Визуальные средства разработки программ для малых встраиваемых систем. Опыт применения

Александр Елисеев (г. Вильнюс, Литва)

Интенсивное развитие встраиваемых систем способствует росту интереса к визуальным средам разработки программ для них. Визуальная разработка может существенно ускорить создание программного обеспечения, но только при условии адекватного её применения, поскольку существует множество концепций и методик этого процесса. В статье приведён краткий обзор существующих подходов и частный случай внедрения среды визуальной разработки.

Введение

Предприятие автора занимается разработкой систем малой автоматизации и внедрением M2M-технологий (автоматизацией технологического оборудования, малых энергогенерирующих объектов, объектов распределённых инфраструктур, систем охраны и т.д.). В работе используются микроконтроллерные платформы собственной разработки – так называемые «малые встраиваемые системы» (Low-end systems согласно классификации Embedded Systems Research), как правило, работающие в режиме жёсткого реального времени под управлением простейших операционных систем или без них, общей ценой до 500 долл., разрабатываемые силами одного-двух человек.

Обычно при разработке программного обеспечения (ПО) используется язык Си, реже – Ассемблер. Большая часть разработок начинается с готовых шаблонов, включающих операционную систему, набор коммуникационных модулей, фрагменты пользовательского интерфейса. Сама прикладная функциональность, специфицированная заказчиком, может занимать небольшую долю всего ПО, но она же является и его самой динамичной частью.

В нашей практике часто случается так, что множество подобных объектов с одинаковой микроконтроллерной платформой требуют нескольких различных алгоритмов управления или даже сценариев использования, причём нередко модифицируемых со временем. В таких случаях незначительные отличия с точки зрения заказчика для разработчиков могут оборачиваться неделями отладки и поиска программных ошибок после кажущейся «безобидной» коррекции исходных текстов. Не менее трудоёмко восстановление в памяти контекста старых проектов или попытки нахождения общего языка с заказчиком при описании функционирования проектируемых систем. Даже наличие наработанных библиотек и согласованных правил оформления исходных текстов и документации недостаточно ускоряет процесс быстрой адаптации ПО под меняющиеся требования.

Мы стали искать способы сделать менее трудоёмким процесс программирования или отдельные его этапы; возможно даже, возложить часть программирования на самого заказчика.

Подобные задачи решают программируемые логические контроллеры (ПЛК) с их языками функциональных блоков и SCADA-системы. Индустрия ПЛК развита очень сильно, здесь можно найти решения очень многих задач. Но часто по критерию цена – эффективность – функциональность в условиях непрерывного роста мощности микроконтроллеров ПЛК проигрывают на поле малых встраиваемых систем.

Способов облегчить различные аспекты программирования встраиваемых систем на самом деле придумано немало: аппаратно-независимые языки типа Java, объектно-ориентированные надстройки типа SystemC, специализированные и гибридные языки под специфику прикладных задач, сложная параметризация вместо программирования и т.д. Мы остановились на технологии визуальной разработки ПО как наиболее привлекательной и простой в освоении.

Тем, кто занимается малой автоматизацией, хорошо знакомы ПЛК серий LOGO фирмы Siemens, ALPHA фирмы Mitsubishi и др. Они пользуются большой популярностью во многом благодаря простоте создания управляющих программ. Кроме того, поставляются бесплатные версии инструментальных программ, при помощи которых алгоритм функционирования контроллера создаётся в графическом виде из связываемых в определённой последовательности функциональных блоков.

В мире наблюдается всё более широкое применение концепции разработки ПО, управляемого моделью (MDD, model-driven development), с использованием нотации UML (унифицированный язык моделирования) и методов объектно-ориентированного программирования. Существующая диспропорция между ростом производительности микроконтроллеров и микропроцессоров (70% в год), средств разработки аппаратного обеспечения (35% в год) и создания ПО (10% в год) заставляет сообщество программистов искать новые технологии создания ПО. В частности, хорошие перспективы есть у MDD, и хотя там и преследуются цели, несколько отличные от наших, тем не менее, визуальная разработка – важнейшая составляющая этой модели.

Анализ вариантов

Рассмотрим предлагаемые на рынке продукты, которые могут быть пригодны для нашей цели. При этом помним, что технология не должна быть дорогой для пользователя и не должна требовать от него значительных усилий по её освоению.

Simulink фирмы MathWorks [1]

Это довольно известный продукт в среде профессиональных разработчиков алгоритмов. Он представляет собой одно из самых мощных средств имитационного моделирования как простых, так и сложных систем, преимущественно для цифровой обработки данных. Продукт снабжается специальными инструментами разработки ПО для определённых микропроцессорных платформ (например, C167, HC12, MPC555, TI C2000, C6000), список которых постоянно пополняется (см. рис. 1).

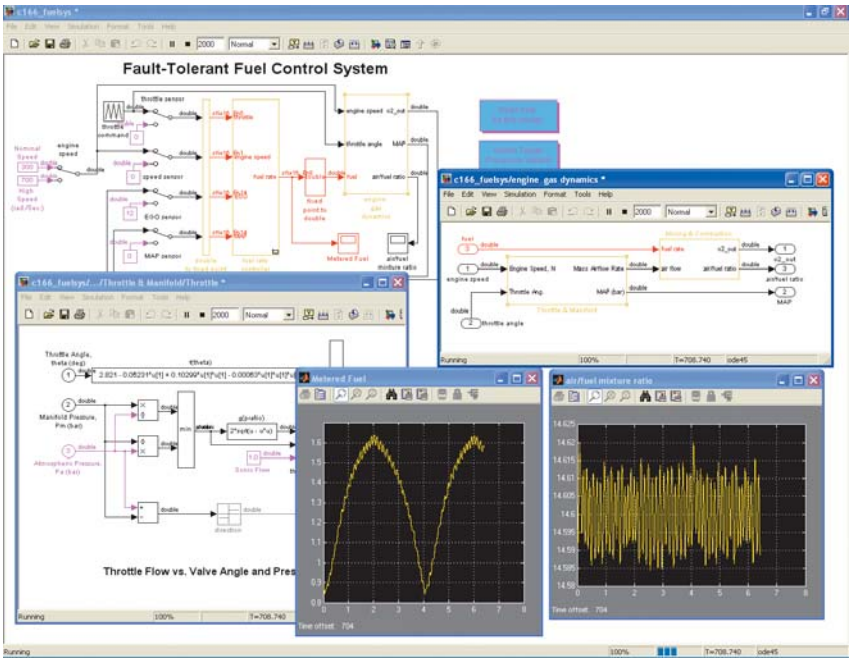

Рис. 1. Окно работающего пакета Simulink

Идёт симуляция модели системы управления подачи топлива в двигатель, предназначенной для 16-битных МК серий C167, XC167, ST10 и оптимизированной для вычислений с фиксированной запятой. Модель системы управления может быть конвертирована и загружена в реальный контроллер без дополнительного редактирования

Пакет Simulink поддерживает построение моделей на основе функциональных блоков и моделей в виде диаграмм перехода состояний. Он имеет библиотеку, содержащую около пятисот функциональных блоков, разбитых по тематикам: от нейросетей и нечёткой логики до механики, электроники и связи, из которых может строиться модель системы. Блоки могут выполнять как элементарные математические и логические операции, так и сложные функции, например, скрэмблерование, сжатие данных, матричные операции и т.д. Среда Simulink позволяет создавать пользовательские функциональные блоки и элементы пользовательского интерфейса, благодаря чему её можно настраивать под узкоспециализированные применения. Уникальна среда Simulink ещё и тем, что способна генерировать алгоритмы как для работы с дискретным временем (сигналы в модели не изменяются между циклами обработки), так и с непрерывным (сигналы изменяются между циклами обработки), их сочетанием, а также алгоритмы на основе конечных автоматов. Благодаря этому в Simulink одинаково легко исследовать электрические цепи, решать дифференциальные уравнения, проектировать цифровые фильтры или разрабатывать алгоритмы автоматизированного управления. Некоторые фирмы производят контроллеры, специально предназначенные для выполнения программ, созданных в Simulink: например, серия 8x38 фирмы ICP DAS.

