Среда разработки программного обеспечения для встраиваемых систем на основе JME

Сафронов А.Ю., Намиот Д.Е.

***Аннотация*—В данной работе производится обзор средств разработки программного обеспечения для встраиваемых систем, рассматривается проблематика современных подходов к разработке ПО для встраиваемых систем, а так же рассматривается предлагаемая к использованию система разработки программного обеспечения на базе JME**

***Ключевые слова*— среда разработки, встраиваемые системы, embedded, IDE, embedded Java, JME,Embedded.**

I. Введение

Встраиваемая система (embedded system) – это компьютерная система, спроектированная для специальных функций управления внутри большой системы, часто с жесткими условиями по временной задержке реакции на события. По сравнению со встраиваемыми системами, персональный компьютер спроектирован для решения широкого круга задач пользователей. Встраиваемая система проектируется для решения более узкого круга задач.

Встраиваемые системы управляют большим количеством устройств в сегодняшнем мире.

Сферы использования варьируются от портативных устройств, такие как цифровые часы и mp3 плееры, автомобильные системы, бытовая техника и т.д., до больших стационарных инсталляций, таких как контроллеры для управления производственным процессом на заводах и систем, контролирующих ядерные электростанции.

Встраиваемые системы содержат процессорные ядра, которые в большинстве представляют собой микроконтроллер или цифровой сигнальный процессор (DSP).

Классы ядер, используемых во ВС очень широк, начиная от 8,16-битных микроконтроллеров и заканчивая 32 битными микроконтроллерами типа ARM, MIPS, PowerPC с частотой десятки и сотни мегагерц, внешней RAM и FLASH памятью объемами в десятки и сотни мегабайт.

Сафронов Андрей Юрьевич – студент второго курса магистратуры факультета вычислительной математики и кибернетики Московского Государственного Университета им. Ломоносова.

Намиот Дмитрий Евгеньевич – кандидат физико-математических наук, старший научный сотрудник лаборатории открытых информационных технологий факультета вычислительной математики и кибернетики Московского Государственного Университета им. Ломоносова.

Так как встраиваемые системы предназначены для решения специализированных задач, то инженеры имеют возможность оптимизировать дизайн систем таким образом, чтобы снизить размер и стоимость продукта и увеличить надежность и производительность. С этой целью очень широко применяются системы на кристалле (*System-on-a-Chip*) при проектировании устройств.

Типичная система на кристалле содержит в себе процессорное ядро, банк памяти RAM и FLASH, а так же различные периферийные устройства(аналоговые преобразователи, внешние интерфейсы USB, Ethernet, SPI). За счет высокой интегрированности устройств различного назначения, получается значительная экономия в размерах, энергопотреблении, а также стоимости проектирования конечных устройств. Благодаря этим характеристикам SoC пользуются большой популярностью, например, одна только компания MICROCHIP к 2011 году поставила заказчикам 10 миллиардную микросхему SoC семейства PIC, это больше чем число процессоров в персональных компьютерах.

II. Обзор средств разработки программного обеспечения для встраиваемых систем.

Не смотря на широкое распространение встраиваемых систем с ограниченными ресурсами, нельзя назвать широко распространенных средств разработки. Наиболее используемая практика – программирование на языке C,C++ и assembler.

Многие компании-разработчики микропроцессоров предлагают пользователям свои средства разработки.

Компания Microchip предлагает для своих продуктов среду разработки MPLAB IDE[7], включающую средства компиляции на базе gcc и средства отладки, графические, сетевые библиотеки и библиотеки вывода. Весь это инструментарий можно использовать в основном только с продуктами компании Microchip.

Компания Atmel производит очень популярные семейства систем на кристалле C51, AVR и так же предлагает свою среду разработки AVR Studio[9]

Компания Texas Instruments поддерживает свой инструментарий — Code Composer Studio[10].

Такой подход к выпуску инструментов разработки программного обеспечения для встраиваемых систем со стороны компаний-производителей систем на кристалле имеет свои минусы. Зачастую компании-разработчики параллельно разрабатывают программные библиотеки

для своих продуктов схожие по функциональности, но ограниченные по применению в рамках свое продукции. При таком подходе появляются определенные трудности, связанные с разработкой качественных графических библиотек, например.

