Моделирование МК с использованием объектно-ориентированных языков высокого уровня

Константин Оськин (г. Пермь)

Статья посвящена составлению имитационной модели микроконтроллера на языке Java. Объектом исследования является взаимодействие ядра микроконтроллера с АЦП и последовательным портом (UART, SPI, I²C и др.) Приведены результаты работы модели.

Проблема исследования микропроцессорных систем

Анализ литературы [1–3] показывает, насколько многогранной может быть задача описания характеристик работы микропроцессорной системы в целом и микроконтроллера в частности. Как правило, разработанные методики сводятся к представлению процессорной системы в качестве системы массового обслуживания (СМО) или более сложной структуры – сети массового обслуживания (СеМО). Затем каждый процесс микроконтроллера представляется в виде потока заявок, который в классической теории СМО рассматривается как Марковский поток либо как композиция Марковских потоков.

Более прогрессивными методами анализа являются асимптотические методы расчёта циклических систем [2]. В этом случае микроконтроллер рассматривается как циклически функционирующая система, внутри которой действуют несколько потоков, обладающих разными приоритетами. Достоинством этого метода является простота расчёта среднего времени прерывания неприоритетной заявки, общего времени выполнения заявки, а также дисперсии и коэффициентов вариации этих величин.

На практике существенной проблемой становится ограничение на вид потока заявок: Марковский поток близок к реальному, но не воспроизводит поток системы, а рассмотрение более сложных видов потоков требует специальных знаний в области теории вероятностей, высшей математики, прикладной теории СМО и занимает много времени, поскольку все методы содержат множество вычислений параметров.

Казалось бы, проще сделать систему без каких-либо вычислений, однако современные архитектуры процессоров ARM и Cortex обладают такой развитой периферией, что расчёт делать придётся. И тут вспоминается ещё один инструмент – имитационное моделирование, которое позволяет наглядно представить функционирование системы в целом и выделить места, оказывающие наибольшее влияние на производительность.

Имитационное моделирование позволяет разработчику заранее провести оптимизацию системы с целью повышения её быстродействия. Однако сложности возникают и здесь. Среди наиболее распространённых средств имитационного моделирования можно выделить специализированный язык GPSS, а также среду Matlab+Simulink. Основным преимуществом специализированных средств является набор готовых конструкций и функций, например, генераторов случайных чисел, подходящих для всех моделей.

Как отмечено в [4], среди десятков языков моделирования невозможно выделить лучший, который превосходил бы остальные по всем показателям, – каждый имеет свои достоинства и недостатки, а также узкую область применения. Кроме того, большинство языков моделирования являются коммерческими продуктами, что накладывает ограничение на их использование.

Вторым фактором является сложность практического использования специализированных языков. В лучшем случае, познания среднестатистического инженера в области языка GPSS ограничены вводным курсом, прослушанным в вузе.

Третий, и, пожалуй, самый главный недостаток заключается в том, что создаваемая при помощи специализированного языка модель оказывается жёстко ограниченной рамками данного средства, что снижает гибкость её описания.

В отличие от моделирования с использованием специализированных средств, моделирование с использованием универсальных языков программирования высокого уровня позволяет организовать гибкий подход к исследуемой модели. Вторым достоинством, актуальным при решении практических задач, является распространённость таких языков и богатый опыт программирования, описанный в литературе. Побочным продуктом процесса моделирования системы могут стать алгоритмы и коды, которые можно перенести в рабочий проект системы.

Автором в качестве средства был выбран язык Java по причине его сходства с языком Си. Кроме того, Java – объектно-ориентированный язык, что предоставляет определённые удобства в работе с проектом. Однако автор не настаивает на использовании именно этого языка в качестве основного средства моделирования.

Основные понятия

Прежде чем приступить к рассмотрению процесса моделирования, требуется определить основные понятия, которые будут использованы в данной статье.