Программа создаётся в несколько этапов. Сначала из функциональных блоков и диаграмм перехода состояний строится графическая модель программы. Производится симуляция работы алгоритма и его отладка. Потом при помощи инструмента Real-Time Workshop Embedded Coder модель конвертируется в текст программы на языке Си. После этого программа компилируется кросс-средствами под определённую микропроцессорную платформу/контроллер. Средствами Simulink можно также отлаживать и параметрировать программу и на целевой платформе (т.е. на реальном устройстве) после её загрузки.

Однако такая технология, безусловно, дорога и сложна в освоении. Пакет Simulink входит как компонент в общую систему математических расчётов и моделирования MATLAB. Достаточно сказать, что дистрибутив MATLAB 7 в сжатом виде имеет размер более 1 Гб.

Помимо этого, пользователю необходимо иметь кросс-средства для используемой им целевой платформы: компилятор языка Си, линкер, отладчик и т.п.

С некоторой точки зрения, использование языка Си на промежуточном этапе полностью ставит крест на идее простоты технологии. Известно, что развитые кросс-средства для Си весьма дороги, кроме того, одного текста программы, выполненного в стандарте ANSI Си, совсем не достаточно для превращения его в исполняемый модуль. Спецификаторы памяти, особенности приведения типов, синтаксиса наречий языка Си в разных компиляторах, обход известных ошибок семантических анализаторов в оптимизирующих компиляторах, учёт упрощений реализации спецификации ANSI Си в стандартных библиотеках компиляторов, углубление в детали аппаратной реализации платформ, оценка и планирование ресурсов памяти и стека, профайлинг (анализ времени выполнения различных участков программы) – вот неполный перечень проблем, к которым привычны разработчики встраиваемых систем, но которые будут непосильны для рядового пользователя. Не случайно модели, генерируемые в Simulink для встроенных приложений, накладывают большое число ограничений на используемые в них блоки и операторы, а также требуют применения только определённых компиляторов.

Естественно, есть исключения, когда компилятор тесно интегрирован с визуальной средой разработки и явно не заставляет пользователя использовать Си. Но тогда производители подобных систем не рискуют генерировать весь исполняемый код для целевой платформы, а предпочитают использовать некую подготовленную пользователем среду исполнения на целевой платформе, гото-

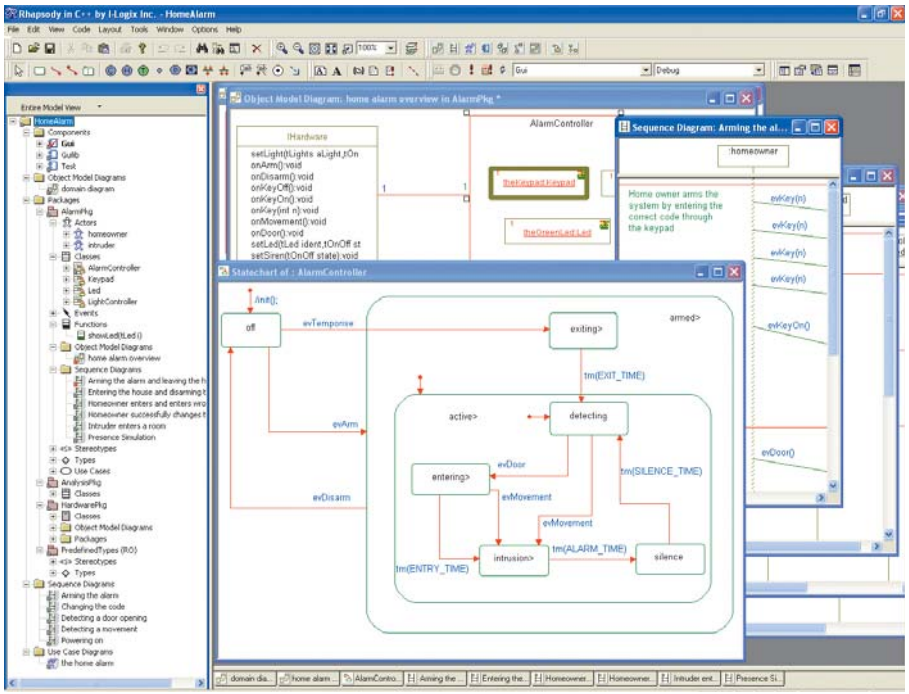

Рис. 2. Окно проекта домашней охранной системы в Rhapsody

Слева – браузер проекта. Справа – диаграммы в нотации UML. Десятки диаграмм для сравнительно простой функциональности – расплата за полноту модели и возможность автоматической генерации кода

вую принять сгенерированный код. Код в таком случае лишён низкоуровневых функций работы с периферией и достаточно независим от архитектуры платформы, а средой может быть либо развитая операционная система, либо специализированные микропрограммные ядра, написанные в соответствии со строгими спецификациями. Такие компиляторы вынуждены быть весьма консервативными: не поддерживать оптимизацию, базироваться только на наиболее общем подмножестве команд микропроцессорных ядер, поддерживать весьма ограниченный набор микропроцессорных ядер.

Rhapsody фирмы I-Logix [2]

Пакет Rhapsody является визуальной средой разработки приложений реального времени для встраиваемых систем. Можно упомянуть также почти аналогичную систему Telelogic TAU. В основу положена концепция разработки управляемой моделью – MDD, и в этом заключается качественное отличие от Simulink. В то время как Simulink в основном специализирован для разработки алгоритмов обработки данных и в меньшей степени – для разработки систем, реагирующих на внешние сигналы и управляемых событиями, Rhapsody решает более глобальные задачи. Основная – это сведение этапов анализа, проектирования, кодирования и тестирования ПО в один технологический процесс на основе модели проектируемой системы. Отсюда и название технологии. На этапе анализа образ модели зарождается; к этапу тестирования модель принимает окончательную форму. Все участники разработки: аналитик, проектировщик, кодировщик и тестер работают с единой документированной моделью – это облегчает их взаимодействие, сокращает количество ошибок, автоматизирует кодирование, ускоряет тестирование, повышает общую производительность труда. Для описания модели используется графическая нотация UML. Язык UML – унифицированный язык описания моделей, использующий принципы объектно-ориентированного программирования. Для всестороннего описания модели в нотации UML предлагается девять видов диаграмм: прецедентов, классов, объектов, кооперации, последовательности, состояний, деятельности, компонентов и развёртывания. В рамках нотации UML можно провести весь цикл проектирования ПО, включающий следующие этапы: задание и моделирование функциональных требований (диаграммы прецедентов), построение статической модели, выявление в системе классов объектов и декомпозиция (диаграммы классов и объектов), построение динамических моделей классов на основе конечных автоматов (диаграммы перехода состояний), построение моделей взаимодействия объектов (диаграммы кооперации и последовательности), разработка организации программных модулей и взаимозависимостей (диаграммы компонентов), разработка физической конфигурации системы (диаграммы развёртывания). Все диаграммы в совокупности и представляют модель системы в технологии MDD (см. рис. 2).

