Владимир Руфицкий

ПРИНЦИПЫ РАЗРАБОТКИ МНОГОЗАДАЧНЫХ ПРОГРАММ ДЛЯ ОПЕРАЦИОННОЙ СРЕДЫ ОСРВ "КВАНТОС"

Операционная система реального времени с периодической активизацией задач (ОСРВ-ПА) для микроконтроллеров, реализованная на AVR-ассемблере, была рассмотрена в работе /1/. Она с успехом использовалась и продолжает использоваться в ряде проектов, обеспечивших создание надежных изделий специального назначения. В данной статье рассматриваются принципы разработки многозадачных программ, исполняемых в операционной среде СИ-версии ОСРВ-ПА. Данная версия операционной системы называется "КВАНТОС". ОСРВ "КВАНТОС" скомпонована в подключаемый СИ-файл, содержащий набор директив, используя которые разработчик может легко создавать многозадачные СИ-программы, обеспечивающие исполнение целевых функций разрабатываемых изделий. В статье подробно описана специфика используемых в ОСРВ-ПА понятий многозадачного программирования, таких как задача, объект контроля, дискретизация процессов работы задач, рассмотрены способы создания системы функциональных блоков, реализующих алгоритмы работы задач. Предложена аппаратная модель задачи, приведена технология создания исходного многозадачного СИ-файла.

Задачи и объекты контроля в ОСРВ-ПА

При разработке программ, исполняемых в операционной среде многозадачной операционной системы компьютера или микроконтроллера, используются различные базовые понятия – поток, задача, подзадача, процесс, приложение и даже подпрограмма /2, 3, 4/ . Функциональный смысл используемых понятий и терминов обычно уточняется при рассмотрении конкретных многозадачных операционных систем. Представляется необходимым сделать это и для ОСРВ-ПА.

В ОСРВ-ПА в качестве базового используется понятие задачи, как программного исполнителя, представляющего собой фрагмент многозадачной программы, обеспечивающий контроль одного или нескольких объектов контроля. Под контролем понимается комплекс действий, в который входят прием и обработка информации, формируемой объектом контроля или получаемой объектом контроля от внешнего источника, формирование сигналов, управляющих объектом контроля, передача объекту контроля необходимой информации.

Объекты контроля разделяются на две естественные категории – аппаратную и программную.

К аппаратным объектам контроля относятся внутренние и внешние периферийные устройства микроконтроллера – порты, таймеры, ЦАП и АЦП, индикаторы и т.д., способные, как правило, работать автономно, то есть вырабатывать новую информацию для контролирующих задач или воспринимать от задач и автономно обрабатывать управляющую информацию. Аппаратные объекты контроля взаимодействуют с задачами путем обмена значениями определенных в многозадачной программе информационных и (или) управляющих глобальных переменных (элементов данных), сохраняемых в регистрах аппаратного объекта контроля.

К объектам контроля программной категории относятся определенные в многозадачной программе и сохраняемые в памяти микроконтроллера элементы данных (глобальные переменные, указатели, массивы, структуры и объединения), которые совместно контролируются несколькими различными задачами. Это буферы, очереди, интерфейсы между задачами – флажки, семафоры и т.п. Далее они называются системными переменными, системными массивами или системными объектами контроля. Одна или несколько контролирующих задач могут получать (считывать) из системного объекта контроля новую информацию, сформированную и записанную в системный объект контроля другими контролирующими задачами. Из этого следует, что по отношению к любой контролирующей задаче, системный объект контроля, подобно аппаратному объекту контроля, обладает способностью автономной работы. Сама же способность автономной работы при таком подходе может быть определена как общая отличительная характеристика аппаратных и системных объектов контроля.

Необходимо отметить, что используемое в ОСРВ-ПА понятие "объект контроля", в отличие от абсолютного понятия "объект", используемого в языках ООП, является понятием относительным, связанным со своим субъектом – задачей, осуществляющей контроль объекта контроля.

В процессе автономной работы аппаратные объекты контроля могут взаимодействовать друг с другом согласно своему назначению и архитектуре создаваемого изделия. Кроме того, аппаратные и системные объекты контроля могут обмениваться информацией в соответствии с алгоритмами многозадачной программы. В ОСРВ-ПА эти взаимодействия не регламентируются.

Задачи (субъекты), как правило, взаимодействуют друг с другом путем обмена информацией через общие объекты контроля. Обеспечена также возможность взаимных запусков и остановов задач посредством директив ОСРВ-ПА, включаемых в текст многозадачной программы. Благодаря этому,

при необходимости, задачи в многозадачной программе могут подразделяться на управляющие (ведущие) и управляемые (подчиненные).

Базовые понятия задачи, как исполнителя, и объекта контроля, как предмета ответственности исполнителя или нескольких исполнителей, хорошо согласуются с общепринятой терминологией организации работ в коллективах исполнителей. В роли организаторов работ выступают сами разработчики многозадачных программ.

Для наглядного представления процессов работы задач как в среде ОСРВ-ПА, так и в среде большинства ОСРВ с вытесняющей и кооперативной многозадачностью, используется рассмотренная ниже аппаратная модель задачи .

Аппаратная модель задачи

Поскольку задача в ОСРВ-ПА определена как исполнитель, выполняющий комплекс математических и логических действий, ее модель достаточно просто реализуется аппаратно, подобно системной модели последовательностной логической схемы /5/. Предлагаемая аппаратная модель задачи представлена на рис.1

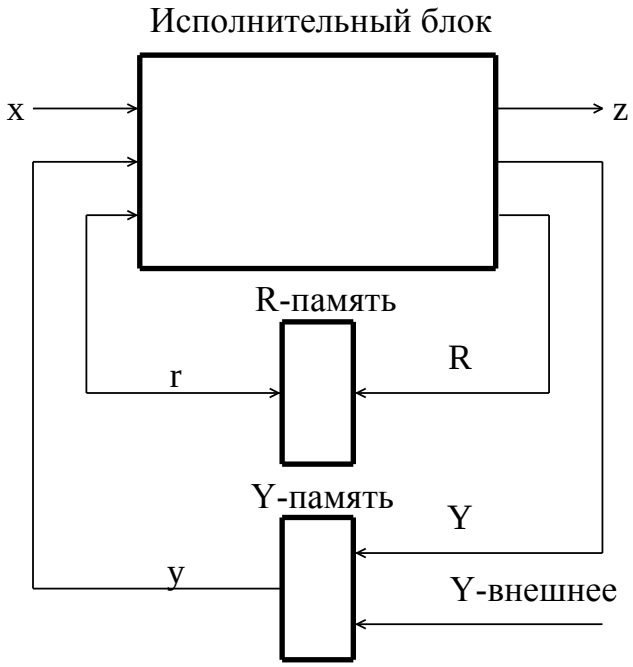


Рис.1 Аппаратная модель задачи.