Некоторые компании разработчики ПО, предлагают свои решения в области тех же графических библиотек, например компания Amulet Technologies является одним из известных разработчиков высокопроизводительных GUI решений для встраиваемых систем. Компания поставляет свою систему разработки GEMstudio GUI Design Software[11].

В мире десктоп-систем и мобильных систем с достаточным количеством ресурсов существуют распространенные способы решения таких проблем, например разработчики для создания кроссплатформенного программного обеспечения могут использовать Qt библиотеки или платформу разработки Java. К сожалению, такие средства разработки не применимы к большинству систем на кристалле в силу ограниченности вычислительных ресурсов. В системах на кристалле зачастую отсутствует ОС и размер оперативной памяти измеряется 10кб.

Кроме разработки программного инструментария для встраиваемых систем с нуля, существует подход к разработке инструментария, основанный на адаптации программного обеспечения для десктоп-систем и мобильных систем с большим количеством вычислительных ресурсов. То есть, на инструментарий для разработки программного обеспечения, накладываются определенные ограничения, и этот инструментарий адаптируется к области встраиваемых систем.

Можно выделить для рассмотрения две технологии разработки программного обеспечения для систем уровня десктоп, которые в настоящее время адаптируются для встраиваемых систем. Это платформа .NET и платформа JEE/JME

Такая крупная компания как Microsoft не обошла своим вниманием рынок встраиваемых систем и представила свое решение для встраиваемых систем — адаптированную версию платформы .NET[1].

A. Платформа разработки .NET Micro FrameWork.

Для разработки встраиваемых систем, преимущества .NET Micro FrameWork, могут быть обобщены следующим образом:

- низкий уровень стоимости разработки, чем у традиционных встраиваемых платформ;
- более быстрый выход готового продукта на рынок;
- более низкая стоимость устройств managed platforms;
- меньший размер устройств managed platforms;
- более низкое энергопотребление managed platforms;
- нет ограничений по специфическому чипсету;
- составляет важную часть “Microsoft embedded strategy”.

Следующие возможности платформы .NET Micro FrameWork обеспечивают снижение стоимости разработки и более быстрый выход готового продукта на рынок, чем у традиционных встраиваемых платформ:

1) управляемый код и все его преимущества;

2) программирование на современном языке с помощью Visual C#;

3) широкий выбор библиотеки базовых классов (подмножество обычной платформы .NET);

4) возможность низкоуровневой работы с устройствами с помощью драйверов;

5) интеграция с Visual Studio;

6) большое сообщество .NET разработчиков;

7) возможность прототипирования и отладки с помощью эмулятора.

.NET MFW позволяет разрабатывать приложения для встраиваемых систем с использованием языка C# и Visual Studio.

При использовании .NET MFW не требуется наличие операционной системы во встраиваемых устройствах, MFW может выполняться низкоуровневым способом, т. к. MFW содержит сервисы, обычно предоставляемые операционной системой:

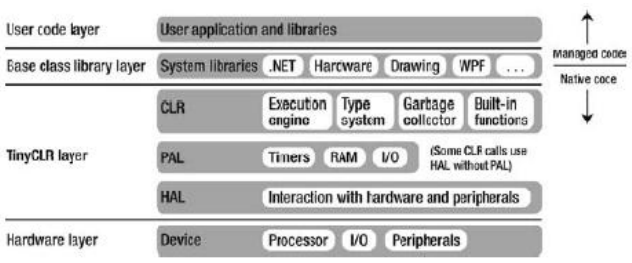
1) загрузочный код;

2) исполнение кода.

Рассмотрим требования к вычислительным ресурсам у платформы .NET Micro FrameWork.

В последних версиях платформы, разработчикам удалось снизить требования к минимальной конфигурации встраиваемой системы до 256kb Flash памяти и 64kb оперативной памяти, разрядность процессора — 32 бит. Для исполнения кода, .NET Micro FrameWork использует интерпретатор. Указанные минимальные характеристики в виде разрядности процессора и интерпретатора ограничивают сферу применения .NET Micro FrameWork.

Ниже приведена иллюстрация архитектуры .NET Micro FrameWork из работы [1].



К платформе JEE/JME также наблюдается интерес со стороны производителей средств разработки для встраиваемых систем.

