает на выполнение необходимые инструментальные программы (компилятор, компоновщик и пр.).

Static Library – коллекция объектных файлов, которые можно скомпоновать в составе другого приложения (libxx.a). CDT объединяет объектные файлы (*.o) в архив (*.a), который непосредственно компонуется в составе исполняемого приложения. Make-файл проекта для этого типа генерируется автоматически.

Makefile Project – пустой проект без файлов мета-данных. Выбор это-

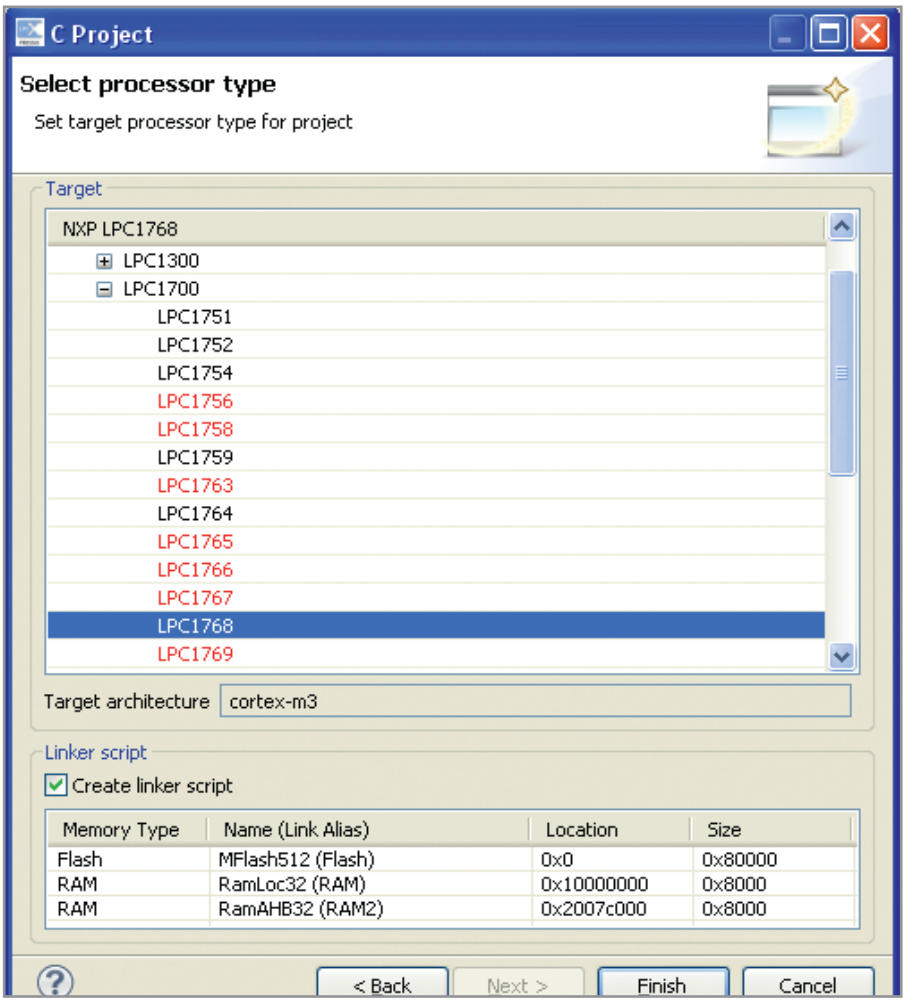

Рис. 7. Диалоговое окно выбора МК целевой системы проекта *Select processor type*

