Информатика, вычислительная техника и управление

УДК 004.432.2

О. С. Большаков

В. Г. Шаров

Рыбинский государственный авиационный технический университет имени П. А. Соловьева

А. В. Петров

Общество с ограниченной ответственностью "Научно-производственное предприятие Системы автоматизации технологических и энергетических комплексов плюс"

РЕАЛИЗАЦИЯ КРОССПЛАТФОРМЕННОСТИ ПРОГРАММ ДЛЯ СПЕЦИАЛИЗИРОВАННЫХ РАСПРЕДЕЛЕННЫХ ВСТРАИВАЕМЫХ СИСТЕМ

O. S. Bolshakov

V. G. Sharov

Soloviev Rybinsk State Aviation Technical University (RSATU)

A. V. Petrov

NPP SATEK ltd

CROSS-PLATFORM PROGRAMS REALISATION FOR SPECIALIZED EMBEDDED DISTRIBUTED EMBEDDED SYSTEMS

Выявлены недостатки существующих подходов по обеспечению кроссплатформенности программного обеспечения для специализированных встраиваемых систем, предложен новый подход по обеспечению кроссплатформенности программ с использованием средств языка Embeddecy. Подход основан на представленной обобщенной модели микроконтроллера, представлен пример реализации модуля управления аналого-цифровым преобразователем микроконтроллера с разделением аппаратно-зависимых и аппаратно-независимых функций.

Встраиваемые системы, кроссплатформенность, микроконтроллеры, языки программирования

Disadvantages of existing approaches of cross-platform software development for specialized embedded systems are revealed, a new approach to cross-platform programs development in Embeddecy language is disclosed, the approach is based on the generalized model of a

microcontroller, an example of the implementation of the analog-to-digital converter control module of the microcontroller with division of hardware-dependent and hardware-independent functions is provided.

Embedded systems, cross-platform, microcontrollers, programming languages

Введение.

Одной из тенденций развития встраиваемых систем в настоящее время является значительное разнообразие используемых типов вычислительных устройств. При этом возрастает гетерогенность системы и необходимость взаимодействия разнородных компонентов, в связи с чем у разработчиков программ управления возникает проблема переносимости программного обеспечения между разными типами вычислителей.

Зависимость программного кода от модели вычислителя связана не только с разрядностью ядра микроконтроллера, но и его архитектурой – наличием или отсутствием тех или иных периферийных модулей, а также их особенностями у микроконтроллеров одного семейства. На практике при создании специализированных встраиваемых систем обычно не требуется полная автоматическая переносимость кода, которая в мире "больших компьютеров" достигается, например, наличием дополнительной среды исполнения и JIT-компиляцией. Это связано с тем, что специализированные встраиваемые системы, как правило, обладают ограниченным объемом ресурсов (частота работы процессора, объем оперативной памяти и памяти данных и программ) и важнейшим требованием является эффективность исполняемого кода. Следует отметить, что проблема кроссплатформенности наиболее остро встает при разработке библиотек программных модулей, поскольку на библиотеки возлагается задача по унификации программного интерфейса для доступа к аппаратным компонентам системы.

В настоящее время наиболее распространен подход, обеспечивающий, некоторую степень переносимости программ для встраиваемых систем, как правило, только в рамках одного семейства вычислителей за счет использования средств препроцессинга: макросов и директив условной компиляции [4]. Это дает эффективный код, однако программа превращается в «лоскутное одеяло»: логика программы на языке программирования разбивается на множество фрагментов, что уменьшает прозрачность программного кода и увеличивает сложность разработки. Сами по себе возможности препроцессинга не являются средством расширения языка; они позволяют представлять программный код лишь как набор символьных строк и производить в нем текстовые замены без учета структуры программного кода.

Другим подходом является использование виртуальных машин, обеспечивающих выполнение унифицированного программного кода [1]. Такой подход обеспечивает наилучшее решение по степени переносимости кода, однако предполагает использование на встраиваемом вычислителе ресурсоемкой виртуальной машины, что делает его неэффективным в условиях специализированных встраиваемых систем на базе микроконтроллеров. К примеру, реализация виртуальной java-машины NanoVM для микроконтроллеров Atmel занимает 8 Кбайт памяти программ [5], т.е. всю память большинства микроконтроллеров этого семейства

В работе [2] предлагается использовать унифицированный язык ассемблера и кросс-компилятор для разных типов систем. В этом случае программисту предлагается низкоуровневый инструмент, что также негативным образом сказывается на эффективности разработки программного обеспечения для целевых микроконтроллерных устройств.