B. Виртуальная машина KVM

Компания Sun/Oracle предлагает свою технологию, для разработки ПО для встраиваемых систем. Предлагаемая технология, основана на платформе J2ME и виртуальной машине KVM [2].

Виртуальная машина K (KVM), являющаяся ключевым элементом архитектуры J2ME и представляет собой переносимую виртуальную машину, спроектированную заново для устройств с маленькой памятью, ограниченными ресурсами, и которые подключены к сетям, такие как сотовые телефоны и т.д. Эти устройства обычно включают в себя 16-битный или 32-битный процессор и с минимум оперативной памяти в районе 128 килобайт. Однако, KVM может быть гибко развернута на широком классе устройств для различных применений и с широким диапазоном вариаций процессоров, размера памяти, характеристик устройств.

Общая идея дизайна KVM была в создании наименьшей возможной «полной» Java виртуальной машины, которая будет поддерживать все главные аспекты языка программирования Java, но может быть запущена в устройстве с ограниченными ресурсами с несколькими сотнями kB оперативной памяти.

Более подробно, KVM должна была удовлетворять требованиям:

1) занимать мало места, от 40 до 80 kB;

2) быть легко переносимой;

3) быть модульной и конфигурируемой;

4) быть «полной» и «быстрой» насколько это возможно.

Платформа J2ME вместе с KVM подходит для встраиваемых систем с объемом RAM больше 128к и достаточно высоким быстродействием CPU, поскольку Java байт-код интерпретируется виртуальной машиной. Но для большинства систем на кристалле, виртуальная машина KVM все же довольно велика по объему занимаемой памяти и недостаточно эффективна по быстродействию вследствие интерпретации.

C. Excelsior JET Embedded.

Компания Excelsior предлагает программное решение Excelsior JET Embedded[5], для использования Java во встраиваемых системах. Вместо медленной интерпретирующей байт-код виртуальной VM, Excelsior JET включает в себя оптимизирующий Ahead-Of-Time (AOT) компилятор. Java разработчики могут использовать его для трансформации jar файлов приложения в оптимизированный исполняемый бинарный код целевой архитектуры. В результате, Java приложения исполняются на высокой скорости с момента запуска приложения. Компилятор AOT использует jar и class файлы на входе и создает исполняемый код на выходе, runtime так же включает в себя JIT компилятор для обработки классов, которые не были предварительно скомпилированы AOT.

Все конечно выглядит хорошо, тем не менее, для вычислительных мощностей embedded системы предъявляются достаточно высокие требования по ресурсам и быстродействию.

Требования к системе:

CPU	Intel Pentium II, 200 MHz и выше
RAM	5 MB минимум, рекомендуется 16+MB
ROM/FLASH	Минимум 15 MB
OS	Windows Linux: kernel 2.6.x, glibc 2.3+, NPTL 2.3+

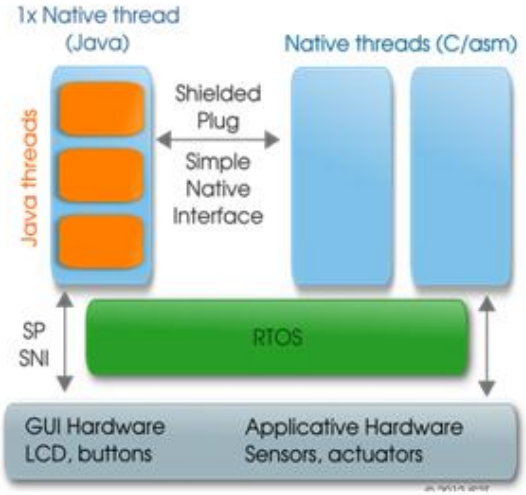
D. Java платформа MicroEJ.

Компания IS2T предлагает свой подход к использованию Java для разработки приложений для встраиваемых систем [15].

Компанией разработана своя версия Java машины с учетом вычислительных возможностей встраиваемых систем. Виртуальная машина занимает в минимальном виде 28кб Flash памяти и 1кб оперативной памяти.

Есть варианты использования разработанной виртуальной машины как в составе с RTOS так и без нее.

Ниже представлен рисунок из статьи [26], показывающий интеграцию разработанной JVM с RTOS.



Компания IS2T также предлагает свою реализацию библиотек, в том числе графических.

