DAVE 3 — среда компонентно-ориентированного программирования для встраиваемых систем

Компания Infineon серийно выпускает ARM-микропроцессоры для промышленных систем реального времени и коммуникационных приложений. Специально для этих процессоров Infineon разработала новую версию уже отлично зарекомендовавшего себя пакета DAVE (Digital Application Virtual Engineer), который используется для создания программного обеспечения и отладки программ на базе кристаллов Infineon. Среда DAVE 3 имеет все необходимые компоненты стандартных IDE (Integrated Development Environment) — редактор, компилятор, отладчик, библиотекарь и т. д. Но ее основной особенностью является добавленный программный инструмент Code Engine, представляющий собой автоматический генератор кода, разработанный в соответствии с концепцией компонентно-ориентированного программирования (Component Based Programming, CBP) для встраиваемых систем реального времени. В статье рассмотрены основные отличия концепции, применяемой для CBP, от объектно-ориентированного программирования и применение CBP для практической разработки встраиваемых систем.

Олег ИВАНОВ
oyi@efo.ru

Обзор системы DAVE 3

Не секрет, что выбор микропроцессора и программного обеспечения является одним из важных критериев для разработчика. Конечно, в первую очередь обычно смотрят на технические характеристики самого микропроцессора. Но совокупность доступных средств разработки и отладки также играет немаловажную роль, включая простоту перехода на новый инструментарий и стоимость всей совокупности аппаратных и программных средств для разработки и отладки.

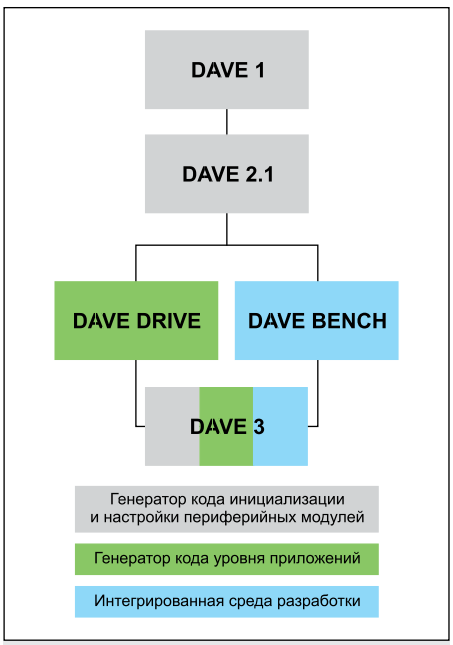

Рис. 1. Развитие продуктов DAVE

Среда обеспечивает быстрый старт разработки на процессорах Infineon. Это полнофункциональный комплект, позволяющий выполнить все этапы разработки и отладки. Система DAVE 3 будет также полезна тем, кто привык использовать коммерческие среды разработки (например, IAR или KEIL) при генерации драйверов для частей приложения, а дальше продолжает работать в привычном окружении. Так, например, KEIL имеет плагины, позволяющие сконвертировать проект из DAVE 3 одним нажатием клавиши.

DAVE 3 прошла длительный путь в своем развитии. Первая версия среды была выпущена в конце 1990-х годов. За ней последовала версия DAVE 2.1, главное отличие которой состояло в наличии генератора кода для периферии микропроцессоров Infineon. DAVE 1 и 2.1 имели графический интерфейс, позволявший наглядно представить ядро и все периферийные блоки процессоров, а также сконфигурировать их в зависимости от задач проекта и сразу же сгенерировать С-код для начальной инициализации и драйверов для работы с периферией. Это помогало разработчику легче понять, как сложная периферия может быть сконфигурирована и использована в проекте. Для примера: DAVE 1 и 2.1 могли сгенерировать функции для инициализации АЦП или пересылки по CAN-шине.

