МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ПРОГРАММ УПРАВЛЕНИЯ РАСПРЕДЕЛЕННЫМИ МИКРОКОНТРОЛЛЕРНЫМИ СИСТЕМАМИ

Большаков Олег Сергеевич
РГАТУ имени П. А. Соловьева
Россия, Ярославская обл., г. Рыбинск, ул. 50 лет ВЛКСМ д. 42 кв. 97, bolsh.os@gmail.com
Петров Александр Викторович
ООО “НПП САТЭК плюс”
Россия, Ярославская обл., г. Рыбинск, ул. Куйбышева, д.3 кв. 4, gmdidro@gmail.com

Аннотация

В работе рассматриваются особенности разработки программ для специализированных микроконтроллерных систем как подвида встраиваемых систем. Рассматриваются недостатки существующих подходов к программированию специализированных микроконтроллерных систем. Предлагается четырехуровневая модель представления программ управления распределенными микроконтроллерными системами, на базе которой разрабатывается интегрированная среда разработки программ для микроконтроллерных систем, которая позволяет повысить уровень программирования, генерировать реализации высокоуровневых пользовательских протоколов, описывать кроссплатформенные программы с генерацией программного кода для различных моделей микроконтроллеров на различные языки и под различные существующие компиляторы.

1. Введение

Одной из современных тенденций развития возможностей вычислительных устройств является создание встраиваемых систем и сетей (embedded systems & networks). Встраиваемые системы и сети принято определять как специализированные микропроцессорные системы, непосредственно взаимодействующие с объектом контроля или управления и объединенные с ним конструктивно [1].

Проектирование встраиваемых систем имеет следующие особенности [2, 3]:

- специализированность задач проектирования, уникальность систем;
- 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 со спецификой применения встраиваемые системы очень разнообразны как с точки зрения выполняемой функциональности, так и с точки зрения используемых аппаратных и программных платформ, и, следовательно, методов, применяемых для проектирования таких систем [1]. Несмотря на большое количество готовых программных и аппаратных решений, разработчик каждый раз создает специализированную систему, и поэтому в процессе проектирования ему приходится анализировать все уровни ее организации.

Среди аппаратных платформ, на которых основываются встраиваемые системы, выделяют системы, базирующиеся на микроконтроллерах - однокристальных микросхемах, включающих в себя микропроцессор, оперативное запоминающее устройство, постоянное запоминающее устройство, периферийные блоки управления [4]. Достоинством такой вычислительной платформы является баланс эффективности, универсальности аппаратуры и стоимости всего решения: разработчик выбирает готовое аппаратное решение с фиксированным набором периферийных блоков управления, потребляющих малое количество энергии.

2. Специализированные микроконтроллерные системы как особый вид встраиваемых систем

При проведении классификации микроконтроллеров можно выделить два их типа по степени специализированности под решаемые задачи:

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

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

К первому типу микроконтроллеров относятся 32-битные микроконтроллеры (иногда в этих же целях используются 32- и 64-битные микропроцессоры как отдельные микросхемы). Наиболее популярными представителями данного типа микросхем являются микроконтроллеры на базе архитектуры микропроцессора ARM. Ко втором типу устройств относятся энергоэффективные 8- и 16-битные микроконтроллеры (например, микроконтроллеры семейств AVR, PIC, STM, MCS 51**,** MSP430), а также некоторые 32-битные микроконтроллеры (например, ARM Cortex-M0, ARM Cortex-M0+). На устройства, в состав которых входят микроконтроллеры первого типа, для реализации многофункциональности устанавливают операционные системы общего назначения: Linux (в том числе Android), iOS, Window Phone и другие. Многие из устройств на базе данного типа микроконтроллеров представляют собой мобильные компьютеры и предназначены для выполнения широкого круга задач в разнообразных сферах деятельности человека. Операционные системы в таких устройствах управляют приложениями и различными видами ресурсов, что обеспечивает максимальную гибкость и расширяемость функциональности при наличии достаточно универсального набора интерфейсов (USB, Bluetooth, Wi-Fi) и датчиков (микрофон, сенсор экрана, гироскоп, акселерометр, компас, цифровая камера, GPS, датчик света, датчик уровня заряда аккумулятора). В данном виде систем часто используются различного рода виртуальные машины и в целом процесс проектирования программного обеспечения для них схож с процессом проектирования для общецелевых компьютерных систем - используются те же методологии и схожие технологии.

