Микроконтроллеры AT32UC3 с ядром AVR32

(часть 4)

Павел Редькин (г. Ульяновск)

Четвёртая часть статьи рассказывает о существующих инструментальных средствах разработки, отладки и программирования пользовательских приложений на базе МК с ядром AVR32.

Разработка приложений на базе МК семейства AVR32

На момент написания этой статьи автору было известно о двух программных инструментальных средствах разработки/отладки приложений на базе МК с архитектурой AVR32. Первое из них – программный пакет интегрированной среды разработки/отладки (IDE) IAR Embedded Workbench® for AVR32 от фирмы IAR. Альтернативным средством разработки/отладки является пакет IDE AVR32Studio от фирмы Atmel.

Помимо инструментальных средств разработки/отладки, фирма Atmel свободно распространяет большое количество исходных текстов примеров, а также программных драйверов узлов и модулей МК AVR32. Указанная информация для каждого семейства МК типа AVR32 может быть бесплатно скачана с сайта www.atmel.com в виде архивных файлов AVR32-SoftwareFramework. В этих архивах содержатся, в частности, следующие компоненты:

- приложения и сервисы для МК семейства AVR32 с примерами их использования:
  - библиотека AVR32 DSPLib, содержащая набор цифровых функций обработки сигналов,
  - файловая система FAT12/16/32,
  - исходные коды ядра ОС freeRTOS.org и пример демонстрационного приложения под freeRTOS.org,
  - пример организации стека lwIP TCP/IP для МК типа AT32UC3 с контроллером MACB,
  - рекомендации по организации различных интерфейсов памяти (в том числе, с файловой системой),
  - пример построения устройства USB CDC,
  - пример использования USB-загрузчика для обновления встроенного программного обеспечения с помощью внутрисистемного программирования (ISP) МК,
  - пример построения устройства USB HID («мыши»),
  - пример построения запоминающего устройства/хоста USB Mass Storage,
  - программные драйверы системных устройств и встроенной периферии МК типа AVR32 с примерами их использования;
- программные драйверы внешних устройств МК AVR32 с примерами их использования:
  - четырёхстрочный символьный LCD-дисплей DIP204,
  - манипулятор типа «джойстик»,
  - микросхемы памяти архитектуры Dataflash семейства AT45DB производства Atmel,
  - карты памяти SD/MMC,
  - микросхемы памяти SDRAM фирмы Micron MT48LC16M16A2TG-7E (с использованием интерфейса внешней памяти EBI);
- командные файлы компоновщиков и файлы макроопределений для компиляторов IAR и GCC (последний используется в AVR32Studio), а также заголовочные файлы для всей встроенной периферии семейства AVR32.
- заголовочные файлы для оценочных плат с МК типа AVR32 производства Atmel.

Исходные тексты и описание сложного демонстрационного приложения «Панель управления» (Control Panel) на базе оценочной платы Atmel EVK1100, выполняющегося под ОС freeRTOS.org. Приложение реализует сбор данных с помощью встроенных средств оценочной платы и их вывод через различные коммуникационные интерфейсы, поддерживаемые МК; Control Panel использует большинство вышеперечисленных программных драйверов и сервисов.

Многие примеры и программные драйверы из AVR32-SoftwareFramework представлены исходными текстами на языке С в двух вариантах: для компиляторов IAR и GCC.

Пакет программ IDE Embedded Workbench

Пакет IDE IAR Embedded Workbench for AVR32 обеспечивает полную поддержку всех МК семейств AP7000 и UC3 с архитектурой AVR32. В пакет IDE включены готовые файлы конфигурации устройств и примеры проектов для оценочных плат Atmel EVK1100 и STK1000.

Поддержка IAR Embedded Workbench эмулятора Atmel AVR32 JTAGICE-mkII включает в себя поддержку трассировки режима NanoTrace, загрузчика флэш-памяти, аппаратных и программных точек останова. Поддержка IAR Embedded Workbench RTOS для AVR32 включает поддержку интерфейса OSEK Run Time Interface (ORTI).

Файл дистрибутива пакета IDE IAR Embedded Workbench for AVR32 доступен для бесплатного скачивания на сайте производителя www.iar.com в двух демонстрационных версиях: с ограниченным объёмом выходного бинарного файла (32 Кб), но без ограничения времени использования, и без ограничения объёма кода, но с ограниченным временем использования (30 дней).

Пакет включает в себя набор инструментальных средств, которые интегрированы в единую программную оболочку с удобным оконным интерфейсом, работающую под ОС Windows Microsoft. Пакет IAR Embedded Workbench for AVR32 состоит из следующих компонентов:

- собственно IDE (текстовый редактор, менеджер проектов и т.д.);

- компилятора IAR C/C++;
- ассемблера IAR;
- универсального компоновщика IAR XLINK Linker;
- программы построения библиотек IAR XAR Library Builder;
- набора библиотек IAR XLIB Librarian;
- отладчика IAR C-SPY Debugger с симулятором;
- файлов конфигурации для всех устройств AVR32;
- контекстно-зависимой интерактивной справки.

