Разработка и отладка проектов на базе МК LPC2000

Павел Редькин (г. Ульяновск)

Предлагаемая вниманию читателей статья является логическим продолжением ранее опубликованного цикла, описывающего архитектуру и способы программирования микроконтроллеров (далее – МК) семейства LPC2000. В этой статье будет рассказано о порядке и способах разработки-отладки пользовательских проектов на базе МК LPC2000 в интегрированной среде IDE IAR Embedded Workbench™ (IAR EWARM) с компилятором Си от фирмы IAR. Эта широко распространённая отладочная среда является мощной альтернативой конкурирующей среды µVISION с компилятором Си от компании Keil Elektronik. Помимо этого статья содержит подробную информацию о способах записи в МК LPC2000 пользовательских программ.

Общее описание IDE IAR Embedded Workbench™

Программная среда IAR EWARM [1, 2] относится к числу сред, регулярно обновляемых производителем (на момент написания этой статьи вышла версия 4.41A), имеет простые интуитивно понятные правила работы и поддерживает в числе прочих широко распространённый аппаратный отладочный драйвер JTAG Wiggler, использование которого будет описано ниже. Бесплатные демоверсии IAR EWARM в двух вариантах – с ограниченным сроком действия (1 месяц), но без ограничения объёма компилируемого кода, и с неограниченным сроком действия, но с ограниченным объёмом кода (32 К), – доступны на сайте www.iar.com. IAR EWARM позволяет создавать законченные прикладные проекты на базе 8-, 16- и 32-разрядных МК различных производителей и в том числе на базе МК с ядрами ARM. IDE включает в себя набор инструментальных средств, интегрированных в единую программу-оболочку с удобным оконным интерфейсом, работающую под Windows Microsoft:

- компилятор IAR ARM C/C++,
- ассемблер IAR ARM,
- универсальный компоновщик IAR XLINK Linker™,
- программа построения библиотек IAR XAR Library Builder™,
- набор библиотек IAR XLIB Librarian™,
- текстовый редактор для подготовки исходных текстов программ,
- менеджер проектов,
- утилита построения командной строки,
- отладчик языка высокого уровня IAR C-SPY Debugger™.

Компилятор, ассемблер и компоновщик могут запускаться на выполнение не только из оболочки IDE, но и из командной строки, поддержка которой обеспечивается соответствующей утилитой IAR EWARM. Запуск из командной строки применяется в случае, когда есть необходимость использовать компилятор, ассемблер и компоновщик как внешние инструментальные средства в уже установленной проектной среде.

IDE IAR Embedded Workbench поддерживает развитые функции управления проектами, дающие возможность пользователю управлять всеми проектными модулями, например, файлами исходного текста на C или C++, ассемблерными файлами, подключаемыми файлами и другими связанными модулями. Файлы могут быть сгруппированы с различными опциями, заданными на уровне всего проекта, группы или только файла. Сам проект в целом, а также относящиеся к нему группы файлов и отдельные файлы называются узлами. При этом проект в целом является узлом самого верхнего уровня, а отдельные файлы – узлами самого низкого уровня. Под проектным модулем понимается файл.

Для управления проектами IDE предоставляет следующие основные средства и возможности:

- шаблоны для создания проектов;
- иерархическое представление проекта;
- браузер исходного файла с иерархическим символьным представлением;
- установка опций глобально, для групп исходных файлов или для индивидуальных исходных файлов;
- утилита Make, которая перетранслирует, повторно ассемблирует и компонует файлы, когда это необходимо;
- текстовые файлы проектов;
- утилита Custom Build, разворачивающая стандартный инструментальный набор простым способом;
- командная строка, формирующая проектный файл на входе.

Структура IAR Embedded Workbench™

В ходе инсталляции IDE IAR Embedded Workbench на жёстком диске компьютера создаётся несколько каталогов, содержащих различные типы файлов, используемых IDE. Ниже приводится описание файлов, содержавшихся по умолчанию в каждом каталоге.

Корневой каталог, создаваемый по умолчанию инсталляционной процедурой, находится по пути:

x:\Program Files\IAR Systems\ Embedded Workbench 4.n\ <директории IDE>,

где *x* – диск, на котором установлена операционная система Windows Microsoft, *4.n* – номер версии IDE IAR Embedded Workbench.

В каталоге *Embedded Workbench 4.n* содержатся два подкаталога: common и ARM.

Подкаталог *common* содержит подкаталоги компонентов, общих для всех программ IDE IAR Embedded Workbench™.

Подкаталог *common\bin* содержит исполняемые файлы компонентов,

общих для всех программ IDE IAR Embedded Workbench, типа IAR XLINK Linker, IAR XLIB Librarian, IAR XAR Library Builder, редактор и графический пользовательский интерфейс. Здесь также расположен исполняемый файл IAR Embedded Workbench.

Подкаталог *common\config* содержит файлы, используемые IAR Embedded Workbench для того, чтобы поддерживать параметры настройки IDE.

Подкаталог *common\doc* содержит справочные файлы (readme) с дополнительной информацией о компонентах, общих для всех программ IAR Embedded Workbench™. Каталог также содержит интерактивную версию руководства *IAR Linker and Library Tools Reference Guide* в формате PDF.

Подкаталог *common\src* содержит исходные файлы для компонентов, общих для всех программ IAR Embedded Workbench™ в выходном формате SIMPLE.

Подкаталог common\plugins содержит исполняемые файлы и файлы описаний для компонентов, которые могут быть загружены как сменные модули.

Подкаталог *ARM* содержит подкаталоги, определяемые спецификой продуктов ARM, поддерживаемых IAR Embedded Workbench.

Подкаталог *arm\bin* содержит исполняемые файлы для специфических компонентов ARM, типа компилятора ARM IAR C/C++, ARM IAR ассемблера и драйверов IAR C-SPY™.

Подкаталог *arm\config* содержит файлы, используемые для разработки проектов, например:

- шаблоны командного файла компоновщика (**.xcl*),
- параметры настройки МК (**.i79*),
- файлы описания устройства C-SPY (**.ddf*),
- синтаксическая «окраска» файлов конфигурации (**.cfg*),
- шаблоны для прикладных и библиотечных проектов (**.ewp*), соответствующие им библиотечные файлы конфигурации.

Подкаталог *arm\doc* содержит справочные файлы (readme) с дополнительной информацией об инструментальных средствах ARM. Каталог также содержит интерактивные версии нескольких руководств по специфическим компонентам ARM в формате PDF.

Подкаталог *arm\inc* содержит файлы, рекомендуемые для использования в качестве подключаемых, типа файлов заголовка для стандартных библиотек C или C++. В подкаталоге содержатся также заголовочные файлы, определяющие регистры специальных функций (SFR) МК. Эти файлы используются компилятором и ассемблером.