Но при всей своей мощности обе предыдущие версии имели существенное ограничение: они могли генерировать только низкоуровневые драйверы, а остальную часть проекта приходилось разрабатывать и отлаживать в системах сторонних фирм. Дальнейшим развитием среды стал DAVE Drive, который позволял генерировать и верхний уровень программ, но в основном только для нескольких различных алгоритмов управления электродвигателями. С помощью графической оболочки этого приложения программист мог выбрать тип электродвигателя, алгоритм управления, конфигурацию обратной связи, временные характеристики и т. п., то есть реализовать полностью функционирующее приложение управления двигателями.

Следующим этапом эволюции DAVE стал DAVE Bench — базирующийся на Eclipse IDE и предназначенный для 8-разрядных микроконтроллеров Infineon семейства XC800. Он позволял редактировать, компилировать и отлаживать программное обеспечение, сгенерированное системами DAVE 2.1 и DAVE Drive. Но при этом DAVE Bench не мог самостоятельно генерировать код.

Последняя версия среды DAVE 3 стала намного функциональнее своих предшественников (рис. 1). Она наиболее близко по своим возможностям напоминает DAVE Bench. Тем не менее включенный генератор кода стал гораздо более мощным, чем ранее использовавшийся в системах DAVE 2.1 или DAVE Drive, чему способствовал компонентно-ориентированный подход.

Компонентно-ориентированное программирование

Компонентно-ориентированное программирование (или разработка программного обеспечения, базирующаяся на понятии компонентов) концептуально основана на разделении проекта на отдельные независимые функциональные блоки, которые можно использовать многократно в пределах программы или даже в других проектах. Эти блоки программного обеспечения получили название компонентов.

Использование компонентно-ориентированного подхода позволяет облегчить создание программ и сэкономить время на отладке. В целях дальнейшего использования программного компонента интерфейс и порядок его использования должны быть четко определены и документированы. Это позволяет полагать, что компонент представляет собой «черный ящик», и не заострять внимания на его содержимом. Правда, сразу вспоминаются недостатки использования уже скомпилированного объектного кода, когда при необходимости что-то изменить требуется дополнительно покупать исходный код.

Хотя, используя систему DAVE 3, как правило, нет необходимости беспокоиться по поводу внутреннего содержания объекта: команда разработчиков Infineon очень внимательно относится к включению подробной документации и исходных кодов, чтобы гарантировать разработчику возможность гибкой модификации компонента в соответствии со своими требованиями.

Для того чтобы показать на примере, как можно использовать компонентно-ориентированное программирование (CBP) для упрощения разработки приложения по сравнению с объектно-ориентированным подходом, рассмотрим простой пример с использованием встроенного в микроконтроллер АЦП для ввода какого-либо аналогового сигнала и отображения амплитуды этого сигнала на дисплее.

Существует несколько способов разделить эту систему на части, но, тем не менее, наиболее важными останутся три: выполнение аналогового преобразования и чтение полученных результатов, перевод значения, полученного в результате преобразования, в код, который будет отображаться на дисплее, и выдача символов на дисплей. Каждая из этих основных функций, которые могут быть представлены в виде компонентов, показаны

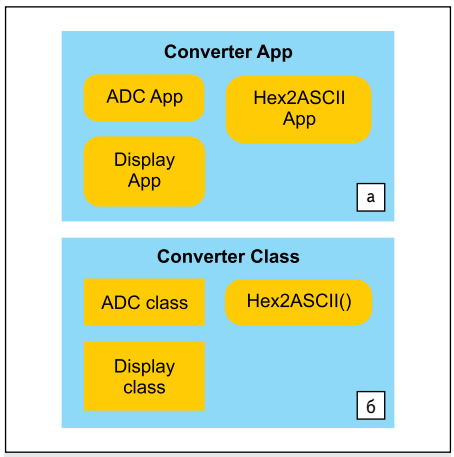

Рис. 2. Разбиение задачи в зависимости от примененного подхода

на рис. 2. В терминологии DAVE 3 эти компоненты называются Apps (рис. 2а). Отметим, что эти компоненты описываются в основном глаголами, поскольку они действительно что-то выполняют. Для того чтобы эти компоненты можно было использовать в дальнейшем, необходимо выполнить еще несколько действий, которые мы рассмотрим позже.