На основе модели пакет Rhapsody позволяет сгенерировать для встраиваемой системы исходные тексты на языках Си, C++, Java или Ada. Отсюда и тот же недостаток, что у Simulink, хотя и в меньшей степени. Некоторые производители компиляторов, в частности, фирма Green Hills, производящая компиляторы Multi 2000 для различных микропроцессорных платформ, сами разрабатывают отладочные инструменты для работы с Rhapsody. Rhapsody создаёт код, работающий под управлением ОС, таких как Windows, Linux, Integrity, VxWorks, QNX и т.д. Принципиально эти ОС могут быть размещены в малых встраиваемых системах, однако это довольно дорого. Для создания кода, выполняющегося на контроллерах без операционной системы, Rhapsody предлагает среду, управляемую прерываниями, но в этом случае разработчик должен сам позаботиться обо всех низкоуровневых функциях работы с периферией.

В UML нет места функциональным блокам, поэтому для написания алгоритмов обработки данных приходится обращаться всё к тем же старым текстовым нотациям, включаемым, например, в диаграммы перехода состояний.

Однако в пятой версии Rhapsody уже введена возможность построения диаграмм на основе функциональных блоков для создания моста между концепциями структурного и объектно-ориентированного программирования. Однако по-прежнему отсутствуют библиотеки функциональных блоков и эффективные инструменты работы с ними.

В целом для большинства пользователей встроенных систем парадигма MDD, придуманная профессиональными программистами, является, по нашему мнению, слишком

сложной. С другой стороны, для малых встраиваемых систем проблемы, решаемые Rhapsody, не являются актуальными. Пакет Rhapsody, так же как и Simulink, требует наличия у пользователя кросс-средств разработки на языке высокого уровня для целевой платформы. И это тоже становится серьёзным препятствием для использования Rhapsody при решении нашего круга задач.

IAR visualSTATE фирмы IAR [3]

Этот пакет концептуально напоминает Rhapsody, но значительно упрощён (см. рис. 3). Из всего стандарта UML оставлены только диаграммы состояний. Однако при этом фирма IAR сама разрабатывает хорошо зарекомендовавшие себя компиляторы языка Си для разнообразных микропроцессорных платформ. Таким образом, проблема совместимости сгенерированного кода на Си или C++ с компилятором легко решается. IAR visualSTATE ориентирован на более низкий уровень встраиваемых систем, чем Rhapsody, – на 8-, 16- и 32-разрядные микроконтроллеры без использования операционных систем или под управлением RTOS OSEK. Как и в Rhapsody, для «голых» платформ (при отсутствии RTOS) нельзя обойтись без дополнительного кодирования – пользователь должен самостоятельно писать драйверы аппаратных ресурсов, что для visualSTATE является совсем не простым делом.

Пакет visualSTATE можно применить для весьма ограниченного круга задач, он не является законченной системой, предусматривающей отладку и развёртывание на целевой платформе, и в целом не годится для обозначенных целей.

CoDeSys фирмы S3 [4]

Этот пакет – универсальный инструмент программирования, соответствующий стандарту IEC 61131-3, для широкого спектра целевых платформ и широко применяется для программирования промышленных логических контроллеров (ПЛК). Наряду с ним можно упомянуть пакеты ISaGRAF, UltraLogic, SOFTLOGIC и т.д., но рассчитанные на PC-совместимые системы или развитые операционные системы – QNX, VxWorks, OS/9 и т.д.

Среда CoDeSys позволяет создавать программы на пяти языках, описанных в IEC 61131-3. Большинство ПЛК малого и среднего класса используют ту или иную вариацию языков из IEC 61131-3. Три из них представляют собой графическую нотацию (у языков последовательных функциональных схем (SFC), функциональных блоковых диаграмм (FBD) и релейных диаграмм (LD)), и два – текстовую. В CoDeSys есть также язык в графической нотации, не стандартизированный IEC, – язык непрерывных функциональных схем (CFC). Язык SFC представляет собой некоторое подобие диаграммы состояний в UML, описывающей конечный автомат с той разницей, что в SFC может быть несколько одновременно активных состояний. Язык FBD применяется для построения простых цепей из функциональных блоков с довольно ограниченными возможностями. Язык LD во многом копирует стиль старых релейных управляющих схем, применявшихся до появления логических контроллеров. Язык CFC (см. рис. 4) напоминает язык функциональных блоков, применяющийся в Simulink, но с меньшими возможностями.

Для работы с CoDeSys не требуется наличия на целевой платформе ОС, но на ней обязательно должна присутствовать специализированная система времени выполнения, называемая RTS. При отсутствии операционной системы RTS берёт на себя некоторые её функции, а также предоставляет коммуникационные, отладочные и другие сервисы, необходимые для стыковки со средой разработки на PC. Для 8-, 16- и 32-разрядных микроконтроллеров необходимы разные типы RTS.

Среда разработки CoDeSys позволяет симулировать выполнение программы, загружать и отлаживать её на целевой платформе. Кроме того, CoDeSys снабжена развитыми средствами визуализации на PC для контроля процессов, происходящих в ПЛК. Однако элементы визуализации – анимированные индикаторы, кнопки, графики, рисунки и т.д. невозможно разместить на самих диаграммах, как, например, в Simulink. Для этого требуется создавать отдельные окна визуализации. Поведение элементов визуализации определяется глобальными переменными, назначаемыми пользователем вручную и предварительно описанными текстовой нотацией в окнах диаграмм. В CoDeSys также есть инструменты для разработки интерфейсов человек-машина (HMI) для самих ПЛК, снабжённых дисплеем и клавиатурой.

Ориентацией системы на промышленные контроллеры продиктовано наличие таких компонент, как OPC-

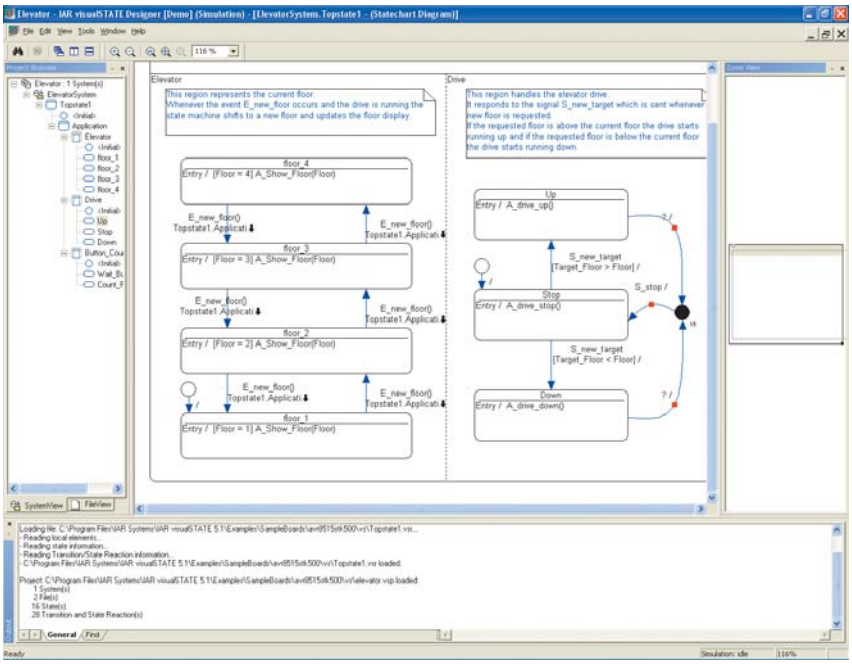

Рис. 3. Упрощённая модель системы управления лифтом для 8-битного МК серии AVR в окне построителя диаграмм состояний пакета visualSTATE