В аппаратную модель задачи вместо комбинационной логики, являющейся исполнительным блоком модели последовательностной схемы, введен программно управляемый исполнительный блок, реализующий алгоритм работы задачи. Предполагается, что исполнительный блок содержит все необходимые аппаратные средства для исполнения последовательностей команд и подпрограмм ассемблера (АЛУ, рабочие регистры, аппаратный стек, устройство управления, память программы задачи), а также стек данных и регистр-указатель его вершины для исполнения СИ-функций.

Как известно, изменения входных сигналов последовательностных схем допускаются только по истечению временных интервалов, позволяющих комбинационной логике сформировать значения булевых функций, соответствующие фиксированным значениям входных сигналов. Эти временные интервалы определяют сменяющие друг друга рабочие циклы последовательностной схемы. Подобным же образом подвергаются дискретизации – представляются в виде последовательности рабочих циклов и процессы работы задач.

В многозадачной программе рабочие циклы задачи, в отличие от последовательностной схемы, состоят не из одной, а из двух фаз – активной и пассивной. В активной фазе исполнительный блок считывает из объектов контроля и обрабатывает значения входных СИ-переменных, которые определяют входное состояние задачи **x.** В результате обработки формируются и записываются в объекты контроля значения выходных СИ-переменных, которые определяют выходное состояние задачи **z.** В общем случае, в активной фазе рабочего цикла задачи, допускается многократное считывание значений входных переменных и формирование последовательности выходных состояний.

По завершению обработки входной информации, считанной в текущем рабочем цикле, и записи выходной информации, задача переходит из активной фазы в пассивную. При этом исполнительный блок формирует и сохраняет в Y-памяти значение переменной следующего внутреннего состояния **Y**. Сохраняемое значение является адресом команды, с исполнения которой должен начаться следующий рабочий цикл задачи. Следующее внутреннее состояние может формироваться не только исполнительным блоком задачи, но и другими задачами при взаимодействии задач. Формируемое другими задачами внутреннее состояние **Y- внешнее** всегда заменяет состояние **Y**, сформированное исполнительным блоком .

При обработке входной и формировании выходной информации исполнительный блок использует как в качестве операндов, так и для

возврата результатов из СИ-функций, контекстные СИ-переменные, условно размещенные в R-памяти модели задачи. Значения контекстных переменных определяют текущее контекстное состояние задачи **r** в активной фазе рабочего цикла. Значения контекстных переменных, сохраняемые в R-памяти для использования в следующем рабочем цикле задачи, определяют следующее контекстное состояние **R**. Следующее контекстное состояние **R** в наибольшей степени подходит и обычно используется для представления смысловой составляющей процесса работы задачи.

Переход задачи из активной фазы в пассивную фазу рабочего цикла завершается передачей управления ядру ОСРВ-ПА (далее активизатору задач) для начала следующего цикла другой задачи или той же самой задачи.

Помимо задач к числу программных исполнителей многозадачной программы относятся обработчики прерываний, функционирующие под управлением аппаратно-программного "механизма" обработки прерываний. Обработчики прерываний обеспечивают оперативную реакцию микроконтроллера на изменения состояний аппаратных объектов контроля.

По сути, обработчик прерываний это привилегированная задача, которая может автоматически вызываться из любой точки многозадачной программы при наступлении определенного события. Обработчик обычно захватывает процессор микроконтроллера на все время, которое требуется для выполнения возложенных на него функций. В ОСРВ-ПА обработчик прерывания, как и задача, всегда возвращает управление в точку вызова, поскольку завершается штатной командой возврата из прерывания.

При создании многозадачной программы для операционной среды ОСРВ-ПА, разработчик должен определить состав используемых задач, обработчиков прерываний и объектов контроля, а также взаимосвязи (взаимодействия) как между исполнителями, так и между объектами контроля, сформировав тем самым многозадачную систему, реализующую необходимые целевые функции изделия.

Периодическая активизация задач

Из приведенного выше описания аппаратной модели задачи следует, что дискретизация процессов работы задач – представление в виде последовательности рабочих циклов, обеспечивает не только детерминированную обработку входной информации, но и возможность организации псевдопараллельной работы задач . Дискретизация может осуществляться либо в явной, либо в неявной форме.

В традиционных ОСРВ с вытесняющей и кооперативной многозадачностью процессы работы задач подвергаются неявной дискретизации, так как моменты следующей активизации каждой задачи, а следовательно и полная длительность рабочего цикла, задаются неявно. Длительность пассивной фазы рабочего цикла в этих ОСРВ зависит от числа запущенных задач, установленных приоритетов задач, суммарной длительности активных фаз рабочих циклов запущенных задач и является в общем случае трудно предсказуемой. В ОСРВ с вытесняющей многозадачностью рабочие циклы задач отличаются еще и функциональной неопределенностью из-за принудительного завершения активной фазы рабочего цикла каждой задачи.

При использовании ОСРВ-ПА осуществляется явная дискретизация процессов работы задач посредством задания временных интервалов или периодов активизации задач, определяющих полную длительность каждого рабочего цикла задач. Единицей отсчета временных интервалов в ОСРВ-ПА является квант /2/.

Периоды и временные интервалы активизации устанавливаются для каждой задачи отдельно, и по длительности могут составлять от нескольких единиц до сотен квантов. Их выбор определяется скоростью работы объектов контроля и моментами взаимодействий задач многозадачной системы, которые задаются разработчиком проекта при организации взаимодействий между задачами. Период активизации задачи часто задается также на основании системных требований к целевым функциям изделия, например, требования к минимальному времени фиксации сообщения на индикаторе, которое может составлять несколько секунд. Скорости работы объектов контроля, моменты взаимодействий задач и системные требования фактически определяют реальную потребность объектов контроля в обслуживании, а соответствующих задач в активизации. В соответствии с этой потребностью и устанавливаются значения периодов и временных интервалов активизации задач.

