ISBN 5-7163-0089-8

Системы на микроконтроллерах и БИС программируемой логики

СОВРЕМЕННАЯ МИКРОПРОЦЕССОРНАЯ ТЕХНИКА

Бродин В.Б., Калинин А.В.

Системы на микроконтроллерах и БИС программируемой логики

При поддержке фирмы «Точка Опоры»

- Архитектура MCS–51 и микроконтроллеры Atmel AT89
- Микроконтроллеры MCS–51 с модернизированными структурами
- Микроконвертер ADuC812 фирмы Analog Devices
- Микроконтроллеры Atmel с архитектурой AVR
- Проектирование цифровых устройств на основе ППИС
- Язык AlteraHDL
- Технология проектирования систем на микроконтроллерах и ПЛИС
- Учебные практикумы для студентов и специалистов
- Инструментальные средства и методы разработки

ЭКОМ

Бродин В.Б., Калинин А.В.

Системы на микроконтроллерах и БИС программируемой логики

Издательство
ЭКОМ
Москва, 2002

ББК 32.97
Б88
УДК 681.3

Б 88 **Бродин В.Б., Калинин А.В.**

Системы на микроконтроллерах и БИС программируемой логики — М.: Издательство ЭКОМ, 2002. — 400 с.: илл.

ISBN 5-7163-0089-8

Рассматривается технология проектирования микроэлектронных систем на основе микроконтроллеров и БИС программируемой логики (ПЛИС). Описаны «ядра» MCS-51 и AVR современных микроконтроллеров, микроконвертер ADμC812 фирмы Analog Devices, семейства ПЛИС на примере изделий фирмы Altera. Рассмотрено проектирование цифровых устройств на основе ПЛИС с использованием языка AHDL. Описаны различные средства программирования и отладки микропроцессорных контролеров. Книга включает многочисленные примеры схем и фрагменты программ, которые могут быть использованы в практической работе.

Одновременно с вопросами проектирования обсуждаются проблемы обучения студентов и повышения квалификации специалистов, возникающие в связи с быстрым изменением элементной базы. В книге описаны учебные практикумы, построенные по принципу «делай как я» и позволяющие быстро передать практические навыки разработки систем на основе рассматриваемых микроконтроллеров и ПЛИС. Особенностью продемонстрированного подхода к обучению является использование только профессиональных инструментальных средств и методов, что исключает необходимость дальнейшего переучивания.

Книга для широкого круга специалистов в области проектирования микроэлектронных систем, аппаратуры и программного обеспечения. Материал соответствует ряду учебных курсов, необходим преподавателям и студентам.

Текст печатается в авторской редакции.

ББК 32.97

ISBN 5-7163-0089-8

СОДЕРЖАНИЕ

ОТ АВТОРОВ

Современный этап развития микропроцессорных систем управления характеризуется комбинированным применением микроконтроллеров и БИС программируемой логики (ПЛИС). Это позволяет значительно улучшить характеристики систем на основе известных и освоенных разработчиками микроконтроллерных архитектур за счет дополнения их специализированными блоками, реализованными с учетом особенностей объектов управления. Значительно ускоряют проектирование и конструирование полноцикловые системы автоматизации проектирования типа EDA (Electronic Design Automation), пришедшие на смену традиционным системам типа CAD (Computer Aided Design), которые помогали разработчику на отдельных этапах, например при проектировании печатных плат.

В области микроконтроллеров, на наш взгляд, наиболее широкий круг задач покрывают семейства с архитектурой MCS-51 (архитектура фирмы Intel) и AVR фирмы Atmel (название этой фирмы является аббревиатурой от Advanced Technology for Memory and Logic).

Архитектура MCS-51 получила в последнее время новый импульс развития с появлением таких приборов, как 89C51 фирм Atmel и Philips, ADuC812 фирмы Analog Devices, W78 фирмы Winbond, AN21 фирмы Cypress, P-51 фирмы Cybernetic Micro Systems и ряда других. В них на одном кристалле с ядром MCS-51 объединены flash-память объемом до 64 Кбайт, 12-разрядные АЦП и ЦАП, интерфейсы USB, CAN и (E)ISA. Это дает возможность разработчикам использовать при решении новых задач большой имеющийся задел. Дополнительные возможности предоставляют версии микроконтроллеров MCS-51 с пониженным до 1,8 В напряжением питания, а также приборы со сжатой тактовой сеткой и повышенной (до 40 МГц) тактовой частотой.