Скудность инструментов создания модели компенсируется полной интеграцией с кросс-средствами языка Си для большого класса микроконтроллеров, включающего серии AVR, PIC, NEC 78K, NEC V850, ARM7, MSP430, M16C, R8C

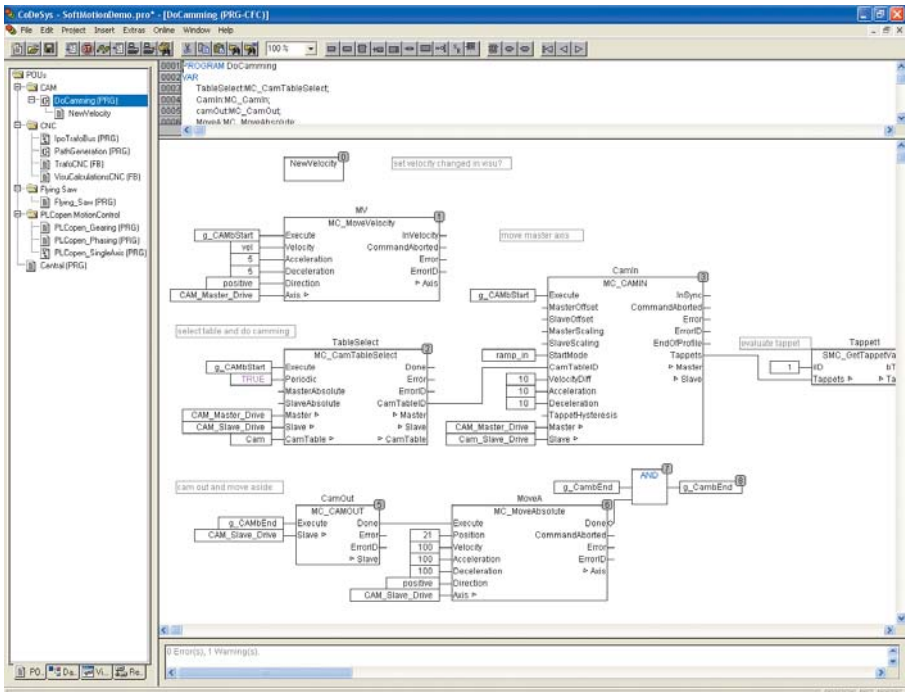

Рис. 4. Среда разработки CoDeSys

Пример программы в нотации CFC. Следование стандартам приводит к сухости графической нотации, мало чем выигрывающей перед текстовой. Для визуализации в CoDeSys нужно строить отдельные окна

сервер для интеграции ПЛК в SCADA-программы и модули работы с промышленными протоколами связи.

Для того чтобы пользователь мог переносить программы, созданные в CoDeSys, во встраиваемую систему, он предварительно должен адаптировать и перенести туда RTS. CoDeSys компилирует диаграммы непосредственно в код целевой платформы; для этого в систему встроены компиляторы для некоторого набора семейств микроконтроллеров – от 8- до 32-разрядных (лицензия на каждый продаётся отдельно). Встроенные компиляторы поддерживают обычно только базовый набор команд семейств микроконтроллеров, и для создания по-настоящему эффективных программ на этапе подготовки целевой платформы может понадобиться создание библиотек при помощи внешних оптимизирующих компиляторов.

Фирма S3 применяет систему лицензирования, согласно которой за каждый выпущенный контроллер с RTS производитель ПЛК выплачивает фирме определённые отчисления, но при этом саму среду CoDeSys получает бесплатно. Для разработки ПЛК производитель также, естественно, должен приобрести и саму RTS.

В целом CoDeSys мог бы стать приемлемым вариантом, если бы мы не нашли более дешёвой альтернативы.

ICon-L фирмы ProSign [5]

Итак, поиск привёл нас к продукту фирмы ProSign, называемому iCon-L. Идея этого пакета во многом повторяет CoDeSys, но рассчитан он на более широкий круг применений, чем только промышленная автоматизация. iCon-L поддерживает два языка в графической нотации: язык функциональных блоков, отражающий потоки данных, и язык блок-схем, отражающий потоки управления. Язык функциональных блоков по выразительности ничем не уступает аналогичному языку в Simulink и значительно превосходит аналогичный в CoDeSys. Поэтому iCon-L находит применение в самых различных областях – от управления детскими роботами (см. рис. 5) до оригинальных преобразователей протоколов (CAN, RS-485, Ethernet и т.д.) и промышленных ПЛК (см. рис. 6, 7).

Создание программ в среде iCon-L интуитивно понятно пользователю (см. рис. 8). Это одна из немногих систем, где не приходится применять текстовую нотацию в разработке программы. Здесь не нужно заблаговременно объявлять никаких переменных и не нужно знать никаких подробностей о карте памяти целевой платформы/контроллера или типе процессорного ядра, как, например, в CoDeSys. Элементы визуализации внедряются в саму диаграмму с функциональными блоками, что значительно повышает «читабельность» программы. Диаграммы могут иметь иерархическую структуру и поддерживают макросы. Человеко-машинный интерфейс целевой платформы может быть спроектирован на той же диаграмме функциональных блоков благодаря уникальной гибкости среды разработки iCon-L. Помимо большой библиотеки стандартных и специализированных функциональных блоков пользователи iCon-L имеют

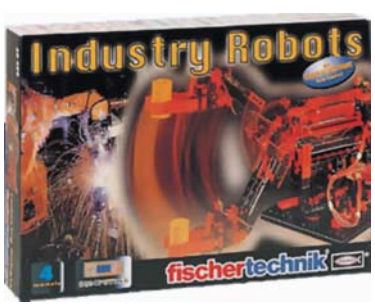

Рис. 5. Роботы-конструкторы фирмы Fischertechnik

Роботы, собранные из конструкторов, снабжаются встраиваемой системой с 16-битным МК, способным выполнять программы, созданные в iCon-L

Рис. 6. Микроминиатюрный контроллер/web-сервер фирмы BECK в корпусе DIL32 размером 22 × 44 × 9,5 мм

Контроллер снабжён 16-битным МК, работающим на частоте 40 МГц, Ethernet-интерфейсом, Flash-памятью, ОЗУ, портами ввода/вывода и RTOS. Изделие находит применение в ПЛК фирмы FESTO, широко используется во встраиваемых системах и служит идеальной платформой для iCon-L

Рис. 7. Промышленные PC-совместимые контроллеры фирмы Beckhoff серии CX1000

Для них был разработан весьма недорогой вариант интеграции с iCon-L с использованием ОС Linux и надстройкой RTAI, обеспечивающей работу Linux в реальном времени

возможность создавать собственные заготовки функциональных блоков. Стандартная библиотека включает группы блоков, реализующие логические операции (включая различные триггеры и сдвигающие регистры), арифметические операции, математические функции, вычисление произвольных функций, PID, PI и другие регуляторы, нелинейные преобразователи, селекторы, мультиплексоры, компараторы, блоки ввода/вывода, архивирования данных в памяти, работы со временем, строками, блоки визуализации и многое другое. Данные могут быть битами, целыми числами, длинными целыми числами, числами с плавающей запятой, строками, массивами из перечисленных типов. Кроме того, могут быть созданы пользовательские типы данных.