Совершенно иначе обстоит дело со вторым типом микроконтроллеров, которые специализированы под узкий круг решаемых задач. На базе таких микроконтроллеров обычно создают либо автономные устройства, либо такие микроконтроллеры используются как посредники, передавая информацию от датчиков до центрального устройства системы, которым может быть устройство с микроконтроллером первого типа. Так, например, специализированные микроконтроллеры входят в состав мобильных систем для работы с периферией мобильного устройства - клавиатурой, сенсором, графическим дисплеем и т. д. Особенностью специализированных микроконтроллеров является то, что они обладают на несколько порядков меньшими объемами памяти и частотой процессора, чем микроконтроллеры первого типа. Фактически задачей такого микроконтроллера является генерация определенных видов сигналов для управления внешним оборудованием: либо сигналов специальной формы (широтно-импульсная модуляция, тактирование, генерация аналогового сигнала), либо общение по одному из стандартизованных низкоуровневых протоколов (SPI, RS-232, TWI, USB, CAN и других). Генерация сигналов может осуществляться аппаратно с использованием периферийной системы, режимы и состояние которой задаются периферийными регистрами. Особенностями микроконтроллеров является наличие эффективных режимов энергосбережения: различные режимы сна и варианты пробуждения из них, или использование асинхронной событийной системы, позволяющей передавать между периферийными модулями события без участия центрального процессора, который при работе на высокой частоте потребляет много энергии.

В таблице 1 приведено сравнение различных архитектур микроконтроллеров. Сравнение производится по основным видам ресурсов: количество памяти, битность регистров, количество оперативной памяти и максимальная частота процессора. Как видно из таблицы, 32-битные

микропроцессоры и микроконтроллеры ARM по количеству ресурсов на несколько порядков превосходят энергоэффективные 16- и 8-битные микроконтроллеры. Именно это и позволяет использовать на данных контроллерах мобильные операционные системы общего назначения.

Архитектуры	ARM	PIC, MSP430	AVR, PIC, STM8
Битность	32, 64	16	8
ОЗУ, порядок	Внешняя DRAM,	< 100 Кб	< 10 Кб
Частота	500МГц..2 ГГц	20..70 МГц	< 20 МГц
Использование ОС общего назначения	возможно	невозможно	невозможно

Таблица 1 - Сравнение особенностей архитектур микроконтроллеров

3. Недостатки существующих технологий для разработки программ для специализированных микроконтроллеров

Выделим два важных аспекта технологий разработки программ:

1) предоставляемый уровень абстракции;

2) предоставляемый уровень динамичности.

Под предоставляемым уровнем абстракции будем понимать набор слоев абстракции, на которых пользователь может описывать программу. Низший слой абстракции называется низкоуровневым, высший слой абстракции называется высокоуровневым. Под предоставляемым уровнем динамичности будем понимать набор возможностей по изменению состояния программы и ее функциональности в процессе ее выполнения. Под состоянием программы понимается количество и структура выделенной памяти, а также хранимые в ней значения. Под функциональностью программы понимается набор исполняемых ею алгоритмов.

Как известно, возможность использования высших слоев абстракции является преимуществом любой технологии разработки программ, однако улучшение этого показателя зачастую коррелирует со снижением эффективности разрабатываемой программы, что связано с накладными расходами на обеспечение высокого уровня. То же касается и аспекта динамичности: высокие показатели данного аспекта используемой технологии позволяют делать гибкие, легко изменяемые программы как до выполнения, так и во время. В то же время при повышении уровня динамичности программы в общем случае так же уменьшается её эффективность, что связано либо с дополнительными проверками, необходимыми для корректности изменения параметров программы, либо с использованием дополнительных ресурсов (используется большее количество памяти). При проектировании программного обеспечения разработчику приходится делать выбор технологии согласно выделенным аспектам: таким образом у него возникает задача выбора технологии с приемлемой высокоуровневостью технологии, динамичностью и эффективностью получаемых программ.