Микроконтроллеры семейства AVR фирмы Atmel, появившиеся на мировом рынке в 1997г., имеют современную RISC-архитектуру, которая в сочетании с технологией flash-памяти обеспечивает очень хорошие показатели по таким критериям, как скорость выполнения кода программы, эффективность генерации кода при использовании языков высокого уровня (поддержка ЯВУ системой команд), низкая цена.

Важным преимуществом некоторых микроконтроллеров AVR является их совместимость по функциям выводов с микроконтроллерами архи-

тектуры MCS-51. Это позволяет во многих случаях увеличить производительность имеющейся системы управления посредством замены микроконтроллера, разработки и отладки рабочей программы.

Одним из наиболее известных производителей ПЛИС является фирма Altera. Ее продукция стала одним из стандартов «де-факто», микросхемы и САПР отличаются высокими характеристиками, весьма скромны требования к инструментальному компьютеру разработчика. Особенностью проектирования на ПЛИС является, в настоящее время, широкое использование высокоуровневых языков описания аппаратуры AHDL, VHDL.

Несмотря на публикации в периодической печати, издание нескольких монографий [1-3], ощущается значительный неудовлетворенный спрос на информацию об элементной базе, методах и инструментах для разработки реальных современных систем управления на микроконтроллерах и ПЛИС. Эта книга создавалась как методическое и информационное ядро комплекса средств проектирования и обучения, разработанных в лаборатории «Микропроцессорные системы» МИФИ. В ней мы изложили свое понимание современного этапа развития микропроцессорных систем управления, методов их проектирования, рассмотрели характеристики современной элементной базы в части 8-разрядных микроконтроллеров и ПЛИС, описали наши инструментальные и учебные средства.

Одновременно с вопросами проектирования обсуждаются проблемы обучения и повышения квалификации специалистов, возникающие в связи с быстрым изменением элементной базы. Концептуально с книгой связаны учебные практикумы, построенные по принципу «делай как я» и позволяющие быстро передать практические навыки разработки систем управления на основе рассматриваемых микроконтроллеров и ПЛИС. Главы книги включают примеры схем и фрагменты программ, которые могут быть использованы при практической работе, часть материала взята из упомянутых практикумов. Особенностью нашего подхода является то, что в практической и учебной работе мы используем только профессиональные инструментальные средства и методы, так что после разбора примеров в дальнейшем не требуется переучивания.

Первая глава посвящена анализу современного состояния микропроцессорных систем управления, методов их проектирования с использованием механизма квазипараллельных процессов и разделения функций между микроконтроллерами и ПЛИС.

Во второй главе описывается архитектура MCS-51, ставшая одним из стандартов «де-факто» на мировом рынке микроконтроллеров. Приводится обзорная информация по микроконтроллерам серии AT89 фирмы Atmel.

В третьей главе рассмотрен бесстселлер мирового рынка от фирмы Analog Devices – микроконвертер ADμC812, представляющий собой однокристальную систему сбора данных и управления, реализованную в виде комбинации микроконтроллерного ядра MCS-51 с прецизионным 12-

разрядным АЦП и двумя 12-разрядными ЦАП. Подобный прибор ждали и он сразу же привлек большой интерес отечественных разработчиков.

В четвертой главе книги на примере микроконтроллера ATmega103 рассмотрена фирменная RISC-архитектура AVR компании Atmel.

Пятая глава посвящена вопросам проектирования цифровых устройств на ПЛИС. Рассмотрены: характеристики семейств ПЛИС фирмы Altera, способы описания проекта в графической форме и на языке AHDL, обработка проекта системой автоматизированного проектирования MAX+plusII, синтаксические конструкции языка AHDL.

В шестой главе описаны инструментальные и учебные средства, разработанные в лаборатории «Микропроцессорные системы» МИФИ.

Книга написана сотрудниками лаборатории «Микропроцессорные системы» МИФИ, имеющими опыт проектирования микропроцессорных систем космических приборов, сложных технологических установок и серийно выпускаемых медицинских аппаратов. Отражен опыт обучения студентов и специалистов промышленности.

Книга адресована широкому кругу специалистов в области проектирования информационных и управляющих систем, микроэлектронной аппаратуры, программного обеспечения. Изложенный материал соответствует планам ряда учебных курсов, необходим преподавателям и студентам. Мы считаем, что эту книгу целесообразно будет иметь библиотекам технических ВУЗов.

Благодарности:

Заведующий кафедрой микроэлектроники МИФИ профессор В.С.Першенков тщательно поддерживает доброжелательную и творческую атмосферу, которая издавна существует в коллективе. Ему и всем коллегам выражаем признательность.