Наиболее эффективным из известных подходов к организации кроссплатформенных программных библиотек представляется разработка на основе парадигмы обобщенного программирования. В рамках этой парадигмы создаются программные модули и функции с параметрами (параметрический полиморфизм), причем параметризация является статической, т.е. осуществляется на этапе компиляции. Данная концепция присутствует в некоторых объектно-ориентированных языках программирования, в частности, в C++ и реализована в виде механизма шаблонов классов и функций. Однако, к примеру, при объявлении на языке C++ параметра-типа у некоторого шаблона нет возможности задать ограничения на список возможных фактически передаваемых типов. Кроме того, возникают проблемы при реализации гибкого модуля управления, способного работать с некоторым конфигурируемым выводом микроконтроллера: разработчику либо приходится использовать дополнительные ресурсы для хранения соответствия номера регистров-портов их адресам, либо использовать средства препроцессинга.

В качестве специализированного языка высокоуровневой разработки управляющих программ для микроконтроллерных систем был разработан язык Embeddecy [3], который позволяет обеспечить явное выделение платформенно-зависимых программных функций и упрощает перенос программ с одной микроконтроллерной архитектуры на другую.

1. Обобщенная модель микроконтроллера.

Переносимость программного кода между разными вычислителями обеспечивается в Embeddecy абстрагированием от деталей работы этих вычислителей и выделением общих признаков. Поэтому одной из первоочередных задач при решении проблемы кроссплатформенности является создание обобщенной модели микроконтроллера.

На рисунке 1 представлена обобщенная модель специализированных микроконтроллеров наиболее популярных семейств AVR, PIC18 и STM8. В модели представлена обобщенная архитектура микроконтроллеров указанных семейств, в которую вошли основные модули, в том или ином составе присутствующие в различных моделях микроконтроллеров рассматриваемых семейств. При этом к модулям управления интерфейсами относятся UART, SPI, I2C, 1-Wire, USB, Bluetooth, CAN, Ethernet; к модулю общих настроек микроконтроллера относится, например, сторожевой таймер, управление частотой микроконтроллера, событийная система событий и прочие функции, отражающиеся на работе всего чипа.

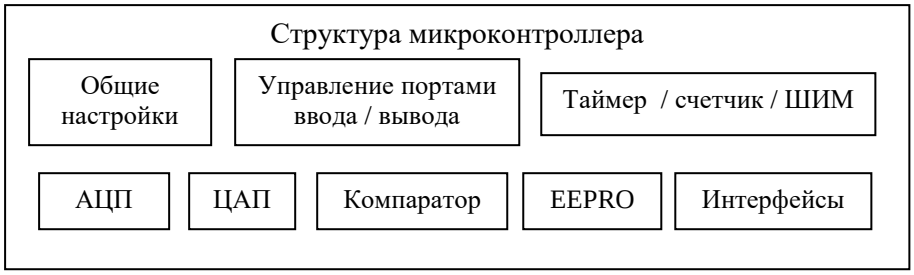

Рисунок 1 – Обобщенная структура микроконтроллера

Представленные на рисунке 1 унифицированные модули системы управления могут быть разбиты на аппаратно-зависимые и аппаратно-независимые, при этом аппаратно-независимые располагаются на более высоких уровнях абстракции. На рисунке 2 представлена схема распределения программных модулей системы управления по уровням.