На данный момент существует множество инструментов для программирования специализированных микроконтроллеров: Atmel Studio, CodeVisionAVR, WinAVR, AtmanAVR, IAR Embedded Workbench, MicroC/Pascal/Basic PRO for AVR/PIC, ICC AVR, AVR Eclipse Plugin, Algorithm Builder, BASCOM-AVR, LDMicro, MPLAB IDE, Proton Development Suite, Gimpel Software PC-Lint, Flowcode, MicroCode Studio, Pic Micro Pascal, Code Composer Studio, ST Visual Develop IDE и другие [5, 6, 7]. Несмотря на большое разнообразие инструментов для программирования микроконтроллеров, набор возможных технологий является скудным:

большинство из существующих подходов к программированию специализированных микроконтроллеров предполагают разработку программ на языке С. Реже используется С++ с ограничениями и в редких случаях другие текстовые языки; существуют реализации графических языков (LD, блок-схемы).

Рассмотрим перечисленные технологии программирования микроконтроллеров с точки зрения выделенных ранее аспектов.

1) предоставляемый уровень абстракции

Существующие технологии, предполагающие разработку на языках С, Pascal, языке блок-схем и т. п., нельзя назвать высокоуровневыми, т. к. в рамках этих технологий программист традиционно пишет программу для микроконтроллера, настраивая биты периферийных регистров устройства и считывая с них значения. Существуют следующие подходы, которые частично решают проблему высокоуровневости для данной технологии:

а) использование метода восходящего проектирования для вложения функций в обобщенные функции-агрегаты, имеющие большее количество параметров, чем исходные функции;

б) создание генераторов инициализационного кода по настройке периферии микроконтроллера.

Первый подход наиболее распространен при разработке библиотек кода и неизбежно приводит к избыточности, т. к. при увеличении количества параметров, в коде функций появляются избыточные проверки для принятия решений о вызове тех или иных низкоуровневых функций. Проблему избыточности пытаются решать средствами условной компиляции, в результате чего код таких библиотечных функций становится крайне сложным для анализа и расширения, что связано с неприспособленностью используемых языков предоставлять возможности для повышения степени абстракции с учетом статических свойств программ.

Второй подход заключается в том, что пользователь задает настройки периферии и с помощью генератора получает исходный код, который выполняет эту инициализацию. В данном случае повышение уровня абстракции происходит только при задании настроек периферии и не распространяется на всю программу. Тем не менее это позволяет немного повысить уровень разработки части программ: для создания генераторов используются либо наборы диалоговых окон (программная система CodeVisionAVR), либо используются графические редакторы, схематически показывающие связь пользовательских настроек с непосредственной работой аппаратной части микроконтроллера и специализированные калькуляторы для расчета задаваемых числовых параметров (инструмент QtouchComposer в составе среды Atmel Studio для микроконтроллеров AVR [8], STM32Cube для микроконтроллеров STM32 [9], инструмент Grace в составе Code Composer Studio для микроконтроллеров MSP430 [10]).

Как видно, оба подхода не решают проблему комплексно: первый подход сильно 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.

2) предоставляемый уровень динамичности

В рассматриваемых технологиях существует возможность динамического управления памятью (выделение блоков памяти во время исполнения программы). Тем не менее, данная возможность в специализированных микроконтроллерах используется редко. При изучении особенностей программ для специализированных микроконтроллеров становится очевидным важное их свойство, которое обуславливает как архитектуру их аппаратуры, так и ограничения используемых технологий - статичность. Статичность программ для микроконтроллеров в сравнении с программами для компьютеров обуславливается особенностями микроконтроллерных систем:

- специализированные микроконтроллерные системы управления являются функционально распределенными, т. е. еще до разработки программы для устройства известно,

какие функции оно будет выполнять, и набор этих функций не меняется в процессе работы устройства;

- ограничения по габаритам, массе и энергопотреблению часто вынуждают использовать автономное питание, малое количество оперативной памяти, низкие частоты микропроцессора - такие процессы, как динамическое управление памятью, обработка исключений становятся избыточными, и поэтому обычно память под переменные в программе выделяется статически, т. е. на этапе компиляции / линковки модулей;

- микроконтроллеры предназначены для использования в системах управления и не требуют серьезных вычислений - для ресурсоемких вычислений, требующих больших объемов памяти, существуют более мощные вычислители (цифровые сигнальные процессоры, ПЛИС).