Эти же объекты могут выглядеть немного по-другому, когда мы используем объектно-ориентированное программирование. Концепция объектно-ориентированного программирования рассматривает разделение системы на объекты по типу данных. В языке C++ объекты называются классами и включают типы данных, пользовательские функции, конструкторы/деструкторы и операторы возможных событий. На рис. 2б показано, как можно разделить наше приложение на объекты, используя объектно-ориентированное программирование. Двумя основными объектами в этом приложении являются АЦП и дисплей. А функция преобразования значения, полученного с АЦП, в вид для представления на дисплее, будет пользовательской функцией в объекте. В объектно-ориентированном программировании объект часто более тесно ассоциируется с существительным. Поэтому ADC-класс представляет собой именно какой-то аналогово-цифровой преобразователь, а в компонентно-ориентированном ADC App представляет действия начальной подготовки АЦП к преобразованию и само аналогово-цифровое преобразование. В объектно-ориентированном программировании каждый дополнительный АЦП будет представлен в виде нового объекта. Используя концепцию компонентно-ориентированного программирования, мы бы представили дополнительный АЦП в виде нового компонента, выполняющего преобразование. Это не означает, что объектно-ориентированный и компонентно-ориентированный подходы несовместимы. Объекты могут быть в то же самое время и компонентами, если они разрабатываются для выполнения независимых функций.

Компонентно-ориентированное программирование для встраиваемых приложений

Функция преобразования результатов, полученных с АЦП, в вид для вывода на дисплей (как, например, широко используемая Hex2ASCII), может использоваться многократно в различных приложениях, потому что входные и выходные параметры хорошо определены, а сама функция аппаратно-независима, то есть не предъявляет каких-либо специфических требований к аппаратной части. Остальные компоненты в нашем примере на рис. 2а не так универсальны, потому что должны взаимодействовать с различными аппаратно-зависимыми модулями. Это является препятствием для широкого использования компонентно-ориентированного программирования для разработки встраиваемых систем.

Значительная часть программного обеспечения встраиваемых систем тесно связана с особенностями встроенной в микроконтроллер периферии или других аппаратных ресурсов. В любое время, когда программа должна взаимодействовать с аппаратным обеспечением, вы можете сомневаться в возможности применения данного компонента. Для примера, рассмотренные компоненты ADC и Display могут легко обеспечить работу с определенным АЦП и дисплеем, но даже небольшие изменения, такие как подстройка времени преобразования АЦП или изменение назначения физических выводов, к которым подключен дисплей, могут вызвать значительные изменения в реализации компонента.

DAVE 3 реализует среду использования компонентов (называемых DAVE Apps), которая была написана на основе концепции виртуальных аппаратных ресурсов. Для примера, компонент ADC App реализован так, что может использовать любой канал АЦП в любом модуле аналогово-цифрового преобразования. Благодаря этому, App можно применить столько раз, сколько это необходимо. В семействе процессоров XMC4000 и XMC1000 возможность многократного применения компонента возрастает из-за повторяемости нескольких аналогичных блоков. В процессорах этих семейств периферийные блоки «повторяются» двумя способами. Во-первых, многие блоки периферии многократно продублированы (так, например, четыре модуля CCU4, два модуля CCU8, четыре ADC и т. п.). Во-вторых, каждый периферийный модуль может состоять из нескольких повторяющихся разделов. Например, каждый модуль CCU4 составлен из четырех одинаковых разделов, каждый аналогово-цифровой преобразователь состоит из восьми аналогичных разделов и т. д.