Подкаталог *arm\lib* содержит предварительно подготовленные библиотеки и библиотеки файлов конфигурации, используемые компилятором.

Подкаталог *arm\plugins* содержит исполняемые файлы и файлы описания для компонентов, которые могут быть загружены как сменные модули.

Подкаталог *arm\src* содержит исходные файлы для некоторых библиотечных функций с перестраиваемой конфигурацией и примеры кода приложений. Этот каталог также содержит исходный код библиотек.

Подкаталог *arm\drivers* содержит дистрибутивы драйверов JTAG-интерфейса (например, Wiggler), которые могут быть проинсталлированы на компьютер хоста и позволяют производить отладку пользовательских проектов и загрузку пользовательских программ в память МК целевых пользовательских систем.

Подкаталог *arm\tutor* содержит файлы демонстрационных обучающих проектов, которые могут использоваться для освоения пользователями работы с IDE.

Подкаталог *arm\examples* содержит примеры работающих проектов на базе МК различных производителей и семейств.

Создание проектов приложений в IDE

Параметры, используемые для настройки нового проекта

IDE IAR Embedded Workbench позволяет создавать простые и расширенные проектные модели. При создании расширенной проектной модели сначала создаётся *рабочая область*, к которой добавляется один или несколько *проектов*. В IDE имеются готовые *шаблоны проектов* как для прикладных, так и для библиотечных проектов. Каждый проект может содержать иерархию из *групп*, в которые собираются *исходные файлы* пользователя. Для каждого проекта может быть определена одна или несколько *конфигураций компоновки*.

Поскольку рассмотренный ниже пример приложения является простым проектом с малым количеством файлов, для него нет необходимости в расширенной проектной модели.

Производитель рекомендует, чтобы при создании нового проекта пользователь создавал отдельный каталог, где сохранялись бы все файлы этого проекта. Прежде чем создать проект, необходимо сначала создать рабочую область (окно рабочей области).

Создание окна рабочей области

Первый шаг при создании нового проекта заключается в создании новой рабочей области. При запуске IAR Embedded Workbench первый раз имеется уже готовая автоматически созданная рабочая область, которую сразу же можно использовать для проектов. Если разработчик использует эту рабочую область, то первый шаг можно игнорировать:

1. Выбрать в меню *File>New* и выбрать Workspace в диалоговом окне *New*. «Кликнуть» мышью на кнопке *OK*, после чего отобразится пустое окно рабочего рабочей области;
2. Чтобы сохранить созданную рабочую область, необходимо выбрать в меню *File>Save Workspace*, а затем задать, где должен быть сохранён файл рабочей области, набрав его имя в поле *File name* и нажав кнопку *Save*. По умолчанию этот файл сохраняется в каталоге *projects*. Файл рабочей области будет иметь расширение имени файла *eww*. Этот файл содержит перечисление всех проектов, которые пользователь добавляет к данной рабочей области. Информация, связанная с текущим сеансом, типом и размещением окон и контрольных точек, содержится в файлах, создаваемых в каталоге *projects\settings*. Теперь можно создать сам проект и добавить его к рабочей области.

Следует заметить, что сохранить рабочую область вышеописанным способом можно будет только после создания в ней хотя бы одного нового проекта (см. ниже).

Создание нового проекта

Создание нового проекта производится в такой последовательности:

1. Выбрать в меню *Project>Create New Project*. Появившееся диалоговое окно *Create New Project* предоставляет возможность использования в качестве основы нового шаблона проекта. В общем случае можно выбрать шаблон *Empty project*, ко-

торый просто создаёт проект, использующий заданные по умолчанию параметры настройки;
2. В поле *Tool chain* выбрать *ARM* и «кликнуть» на кнопке *OK*;
3. В появившемся стандартном диалоговом окне *Save As* следует определить, где требуется сохранить файл проекта, например, в каталоге *projects*. В поле *File name* следует написать имя файла проекта, например, *project1,* и нажать на кнопку *Save*, чтобы завершить создание нового проекта. Проект появится в окне рабочей области.

По умолчанию создаются две конфигурации компоновки: отладки и выгрузки (*Debug* и *Release*). В приведённом далее примере будет использоваться только конфигурация отладки *Debug*. Конфигурация компоновки выбирается из раскрывающегося меню в верхней части окна рабочей области. Звёздочка на имени проекта указывает, что произведённые изменения не были сохранены.

Файл проекта с расширением имени файла *ewp* теперь создан в каталоге *projects*. Этот файл содержит информацию о параметрах настройки проекта, например, таких, как вариант компоновки.

Добавление файлов к проекту

Любой мало-мальски сложный проект, как правило, содержит не один, а несколько файлов: исходный текст основной программы, исходные тексты поддержки аппаратных модулей, заголовочные файлы к ним и т.д.

Возможность создания нескольких групп позволяет организовать исходные файлы, входящие в состав проекта, в соответствии с их содержимым или согласно иным потребностям пользователя. В данном проекте будет только два файла, поэтому нет никакой необходимости создавать группу.

Добавление файлов к проекту производится в такой последовательности:
1. В окне рабочей области курсором выбрать объект, к которому планируется добавить исходный файл, Это может быть группа или, как в нашем случае, непосредственно проект;
2. Выбрать в меню *Project>Add Files*, после чего откроется стандартное диалоговое окно обзора файлов и каталогов. Поочерёдно выбираем файлы из списка и нажимаем кнопку *Open*. После этого выбранные файлы добавляются к проекту *project1*.

Таблица 1. Глобальные параметры настройки для проекта *project1*

Страница окна	Параметр: значение
Target	Core: ARM7TDMI-S
Output	Output file: Executable
Library Configuration	Library: Normal

Таблица 2. Параметры настройки компилятора C/C++ для проекта *project1*

Страница окна	Параметр: значение
Code	Optimizations, Size: None (Best debug support)
Output	Generate debug information
List	Output list fileAssembler mnemonics

Задание опций проекта

Для прикладных пользовательских проектов опции могут быть заданы для узлов проекта всех уровней. Сначала производится задание генеральных (общих) опций, например, таких как конфигурация процессора. Эти опции должны быть одинаковыми для всей структуры конфигурации, поэтому они устанавливаются для узла уровня проекта.

Задание опций проекта производится в такой последовательности:
1. Выбирается позиция табуляции нужного проекта, например, *project1 – Debug* в окне рабочей области, а в меню выбирается *Project>Options*. После этого откроется страница *Target* в категории *General Options*. На страницах *General Options* следует проверить установки параметров настройки, приведённые в табл. 1. Необходимо заметить, что задание типа ядра МК (например, ARM7TDMI-S) опцией *Core* в поле *Processor variant* производится только в случае, если создаваемый проект предназначен, например, для программной симуляции без последующего переноса его в память целевой пользовательской системы. В случае, если проект создается под конкретную модель МК, можно непосредственно задать нужное устройство опцией *Device*, выбрав его из открывающегося списка. После установки глобальных опций следует установить опции компилятора для проекта.
2. Выбрать *C/C++ Compiler* в списке *Category*, с тем чтобы отобразить страницы группы опций компилятора.
3. Проверить установки параметров настройки, приведённые в табл. 2.
4. «Кликнуть» на кнопке *OK*, чтобы установить значения, которые были выбраны.