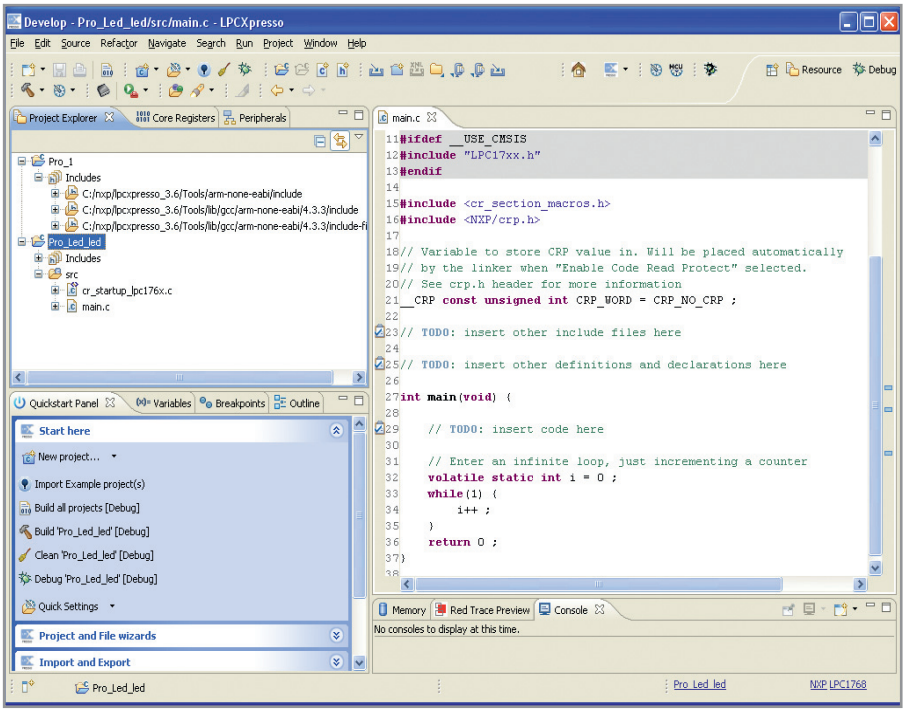

Рис. 8. Главное окно перспективы LPCXpresso IDE с проектами, отображаемыми в окне проводника проектов *Project Explorer*

го типа проекта может использоваться, чтобы импортировать и изменять существующие makefile-проекты. Make-файл проекта для этого типа автоматически не генерируется.

Задав тип, название, местоположение и инструментарий проекта, щёлкаем на кнопке *Next*, после чего открывается диалоговое окно выбора конфигурации компоновки проекта *Select Configurations*. В этом окне по умолчанию предлагается создать проект в двух конфигурациях компоновки: *Debug* и *Release* (отладка и выгрузка). Не изменяя заданные по умолчанию настройки, щёлкаем на кнопке *Next*, после чего открывается диалоговое окно выбора МК целевой системы проекта *Select processor type*, как показано на рисунке 7. В этом окне красным цветом указаны те МК, размер флэш-памяти которых превышает максимальный размер кода, генерируемого LPCXpresso IDE (в нашем случае 128 Кб). Выбрав МК целевой системы, щёлкаем на кнопке *Finish*, после чего позиция табуляции папки проекта появляется в окне проводника проектов *Project Explorer*, как показано на рисунке 8. Эта позиция может быть развёрнута.

При необходимости иметь в составе проекта создаваемый по умолчанию код запуска выбранного МК выбираем в окне выбора типа нового проекта *New Project* (см. рис. 5) позицию папки *LPCXpresso C Project* и щёлкаем на кнопке *Next*. Открывается диалоговое окно *New LPCXpresso C Project* (см. рис. 9), в котором можно выбрать семейство МК целевой системы. Для примера выбираем позицию *NXP LPC1700 C Project* и щёлкаем на кнопке *Next*, после чего откроется диалоговое окно задания названия нового проекта и пути к нему (*New Project*). Задав название (например, *Pro_Led_led*) и местоположение каталога размещения нового проекта, щёлкаем на кнопке *Next*, после чего открывается диалоговое окно выбора МК, аналогичное показанному на рисунке 7. Выбрав нужный МК, например LPC1768, щёлкаем на кнопке *Next*, после чего откроется диалоговое окно задания параметров создаваемого проекта, показанное на рисунке 10.

Параметр этого окна *Use CMSIS peripheral header files and initialization code* задаёт использование в качестве т.н. опорной информации (reference) нашего проекта заголовочных файлов и кода инициализации периферии из стандарта программного интерфейса МК Cortex CMSIS. Этот стандарт представляет собой набор файлов (модулей) с исходными и заголовочными текстами, описывающими встроенную периферию МК с ядром Cortex (регистры специальных функций), а также векторы исключений. Необходимо от-

метить, что для использования этой возможности в создаваемом проекте в нашу рабочую область должен быть предварительно импортирован библиотечный проект CMSIS из подкаталога *Examples*, созданного по умолчанию при инсталляции LPCXpresso IDE. Порядок импорта будет описан ниже.

Параметр *Enable CRP in the target image* задаёт включение защиты кода в памяти МК от чтения.

Название подкаталога в каталоге проекта, содержащего исходные файлы пользователя, может быть задано в поле *User source directory*. По умолчанию подкаталог называется *src*.

Задав параметры проекта, щёлкаем на кнопке *Finish*, после чего позиция табуляции папки проекта появляется в окне обозрения проводника проектов *Project Explorer* (см. рис. 8). Эта позиция может быть развёрнута. Созданный нами проект по умолчанию содержит файл кода запуска для выбранного МК cr_startup_lpc176x.c, а также файл исходного текста основной функции main.c с практически пустым главным циклом управляющей программы. Двойным щелчком мыши на позиции табуляции файла main.c можно отрыть его для просмотра и редактирования в окне редактора. Фактически этот файл представляет собой заготовку, в которую разработчик может вставить собственный исходный текст. При этом в файле main.c уже имеются корректные директивы подключения необходимых заголовочных и исходных файлов, связанных с выбранной аппаратной платформой и опорной информацией. После создания проекта можно вручную сохранить его на диске ПК, выбрав в меню перспективы *File* > *Save*, или комбинацией клавиш CTRL+S.