На рис. 3 показан обобщенный вид микроконтроллера, входящего в семейство XMC4000. Он состоит из центрального процессорного ядра со схемой тактирования, системы обслуживания прерываний системных событий

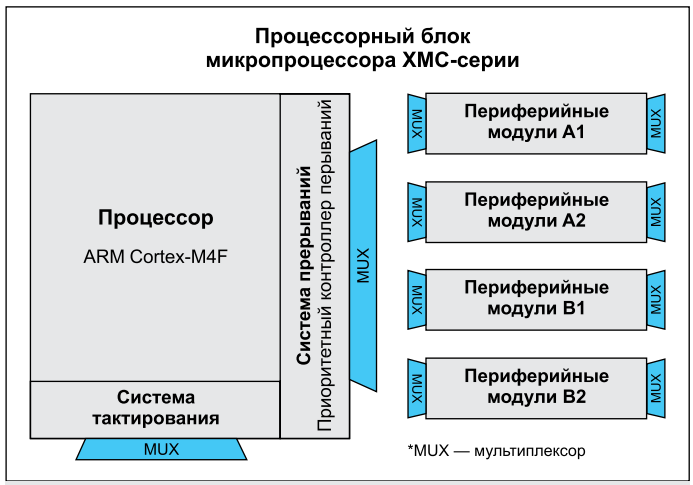

Рис. 3. Организация микроконтроллера с ядром Cortex

и периферии. Все входы и выходы периферии связаны с мультиплексорами, что позволяет подключать их к необходимым тактирующим сигналам и обслуживанию прерываний.

Эта гибкая система получила название матрицы подключений. Благодаря этой матрице, события из одного периферийного блока могут вызвать какое-либо событие в другом периферийном блоке. Наиболее простой пример — запуск аналогово-цифрового преобразования или прерывание по событию, вызванному истечением временного промежутка, от таймера.

В DAVE 3 все DAVE Apps написаны так, чтобы можно было использовать модули, которые повторяются в периферийных блоках микроконтроллера. (На рис. 3 эти модули заштрихованы серым цветом.) Система DAVE работает с файлами описания устройств, которые указывают, какие периферийные модули и какие соединения возможны для данного варианта микропроцессора. Затем отдельная программа, называемая Solver, используя параметры DAVE Apps и реально доступные аппаратные модули, вычисляет правильные установки мультиплексоров.

Использование DAVE 3 в соответствии с концепцией CBP

Вместо того чтобы потратить большое количество времени на объяснение теории использования компонентно-ориентированного программирования для встраиваемых систем, лучше посмотреть на примеры, которые показывают, как DAVE 3 управляет компонентами, требующими аппаратных ресурсов. Напомним, что в среде DAVE 3 компоненты носят название DAVE Apps.

На рис. 4 приведен простой проект на основе DAVE Apps. Система принимает информацию через UART и отображает ее на графическом OLED-дисплее, который подключен к микропроцессору через интерфейс SPI. Когда запускается новый DAVE 3CE проект, в правой части рабочего окружения появляется окно **App Selection View**, показывающее, какие DAVE Apps доступны. Они загружаются и устанавливаются в локальное хранилище с сайта Infineon и могут представлять собой как отдельные компоненты, так и целую систему, например, для того, чтобы реализовать полнофункциональный web-сервер. Двойным кликом по любому приложению DAVE Apps вы вставляете его в свой проект, и он сразу становится виден в нижнем окне **App Dependency TreeView**. В этом окне отображаются объекты, связанные с данным компонентом, и виртуальные сигналы, то есть визуальное представление матрицы соединений. Они называются виртуальными, потому что только с помощью Solver впоследствии получат реальное воплощение в виде связей с конкретными физическими сигналами.