На этом этапе создание нового проекта можно считать законченным.

Компилирование файлов приложения

После создания проекта приложения следует его откомпилировать и скомпоновать. При этом создаётся файл листинга компилятора и файл карты компоновщика.

Компилирование исходных файлов производится в такой последовательности:
1. Чтобы откомпилировать, например, файл *Utilities.c*, следует выбрать его в окне рабочей области;
2. Выбрать в меню *Project>Compile*. Альтернативно можно нажать на кнопку *Compile* в инструментальной панели или выбрать команду *Compile* из контекстного меню, которое появляется, если щёлкнуть правой кнопкой мыши на выбранном файле в окне рабочей области. Процесс и результаты компиляции будут отображаться в окне сообщений *Build*;
3. Откомпилировать файл *Tutor.c* тем же самым способом.

По завершении компиляции IAR Embedded Workbench создаст новые подкаталоги в каталоге проекта. Поскольку нами используется конфигурация компоновки *Debug*, то в каталоге *Debug* будут созданы каталоги *List*, *Obj* и *Exe*:
- каталог *List* – каталог хранения файлов листинга. Файлы листинга имеют расширение *lst*;
- каталог *Obj* – каталог хранения объектных файлов компилятора и ассемблера. Эти файлы имеют расширение *r79* и в дальнейшем будут использоваться как входные для компоновщика IAR XLINK;
- каталог *Exe* – каталог хранения исполняемого файла. Этот файл имеет расширение *d79* и в дальнейшем будет использоваться как входной для отладчика IAR C-SPY. Следует заметить, что этот каталог будет оставаться пустым, пока не будет произведена компоновка объектных файлов.

«Кликните» на значке «+» в окне рабочей области, чтобы полностью развернуть дерево файлов проекта. Как можно видеть, IAR Embedded Workbench в результате компиляции соз-

дал в рабочей области позицию табуляции папки *Output*, в которой содержатся все созданные выходные файлы. В дереве проекта отображаются также все подключаемые файлы заголовков.

Для того чтобы открыть любой файл из окна рабочей области в окне редактора IAR Embedded Workbench, например, для редактирования, необходимо дважды «кликнуть» на его имени в окне рабочей области.

Компоновка приложения

На этапе компоновки необходимо задать опции настройки компоновщика IAR XLINK Linker™. Для этого нужно выполнить определённую последовательность действий:

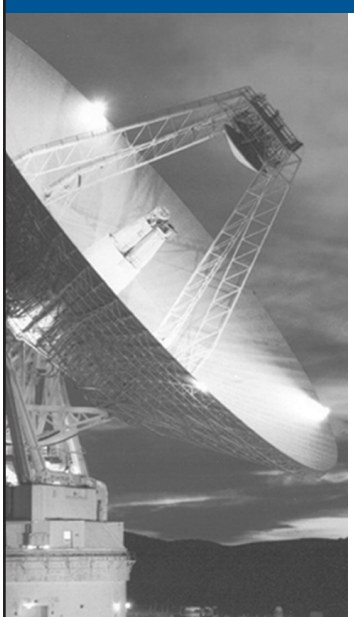
1. Выбрать позицию табуляции нужного проекта, например, *project1* – Debug в окне рабочей области, а затем выбирать в меню *Project>Options*. После этого выбрать строку *Linker* в списке *Category*, чтобы отобразить страницы опций компоновщика XLINK;
2. Задать установки параметров настройки в соответствии с табл. 3. При компоновке следует выбирать такой формат выходного файла компоновщика (*Output file*), который соответствует дальнейшим целям пользователя. Если планируется загрузить выходной файл в отладчик, то выходной файл должен содержать информацию отладки. Если планируется загрузить выходной файл в программатор PROM, то выходной файл не должен содержать информацию отладки и быть, например, в формате Intel-hex или Motorola S-records. Для своей работы компоновщик использует командный файл, который задаётся на странице *Config* опций компоновщика XLINK. Для данного проекта командный файл компоновщика не используется. Командные файлы компоновщика для некоторых оценочных плат и МК находятся в *src/examples*, о чём будет подробнее рассказано ниже. Для получения дополнительной информации о командных файлах компоновщика рекомендуется обратиться к [3, 4];
3. «Кликнуть» на кнопке *OK*, чтобы сохранить установленные опции XLINK. Теперь можно скомпоновать объектный файл и сгенерировать код, который в дальнейшем может быть отлажен;
4. Выбрать в меню *Project>Make*. Ход процесса компоновки будет отображаться в сообщениях в окне компоновки. В результате компоновки будет создан файл *project1.d79*, содержащий код с информацией отладки, и файл карты компоновщика *project1.map*.

Таблица 3 Параметры настройки XLINK для проекта *project1*

Страница окна	Параметр: значение
Output	Debug information for C-SPY With runtime control modules With I/O emulation modules
List	Generate linker listing Segment map Symbols: Module map

Отладка приложения в режиме симуляции

Встроенный в IDE EWARM отладчик IAR C-SPY Debugger является отладчиком языка высокого уровня. Он предназначен для использования с компилятором ARM IAR C/C++ Compiler и с ассемблером ARM IAR Assembler и полностью интегрирован в IAR IDE, обеспечивая одновременную разработку и

CREE®
Высокотемпературные радиационностойкие
SiC и GaN СВЧ-транзисторы



• Диапазон частот: до 4 ГГц
• Напряжение питания: 28...48 В
• Мощность: 10...60 Вт
• КПД: > 45%
• Температура перехода: > 255°C
• Наработка на отказ (MTTF) SIC СВЧ-приборов:
2,2 млн. часов при +225°C
60 млн. часов при +175°C
Реклама
ПРОСОФТ – официальный дистрибьютор компании Cree
ProSoft®
ПРОСОФТ — АКТИВНЫЙ КОМПОНЕНТ ВАШЕГО БИЗНЕСА
Телефон: (495) 232-2522 • E-mail: info@prochip.ru • Web: www.prochip.ru

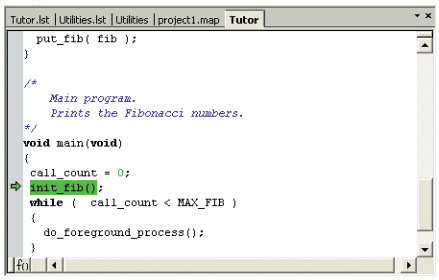


Рис. 1. Просмотр команд исходного текста

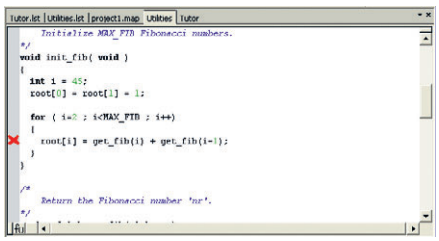


Рис. 2. Установка точек останова

отладку пользовательского приложения. При запуске C-SPY Debugger все окна, открытые в этот момент в рабочей области, останутся открытыми. Кроме того, будет открыт набор специфических окон C-SPY. Пользователь может изменить исходный текст своей программы в окне редактора в течение сеанса отладки, однако, эти изменения не вступят в силу, пока не будет произведён выход из отладчика. Интеграция отладчика в IDE позволяет установить контрольные точки средствами текстового редактора в любом месте пользовательской программы в ходе её написания и редактирования. Также можно просматривать и изменять контрольные точки, когда отладчик не выполняется.