Под *заявкой* мы будем понимать процесс или последовательность действий, которые микроконтроллер выполняет при наступлении соответствующего события. *Трудоёмкость заявки* будет равна длительности выполнения этих операций. Например, заявкой будет сообщение, пришедшее по протоколу обмена, а трудоёмкостью – длительность его обработки.

Очередь является интуитивно понятным термином. Заявки могут накапливаться в очереди. Уходом заявки из очереди является момент начала её обслуживания. Примером реальной очереди в микроконтроллерах может быть накопление присылаемых байтов в буфере до получения сообщения целиком.

Загрузкой системы ρ называется отношение трудоёмкости заявок к общему времени наблюдения за системой. Загрузка может быть рассчитана как для каждого отдельного элемента, так и для системы в целом. В последнем случае загрузка определяется как сумма загрузок всех элементов системы. При этом суммарная загрузка не может быть больше 1.

Под *фоновыми операциями* будем понимать регулярно возникающие операции, не критичные к длительности обслуживания.

Событием будем считать изменение состояния какого-либо элемента системы в соответствующий момент времени. В модели событие характеризуется временем его возникновения и источником события – элементом, изменившим своё состояние.

Подробные определения этих терминов, а также математические основы теории СМО изложены в [3].

Описание моделируемой системы

Рассмотрим микроконтроллер, в состав которого входит многоканальный АЦП, опрашивающий измерительные каналы с частотой F_0. Существует некоторое расписание опроса каналов, состоящее из k тактов длительностью $1/F_0$ (с). Следует отметить, что процесс составления расписания опроса является одним из предметов исследования многоканальных измерительных систем, его подробное описание можно найти в [1].

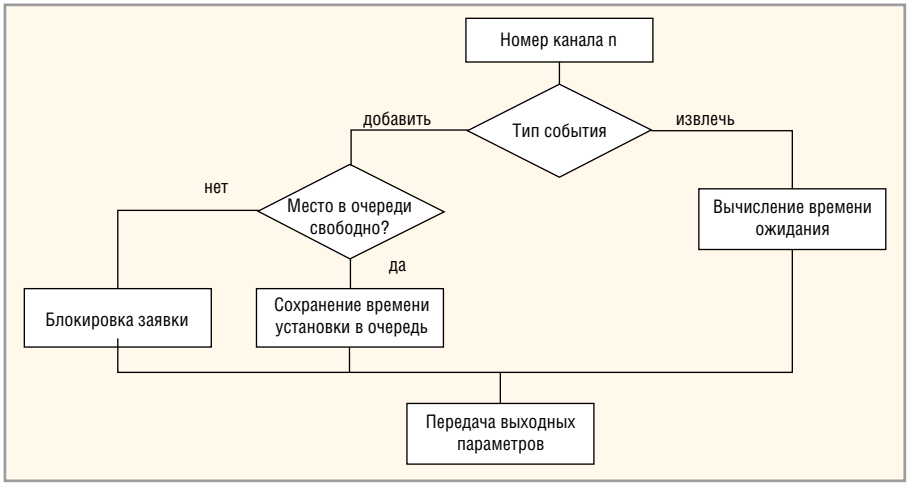

Рис.1. Блок-схема алгоритма функционирования буфера заявок

Система подключена к шине передачи данных, по которой поступают информационно-управляющие сообщения, являющиеся командами системы управления, передаваемыми по шине передачи данных. Интервал поступления и длительность обслуживания этих сообщений являются случайными величинами. Предположим, что приблизительно оценена средняя трудоёмкость информационно-управляющей заявки, равная φ (мкс), а также известна средняя интенсивность поступления заявок λ.