Работа пользователя с пакетом IDE IAR Embedded Workbench во многом схожа с работой с пакетом IDE IAR Embedded Workbench ARM (IAR EWARM) [4]. Для получения информации по IAR Embedded Workbench рекомендуется обратиться к [5].

Разработка приложения в IDE начинается с создания нового проекта. Перед созданием проекта пользователь должен создать (или использовать созданную по умолчанию) в IDE рабочую область. При создании проекта пользователь последовательно задаёт его имя, тип, параметры и функции (тип МК целевой системы и аппаратной платформы, конфигурацию компоновки и т.д.). В процессе работы с проектом к нему можно добавлять различные файлы (исходные, заголовочные, подключаемые и т.п.). Затем созданный проект подвергается компиляции и компоновке. При этом создаётся файл листинга компилятора, файл карты компоновщика, а также другие выходные файлы, тип и наличие которых определяется заданными для проекта настройками компоновщика.

После успешного завершения компиляции и компоновки проект может быть отлажен. Встроенный в IDE высокоуровневый отладчик *IAR C-SPY Debugger* обеспечивает возможность одновременной разработки и отладки приложения. Наиболее простой способ отладки заключается в использовании программного драйвера симулятора C-SPY Debugger. Симулятор программно моделирует все функции объектного процессора, что позволяет отладить приложение до его записи в целевую систему. Поскольку никакие аппаратные средства при этом не требуются, указанный способ отладки является самым рентабельным решением для многих приложений. Симулятор *C-SPY Debugger* может имитировать прерывания ЦПУ, а также периферийные устройства (с использованием системы макрокоманд C-SPY). Симулятор также позволяет моделировать трассировку, проверять корректность доступа к памяти, использовать точки останова, а также производить мониторинг памяти и регистров МК.

Отладчик *C-SPY Debugger* наряду с отладкой в режиме симуляции может быть использован для отладки непосредственно в «железе», т.е. в составе целевой системы [6]. Отладка в целевой системе осуществляется с помощью аппаратного драйвера (кабеля), обеспечивающего физическую связь с целевой системой. Используемая автором версия IAR Embedded Workbench поддерживает только один тип аппаратного драйвера *C-SPY Debugger – JTAGICE-mkII* производства Atmel. Кабель JTAGICE-mkII подключается к МК целевой системы через встроенный интерфейс JTAG, как показано на структурной схеме, изображённой на рисунке 14.