В качестве примера работы с отладчиком IAR C-SPY Debugger рассмотрим отладку приложения *project1.d79*, создание которого было описано в предыдущем разделе. В процессе отладки пользователь может наблюдать за переменными, устанавливать контрольные точки, просматривать код в режиме дизассемблирования, управлять содержимым регистров и памяти и выводить на печать выходные данные программы в терминальном окне ввода-вывода.

Перед стартом отладчика IAR C-SPY Debugger необходимо задать несколько его опций. Задание опций производится в такой последовательности:

1. Выбрать в меню *Project>Options*, а затем категорию *Debugger*. На странице *Setup* убедиться, что выбрана строка симулятор *Simulator* из раскрывающегося списка *Driver* и пункт *Run to main*. «Кликнуть» мышью на кнопке *OK*;
2. Выбрать *Project>Debug*. Альтернативно можно нажать кнопку *Debugger* в инструментальной панели. Старт отладчика IAR C-SPY Debugger производится с загруженным приложением *project1.d79*. В дополнение к окнам, уже открытым в IAR Embedded Workbench, пользователю теперь станет доступен набор специфических окон IAR C-SPY Debugger.

Для просмотра команд исходного текста необходимо выполнить следующие действия:

1. Дважды «кликнуть» мышью на файле *Tutor.c* в окне рабочей области;
2. Когда содержимое файла *Tutor.c* отобразится в окне редактора, следует выбрать команду *Debug>Step Over*. Альтернативно можно нажать на кнопку *Step Over* на инструментальной панели. После этого в тексте отображаемого файла появится курсор, текущая позиция которого будет вызовом функции *init_fib*, как показано на рис. 1;
3. Выбрать в меню *Debug>Step Into* для показа передачи управления в функцию *init_fib*. Альтернативно можно нажать кнопку *Step Into* на инструментальной панели.

На уровне исходного текста команды отладчика *Step Over* и *Step Into* позволяют выполнять пользовательское приложение в пошаговом режиме. Выполненная команда *Step Over* показывает передачу управления с заходом его внутрь функций или вызовов подпрограмм, а при выполнении команды *Step Into* передача управления каждой функции показывается как единственный шаг без захода его внутрь функции. Пошаговое движение при выполнении приложения производится на уровне вызова функций, а не на уровне операторов. Чтобы пошаговое движение производилось на уровне операторов, следует выбрать команду *Debug>Next statement*. После этого за один шаг будет выполняться один оператор. Альтернативно можно нажать на кнопку *Next statement* на инструментальной панели.

C-SPY позволяет наблюдать переменные или выражения в исходном тексте таким образом, чтобы можно было следить за их значениями в процессе симуляции выполнения приложения. Наблюдать переменную можно несколькими способами, например, устанавливая мышью в окне исходного текста указатель или открывая одно из окон *Locals*, *Watch*, *Live Watch* или *Auto*.

Следует заметить, что когда используется уровень оптимизации *None*, все нестатические переменные полностью доступны для отладки. Когда используются более высокие уровни оптимизации, для отладки полностью доступны не все переменные.

Отладчик IAR C-SPY Debugger содержит мощную систему точек останова. Самый удобный путь её использования состоит в том, чтобы устанавливать точки останова в интерактивном режиме, позиционируя курсор на требуемой команде исходного теста или около неё, а затем выбирая команду *Toggle Breakpoint*. Установка точек останова производится в такой последовательности:

1. Пусть требуется установить точку останова на команде *get_fib(i)*. Сначала «кликните» на позиции табуляции *Utilities.c* в окне редактора, а затем «кликните» на требуемой команде, чтобы позиционировать на ней курсор. После этого выберите *Edit>Toggle Breakpoint*. Альтернативно можно нажать кнопку *Toggle Breakpoint* на инструментальной панели. Точка останова будет установлена на этой команде, а на левом краю окна напротив команды появится красный значок «X», указывающий на установленную точку останова, как показано на рис. 2;
2. Чтобы выполнять приложение только до тех пор, пока управление не достигнет точки останова, следует выбирать в меню *Debug>Go*. Альтернативно можно нажать кнопку *Go* на инструментальной панели. Приложение выполнится до точки останова, которая была установлена ранее. При передаче управления на точку останова выполнение программы будет приостановлено, а команда будет подсвечена;
3. Чтобы удалить ранее установленную точку останова, следует выбирать в меню *Edit>Toggle Breakpoint*.

Окно *Register* позволяет разработчику наблюдать и модифицировать содержимое регистров процессора.

Окно *Memory* позволяет разработчику наблюдать содержимое выбранных областей памяти. Можно, например, просмотреть область памяти, соответствующую переменным root.

Мониторинг памяти осуществляется следующим образом:

1. Выбрать *View>Memory*, чтобы открыть окно *Memory*;
2. Сделать активным окно *Utilities.c* и выбрать *root*, а затем перетащить его из окна исходного файла на С в окно *Memory*. При этом в окне *Memory* будет показано содержимое области памяти, соответствующей расположению переменной *root*;
3. Чтобы отобразить содержимое памяти в виде 16-разрядных слов данных, выберите команду *Memory16* в выпадающем меню на инструментальной панели окна *Memory*.

Можно вручную модифицировать содержимое памяти, редактируя значения в окне *Memory*. Для этого следует поместить курсор на ячейку памяти, подлежащую редактированию, и набрать с клавиатуры её новое значение.

Для выхода из C-SPY следует выбрать *Debug>Stop Debugging*. Альтернативно можно нажать на кнопку *Stop Debugging* на инструментальной панели. На экране отобразится рабочая область.

Использование библиотек

При работе над большим проектом у разработчика, как правило, накапливается коллекция полезных подпрограмм, которые впоследствии можно использовать и в других приложениях. С целью избежать необходимости транслировать эти подпрограммы всякий раз, когда в них есть необходимость, разработчик может сохранить такие подпрограммы как объектные файлы, которые оттранслированы, но не скомпонованы. Коллекция подпрограмм в отдельном объектном файле называется библиотекой. Использование библиотечных файлов рекомендуется, чтобы создавать коллекции связанных подпрограмм, образующих драйверы устройства.

