УДК 004.432.2

МОДЕЛЬ РАСПРЕДЕЛЕННЫХ ПРОГРАММ ДЛЯ ВСТРАИВАЕМЫХ СИСТЕМ

О. С. Большаков, А. В. Петров, В. Г. Шаров

ФГБОУ ВПО «Рыбинский государственный авиационный технический университет имени П. А. Соловьева, г. Рыбинск»

Выявлены недостатки известных моделей представления распределенных программ для встраиваемых систем и языков программирования, предложена новая модель и язык программирования. Сформулированы преимущества предложенных моделей и языка в сравнении с известными.

МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ПРОГРАММ, РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ, ВСТРАИВАЕМЫЕ СИСТЕМЫ, МИКРОКОНТРОЛЛЕРЫ, ЯЗЫК ПРОГРАММИРОВАНИЯ

Введение

Развитие вычислительной техники в последние годы обусловило ее проникновение во все сферы жизни человека. Внедрение вычислительной техники в окружающие человека объекты привело к развитию направления встраиваемых систем и сетей (embedded systems & networks). Встраиваемые системы и сети принято определять как специализированные микропроцессорные системы, непосредственно взаимодействующие с объектом управления и объединенные с ним конструктивно [1].

Согласно классификации встраиваемых систем [2], далее в статье речь пойдет о микроконтроллерных системах, т. е. системах, в основе которых лежит микроконтроллерная платформа.

В статье [3] встраиваемые системы на базе специализированных контроллеров рассматриваются как особый вид встраиваемых систем. При этом выделяется два типа микроконтроллеров по степени их специализации для решаемых задач:

1) общецелевые микроконтроллеры (и микропроцессоры в виде отдельных микросхем), используемые, например, в многофункциональных мобильных системах (смартфонах и мобильных компьютерах);

2) специализированные микроконтроллеры для генерации сигналов определенных форм и реализации определенного набора интерфейсов.

Отмечается, что для второго класса вычислителей процесс разработки распределенных программ осложнен следующими особенностями [4]:

- узкая специализация и нестандартность задач проектирования, уникальность создаваемых систем, необходимость учета множества нефункциональных ограничений;

- большое разнообразие вычислителей (гетерогенность);
- ограниченность ресурсов (память – на 5-6 порядков меньше, чем в персональных компьютерах, частота – на 3 порядка);
- многообразие интерфейсов и протоколов физического и канального уровней.

К особым нефункциональным ограничениям встраиваемых систем относятся следующие:

- ограничения по габаритам и массе аппаратуры;
- ограничения по энергопотреблению (часто используется автономное питание);
- ограничения по условиям эксплуатации;
- дополнительные требования надежности системы, прогнозируемого времени реакции на определенные события.

В связи с перечисленными особенностями специализированных микроконтроллерных систем в области разработки программного обеспечения для них сложилась такая ситуация, что разработчики не могут использовать современные технологии и системы программирования из области компьютерных систем.

Целью данной работы является анализ недостатков существующих технологий разработки программ для микроконтроллерных систем, используемых моделей и языков программирования, а также создание новой модели представления программ и специализированного языка программирования. Это позволит ускорить процесс разработки программ для микроконтроллерных систем посредством их синтеза по описанию на высоком уровне.

1 Анализ моделей, используемых для синтеза распределенных программ во встраиваемых микроконтроллерных системах

Известно множество моделей вычислительных систем и процессов. Ниже будут рассмотрены только те модели, которые позволяют решать задачу синтеза программ для указанного класса систем. Под синтезом программ будем понимать технологию автоматизированной генерации низкоуровневого кода программы по ее высокоуровневому описанию [5].

Сегодня большинство инструментов разработки программ для микроконтроллерных систем используют структурную технологию программирования и соответствующие языки программирования. Как известно, в основе структурного программирования лежит модель представления программы, основанная на иерархической структуре блоков [6].

К недостаткам такой модели программы с точки зрения разработки ПО для распределенных встраиваемых систем можно отнести:

1) отсутствие в модели средств выражения параллелизма процессов и способов взаимодействия параллельных процессов;
2) необходимость использования низкоуровневого кода обращения к аппаратуре – в логике записи и считывания битов определенных ячеек памяти.