В данной платформе предусмотрен JNI интерфейс для написания низкоуровневых приложений.

В целом решение на базе Java для разработки программного обеспечения для встраиваемых систем от компании IS2T можно назвать интересным и заслуживающим внимание. Такой подход приносит

некоторое новаторство в область встраиваемых систем. Компания за свой продукт также была отмечена различными наградами. Так, на конференции в Париже в 2012 году «5th Assises de l'Embarqué Conference», компания получила приз в категории «Embedded Technologies», что говорит об актуальности такого подхода.

Вместе с тем, данное решение обладает определенными недостатками. Прежде всего это наличие виртуальной машины, т. е. Java байт-код интерпретируется и выполняется существенно медленнее того же кода написанного на языке C.

Второй недостаток, это ограниченный набор платформ, для которых сделана реализация виртуальной машины, в данный момент реализованы только версии для ARM и AVR32.

E. Исследовательские работы в области Java для встраиваемых систем

В академических кругах также наблюдается интерес к тематике использования Java для встраиваемых систем. Можно отметить два проекта в этой области.

Первый проект – это виртуальная машина Java для встраиваемых систем HVM (Hardware near Virtual Machine) [15]. В разработке данной среды принимают участие сотрудники VIA University College и Aalborg University. С результатами исследований по данному проекту можно ознакомиться в статьях [17][18][22]

Среда разработки HVM интегрирована с существующими методами разработки, использующие язык C. Для работы HVM в минимальной конфигурации требуется 20kb ROM и 500 байт RAM. Также проведена интеграция HVM с IDE Eclipse,

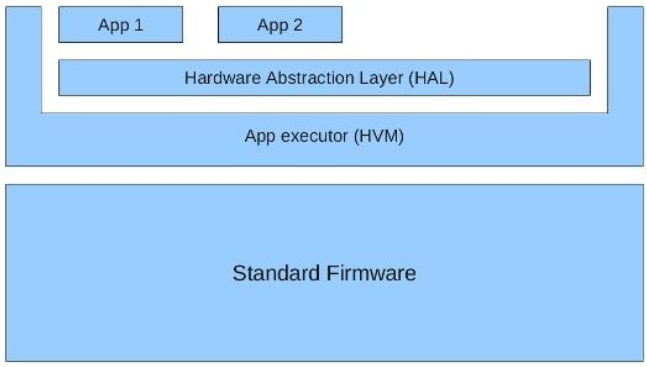
сделана поддержка загрузки и обновлений Java приложений.

Ключевые особенности HVM:

- вместе с приложением компонуются только те библиотеки, которые могут быть доступны во время выполнения;
- независимость от SDK. HVM работает со стандартными библиотеками Java из SDK Sun, так же могут быть использованы библиотеки из SDK других производителей;
- не требует для работы операционной системы;
- простая процедура сборки приложений;
- HVM поддерживает работу с низкоуровневыми (аппаратными) объектами и обработку прерываний низкого уровня;
- минимизация использования памяти RAM и ROM. Интерпретатор занимает 30kb ROM и размер Java кучи по умолчанию 4kb;
- комбинированный способ выполнения. Некоторое подмножество методов может быть скомпилировано с помощью Ahead-of-Time компилятора. Остальные методы интерпретируются.

Компилируемое множество методов задается пользователем.

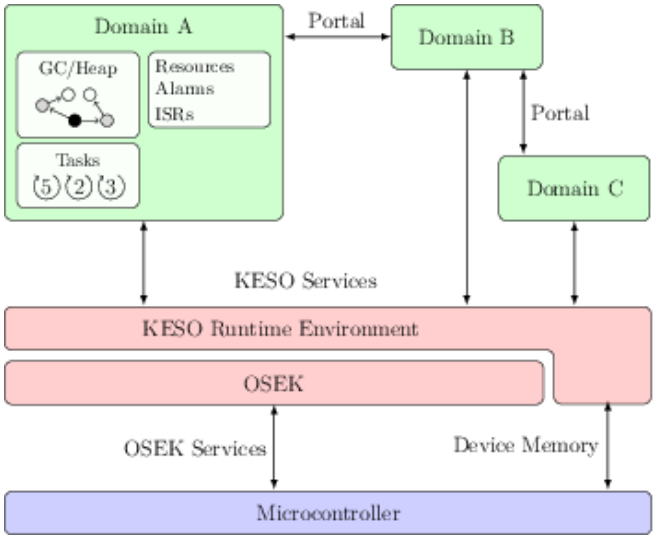
Ниже приведена общая схема архитектуры с сайта проекта[15].



