2. Разумейчик, В.С. Структурно-химическое моделирование гидратации цементного камня / В.С. Разумейчик // Вестник БрГТУ. – 2006. – №1(38): Строительство и архитектура. – С. 91–96.
3. Дзівінец, А.А. Распрацоўка праграмнага забеспячэння для мадэлявання дысперсных матэрыялаў / А.А. Дзівінец, С.С. Дзерачэннік, В.С. Разумейчык // П'ята науково–практична конференція FOSS LVIV 2015: збірник наукових праць, Львів, 23–26 квітня 2015 р. – Львів: Львівский національний університет імені Івана Франка, 2015. – С. 23–26.
4. Королев, В.Ю. О распределении размеров частиц при дроблении / В.Ю. Королев // Информатика и ее применение. – 2009. – Т. 3. – С. 60–68.
5. Барон, Л.И. Проверка применимости уравнения Розина-Рамлера для исчисления диаметра среднего куска при взрывной отбойке горных пород / Л.И. Барон, Г.М. Сиротюк // Взрывное дело. – М.: Недра, 1978.
6. Дивинец, А.А. Выбор вероятностного закона распределения для модельного описания дисперсности заполнителя бетонного композита / А.А. Дивинец, С.С. Дереченник, В.С. Разумейчик // Вестник БрГТУ. – 2015. – №5(95): Физика, математика, информатика. – С. 54–57.
7. Дзівінец, А.А. Распрацоўка праграмнага забеспячэння для генерацыі грануламетрычнага складу матэрыялаў на аснове размеркавання Вейбулла / А.А. Дзівінец, С.С. Дзерачэннік, В.С. Разумейчык, С.В. Лапіч // Шоста науково–практична конференція FOSS LVIV 2016 : Збірник наукових праць, Львів, 19–22 квітня 2016 р. – Львів: Львівский національний університет імені Івана Франка, 2016. – С. 13–17.
8. Разумейчик, В.С. Стохастическая структурно-фазовая модель гидратирующих цементных систем: автореферат диссертации на соискание ученой степени кандидата технических наук: 05.23.05. – Брест: БрГТУ, 2012. – 25 с.
9. Адамсон, А. Физическая химия поверхностей / А. Адамсон. – М.: Мир, 1979. – 568 с.
10. Бузмакова, М.М. Перколяция сфер в континууме / М.М. Бузмакова // Известия Саратовского университета. Новая серия. Серия: Математика. Механика. Информатика. – 2012. – Т. 12. – № 2. – С. 48–56.

Материал поступил в редакцию 10.01.2017

DIVINETS A.A. Development of the disperse materials simulation software

A software system is presented, targeted at creating disperse material models, as far as at primary analysis of those models. The sofware complex contains three modules, making it possible to create material composition based on Weibull distribution, to generate disperse system structure, and to define models characteristics. The main application area of the project is building industry.

УДК 004.2

Коваленко В.Ю., Костюк Д.А., Николаюк-Ртищева М.В., Поляков В.И., Склипус А.А., Четверкина Г.А.

ПОСТРОЕНИЕ ПРАКТИКУМА ПО ПРОГРАММИРОВАНИЮ ARM-СОВМЕСТИМЫХ МИКРОКОНТРОЛЛЕРОВ С ИСПОЛЬЗОВАНИЕМ СРЕДСТВ ВИРТУАЛИЗАЦИИ И УДАЛЕННОЙ ОТЛАДКИ

Введение. Наблюдавшийся в последние годы рост производительности микроконтроллеров, обусловленный расширившимся применением 32-битных процессоров на базе архитектуры ARM, существенно повлиял на рынок электроники, приведя к повсеместному использованию принципов и подходов разработки, характерных для встраиваемых электронных систем (embedded systems), где цена и габаритные ограничения менее важны по сравнению с производительностью.