Для формирования библиотек служит входящая в IAR Embedded Workbench программа построения библиотек IAR XAR Library Builder™. Программа позволяет следующим образом управлять библиотеками:

- изменять тип модулей с PROGRAM на LIBRARY и наоборот;
- добавлять или удалять модули из библиотечного файла;
- создавать листинг имен модулей, имен точек входа, и т.д.

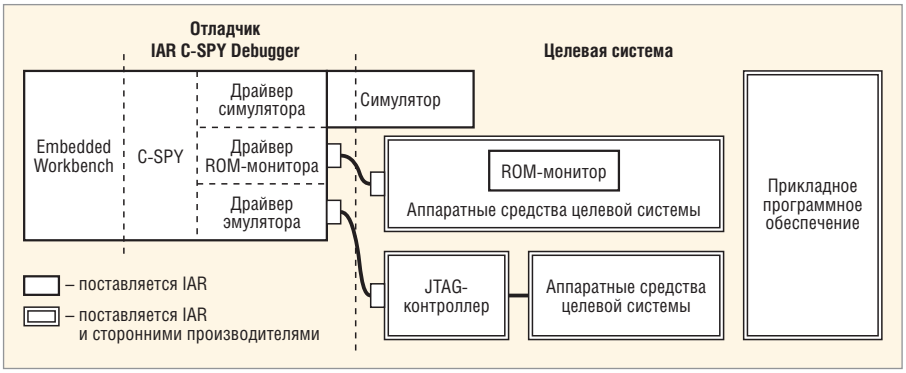


Рис. 3. Взаимодействие отладчика C-SPY Debugger и целевой пользовательской системы

Для получения дополнительной информации об использовании библиотек и создании библиотечных проектов рекомендуется обратиться к книге [5].

Универсальный компоновщик IAR XLINK Linker™

Компоновщик (линковщик) IAR XLINK преобразует один или более перемещаемых объектных файлов, которые являются продуктами работы ассемблера IAR или компилятора, в машинный код для выбранного процессора целевой пользовательской системы.

Компоновщик IAR XLINK поддерживает пользовательские библиотеки и включает в машинный код только те библиотечные модули, которые фактически необходимы программе, с которой они связаны.

Конечным продуктом работы компоновщика IAR XLINK является абсолютный, пригодный для выполнения в целевой пользовательской системе объектный файл, который может быть записан в оперативную или постоянную память МК, загружен в аппаратный эмулятор или может выполняться непосредственно в хосте с использованием отладчика IAR C-SPY Debugger (в режиме симуляции).

При компоновке программы компоновщик IAR XLINK выполняет четыре различных функции:

- преобразует в загрузочные модули исполняемый код или данные из входного файла (файлов);
- связывает различные модули вместе, разрешая применение глобальных, т.е. используемых всей программой символов, применение которых не могло быть разрешено ассемблером или компилятором;
- преобразует в загрузочные модули, необходимые программе, определённые пользователем IAR-библиотеки;
- задаёт местоположение каждого сегмента кода или данных в определённом пользователем адресе.

Настройки XLINK в IDE IAR Embedded Workbench

При выборе в главном меню IDE пункта *Project>Options* и последующего выбора строки *Linker* в списке Category откроется окно отображения страниц опций XLINK. На странице *Output* этого окна можно выбрать требуемый формат выходного файла компоновщика. Если активизировать кнопку *Debug information for C-SPY*, то выходной файл XLINK будет содержать информацию отладки. Если активизировать кнопку *Other*, то в списках *Output format* и *Format variant* можно выбрать формат выходного файла XLINK под требуемую аппаратную платформу и дополнительно задать его варианты. Следует заметить, что для загрузчика LPC2000 ISP (*BootLoader*) требуется задать формат выходного файла *intel-extended*. Для отладчика C-SPY лучше всего выбрать совокупное задание опций *Debug information for C-SPY*, *With runtime control modules* и *With I/O emulation modules*.

Поле *Output file* можно использовать, если есть необходимость сменить имя и/или расширение выходного файла.

На странице *Config* окна отображения страниц опций XLINK в области *Linker command file* можно задать путь к используемому командному файлу компоновщика с картой памяти выбранного МК (с расширением *xcl*) или, используя кнопку *Command file configuration tool*, открывающую одноименное окно редактирования,

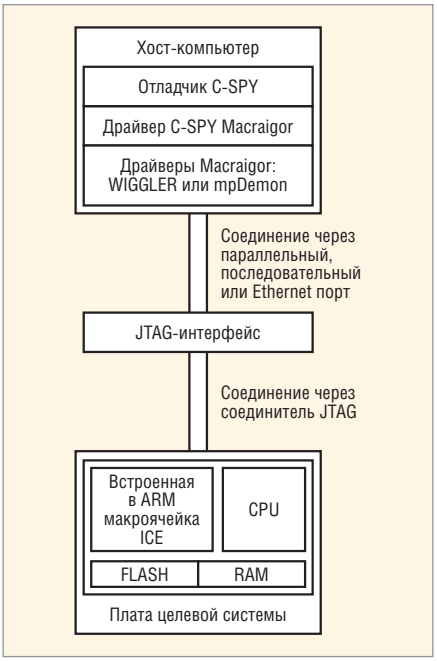


Рис. 4. Блок-схема связи хоста с целевой системой через драйвер Macraigor

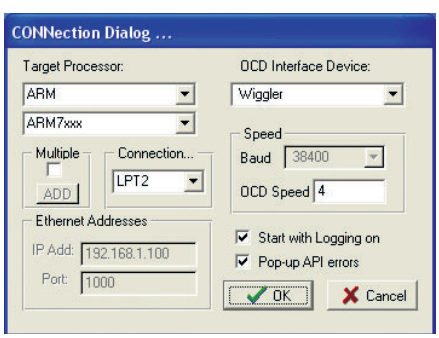


Рис. 5. Окно настройки драйвера Macraigor

создать, редактировать и сохранить командный файл вручную.

В качестве командных файлов компоновщика для устройств LPC2000 при разработке реальных проектов нами будут использоваться файлы *LPC2129_FLASH.xcl* и *LPC2129_SRAM.xcl*, предназначенные для использования в проектах на базе МК LPC2129. Эти файлы вместе с командными файлами компоновщика для других устройств семейства LPC2000 свободно распространяются производителем, а также имеются в составе IAR EWARM. Командный файл *LPC2129_FLASH.xcl* позволяет компоновщику генерировать объектный код, предназначенный для записи во Flash-память устройства LPC2129, а командный файл *LPC2129_SRAM.xcl* – код, предназначенный для записи в оперативную память (SRAM) LPC2129. Запись кода в SRAM с его последующим выполнением может использоваться разработчиком для отладки пользовательских программ непосредственно в МК целевой системы, но без расходования ресурса циклов стирания-записи встроенной Flash-памяти МК.