Исходные файлы HVM свободно распространяются и могут быть использованы для решения некоторых задач, но в основном эти задачи имеют учебный характер.

Второй исследовательский проект в области использования Java, который хочется отметить – это среда разработки KESO. Данная среда разрабатывается в Университете имени Фридриха-Александра в Эрлангене и Нюрнберге [3][4],[19].

Ниже приведен рисунок из [4].



Вот как описывают KESO на сайте ее разработчиков[5].

Система KESO предоставляет возможность запуска одновременно несколько JVM (Multi-JVM) на одном устройстве, для каждой задачи предназначена своя виртуальная машина.

Данная Multi-JVM построена на OSEK/VDX или AUTOSAR OS операционных системах, которые позволяют реализовать механизмы планирования, синхронизации для поддержки многозадачности на одном микроконтроллере.

Система KESO разработана для статических встраиваемых систем. Для таких систем есть возможность применять методы анализа, с помощью

которых можно создать систему высоко-адаптированную под приложения Java. Т.к. не требуется динамическая загрузка классов Java в такой системе, то размер результирующего кода может быть очень небольшим. Вместо интерпретации байт-кода на микроконтроллере, Java байт-код компилируется в код целевой платформы.

Приложения KESO разрабатываются на Java и используют унифицированную модель программирования.

Система KESO предоставляет концепцию, похожую на концепцию процессов в современных операционных системах для персональных компьютеров. Вместо процессов в KESO есть домены, которые позволяют безопасно сосуществовать множеству задач на одном и том же микроконтроллере. Домены взаимодействуют с помощью унифицированного механизма, работающего похожим образом как Java Remote Method Invocation (RMI) или Remote Procedure Calls (RPC).

Система KESO позволяет делать программную репликацию критически важных приложений. Копии изолированы друг от друга с помощью доменов. При сбое работы приложения, вместо него может быть использована реплика.

KESO имеет экспериментальную поддержку сети, которая позволяют размещать домены на разных сетевых узлах. Эти домены имеют возможность взаимодействовать, как и домены на одном микроконтроллере.

Уникальные особенности KESO, позиционируемые ее разработчиками:

- первая Multi-JVM для встраиваемых систем;
- предоставляются OSEK/VDX API вызовы и системные примитивы (Java) для разработчика, включая защиту сервисов не предоставляемую OSEK/VDX операционными системами;
- есть возможность разрабатывать драйверы аппаратуры на чистом языке Java;
- предоставляется настраиваемое для каждого домена управление кучей (сборка мусора).

Сравнение двух технологий разработки на базе Java, HVM и KESO можно найти в статье [20]

Две системы разработки программного обеспечения для встраиваемых систем, HVM и KESO, постоянно развиваются и совершенствуются, но все же далеки пока от применимости на практике и по большей части являются полигоном для исследований в образовательных целях. Общая проблема у этих систем — отсутствие единой удобной среды разработки и некоторые недостатки реализации Java вроде использования интерпретаторов и необходимости использования операционных систем.

Существуют еще другие проекты JVM для встраиваемых систем, например JamVM[24] JamaicaVM[25], Cacao JIT[23], а также Ahead-of-Time компиляторы Java → C вроде FijiVM[26]. Недостаток всех этих систем в том, что в основном это исследовательские проекты, которые не могут считаться в полной мере средами разработки.

III. Проблематика современных подходов к разработке программного обеспечения для встраиваемых систем

Весьма актуальной проблемой, связанной с разработкой ВС, является проблема выбора технологий и средств разработки программного обеспечения для встраиваемых систем.

Для класса систем, в которых применяются производительные процессорные ядра типа ARM, MIPS, с внешней памятью с размером больше мегабайта, используются операционные системы Linux, Windows Embedded и т. п. И для таких ВС могут применяться те же самые средства разработки, которые применяются для desktop систем.