Следует отметить, что создание (импорт) проекта, а также другие связанные с ним операции (формирование, отладка и пр.), которые будут рассмотрены ниже, удобно производить с помощью открытого по умолчанию раздела *Start here* окна обозрения *Quickstart Panel* (см. рис. 8). При необходимости можно отрыть это окно, выбрав в меню перспективы *Window* > > *Show View* > *(Other...)* > *Quickstart* > > *Quickstart Panel*. В разделах этого окна содержатся и другие полезные команды, например, команда создания нового исходного файла на языке С в составе текущего проекта *Create New C source file* (раздел *Project and File wizards*),

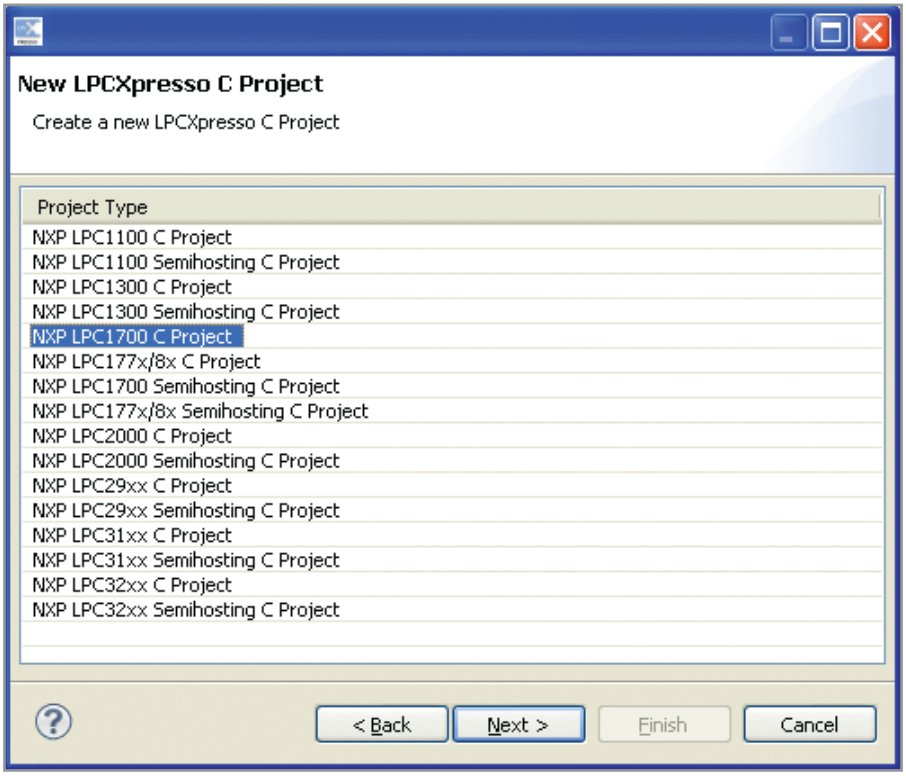

Рис. 9. Диалоговое окно *New LPCXpresso C Project* выбора семейства МК целевой системы

группа команд *Additional resources* (раздел *Extras*), позволяющих получить быстрый доступ к интернет-ресурсам пакета программ LPCXpresso.

Помимо создания проектов на базе различных шаблонов, LPCXpresso IDE предоставляет разработчику возможность импортировать в рабочую область уже готовые проекты, в частности, содержащиеся в самом пакете LPCXpresso IDE. Рассмотрим импорт в рабочую область библиотечного проекта CMSIS из подкаталога *Examples*, чтобы в рабочей области можно было создавать новые исполняемые проекты, поддерживающие CMSIS.