Отсчет периодов и временных интервалов активизации задач в ОСРВ-ПА выполняет обработчик прерывания от одного из таймеров. Таймер, отсчитывающий длительность кванта, называется далее квантователем. Обработчик прерываний от квантователя работает независимо от других компонентов многозадачной системы и является общим для всех задач "часовым механизмом", синхронизирующим работу задач.

В рабочих циклах задачи длительность активной фазы, жестко не фиксируется, так как предполагается гарантированное завершение автономных действий задачи и добровольное освобождение процессора за

время существенно меньшее периода активизации. Благодаря высокому быстродействию современных микроконтроллеров не только исключается необходимость длительного захвата процессора, но обычно обеспечивается надежная реализация псевдопараллельной работы задач даже при совпадении моментов готовности к активизации нескольких задач.

Процесс исполнения 3-х конфликтующих задач поясняют временные диаграммы активизации задач **T1**, **T2**, **T3**, представленные на рис.2. На временных диаграммах "1" соответствует активной фазе рабочего цикла задачи, а "0" – пассивной фазе рабочего цикла. В том же масштабе времени на рис.2 изображены:

S0 – шкала отсчета периодов активизации, имеющая шаг t (квант);
S1 – шкала дискретизации процесса работы задачи 1, имеющая шаг 3t;
S2 – шкала дискретизации процесса работы задачи 2, имеющая шаг 6t;
S3 – шкала дискретизации процесса работы задачи 3, имеющая шаг 12t .

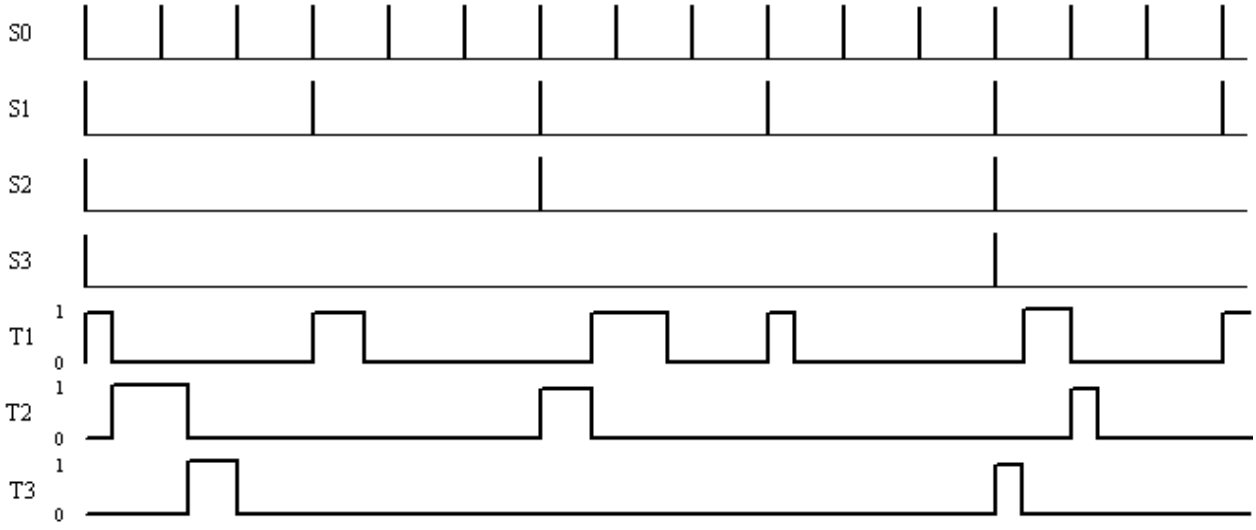


Рис.2 Временные диаграммы работы 3-х конфликтующих задач.

Опрос готовности задач к активизации в ОСРВ-ПА выполняется по кольцевому принципу, поэтому активизация конфликтующих задач может начинаться с любой задачи, но продолжается в порядке возрастания номера задачи (см. рис 2).

Длительность кванта в ОСРВ-ПА может составлять от сотен микросекунд до нескольких миллисекунд, в зависимости от необходимой точности отсчета периодов и временных интервалов активизации задач. За один такой квант современный микроконтроллер, работающий с частотой синхронизации 5-20 МГц, способен выполнить нескольких тысяч и даже

десятков тысяч ассемблерных команд программного кода. В реальных проектах такого числа команд оказывается более чем достаточно для выполнения всех функций рабочего цикла задачи.

Если длительности активных фаз рабочих циклов задач будут укладываться в длительность кванта, моменты начала рабочих циклов можно сдвигать с квантовой дискретностью, чтобы исключать возможные совпадения моментов готовности разных задач к активизации. Такое фазирование процессов работы задач может обеспечить бесконфликтную многозадачность в операционной среде ОСРВ-ПА.

Бесконфликтную работу задач поясняют временные диаграммы активизации задач **T1**, **T2**, **T3**, представленные на рис.3. В отличие от примера, приведенного на рис.2, задачи не конфликтуют друг с другом, так как периоды активизации задач сдвинуты на один квант.