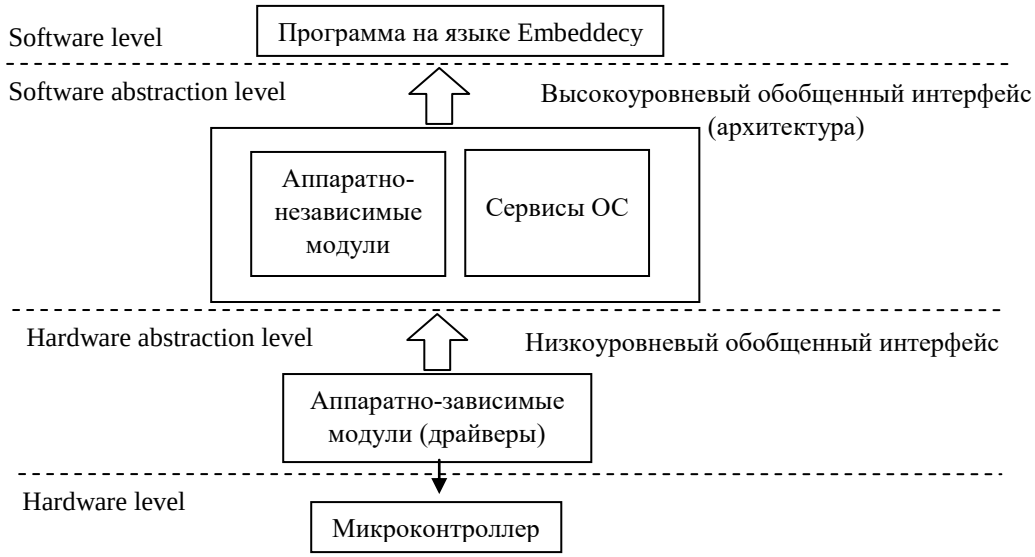

Рисунок 2 - Уровни аппаратно-зависимых и аппаратно-независимых программных модулей системы управления

Программные модули в соответствии со своим назначением располагаются на двух уровнях – уровне программных абстракций (SAL) и уровне аппаратных абстракций (HAL). Модули HAL-уровня реализуют аппаратно-зависимый код, непосредственно

взаимодействующий с аппаратными модулями микроконтроллера. Работа с модулями этого уровня производится либо через обобщенный интерфейс (и такие команды являются переносимыми), либо используются специфические функции конкретного модуля (такие команды являются непереносимыми).

Уровень SAL включает в себя аппаратно-независимые модули: алгоритмические модули или модули, эмулирующие работу недостающих аппаратных модулей, например, виртуальный модуль управления интерфейсом USB или I2C, а также сервисы операционной системы. Аппаратно-независимые модули могут взаимодействовать между собой либо с аппаратно-зависимыми модулями по обобщенному интерфейсу.

С точки зрения кроссплатформенности, наибольший интерес представляет взаимодействие модулей HAL и SAL уровней, которое специфицируется с помощью низкоуровневого обобщенного интерфейса. Механизм создания низкоуровневого обобщенного интерфейса реализован в языке Embeddecy.

2. Язык программирования Embeddecy.

Язык программирования Embeddecy можно считать расширением языка С при разработке распределенных специализированных встраиваемых систем. Основным мотивом создания нового языка был неудовлетворительный баланс между эффективностью и высокоуровневостью существующих языков программирования, используемых при разработке встраиваемых систем. Анализ типовых задач в данной области, а также появление альтернативных инструментов, подтвердили актуальность разработки языка программирования, сочетающего в себе эффективность кода на языке С и высокоуровневые концепции языков C# и Java. Одним из успешных примеров сочетания высокуровневости и эффективности можно считать язык Ada, который, однако, в практике разработки микроконтроллерных систем используется редко, в частности, из-за ограничений в средствах синхронизации программных процессов и чрезмерно строгой типизации.

В ходе анализа требований была разработана многоуровневая модель представления программного обеспечения для распределенных встраиваемых систем [3]. Язык Embeddecy включает в себя поддержку всех элементов модели (задачи, пакеты, шаблоны), а также следующие конструкции, дополняющие язык С: конструкции для обеспечения параллельного программирования (отправка синхронная / асинхронная, принятие сообщения), пространства имен, анонимные функции, типы делегатов и переменные-делегаты, события, развитые средства описания статических шаблонов, конструкция via для передачи сообщений между распределенными модулями, конструкции, определяющие области видимости.

Существенной особенностью языка является концепция шаблонов модулей, позволяющая разрабатывать на языке Embeddecy кроссплатформенные библиотечные модули. Шаблон модуля – это параметризированный модуль, значения параметров которого задаются на этапе времени компиляции. Допускается использование параметров пяти видов: параметр-макрос, параметр-вывод (пин микросхемы), параметр-выражение определенного типа, параметр-тип, параметр-модуль.