Другой положительной чертой iCon-L является совместимость набора функциональных блоков со стандартом IEC61131-3, благодаря чему программы из промышленных ПЛК легко могут быть перенесены в iCon-L. Более того, на смену устаревающему стандарту IEC61131-3 приходит новый стандарт – IEC61499, лучше учитывающий особенности систем, работающих в реальном времени, и распределённых систем. Примером того, как в iCon-L можно реализовать программы, управляемые событиями согласно IEC61499 в противовес обычному циклическому выполнению, может служить пакет DACHSview фирмы Steinhoff, основанный на iCon-L.

Обычный сценарий использования iCon-L для встраиваемой системы таков. Сначала разработчик покупает лицензию на среду разработки iCon-L и адаптирует визуальную оболочку iCon-L для будущих задач, используя средства разработки на языке Си. Для этого он создаёт дополнительные специализированные библиотеки функциональных блоков (ФБ) для визуальной оболочки и целевой платформы. Затем необходимо создать саму целевую платформу или подготовить уже имеющуюся. После этого подготовленные устройство и визуальная среда разработки iCon-L передаётся конечному пользователю, который в дальнейшем разрабатывает свои программы, используя только графическую нотацию среды iCon-L.

Система iCon-L существенно дешевле пакетов-конкурентов благодаря оригинальной технологии однократной компиляции. Идея заключается в том, что код функциональных блоков для контроллера компилируется разработчиками один раз на этапе подготовки системы времени исполнения для целевой платформы и остаётся в постоянной памяти контроллера. При загрузке же программы пользователя в контроллер iCon-L не производит компиляции в обычном смысле, а подготавливает связанную таблицу вызовов функций блоков по известным адресам в контроллере. Эта таблица затем и служит программой для исполняющей системы контроллера. Таким образом, для работы iCon-L не требуется компилятор, что значительно удешевляет среду разработки, при этом достигается большая надёжность создаваемых программ и большая временная детерминированность по сравнению с регулярно перекомпилируемым кодом обычных систем. Известно, что обычной практикой для профайлинга и контроля качества генерируемого машинного кода является идентификация и просмотр ассемблерных текстов, создаваемых компиляторами. Для систем с жёстким реальным временем или бюджетных систем с малыми резервами временных ресурсов разработчикам приходится раз за разом использовать этот способ при оптимизации программ. В случае однократной компиляции разработчик может быстрее и проще оценить потребность программы во временных ресурсах и ресурсах памяти либо даже автоматизировать этот процесс, не прибегая к дорогостоящим статистическим методам анализа на реальных объектах, поскольку профайлинг может быть выполнен ещё на стадии подготовки целевой платформы.

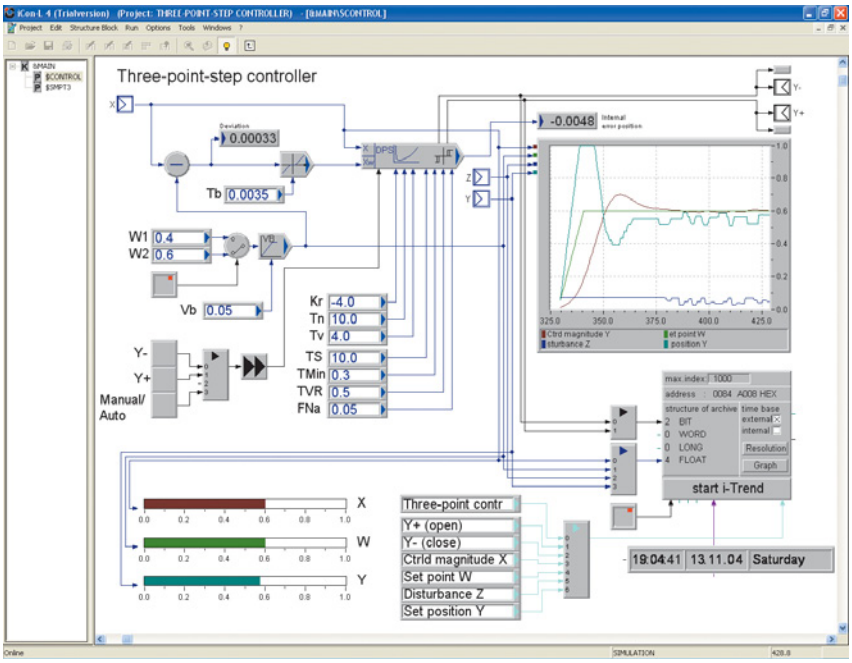

Рис. 8. Пример программы в среде iCon-L

Исполнение программы по связанной таблице происходит быстрее, чем интерпретируемой программы, и медленнее, чем обычной скомпилированной программы, но благодаря высокой оптимизации прекомпилированных функциональных блоков в системе времени исполнения проигрыш в быстродействии оказывается незначащим.

Центром архитектуры iCon-L (см. рис. 9) является визуальная среда разработки с одноимённым названием. В ней производится создание программы из функциональных блоков, симуляция её выполнения, загрузка в контроллер и наблюдение за работой программы на контроллере при помощи функциональных блоков визуализации.

Как и в CoDeSys, целевая платформа/контроллер для работы с iCon-L должна содержать специализированную систему времени исполнения. Здесь она называется VICK. На диаграмме компоненты VICK изображены чёрным цветом. VICK предлагается фирмой ProSign в виде исходных текстов на языке ANSI Си. Исходные тексты VICK представляют собой хорошо структурированный набор мо-

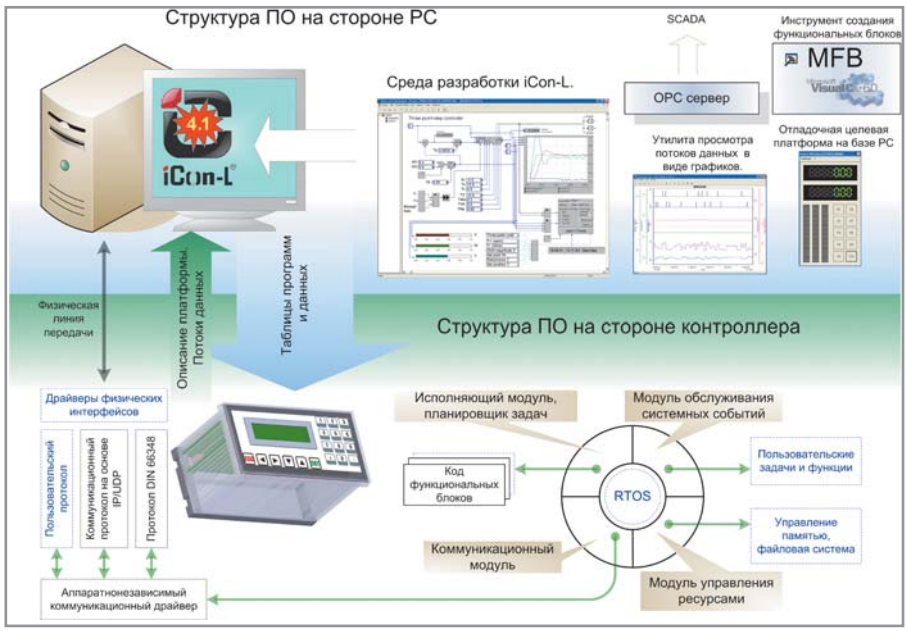

Рис. 9. Архитектура iCon-L

дулей с минимальной аппаратной зависимостью, поэтому они могут быть с одинаковым успехом скомпилированы и для 8-, 16-, и для 32-разрядных платформ (например, 8-разрядных контроллеров серии PIC18 фирмы Microchip, 16-разрядных МК серии C166 и 32-разрядных МК на базе ядра ARM).