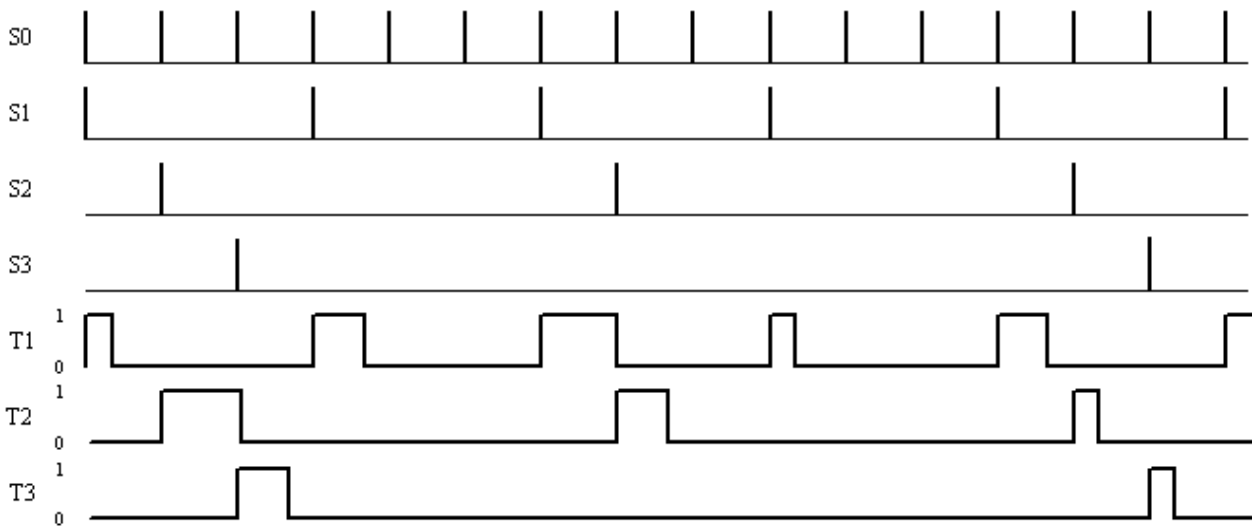


Рис.3 Бесконфликтная работа 3-х задач.

Оценка длительности активных фаз рабочих циклов задач и "укладывание" этой длительности в длительность кванта, а тем самым и в длительность периода активизации задачи, в определенной степени соответствуют принципам проектирования синхронных последовательностных схем и организации обмена информацией между регистрами цифровых устройств /6/. В связи с этим можно считать, что каждая задача работает в операционной среде ОСРВ-ПА по синхронному принципу . Сигналом синхронизации для каждой задачи является код завершения текущего периода активизации, формируемый обработчиком прерывания от квантователя в счетчике периода активизации задачи.

Следует уточнить, что по указанному сигналу синхронизации формируется лишь запрос к ОСРВ-ПА на перевод задачи в активную фазу. Этот запрос безусловно обслуживается в порядке установленной очередности. А вот переход задачи в пассивную фазу осуществляется асинхронно, по завершению исполнения функций текущего рабочего цикла задачи. Это позволяет не синхронизировать без необходимости процессы исполнения различных задач и последовательно обслуживать одновременные запросы нескольких задач с минимальными задержками (см.рис.2).

Периодическая активизация задач, обеспечивая явную дискретизацию задач, одновременно исключает необходимость задания и отдельной обработки приоритетов задач, поскольку длительность периода активизации в неявной форме определяет приоритет задачи /3/ . Возможность программного изменения периода активизации в процессе работы ОСРВ-ПА делает приоритет задачи динамическим, что позволяет организовывать надлежащий доступ нескольких задач к одному общему аппаратному объекту контроля .

Формирование системы функциональных блоков

При разработке многозадачных программ для операционной среды ОСРВ-ПА, в целях обеспечения функциональной определенности рабочих циклов задач, алгоритмы работы задач разделяются на функциональные блоки (ФБ). Каждый ФБ представляет один или несколько фрагментов алгоритмической схемы задачи, исполняемых на соответствующих рабочих циклах или периодах активизации задачи. Взаимосвязанные ФБ образуют систему ФБ.

При наличии алгоритмической схемы задачи система ФБ достаточно легко формируется эмпирически. Различные ФБ отделяются друг от друга начальными точками выполнения программного кода на следующем рабочем цикле (внутренними состояниями **Y**) и смысловыми результатами работы задачи (следующими контекстными состояниями **R**).

Система ФБ может быть создана и на основе диаграммы состояний, эквивалентной алгоритмической схеме задачи /6/. Диаграммы состояний могут быть составлены для достаточно простых задач, в которых каждый рабочий цикл характеризуется однократным считыванием значений входных переменных и однократным формированием значений выходных переменных. Чтобы сформировать систему ФБ по диаграмме состояний достаточно заменить каждую вершину диаграммы состояний на ФБ,

выполняющий функции фрагментов алгоритмической схемы, соответствующих всем дугам (переходам между состояниями), исходящим от этой вершины. При этом конфигурация дуг диаграммы состояний превращается в сеть связей между ФБ задачи. Допускается замена нескольких вершин на один "укрупненный" ФБ, если вершины не будут иметь взаимных дуг. Но в этом случае внутреннее состояние, идентифицирующее вершину, входящую в "укрупненный" ФБ, будет определяться не только именем этого ФБ, но и дополнительным идентификатором вершины внутри "укрупненного" ФБ, или идентификатором входа этого ФБ. Переменная, представляющая такой идентификатор, включается в состав контекста задачи и анализируется в начале исполнения "укрупненного" ФБ.

На рис.4 представлена диаграмма состояний и соответствующая ей система ФБ, в которой один ФБ соответствует одной вершине диаграммы состояний. Рис. 5 иллюстрирует замену двух вершин диаграммы состояний **Y1, Y2** на один ФБ – **FB1**.