Используемый пример включает в себя семь DAVE Apps. Низкоуровневый драйвер обслуживания OLED (GUILC001), SPI (SPI001), виртуальный сигнал ввода/вывода I/O pin (IO002), сигнал сброса reset (RESET001) и тактовый сигнал clock (CLK001) были автоматически добавлены в проект, как только мы импортировали UART и Segger GUI приложения. В соответствии с нашими ожиданиями сигнал тактирования clock, используемый для настройки основного синхрогенератора, и сигнал reset были подключены к UART и SPI интерфейсам. Эти два приложения являются единственными в микроконтроллере и, соответственно, могут быть включены в проект лишь однажды. Другие приложения Apps могут повторяться в проекте столько раз, сколько необходимо: например, программа может взаимодействовать с несколькими дисплеями с помощью нескольких последовательных портов.

До сих пор мы рассматривали компоненты только как часть программного кода, который выполняется микроконтроллером. Но среда разработки с DAVE Apps — это нечто большее, чем просто С-код, поскольку они могут включать в себя графический пользовательский интерфейс, используемый для конфигурирования соответствующего приложения. Такой интерфейс, включающий все необходимые настройки для конфигурирования UART App, изображен на рис. 5.

Все возможности DAVE Apps хорошо документированы и снабжены Help-файлами. На рис. 6 показана начальная страница описания Segger Gui DAVE Apps, которая включает всю необходимую информацию для понимания того, что представляет собой это приложение,