Выделим в рассматриваемой системе следующие блоки: n-канальный АЦП,

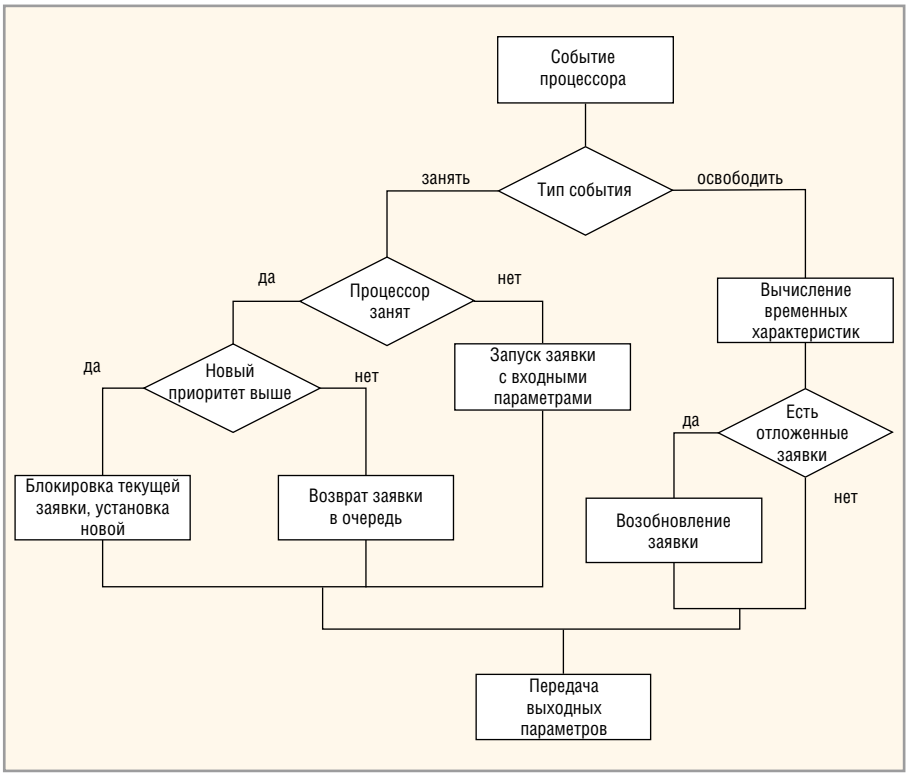

Рис. 2. Блок-схема алгоритма функционирования процессора

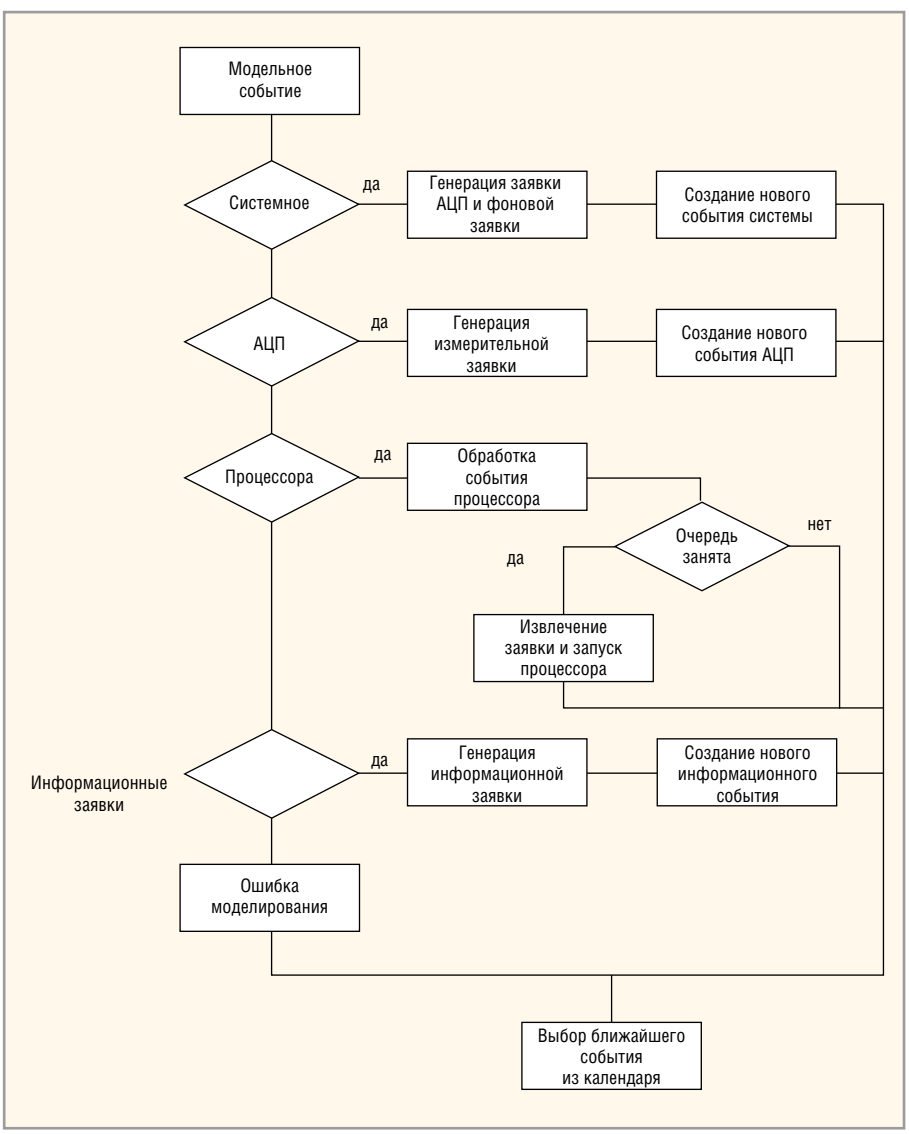

Рис.3. Блок-схема алгоритма управления модельными событиями

многоканальный буфер заявок, процессор, источник информационно-управляющих заявок и буфер информационно-управляющих заявок. В процессе моделирования нас будет интересовать время обработки неприоритетных заявок. Кроме того, потребуется определить влияние общей загрузки на характеристики системы.

Составим алгоритм работы каждого элемента системы. С точки зрения объектно-ориентированного подхода каждый элемент системы представляет собой объект, обладающий своими свойствами и методами. Так, АЦП является объектом, генерирующим измерительные заявки с частотой коммутации F_0. Время генерации заявки (время преобразования АЦП) составляет μ (мкс). Поскольку АЦП является управляемым устройством, номер канала АЦП, сгенерировавшего заявку, преобразователь получает в качестве входного аргумента.

После генерации заявка передаётся на обработку процессору либо, если он занят, попадает в буфер, где под каждую заявку в канале отведено заранее определённое количество мест. Следовательно, каждая входящая заявка характеризуется номером канала АЦП, временем генерации и временем постановки в буфер. Если место в буфере занято, более ранняя заявка удаляется и её место занимает другая. Алгоритм функционирования буфера заявок может быть представлен в виде блок-схемы, приведённой на рисунке 1.