В окне обозрения *Quickstart Panel* > > *Start here* следует выбрать *Import Example project(s)*, после чего откроется одноимённое окно. В поле *Project*

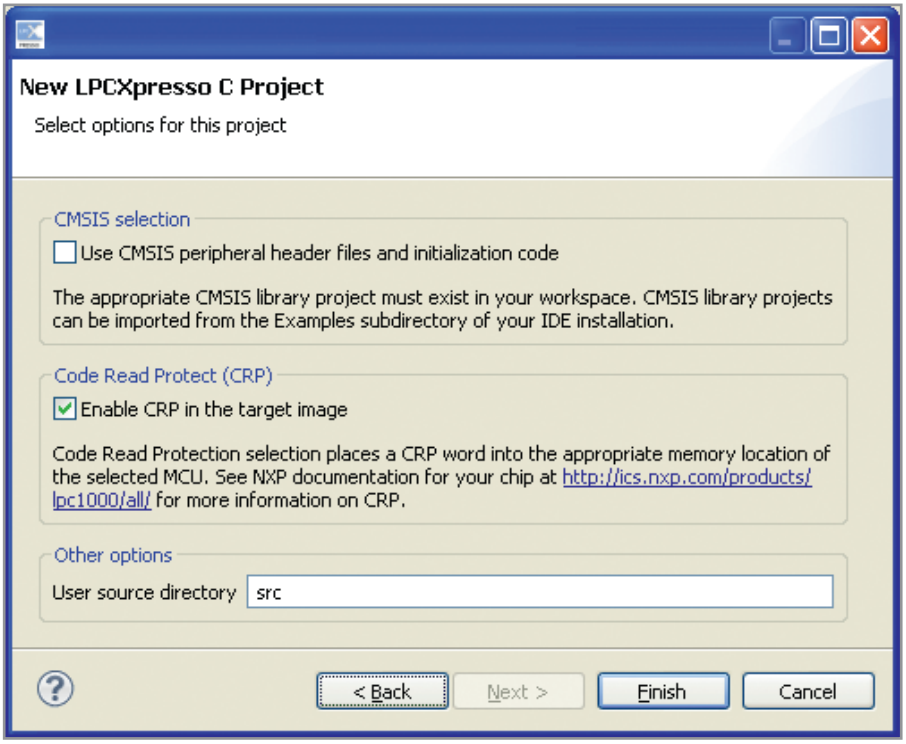

Рис. 10. Диалоговое окно задания параметров создаваемого проекта

arhive этого окна указываем путь к архиву с проектами, например: C:\nxp\lpcxpresso3.6\Examples\NXP\LPC1000\LPC17xx\CMSISv1p30_LPC17xx.zip, после чего щёлкаем на кнопке *Next*. После этого отрывается окно задания каталога для распаковки выбранного архива. Не изменяя заданных по умолчанию настроек, щёлкаем в этом окне на кнопке *Finish*, после чего позиция табуляции папки библиотечного проекта CMSISv1p30_LPC17xx появляется в окне проводника проектов *Project Explorer*. К числу заголовочных и исполняемых файлов поддержки CMSIS относятся файлы core_cm3.h, LPC17xx.h, system_LPC17xx.h, core_cm3.c, system_LPC17xx.c, содержащиеся в библиотечном проекте CMSISv1p30_LPC17xx.

После импорта библиотечного проекта CMSIS можно удалить созданный ранее исполняемый проект *Pro_Led_led*, щёлкнув на его позиции табуляции правой кнопкой мыши и выбрав в открывшемся контекстном меню команду *Delete*, а затем создать его заново, уже с поддержкой CMSIS (включенной параметром *Use CMSIS peripheral header files and initialization code* на рисунке 10). Заметим, что поддержка CMSIS существенно ускоряет и облегчает написание встроенных программ.

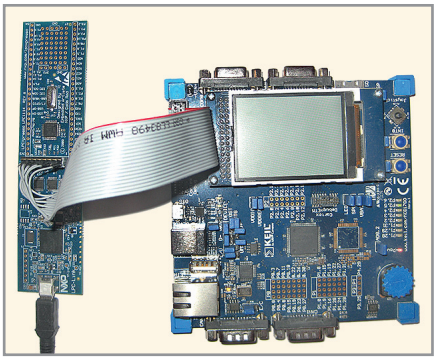

Рис. 11. Подключение демонстрационной платы LPCXpresso к отладочной плате MCB1760 через JTAG

После создания и сохранения проекта пользователь может запустить его формирование, выбрав в меню перспективы *File* > *Build Project*, или использовать комбинацию клавиш CTRL+B, когда позиция табуляции данного проекта выбрана курсором в окне обозрения *Project Explorer*. Для запуска формирования можно также воспользоваться соответствующей командой *Build* окна обозрения «быстрого запуска» *Quickstart Panel* > *Start here*.

Под формированием проекта здесь понимается процесс компиляции его исходных файлов, а также их последующей компоновки (линковки). Формирование проекта осуществляется CDT в последовательности, заданной в make-файле этого проекта. Результаты формирования и возникшие при этом проблемы можно наблюдать в окнах обозрений *Console* и *Problems* соответственно. Конечным результатом формирования (компоновки) проекта является двоичный исполняемый файл с расширением *.axf (*.elf), который по умолчанию генерируется в одной из автоматически создаваемых LPCXpresso IDE папок *Debug* или *Release* данного проекта.

