Windows Embedded CE 6.0 в системах реального времени

В статье рассказывается о возможностях ОС Windows Embedded CE 6.0 в аспекте систем реального времени. Описана методика аппаратного измерения задержек по обработке прерываний в Windows Embedded CE 6.0 и в качестве примера приведены результаты тестирования для платформы Beagleboard на базе процессора TI OMAP3530

ООО «Кварта Технологии», г. Москва

Если на заре разработки специализированных устройств, в том числе ориентированных на системы реального времени, приходилось вести разработку с нуля, то на текущий момент большую популярность имеют системы-конструкторы. Задача разработчика, использующего средства разработки подобных систем, заключается в выборе необходимых «блоков» системы, которые в совокупности сформируют базовый функционал устройства. Таким образом, значительно сокращается цикл разработки устройств, поскольку основное внимание в процессе разработки и тестирования уделяется лишь вновь добавленным функциям. Примером одной из таких систем-конструкторов является Windows Embedded CE. Отставляя за рамками статьи описание широкого спектра возможностей и технологий, поддерживаемых Windows Embedded CE [«Введение в Windows Embedded CE 6.0 R2» и «Windows Embedded CE 6.0 R2. Практическое руководство»], хотелось бы отметить следующие ключевые особенности:

1_малый размер образа исполнения. С развитием информационных технологий ограничения, накладываемые объемом носителя, как правило, отходят на второй план. Тем не менее чем меньше размер образа, тем меньше время готовности устройства;

2_компонентность системы. Это важный параметр для построения специализированных систем, благодаря которому достигается высокий уровень производительности, безопасности и надежности;

3_поддержка разнообразных процессорных архитектур. Фактически Windows Embedded CE является единственной ОС компании Microsoft, поддерживающей такой широкий спектр архитектур: ARM, MIPS, SH4 и X86;

4_и наконец, возможность, которая непосредственно относится к теме нашей статьи, – поддержка жесткого реального времени, чем в семействе Windows также может похвастаться только CE, начиная с версии 3.0.

Для разработчика, помимо возможностей самой системы, важна среда разработки, привычность и удобство инструментария. Visual Studio 2005, основной инструмент разработчика Windows Embedded CE 6.0, в полной мере отвечает современным требованиям по разработке и является, по сути, стандартом в среде разработчиков для Windows-платформы, и не только.

Завершая краткий обзор Windows Embedded CE 6.0, следует упомянуть о первоочередном моменте, который необходимо учитывать при принятии решения по разработке устройства на конкретной аппаратной платформе с использованием той или иной системы – это наличие пакета аппаратной поддержки BSP (Board Support Package). Для любой системы с поддержкой широкого спектра аппаратных платформ отчетливо видна тенденция абстрагирования компонентов системы от специфики конкретной платформы. Связующим звеном в данном случае как раз и выступает пакет аппаратной поддержки. Тема BSP заслуживает целого цикла статей, но остановимся лишь на основных моментах.

В зависимости от типа устройства требуемый уровень проработки BSP может быть различным. Например, для мультимедиа-устройства, помимо базовых функций ввода-вывода, желательно иметь поддержку аппаратного декодирования аудио- и видеоконтента, плавной отрисовки интерфейса средствами OpenGL или DirectDraw, а также расширенные коммуникативные возможности, включающие поддержку Wi-Fi, Bluetooth и пр. С другой стороны, для контроллера в системе контроля и управления технологическим процессом, в задачи которого входит формирование управляющих сигналов в соответствии с текущими входными параметрами, достаточно запустить минимально необходимый набор компонентов и обеспечить считывание и передачу сигналов по определенным шинам данных с учетом требований по работе в режиме реального времени. И если разработка BSP для контроллера может оказаться вполне посильной задачей, укладывающейся в финансовые и временные рамки проекта, то в случае богатых в функциональном плане устройств рациональнее использовать готовый пакет аппаратной поддержки, который, как правило, предоставляется на свободной или платной основе производителем аппаратной платформы или партнерами, специализирующимися на разработке BSP (Adeneo, BSquare и др).

Итак, обрисовав в общем нюансы разработки с использованием Windows Embedded CE, перейдем к более детальному описанию особенностей операционной системы, которые позволяют создавать на ее базе системы реального времени.