Профессор А.В.Шальнов, многие годы ректор МИФИ и заведующий кафедрой микроэлектроники, первым проделал практикум «Проектирование цифровых устройств на ПЛИС с использованием системы MAX+plusII». Выражаем ему признательность за всегда внимательное и конструктивное обсуждение всех проблем, желаем здоровья.

Профессор И.И.Шагурин в 1985г. основал отраслевую лабораторию министерства электронной промышенности СССР «Микропроцессорные системы» МИФИ. Свидетельствуем ему свое уважение и благодарим за многолетнюю совместную работу.

Аспирант Л.Хохлов является одним из соавторов практикума «Проектирование цифровых устройств на ПЛИС с использованием системы MAX+plusII». В пятой главе книги параграф «Проект АЛУ RISC-микроконтроллера» написан по материалам его работы. Выражаем ему свою благодарность и желаем успехов.

Авторы считают приятной обязанностью выразить признательность Т.Ю.Макаровой за помощь в подготовке книги.

ГЛАВА 1

РАЗРАБОТКА СИСТЕМ НА МИКРОКОНТРОЛЛЕРАХ И БИС ПРОГРАММИРУЕМОЙ ЛОГИКИ

1.1. Предпосылки нового подхода к проектированию

До последнего времени в отечественной практике микроконтроллеры использовались для создания систем управления, а БИС программируемой логики (ПЛИС) служили элементной базой вычислителей и других специализированных устройств.

Системы управления на основе микропроцессоров и микроконтроллеров быстро получили массовое применение, поскольку были сформулированы ясные правила их проектирования. Создание аппаратной компоненты велось на основе магистрально-модульной структуры, прикладные программы вначале были несложными и разрабатывались известными методами. В качестве метода комплексной отладки аппаратуры и программного обеспечения фирмой Intel уже в середине 70-х годов был предложен метод внутрисхемной эмуляции. Идея совместной работы нескольких микропроцессорных БИС на одну магистраль в области систем управления не получила развития. При необходимости увеличить производительность системы разработчики шли по пути увеличения разрядности – переходили от 8- к 16- или 32-разрядной микропроцессорной БИС. До середины 90-х годов увеличение разрядности означало, кроме прямого результата, переход в

другой класс качества микропроцессорных БИС. «Серьезные» 16- и 32-разрядные микропроцессорные БИС имели передовые архитектурные и структурные решения, которых в дешевых 8-разрядных изделиях быть не могло. Заметным явлением в конце 80-х годов стало использование во вновь разрабатываемых 8- и 16-разрядных системах управления преимущественно микроконтроллеров. Микропроцессоры соответствующей разрядности продолжают выпускаться для поддержки серийных изделий. Основной областью применения микропроцессоров стала 32-разрядная обработка данных в компьютерах, сигнальных и коммуникационных процессорах.

Первоначально проектированием устройств на ПЛИС, из которых в нашей стране использовались преимущественно микросхемы фирмы Xilinx, занимался узкий круг высококвалифицированных специалистов, которые, по существу используя традиционный «вентильный» подход, создавали уникальные устройства. Проектирование велось и ведется на уровне графического ввода схемы устройства и ручной трассировки межсоединений ячеек, характеризуется большими сроками выполнения проектов, поскольку возможности БИС существенно превышают возможности системы проектирования в плане автоматизированного синтеза. По существу, это традиционное проектирование, но на основе современных БИС и с привлечением САПР, поддерживающих разработчика на отдельных этапах создания проекта.

Во второй половине 90-х годов ситуация существенно изменилась. Прежде всего, изменилась топология микропроцессорных систем управления. В прежней магистрально-модульной структуре схемы обслуживания отдельных узлов и оконечных устройств были драйверами без интеллекта. Они выполняли функции преобразования протоколов, служили усилителями мощности, но не имели возможности обрабатывать свою информацию. Функцию обработки выполнял ведущий микроконтроллер, к которому от периферийных схем тянулись жгуты проводов. В литературе [4] приводятся данные, что автомобили на этом этапе содержали до трех миль проводов весом более 90 кг. Быстрое удешевление микроконтроллеров привело к целесообразности замены ими некоторых периферийных схем. Появилась возможность предобработки локальных данных и пересылки ведущему микроконтроллеру только результатов, да и то в случае необходимости. Это привело к широкому внедрению существовавших последовательных интерфейсов (RS-232, I^2C, SPI) и разработке новых (CAN, MicroLan). Топология развитых систем управления приобрела характер сети локальных ведомых микроконтроллеров, связанных между собой и с ведущим микроконтроллером через последовательные интерфейсы. В настоящее время используются последовательные каналы как радиального, так и магистрального типов, синхронные и асинхронные.