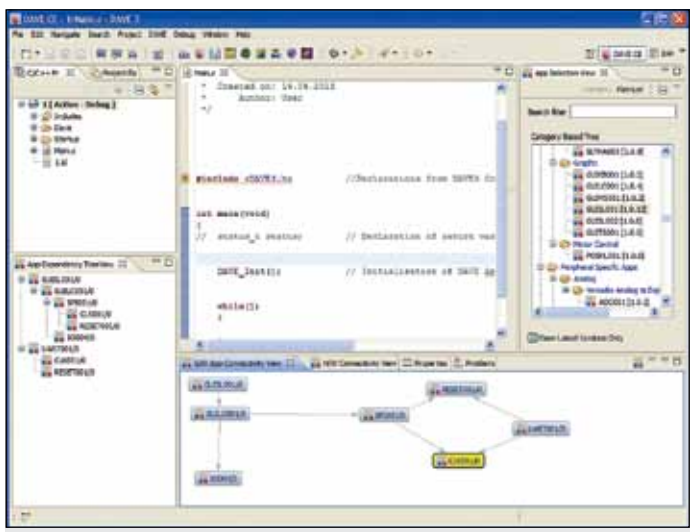

Рис. 4. Выбор GUISL001 и добавление в проект

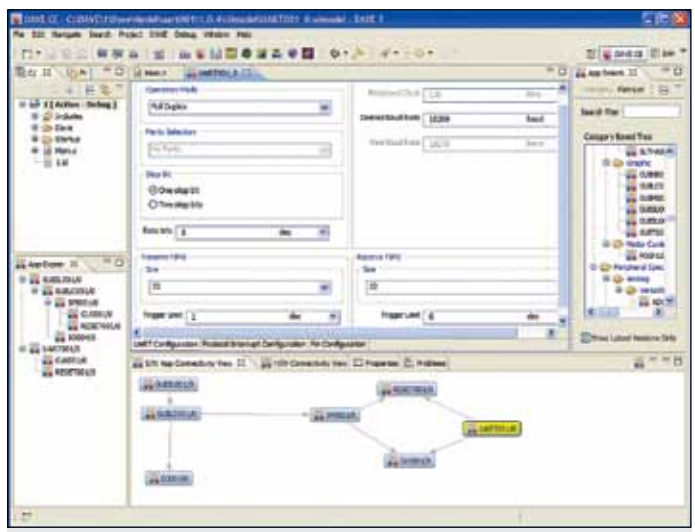

Рис. 5. Настройка параметров UART с помощью графического интерфейса

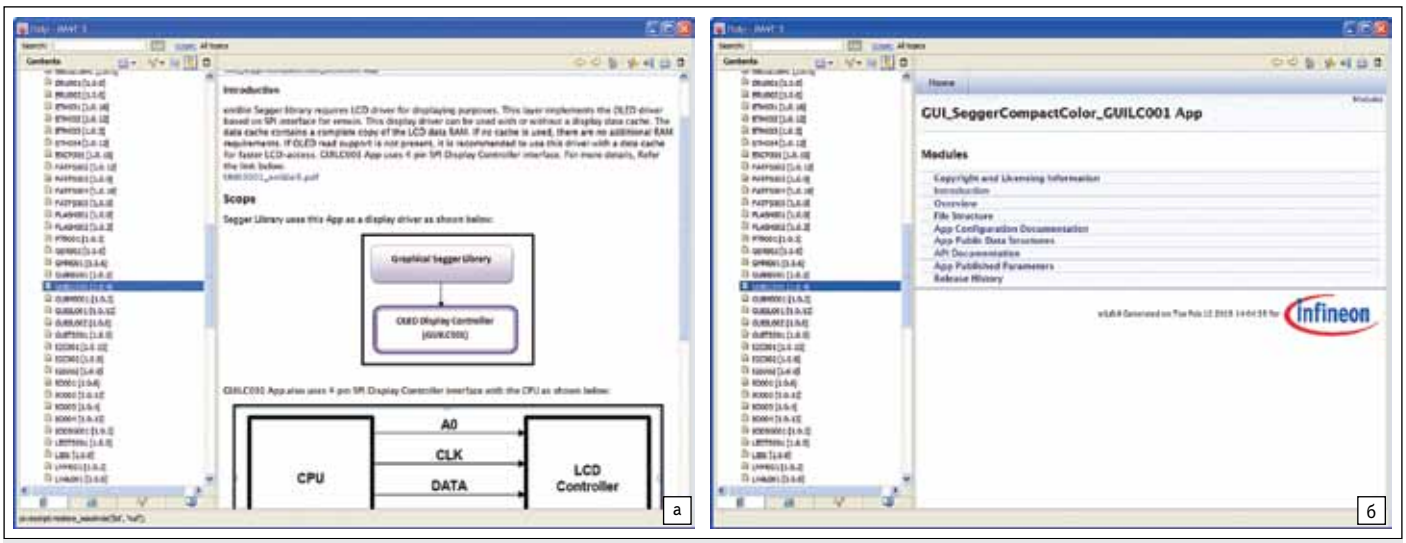

Рис. 6. а) Вид справочной информации о приложении DAVE Apps; б) один из подразделов справочной информации о GUILC001

какие имеет ресурсы и как его следует использовать. Варианты доступа и вид справочной информации показан на рис. 6.

С этого момента мы могли бы запустить программу Solver и сгенерировать код, в результате автоматически были бы выбраны соответствующие USIC (Universal Serial Interface Channel) модули, каналы и сигналы ввода/вывода для успешного выполнения функций UART и SPI-интерфейсов. В большинстве случаев разработчик может сам назначить необходимые ему выводы микроконтроллера. Для примера, если вы желаете запустить рассматриваемое приложение на отладочном комплекте, вы должны выбрать физические сигналы, через которые дисплей подключен к SPI. Система DAVE 3 легко позволяет определить необходимые сигналы. Нажав правую кнопку мыши, вы можете выбрать **Manual Pin Assignment** и принудительно определить желаемые параметры. На рис. 7 показано окно программы, где отображена эта процедура. Далее переопределенную таблицу можно сохранить в формате Excel.

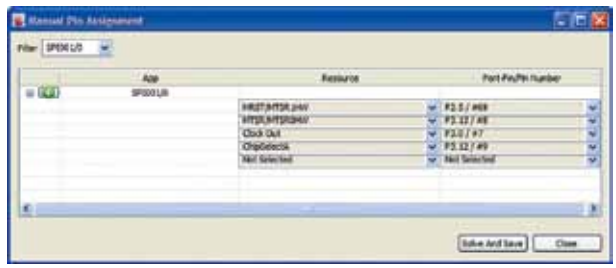

Рис. 7. Связывание виртуальных сигналов с реальными выводами

К этому моменту наш пример включал все необходимое для инициализации приложения и низкоуровневых драйверов, обеспечивающих начальную установку и функции передачи данных. Теперь мы можем добавить несколько функций для дальнейшей обработки полученных данных или расширить функционал программы с помощью добавления приложений, например DAVE RTOS App, и сконфигурировать периодическую выдачу принятых данных через UART FIFO.