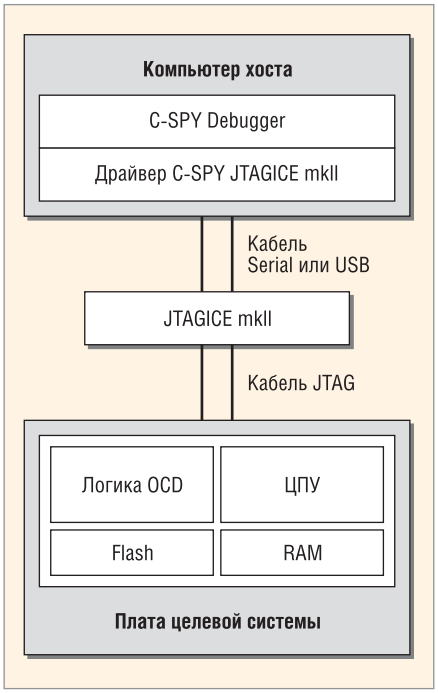

Рис. 14. Блок-схема подключения JTAGICE-mkII к МК целевой системы

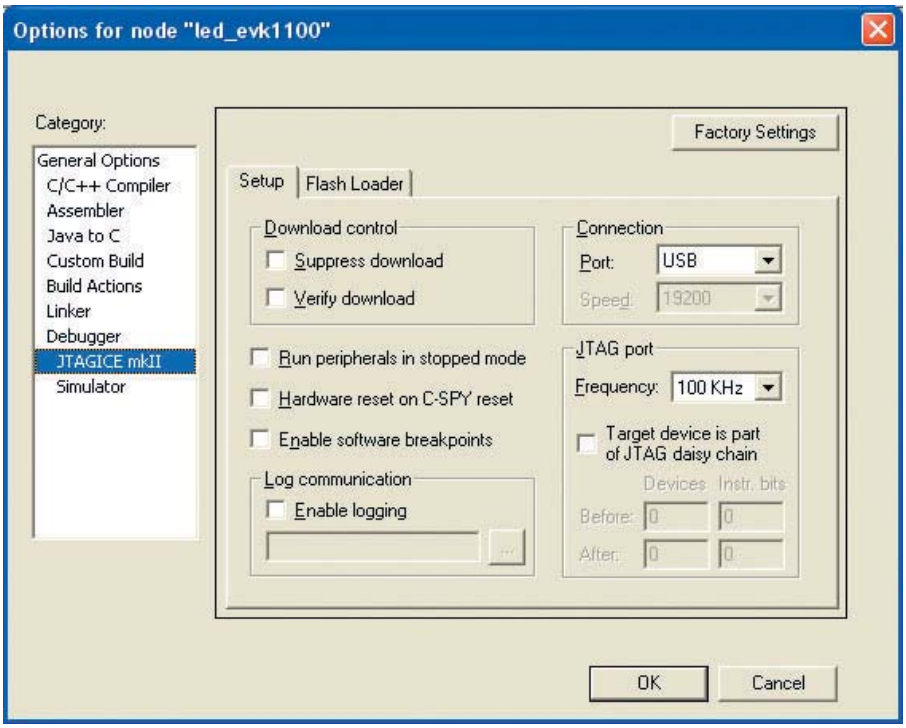

Рис. 15. Выбор опций управления загрузкой

В начале сеанса отладки пользовательское приложение по умолчанию автоматически загружается во флэш-память МК, однако в случае необходимости эта процедура может быть отменена. Перед запуском сеанса отладки в целевой системе необходимо задать в проекте установки, касающиеся отладки. В частности, эти установки определяют выбор драйвера на странице *Setup* категории *Debugger* (следует выбрать *JTAGICE mkII*), а также выбор параметров управления загрузкой на странице *Setup* категории *JTAGICE mkII*, показанной на рисунке 15.

Как и в случае программной симуляции, отладчик *C-SPY* обеспечивает возможность трассировки приложения в целевой системе. Данные трассировки при этом будут поступать в хост-компьютер через *JTAGICE-mkII* и отображаться в окне трассировки, как показано на рисунке 16.

При отладке в целевой системе *C-SPY* позволяет установить в программе до шести аппаратных точек останова или неограниченное количество программных точек останова (параметр *Enable software breakpoints* на странице *Setup* категории *JTAGICE mkII*). Точки останова данных при отладке в целевой системе недоступны.

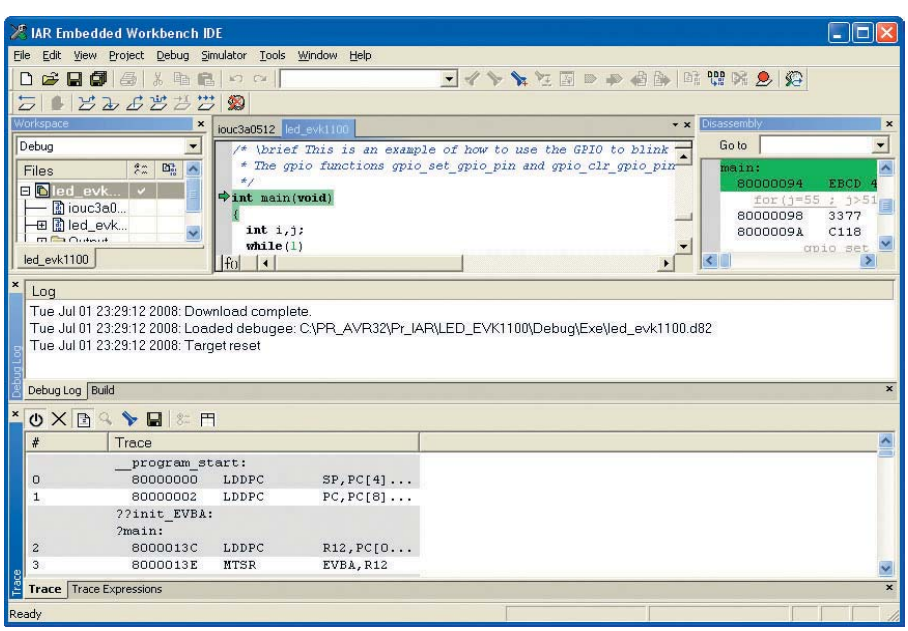

Рис. 16. Разработка проекта в IDE IAR Embedded Workbench for AVR32

Пакет программ IDE AVR32Studio

Программный пакет IDE AVR32Studio 2.0, который использовал автор, поддерживает разработку автономных (без ОС) приложений для всех устройств семейств AT32AP7000 и AT32UC3, а также приложений под ОС Linux для устройств AT32AP7000. Пакет AVR32Studio также поддерживает все инструментальные средства производства Atmel, разработанные для поддержки архитектуры AVR32, включая JTAG-адаптеры JTAGICE-mkII и AVR ONE!, а также отладочные платы EVK1100, EVK1101, NGW100, STK1000, STK1002 и STK600.