Значительное улучшение характеристик собственно микроконтроллеров, достигнутое в это время, связано с внедрением RISC-архитектур, flash-памяти программ и EEPROM-памяти данных, прецизионных блоков АЦП и ЦАП. Гарвардская архитектура микроконтроллеров (с раздельными матрицами памяти программ и данных) позволила оптимизировать формат команд и обеспечить выборку и выполнение большинства из них за один машинный такт. Производительность, например, микроконтроллеров AVR фирмы Atmel достигла значения 10 MIPS на частоте 10 МГц.

Технология flash-памяти обеспечила резкое снижение стоимости микроконтроллеров с перезаписываемой памятью программ, позволив отказаться от металлокерамических корпусов с кварцевым стеклом, которые были необходимы для памяти с ультрафиолетовым стиранием. Важным аспектом для надежности и мобильности систем стало появление в структуре микроконтроллеров энергонезависимой памяти данных.

Совершенствование микроэлектронной технологии позволило фирме Analog Devices объединить на одном кристалле ядро микроконтроллеров MCS-51 с прецизионными 12-разрядными модулями АЦП и ЦАП. Микроконвертер ADμ812, по существу, является первой измерительно-управляющей микропроцессорной системой на кристалле.

Важно, что передовые структурные решения в настоящее время применяются в классе 8-разрядных микроконтроллеров. Именно здесь рост степени интеграции направляется на реализацию новых функциональных модулей и структурных решений.

Качественные изменения в области применения БИС программируемой логики наступили, когда в качестве средств описания проектов стали применяться языки высокого уровня типа HDL (Hardware Description Language). В это время ведущие мировые производители ПЛИС путем постепенного согласованного усложнения элементной базы и средств проектирования решили задачу автоматического синтеза устройства на основе текстового описания. Таким образом, появилась возможность значительно сократить сроки разработки проектов на ПЛИС и сделать процесс проектирования доступным широкому кругу инженеров. В настоящее время ПЛИС имеют степень интеграции до нескольких миллионов эквивалентных вентилей, а их быстродействие (ввода/вывода) достигло рубежа 400 МГц. Из наиболее известных производителей ПЛИС следует отметить фирму Altera. Ее микросхемы и системы проектирования дают возможность быстро реализовать нужную функцию на кристалле, используя автоматическую компиляцию графического или текстового описания проекта. Качество и стоимость такого метода проектирования вполне соответствуют требованиям интеграции разрабатываемой схемы в структуру системы управления на основе микроконтроллеров.

Микроконтроллеры и ПЛИС настолько удачно дополняют друг друга, что несколько фирм приступили к выпуску БИС, интегрирующих на од-

ном кристалле и в одном корпусе как микроконтроллер, так и матрицу программируемой логики. В качестве примера можно привести семейство БИС серии E5 фирмы Triscend. Старшая модель семейства объединяет на кристалле ядро микроконтроллера 8051, ОЗУ объемом до 64 Кбайт и матрицу программируемой логики объемом до 3200 ячеек.

Другим примером может служить семейство FPSLIC фирмы Atmel. БИС этого семейства включают ядро RISC-микроконтроллера AVR, статическую память SRAM и матрицу FPGA типа AT40K. Память разделена на ОЗУ программ, память данных и память общего назначения. Поскольку микроконтроллер выбирает команды из быстродействующего ОЗУ, его производительность достигает значения 40 MIPS на частоте 40 МГц. Кроме того, к ядру AVR-микроконтроллера добавлен 8-разрядный умножитель. С конфигурационной EEPROM AVR-микроконтроллер общается через интерфейс I^2C. Матрица FPGA программируется таким образом, что разработанное устройство доступно в общем адресном пространстве микроконтроллера.

1.2. Технология разработки микропроцессорных контроллеров

Технология проектирования контроллеров на основе микропроцессоров и микроконтроллеров полностью соответствует *концепции неразрывности процесса проектирования и отладки аппаратной и программной составляющих*, принятой во всей микропроцессорной технике. Единый процесс проектирования микропроцессорной системы и ее отладки в англоязычной литературе обозначается словом *development*, мы будем употреблять термин *разработка*. Важной особенностью применения контроллеров является работа в реальном масштабе времени, т.е. гарантированная реакция на внешние события в течение определенного интервала времени. Очевидно, что решение задачи комплексной разработки аппаратуры и программного обеспечения *в реальном масштабе времени* при произвольной структуре и схемотехнике контроллера является весьма сложной, дорогостоящей и долговременной работой.