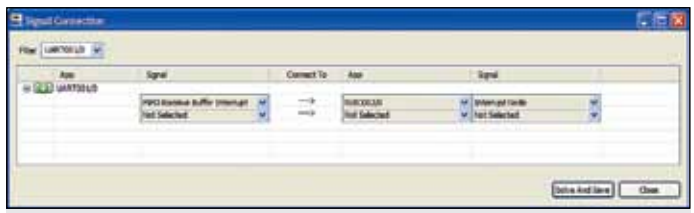

Рис. 8. Соединение виртуальных сигналов различной периферии

Наиболее простой путь для решения этой задачи — генерация прерывания каждый раз, когда UART принимает очередной байт (рис. 8). Это легко обеспечивается добавлением Nested Vectored Interrupt Controller (NVIC002) и соединением его виртуальным сигналом с приемным буфером FIFO.

В дальнейшем мы можем просто вызывать UART и GUI API внутри процедуры обслуживания прерывания. На рис. 9 показано, как можно выполнить эти шаги. В результате мы получим полностью документированную программу со всеми исходными кодами.

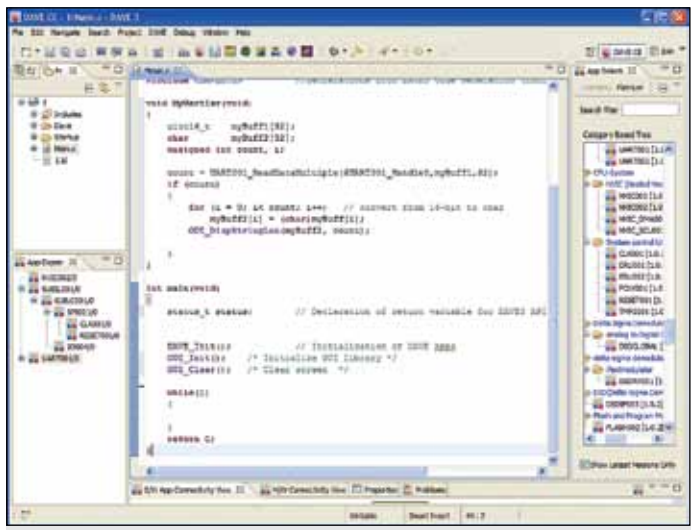

Рис. 9. Определение функции работы с буфером FIFO через UART API

Функция чтения данных из FIFO считывает данные до тех пор, пока не опустошится буфер или когда будет прочитано точное количество байт (в данном случае 32). Для выполнения этой задачи используется функция UART001_ReadDataMultiple, которая определена в UART DAVE App API. В качестве одного из параметров функция получает **UART001_Handle0**. Это ключевой вопрос для работы DAVE Apps. С того момента, как Solver связал UART App с каким-либо располагаемым последовательным каналом (обеспечив выполнение всех пользовательских условий), DAVE снабжает DAVE Apps уникальным дескриптором (id). Это позволяет писать код, абстрагируясь от конкретной периферии, для которой API уже использована, и многократно применять те же функции API, не волнуясь об их влиянии