После безошибочного завершения формирования проекта можно перейти к его выполнению/отладке. Поскольку в состав LPCXpresso IDE не входит программный симулятор целевого МК, производить выполнение/отладку проектов в LPCXpresso IDE можно только в «железе», т.е. в МК конечной системы. Однако прежде чем начать выполнение/отладку, создадим в управляющей программе нашего проекта некое функциональное наполнение, с тем чтобы можно было наглядно наблюдать её выполнение в МК.

При работе с LPCXpresso IDE автор статьи использовал демонстрационную плату LPCXpresso не как целевую систему, а только в качестве аппаратного отладчика LPC-Link. В качестве целевой системы использовалась отладочная плата MCB1760 фирмы Keil Software [4]. В связи с этим файл main.c созданного ранее проекта *Pro_Led_led* был дополнен несложным кодом, реализующим эффект последовательного зажигания и гашения («бегущий огонь») светодиодов этой платы.

При использовании режима отладки LPCXpresso IDE необходимо предварительно подключить к ПК аппаратный отладчик LPC-Link (плату LPCXpresso) с помощью кабеля USB 2.0 A/Mini-B. В свою очередь, к отладчику LPC-Link необходимо подключить целевую систему (в нашем случае отладочную плату MCB1760) через разъём JTAG, например, как показано на рисунке 11. За отсутствием в распоряжении автора миниатюрного десятиштырькового разъёма JTAG/SWD, отладчик LPC-Link был подключен к системе через один ряд контактов 16-штырькового разъёма J4 платы LPCXpresso. Принципиальные схемы разводки указанных разъёмов можно найти в [2].

Для запуска режима отладки следует выбрать в окне обозрения *Quickstart Panel* > *Start here* > *Debug*, после чего LPCXpresso IDE произведёт повторное формирование проекта. В случае обнаружения компилятором или компоновщиком каких-либо ошибок в исходном файле (файлах) проекта перечень этих ошибок отобразится в окне *Problems*, при этом информация

о каждой ошибке может быть детализирована.

В случае безошибочного завершения формирования программа начинает инициализацию отладчика LPC-Link с открытием соответствующего окна. При успешном завершении инициализации код программы приложения записывается во флэш-память МК, после чего программа LPCXpresso IDE переходит в режим отладки.

Информация о действиях, производимых LPCXpresso IDE и отладчиком LPC-Link над приложением и МК при переходе в режим отладки (формирование приложения, обнаружение и инициализация МК, запись кода приложения во флэш-память МК, верификация этого кода, сброс МК и т.д.), отображается в окне *Console*, как показано на рисунке 12.

Заметим, что отладку приложения удобнее производить не в открытой по умолчанию перспективе разработки *Develop*, а в перспективе отладки *Debug (Window > Open Perspective > > (Other...) > Debug)*. При переходе в режим отладки автоматически открывается окно обозрения *Debug*, как показано на рисунке 13. В ходе отладки в этом окне отображается следующая информация:

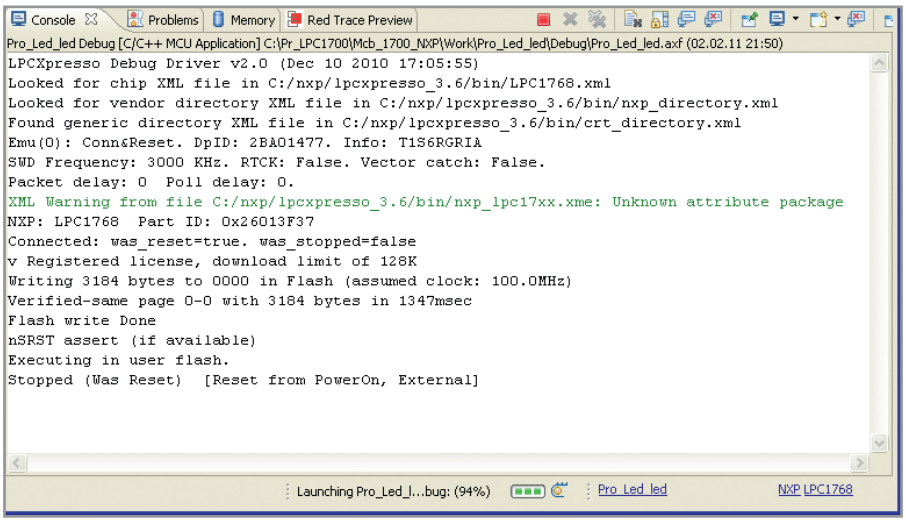

Рис. 12. Информация о действиях, производимых LPCXpresso IDE и отладчиком LPC-Link над приложением и целевым МК при переходе в режим отладки

- текущее состояние отладчика и программы приложения: выполнение (*Running*) или останов выполнения (*Suspended*). В состоянии останова в окне отображаются все функции программы, выполнение которых было приостановлено, например, на время передачи управления во вложенные в них функции. При наличии нескольких уровней вложения перечень таких функций представлен в окне в виде стека. Отображаемая в окне информация может быть развёрнута;
- каждый процесс, представленный в виде узла (символа и строки табуляции) в дереве процессов.