Определенными недостатками обладают и сами языки программирования. Наиболее распространенным языком программирования является язык С, достаточно популярен и язык С++. Однако для программирования микроконтроллеров язык С++ используется в редуцированной форме – компиляторами не поддерживаются объектно-ориентированные возможности этого языка. С++ используется из-за нехватки в языке С модульности на уровне языка, позволяет использовать шаблоны, однако на нем невозможно описывать гибкие шаблоны модулей управления, которые могли бы составлять библиотеку модулей для генерации сигналов на разных выводах устройства без использования дополнительных ресурсов [7].

Известны попытки решить проблему низкоуровневости работы с аппаратурой за счет вложения функций в обобщенные функции-агрегаты, имеющие большее количество параметров, чем исходные функции, а также за счет создания генераторов кода инициализации и настройки периферии микроконтроллера [8]. Оба подхода не решают проблему комплексно: первый подход существенно усложняет построение и использование программ, лишает их прозрачности и гибкости. Второй подход упрощает лишь часть работы пользователя, никак не изменяя уровня абстракции при написании основной части программы управления микроконтроллерной системой.

В работе [9] предлагается использовать в качестве модели представления программ сети Петри и конечные автомата. Авторы используют парадигму автоматного программирования и множество графических и текстовых языков, поддерживающих эти модели, например, UML StateChart, SFC, Siemens Graph7, Esterel, SyncCharts, ДРАКОН, Рефлекс и множество других инструментов, поддерживающих данный подход. Недостатком используемой при этом модели является неявность задания в ней взаимодействия параллельных процессов. В рамках такого подхода есть возможность выделить параллельные процессы и для каждого описать поведение в виде конечного автомата, однако при попытке организовать взаимодействие между этими процессами возникает необходимость излишней декомпозиции программы каждого процесса на состояния, поскольку взаимодействие между процессами возможно только при переходе автомата из одного состояния в другое. Искусственная декомпозиция, связанная с ограничениями модели, увеличивает время разработки, излишне громоздка и, как следствие, редко используется на практике.

В работе [10], опубликованной научной группой университета Беркли приводится описание 11 моделей встраиваемых систем, которые можно классифицировать на несколько групп:

- обобщенные модели: (DE, CT, FSM);
- модели потоков данных (SDF, PSDF, DDF, DDE, HDF);
- модели параллельных процессов (Rendezvous, CSP, PN).

Наиболее интересными с точки зрения возможности применения их для разработки и синтеза распределенных программ для микроконтроллерных систем является третья группа моделей, позволяющая представить систему как набор взаимодействующих параллельных процессов, описать работу этих процессов и их взаимодействие.

Недостатками данных моделей является то, что они не допускают возможность асинхронного взаимодействия между процессами (только через механизм рандеву) и не содержат механизмов для создания гибких шаблонов модулей управления с возможностью работы с разными выводами устройства.

2 Многоуровневая модель распределенных программ для встраиваемых систем

Для решения рассмотренных проблем, возникающих при разработке распределенных программ для встраиваемых систем предлагается новая многоуровневая модель (рисунок 1).