Отладка приложений в целевой системе

Входящий в состав IDE IAR Embedded Workbench отладчик IAR C-SPY Debugger наряду с отладкой в режиме симуляции может быть использован для отладки непосредственно в «железе», т.е. в составе целевой системы. Блок-схема, показывающая общую структуру отладчика IAR C-SPY Debugger и способы его взаимодействия с целевой системой, приведена на рис. 3. Целевая система может состоять из аппаратных средств оценочной платы или аппаратных средств конкретного пользовательского устройства. Целевая система может также полностью или частично моделироваться программным обеспечением. Для каждого отдельного типа целевой системы может использоваться специализированный драйвер C-SPY.

В рамках данной статьи будет рассматриваться вариант отладки в целевой системе, когда пользовательская прикладная программа выполняется в SRAM МК (в нашем случае – LPC2129). Для загрузки программы в SRAM и её корректного выполнения необходимо соблюдение двух основных условий:

1. Использование соответствующего командного файла компоновщика. Например, для устройства LPC2129 следует использовать файл *LPC2129_SRAM.xcl*;
2. Корректная инициализация специального регистра МК MEMMAP, который управляет переотображением в адресном пространстве памяти векторов прерываний LPC2000. Для выполнения программы в SRAM регистр MEMMAP должен быть инициализирован значением 2.

Разумеется, помимо перечисленных условий необходимо также подключить хост с отладчиком к целевой системе, а также соответствующим образом настроить аппаратные средства отладчика, о чём будет рассказано ниже.

Отладчик IAR C-SPY состоит из общей для всех драйверов части, которая обеспечивает основной набор операций отладки, и драйвера (драйверов). Драйвер C-SPY – это его часть, которая обеспечивает связь с целевой системой и управление ею. Драйвер C-SPY имеет пользовательский интерфейс в виде специального меню, окон и диалоговых окон, отображающих определённые функции целевой пользовательской системы.

В рамах этой статьи описывается драйвер JTAG-интерфейса Macraigor (по названию фирмы-производителя), с помощью которого в дальнейшем нами будет производиться отладка пользовательских приложений и их загрузка в память МК LPC2000. Этот драйвер позволяет задавать некоторое ограниченное количество точек останова кода и контрольных точек данных, а также производить выполнение кода в реальном времени с обработкой прерываний. Драйвер Macraigor не занимает никакого объёма в памяти МК, поскольку его код размещён целиком в хосте.

Блок-схема, отображающая структуру связи хоста с целевой пользовательской системой при использовании драйвера JTAG-интерфейса Macraigor, приведена на рис. 4.

Драйвер IAR C-SPY Macraigor может соединяться со встроенным интерфейсным модулем JTAG МК через аппаратные интерфейсы (кабели) Wiggler или mpDemon. Нами будет использоваться интерфейс Wiggler. Он выполнен в виде кабеля и подключается к параллельному порту компьютера хоста. Используемый драйвером Wiggler интерфейс JTAG МК – стандартное встроенное отладочное подключение, доступное в большинстве процессоров ARM, в том числе и в устройствах LPC2000.

Для задания драйвера отладчика C-SPY в меню IAR Embedded Workbench следует выбирать *Project>Options*. В открывшемся окне *Options* следует выбрать категорию *Debugger*, после чего в этой категории откроется страница *Setup*. Выбор драйвера для отладчика производится в поле *Driver* этой страницы. Для работы с кабелем Wiggler необходимо выбрать *Macraigor*. Для настройки выбранного драйвера JTAG Macraigor следует в окне *Options* выбрать категорию *Macraigor*, после чего в этой категории откроется одноименная страница. В этой странице необходимо задать настройки следующих полей:

- *OCD Interface device* – выбор устройства JTAG-конвертера. В нашем случае выбираем Wiggler;

- *JTAG speed* – скорость обмена хоста с МК через отладочный интерфейс JTAG. Задаваемое в этом поле значение должно лежать в диапазоне 1...8. Чем больше это число, тем меньше скорость. Как правило, кабель Wiggler хорошо работает при скорости, соответствующей 1, а для кабеля mpDemon может потребоваться значение 2 или 3. В случае неустойчивой связи хоста с МК скорость обмена рекомендуется уменьшить.

Необходимо заметить, что прежде чем использовать подключаемые через параллельный порт аппаратные драйверы Macraigor (в частности, Wiggler), необходимо установить на компьютер хоста программные драйверы Macraigor или OCDemon (программа *Macraigor JTAG drivers*). Эти драйверы входят в состав IDE IAR Embedded Workbench и могут быть установлены из начального инсталляционного меню IDE. Окно настройки *CONNection Dialog* установленной программы Macraigor JTAG drivers показано на рис. 5. Параметры подключенного интерфейсного устройства, заданные в этом окне (*OCD Interface Device* (*Wiggler*), порт *хоста Connection* (*LPTn*), скорость обмена через *JTAG OCD Speed*), должны совпадать с соответствующими параметрами, заданными в опциях *Debugger* и *Macraigor* C-SPY.

Перед началом отладки следует убедиться, что на вывод P0.14 МК LPC2000 целевой системы подан внешний низкий уровень, например, с помощью перемычки. Кроме того, линия RTCK интерфейсного подключения JTAG целевой системы также должна быть подключена к низкому уровню (через резистор сопротивлением 10 кОм).

Старт отладчика IAR C-SPY Debugger производится с предварительно загруженным в IAR EWARM пользовательским приложением. Для старта сеанса отладки следует выбрать *Project>Debug*. Альтернативно можно нажать кнопку *Debugger* в инструментальной панели. Следует заметить, что для драйвера Macraigor типичной является ситуация, когда хосту с первой попытки не удаётся установить связь с целевой системой через JTAG, о чём сигнализирует появление соответствующего сообщения. В этом случае придётся 1–2 раза «кликнуть» на кнопке *Повтор*, чтобы инициировать повторную попытку загрузки программы в SRAM МК. Ход успешно начавшегося процесса загрузки пользовательской программы в SRAM отображается графической шкалой. После перехода в режим отладки в дополнение к окнам, уже открытым в IAR Embedded Workbench, пользователю станет доступен набор специфических окон IAR C-SPY Debugger (рис. 6).

При отладке в целевой системе, как и при отладке в режиме симуляции, команды отладчика *Step Over* и *Step Into* позволяют выполнять пользовательское приложение в пошаговом режиме. Для выполнения программы в SRAM в реальном времени следует выбрать в меню *Debug>Go*. Именно в таком темпе (или чуть медленнее) программа будет выполняться в целевой системе после её записи во Flash-память МК.