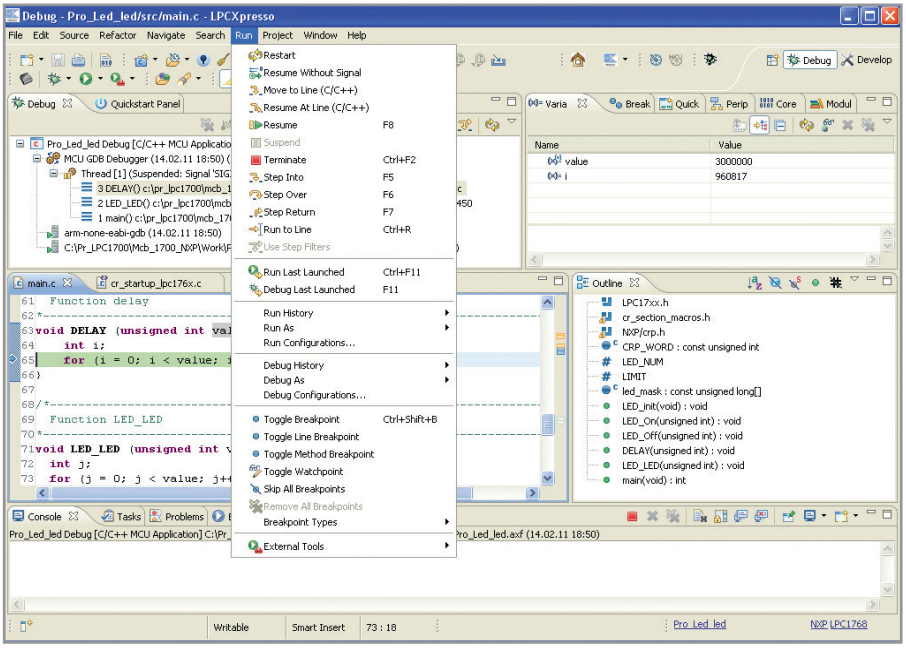

Рис. 13. Главное окно перспективы *Debug* в режиме отладки

Необходимо отметить, что многие возможности отладки в LPCXpresso IDE являются контекстно-зависимыми. Если созданная пользователем рабочая область содержит библиотечные проекты, например, типа CMSIS, то возможности отладки LPCXpresso IDE будут неактивны, когда пользователь будет редактировать исходный файл на языке С, который является частью такого библиотечного проекта.

В ходе отладки разработчик может получать информацию о значениях локальных переменных, содержащихся в любой функции отлаживаемой программы приложения. Для этого следует щёлкнуть мышью на позиции табуляции требуемой функции в окне *Debug*, после чего текущие значения содержащихся в ней переменных отобразятся в окне обозрения *Variables* (см. рис. 13). В режиме отладки команда исходного текста, на которую будет передано управление при следующем шаге выполнения программы, подсвечивается курсором в окне редактора. Набор команд управления отладкой доступен через пункт меню *Run* перспективы *Debug* (а также перспективы *Develop*). Кроме того, кнопки этих же команд расположены на инструментальной панели меню окна обозрения *Debug*.

Команда *Restart* вызовет сброс МК целевой системы, команда *Resume* (F8 клавиатуры ПК) – непрерывное выполнение программы приложения в реальном времени, команда *Run to Line* (Ctrl +R) – непрерывное выполнение программы приложения в реальном времени с остановом на строке исходного текста, отмеченной курсором в окне редактора.

В распоряжении пользователя имеются также команды, позволяющие выполнять программу приложения на уровне исходного текста в пошаговом режиме. Команда *Step Into* (F5) вызовет выполнение одного шага программы с «показом» захода управления в функцию, команда *Step Over* (F6) – выполнение одного шага без «показа» захода управления в функцию, команда *Step Return* (F7) вызовет завершение выполнения текущей функции (выход управления из нее) и останов выполнения. Подача команды *Suspend* приостановит выполнение программы приложения. Выполнение команды *Terminate* (Ctrl +F2) приведёт к выходу из режима отладки.

В ходе сеанса отладки пользователь может устанавливать и удалять в программе приложения точки останова, производить мониторинг значений программных переменных, содержимого, памяти, регистров и т.д.