В качестве основного метода разработки микропроцессорных систем фирмой Intel в 70-х годах был предложен *метод внутрисхемной эмуляции*. Основой этого метода является моделирование разрабатываемой системы с использованием средств специализированного инструментального компьютера – схемного эмулятора. В соответствии с первоначальной идеей схемный эмулятор, представляющий из себя по сути конструктор, должен был иметь все аппаратные средства, которые могли понадобиться для реализации целевой системы, плюс средства управления отладкой. Вначале контроллер конфигурировался из аппаратуры эмулятора и разрабатыва-

лась управляющая программа, далее на пустую макетную плату устанавливалась розетка целевого микропроцессора и схемы ближайшего обрамления, эмулятор своей эмуляционной вилкой включался в розетку вместо микропроцессора и начинался поэтапный перенос аппаратных средств с соответствующими программными фрагментами из эмулятора на плату контролера. Чтобы обеспечить такой перенос, адресное пространство памяти картируется с образованием сегментов, которые физически могут находиться как в эмуляторе, так и на целевой плате.

Основным преимуществом метода внутрисхемной эмуляции является упорядоченность процесса проектирования, который может быть разделен на ряд отдельных этапов. На каждом этапе все ошибки локализованы во вновь создаваемом аппаратном и программном обеспечении, фундаментом являются заведомо работоспособные средства эмулятора. Основным недостатком такого глобального подхода к эмуляции является дороговизна инструментальных средств. Это усугублялось в первое время тем, что до появления персональных ЭВМ фирма Intel пошла по пути создания отладочных комплексов типа Intellec в виде собственной специализированной мини-ЭВМ с оригинальной операционной системой ISIS.

В настоящее время средства моделирования в схемных эмуляторах, которые мы называем *имитирующим процессором*, в большинстве случаев замещают только целевую микропроцессорную БИС и память. Разработка остальных частей контроллера, который в настоящее время часто представляет собой сложную систему управления, ложится на разработчика. В лаборатории «Микропроцессорные системы» МИФИ при проектировании и отладке микропроцессорных контроллеров используется методика типовых функционально-топологических и программных (ФТП) модулей. Термин *функционально-топологический и программный модуль* в данном случае подразумевает, что аппаратура выполняет законченную типовую функцию, реализована в виде устоявшейся совокупности компонент (микросхем, пассивных элементов и т.п.) на печатной плате с определенными технологическими нормами, процедура управления аппаратурой представляет собой программный модуль. Топология является неотъемлемой составляющей модуля, поскольку характеристики современной прецизионной элементной базы могут быть реализованы только при правильном конструировании печатной платы. Таким образом, после определения схемотехнических решений не рекомендуется использовать автоматическое размещение элементов и трассировку, а после получения хороших результатов эксплуатации нельзя произвольно менять топологию функционального модуля.

Предпосылкой к использованию ФТП модулей является принцип магистрально-модульного построения систем на основе микропроцессоров и микроконтроллеров (рис. 1.1).