Одной из особенностей перехода к 32-битным высокопроизводительным процессорным ядрам в микроконтроллерах является возможность отказаться от части периферии, решавшей проблемы маломощных вычислительных ядер 8- и 16-битных систем. Сочетание высокопроизводительных ядра и сопроцессора, периферии, интерфейсов последовательной передачи данных, драйвера ЖК-дисплея позволяет легко использовать современный микроконтроллер в качестве основы для систем управления, контроля и мониторинга, цифровых систем промышленных приложений реального времени. Такие платформы не только производительны, но и удобны для разработки средствами высокоуровневого программирования на языках C, C++, удаленной отладки [1].

Отметим, что изучение студентами-программистами архитектуры встраиваемых систем и соответствующих принципов разработки важно не только с точки зрения требований рынка, предъявляющего повышенный спрос на подобных специалистов. В ряде случаев система на основе 32-битного микроконтроллера может служить более простым и наглядным пособием по программированию микропроцессоров и принципам архитектуры ПК, а без знания языка машинных команд и архитектуры невозможно в достаточной степени освоить принципы работы вычислительной техники. Даже помимо областей, в которых применяется непосредственно низкоуровневое программирование (разработка драйверов устройств, компиляторов, жесткая оптимизация критических модулей), пробелы в данной сфере компетенции служат источником неоптимального и уязвимого кода на языках высокого уровня, приводят к заведомо неверным решениям при разработке программных систем [2, 3].

При всей актуальности, создание учебных курсов по программированию 32-битных встраиваемых систем сопряжено сразу с несколькими техническими проблемами. Если микропроцессор программируется для использования без операционной системы, возникает проблема взаимодействия с пользователем: в отсутствие уровня абстракции, предоставляемого ОС, современные устройства ввода-вывода слишком сложны, чтобы их программирование могло стать фрагментом лабораторного практикума. Искусственное же упрощение периферии лишает учебный курс универсальности, жестко привязывая его к какому-либо конкретному аппаратному комплекту разработчика (starterkit) из имевшихся на рынке в момент создания. В ряде учебных курсов эта проблема обходится использовани-

Коваленко Владимир Юрьевич, *старший преподаватель кафедры ЭВМиС Брестского государственного технического университета.*
Костюк Дмитрий Александрович, *к.т.н., доцент, доцент кафедры ЭВМиС Брестского государственного технического университета.*
Николаюк-Ртищева Марина Владимировна, *старший преподаватель кафедры ЭВМиС Брестского государственного технического университета.*
Поляков Виктор Иванович, *к.т.н., доцент, профессор кафедры ЭВМиС Брестского государственного технического университета.*
Склипус Алла Арсеньевна, *старший преподаватель кафедры ЭВМиС Брестского государственного технического университета.*
Четверкина Галина Андреевна*, старший преподаватель кафедры ЭВМиС Брестского государственного технического университета.*
Беларусь, БрГТУ, 224017, г. Брест, ул. Московская, 267.

ем микроконтроллерных устройств, разработанных исключительно для учебных целей – таких, как робототехнический комплект-конструктор LEGO Mindstorms. Подобные решения имеют развитую периферию, включающую различные исполнительные механизмы; однако из-за специфики целевой аудитории они комплектуются сильно упрощенными средствами и языками программирования, имеющими мало общего с реальной встраиваемой системой [4, 5]. Наконец, курс программирования полноценных ARM-устройств с ОС, хоть и более востребован с точки зрения текущего состояния индустрии [6], по сути оказывается курсом программирования под системные вызовы конкретной ОС, а в ряде случаев и вовсе проводится на традиционных персональных компьютерах [7, 8].

В настоящей статье представлены результаты разработки лабораторного практикума, создававшегося на кафедре электронных вычислительных машин и систем БрГТУ и ориентированного на преодоление ограничений, озвученных выше.