Пакет IDE AVR32Studio содержит законченный набор компонентов, в число которых входят:

- менеджер проектов с конкурентной системой версий (CVS);
- редактор языка C/C++ с поддержкой выделения синтаксиса и завершённого кода;
- отладчик, поддерживающий управление пошаговым выполнением на уровне команд исходного кода, а также точки останова;
- средства мониторинга регистров, памяти и подсистем ввода-вывода МК;
- средства конфигурирования и управления целевой системой.

Как можно видеть из этого перечня, собственного компилятора C в сос-

таве AVR32Studio нет. Поэтому для формирования и отладки проектов AVR32Studio использует дополнительные программные инструментальные средства из пакета AVR32 GNU Toolchain (компилятор, набор необходимых вспомогательных программ и библиотек). Компилятор GNU C (GCC) используется в AVR32Studio для компиляции программ C/C++, а отладчик GNU (GDB) – для отладки приложения в целевой системе.

Таким образом, для обеспечения возможности подготовки в AVR32Studio исходных кодов на языке C/C++, на хост-компьютер необходимо установить не только дистрибутив IDE AVR32Studio (файл AVR32Studio-2.x.x-Setup.exe), но и дистрибутив AVR32 GNU Toolchain (файл avr32-gnu-toolchain-2.x.x.exe). Оба этих файла доступны для бесплатного скачивания на интернет-странице www.atmel.com. Пакет AVR32Studio предлагается для скачивания в двух версиях: под ОС Linux и под ОС Windows.

Подобно пакету IDE IAR Embedded Workbench, при работе в пакете IDE AVR32Studio используется понятие *рабочая область*. Каждая рабочая область содержит одну или более *перспектив*. Перспективы, в свою очередь, содержат *обозрения* и *редакторы*, доступ к которым возможен через соответствующие меню и инструментальные панели. На рабочем столе одновременно может существовать несколько окон рабочей области.

Перспектива определяет начальный набор и размещение обозрений в окне рабочей области. В пределах окна каждая перспектива совместно использует один и тот же набор редакторов. Каждая перспектива предоставляет пользователю набор функциональных возможностей, нацеленных на выполнение определённого типа задач, или работает с определёнными типами ресурсов. Например, перспектива C/C++ объединяет обозрения, которые обычно используются при редактировании исходных файлов, а перспектива *Debug* содержит обозрения, которые используются при отладке программ. При работе с различными проектами в рабочей области пользователь может производить переключение перспектив.

Перспективы могут содержать собственные меню и инструментальные панели. Они определяют видимые наборы действий, которые пользователь может задействовать, чтобы настроить перспективу. Для открытия требуемой перспективы в окне рабочей области следует выбрать в главном меню *Window* > *Open Perspective*. Чтобы увидеть список всех поддерживаемых перспектив, следует выбрать *Window* > *Open Perspective* > *Other...*, после чего откроется окно *Open Perspective*, как показано на рисунке 17.

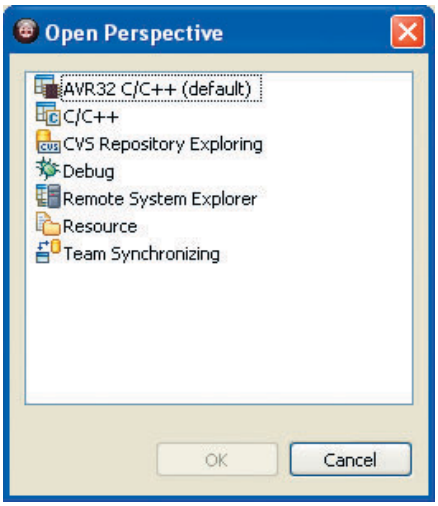

Рис. 17. Окно выбора перспективы

Обозрения поддерживают редакторы и обеспечивают альтернативные представления, а также способы управления информацией в рабочих областях. Например, проводник проектов (*Project Explorer*) и другие навигационные обозрения управляют отображением проектов и других ресурсов, с которыми работает пользователь. Обозрения также имеют свои собственные меню. Чтобы сделать это меню видимым, следует кликнуть кнопкой мыши на иконке в левой части области заголовка обозрения. Некоторые обозрения также имеют свои собственные инструментальные панели.

Вид окна рабочей области AVR32Studio показан на рисунке 18. По умолчанию в нём открывается перспектива *AVR32 C/C++*, в которой, в свою очередь, открыты окна нескольких обозрений; каждое окно обозрения содержит собственную инструментальную панель и набор окон, по умолчанию расположенных в виде «стека».

Первое, что следует сделать в начале разработки приложения AVR32, – создать новый проект. Для этого в меню *File* окна рабочей области следует выбрать пункт *New...*, а затем выбрать тип создаваемого проекта, как показано на рисунке 19. Если выбрать один из явно предлагаемых в меню типов, то по умолчанию будет создан проект вида *Managed make*, который не предусматривает самостоятельное управление со стороны пользователя make-файлом проекта. При этом откроется окно создания нового проекта, в котором необходимо заполнить поля названия проекта, типа платформы (МК семейства AVR32) и типа проекта.