Аналогичным образом функционирует буфер информационно-управляющих заявок. Отличия заключаются в наличии одного канала буфера информационных заявок и вариации времени обслуживания процессором. Таким образом, дополнительным параметром, характеризующим поступившую заявку, является время обработки процессором, распределённое по заранее определённому закону.

Теперь рассмотрим работу микроконтроллера. При моделировании будем считать, что после приостановки обслуживания заявки её обработка возобновляется с места остановки. Прерывание обработки неприоритетной заявки увеличивает время её обслуживания на суммарное время прерывания. Также допустим, что загрузка процессора фоновыми операциями распределена равномерно на всей длительности цикла опроса каналов и равна ψ (мкс) за такт. Время обработки измерительной

ной заявки будем считать постоянной величиной, равной *t* (мкс).

В процессе работы процессор выбирает заявки из буферов измерительных и информационных заявок. Если в буфере приоритетных заявок появляется элемент, то обслуживание текущей неприоритетной заявки приостанавливается до тех пор, пока не освободится буфер приоритетных заявок. С точки зрения моделирования, объект «процессор» характеризуется набором состояний: «занят приоритетной заявкой», «занят неприоритетной заявкой», «свободен» и т.д. Входными параметрами процессора являются тип поступившей заявки, трудоёмкость и время поступления. Исходя из описания функционирования процессора, составлен алгоритм его работы, показанный в виде блок-схемы на рисунке 2.