Установка и удаление в программе точек останова осуществляются следующими способами:

- выбором в меню *Run* команд *Toggle Breakpoint*, *Toggle Line Breakpoint* (см. рис. 13). При этом точка останова будет установлена на команде или на строке исходного текста, отмеченной курсором в окне редактора. При передаче управления на команду (строку) программы приложения с установленной точкой останова произойдёт останов выполнения программы. При выборе в меню *Run* команды *Toggle Method Breakpoint* точка останова будет установлена на входе в функцию, содержащую строку исходного текста, которая отмечена курсором в окне редактора. Повторный выбор команд *Toggle Breakpoint*, *Toggle Line Breakpoint*, *Toggle Method Breakpoint* для отмеченной курсором команды (строки, функции) с установленной точкой останова приведёт к удалению этой точки;
- двойным щелчком левой кнопкой мыши на нужной строке в исходном тексте в окне редактора. При этом точка останова будет установлена на этой строке. Повторный двойной щелчок на строке с установленной точкой останова приведет к её удалению;
- заданием адреса в памяти, например, с помощью команды *Set address breakpoint* из меню окна обозрения *Quick Disassemble*.

Включение через меню *Run* настройки *Skip All Breakpoints* приведёт к отключению (но не удалению) всех ранее установленных точек останова. Можно снова включить точки останова, сняв указанную настройку.

Информация обо всех установленных точках останова с указанием их текущего состояния (включена/отключена) отображается в окне обозрения *Breakpoints*, как показано на рисунке 14. Это окно можно открыть, выбрав в меню *Window* > *Show View* > > *(Other...)* > *Breakpoints*. Для отключения точки останова следует щелчком мыши снять галочку из квадрата слева на позиции табуляции этой точки. Кроме того, для управления точками останова можно воспользоваться контекстным меню окна обозрения *Breakpoints*, которое открывается щелчком правой кнопки мыши. Установленная точка останова индицируется кружком с галочкой с левой стороны окна редактора напротив строки исходного текста. Установка/удаление, включение/отключение точек останова возможны как в режиме выполнения, так и в режиме останова выполнения программы.

Информация о текущем содержимом регистров ЦПУ целевого МК доступна разработчику в окне обозрения *Core Registers*; его можно открыть, выбрав в меню *Window* > *Show View* > > *(Other...)* > *Core Registers*. Позиции табуляции регистров, содержимое кото-

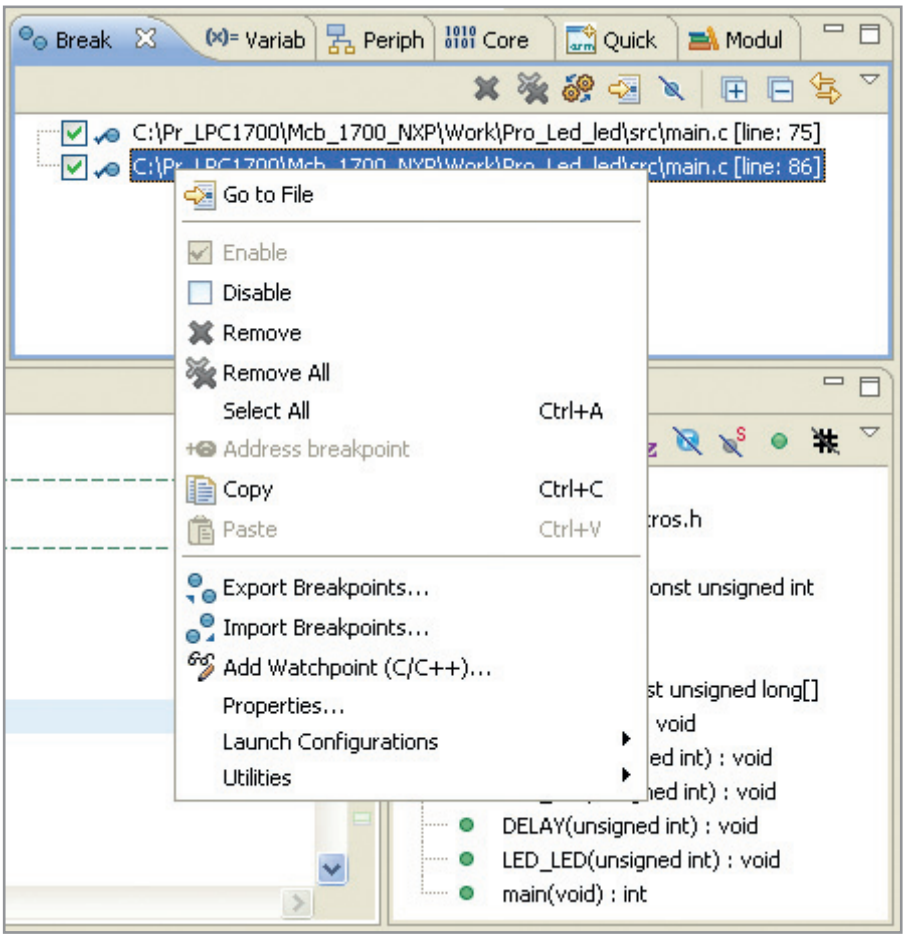

Рис. 14. Окно обозрения точек останова *Breakpoints*

рых обновлялось с момента последнего останова выполнения, подсвечиваются жёлтым цветом. Обновление текущего содержимого регистров производится программой только после останова выполнения.