1. Выбор программной платформы и общих принципов построения практикума. С точки зрения базиса для изучения низкоуровневого программирования у широко продаваемых комплектов разработчика на базе процессоров ARM помимо ценовой доступности можно отметить также ряд важных для учебного процесса преимуществ – таких как расширяемость, наличие специальных отладочных портов (JTAG), доступность принципиальных схем (что позволяет легче иллюстрировать особенности работы аппаратуры), а также качественная поддержка архитектуры ARM ядром Linux и соответствующими инструментами разработки. Последнее обстоятельство важно с учетом энергичного роста (переходящего в доминирование) доли GNU/Linux в секторе встраиваемой электроники, а также того, что открытость, конфигурируемость и масштабируемость данной ОС делают ее подходящим кандидатом на роль учебного инструмента [7].

В рамках разработанного практикума студенты знакомятся с двумя подходами к управлению микроконтроллерными устройствами:

1) запуск программы, написанной на языке Ассемблер и откомпилированной кросс-компилятором, непосредственно на процессоре ARM;
2) выполнение программы под управлением ОС (GNU/Linux), запущенной на ARM-устройстве.

Принцип работы с написанной программой заключается в ее первоначальном запуске в эмуляторе для проверки работоспособности и поиска ошибок и последующей записи на отладочную плату для тестирования на реальном оборудовании, а также получения навыков использования средств удаленной отладки и управления. В качестве отладочной платы подходит практически любой starterkit на платформе ARM.

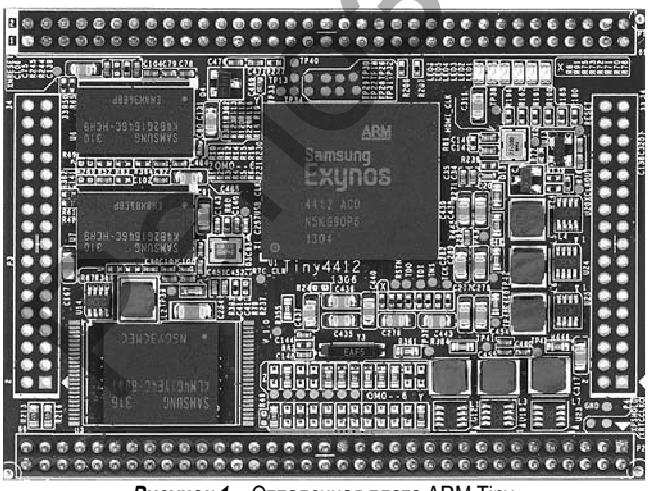

Рисунок 1 – Отладочная плата ARM Tiny

В нашем опыте в качестве аппаратной платформы в учебном процессе использована одна из разновидностей подобных устройств – FrendlyARM Tiny на базе четырехъядерного ARM-процессора Exynos4412 с ядром Cortex-A9, 1 ГБ ОЗУ и поддержкой флеш-памяти объемом до 32 ГБ (рис. 1). Периферия данного устройства достаточно обширна (сенсорный дисплей, USB-host, Audio-кодек, G-sensor, miniPCIe, сеть Ethernet), что оставляет значительную свободу при выборе тем занятий практической части учебного курса. Поддерживаемыми операционными системами для данного устройства (как и для широкого спектра других отладочных плат данного класса) являются разновидности GNU/Linux (в комплекте поставляется специализированная версия дистрибутива Ubuntu, а также версия Android).

Первичная проверка работы программы в эмуляторе упрощает многочисленные запуски и пересборки проекта, а также получение информации о состоянии процессора. Нами использован эмулятор QEMU, обладающий широкой поддержкой различных архитектур, включая ARM и при этом достаточно часто используемый при профессиональной разработке [8].