Возможно создание нового проекта на языке C/C++ «с нуля» (*AVR32 C Project*) на базе одного из имеющихся в AVR32Studio шаблонов (*AVR32 C Project from a template*), а также на базе одного из имеющихся в AVR32Studio примеров (*Project... > Examples*). При выборе примера автоматически производится и выбор аппаратной платформы, поскольку каждый из предлагаемых примеров сопоставлен в списке конкретной платформе – отладочной плате производства Atmel. После создания нового проекта на жёстком диске компьютера будет создан каталог с выбранным именем проекта. В этот каталог можно помещать исходные файлы проекта и открывать их в окне редактора AVR32Studio, выбирая в меню *File > Open File* и используя соответствующие перспективы. Как уже упоминалось, пакет AVR32Studio обеспечивает поддержку редактирования кода, написанного на C/C++.

Каждый вновь создаваемый проект должен быть сформирован (скомпонован). Система управления формированием AVR32Studio имеет два операционных режима: внутреннее (непосредственное) формирование (*Internal build*) и управление формированием (*Managed make*). Режим Managed make автоматически создаёт make-файл для формируемого проекта, используя задаваемые пользователем параметры настройки формирования. Указанный режим является наиболее типичным и задаётся по умолчанию.

Создание проекта с непосредственным формированием производится путём выбора в меню *File > New... > Project...* с последующим выбором нужного мастера проекта, как показано на рисунке 20.

Формирование запускается выбором в меню *Project > Build Project* или нажатием на клавиши Ctrl+B, когда файл проекта открыт в окне редактора. Результаты формирования можно видеть в окнах *Console* и *Problems* обозрения перспективы *Debug*.

Для задания в проекте целевой системы необходимо выбрать в меню окна рабочей области *Window > Show view > (Other... > AVR32 Development) > AVR32 Targets*, после чего откроется окно целевых систем *AVR32 Targets* (см. рис. 18). Кликнув в нём правой кнопкой мыши, следует запустить функцию *Scan Targets* из контекстного меню. Предварительно JTAG-адаптер и отладочная плата должны быть подключены к компьютеру. Когда пакет AVR32Studio обнаружит подключенную к компьютеру целевую систему, информация о ней отобразится в окне *AVR32 Targets*. В случае отсутствия подключенной целевой системы AVR32Studio считает таковой программный симулятор, информацию о котором отображает в окне, как показано на рисунке 18. Пользователь может также вручную добавить в список новую целевую систему (*New Target*), кликнув на кнопке *Creates a new target* в меню окна *AVR32 Targets*. Свойства (компоненты) целевой системы из списка можно просмотреть и отредактировать, поместив на её позицию в окне курсор и нажав на клавишу Enter.

В зависимости от конфигурации имеющейся целевой системы, пакет AVR32Studio поддерживает выполнение с ней различных операций (запись, чтение, верификация, стирание и т.д.). Все доступные для целевой системы операции отображаются в её контекстном меню, которое активизируется кликом правой кнопкой мыши на позиции целевой системы в окне *AVR32 Targets*.

Поддерживаемые пакетом AVR32Studio возможности по конфигурированию запуска позволяют задавать параметры запуска отладки (*Debug*) или параметры запуска выполнения