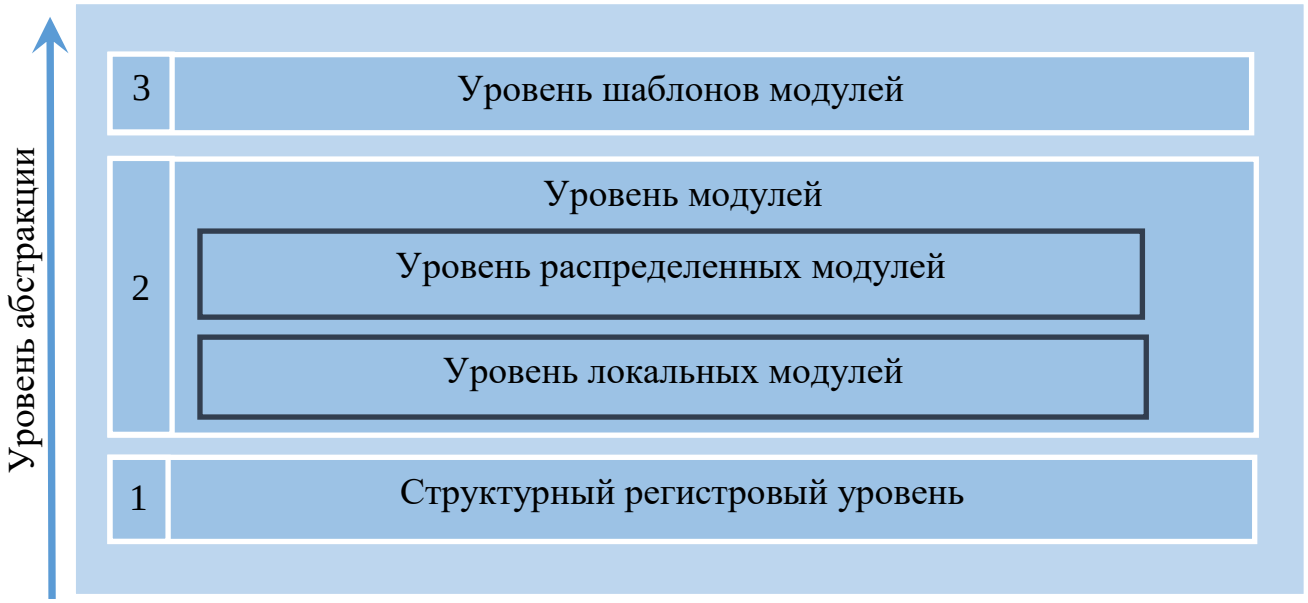

Рис. 1. Уровни предлагаемой модели представления программ управления

Нижний уровень соответствует модели структурного программирования и позволяет обеспечивать совместимость с рассмотренными выше моделями, остальные уровни являются надстройкой, предоставляя новые уровни абстракции.

На нижнем уровне программа устройства описывается в виде инструкций по настройке битов периферийных регистров – определенных ячеек памяти микроконтроллера. Значения битов периферийных регистров интерпретируются электронной схемой устройств и тем самым реализуется выполнение программы микроконтроллером. В соответствии с моделью структурного программирования, инструкции по работе с регистрами описываются последовательно и группируются в подпрограммы с параметрами. Помимо регистров инструкции также могут записывать и считывать значения из переменных.

Второй уровень, уровень модулей, содержит новый вид абстракций – модули. Модули являются единицами высокоуровневой программы в рамках предлагаемой модели. Могут использоваться два вида модулей: задачи и пакеты, объединяющие в себе все основные элементы модели. Пакеты в рамках данной модели идентичны

понятию пакетов в языке Ада или модулей в структурных языках программирования, поэтому наибольший интерес представляют задачи, которые и будут рассматриваться далее. Таким образом, на втором уровне задается множество задач $Tasks = \{task\}$, где элемент множества представляет собой неупорядоченный кортеж, состоящий из названия задачи, множества функций, которые может выполнять задача, множества сигнатур сообщений, которые задача может принимать, и множества событий, которое может в задаче происходить:

$$task = \langle name_{task}, Vars_{task}, Funcs_{task}, MesSigs_{task}, Events_{task} \rangle,$$

где $name_{task} \in ID$ – название задачи $task$ (идентификатор),

$Vars_{task}$ – набор внутренних переменных задачи $task$

$Funcs_{task}$ – множество функций задачи $task$

$MesSigs_{task}$ – множество сигнатур сообщений задачи $task$

$Events_{task}$ – множество событий задачи $task$

Программа в предлагаемой модели имеет статическую структуру, т. е. пользователь на этапе проектирования программы определяет количество и состав задач, которые будут реализованы в каждом устройстве распределенной системы. Данное ограничение обусловлено особенностями микроконтроллеров как вычислителей и спецификой решаемых ими задач (задачи управления системами со статической структурой узлов). Кроме того, это позволяет избежать затрат, связанных с реализацией ресурсоемких алгоритмов, необходимых для поддержки динамической структуры систем управления.

Если пакет является простым объединением переменных, функций и событийных структур, то задача содержит также поток управления, исполняется параллельно другим задачам и содержит список сигнатур принимаемых параметризируемых сообщений и инструкции по принятию и отправке сообщений. Каждая задача может принимать сообщения, отправлять сообщение другой задаче и инициировать событие с параметрами.

Взаимодействие с модулем-задачей осуществляется только за счет обмена сообщениями с ней. Таким образом, задача является модулем не только с точки зрения структуры, но и параллелизма: все функции задачи могут быть вызваны только из потока управления данной задачи. Важно отметить, что модули в рамках предлагаемой модели являются статическими, поскольку до запуска программы известно их количество и их состав.