Одним из важных вопросов имитационного моделирования является выбор модельного времени [4]. Ряд событий системы наступает случайным образом, следовательно, целесообразно применять событийно-ориентированный подход, но тогда возможно нарушение причинно-следственных связей (causality error) [4]. Суть проблемы заключается в том, что при параллельном моделировании событий может возникнуть ситуация, когда в календарь ставится событие с более ранним моментом начала, а моделирующая система на данном шаге сделала выбор в пользу более позднего события. Таким образом, происходит нарушение причинно-следственных связей.

Для управления потоком событий моделируемая система должна быть снабжена механизмом управления модельными событиями. При этом в качестве единицы модельного времени целесообразно выбрать величину, наиболее близкую к длительности изменения состояния элементов системы. Для моделирования высокопроизводительных информационно-измерительных систем целесообразно выбирать микросекунду.

На каждом шаге моделирования требуется выбирать из календаря событие, которое наступит раньше остальных, что снижает производительность системы моделирования. Решением может стать конвейер операций, на который будут поступать несколько ближайших событий в порядке их следования. Новые события, вычисленные на текущем шаге, будут устанавливаться в конвейер в том случае, если они наступят раньше, чем установленное событие. Описание конвейера представлено в листинге 1.

Листинг 1

```
// Входной параметр: n – номер источника события (система,
процессор, АЦП, фон)
// evt[n] -календарь событий (массив событий)
// cnvh, cnvl – номера двух источников с ближайшими событиями

public static void cnv_check (int n){

       if (cnvh == n) {
       // Если подвинули время следующего по порядку события,
       // сбрасываем конвейер, чтобы не нарушить
       причинно-следственные связи
              cnvh = -1;
              cnvl = -1;
       }
       else {
              if (cnvh >=0) {
                     if (evt[n] < evt[cnvh]){
                            cnvl = cnvh;         // Сдвигаем очередь
                            cnvh = n;
                     }
                     else {
                            if (cnvl >= 0){
                                   if (evt[n] < evt[cnvl]) cnvl = n;
                            }
                     }
              }
       }
}

public static void main(String[] args){
// инициализация модели
// -------
// Моделирование системы
while (t < tmax){

       // imit_sys(int i) – процедура обработки события от источника n
       if (cnvh >= 0) imit_sys(cnvh);
       // Пересчитываем следующий шаг
       if (cnvh >=0) t =evt[cnvh];
       else {
       // Если конвейер не установлен
              cnvh = 0;
              cnvl = 1;
              // Ищем два наиболее ранних события
              for (int i = 1; i <4; i++){
                     if (evt[i] < evt[cnvh]){
                            cnvl = cnvh;
                            cnvh =i;
                     }
                     else if (evt[i] < evt[cnvl]) cnvl = i;
              }
              t = evt[cnvh];
       }
}
}
```

Алгоритм управления модельными событиями показан на рисунке 3. Он объединяет все описанные элементы моделируемой системы и фактиче-

Листинг 2

```
public static double
rndExp(double lambda){
          double u;
   // нельзя, чтобы u
// получилось равное 0,
// т.к. логарифм от нуля
// не существует
          while ((u = random.
nextDouble()) <= 0.00001);
          return -1.0/
lambda*Math.log(u);
}

// Функция возвращает случайную
// величину, с параметром lam,
// не менее 10
// случайная величина – сумма
// ЭСВ
public static double rnd_sum_exp
(double lam){
       double tim;
       tim = rndExp(lam);
       while (tim < 10) {
              tim = tim +
rndExp(lam);
       }
       return tim;
}
```