Для нормального функционирования VICK на конкретной платформе разработчики также должны реализовать небольшое количество низкоуровневых функций работы с аппаратной частью (на рис. 9 показаны синим пунктиром). Это драйверы физических интерфейсов, ввода/вывода, функции чтения/записи энергонезависимой памяти, генерации временных интервалов и т.д. ОС реального времени может не использоваться, поскольку VICK сам предоставляет механизм кооперативной многозадачности. Система может поддерживать до 15 задач с различными уровнями приоритетности и независимыми продолжительностями циклов выполнения. Однако наличие вытесняющей RTOS упрощает создание драйверов и помогает оптимизировать использование ресурсов микроконтроллера.

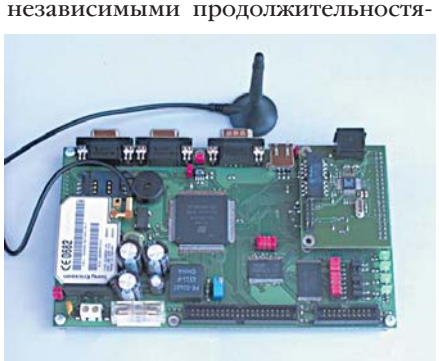

Рис. 10. Тестовая целевая платформа для работы с iCon-L

Пример целевой платформы

Нами была проведена адаптация VICK для целевой платформы на базе 16-битного микроконтроллера ST10F269 с тактовой частотой 40 МГц (см. рис. 10). Платформа располагает двумя интерфейсами RS-232, двумя CAN-интерфейсами, интерфейсом Ethernet 10Base-T, интерфейсом графического дисплея, Flash-памятью ёмкостью 2 Мб и ОЗУ ёмкостью 512 Кб, встроенным GSM-GPRS-модемом и энергонезависимыми часами реального времени.

Кратко опишем принципиальную схему целевой платформы (см. рис. 11, 12).

Микроконтроллер DD1 в рабочем варианте выполняет программу из своей внутренней памяти программ, но для целей отладки в схеме предусмотрено ОЗУ большой ёмкости (512 Кб) DD2, куда можно загрузить отладочный вариант программ вместе с модулем отладчика (отладочный монитор). Благодаря размещению программы в ОЗУ и отладочному монитору специальная программа-отладчик на PC может устанавливать в программе контроллера точки останова, выполнять её по шагам, просматривать и изменять содержимое всех регистров микроконтроллера и ячеек внешнего ОЗУ. При обращении к внешнему ОЗУ МК использует самый быстрый 16-битный режим работы внешней шины. Для изменения области выполнения программы с внутренней на внешнюю нужно изменить положение перемычки External bus enable в группе перемычек J4. Группами перемычек J4, J6 также задаются все важнейшие параметры работы МК перед стартом программы после сброса: частота тактового генератора, тип внешней шины и её разрядность, тип управления внешней шиной, режим начальной загрузки и т.д. Режим начальной загрузки позволяет загрузить во внутреннее ОЗУ МК данные или программу и запустить её до того, как МК начнёт выполнять собственную программу. Этот режим используется при отладке и программировании памяти программ МК.

Для хранения файлов и программ пользователя в среде VICK используется внешняя Flash-память DD3. В этой памяти хранятся файлы системных настроек, конфигурации VICK, файлы статических HTML- и WML-страниц (для доступа с мобильных устройств), графика, журналы системных событий и т.д. Линейная файловая система, реализованная в этой памяти, обеспечивает очень быстрый доступ к данным, так как в ней исключена фрагментация. Но при этом на одном логическом разделе файловой системы на запись может быть открыт только один файл. Эта трудность преодолевается созданием достаточного количества логических разделов.

Наличие только одного встроенного UART вынудило применить дополнительный внешний UART DD4. Использование драйвера USB-интерфейса DD9 было продиктовано желанием ускорить обмен с PC.

Микросхема часов реального времени DD6 при отсутствии напряжения питания поддерживает ход часов от литиевой батарейки GB1. Её ресурса должно хватать на 10 лет автономной работы. Точность хода часов при указанной марке кварцевого резонатора – около нескольких секунд в месяц в ту или иную сторону в зависимости от температуры окружающей среды.

На микросхеме DA1 собран узел внешнего сброса, внешний сторожевой таймер и схема предупреждения о снижении напряжения питания.