- в случае реализации систем с динамическими связями (сенсорные сети и т.д.), реализация динамических связей осуществляется за счет заранее известного протокола (что тоже позволяет выделять память статически).

Все вышеприведенные причины статичности микроконтроллерных систем управления отражаются и на архитектуре, по которой построена аппаратная часть микроконтроллеров. Так энергоэффективные микроконтроллеры (AVR, PIC, STM8) построены на базе Гарвардской архитектуры (в случае PIC - расширенной Гарвардской архитектуры), что предполагает раздельное хранение данных и инструкций программ, причем у программы отсутствует возможность полиморфизма (изменение набора инструкций) в режиме ее исполнения (только на этапе “прошивки” микроконтроллера программой).

Именно из-за свойства статичности программ управления для микроконтроллерных систем в данной области не используются технологии из компьютерных систем или используются их усеченные аналоги, позволяющие реализовывать статические программы, что нивелирует в таких средствах разработки те достоинства, которые изначально были в них заложены. Примером использования технологии из области компьютерных систем может являться технология объектно-ориентированного программирования на базе языка C++: ситуация с использованием данной технологии такова, что существующие компиляторы под специализированные микроконтроллеры не позволяют использовать все возможности объектно-ориентированного языка, поскольку динамическое инстанцирование и удаление объектов, хранение таблиц методов, поиск метода при его вызове, использование механизма исключений являются крайне ресурсоемкими, и как следствие, избыточными процессами для специализированных микроконтроллеров. Именно по этой причине из реализаций C++ для, например, микроконтроллеров AVR, исключены операторы для динамического инстанцирования и удаления объектов. C++ в данном случае фактически используется только с той целью, чтобы привнести в язык модульность на уровне статических классов, которой нет в языке C. По этим же причинам в специализированных микроконтроллерах не используется технология .Net Micro Framework [11], содержащая интерпретатор, среду для выполнения объектно-ориентированной программы, механизмы обработки исключений, сборщик мусора.

К недостаткам технологий в области программирования микроконтроллерных систем стоит отнести не только отсутствие подходов, предоставляющих адекватную реализацию программ в аспектах высокоуровневости и одновременно статичности программ, но также следующие проблемы существующего инструментария, отстающего от инструментария в области компьютерных систем:

- у разработчиков отсутствует возможность отлаживать программу на распределенной системе (или ее модели);

- используемые языки программирования не содержат конструкций для организации параллельного исполнения процессов; отсутствуют стеки протоколов и в то же время отсутствуют какие бы то ни было технологии, упрощающие создание протоколов.

Отсутствие подходов, позволяющих реализовывать эффективные программы для специализированных микроконтроллерных систем с точки зрения высокоуровневости и одновременно статичности программ приводят к выводам о необходимости разработки нового

подхода, которые бы смог повысить уровень и вместе с тем предоставить адекватные средства для реализации статичных программ.

4. Модель представления программ управления распределенными микроконтроллерными системами

Ниже представляется новый подход к разработке программ для специализированных микроконтроллеров, предполагающий описание программы в рамках новой многоуровневой модели, низший слой которой соответствуют уровню абстракции языка С при программировании микроконтроллеров. Предлагаемая модель имеет архитектуру, представленную на рисунке 1.