2. Подсистема эмуляции. Простейший способ выполнения кода без использования операционной системы, на процессоре, эмулируемом QEMU, когда он сброшен в нулевое состояние (после запуска виртуальной машины) – это размещение откомпилированной программы в памяти по нулевому адресу. Этот способ используется нами для программ, разрабатываемых в начальной части практикума, когда студенты еще не знакомы с системой прерываний и не могут ее использовать. В QEMU встроена возможность эмуляции нескольких отладочных плат для ARM. Чтобы протестировать программу, в рамках практикума требуется создать файл-образ, размер которого соответствует объему памяти выбранной для эмуляции отладочной платы. Для этого предлагается использовать стандартную для Unix-подобных систем команду «dd», сначала для копирования заданного числа нулевых байтов в файл flash.bin, а затем для копирования содержимого откомпилированного бинарного файла с программой в начало файла flash.bin. Далее имя созданного файла образа передается при запуске в качестве одного из параметров эмулятору qemu-system-arm. В простейшем случае, когда опрос эмулируемого устройства невозможен, проверка правильности работы программы сводится к анализу содержимого регистров, что выполняется во встроенном мониторе виртуальной машины (qemu monitor) командой «info registers». Общая схема кросс-компиляции и запуска учебных программ показана на рис. 2.

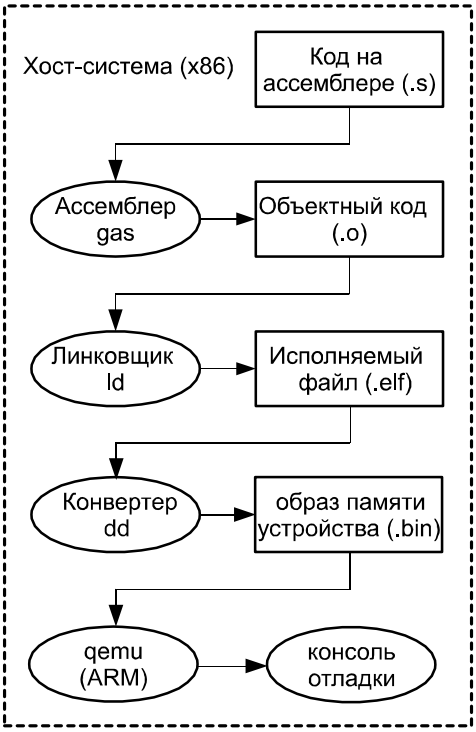

Рисунок 2 – Схема кросс-компиляции и запуска ассемблерных программ в эмуляторе

Для обмена информацией, т. е. обеспечения диалогового режима учебных программ, работающих без ОС, используется UART (универсальный асинхронный приемопередатчик), эмулируемый системой qemu-system-arm. Схема его соединения с консолью, т. е. проброс из хост-системы (снаружи эмулятора) в гостевую систему (внутрь), показана на рис. 3. Как можно видеть, для удобства отладки последовательная консоль эмулятора соединяется с одним из графических окон терминала, запущенных на хост-системе.

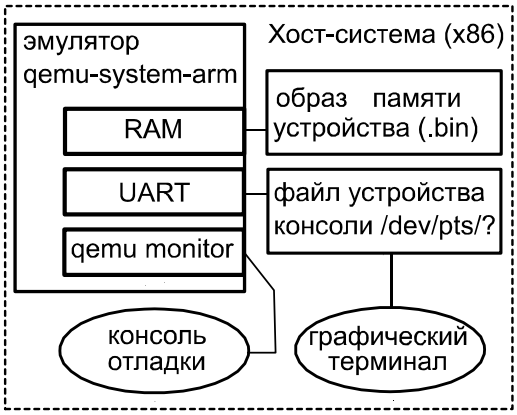

Рисунок 3 – Схема обеспечения диалогового режима

Заметим, что реализованная модель в QEMU не воспроизводит в точности передачу байтов с точки зрения синхронизации и задержек, существующих в реальном устройстве (переданные данные в эмулируемом UART оказываются мгновенно доступны по тому адресу памяти, на который отображается его соответствующий регистр). Это позволяет использовать в начальной части практикума упрощенный механизм взаимодействия, с приближением в последующих работах к особенностям реального устройства (с проверкой значений по дополнительному порту, соответствующему регистру флагов UART, для определения поступления байт во входной FIFO-буфер и/или освобождения выходного).