В первую очередь это планировщик с отличным от настольных систем Windows алгоритмом работы, а также архитектура подсистемы обработки прерываний. Принцип работы планировщика в многопоточной среде Windows Embedded CE достаточно прост и основывается на приоритетах. Для каждого потока (в контексте каждого процесса может быть запущено несколько потоков) задается значение приоритета: от 0 – наивысший приоритет до 255 – наименьший приоритет. По умолчанию поток имеет значение приоритета, равное 251. В соответствии со значением приоритета готовых к исполнению потоков в каждый момент времени исполняется поток с наименьшим его значением. Этот простой принцип позволяет настроить систему для обеспечения детерминизма исполнения. Другим, не менее важным показателем систем реального времени является задержка при обработке прерываний. Обработка прерываний в Windows Embedded CE состоит, в общем случае, из двух этапов. Первый этап заключается в детектировании источника прерывания и выполняется в невытесняемом (non-preemptive) режиме работы системы. В соответствии с этим работа первого этапа может быть прервана только в случае возникновения более высокоприоритетного прерывания – другие потоки системы в этот промежуток времени не исполняются. После того как будет определен источник прерывания, система устанавливает связанное с ним событие. Установка события служит сигналом для второй части, выполняющей непосредственную обработку и являющейся для системы обычным потоком. Таким образом, задержка при обработке прерывания состоит из двух частей. Если первая составляющая зависит исключительно от аппаратной части, ее архитектуры и производительности, то задержка до запуска потока обработки прерывания помимо характеристик аппаратной платформы определяется тем, как долго система находится в невытесняемом режиме и тем, как настроены значения приоритетов других потоков в системе. Наглядно описанная процедура обработки прерываний приведена на рис. 1, где ISR (процедура детектирования источника прерывания), IST (поток обработки прерывания), OAL (слой абстракции ядра от конкретной платформы), являются частью BSP. При использовании приведенной, классической для Windows Embedded CE, схемы обработки прерываний основным инструментом для улучшения значения задержек является настройка приоритетов потоков системы. В случае если добиться желаемого результата данным способом не получается, то следующим шагом является сокращение времени нахождения системы в невытесняемом режиме. Эта задача решается оптимизацией кода всех ISR, задействованных при работе системы, и в крайнем случае временным отключением прерываний. Если же приведенные ранее меры не дали должного эффекта, то есть возможность отказаться от двухуровневой схемы обработки прерываний и поместить весь код по обработке в ISR. При этом следует понимать, что во время работы ISR не будут детектироваться менее приоритетные прерывания и будет приостановлено исполнение всех потоков системы. Дополнительным поводом отказаться от обработки прерываний в ISR является тот факт, что в ISR доступен лишь

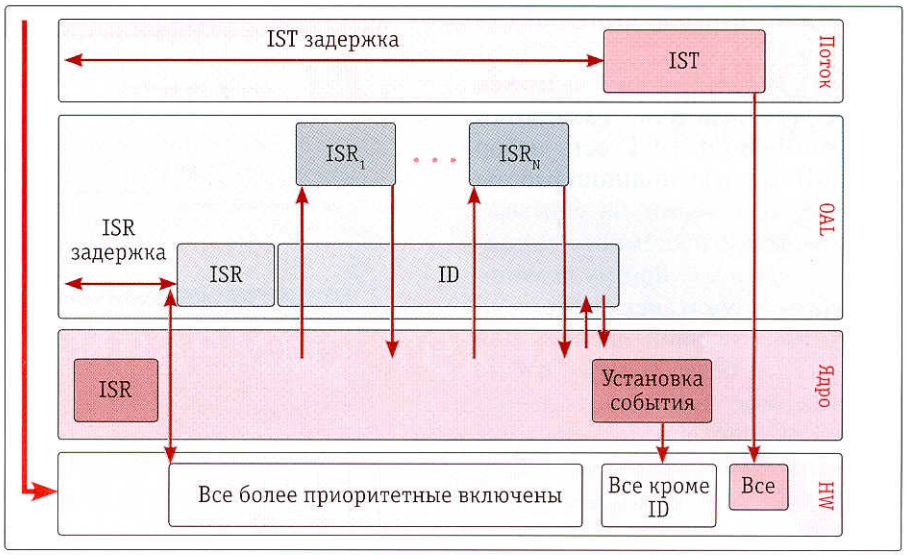

Рис. 1. Архитектура обработки прерываний

ограниченный набор функций системного API.

Для тестирования задержек в поставке средств разработки Windows Embedded CE есть набор утилит. Поскольку принцип работы этих утилит основан на считывании показаний системного таймера, то полученные при их помощи результаты служат исключительно для предварительной оценки. Для детального и более точного изучения задержек рекомендуется использовать аппаратные средства измерения. Познакомимся с одной из методик аппаратного измерения задержек.