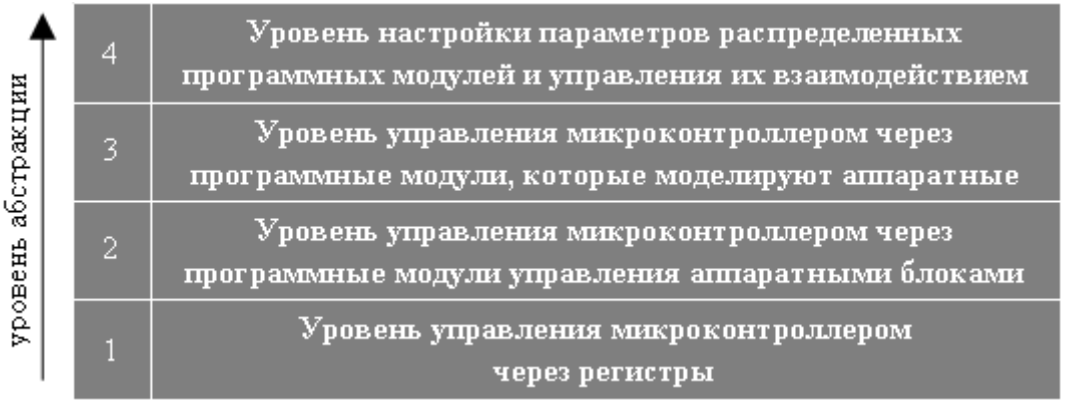

Рис. 1. - Архитектура предлагаемой модели представления программ управления

Предлагается выделить следующие уровни представления программы управления распределенной системой в порядке увеличения степени абстракции:

1) уровень управления микроконтроллером через регистры;

2) уровень управления микроконтроллером через программные модули управления аппаратными блоками микроконтроллера;

3) уровень управления микроконтроллером через программные модули, которые моделируют аппаратные модули микроконтроллера;

4) уровень настройки параметров распределенных программных модулей управления взаимодействием программных модулей и настройки их параметров.

В качестве средства, позволяющего описать программу в рамках данной модели предлагается новый высокоуровневый язык, являющийся надстройкой над языком С, и позволяющий описывать программы в рамках разрабатываемой интегрированной среды программирования.

Поясним значение каждого уровня на примере несложной системы: два микроконтроллера взаимодействуют друг с другом по беспроводному каналу связи (через радиомодули). К одному из них подключен переменный резистор, к другому - светодиод. Необходимо разработать программы для микрокроконтроллеров так, чтобы при изменении напряжения на потенциометре, менялась яркость светодиода, подключенного к другому микроконтроллеру (рис. 1).

На данном уровне программа для устройства описывается в виде инструкций по настройке периферийных регистров - определенных ячеек памяти микроконтроллера. Настройки периферийных регистров интерпретируются электронной схемой устройства и тем самым реализуется аппаратное выполнение программы. Описание функций тех или иных регистров обычно производится в технической документации к программируемым устройствам. К типичным инструментам для программирования микроконтроллеров на этом уровне относятся языки программирования assembler, С. Для управления микроконтроллером программист записывает данные в память периферийных регистров и считывает данные из нее.

Часть из этих регистров позволяет напрямую задавать состояния и считывать их с пинов микроконтроллера, остальные необходимы для изменения внутренних режимов работы устройства. Пример программного кода на языке С в логике состояния регистров, позволяющего задать настройки периферийного интерфейсного блока UART и передать два байта (3,4) приведен на рисунке 2.

Рис. 2. – Пример микроконтроллерной схемы

1) Уровень управления микроконтроллером через регистры

```
/* настройка UART  */
UBRRH=00;
UBRRL=25;//установка делителя частоты на скорость 19200 бит/с
UCSRB=(1<<RXEN)|(1<<TXEN); //включение приемника и передатчика
UCSRC=(1<<URSEL)|(1<<USBS)|(3<<UCSZ0); //формат кадра данных:
//слово данных 8 бит, 2 стоп-бита
/* Отправка по UART пакета из двух байт: 3, 4 */
while ( !( UCSRA & (1<<UDRE)) ) ; // ждать, пока не будет готов буфер отправки
UDR = 3; // положить данные в буфер для отправки
while ( !( UCSRA & (1<<UDRE)) ) ; // ждать, пока не будет готов буфер отправки
UDR = 4; // положить данные в буфер для отправки
```

Рис. 2. – Программный код работы с UART на уровне регистров

2) Уровень управления микроконтроллером через программные модули управления аппаратными блоками микроконтроллера

На данном уровне микроконтроллер представляется как набор аппаратных блоков, выполняющих аппаратные процессы, программа управляет этими процессами через соответствующие программные модули: задачи и пакеты, в которых выполняется настройка регистров. Большинство аппаратных блоков предназначены для реализации интерфейсов, т. е. стандартизованных в области микроконтроллерных систем протоколов канального уровня (UART, SPI, TWI, CAN, USB и т. д.) для передачи данных между программируемым контроллером и внешним оборудованием, подключенным к нему, кроме того блоки обеспечивают генерацию определенных форм сигналов и передачу данных по более простым цифровым и аналоговым интерфейсам. К числу таких блоков относятся таймеры, счетчики, ШИМ-блоки, АЦП, аналоговые компараторы, блоки организации внешних прерываний и т. д.. Вместе с тем, существуют аппаратные блоки, предназначенные строго для внутреннего использования программой микроконтроллера (некоторые таймеры / счетчики) и не выдающие сигналов для внешнего оборудования.