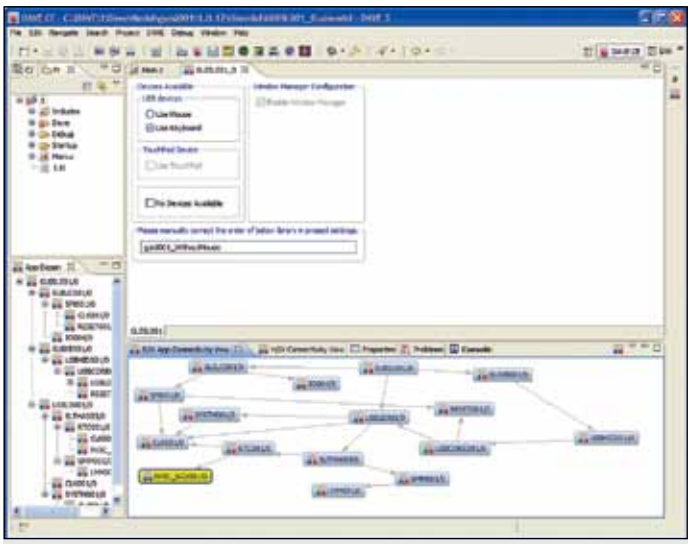

Рис. 10. Изменение проекта путем удаления UART и добавления USB-клавиатуры

друг на друга. Таким образом, если нам требуется шесть UART, мы можем просто добавить в проект UART App шесть раз, но на выходе мы получим только один экземпляр сгенерированного кода и, соответственно, один заголовочный файл.

Реализация предыдущего простого примера заняла всего несколько минут. Основные проблемы были связаны с OLED GUI библиотекой, но они были успешно решены при помощи DAVE Apps. Среда DAVE Apps располагает и другими компонентами для реализации более сложных систем. Для примера, если вы решите исключить UART-интерфейс и подключить USB-клавиатуру напрямую к микроконтроллеру (через USB OTG интерфейс в XMC4500), то достаточно будет удалить UART и NVIC DAVE Apps из проекта и выбрать опцию «Использовать клавиатуру» в Segger GUI DAVE App (GUISL001) UI. Это позволит автоматически вставить USB Keyboard DAVE App (USBKBD001) вместе в USB-драйвером (USBLD001 и USBCORE001) и USB HID (GUIKB001) DAVE Apps. Результат этих действий показан на рис. 10.

При условии, что DAVE Apps не использует предварительно скомпилированные библиотеки, все исходные коды и результаты компиляции будут включены в проект (в папку **DAVE\Generated folder**) и могут быть легко исследованы, модифицированы и конвертированы в другие среды разработки. Даже шаблоны, показывающие, какие установки UI были установлены при генерации кода, будут снабжены.jet-файлами для возможности последующей модификации.

Выводы

Компонентно-ориентированное программирование становится все более популярным. Среда DAVE 3 компании Infineon привносит в программирование встроенных систем все преимущества этого метода без каких-либо ограничений для систем реального времени, а также устраняет компромисс между удобством многократно используемого кода и аппаратными особенностями выбранного микроконтроллера посредством применения виртуальных ресурсов и утилиты Solver, которая обеспечивает создание связей между компонентами с учетом всех аппаратных ограничений.

К интегрированной среде разработки DAVE 3 обеспечен свободный доступ на сайте **www.infineon.com**, откуда ее можно загрузить. Эта среда поддерживает все процессоры фирмы Infineon, основанные на ядре Cortex (XMC1000 и XMC4000). Работа с DAVE 3 позволит ближе познакомиться с принципами компонентно-ориентированного программирования и, возможно, выбрать процессоры с ARM-ядром Cortex для реализации своих будущих проектов. Компания Infineon заявила о полной поддержке таких сред разработки, как Keil и IAR. В ближайшее время также планируется выпуск SDK, который позволит самостоятельно разрабатывать DAVE Apps.

Литература

1. **http://www.infineon.com/dgdl/Embedded++Computing+Based+Programming+with+DAVE+3.pdf**
2. **http://www.monkeys-at-keyboards.com/java3/1.shtml**
3. **http://en.wikibooks.org/wiki/Computer_Programming/Component_based_software_developmet**
4. Froehlich P. Component-Oriented Programming Languages: Why, What, and How. University of California, Irvine, USA. 2003.