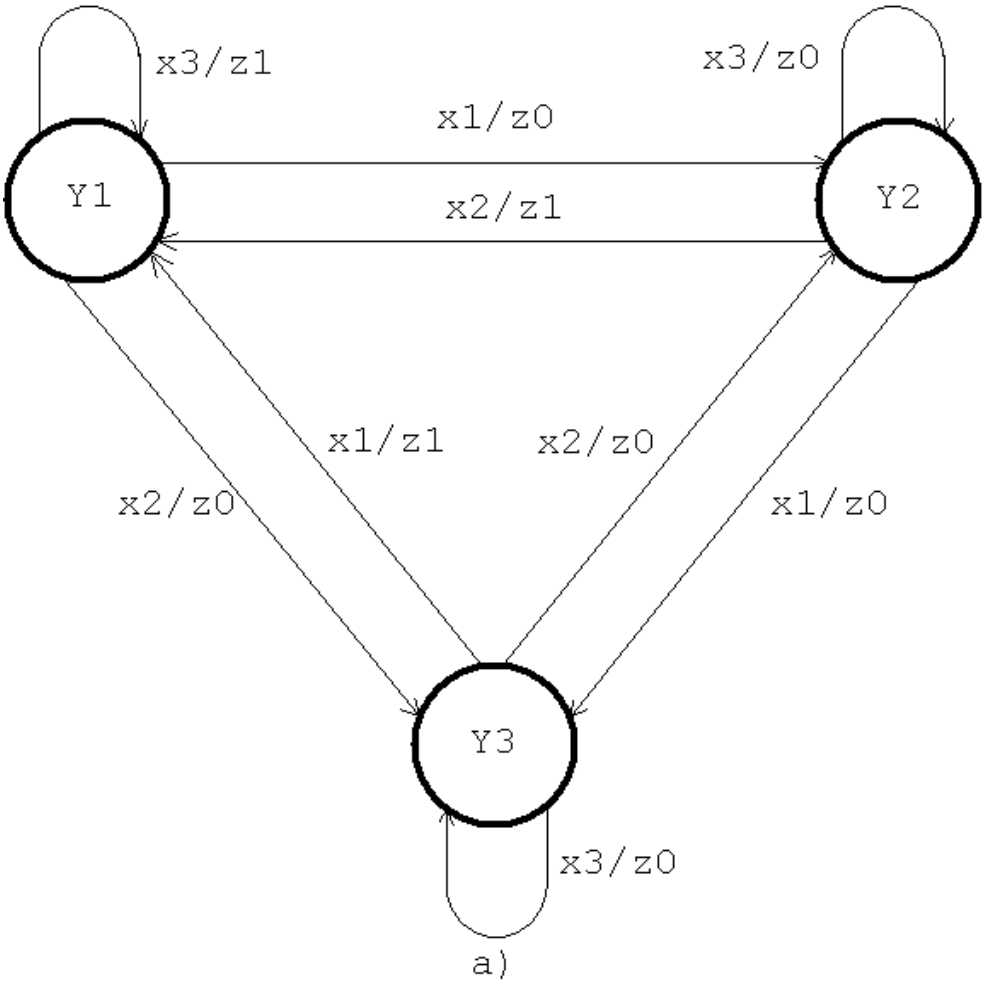
На рис.4 и 5 обозначены:

x1, x2, x3 – значения входного состояния задачи **x**;
z0, z1 – значения выходного состояния задачи **z** ;
Y1, Y2, Y3 – значения следующего внутреннего состояния задачи **Y**;
R1, R2 – значения следующего контекстного состояния задачи **R**;
FB1, FB2, FB3 – функциональные блоки;
FBi –> **FBj** – передачи управления от **FBi** (i=1, 2, 3) к **FBj** (j=1, 2, 3) для активизации **FBj** на следующем рабочем цикле задачи.

Завершая свою работу, каждый ФБ подготавливает активизацию следующего ФБ на следующем рабочем цикле задачи (передает управление следующему ФБ) и тем самым переводит задачу в следующее внутреннее состояние **Y**, которому соответствует одна из вершин диаграммы состояний.

Функциональная насыщенность каждого ФБ задачи должна быть такой, чтобы программный код ФБ микроконтроллер мог исполнить за минимально возможное время, соизмеримое с длительностью кванта. Для выполнения этого требования соответствующие дуги диаграммы состояний могут быть "укорочены" посредством добавления новых вершин, позволяющих сформировать вместо одного ФБ, группу более "коротких" ФБ. Данная операция соответствует разделению рабочего цикла задачи на несколько рабочих циклов с более короткими активными фазами. Например, если функции, реализуемые при переходах из вершины **Y3** в вершины **Y1**, **Y2** и **Y3**, диаграммы состояний, представленной на рис.5а, требуют больших

временных затрат, в соответствующие дуги этой диаграммы могут быть добавлены вершины **Y4, Y5, Y6,** как показано на рис.6.



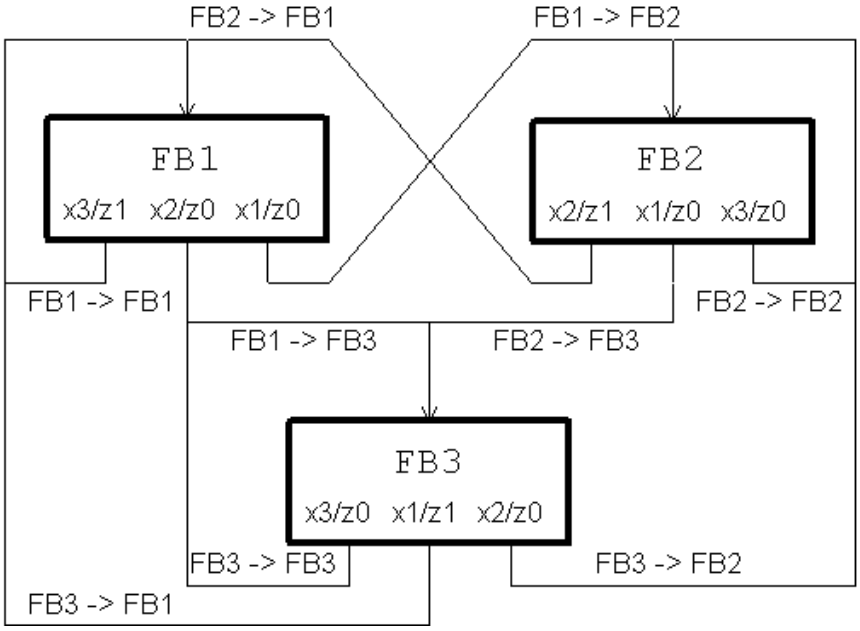


Рис.4 Формирование системы ФБ по диаграмме состояний задачи.

а) Диаграмма состояний задачи

б) Система ФБ, соответствующая диаграмме состояний.