В рамках разрабатываемой интегрированной среды программирования существует библиотека программных модулей управления блоками, в которой описываются шаблоны модулей, для которых необходимо задать конкретные значения для возможность использования данных модулей, т. е. произвести инстанцирование модуля по шаблону. На рисунке 3 приведен

пример программного кода на высокоуровневом языке программирования, в котором производится инстанцирование модуля управления аппаратным блоком UART из шаблона, с помощью этого модуля затем отправляется сообщение по одноименному интерфейсу микроконтроллера.

3) Уровень управления микроконтроллером через программные модули, которые моделируют аппаратные модули микроконтроллера

Данный уровень позволяет абстрагироваться от природы процессов за счет добавления виртуальных блоков, т. е. блоков, логика которых полностью реализуется программными модулями с возможностью использования модулей управления аппаратными блоками. Управление регистрами в данном случае используется как внутренняя реализация блока и служит для осуществления непосредственного считывания и вывода цифровых сигналов с ножек микроконтроллера. Кроме того, на этом уровне управления возможна иерархическая компоновка программных модулей, т. е. создание программных модулей, которые используют другие модули. Поэтому на этом уровне появляются модули, управляющие внешним подключенным к микроконтроллеру оборудованием, что выводит степень абстракции за рамки микроконтроллера. На рисунке 4 показан пример высокоуровневого кода, в котором происходит инстанцирование модуля управления виртуальные блоком RadioModuleControl, который управляет радиомодулем по интерфейсу UART (т. е. при использовании модуля управления аппаратным блоком UART).

```
#init_template UART
{
  baud_rate = 19200;
  transmit = true;
  receive = true;
  word_size = UART.WORS_SIZES.bit8;
  stop_bits = UART.STOP_BITS.TWO
}
// установить радио модуль в режим готовности
UART.TransmitBytes( 0x06, 0x00 );
UART.TransmitBytes( 0x05, 0x00 );
UART.TransmitBytes( RF23B_PWRSTATE_REGISTR, RF23B_PWRSTATE_READY );
UART.TransmitBytes( 0x3E, 2); //  установить количество передаваемых байт (2)
UART.TransmitBytes( 0x7f, 5 ); // отправить байт “5”
UART.TransmitBytes( 0x7f, 6 ); // отправить байт “6”
// начать беспроводную передачу байтов
UART.TransmitBytes(RF23B_PWRSTATE_REGISTR, RF23B_PWRSTATE_TX);
UART.TransmitBytes(RF23B_INTERRUPT_REGISTR, RF23B_PACKET_SENT_INTERRUPT);
```

Рис. 3. – Программный код по работе с UART через модуль управления аппаратным блоком

```
#init_template UART
{
  <...>
}
#init_template RadioModuleControl
{
UARTblock = UART; // задать UART в качестве используемого интерфейсного блока
}
RadioModuleControl.EnqueueBytes(5, 6); // додавить в очередь байты “5”, “6”
RadioModuleControl.StartTransmit(); // начать беспроводную передачу пакета данных
```

Рис. 4. – Программный код для отправки сообщения радиомодулю через модуль управления виртуальным блоком

4) Уровень управления взаимодействием программных модулей и настройки их параметров

На данном уровне распределенная система управления представляется как набор программных модулей (задач и пакетов), непосредственно взаимодействующих друг с другом, что позволяет абстрагироваться от участвующей в обмене информацией аппаратуры и низкоуровневых протоколов, по которым производится взаимодействие.
Задачи выполняются параллельно относительно друг друга. Возможно как асинхронное взаимодействие между задачами, так и синхронное. Синхронное взаимодействие осуществляется при использовании механизма “рандеву” [12].