При объявлении модуля задаются параметры шаблона, и код шаблона преобразуется в код инстанцируемого модуля путем подстановки фактических значений параметров вместо формальных. Например, при использовании в объявлении шаблона модуля параметра-вывода (пина) выполняется генерация конструкций для записи и считывания данных в отображаемый на вывод устройства бит памяти. В языке допускается также частичная специализации шаблонов аналогично подобному механизму языка C++. В этом случае на базе шаблона с несколькими параметрами создается другой шаблон, в котором часть формальных параметров специализируется фактическими значениями, а часть параметров (не менее одного) остается формальными.

Формат описания шаблона и его параметров приведен на рисунке 2, где *module_type* – тип модуля: package / task, *param_macro_name* – название макропараметра, *param_pin_name* – название параметра-пина, *param_expr_name* – название параметра-выражения, *expr_type_name* – название типа параметра-выражения, *param_type_name* – название параметра-типа, *param_module_name* – название параметра-модуля, *interface_name* – название интерфейса, который должен реализовывать модуль.

```
template <module_type> <name>
<
     macro <param_macro_name>,
     type <param_type_name> is <type1> | <type2> | … | <typen>,
     pin <param_pin_name>,
     value <expr_type_name> <param_expr_name>,
     module <interface_name> <param_module_name>
>
{ <...> // тело шаблона модуля  }
```

Рисунок 2 – Формат описания шаблона модуля и его параметров

3. Шаблоны интерфейсов программных библиотечных модулей для явного отделения переносимого кода от непереносимого.

Одной из целей введения в язык Embeddecy интерфейсов является разделение аппаратно-зависимого кода и аппаратно-независимого. Использование этого механизма позволяет создать системную библиотеку модулей, при использовании которой разработчик будет иметь четкое представление о переносимости или непереносимости отдельных частей программы и о механизмах переносимости.

Разработка библиотеки модулей в рамках описываемого решения строится следующим образом.

1) Для каждого выделенного типа аппаратно-зависимого модуля в модели обобщенного микроконтроллера разрабатывается унифицированный шаблон интерфейса со списком общих для всех подобных модулей функций и типов данных.

2) Разрабатываются шаблоны аппаратно-зависимых модулей под каждую модель микроконтроллера, которые реализуют обобщенный интерфейс. При этом происходит частичная спецификация шаблона обобщенного интерфейса.

3) Для работы с модулями разработчик программы управления в коде объявляет аппаратно-зависимый модуль по нужному аппаратно-зависимому шаблону. Для того, чтобы код обращения к модулю был кроссплатформенным, необходимо обращаться к нему через обобщенный интерфейс.

4) После этого разработчик может объявлять аппаратно-независимые модули и задавать в качестве параметров аппаратно-зависимые модули, работа с которыми будет вестись через обобщенный интерфейс.

Рассмотрим механизм разделения кроссплатформенного кода и некроссплатформенного на примере реализации модуля управления АЦП в микроконтроллерах AVR и STM (рисунок 3).

В данном примере на языке Embeddecy описан кроссплатформенный шаблон интерфейса `ADC_GENERAL` и шаблон аппаратно-зависимого модуля АЦП (ADC_STM). Для обеспечения переносимости программы, работающей с аппаратно-зависимыми модулями, создается псевдоним (adc_crossplatform). Все обращения к структурам данным и функциями модуля через псевдоним adc_crossplatform являются кроссплатформенными. Если же требуется использование специфичных функций аппаратуры, тогда обращаться к этим функциям необходимо через аппаратно-зависимый псевдоним (в примере - adc_hardware), т. е. через имя модуля, инстанцированного напрямую по аппаратно-зависимому шаблону. При этом обращения к некоторым функциям и структурам данных могут стать непереносимыми. Стоить заметить, что внедрение в язык Embeddecy концепции супертипов позволяет ограничить возможный тип, который можно передать как параметр шаблона. К примеру, при описании обобщенного интерфейса модуля работы с АЦП (ADC_GENERAL) указано, что в качестве типа ValueType может быть выбран либо целочисленный int, либо беззнаковый однобайтовый тип byte. Такой подход существенно упрощает использование шаблонов для разработки библиотек программных модулей.

```
template interface ADC_GENERAL // обобщенный интерфейс модуля управления АЦП
<module MCU_GeneralConfig generalConfig,
type ValueType is int | unsigned byte,
type BitsResolution_enum is ADC_AVR.BitsResolution_enum |
ADC_STM.BitsResolution_enum | ADC_PIC.BitsResolution_enum, ...> {
    void Init();
    <...>
    ValueType ConvertGetSample (Channels_enum channel)
    <...>
}
template package ADC_STM <module MCU_GeneralConfig generalConfig, type
ValueType is int | unsigned char> implements ADC_GENERAL < MCU_GeneralConfig
generalConfig, type ValueType is int | unsigned char, BitsResolution_enum =
ADC_STM.BitsResolution_enum, ...>{
      public typedef enum BitsResolution {…};
      <...>
      <...> // реализация функций ADC_GENERAL

      // ниже идут некроссплатформенные функции, которых нет в интерфейсе
      public bool getOverrunInterruptEnabled();
      <...>
}
<...>
// После этого можем инстанцировать пакет для работы с АЦП STM
// с указанием модулям
#define_module stm8l152c6cfg = STM8l152c6_GeneralConfig<...>
#define_module adc_hardware = ADC_STM <stm8l152c6cfg, unsigned char> ;
#define_module adc_crossplatform = (ADC_GENERAL < generalConfig, ValueTYpe,
BitsResolution_enum = ADC_STM.BitsResolution_enum>, ...) adc_hardware;
<...>
adc_crossplatform.Init(); // кроссплатформенная инструкция
adc_hardware.getOverrunInterruptEnabled();// некроссплатформенная инструкция
```