Запуск отлаженной программы на реальном устройстве выполняется аналогичным образом. Для этого также требуется записать бинарный файл программы по адресу, на который передается управление после старта процессора. В начальной части практикума используется отладочный интерфейс JTAG, хотя предусмотрена также возможность делегирования данной функции загрузчику uBoot. В работах, в которых функции ОС недоступны, для получения информации об исполнении программы также используется JTAG (с помощью утилиты OpenOCD).

3. Выбор инструментов разработки и структура практикума. В качестве системы для разработки и кросс-компиляции на ПК в практикуме использован дистрибутив Ubuntu Linux. Помимо стандартного набора инструментов разработчика GNU toolchain, входящего в состав дистрибутива, использованы следующие пакеты [5]:

1. Sourcery CodeBench ARM – кросс-компилятор, точнее, одна из существующих сборок компилятора GCC с поддержкой ARM (используются транслятор arm-none-linux-gnueabi-as, компоновщик arm-none-linux-gnueabi-ld, а также утилита objcopy для преобразования форматов объектных файлов);
2. BuildRoot – пакет для автоматизированной сборки файловой системы и дополнительных библиотек, позволяющий откомпилировать версию GNU/Linux для отладочной платы и подготовить образ флеш-памяти, содержащий файловую систему (включающий собственно ядро Linux, пакеты uBoot для загрузки и BusyBox в качестве минимального набора системных утилит);
3. интегрированная среда для удобства написания ПО и удаленной отладки (студентам предлагается на выбор несколько вариантов; наиболее популярным является использование среды Eclipse, достаточно широко применяемой в embedded-разработке).

В структуре практикума предусмотрено шесть работ, которые можно разделить на две группы: работы по изучению аппаратной архитектуры и системы команд ARM, а также базового программного инструментария, и работы, посвященные функционированию микроконтроллерного устройства под управлением ОС:

Часть практикума, непосредственно касающаяся архитектуры ARM, включает четыре работы.

- Вводная работа позволяет познакомиться со структурой программы и синтаксисом ассемблера ARM, использованием аппаратных регистров, базовых инструментальных средств для кросс-компиляции и тестирования программы.
- Далее более подробно изучаются параметры запуска ассемблера GAS, использование инструментария make для сборки многофайлового проекта, особенностей организации оперативной памяти и доступа к ней. Также в этой работе затрагивается использование средств генерации листинга, и в ходе изучения его структуры выполняется практическое ознакомление с влиянием директив ассемблера на работу транслятора и компоновщика.
- Следующая работа посвящена изучению вычислительных возможностей процессора ARM, а также особенностей использования различных типов памяти, включая энергонезависимую FLASH-память, для хранения результатов вычислений.
- Финальная работа, связанная с ассемблером платформы ARM, касается изучения системы прерываний, обеспечения корректной работы программы с векторами прерываний, а также совмещения ассемблерного кода и модуля, написанного на языке С.

Часть практикума, связанная с функционированием устройства под управлением ОС, состоит из двух работ:

1. Первая работа посвящена ознакомлению с базовым набором системных программных средств, необходимых для компиляции и установки на ARM-устройство дистрибутива GNU/Linux.
2. Во второй работе изучаются принципы и инструменты удаленной отладки, и выполняется отладка программы, запущенной на ARM-устройстве под управлением сформированного на предыдущем этапе образа ОС, с помощью отладчика GDB.

Заключение. В качестве результатов подобного планирования практикума можно отметить значительное снижение порога вхождения в низкоуровневое программирование для архитектуры ARM, достигаемое применением виртуализации и описанных композиционных решений. Однако при этом с самых первых работ сохраняется непосредственное взаимодействие студентов с машинной архитектурой и системой команд, с постепенным приближением создаваемого машинного кода к уровню сложности, осмысленному для применения на реальных системах. Вторая часть практикума дает возможность практического знакомства с подготовкой встраиваемой системы, функционирующей под управлением ядра Linux, без значительного углубления во внутренние особенности выбранной ОС (т. е. в материалы курсов по операционным системам и системному программному обеспечению), за счет использования средств ее упрощенного развертывания.