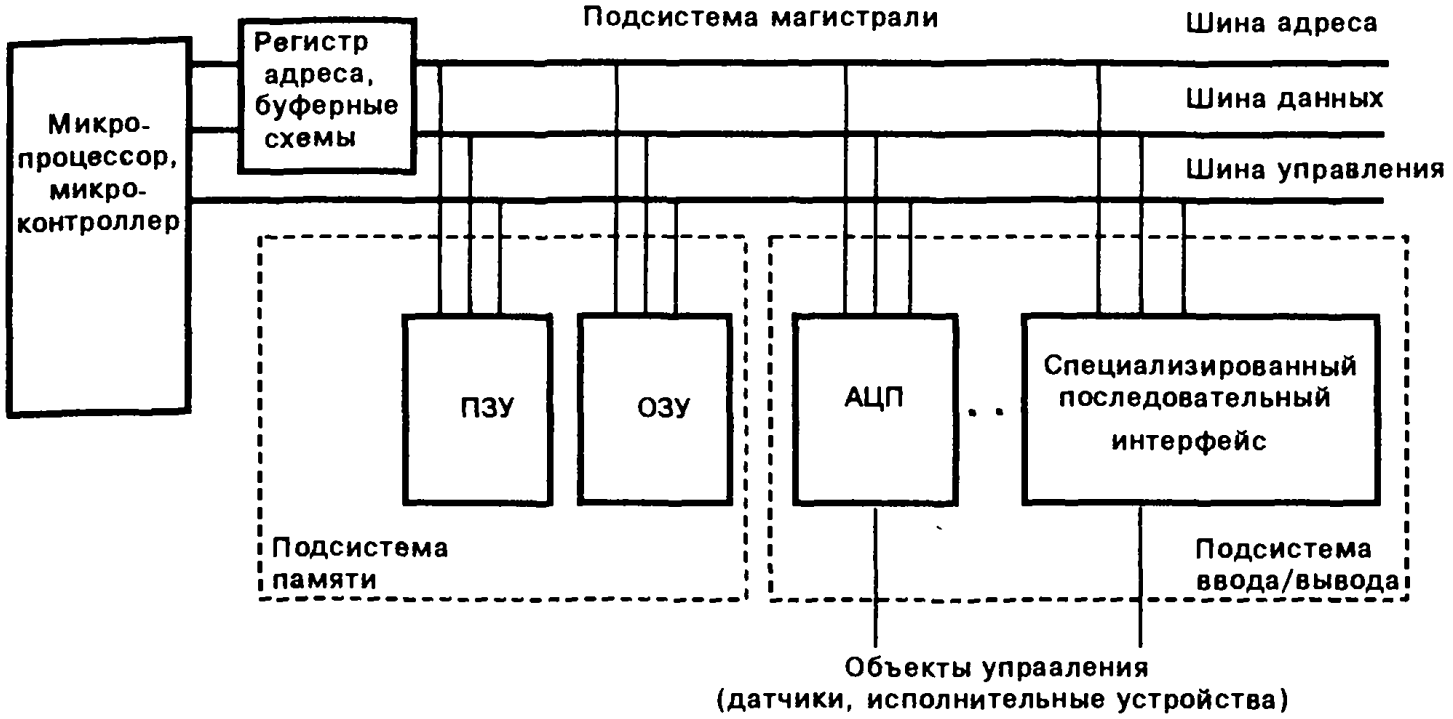

Рис. 1.1. Магистрально-модульная организация микропроцессорных систем

Магистрально-модульная организация подразумевает, что память и интерфейс ввода/вывода выполнены в виде модулей, которые обмениваются информацией с микропроцессорной БИС через магистраль. Обработку данных осуществляет микропроцессорная БИС, как правило и обмен данными ведется под ее управлением и через ее внутренние регистры. На логическом уровне микропроцессорная система представляется в виде регистровой модели. Обращение микропроцессорной БИС к ячейкам памяти и регистрам модулей ввода/вывода осуществляется по адресам, которые сводятся разработчиком в карту памяти и устройств ввода/вывода. Параметры магистрали (разрядность шин адреса, данных и управления, протокол и диаграммы обмена) определяются либо каким-либо официальным стандартом, либо аналогичными параметрами микропроцессорной БИС, которые являются в этом случае стандартом «де-факто». Параметры магистрали во многом определяют предельные характеристики производительности микропроцессорной системы.

Проектирование микропроцессорных контроллеров с использованием ФТМ модулей осуществляется в два этапа. На первом этапе на макетной плате создается и испытывается сам модуль, разрабатывается программное обеспечение для его обслуживания. При этом макетная плата проектируется с полноценной топологией печатных проводников, обеспечивающей согласование по волновому сопротивлению и экранирование от наводок, но с дополнительными элементами, обеспечивающими отладку. Модуль отлаживается во взаимодействии с работоспособным процессорным модулем, с которым он в дальнейшем будет объединяться на плате. На втором этапе на основе процессорного ядра и отработанных ФТП модулей обрамления

разрабатывается заказной контроллер. Очевидно, что набор ФТП модулей в значительной степени привязан к тому процессорному ядру, с которым был отлажен.

Особенностью микропроцессорных контроллеров является то, что они сами интегрируются в некоторый объект (embedded controllers). Это предполагает, что перед разработчиком микропроцессорной системы такого рода стоит задача полного цикла проектирования, начиная от разработки алгоритма функционирования и заканчивая комплексными испытаниями в составе изделия, а возможно и сопровождением при производстве. Основные этапы цикла разработки микропроцессорного контроллера отображены на рис. 1.2.

Технические требования начинают цикл проектирования микропроцессорного контроллера. Особенностью именно микропроцессорных контроллеров является то, что возможности их программирования подвигают заказчика заложить максимально широкие функции управления, чтобы иметь возможность использовать контроллер для управления целой гаммой аналогичных приборов. Критерием выбора должна служить экономическая целесообразность любого увеличения объема аппаратных средств, что определяется в результате исследования рынка приборов данного типа, и максимальное улучшение показателя цена/функциональные_возможности. На этом этапе явно или неявно формулируются требования к типу используемого микропроцессора или микроконтроллера.