Рисунок 3 – Пример аппаратно-зависимого модуля управления АЦП и кроссплатфменного интерфейса на языке Embeddecy

Выводы.

В работе рассмотрены проблемы создания программного обеспечения для специализированных гетерогенных встраиваемых систем на базе микроконтроллеров. Обоснована актуальность технологий кроссплатформенной разработки программ, анализируются существующие подходы к решению проблемы переносимости программного обеспечения. Представлена обобщенная модель микроконтроллера, положенная в основу предлагаемого подхода к формированию унифицированной библиотеки программных модулей на языке Embeddecy, обеспечивающего высокий уровень кроссплатформенности. Приводится пример реализации кроссплатформенного шаблона модуля на разработанном языке Embeddecy.

Литература

1. Платунов А. В. Теоретические и методологические основы высоко-уровневого проектирования встраиваемых вычислительных систем: автореферат дис. докт. техн. наук: 05.13.12. – СПб., 2010. – 39 с.

2. Белых А. А. Унификация архитектур однокристальных микроконтроллеров и ее применение для разработки программного обеспечения встраиваемых систем: автореферат дис. канд. техн. наук: 05.13.15. – М., 2006. – 20 с.

3. Большаков О. С. Шаров В. Г. Петров А. В. Модель распределенных программ для встраиваемых систем // Вестник Рыбинского государственного авиационного технического университета имени П. А. Соловьева. – Рыбинск, 2015. – №1 (32). С.165–171.

4. Ilya Paramonov, Andrew Vasilev, Denis Laure, Nikita Kozhemyakin Preprocessor Based Approach for Cross-Platform Development with Qt Quick Components // Proceedings of the 11th Conference of Open Innovations Association FRUCT. – St-Petersburg, 2012. – URL: https://fruct.org/publications/fruct11/files/Par.pdf

5. The NanoVM - Java for the AVR. URL: http://www.harbaum.org/till/nanovm/index.shtml

Сведения об авторах (на русском и английском языках)

БОЛЬШАКОВ Олег Сергеевич / BOLSHAKOV Oleg Sergeevich	Аспирант ФГБОУ ВПО РГАТУ имени П. А. Соловьева / Postgraduate student RSATU, +7-951-284-18-26, +7(4855)55-58-54, bolsh.os@gmail.com
ШАРОВ Владимир Григорьевич / SHAROV Vladimir Grigorievich	Кандидат физико-математических наук, профессор, заведующий кафедрой МПО ЭВС, ФГБОУ ВПО РГАТУ имени П. А. Соловьева / Candidate of Physical Mathematical Sciences, professor, Head of Department MPO EVS, RSATU +7(4855)28-04-72 sharov@rsatu.ru
ПЕТРОВ Александр Викторович / PETROV Aleksandr Viktorovich	Генеральный директор, Общество с ограниченной ответственностью "Научно-производственное предприятие Системы автоматизации технологических и энергетических комплексов плюс" / General Manager, NPP SATEK ltd, +7-980-740-09-01 gmdidro@gmail.com

Россия, Ярославская область, г. Рыбинск, улица Пушкина д. 53