Еще одним важным отличием задач от известных моделей является то, что взаимодействие между задачами может осуществляться не только синхронно с использованием механизма «рандеву» [19], как это делается в рассмотренных моделях, но также и асинхронно. При этом модуль не ожидает подтверждения принятия ею этого сообщения, что позволяет более эффективно организовывать взаимодействие задач.

Второй уровень модели состоит из двух подуровней: уровень локальных модулей и уровень распределенных модулей. Уровень локальных модулей позволяет

описать программу в виде взаимодействующих модулей в рамках одного устройства. Уровень распределенных модулей позволяет описывать программу в виде взаимодействующих модулей, находящимся в разных устройствах. Кроме того, на уровне распределенных модулей происходит абстрагирование от свойств пассивности / активности оборудования и коммуникационных протоколов: можно считать, что каждое устройство может быть активным и любой из его модулей может инициировать отправку сообщений модулю другого устройства и при этом за реализацию коммуникационных протоколов отвечают особые интерфейсные модули с известным набором сигнатур функций, которые в них можно использовать.

На третьем уровне модели находится множество шаблонов

$$Templs = \{templ^m\},\ m \in \{1..k\},\ k > 0,$$

где m – размерность шаблона, определяемая ограниченным количеством его параметров, а сам шаблон представляет собой кортеж, состоящий из названия шаблона и функции отображения некоторого фрагмента инструкций и параметров шаблона на множество задач.

$$templ^m = \left\langle name_{templ^m}, Finst^m \right\rangle$$
$$Finst^m : C \times Params_{templ}{}^m \to Tasks_{inst},\ Tasks_{inst} \in Tasks$$
$$Params^m = Param_1 \times \ldots \times Param_m$$

Шаблоны являются статическими и имеют набор параметров, при задании конкретных значений которых (при инстанциации шаблона) шаблон становится собственно модулем. Существует три вида параметров шаблонов: параметр-текстовая подстановка, параметр-вывод устройства, параметр-модуль. Таким образом, шаблоны позволяют создавать универсальные модули, способные работать с различными выводами устройства и обращаться к различным сторонним модулям. При этом заложенная в шаблоны гибкость обеспечивается статическим заданием параметров. Поддержка систем статической структуры позволяет задать априорную информацию о работе программе, тем самым уменьшая количество используемой вычислителем памяти и количество действий, необходимых для поддержания целостности динамической структуры и повысить уровень абстракции при разработке распределенных программ.

3 Язык, поддерживающий предложенную модель представления программ

Для разработки распределенных программ для встраиваемых систем в терминах предложенной модели предлагается использовать новый язык, Embeddecy, который по сути является расширением языка C (реализующего нижний слой модели). Язык выключает в себя поддержку всех элементов модели (задачи, пакеты, шабло-

ны), а также некоторые другие необходимые конструкции (помимо конструкций языка С):

- конструкции для обеспечения параллельного программирования (отправка синхронная / асинхронная, принятие сообщения)
- пространства имен;
- анонимные функции;
- типы делегатов и переменные-делегаты;
- события;
- макродирективы для описания параметров шаблонов;
- конструкция via для передачи сообщения между распределенными модулями через указанные интерфейсный модуль;
- разработаны конструкции, позволяющие организовать видимость элементов модулей (области видимости).

Пример кода, включающий основные новые конструкции приведен на рисунке 2.

```
namespace MyNameSpace{ // пространство имен
    importc "headerfile.h"; // импорт С-файла
      import MCUBlocks.atmel.atmega1280.UART1;// импорт шаблона UART1

      #init_module UART1 MY_UART1 // инициализация модуля MY_UART1 по
                        // шаблону UART1
      {baudrate = 9600;} // инициализация статического параметра

      delegate void mydelegate(int a, int b); // описание типа делегата
      mydelegate d = somefunc1; // инициализация делегата somefunc1


      package p1{ // пакет p1
            #main // метка точки входа в программу
            public void init(){
            // подписка анонимной функции на событие, в которой через модуль UART1
            // отсылается сообщение mes1 в задачу task1
                  d += (a, b)->{send task1.mes1(a); via UART1};
                  t1.start(); // запуск задачи
      }}

      task t1{  // задача t1
            message mes1(int x); // сигнатура сообщения mes1
            event event2(int y); // событие 2

            body{
                  while (1){
                        accept mes1(z) do {event2(z);} // принятие сообщения mes1
                        // и вызов события event2
}}}}
```

Рисунок 2 – Пример кода на предлагаемом языке программирования

В таблице 1 приведено сравнение разработанного языка Embeddecy с наиболее популярными языками, используемыми при разработке встраиваемых систем.