Этап разработки алгоритма управления является наиболее ответственным, поскольку ошибки этого этапа обнаруживаются при испытаниях законченного изделия и приводят к дорогостоящей переработке всей системы управления. Прорабатывается несколько вариантов алгоритма, обеспечивающих выполнение технических требований с использованием наработанных ранее функционально-топологических модулей. Основные варианты отличаются соотношением объема программного обеспечения и аппаратуры. Критерием выбора является максимальное увеличение программы и уменьшение аппаратуры при обеспечении заданных показателей быстродействия и надежности в полном диапазоне эксплуатационных воздействий. Часто определяющим требованием является возможность размещения кода управляющей программы во внутренней памяти микроконтроллера, что позволяет обеспечить ее защиту. На этом этапе окончательно определяется тип микропроцессорной БИС и важнейших схем обрамления (flash-памяти, ПЛИС, программируемых интерфейсов, АЦП и т.п.).

На этапе разработки структуры микропроцессорного контроллера окончательно определяется состав имеющихся и подлежащих разработке аппаратных модулей, протоколы обмена между модулями, типы разъемов. Поскольку контроллер встраивается в изделие, выполняется предварительная проработка конструкции плат. В части программного обеспечения

определяется состав и связи программных модулей, язык программирования. На этом же этапе производится выбор средств проектирования и отладки.