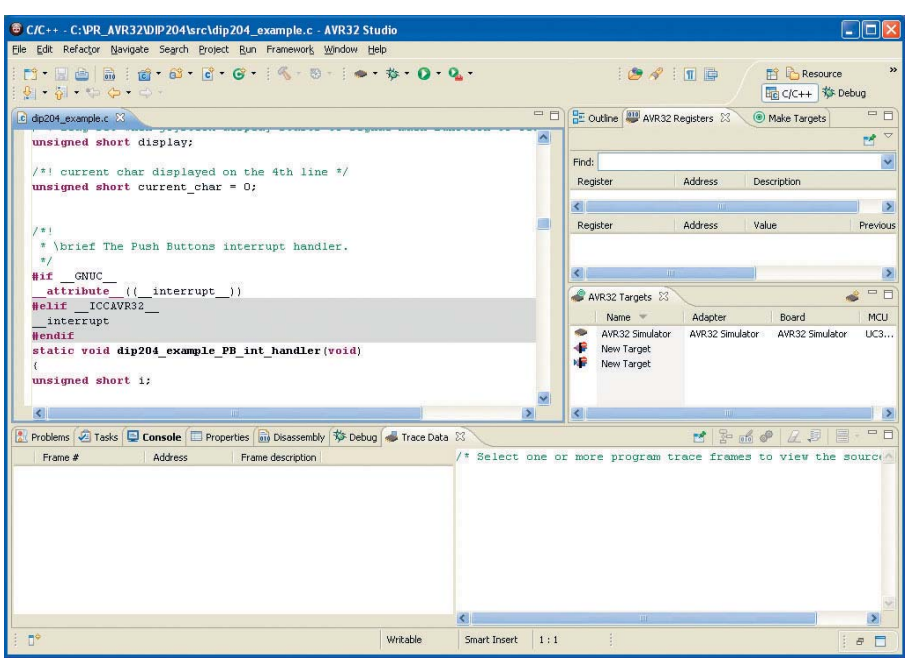

Рис. 18. Вид окна рабочей области AVR32Studio

(*Run*) программы. Запуск отладки может использоваться для подключения отладчика GDB к целевой системе и осуществления трассировки, в то время как запуск выполнения инициирует простое выполнение бинарного файла «прошивки» в МК целевой системы. Диалоговое окно задания параметров запуска отладки открывается при выборе в главном меню *Run* > *Open Debug Dialog...* (см. рис. 21), а диалоговое окно задания параметров запуска выполнения – *Run* > *Open Run Dialog...*

Конфигурация запуска отладки/выполнения включает в себя четыре раздела:

- главную конфигурацию;
- конфигурацию отладчика;
- конфигурацию трассировки;
- общую конфигурацию.

Каждому из этих разделов в диалоговом окне посвящена отдельная страница. Чтобы эти страницы стали доступными, необходимо сначала задать приложение для отладки/выполнения. Для этого следует выбрать курсором тип приложения в левой части окна, как показано на рисунке 21, а затем кликнуть на иконке *New launch configuration* инструментальной панели окна. При этом создаётся новая конфигурация запуска отладки/выполнения, для которой откроются страницы конфигурации, как показано на рисунке 22.

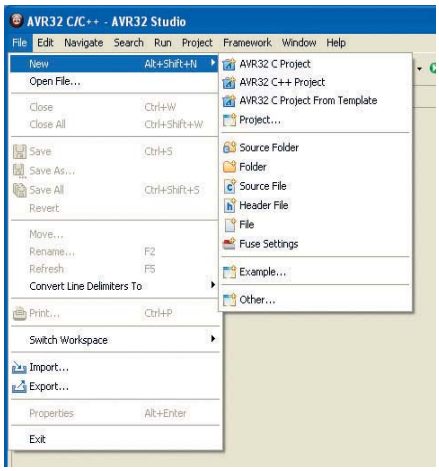

Рис. 19. Выбор типа создаваемого проекта

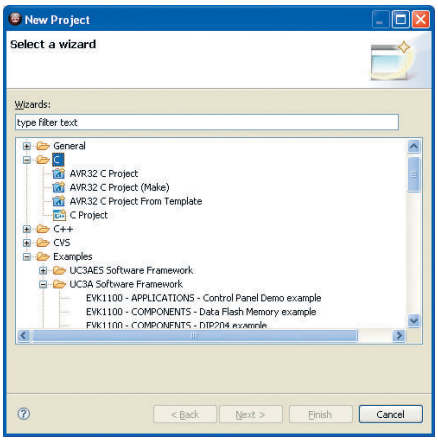

Рис. 20. Создание проекта с непосредственным формированием

Первая страница в диалоге конфигурации запуска *Maim* содержит главные параметры настройки. Здесь пользователь должен определить конфигурацию запуска для проекта, тип двоичного файла, который должен быть загружен в целевую систему, а также используемый проводник запуска.

Страница конфигурации отладчика *Debugger* содержит параметры настройки отладчика.

Страница общей конфигурации *Common* используется для задания общих параметров конфигурации запуска.

Страница конфигурации трассировки Trace содержит параметры настройки трассировки. Чтобы разрешить трассировку, пользователь должен включить параметр *Enable trace*.