Листинг 3

```
// Функция возвращает случайно
// выбранную величину
// из набора ttp
static final double ttp[] =
{2.3, 5.7, 1.1, 7.8, 3.2, 4.1};

public static double
rndNorm(double alpha, double
sigma)
{
       // получаем случайное
// число, имеющее стандартное
// нормальное распределение
   double u = random.
nextGaussian();
   // приводим его к нормальному
// распределение с параметрами
// alpha и sigma
   return alpha + sigma*u;
}

public static double rnd_set
int i){
       i = (int)rndNorm(4, 4);
       while ((i > 5)||(i < 0)){
       i = (int)rndNorm(4, 4);
       }
       return ttp[i];
}
```

Таблица 1. Экспериментальные данные

Частота F_0, кГц	Параметры	Среднее время выполнения заявок, мкс		Средняя загрузка процессора заявками, %		
		Информационных	Фоновых	Измерительными	Информационными	Фоновыми
250	τ = 2,5 мкс	12,96	3,99	62,49	64,17	≈ 100
	τ = 2,0 мкс	9,85	3,94	49,99	49	98,62
	τ = 1,5 мкс	7,77	2,89	37,49	39,04	72,33
	τ =1,0 мкс	6,53	2,03	25	32,57	50,83
	φ = 5,0 мкс	10,01	3,99	49,99	50,29	99,82
	φ = 4,5 мкс	9,05	3,74	49,99	45,28	93,66
	φ = 4,0 мкс	8,04	3,48	49,99	40,24	87,04
	φ = 3,5 мкс	7,04	3,2	49,99	34,97	80,01
	λ = 0,04	9,84	3,45	49,99	39,45	86,24
	λ = 0,03	9,87	2,97	49,99	29,79	74,19
	λ = 0,02	9,85	2,48	49,99	19,86	62,17
	λ = 0,01	9,89	2,02	49,99	10,02	50,45
	ψ = 1,1мкс	9,77	3,99	49,99	48,88	≈ 100
	ψ = 0,9 мкс	9,8	3,7	49,99	49,22	92,72
	ψ = 0,8 мкс	9,85	3,49	49,99	49,35	87,32
	ψ = 0,7 мкс	9,87	3,27	49,99	49,08	81,84
232	τ = 2,5 мкс	11,72	4,29	58,14	58,54	≈ 100
	τ = 2,0 мкс	9,1	3,67	46,51	45,87	85,39
	τ = 1,5 мкс	7,54	2,32	34,88	37,63	53,95
	τ =1,0 мкс	6,43	2,11	23,55	32,17	48,98
	φ = 5,0 мкс	9,35	3,72	46,51	46,71	86,61
	φ = 4,5 мкс	8,38	3,44	46,51	42,02	80,2
	φ = 4,0 мкс	7,52	3,19	46,51	37,58	74,2
	φ = 3,5 мкс	6,64	2,95	46,51	33,4	68,56
	λ = 0,04	9,17	3,16	46,51	36,74	73,48
	λ = 0,03	9,18	2,66	46,51	27,5	61,87
	λ = 0,02	9,24	2,19	46,51	18,47	50,86
	λ = 0,01	9,19	1,71	46,51	9,1	39,82
	ψ = 1,1 мкс	9,19	3,91	46,51	45,94	90,99
	ψ = 0,9 мкс	9,11	3,44	46,51	45,59	80,02
	ψ = 0,8 мкс	9,17	3,26	46,51	45,96	75,77
	ψ = 0,7 мкс	9,12	3,07	46,51	46,08	71,48
217	τ = 2,5 мкс	10,81	4,59	54,35	54,29	≈ 100
	τ = 2,0 мкс	8,64	2,63	43,48	42,61	57,35
	τ = 1,5 мкс	7,32	2,37	32,61	36,4	51,56
	τ =1,0 мкс	6,25	2,15	21,73	31,31	46,82
	φ = 5,0 мкс	8,87	2,72	43,48	44,77	59,11
	φ = 4,5 мкс	7,92	2,54	43,48	39,86	55,23
	φ = 4,0 мкс	7,17	2,39	43,48	35,87	52,03
	φ = 3,5 мкс	6,17	2,21	43,48	30,93	48,03
	λ = 0,04	8,73	2,35	43,48	34,86	51,08
	λ = 0,03	8,71	2,02	43,48	26,21	43,94
	λ = 0,02	8,72	1,69	43,48	17,63	36,73
	λ = 0,01	8,74	1,34	43,48	8,68	29,16
	ψ = 1,1 мкс	8,62	2,73	43,48	43,18	59,36
	ψ = 0,9 мкс	8,66	2,6	43,48	43,04	56,44
	ψ = 0,8 мкс	8,64	2,51	43,48	43,06	54,51
	ψ = 0,7 мкс	8,62	2,43	43,48	42,99	52,84