СПИСОК ЦИТИРОВАННЫХ ИСТОЧНИКОВ

1. Павлов, П. Тенденции развития микропроцессоров и микроконтроллеров // Современная электроника. – 2007. – № 2. – С. 12–15.
2. Костюк, Д.А. Выполнение лабораторного практикума по основам языка ассемблера на платформе GNU/Linux / Д.А. Костюк, А.М. Жук // Информационные технологии в образовании: материалы Международной научно-практической конференции, 21–22 мая 2009 г. – Минск: БНТУ, 2009. – С. 84–88.
3. Костюк, Д.А. Методика переноса изучения низкоуровневого программирования и вычислительной архитектуры на платформу GNU/Linux / Д.А. Костюк, Г.А. Четверкина, А.М. Жук, Т.П. Сидорович // Вестник БрГТУ. – 2011. – № 5(71): Физика, математика, информатика. – С. 62–64.
4. Talaga, P. Combining AIMA and LEGO mindstorms in an artificial intelligence coursetobuild realworldrobots / P. Talaga, J.C. Oh // Journal of Computing Sciences in Colleges. – V. 24, Iss. 3, 2009. – P. 56–64.

5. Абдулгалимов, Г.Л. Обучение робототехнике: от элементарных понятий до программирования микроконтроллеров вузов // Новые информационные технологии в образовании: сборник научных трудов 16-й Международной научно-практической конференции «Новые информационные технологии в образовании» 2–3 февраля 2016 г. / Г.Л. Абдулгалимов, В.Н. Казагачев, А.А. Гулюта; под общ. ред. проф. Д.В. Чистякова. – М.: ООО «1С-Паблишинг», 2016. – Часть 2. – С. 309–311.
6. Пынькин, Д. Linux-образование – симбиоз вузов, коммерческих компаний и LUG / Д. Пынькин, Ю. Адамов // Девятая конференция «Свободное программное обеспечение в высшей школе»: тезисы докладов. Переславль, 25-6 января 2014 г. – М.: Альт Линукс, 2014. – С. 41–45.
7. Костюк, Д.А. Построение практикумов по программированию встраиваемых систем / Д.А. Костюк, И.С. Кутень, В.С. Разумейчик // Десятая конференция «Свободное программное обеспечение в высшей школе»: тезисы докладов Переславль, 24–25 января 2015 года. – М.: Альт Линукс, 2015. – С. 14–18.
8. Пынькин, Д. Обучение Linux в корпоративном секторе / Д. Пынькин, В. Шахов // Открытые технологии: сборник материалов девятой Международной конференции разработчиков и пользователей свободного программного обеспечения Linux Vacation / Eastern Europe 2013, Гродно, 27–30 июня 2013 г. – Брест: Альтернатива, 2013. – С. 105–109.
9. Костюк, Д.А. Построение практикумов по программированию периферийных устройств и архитектуре ЭВМ на базе GNU/Linux / Д.А. Костюк, А.М. Жук // Тези міжнародної науково-практичної конференції FOSS LVIV–2011, Львів, 1–6 лютого 2011р. – Львів: Вид. ЛНУ ім. І. Франка, 2011. – С. 73–75.
10. Каваленка, У.Ю. Дэманстрацыйна-тэставая ферма праграм для платформы Android з вэб-інтэрфэйсам / У.Ю. Каваленка, Д.А. Касцюк // П'ята науково-практична конференція FOSS Lviv 2015: збірник науковых праць, Львів, 23–26 квітня 2015 р. – Львів, 2015. – С. 42–45.

Материал поступил в редакцию 09.01.2017

KOVALENKO V.Yu., KOSTIUK D.A., NIKOLAYUK-RTSISHCHAVA M.U., POLYAKOV V.I., SKLIPUS A.A., CHETVIORKINA G.A. Building the practical programming course for ARM-based microcontrollers with use of virtualization and remote debugging

The specifics of ARM processors and GNU/Linux operating system usage to teach students embedded systems programming is investigated. Advantages of the proposed course structure are analyzed. Proposed set of software tools includes development and debugging instruments, as far as virtualization software. A sequence of laboratory works is defined, which allows to study specifics of ARM-based microcontroller systems programming as far as general build priciples of embedded systems.