Описание исследуемой системы

Исследования задержек будут проводиться на платформе BeagleBoard, а именно на EBVBeagle ревизии C2 (рис. 2), произведенной компанией EBV. Данная платформа отличается компактностью в сочетании с достаточно высокой производительностью – в ее основе лежит процессор TI OMAP3530 со встроенным блоком цифровой обработки данных (DSP). Проект BeagleBoard организован фактически энтузиастами и не имеет прямого отношения к производителю процессора – Texas Instruments. На текущий момент, как минимум

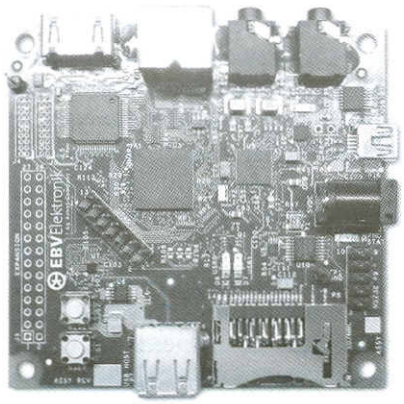

Рис. 2. Плата EBVBeagle

в публичном доступе, отсутствует BSP, полностью реализующее функции платформы BeagleBoard под Windows Embedded CE 6.0. При этом есть несколько BSP для платформ, использующих аналогичный процессор. Это BSP для отладочной платы EVM3530 и BSP для Gumstix Overo (http://www.gumstix.com), разработанное компанией

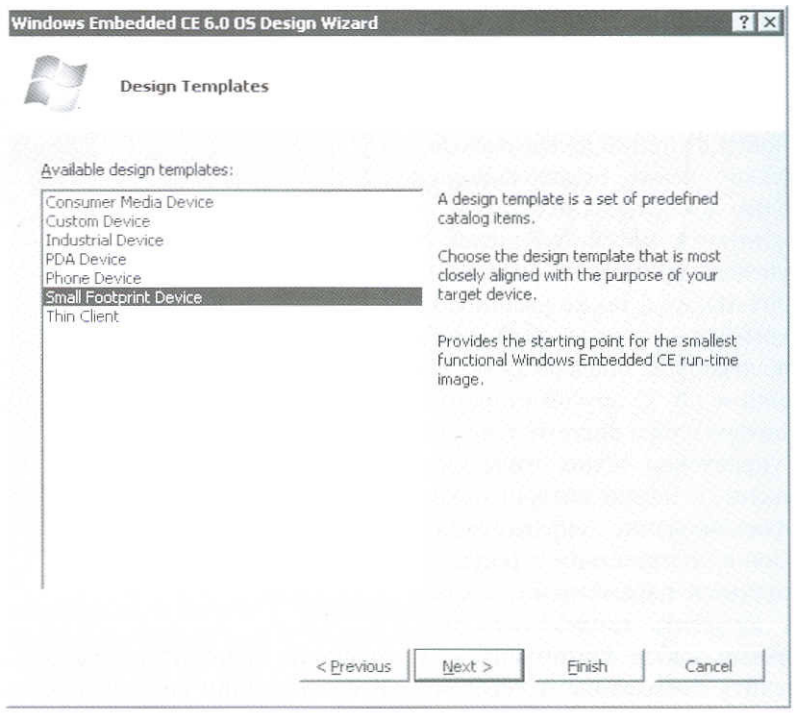

Рис. 3. Выбор шаблона операционной системы

Adeneo. Отладочная плата EVM3530 поддерживается Texas Instruments и с недавних пор BSP для нее доступно по запросу на сайте TI (http://ww.ti.com).

BSP предоставляется с определенными лицензионными ограничениями, в частности запрещена в том или ином виде публикация исходных кодов из его поставки. Благодаря тому что платформы используют один и тот же процессор, подавляющее количество функций – загрузчик, код инициализации платформы и некоторые драйверы – может быть использовано без доработки. При этом ввиду схемотехнических отличий в разводке плат и используемых вспомогательных микросхем необходима доработка кода инициализации DVI-выхода и USB-хост контроллера. Информация по доработке BSP EVM3530 размещена на ресурсе http://evmonbeagle.codeplex.com/ в теме EBVBeagle rev. C2 раздела Discussions. Для тестирования задержек по обработке прерываний BSP EVM3530 может быть использовано без дополнительной доработки.