Также на данном уровне происходит абстрагирование от свойств пассивности / активности оборудования в схеме: можно считать, что каждое устройство может быть активным и любой из его модулей может инициировать отправку сообщений модулю другого устройства.

На рисунке 5 приведен пример на высокоуровневом языке, на котором одной строчкой кода настраивается автоматическая отсылка сообщения OnWithBrightnees модуля управления светодиодом при каждой оцифровке значения, производимого модулем управления потенциометром.

```
// задание параметров шаблона модуля управления потенциометром
#initblock PotentiometerControl
{
   pin = [PORTB, 2]; // настройка вывода, через который подключен потенциометр
}
<...> // задание параметров шаблонов модулей управления UART, RadioModuleControl
// автоматическое изменение яркости светадиода
// при изменении значения потенциометра
PotentiometerControl.OnValueRecieved += mcu2.LedControl.OnWithBrightness;
```

Рис. 5. - Программный код настройки автоматического изменения яркости светодиода при каждой оцифровке значения с потенциометра

5. Заключение

В работе рассмотрены особенности разработки программ для специализированных микроконтроллерных систем как подвида встраиваемых систем. Рассмотрены недостатки существующих подходов к программированию специализированных микроконтроллерных систем, не способных предоставлять одновременно высокий уровень разработки программ и реализовывать необходимые для эффективности статические элементы. В работе предложена новая модель представления программ управления распределенными микроконтроллерными системами. На основе предложенной модели разрабатывается инструментальная среда разработки программ для микроконтроллерных систем, использование которой дает следующие преимущества:

- повышается уровень программирования с уровня регистров до уровня распределенных по устройствам модулей управления - для этого используется язык программирования высокого уровня и графические редакторы-конфигураторы схемы оборудования;
- появляется возможность естественно представить в программе паралеллизм распределенных процессов и упростить их синхронизацию;
- производится абстрагирование от коммуникационных протоколов, по которым производится взаимодействие программных модулей: системой производится генерация протоколов на основе настроек интерфейсных блоков;
- появляется возможность описывать кросс-платформенные программы управления с возможность получения кода для нужной модели микроконтроллера на нужном языке и для нужного компилятора.

Литература

1. Платунов А.Е., Постников Н.П. Высокоуровневое проектирование встраиваемых систем. – СПб.: НИУ ИТМО, 2011. – 121 с.
2. 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.
3. Платунов А.Е. Встраиваемые системы управления // Control Engineering Россия, №1 (43), 2013.
4. Gridling G., Bettina W. Introduction to microcontrollers. — Vienna University of Technology, Institute of Computer Engineering, Embedded Computing Systems Group. — Vienna, 2006.
5. Atmel Studio [Electronic resource]. – Режим доступа: http://www.atmel.com/ru/ru/tools/atmelstudio.aspx, свободный. – Загл. с экрана.
6. Code Composer Studio (CCStudio) Integrated Development Environment (IDE) v5 [Electronic resource]. – Режим доступа: http://www.ti.com/tool/ccstudio, свободный. – Загл. с экрана.
7. ST Visual develop IDE for developing ST7 and STM8 applications [Electronic resource]. – Режим доступа: http://www.st.com/web/en/catalog/tools/FM147/CL1794/SC1807/SS1747/PF210567, свободный. – Загл. с экрана.
8. QTouch Composer — полная интеграция [Electronic resource]. – Режим доступа:http://www.atmel.com/ru/ru/Microsite/atmel_studio6/qtouch_tools.aspx, свободный. – Загл. с экрана.
9. TM32Cube initialization code generator (UM1718) [Electronic resource]. – Режим доступа: http://www.st.com/web/catalog/tools/FM147/CL1794/SC961/SS1743/PF259242?icmp=stm32cubemx_pron_pr-stm32cubef2_apr2014, свободный. – Загл. с экрана.
10. Grace – Graphical Peripheral Configuration Tool [Electronic resource]. – Режим доступа: http://www.ti.com/tool/grace, свободный. – Загл. с экрана.
11. .Net Micro Framework [Electronic resource]. – Режим доступа: http://www.netmf.com/, свободный. – Загл. с экрана.
12. Деревянко А.С., Солощук М.Н. Операционные системы: Учебное пособие. - Харьков: НТУ "ХПИ", 2002. - 573с.