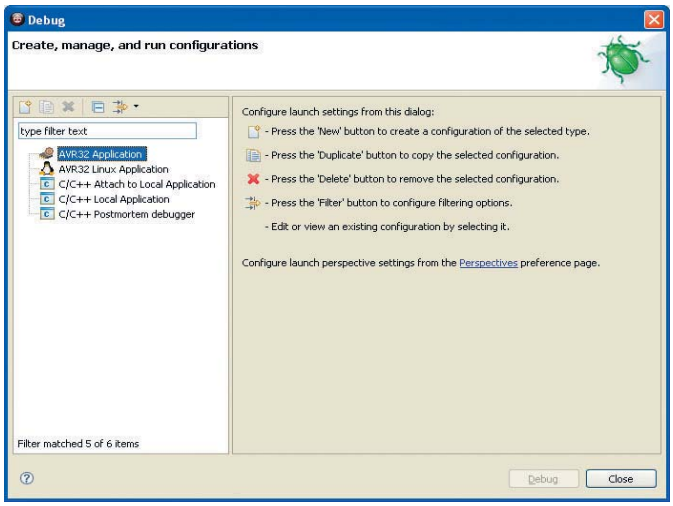

Рис. 21. Диалоговое окно задания опций запуска отладки

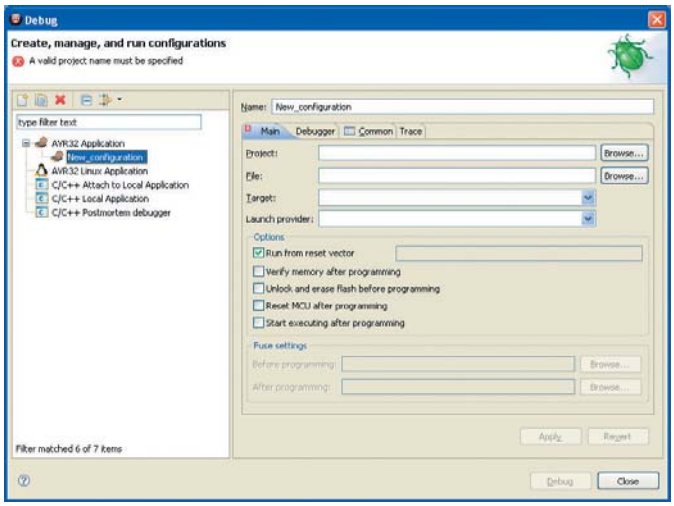

Рис. 22. Страницы конфигурации отладки/выполнения приложения

Кроме того, необходимо выбрать используемый метод трассировки из двух имеющихся: *NanoTrace* и *Auxiliary Trace*. Запуск трассировки производится из обозрения трассировки или посредством задания точки трассировки, как будет показано ниже.

Рис. 23. Комплект адаптера AVR ONE!

Рис. 24. Комплект адаптера JTAGICE-mkII

Рис. 25. Отладочная плата EVK1100

Под точками трассировки (tracepoints) понимается адрес в памяти или глобальная переменная в исходном файле, при осуществлении заданного типа доступа к которым происходит заданное событие, связанное с трассировкой: старт/стоп трассировки или генерация сообщения трассировки.

Точки трассировки могут быть установлены в двух различных обозрениях. При редактировании кода в редакторе C/C++ пользователь может добавить строку точки трассировки, нажав на нужной строке кода Ctrl+Alt+T. При этом откроется окно настроек, содержащее меню настройки *Tracepoint*. Та же самая операция будет выполнена, если в обозрении *Disassembly* ввести адрес точки трассировки. Строка исходного теста, на которой создана точка трассировки, будет подсвечиваться в окне редактора. Нажатие Ctrl+Alt+T на строке, где уже существует точка трассировки, приведёт к её удалению.

Поле *Tracepoint type* в меню *Tracepoint* используется для задания вида трассировки. Как трассировка данных, так и трассировка программы могут быть запущены триггером трассировки. Когда задана трассировка данных, пользователь может также задать, какая область памяти подлежит трассировке. Поле Trace operation используется для задания операции трассировки, автоматически выполняемой при срабатывании точки трассировки.

Для открытия обозрения регистров МК целевой системы необходимо выбрать в меню *Window* > *Show view* > *(Other...)* > *AVR32 Registers*, после чего откроется окно *AVR32 Registers* (см. рис. 18). Обозрение регистров позволяет производить мониторинг всех регистров и битовых полей из области, используемой в активном проекте или принадлежащей текущему сеансу отладки. Помимо мониторинга, возможно также редактирование содержимого регистров. Выбрав курсором позицию нужного регистра, можно редактировать содержимое его битов. Двойной клик на позиции регистра в дереве его отображения или нажатие клавиши Enter приведёт к вызову редактора. Значения отображаемых регистров будут изменяться всякий раз при приостановке ЦПУ.

Аппаратные инструментальные средства

Для разработки/отладки/программирования приложений на базе МК с архитектурой AVR32 фирма Atmel предлагает ряд аппаратных инструментальных средств собственного производства. Это, прежде всего, JTAG-адаптеры JTAGICE-mkII и AVR ONE!, а также отладочные платы EVK1100, EVK1101, NGW100, STK1000, STK1002 и STK600.

Наиболее мощным и функционально полным аппаратным отладочным средством для МК семейства AVR32 является адаптер AVR ONE!. Он поддерживает отладку на кристалле, ISP-программирование, а также Nexus AUX-трассировку всех выпускаемых в настоящее время МК типа AVR32, и совместим с пакетом программ IDE AVR32Studio. Версия программной прошивки AVR ONE! 1.1, предлагаемая производителем на момент написания этой статьи, поддерживает отладку (программирование) только

через два интерфейса МК AVR32: JTAG и Nexus 2.0 (IEEE-ISTO 5001-2003).