Следующим этапом по подготовке к тестированию является сборка образа Windows Embedded CE. Преследуя исключительно цели тестирования задержек, воспользуемся минимальным образом. Для сборки минимального образа служит специальный шаблон Small Footprint Device, который выбирается в Visual Studio 2005 (рис. 3) при создании дизайна операционной системы. Данный шаблон включает минимальный набор модулей для запуска Windows Embedded CE и в случае BeagleBoard составил 540 Кбайт. В отличие от привычной многозадачной конфигурации Windows, как правило, с поддержкой менеджера окон в минимальном образе запущен только один процесс, собственно ядро, в контексте которого может исполняться формально неограниченное количество потоков (максимальное количество потоков фактически определяется возможностями аппаратной платформы). В данной конфигурации может быть полностью исключено влияние посторонних потоков на результаты измерений. Такой вариант системы отлично подходит для контроллеров, в которых не задействованы или задействованы лишь основные возможности Win32 и основная задача – это обработка сигналов на низком аппаратном уровне.

Схема измерения

В общем случае схема измерения задержек содержит генератор

Рис. 4. Контакты платы, задействованные при измерении задержек

импульсов, 2-канальный осциллограф и исследуемую платформу [Статья в MSDN «Benchmarking Real-time Determinism in Microsoft Windows CE» (http://msdn.microsoft.com/en-us/library/ms836535.aspx)]. На вход исследуемой платформы и один из каналов осциллографа подаются импульсы с генератора. При возникновении импульса на входе платформы генерируется прерывание, а результатом его обработки является изменение уровня сигнала на выходе, который подключен ко второму каналу осциллографа. Таким образом, задержка при обработке прерывания может быть измерена как разница между моментами времени возникновения импульса генератора и изменения сигнала на выходе платформы.

В качестве генератора импульсов воспользуемся одним из таймеров (всего в платформе доступно 12, из них 3 таймера позволяют скоммутировать выходной сигнал на контактную площадку для подключения плат расширения). Далее сигнал таймера подадим на вход порта ввода/вывода общего назначения (GPIO), настроив его на генерацию прерываний по приходу импульса. Для формирования выходного сигнала воспользуемся вторым портом ввода/вывода. Чтобы измерить обе задержки, ISR и IST, будем устанавливать высокий уровень выходного сигнала при входе в ISR и низкий – при запуске IST. Таким образом, фронт выходного импульса покажет ISR-задержку, а его длительность – IST-задержку. На рис. 4 показаны контакты платы, задействованные для реализации описанной схемы измерения. Схема подключения приведена на рис. 5.

Анализ результатов

Перейдем к анализу результатов. Сначала для случая, когда в системе запущен всего один поток. На рис. 6 приведен снимок экрана осциллографа, где желтым цветом (1) показан импульс на входе, а голубым (2) – на выходе платформы.