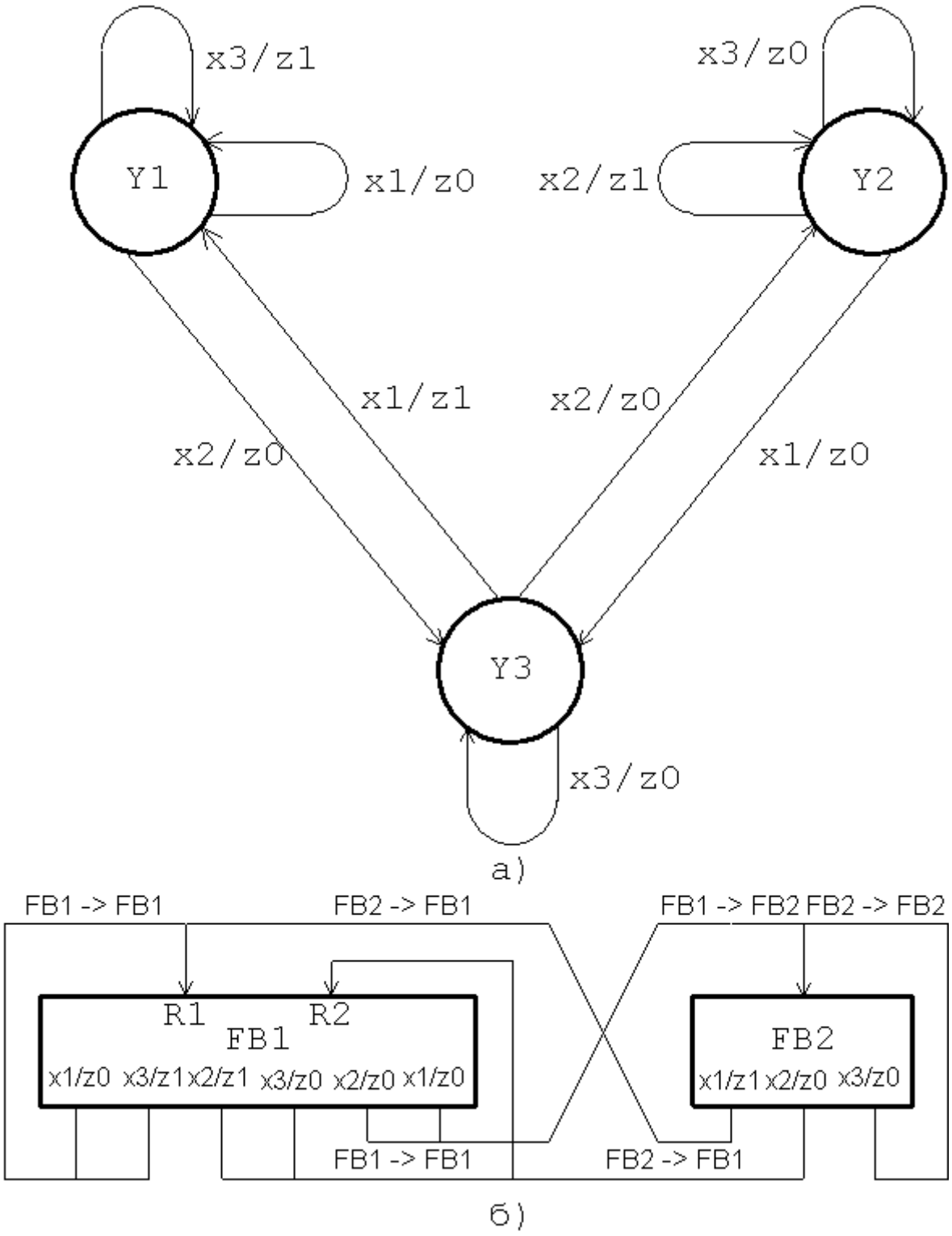


Рис.5 Замена двух вершин диаграммы состояний на один ФБ.

а) Диаграмма состояний задачи

б) Система ФБ, соответствующая диаграмме состояний.