ски является алгоритмом её функционирования. Поскольку при описании каждого элемента был использован объектно-ориентированный подход, то блоки в представленном алгоритме являются объектами, реализующими свои процедуры с входными аргументами и возвращающие выходные параметры. В данном случае проявляется достоинство объектно-ориентированного подхода – наглядное представление системы в виде совокупности составляющих элементов, а также оперирование объектом, а не набором его свойств, при составлении алгоритма функционирования.

Ещё один момент, на котором следует остановиться, – генерация случайных чисел. На практике генераторы экспоненциально распределённых случайных величин (ГЭСВ) нередко вступают в противоречие со здравым смыслом. Так, например, в системе существует наименьшая и наибольшая трудоёмкость заявки. Тем не менее, ГЭСВ способен сгенерировать как крайне малую, так и неразумно большую величину трудоёмкости, при этом среднее значение выдаваемых величин будет рав-

но требуемому. Решением проблемы может стать описание собственного закона распределения, отвечающего условиям решаемой задачи.

В листинге 2 представлен генератор случайных чисел, генерирующий заявки по сложному закону распределения. Листинг 3 иллюстрирует ГСВ, выбирающий случайным образом число из детерминированного набора. Приведённые примеры иллюстрируют ещё одно достоинство моделирования систем на языке Java: гибкость реализации процедур и прозрачность получаемых результатов.

Место для творчества

Разумеется, процесс моделирования не ограничивается указанными методами. Модель может быть усложнена для решения любой практической задачи. Например, можно ввести в неё второй АЦП. Также среди всех процессов микроконтроллера можно выделить ряд задач, обладающих своим приоритетом и требующих решения в определённое время. Таким образом, в системе возникает дополнительный поток заявок, требующих обработки по отдельному алгоритму.

Экспериментальные данные

При помощи составленной автором моделирующей программы были проведены эксперименты с моделью информационно-измерительной системы. В качестве закона распределения времени обработки информационной заявки и частоты возникновения информационных заявок использовался экспоненциальный закон распределения. Были определены времена обработки заявок в зависимости от загрузки процессора информационными, измерительными и фоновыми заявками. Полученные данные представлены в таблице. Видно, что на время выполнения заявки влияет загрузка системы приоритетными заявками, а также собственное время выполнения заявки. Изменение загрузки системы информационными, измерительными и фоновыми заявками не влияет на время обработки этих заявок системой.

Выводы

Моделирование микропроцессорных систем на языках высокого уровня обладает рядом достоинств, ярко выраженных при использовании результатов моделирования на практике. Применение этих языков при моделировании микропроцессорных систем предоставляет разработчику новые возможности по исследованию системы на ранней стадии проектирования. Полученные с помощью имитационного моделирования результаты могут быть использованы для качественного анализа характеристик разрабатываемой системы.

Литература

1. Цифровые адаптивные информационно-измерительные системы / Под ред. Б.Я. Авдеева и Е.А. Черняховского / СПб.: Энергоатомиздат, 1997.
2. *Задорожный В.Н.* Распределение календарного времени обслуживания неприоритетных заявок в системах с абсолютными приоритетами / Омский научный вестник, 2006. № 8(1). С.124–132.
3. *Алиев Т.И.* Основы моделирования дискретных систем. СПбГУ ИТМО, 2009.
4. *Труб И.И.* Объектно-ориентированное моделирование на C++. Учебный курс. Питер, 2006.