Когда приложение выполняется, кнопка Break на инструментальной панели отладки подсвечивается красным цветом. Остановить выполнение программы приложения можно, «кликнув» на кнопке *Break*. Альтернативно следует выбрать в меню *De-*

Эффективные решения для высокочастотной техники от компании Micrometals

Магнитодиэлектрические сердечники для диапазона ВЧ и ОВЧ

MICROMETALS
IRON POWDER CORES

Основные достоинства и характеристики

- Широкая номенклатура типоразмеров – от 1,27 до 102 мм
- Токоизолирующее полимерное покрытие
- Диапазон частот от 0,01 до 500 МГц
- Высокая стабильность параметров

Области применения:

- Высокодобротные частотно-избирательные цепи
- Согласующие ВЧ-трансформаторы
- Квадратурные мосты
- Трансформаторы на линиях
- ВЧ-фильтры

Оригинальная продукция Micrometals в компании ПРОСОФТ

Реклама

PROSOFT®

ПРОСОФТ — АКТИВНЫЙ КОМПОНЕНТ ВАШЕГО БИЗНЕСА

Телефон: (495) 232-2522 • E-mail: info@prochip.ru • Web: www.prochip.ru

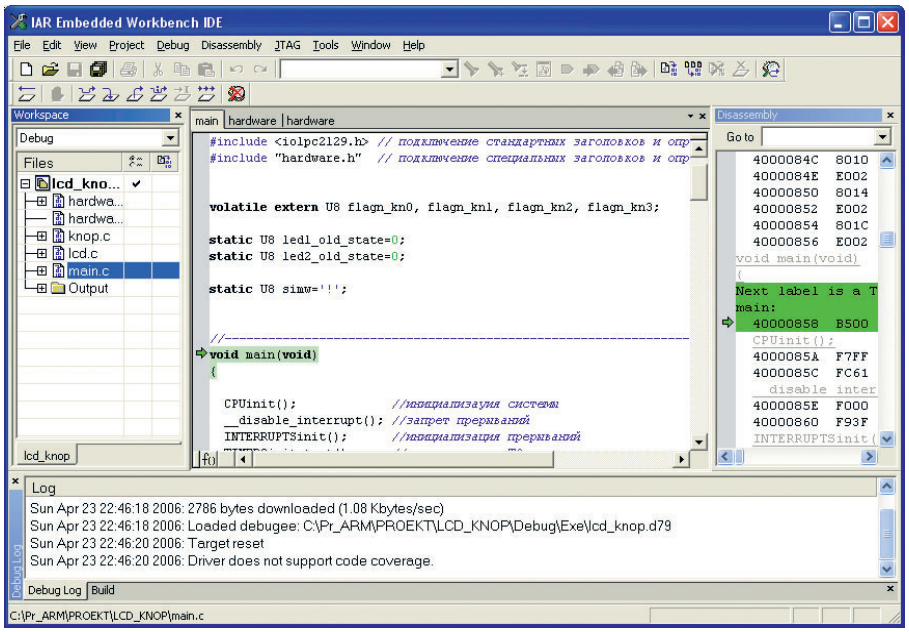


Рис. 6. Сеанс отладки в целевой системе

bug>Break. Выход из C-SPY производится при выборе в меню *Debug>Stop Debugging*. Альтернативно можно нажать кнопку *Stop Debugging* на инструментальной панели.

Мониторинг памяти (SRAM) и мониторинг регистров при отладке в целевой системе внешне производится аналогично мониторингу SRAM и регистров при отладке в режиме симулятора.

Запись пользовательских программ во Flash-память LPC2000

Для создания законченного пользовательского проекта отлаженную в SRAM программу в итоге необходимо записать во Flash-память МК. Для этого принципиально доступны два способа: программирование Flash-памяти через встроенный интерфейс JTAG средствами отладчика C-SPY (или отладчиков других IDE) или программирование Flash-памяти через порт UART0 средствами встроенного загрузчика *bootloader* LPC2000 (ISP-программирование).

Записать программу во Flash-память через JTAG можно следующим способом. Вместе с программой в SRAM необходимо записать собственный промежуточный Flash-загрузчик, который будет исполняться в SRAM и организует запись самой программы во Flash-память. Таким образом действует, например, отладчик C-SPY, о чём будет рассказано ниже.

Для облегчения возможности ISP-программирования Flash-памяти LPC2000 через UART0 средствами *bootloader* производитель LPC2000 свободно распространяет компьютерную программу LPC2000 Flash-Utility, работающую под ОС Windows Microsoft и использующую для программирования Flash-памяти LPC2000 последовательный порт (COM) компьютера хоста. Описание возможностей LPC2000 Flash Utility будет приведено ниже. Описание особенностей режима ISP-программирования Flash-памяти, реализованного в LPC2000 Flash Utility, а также алгоритмов работы загрузчика приведено в [6].

В исходном тексте пользовательской программы, которую планируется загрузить во Flash-память и которая до этого загружалась в целях отладки в SRAM, необходимо соответствующим образом скорректировать константу инициализации специального регистра MEMMAP, который управляет переотображением в адресном пространстве памяти векторов прерываний.

Загрузка Flash-памяти с помощью LPC2000 Flash Utility

Код встроенного загрузчика Flash-памяти *bootloader* выполняется при каждом сбросе или включении питания МК, поскольку точка входа в bootloader расположена по адресу вектора сброса 0x0. В момент сброса загрузчик производит выбор из двух альтернативных возможностей: определить тактовую частоту устройства и затем выполнять обработчик команд ISP-программирования или передать управление пользовательскому прикладному коду. Для запуска обработчика команд ISP-программирования должен быть сформирован соответствующий внешний аппаратный запрос. В качестве такого запроса выступает наличие внешнего низкого уровня на выводе МК P0.14 во время сброса. С целью исключения возможности случайного входа в режим ISP-программирования LPC2000 необходимо предусмотреть на выводе P0.14 наличие внешнего подтягивающего к «плюсу» питания резистора, поскольку после сброса этот вывод находится в высокоимпедансном состоянии.

Программа LPC2000 Flash Utility представляет собой графическую оболочку для выполнения команд ISP-программирования. Из своего главного окна LPC2000 Flash Utility позволяет производить с Flash-памятью МК LPC2000 следующие операции: запись в Flash-память выбранного файла с расширением *hex* – опция *Upload to Flash*, стирание Flash-памяти – опция *Erase*, проверка Flash-памяти на чистоту – опция *Blank Check*, верификация Flash-памяти – опция *Compare Flash*. Помимо этого из главного окна может производиться чтение идентификационного номера (поле Part ID) устройства LPC2000 и номера версии встроенного загрузчика *bootloader* (поле *Boot Loader ID*) с помощью опции *Read Device ID*.