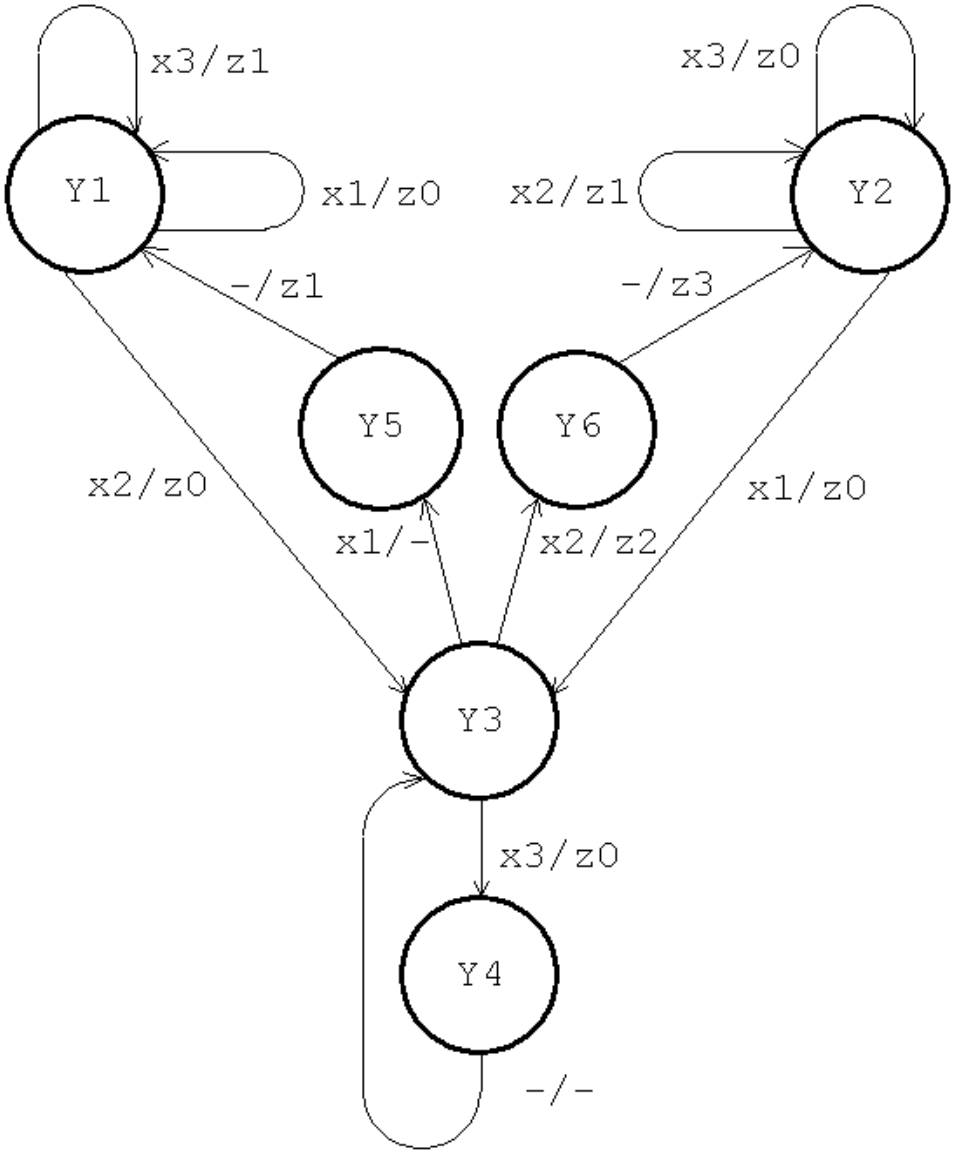


Рис.6 Добавление новых вершин в диаграмму состояний задачи

Обозначения новых переходов на диаграмме рис.6 показывают, что при разделении рабочего цикла задачи на несколько рабочих циклов запись значений выходных переменных задачи и внутренняя обработка информации (работа с контекстом) могут произвольно распределяться между образованными рабочими циклами. При этом не исключается и добавление новых значений выходных состояний **z2, z3**. Но считывание значений входных переменных должно выполняться в первом же рабочем цикле, поскольку в пассивной фазе первого рабочего цикла задачи значения входных переменных могут измениться и исказить результаты работы задачи.

В связи с привязкой ФБ к текущему **r** и следующему **R** контекстным состояниям задачи, ФБ в принципе не создается в качестве библиотечного

программного модуля. ФБ – это лишь специализированная часть задачи, освобождающая процессор микроконтроллера при завершении исполнения своего программного кода. Целесообразно создавать библиотеки исполнителей – задач и обработчиков прерываний, способных работать с конкретными объектами контроля и контекстными переменными, образующими соответствующие контексты.

Технология создания многозадачного СИ – файла

Многозадачная программа формируется с помощью директив ОСРВ – ПА, которые определены в двух взаимозаменяемых формах – функция и макрос. Если директивы часто повторяются в многозадачной программе, использование функциональной формы директив, позволяет существенно сокращать объем программной памяти, необходимый для директив ОСРВ – ПА. Макрос уменьшает время исполнения директив, так как не связан с обращением к стеку данных и сохранением содержимого регистров, но при каждом вызове компилируется в отдельную область программной памяти .

Задачи в ОСРВ "КВАНТОС" идентифицируются числами от 0 до 255, а ФБ задач – произвольными именами, разрешенными в СИ.

В начале СИ – файла многозадачной программы должна быть задана константа MAX_TASKS , определяющая число используемых задач N=1…256:

```
#define       MAX_TASKS      N
```

После задания MAX_TASKS подключается файл QUANTOS.c , содержащий исходный код ОСРВ-ПА "КВАНТОС":

```
#include      " QUANTOS.c"
```

В области объявления глобальных переменных объявляются контексты всех используемых задач и обработчиков прерываний – необходимые глобальные переменные, массивы, структуры. Далее определяются системные объекты контроля, используемые в проекте, – буферы, очереди, интерфейсы и т.д.

В файле QUANTOS.c объявлены два главных системных массива:
- массив периодов и счетчиков активизации задач (типа **unsigned char**);
- массив функциональных указателей на точки исполнения программ различных задач (типа **void**).

К первому из указанных массивов имеют доступ активизатор задач, обработчик прерывания от квантователя, а также задачи, и при необходимости, обработчики прерываний. Элементы данного массива

сгруппированы парами "период – счетчик", в порядке возрастания номеров задач. Изменением значений этих элементов обеспечиваются останов, возобновление и запуск задач, подготовка задачи к активизации на следующем рабочем цикле. Содержимое этого массива полностью определяет временные параметры процессов исполнения задач.

В массиве функциональных указателей в порядке возрастания номеров задач сгруппированы системные переменные, определяющие внутренние состояния задач. Обращение к каждой из этих переменных осуществляется через порядковый номер задачи. При желании порядковые номера задач могут быть заменены именами задач путем объявления перечисления (enum). Значения переменных в данном массиве представляют адреса ФБ, исполняемых на следующих рабочих циклах задач. Эти адреса идентифицируются именами ФБ. Активизатор задач осуществляет автоматическую активизацию ФБ задач, не анализируя и не изменяя значений внутренних состояний задач. В тоже время, задача или обработчик прерываний, управляющие другими задачами, с помощью директив ОСРВ – ПА, могут записывать в этот системный массив значения указателей, адресующих следующие исполняемые ФБ управляемых задач.

Вслед за глобальными переменными либо определяются все ФБ задач, задействованных в текущей версии программы, либо объявляются прототипы ФБ, с последующим определением ФБ по тексту программы.

В СИ-программе ФБ реализуется в форме функции типа **void** без аргументов. Возврат результата в рабочие регистры микроконтроллера не требуется, поскольку ФБ не используется в качестве части выражений СИ – программы. Результаты работы ФБ возвращаются в контекстные переменные и в выходные переменные задачи. Наличие у ФБ аргументов (формальных параметров) не требуется, поскольку в ФБ используются значения объявленных и проинициализированных глобальных переменных, определяющих текущие состояния **x** и **r**. При необходимости, внутри ФБ могут использоваться СИ-функции с параметрами, заданными в форме указателей, адресующих соответствующие глобальные переменные состояний **x** и **r.**

В этой же области программы определяются все используемые обработчики прерываний, в том числе и обработчик прерываний от квантователя. В обработчике прерываний от квантователя необходимо задать директиву отсчета периодов:

__QUANTIZER__INT

Эта директива обеспечивает корректировку (декремент) значений счетчиков активизации в системном массиве периодов и счетчиков активизации задач.

В главной функции **void** main(**void**) после начальной настройки и запуска аппаратных компонентов микроконтроллера инициализируются системные объекты контроля и контексты задач. Объявленные в файле QUANTOS.c массивы и переменные инициализируются директивой:

__OS__RESET

Далее каждая задействованная задача запускается директивой:

__RUN__TASK(task, FB, period), где

task – номер задачи (от 0 до 255).;

FB – имя следующего активизируемого ФБ задачи;

period - период активизации задачи в квантах (от 0 до 254, значение 255 соответствует останову задачи).