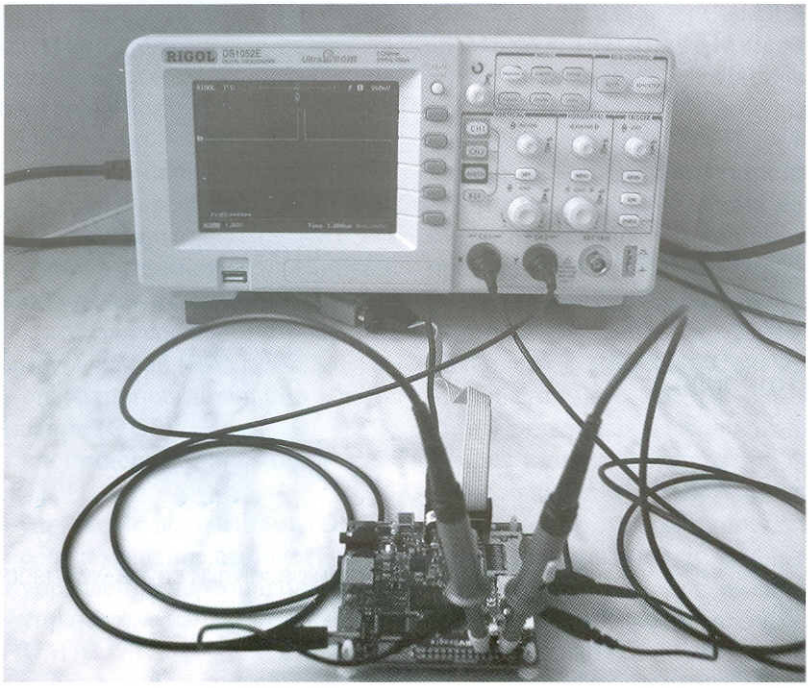

Рис. 5. Схема подключения

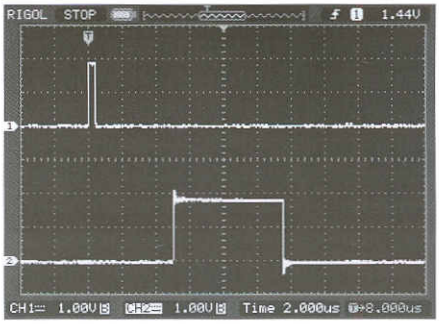

Рис. 6. Осциллограмма для однопоточной задачи

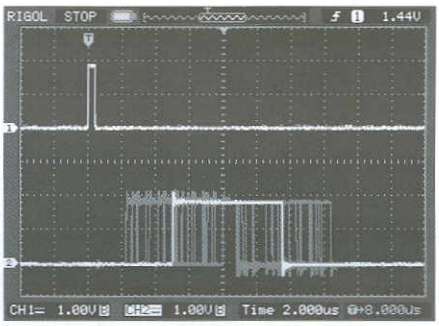

Рис. 7. Осциллограмма для однопоточной задачи с историей показаний

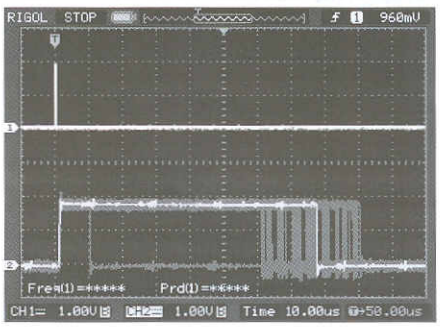

Рис. 8. Осциллограмма для многопоточной задачи с историей показаний

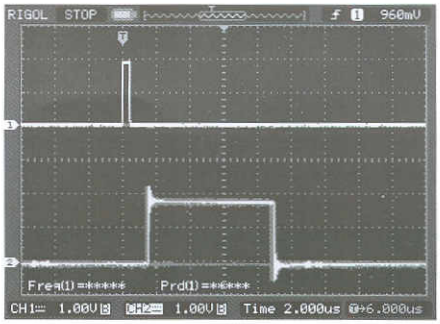

Рис. 9. Осциллограмма для многопоточной задачи с историей показаний после доработки BSP

В соответствии с полученными результатами тестирования аппаратная задержка – задержка до запуска ISR – составляет в среднем 5 мкс. IST-задержка – 6,5 мкс. Если же посмотреть историю показаний (рис. 7), то видно, что ISR-задержка менялась в интервале 2–8 мкс, а

IST оставалась практически неизменной. Постоянное значение IST – идеальный случай, когда отсутствует влияние других потоков системы. В системах с интенсивным переключением потоков значение IST-задержки, как правило, подвержено флуктуации.

Теперь рассмотрим случай, когда в системе запущено несколько потоков. При этом поток обработки прерываний имеет более высокий приоритет. Результаты второго эксперимента, включая историю показаний, приведены на рис. 8. Масштаб был увеличен, с тем чтобы выходной импульс полностью поместился на экране осциллографа. Значение IST-задержки изменило свой постоянный характер и увеличилось в среднем до 75 мкс. Также, судя по истории показаний, иногда задержка совпадает с результатом первого эксперимента. Очевидно, что минимальная задержка соответствует случаю, когда не происходит переключения контекста исполнения. А поскольку основное время поток обработки прерывания простаивает, ожидая установки события и позволяя исполняться другим потокам, то происходит это редко.

Более подробное изучение системы с использованием профилировщика ядра показало, что причиной такого значительного увеличения задержки стала реализация одной из функций BSP, а именно то, как был реализован переменный тик системного таймера. После модификации BSP, с тем чтобы использовался постоянный тик системного таймера, который по умолчанию составляет 1 мс, картина задержек полностью нормализовалась (рис. 9). Отклонения величин обоих задержек составили доли микросекунды. Значение ISR-задержки сократилось до 1,5 мкс, а IST-задержка практически достигла результата для однопоточной задачи, составив 7,5 мкс.

Обобщая полученные результаты, для приведенной конфигурации можно сделать следующие выводы. Во-первых, согласно значениям задержек по обработке прерываний, система полностью удовлетворяет требованиям для построения систем жесткого реального времени. Второй вывод: для достижения оптимальных результатов необходимо модифицировать BSP: либо отказавшись от реализации переменного тика таймера, либо выполнив ее доработку. Сравнивая с результатами независимых исследований [Отчет по результатам независимого тестирования Windows Embedded CE (http://www.microsoft.com/windowsembedded/en-us/products/windowsce/technical-specifications.mspx)], в данном случае были получены достаточно хорошие показатели. Например, для платформы на базе процессора Intel Pentium 200 МГц максимальное значение IST-задержки составило 20 мкс.

П.В. Белевский, системный инженер,
ООО «Кварта Технологии», г. Москва,
тел.: (495) 234-40-18,
e-mail: info@quarta.ru, www.quarta.ru