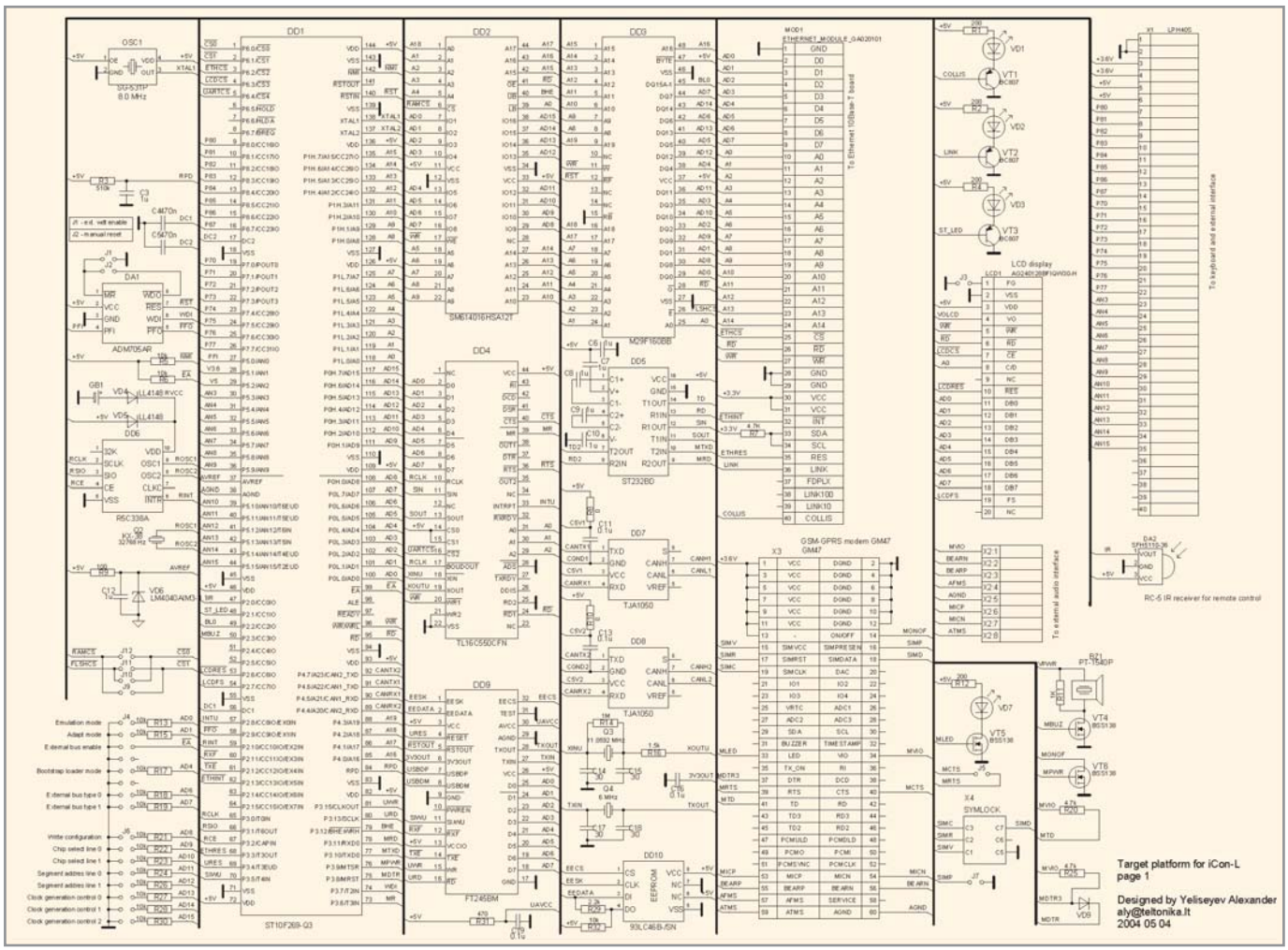

Рис. 11. Схема процессорного модуля целевой платформы

DD7 и DD8 являются физическими драйверами шины CAN. Шине CAN многие эксперты прочат роль последней конкурентоспособной «полевой» шины после того, как эта область будет завоевана шиной Industrial Ethernet. Но покуда Ethernet ещё достаточно дорог, мы применяем для связи с внешними модулями ввода/вывода шину CAN. Эта шина аппаратно реализует все жизненно важные функции, которые, например, в RS-485 приходилось выполнять программно: контроль целостности, квитирование, повторные передачи, арбитраж и т.д. Два CAN-интерфейса позволяют превратить устройство в повторитель или преобразователь протоколов для сетей CAN. На прикладном уровне для работы по CAN был использован протокол CANopen, который применяется во многих промышленных контроллерах и поддерживается многими устройствами и модулями ввода/вывода. CANopen интересен тем, что в режиме обмена в реальном времени он не вносит никаких дополнительных заголовочных или служебных данных в пакеты CAN, как это обычно принято в стеках стандартных протоколов, таким образом нисколько не снижая показатели скорости передачи данных по CAN.

GSM-модуль – непременный атрибут M2M-систем. На схеме применён GM47. Его характерная особенность в том, что он, как и многие подобные модули, имеет нестандартные напряжения питания и импульсный режим потребления мощности, что вынуж-

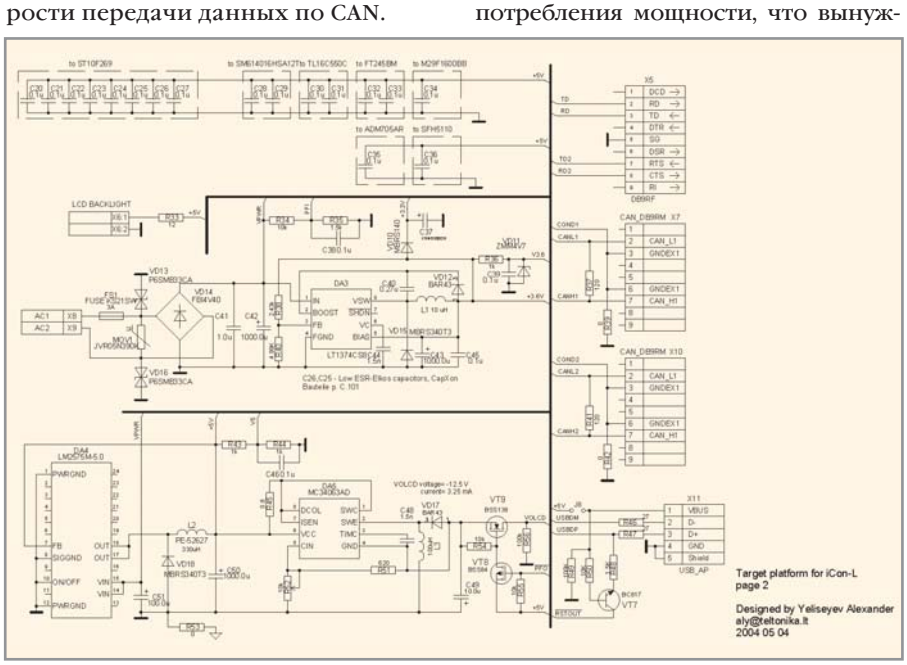

Рис. 12. Схема модуля питания целевой платформы

дает применять отдельные стабилизаторы напряжения и схемы согласования уровней сигналов.

Через разъём MOD1 к схеме подключается внешний модуль интерфейса Ethernet. Внешний модуль может быть выполнен по разным схемам; доступ к нему осуществляется через 8-битный параллельный интерфейс. Ethernet в системе используется для связи с удалёнными клиентами или серверами через Internet. Для работы через Internet был выбран стек протоколов TCP/IP из открытого проекта OpenTCP [6]. Для адаптации OpenTCP под конкретную платформу требуется лишь дописать программный драйвер для контроллера Ethernet. В нашем случае это драйвер для чипа W3100A фирмы WIZnet.

Разъём LCD1 предназначен для подключения графического дисплея – в нашем случае это графический монохромный ЖК-дисплей AG240128BFIQW30-H с разрешением 240×128. Графические дисплеи такого типа довольно дешевы, но характеризуются значительной инерционностью, поэтому в программе желательно использовать механизмы буферизированной прорисовки. Здесь можно порекомендовать известный пакет uC/GUI фирмы Micrium, который полностью совместим с нашей платформой.

Разъём X1 используется для подключения клавиатуры и дополнительных функциональных модулей.

Работа многочисленных драйверов и модулей на нашей платформе координируется простой операционной системой реального времени (ОСРВ или RTOS) uCOS-II фирмы Micrium.

VICK в нашей системе представляет одну из задач RTOS uCOS-II. Для коммуникации со средой разработки iCon-L контроллер может использовать либо RS-232, либо Ethernet. В качестве транспортного протокола по RS-232 VICK использует DIN 66348, а по Ethernet – протокол на основе IP/UDP. Поверх транспортных протоколов реализованы прикладные протоколы взаимодействия со средой разработки iCon-L, обмена файлами и работы в режиме командной строки. Для большей гибкости дополнительно был реализован обмен файлами через Ethernet по протоколу TFTP. Для настройки рабочих параметров контроллера, не связанных с системой iCon-L, был выбран протокол HTTP, позволяющий взаимодействовать с контроллером через обычный web-браузер.

Для разработчиков всегда важны цифры, позволяющие оценить ресурсы памяти, требуемые для реализации системы на целевой платформе. Приведём несколько величин, характеризующих ресурсопотребление VICK на нашей платформе:

- объём постоянной памяти, занимаемый ядром VICK, – 39 Кб;
- объём оперативной памяти для VICK, не включая коммуникационные буферы, – 1,8 Кб;
- объём постоянной памяти, занимаемый кодом 157 функциональных блоков VICK, – 29 Кб;
- объём памяти для коммуникационных буферов и областей, содержащих данные пользовательских программ, – 56 Кб.

Для сравнения приведём данные RTOS uCOS на нашей платформе:

- объём постоянной памяти, занимаемый RTOS, – 11 Кб;
- объём оперативной памяти для RTOS, включая системные стеки для 8 задач, – 3,3 Кб.

Для разработки применялся компилятор TASKING для ST10 версии 7.5. Как показано, VICK требует довольно скромных ресурсов, которые можно дополнительно сократить за счёт оптимизации состава библиотек и сервисов ядра. Всё программное обеспечение для нашей платформы разместилось в 256 Кб внутренней Flash-памяти ST10F269.

Не менее важны для разработчика параметры, характеризующие быстродействие системы. Здесь VICK также показывает неплохие результаты:

- выборка ссылки на очередной функциональный блок (ФБ) и передача ему управления – 2,2 мкс;
- выполнение ФБ сложения двух переменных типа длинное целое, включая передачу аргументов и сохранение результата, – 4,5 мкс;
- выполнение ФБ умножения двух переменных типа длинное целое, включая передачу аргументов и сохранение результата, – 5,9 мкс.