УДК 004.08.01

Осолинский А.Р., Карачка А.Ф., Палий И.О.

ПОДСИСТЕМА ПРИНЯТИЯ РЕШЕНИЙ ДЛЯ ИИС СРЕДНЕГО ЭНЕРГОПОТРЕБЛЕНИЯ МИКРОКОНТРОЛЛЕРОВ

Введение. Встроенные компьютерные системы с автономным питанием используются все шире, в качестве примера можно привести концепцию Internet of Things, IoT (Интернет вещей). Одной из актуальных задач при разработке таких систем является увеличение времени автономной работы микроконтроллера. Одним из перспективных направлений для решения данной проблемы является оптимизация программного обеспечения по энергопотреблению при выполнении команд, инструкций, фрагментов программ и программ в целом.

Проведенный анализ методов и средств измерения мгновенного и среднего энергопотребления МК, которые описаны в [1, 3] показали, что в [1] измерение среднего энергопотребления проводилось без учета остаточного заряда на конденсаторах в цепи питания МК, полученные модели энергопотребления имели низкую точность (погрешность ± 10% и более).

В работе [2] предлагалось измерять мгновенное значение потребления тока МК через «токовое зеркало», но недостатком данного метода является то, что МК работает в нештатном режиме (конденсатор в цепи питания отделен от МК). Кроме того, падение напряжения на транзисторах "токового зеркала" зависит от тока потребления процессором. Изменение такого падения напряжения достигает 0,25 В, что влияет на точность значений вычисляемого энергопотребления. Погрешность метода составляет 7% [4].

Наивысшую точность обеспечивает схема [3]. Но она имеет относительно низкую помехоустойчивость (электромагнитные помехи от электрической сети влияют на результат измерения), при измерении энергопотребления фрагментов программ и программ в целом, схеме [3] свойственно накопление погрешности.

Поэтому было предложено, результаты исследования энергопотребления МК при выполнении отдельных инструкций и команд с помощью схемы [3] контролировать путем их сравнения с результатами измерения среднего энергопотребления, с помощью системы, описанной в [5, 6]. Дополнительным преимуществом системы, описанной в [5, 6], является то, что ее структура близка к схеме устройства, описанного в [3]. Это позволяет без опасения сравнивать полученные результаты измерений мгновенного и среднего энергопотребления – потому что схема питания МК и элементы, которые существенно влияют на погрешности измерения обоих видов энергопотребления, остаются теми же самыми.

Тем не менее одним из важных условий обеспечения высокой точности и помехоустойчивости предложенных в [5, 6] методов измерения среднего энергопотребления МК является своевременное окончание процесса измерения.

Целью данной работы является разработка подсистемы принятия решений для ИИС среднего энергопотребления МК [6], которая позволяет своевременно завершать процесс измерения энергопотребления команд, фрагментов программ и программ в целом, которые выполняются микроконтроллером.

Усовершенствованная система измерения энергопотребления МК. На основе предложенных в [5, 6] методов измерения среднего энергопотребления МК разработана усовершенствованная система измерения энергопотребления микроконтроллеров, структурная схема которой показана на рисунке 1.

***Осолинский А.Р.**, к.т.н., преподаватель кафедры информационно-вычислительных систем и управления фак-та компьютерных информационных технологий Тернопольского национального экономического университета.*
***Карачка А.Ф.**, к.т.н., доцент кафедры информационно-вычислительных систем и управления фак-та компьютерных информационных технологий Тернопольского национального экономического университета.*
***Палий И.О.**, к.т.н., доцент кафедры информационно-вычислительных систем и управления фак-та компьютерных информационных технологий Тернопольского национального экономического университета.*
Украина, 46009, г. Тернополь, Тернопольская область, ул. Львовская, 11.