К недостаткам используемой автором версии LPC2000 Flash Utility можно отнести только одно поддерживаемое расширение файлов «прошивки» Flash-памяти – *hex*. Для того чтобы сгенерировать выходной файл с расширением *hex* средствами IDE IAR Embedded Workbench, следует выбрать в меню *Project>Options*, а затем в категории *Linker* выбрать опию *Allow C-SPY-specific extra output file*. На странице *Extra Output* следует выбрать опцию *Generate extra output file*. В поле Format этой страницы следует выбрать выходной формат *intel-extended*, а в поле *Format variant* – *None*.

Загрузка Flash-памяти LPC2000 с помощью C-SPY

По умолчанию отладчик C-SPY загружает приложение в оперативную память или во Flash-память устройства, когда запускается сеанс отладки.

Однако опции *Download* отладчика позволяют задать параметры такой загрузки для достижения в итоге загрузки во Flash. Flash-загрузчик может быть описан как агент, который загружается в целевую систему. Он получает пользовательский код от отладчика C-SPY и записывает его во Flash-память. Flash-загрузчик использует механизм файлового ввода-вывода, чтобы читать прикладную программу из хоста. Пользователь может выбрать один или несколько загрузчиков Flash-памяти, каждый из которых загрузит выбранную часть пользовательского приложения. IDE IAR Embedded Workbench поддерживает набор загрузчиков Flash-памяти для различных МК, в том числе для LPC2000. Параметры загрузчика, не входящего в этот набор, могут быть заданы в IDE IAR Embedded Workbench вручную.

Задание загрузчика Flash-памяти

Задание загрузчика Flash-памяти в IDE IAR Embedded Workbench производится следующим образом:

1. Выбрать в меню *Project>Options*;
2. Выбрать в категории *Debugger* позицию табуляции *Download*;
3. Выбрать поле *Use Flash loader(s)* и «кликнуть» на кнопке *Edit*;
4. В открывшемся диалоговом окне обзора загрузчиков *Flash Loader Overview* будут перечислены все доступные на текущий момент загрузчики Flash-памяти.

В случае необходимости пользователь может добавить к списку загрузчиков Flash-памяти новый загрузчик, «кликнув» на кнопе *New*. При этом откроется диалоговое окно конфигурации загрузчика *Flash Loader Configuration*, которое предоставляет возможность конфигурировать загрузку. Если ранее на странице *General Options>Target* был произведён выбор устройства, у которого в IAR EWARM имеется встроенный загрузчик Flash-памяти, то в окне *Flash Loader Overview* по умолчанию будет указан этот загрузчик.

С помощью кнопки *OK* выбранный загрузчик(и) Flash-памяти может быть задан для того, чтобы загрузить приложение в Flash-память.

Кнопкой *Delete* можно удалить выбранную конфигурацию загрузчика.

Механизм загрузки во Flash-память

Когда опция *Use flash loader(s)* выбрана и один или несколько загрузчиков Flash-памяти сконфигурированы, при запуске сеанса отладки будут выполнены следующие шаги:

1. C-SPY загружает загрузчик Flash-памяти в оперативную память целевой системы;
2. C-SPY начинает выполнение загрузчика Flash-памяти;
3. Загрузчик Flash-памяти открывает файл, содержащий прикладной код;
4. Загрузчик Flash-памяти читает прикладной код из файла и записывает его во Flash-память;
5. Выполнение загрузчика Flash-памяти завершается;
6. C-SPY переключают контекст на пользовательское приложение.

Шаги 1 – 5 выполняются для каждого выбранного загрузчика Flash-памяти.

Требования к компоновке

Когда пользователь производит формирование приложения, которое будет загружено во Flash-память, необходимо сгенерировать два выходных файла с одинаковыми именами, но разными расширениями, находящихся по одному и тому же пути. Первый из этих файлов обеспечивает отладчик символьной и отладочной информацией и имеет формат *UBROF* (расширение *d79*). Второй – файл простого кода (расширение *sim*), который будет открыт и прочитан загрузчиком Flash-памяти, когда последний будет загружать приложение во Flash-память МК.

Для задания установки создания дополнительного выходной файла следует выбрать в меню *Project>Options*, а затем в категории *Linker* выбрать опию *Allow C-SPY-specific extra output file*. На странице *Extra Output* следует выбрать опцию *Generate extra output file*. В поле *Format* этой страницы следует выбрать выходной формат *simple-code*, а в поле *Format variant* – *None*. В поле *Output file* следует выбрать опцию *Override the default*, тем самым не отменяя заданный по молчанию выходной файл с расширением *sim*.

На странице *Config* категории *Linker* необходимо задать командный файл компоновщика, соответствующий загрузке во Flash-память.

Порядок действий при загрузке во Flash-память

Для записи кода во Flash-память после компилирования и компоновки проекта с описанными выше параметрами необходимо перейти к его отладке, выбрав в меню *Project>Debug*. Предварительно следует убедиться, что на вывод P0.14 МК LPC2000 подан внешний низкий уровень. В случае успешной установки связи через JTAG после завершения процесса записи загрузчика в SRAM, появится шкала, отображающая ход процесса загрузки во Flash-память, и состоится переход в режим отладки. Однако пользовательская программа будет выполняться уже во Flash-памяти МК, в чём можно убедиться, отключив питание отладочной платы и включив его снова со снятой перемычкой, подающей внешний низкий уровень на вывод P0.14. Разъём кабеля JTAG на время выполнения приложения во Flash-памяти следует отключить от платы.

Заключение

Описание основных особенностей IAR EWARM, приведённое выше, наглядно демонстрирует широкие возможности этой профессиональной отладочной среды. Как могли убедиться читатели, разработчик, используя только одну программную оболочку, практически одновременно создаёт и редактирует исходные коды, отлаживает их в симуляторе или в оперативной памяти МК целевой системы, а по завершении отладки – записывает итоговый код в «кристалл». При этом к его услугам богатый набор «подручных» средств и инструментов, облегчающих и ускоряющих творческий процесс.

Литература

1. ARM® IAR Embedded Workbench™ IDE. User Guide for Advanced RISC Machines Ltd's ARM Cores. © Copyright 1999–2005 IAR Systems.
2. Installation and Licensing Guide for the IAR Embedded Workbench™. November 2003, © Copyright 2003 IAR Systems.
3. ARM® IAR C/C++ Compiler. Reference Guide for Advanced RISC Machines Ltd's ARM Cores. © Copyright 1999–2005 IAR Systems.
4. IAR Linker and Library Tools. Reference Guide. Version 4.59. © Copyright 1987–2005 IAR Systems.
5. *Редькин П.П.* Микроконтроллеры ARM7 семейства LPC2000. Руководство пользователя. М.: Додэка-XXI. 2007.
6. Entering ISP mode from user code. AN10356 Application note. Rev. 02 – 25 October 2005.