Нужно заметить, что ко времени выполнения каждого ФБ добавляется время выборки ссылки на этот блок с передачей управления. При выполнении простых блоков большую часть занимает время выборки аргументов и сохранения результатов. Поэтому для повышения быстродействия в программе целесообразно использовать более сложные блоки, выполняющие цепочки операций. Выбор ядра с более быстрыми операциями над длинными указателями также способствует повышению быстродействия.

Программа пользователя на целевой платформе обычно выполняется циклически. Пользователь сам выбирает время цикла и ограничен только минимальным значением этой величины. Если цикл выполняется раньше, то он приостанавливается до истечения установленного времени цикла и вновь выполняется с начала. Порядок выполнения ФБ в программе устанавливается самой средой iCon-L на основе информации о связях между блоками, однако пользователь при необходимости может его изменить. В нашей целевой платформе минимальная длительность цикла выполнения программы равна 10 мс. За такое время, например, контроллер может изменить по сети CAN состояние 720 аналоговых выходов или 5760 дискретных. Выбор такой длительности явился компромиссом между требуемым быстродействием пользовательских программ и коэффициентом использования ими вычислительных ресурсов процессора. При таком базовом цикле около 2,5% процессорного времени уходит на обслуживание сервисов RTOS. При уменьшении длительности цикла доля времени, отнимаемая RTOS, увеличивается, сокращая общее допустимое количество ФБ в пользовательских программах. Если пользовательская программа выполняется дольше, чем длится базовый цикл, или какие-либо сервисы RTOS начинают отнимать слишком много времени, исполняющий модуль VICK сразу же отправляет сообщение о нехватке вычислительных ресурсов в среду iCon-L. Кроме того, на нашей целевой платформе предусмотрена возможность генерации динамических HTML-страниц с информацией о состоянии и потребляемых ресурсах всех задач RTOS, позволяющая дистанционно через Internet диагностировать возможный дефицит ресурсов – как вычислительных, так и памяти.

Как уже говорилось, для встраиваемой системы, вероятнее всего, понадобится создание дополнительного

набора функциональных блоков, обеспечивающих функции ввода/вывода для конкретной платформы, коммуникационные функции и т.д. Создание новых ФБ в iCon-L – процесс сравнительно нетрудный. Для этого служит набор инструментальных средств MFB.

Функциональный блок в iCon-L представляется тремя программными автономными компонентами: модулем визуального представления ФБ в среде iCon-L (двоичный файл с расширением .mco), модулем для симуляции функционирования ФБ (исполняемый в Windows файл с расширением .dll) и модуль ФБ для целевой платформы (исходный текст на Си или библиотека, подключаемые на этапе компиляции и сборки исполняемого модуля для целевой платформы). Все три компонента автоматически генерируются при помощи инструмента MFB из одного исходного текста, написанного на специальном гибридном языке PSL. Это довольно практичный вариант создания ФБ, поскольку при сравнительной простоте PSL, напоминающего Pascal, он позволяет описать все аспекты поведения ФБ в одном текстовом модуле и сгенерировать непротиворечивые программные компоненты ФБ. MFB для генерации dll-файлов использует среду разработки MS Visual C++. В PSL можно использовать богатый набор функций для рисования графических элементов ФБ, выполнения математических и логических операций, вызова внешних функций из dll-файлов, перехвата событий, работы с параметрами и данными и т.д. Автоматическая генерация программных модулей ФБ для целевой платформы не означает, что их нельзя написать вручную на Си или Ассемблере. Мы так и поступили, когда разрабатывали ФБ, представляющий собой внешний модуль ввода/вывода, подключённый по сети CAN. Сгенерированный текст был использован как шаблон с приёмом и возвратом параметров, а само тело функции переписано с использованием сервисов, предоставляемых RTOS. Таким образом, был выполнен блок, отправляющий сообщения по CAN согласно стандарту CANopen с использованием очереди сообщений вместо обычного механизма, использующего разделяемую память, что сэкономило ресурсы процессора и упростило программирование для конечных пользователей.

Разработчикам предоставляются широкие возможности по изменению поведения ФБ в среде iCon-L при помощи вызовов пользовательских функций из dll-файлов в ответ на различные события: щелчок мыши по ФБ, вставка/удаление ФБ, старт/останов программы и т.д. Пользовательские функции могут вызывать произвольные диалоги, передавать и принимать данные и параметры в ФБ, вести журналы событий и т.д. Также работой ФБ в iCon-L могут управлять внешние программы – посылая специальные сообщения в окно оболочки iCon-L.

Для интеграции в SCADA-пакеты система iCon-L предоставляет OPC-сервер – промежуточное программное обеспечение, выполненное с использованием технологии OLE, работающее в среде Windows и занимающееся передачей данных от ПЛК в SCADA-пакеты и обратно.

Если такое решение слишком дорого, то для взаимодействия с целевой платформой можно использовать программные модули на Си из ядра VICK для обмена по протоколу DIN 66348, со встраиванием их в собственные программы на PC. В целом идея использования средства визуальной разработки программ себя оправдала. Программа в iCon-L быстрее редактируется, значительно сокращается количество грубых логических промахов в алгоритмах, а число синтаксических ошибок по сравнению с разработкой программ на Си сводится к нулю. Возможность дистанционной отладки через Internet также даёт много преимуществ. Отпечатанная копия программы из iCon-L сама по себе может являться отличной документацией, поясняющей алгоритм функционирования системы, избавляя от дополнительной рутинной работы. Заказчик также может принять участие в создании программы. Легче решаются вопросы поддержки ПО. В файловую систему целевой платформы можно загрузить сам проект программы в iCon-L, чтобы восстановить его в случае необходимости, не используя архивов.

Одним словом, технология визуальной разработки наступает на всех направлениях. И разработчикам остаётся только не упустить шанс поднять производительность своего труда ещё на одну ступень.

Литература

1. www.mathworks.com.
2. www.ilogix.com.
3. www.iar.com.
4. www.3s-software.com.
5. www.pro-sign.de.
6. www.opentcp.org.

Новости мира News of the World Новости мира

Новая компьютерная программа подражает людям

Сотрудники Лидского университета (Великобритания) создали новую программу «искусственного интеллекта», которая, наблюдая за людьми, смогла изучить правила простейшей игры.

Работа CogVis (такое название дали учёные своей разработке) основана на анализе визуальной информации, передаваемой видеокамерой, а также аудиосигнала, снимаемого микрофонами. В ходе исследования добровольцам было предложено сыграть несколько раз в «камень, ножницы, бумага» при помощи специальных карточек с изображениями соответствующих предметов. Результат игры при этом объявлялся вслух. Сопоставив изображения на карточках с действиями игроков и их фразами, компьютер через некоторое время смог объявлять итоги очередного раунда, – сообщает New Scientist.

По словам одного из исследователей, работу программы можно сравнить с поведением ребенка. Аналогично тому, как дитя пытается повторить действия взрослых, CogVis подражает поведению людей. При этом разработчики постарались отказаться от сложных программных алгоритмов, заменив их реальной информацией, получаемой в ходе наблюдения.

Несмотря на то что во время эксперимента CogVis изучала правила одной из самых простых игр, учёные утверждают, что полученные результаты могут оказать огромное влияние на дальнейшее развитие систем «искусственного интеллекта». В ближайшее время сотрудники Лидского университета намерены обучить CogVis правилам игры в крестики-нолики и шашки. Не исключено, что именно CogVis послужит базой для самоадаптирующихся программ управления роботами.

http://science.compulenta.ru/