Таблица 1 – Сравнение Embeddecy с языками C, C++, Ada.

	C	C++	Ada	Embeddecy
Типизация	Слабая статическая	Слабая статическая, средняя динамическая	Сильная статическая	Возможность использования как слабой, так и сильной статической, утиная динамическая
Поддержка параметрического полиморфизма	-	+	-	+
Возможность создания универсальных шаблонов модулей, способных работать с указанием выводов устройства в качестве параметров шаблона	-	Возможно, но требует непрозрачной реализации, необходимость использования оптимизатора или использования дополнительных ресурсов	-	+
Степень связности модулей	Не ограничивается языком, слабые языковые механизмы	Присутствуют некоторые механизмы	Повышенная связность, в том числе связность параллельных модулей	Высокая связность задач при возможностях реализации несвязных модулей
Языковые средства обеспечения параллельного программирования	-	-	+	+
Делегирование (подписка на сообщения)	-	-	-	Поддерживается, имеется удобный синтаксис с использованием анонимных функций
Средства разработки распределенных программ (абстрагирование от коммуникационных протоколов)	-	-	-	+

Заключение

В работе проанализированы недостатки существующих моделей и языков разработки распределенных программ для встраиваемых микроконтроллерных систем, разработана новая модель представления и поддерживающий ее язык Embeddecy, приведено сравнение языка с популярными в области встраиваемых систем языками

программирования (C, C++, Ada). Разработанный язык является более специализированным, нежели чем вышеупомянутые известные языки, однако позволяет повысить уровень программирования и эффективность разработки в целом.

Библиографический список

1 Белых А. А. Унификация архитектур однокристальных микроконтроллеров и ее применение для разработки программного обеспечения встраиваемых систем: дис. канд. техн. наук: 05.13.15. – М., 2006. – 176 с.

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

3 Большаков О. С. Петров А. В. Модель представления программ управления распределенными микроконтроллерными системами (статья) // Теория и практика системного анализа: Труды III Всероссийской научной конференции молодых ученых. – Т. I. – Рыбинск: РГАТУ имени П. А. Соловьева, 2014. – С. 96 – 107.

4 Платунов А. Е. Встраиваемые системы управления // Control Engineering Россия, №1 (43), 2013.

5 Черемисинов Д. И. Проектирование и анализ параллелизма в процессах и программах / Д. И. Черемисинов. – Минск: Беларус. Наука, 2011. – 300 с. ISBN 978-985-08-1285-8.

6 Дейкстра Э. Заметки по структурному программированию. // Дал У., Дейкстра Э., Хоор К. Структурное программирование. — М.: Мир, 1975. — С. 7–97.

7 Can I use C++ on the AVR? [Electronic resource]. – Режим доступа: http://www.nongnu.org/avr-libc/user-manual/FAQ.html, свободный. – Загл. с экрана.

8 Grace – Graphical Peripheral Configuration Tool [Electronic resource]. – Режим доступа: http://www.ti.com/tool/grace, свободный. – Загл. с экрана.

9 Поликарпова Н. И., Шалыто А. А. Автоматное программирование. 2008 – 167 с.

10 Christopher B., Edward A. Lee, Xiaojun Liu, Stephen Neuendorffer, Yang Zhao, Haiyang Zheng Heterogeneous Concurrent Modeling and Design in Java (Volume 3: Ptolemy II Domains) // Electrical Engineering and Computer Sciences University of California at Berkeley. – 2008.

Сведения об авторах

Большаков Олег Сергеевич – аспирант, ведущий инженер-программист ФГБОУ ВПО «Рыбинский государственный авиационный технический университет имени П. А. Соловьева», г. Рыбинск
Email:bolsh.os@gmail.com

Петров Александр Викторович – ведущий инженер-программист ФГБОУ ВПО «Рыбинский государственный авиационный технический университет имени П. А. Соловьева», г. Рыбинск
Email:gmdidro@gmail.com

Шаров Владимир Григорьевич – кандидат физико-математических наук, профессор ФГБОУ ВПО «Рыбинский государственный авиационный технический университет имени П. А. Соловьева», г. Рыбинск
Email:sharov@rsatu.com