Однако, для класса систем с ограниченными ресурсами(надо сказать, что это самый многочисленный класс систем), использующих систему на кристалле в качестве вычислительного ядра существуют некоторые трудности в выборе технологии разработки.

Как было показано выше, каждая компания-разработчик микропроцессорных устройств, такая как Microchip, Texas Instruments, Atmel, предлагает свою среду IDE для разработки, зачастую свой компилятор C/C++ или версию компилятора gcc, свои библиотеки и т.д. Нет общего удобного стандартизованного подхода (или подходов) к разработке ПО, также есть сложности с переносимостью программного обеспечения с одной платформы на другую вследствие несовместимости интерфейсов библиотек и т.п.

Вследствие различия инструментария разработки для различных систем на кристалле, возникают определенные трудности при смене платформы разработки.

При переходе на новую платформу разработки, что случается достаточно часто в мире разработки hardware, компании-разработчики hardware тратят ресурсы на обучение персонала новой среде разработки или тратят время и силы на поиск сотрудников с уникальной специализацией. Так же возникают трудности с переносом программного обеспечения на другую платформу.

Следует отметить, что с появлением свободных средств IDE, таких как Eclipse и NetBeans, ряд компаний-производителей процессоров, а так же просто разработчиков ПО стали предлагать свои решения для разработки программного обеспечения для встраиваемых систем на основе этих IDE. Такой подход сейчас широко популяризируется, большое количество разработчиков могут с легкостью осваивать такие среды разработки.

Системы разработки, основанные на .NET и Java интересны в плане разработки программного обеспечения для встраиваемых систем, но те доступные

средства, которые сейчас присутствуют на рынке, не выдерживают конкуренции с традиционными средствами разработки на языке C/C++ в плане вычислительных ресурсов и быстродействия.

Учитывая актуальность проблемы выбора среды разработки для встраиваемых систем, а также современные тенденции в этой области, была создана новая среда разработки на базе JM

IV. Среда разработки для встраиваемых систем на базе JME

В мире мобильных устройств, с большими по сравнению с системами на кристалле ресурсами, широко распространена платформа JME. Не последнюю роль в широком распространении сыграла удобная среда разработки, включающая библиотеки на все случаи жизни, а так же популяризация языка Java, изучению которого уделяется внимание практически во всех университетах мира.

Изначально Java платформа разрабатывалась для использования во встраиваемых системах, даже совсем простых бытовых устройствах как утюги и холодильники, но потом использование Java платформы стало повсеместным в десктоп системах и мобильных устройствах, а встраиваемые системы оказались забыты. Основную роль здесь сыграла сложность реализации JVM для встраиваемых систем. Чистая интерпретация слишком расточительно расходует вычислительные ресурсы у систем на кристалле и пагубно сказывается на энергосбережении.

Подход к использованию Java для встраиваемых систем, который основан на применении статической компиляции к платформе J2ME, не оправдывает себя для маленьких систем, поскольку требует наличия операционной системы, а это слишком расточительно для систем на кристалле.

С учетом сформулированной актуальности задачи о создании удобной среды разработки для различных семейств и классов систем на кристалле, было проведено исследование и создана архитектура среды разработки, которая основана на платформе JME для встраиваемых систем с ограниченными ресурсами. Созданную систему разработки, по-сути можно было бы именовать Java embedded, такое название характеризует применения данной системы.

Использование системы, основанной на платформе JME, позволяет внести в мир систем на кристалле стандартный подход к разработке программного обеспечения для различных семейств микроконтроллеров.

При использовании подхода на основе JME, значительно облегчается создание программного обеспечения, в силу того, что в мире существует уже развитая культура программирования с использованием Java. Также существует большое количество разработчиков использующих этот язык и для таких разработчиков не составит труда разрабатывать программного обеспечения для встраиваемых систем без изучения новых сред разработки, особенностей библиотек, GUI и т. д.

В области разработки GUI, становится возможным применение всего множества библиотек, созданных за последние 10 лет.