При большом числе задач можно использовать для этой директивы вместо макроса функциональную форму:

RUN__TASK(task, FB, period);

Если требуется обеспечить бесконфликтную работу задач, следует задать моменты начала исполнения задач (сфазировать задачи) директивами

__DELAY__TASK(task, delay), где

delay – задержка активизации задачи в квантах (от 0 до 254, значение 255 соответствует останову задачи).

Главная функция должна быть завершена циклической директивой активизации задач, запускающей активизатор задач:

__TASKS__ACTUATION

В начале кода этой директивы на несколько тактов разрешаются прерывания. При их отсутствии, проверяется готовность к активизации одной из задач по значению её счетчика активизации. Если значение счетчика активизации равно нулю, активизируется соответствующий ФБ задачи. В начало кода директивы активизации задач каждый ФБ возвращает управление по своему завершению, используя одну из директив выхода из ФБ.

В таблице 1 приведен перечень директив ОСРВ "КВАНТОС" в двух взаимозаменяемых формах – макроса и функции.

Указанные в таблице 1 директивы запуска задач, установки задержки активизации задач, а также директивы установки нового периода активизации задач, останова и возобновления задач, могут задаваться в любом ФБ любой задачи и в любом обработчике прерываний. Это дает

большую свободу для организации взаимодействий между исполнителями многозадачной программы.

Таблица 1 Перечень директив ОСРВ "КВАНТОС"

Директива ОСРВ-ПА	**Макрос, функция**	**Параметры**
Инициализация ОСРВ-ПА	__OS__RESET OS__RESET()	Отсутствуют
Активизация задач	__TASK__ACTUATION TASK__ACTUATION()	
Отсчет периодов	__QUANTIZER__INT QUANTIZER__INT()	
Запуск задачи	__RUN__TASK(task,FB,period) RUN__TASK(task,FB,period)	task – номер задачи (0…255); FB – имя следующего активизируемого ФБ задачи; period – период активизации задачи в квантах (0…254); delay – задержка активизации задачи в квантах (0…254).
Выход из ФБ	__NEXT__FB(FB) NEXT__FB(FB)	
Выход из ФБ с самоостановом	__SNEXT__FB(FB) SNEXT__FB(FB)	
Выход из ФБ с задержкой	__DNEXT__FB(FB,delay) DNEXT__FB(FB,delay)	
Выход из ФБ с новым периодом	__RNEXT__FB(FB,period) RNEXT__FB(FB,period)	
Установка нового периода	__PERIOD__TASK(task,period) PERIOD__TASK(task,period)	
Останов задачи	__SUSPEND__TASK(task) SUSPEND__TASK(task)	
Возобновление задачи	__RESUME__TASK(task) RESUME__TASK(task)	
Возобновление задачи с задержкой	__DRESUME__TASK(task,delay) DRESUME__TASK(task,delay)	
Установка задержки	__DELAY__TASK(task,delay) DELAY__TASK(task,delay)	

Файл QUANTOS.c , содержащий исходный код ОСРВ "КВАНТОС", а также демонстрационный файл 8-TASKS_DEMO.c выложены в открытом доступе на сайте автора статьи **vladruf.ru**.

Заключение

Рассмотренная в статье операционная система реального времени "КВАНТОС" дополняет традиционные "инструменты" систем реального времени, такие как прерывания и опрос, простой и удобной возможностью задания периодов и временных интервалов активизации задач через директивы ОСРВ. Этим обеспечивается явная дискретизация процессов работы задач согласно реальной потребности объектов контроля в обслуживании, а соответствующих задач в активизации.

Явная дискретизация процессов работы задач, совместно с представлением алгоритмических схем задач в виде взаимосвязанных функциональных блоков, исполняемых в пределах коротких активных фаз рабочих циклов задач, позволяет создавать многозадачные программы, в которых процессы работы задач характеризуются полной функциональной определенностью и предсказуемостью. Эти характеристики, по мнению автора, соответствуют самой сути систем реального времени и обеспечивают безусловную реализуемость и высокую надежность создаваемых изделий.

В связи с доступностью исходных кодов ОСРВ "КВАНТОС" состав директив ОСРВ может расширяться разработчиками по их усмотрению. Например, могут быть добавлены директивы захвата общих ресурсов (объектов контроля) или директивы выхода из ФБ с большим тайм-аутом, и т.д. Не исключается также возможность "усовершенствования" директивы активизации задач в части создания очереди задач, готовых к активизации.

Для повышения эффективности работы с большим числом задач, возможно аппаратное объединение квантователя и системного массива периодов и счетчиков активизации задач в отдельном периферийном устройстве, реализованном на сверхмалом микроконтроллере или интегрально с основным микроконтроллером. Это позволит исключить "технологический" захват процессора обработчиком прерываний от квантователя для опроса и декремента счетчиков активизации задач.

В заключение необходимо отметить, что рассмотренная операционная система "органично" вписывается и в концепцию ООП, поскольку задача может быть определена как объект, компонентными функциями которого являются функциональные блоки, а компонентными данными – переменные, определяющие контекст задачи. Если проект позволяет эффективно использовать группы однотипных исполнителей – задач, то преимущества языков ООП, в частности СИ++, становятся очевидными, поскольку компилятор ООП не тиражирует компонентные функции в программной

памяти микроконтроллера. Кроме того, отпадает необходимость многократного присвоения в программе однотипных имен контекстных переменных и функциональных блоков. Но в случае использования в проекте специфических, разнотипных задач, язык СИ является вполне достаточным для реализации целевых функций создаваемого изделия.

Литература

1. Владимир Руфицкий, Операционная система реального времени для микроконтроллеров, Chip News №7, 2008 г.
2. Сергей Сорокин, Системы реального времени, СТА №2, 1997 г.
3. Полное руководство по Windows 95 Питера Нортона, перевод с англ., Москва, БИНОМ, 1998 г.
4. Jean J. Jabrosse, MicroC/OS-II, The Real-Time Kernel,(ISBN 1-57820-103-9).
5. Wen C. Lin, Microprocessor-Based Digital System Design Fundamentals and the Development Laboratory for Hardware Designers and Engineering Executives, Proceedings of the IEEE, August 1977, pp. 1138-1161.
6. Руфицкий В.Н., Стаханов А.И, Способы построения однофазной безусловно реализуемой системы синхронизации, Вопросы радиоэлектроники, серия Электронная вычислительная техника, 1974, вып.1.