Необходимо отметить, что адаптер AVR ONE! не является эмулятором. Принцип его работы основан на интерфейсе со встроенной системой отладки (OCD) МК целевой системы, что и обеспечивает механизм контроля выполнения пользовательского кода. В режиме выполнения кода ЦПУ этот процесс полностью независим от адаптера. Однако AVR ONE! при этом непрерывно контролирует МК целевой системы, чтобы отслеживать, не произошло ли условие останова. Когда это условие наступает, система OCD опрашивает ЦПУ микроконтроллера через его интерфейс отладки, что даёт возможность хост-компьютеру получить через адаптер информацию о текущем состоянии МК. Адаптер AVR ONE! поддерживает как программные, так и аппаратные точки останова.

Внешний вид комплекта адаптера AVR ONE! показан на рисунке 23. Адаптер подключается к хост-компьютеру через порт USB. К недостаткам адаптера AVR ONE! можно отнести его высокую стоимость.

Адаптер JTAG типа JTAGICE-mkII представляет собой более простое и доступное средство отладки. Он совместим как с пакетом IDE AVR32Studio, так и с пакетом IDE AVRStudio версии 4.09 и выше. Адаптер JTAGICE-mkII поддерживает отладку и ISP-программирование всех МК семейства AVR32, а также всех МК с архитектурой AVR (имеющих JTAG), но только через интерфейс JTAG, совместимый со стандартом IEEE 1149.1. Помимо этого, адаптер поддерживает отладку и ISP-программирование всех МК, имеющих интерфейс debugWIRE.

По принципу функционирования адаптер JTAGICE-mkII может быть отнесён к традиционным эмуляторам. Он использует интерфейс JTAG для организации эмуляции реального времени в МК целевой системы, в то время как в МК выполняется пользовательский код. Протокол AVR-отладки на кристалле (AVROCD) предоставляет пользователю законченную систему управления внутренними ресурсами МК. Однако при этом адаптер JTAGICE-mkII поддерживает только программные точки останова.

Адаптер JTAGICE-mkII может быть подключен к хост-компьютеру через COM-порт (поддерживает спецификацию RS-232) или через порт USB (поддерживает спецификацию USB 1.1). В последнем случае возможно питание адаптера от хост-компьютера. Внешний вид комплекта адаптера JTAGICE-mkII показан на рисунке 24. Для своего питания адаптер JTAGICE mkII может использовать внешний источник или шину USB, но не источник целевой системы. По умолчанию выбирается внешнее электропитание.

Для построения приложений на базе МК семейства AT32UC3 фирма Atmel предлагает отладочные платы EVK1100 и EVK1101. Внешний вид платы EVK1100 показан на рисунке 25. Помимо МК типа AT32UC3A0512 (в корпусе QFP144) плата содержит следующие элементы аппаратной «обвязки»: микросхему памяти SPI Dataflash AT35DB642 ёмкостью 8 Мбит, микросхему SDRAM MT48LC16M16A2 ёмкостью 32 Мбит, символьный четырёхстрочный ЖКИ типа EA-DIP204B-4NLW, датчики освещённости, температуры и аналоговый потенциометр, подключенные к входам модуля АЦП микроконтроллера; три кнопки, манипулятор типа «джойстик», подключенный к входам сканирования внешней клавиатуры МК; шесть светодиодов, три подключенных к МК кварцевых резонатора (12 МГц, 12 МГц и 32 768 Гц). Интерфейсы МК выведены через соответствующие разъёмы: два девятивыводных RS-232, mini A-B USB, RJ45 Ethernet, слот SD/MMC (SPI), JTAG, NEXUS, шину EBI, порты ввода-вывода общего назначения и т.д. Имеющиеся на плате стабилизаторы напряжения позволяют осуществлять её питание от шины USB или от внешнего источника постоянного напряжения 8...20 В.

Отладочная плата EVK1101 содержит МК типа AT32UC3B0256 (в корпусе LQFP64) и несколько сокращённый по сравнению с EVK1100 набор аппаратной «обвязки». Отладочные платы NGW100, STK1000 и STK1002 построены на базе МК семейства AT32AP7000.

Литература

1. AVR®32 32-Bit Microcontroller AT32UC3A0512, AT32UC3A0256, AT32UC3A0128, AT32UC3A1512, AT32UC3A1256, AT32UC3A1128 Preliminary. 32058C-AVR32-10/07, http://www.atmel.com.
2. AVR®32 32-Bit Microcontroller AT32UC3B0256, AT32UC3B0128, AT32UC3B064, AT32UC3B1256, AT32UC3B1128, AT32UC3B164 Preliminary. 32059E-AVR32-12/07, http://www.atmel.com.
3. http://www.avr32.ru.
4. *Редькин П.П.* Микроконтроллеры ARM7 семейства LPC2000. Руководство пользователя (+CD). Додэка-XXI, 2007.
5. IAR Embedded Workbench® IDE. User Guide, http://www.iar.com.
6. AVR32 IAR C-SPY® Hardware Debugger System. User Guide for Atmel® Corporation's AVR JTAGICE mkII, http://www.iar.com.