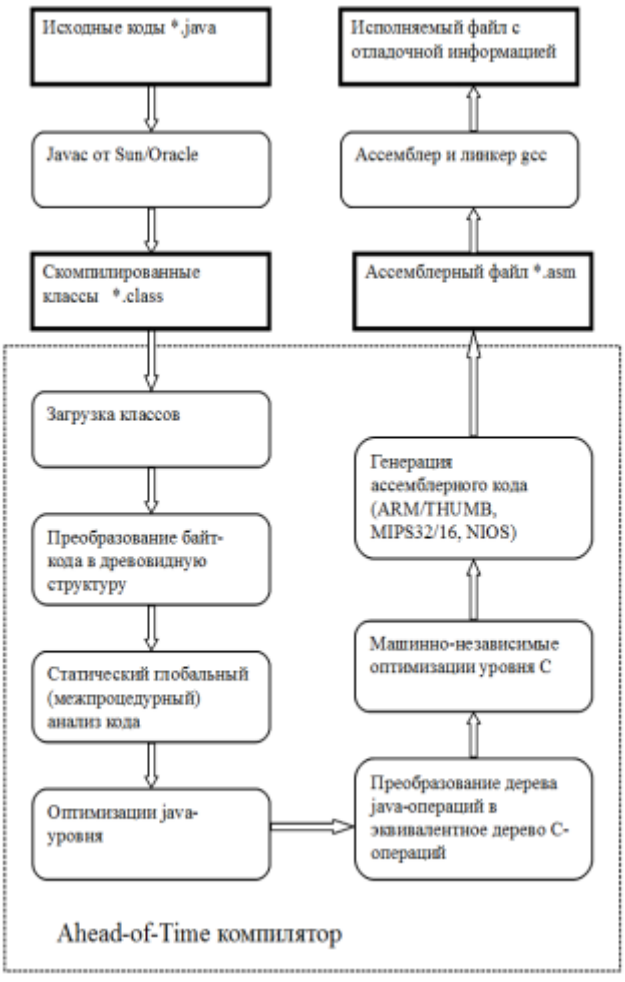
Следует отметить, что при проектировании среды на основе JME был учтено набирающие силу направление в программировании встраиваемых систем. А именно использование свободных IDE, таких как Eclipse и NetBeans. В данном случае была использована NetBeans, как предлагаемая основная среда разработки.

Рассмотрим предлагаемую технологию разработки программного обеспечения для встраиваемых систем с ограниченными вычислительными ресурсами, на основе технологии JME.

В составе среды разработки входит Java Ahead-of-Time компилятор, отладчик, симулятор и библиотеки.

По разработке Java Ahead-of-Time компилятора уже проводились исследования, в частности об этом можно почитать в работе [21].

На рисунке приводится схема работы Ahead-of-Time компилятора, используемого в созданной среде разработки.



Минимальные требования у предложенной технология разработки программного обеспечения:

1) объем памяти RAM 10kb;
2) разрядность процессора 8/16/32 бит.

Минимальным требованиям системы разработки удовлетворяют большинство систем на кристалле.

V. Сравнение среды разработки на основе JME с уже существующими средами разработки для встраиваемых систем.

Рассмотрим сравнение характеристик существующих распространенных систем IDE с предлагаемой средой разработки по ключевым характеристикам

Общие характеристики:

IDE	Среда разработки	Linux	Windows	C/C++	Java	C#
Embedded JME	На основе NetBeans	Да	Да	Да	Да	Нет
AVR Studio	На основе Visual Studio с версии 5.0	Нет	Да	Да	Нет	Нет
MPLAB X	На основе NetBeans с версии 2011	Да	Да	Да	Нет	Нет
.NET Micro NETMF	На основе Visual Studio	Нет	Да	Нет	Нет	Да
Freescale Code Warrior	На основе Eclipse	Да	Да	Да	Нет	Нет
Texas Instruments Code Composer	На основе Eclipse	Да	Да	Да	Нет	Нет
MicroEJ	На основе Eclipse	Да	Да	Нет	Да	Нет

IDE	Интерпретация	Библиотеки	Легкость освоения
Embedded JME	Нет	J2ME	Высокая
AVR Studio	Нет	Собственные	Средняя
MPLAB X	Нет	Собственные	Средняя
.NET Micro NETMF	Да	.NET	Высокая
Freescale Code Warrior	Нет	Собственные и gcc	Высокая
Texas Instruments Code Composer	Нет	Собственные и gcc	Высокая
MicroEJ	Да	Собственные и gcc	Средняя

Специальные характеристики сред разработки применительно к классу систем на кристалле:

IDE	Минимальный объем RAM/FLASH	Разрядность CPU целевой платформы	Быстродействие
Embedded JME	~1-100 kb RAM 10kb FLASH	8/16/32	Высокое
AVR Studio	~1-100 kb RAM 10kb FLASH	8/16/32	Высокое
MPLAB X	~1-100 kb RAM 10kb FLASH	8/16/32	Высокое
.NET Micro NETMF	>64kb 256kb FLASH	32	Низкое (интерпретация)
Freescale Code Warrior	~1-100 kb RAM 10kb FLASH	8/16/32	Высокое
Texas Instruments Code Composer	~1-100 kb RAM 10kb FLASH	16/32	Высокое
MicroEJ	~1-100 kb RAM 10kb FLASH	16/32	Среднее (улучшенная интерпретация)

VI. Заключение.

На основе сравнения характеристик средств разработки для встраиваемых систем, можно сделать вывод, что предлагаемая среда разработки обладает новизной

подхода к решению задач разработки ПО для систем на кристалле, а именно, использование языка программирования Java в тех задачах, где традиционно используется языки C,C++. При этом сохраняется возможность использования языков C/C++ совместно с Java. Среда разработки на основе JME удовлетворяет современным требованиям к средам разработки по легкости освоения, наличия библиотек, кросс-платформенности, что позволяет говорить об эффективности использования среды для решения задач разработки ПО. Так же проведенный анализ показывает, что предлагаемая среда по ключевым характеристикам не уступает существующим решениям, а по ряду характеристик превосходит, предлагает более гибкий подход к разработке.

Библиография

[1] "EXPERT .NET MICRO FRAMEWORK", 2009 by Jens Kühner

[2] "J2ME Building Blocks for Mobile Devices", Sun Microsystems, 2000

[3] "KESO An Open-Source Multi-JVM for Deeply Embedded Systems" Isabella Thomm, Michael Stilkerich, Christian Wawersich, Wolfgang Schröder-Preikschat Friedrich-Alexander University Erlangen-Nuremberg, Germany, 2010

[4] http://www4.cs.fau.de/Research/KESO/

[5] http://www.excelsiorjet.com/embedded/

[6] http://www.embeddedinsights.com/channels/topics/ide/

[7] http://www.microchip.com/pagehandler/en-us/family/mplabx/

[8] http://www.freescale.com/webapp/sps/site/homepage.jsp?code=CW_HOME

[9] http://www.atmel.com/tools/atmelstudio.aspx?tab=overview

[10] http://www.ti.com/tool/ccstudio

[11] http://www.amulettechnologies.com/products/gui-design-software

[12] "Java-through-C Compilation: An Enabling Technology for Java in Embedded Systems" Ankush Varma and Shuvra S. Bhattacharyya, Febraury 2004

[13] http://www.sable.mcgill.ca/publications/thesis/#michaelsMastersThesis

[14] http://www.icelab.dk/

[15] www.iar.com

[16] 'Towards a Real-Time, WCET Analysable JVM Running in 256kB of Flash Memory" Stephan Korsholm, Kasper Søe Luckow, Bent Thomsen, 2011

[17] "Hardware Objects for Java" Martin Schoeberl, Christian Thalinger, Stephan Korsholm, Anders P. Ravn, http://www.jopdesign.com/doc/hwobj.pdf

[18] "Tailor-made JVMs for statically configured embedded systems", Michael Stilkerich, Isabella Thomm, Christian Wawersich, Wolfgang Schröder-Preikschat, Concurrency and Computation: Practice & Experience, June 2012, Pages 789-812

[19] http://www.icelab.dk/evaluation.pdf

[20] "Compiling Java for Embedded Systems", Per Bothner

[21] "Java for Cost Effective Embedded Real-Time Software" Stephan E. Korsholm, Ph.D. Dissertation, August 2012

[22] "Cacao - a 64 bit javavm just-in-time compiler" A. Krall and R. Gra, 1997

[23] http://jamvm.sourceforge.net/.

[24] http://www.aicas.com/jamaica.html

[25] "Real time Java on resource-constrained platforms with Fiji vm" F. Pizlo, L. Ziarek, and J. Vitek, 2009

[26] "Build small footprint GUIs for ARM Cortex-M designs using Java" Régis Latawiec, IS2T