Информация о текущем содержимом специальных (периферийных) регистров целевого МК доступна разработчику в окне обозрения *Peripherals*; оно открывается из меню *Window* > > *Show View* > *(Other...)* > *Peripherals*. Для отображения содержимого специальных регистров необходимо в окне *Peripherals* отметить галочками позиции табуляции требуемых периферийных устройств. После этого побитовое содержимое специальных регистров, относящихся к выбранной периферии, отобразится в автоматически открывшемся окне обозрения *Memory*.

При необходимости наблюдения содержимого памяти МК следует открыть окно обозрения *Memory*, выбрав в меню *Window* > *Show View* > *(Other...)* > > *Memory*, а затем создать т.н. монитор наблюдения требуемой области памяти, щёлкнув на пиктограмме команды *Add Memory Monitor* в меню обозрения *Memory*. В качестве альтернативы можно использовать контекстное меню окна, для открытия которого следует щёлкнуть в окне правой кнопкой мыши. После этого откроется диалоговое окно *Monitor Memory*, в поле которого *Enter address or expression to monitor*: необходимо задать начальный адрес в памяти и щёлкнуть на кнопке OK. После этого в области *Monitors* окна *Memory* отобразится созданный монитор памяти (в виде заданного начального адреса), а в области данных этого окна – содержимое памяти, начиная с заданного адреса. Следует отметить, что в ходе сессии отладки можно одновременно открыть несколько окон обозрения *Memory*, используя их для отображения содержимого различных областей памяти.

Возможности программы LPCXpresso IDE также позволяют разработчику наблюдать дизассемблированный текст, соответствующий загруженному в целевой МК коду приложения. Для этого необходимо открыть окно обозрения просмотра дизассемблированного текста через меню *Window* > *Show View* > *(Other...)* > *Disassembly*. Во время останова выполнения в этом окне отображается фрагмент дизассемблированного текста участка кода приложения, в пределах которого находится управление. В режиме отладки ассемблерная команда исходного текста в окне *Disassembly*, на которую будет передано управление при следующем шаге выполнения программы, подсвечивается курсором. Помимо ассемблерных инструкций, в дизассемблированном тексте также отображаются соответствующие этим инструкциям команды исходного текста на языке С.

Фрагмент доступного для наблюдения дизассемблированного текста, отображаемый в ходе отладки в окне *Disassembly* после каждого останова выполнения программы, весьма невелик. Однако пакет LPCXpresso IDE предоставляет разработчику дополнительные возможности наблюдения и анализа дизассемблированного текста. Для этого необходимо сначала открыть окно обозрения «быстрого» просмотра дизассемблированного текста, выбрав в меню *Window* > *Show View* > *(Other...)* > *Quick Disassemble*, затем уже в меню этого окна выбрать команду *Show disassemble dialog*, после чего откроется диалоговое окно *Disassemble*. Кроме того, можно использовать контекстное меню окна *Quick Disassemble*, для открытия которого следует щёлкнуть в окне правой кнопкой мыши. В поле *Start address or function* окна *Disassemble* следует задать начальный адрес в памяти (или название функции), с которого будет осуществляться «быстрое» отображение дизассемблированного текста программы, в поле *Length* – требуемый размер фрагмента дизассемблированного кода в байтах, и щёлкнуть на кнопке OK. После этого в окне *Quick Disassemble* отобразится дизассемблированный текст, соответствующий заданному участку памяти приложения.

Продолжение следует

Литература

1. Development Tools for ARM-based microcontrollers – Select from the best in support. August 2010. http://ics.nxp.com/literature/other/microcontrollers/pdf/arm.mcu.tools.pdf.
2. LPCXpresso. Getting started with NXP LPCXpresso. User guide. Rev. 7. 15 September 2010 (готовится к выпуску).
3. *Редькин П.П.* 32-битные микроконтроллеры NXP с ядром Cortex-M3 семейства LPC17xx. Полное руководство. Додэка-XXI, 2011.
4. http://www.keil.com/mcb1700/mcb1760.asp.