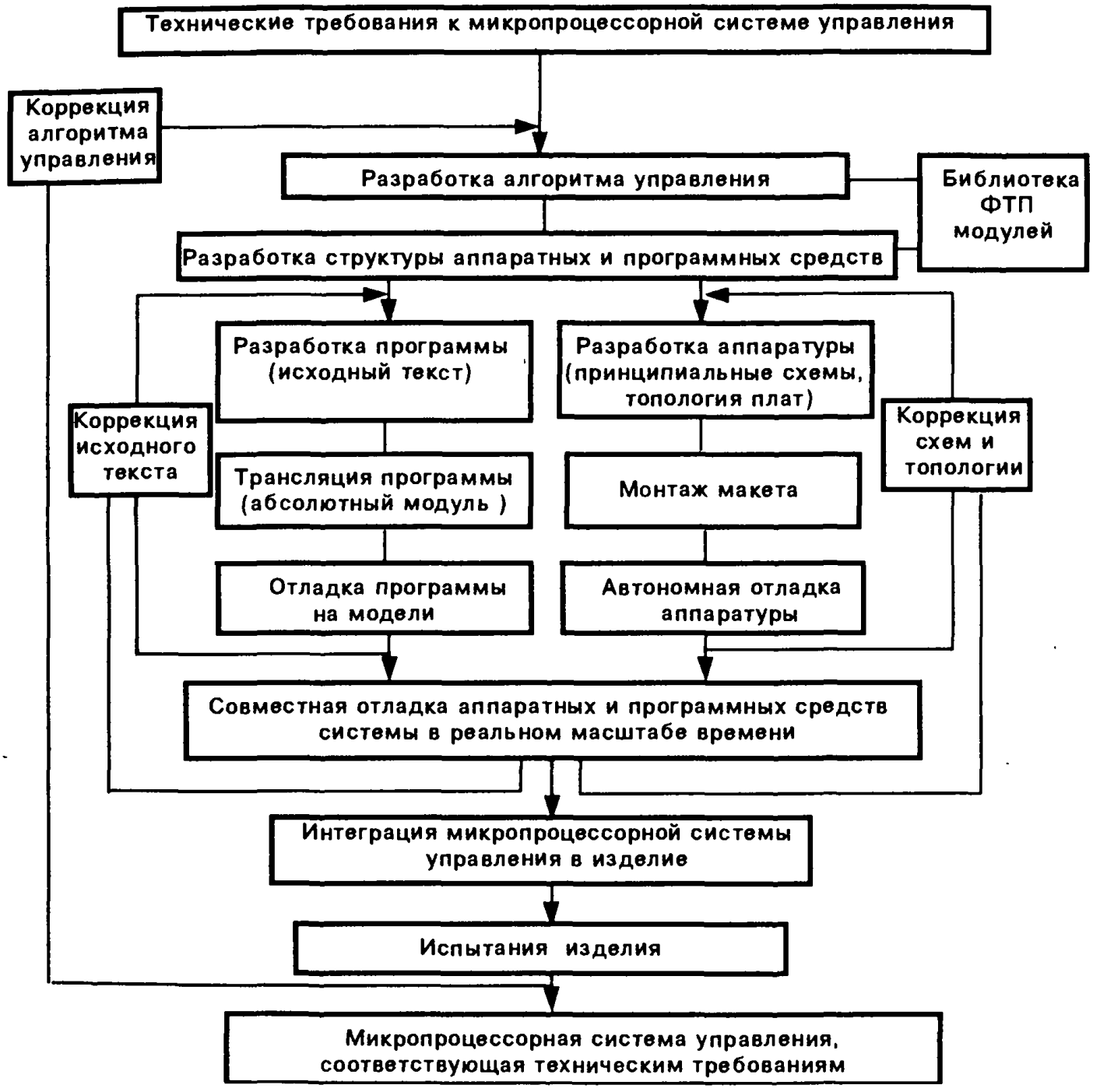

Рис. 1.2. Основные этапы разработки микропроцессорного контроллера

Содержание **этапов разработки исходного текста программы, трансляции и отладки** логических связей на модели существенно зависит от используемых системных средств. В настоящее время ресурсы 8-разрядных микроконтроллеров достаточны для поддержки программирования на языках высокого уровня. Это позволяет использовать все преимущества структурного программирования, разрабатывать программное обеспечение как проект с использованием раздельно транслируемых модулей. Одно-

временно продолжают широко использоваться языки уровня ассемблера, особенно при необходимости обеспечить контролируемые интервалы времени. Задачи предобработки данных часто требуют использования вычислений с плавающей точкой, трансцендентных функций. В настоящее время самым мощным средством разработки программного обеспечения для контроллеров являются интегрированные кросс-системы программирования на языках высокого уровня типа Паскаль, Си. Система Паскаль-51, например, включает редактор текста, компилятор с редактором связей, библиотеку стандартных функций периода выполнения и символический отладчик. Такие системы позволяют резко сократить затраты времени на создание и коррекцию программного обеспечения, что весьма важно, поскольку на рис. 1.2 видно, что эти этапы составляют внутренний, наиболее часто повторяющийся цикл в последовательности этапов разработки микропроцессорной системы.

Другой внутренний цикл, выполняемый параллельно, составляют **этапы создания аппаратуры:** разработка общей прнципиальной схемы и разводка топологии плат, монтаж макета и его автономная отладка. Эти этапы можно считать завершенными после того, как «оживает» магистраль микропроцессорной системы и через нее можно обратиться к памяти и блокам ввода/вывода. Время выполнения этих этапов зависит от имеющегося набора опробованных функционально-топологических модулей и квалификации разработчика. Распространенными системами проектирования, используемыми на этапе ввода принципиальной схемы и разработки топологии являются ACCEL EDA и OrCad. Эффективность их использования значительно зависит от имеющегося у разработчика объема библиотек используемых элементов.

Этап совместной отладки аппаратуры и программного обеспечения в реальном масштабе времени является самым трудоемким и обязательно требует использования таких высокопроизводительных средств (development tools), как схемный эмулятор, эмулятор ПЗУ, логический анализатор и генератор программируемых последовательностей. Выбор одного из перечисленных средств обусловлен используемым методом отладки. Этап завершается, когда аппаратура и программное обеспечение совместно обеспечивают выполнение всех шагов алгоритма работы системы. В конце этапа код программы управления «зашивается» с помощью программатора в энергонезависимую память и проверяется работа контроллера без участия эмулятора. Отладка на этом этапе ведется в лабораторных условия с питанием от источника, обеспечивающего максимальную защиту аппаратуры. Часть внешних источников информации может моделироваться.

Этап интеграции контроллера в изделие заключается в повторении работ по совместной отладке аппаратуры и управляющей программы, но

при работе в собственном отсеке изделия, питании от штатного источника, с информацией от штатных устройств и датчиков. Осложнения, как правило, возникают из-за электромагнитной несовместимости исполнительных устройств, разработанных ранее, с микропроцессорной системой управления. Много времени на этом этапе уходит на ликвидацию одиночных сбоев. Эту проблему можно решить с помощью программного резервирования, но только при наличии резерва памяти программ. На этом же этапе проводится и калибровка прибора с занесением параметров во flash-память.

Испытания изделия с микропроцессорным контроллером можно разделить на комплексные и специальные. Особенностью комплексных испытаний является то, что для наблюдением за микропроцессорным контроллеров в реальных условиях не всегда применимы лабораторные средства отладки. Автономные отладочные средства менее развиты и при этом существенно дороже. Специальные испытания (на электромагнитную совместимость, климатические и т.п.) проводятся по обычным методикам. После успешного проведения испытаний появляется файл с окончательной версией кода управляющей программы для программатора или для завода-изготовителя микроконтроллеров, который осуществляет масочное программирование внутренней памяти программ.

1.3. Квазипараллельные процессы в микропроцессорных системах управления

Задачи управления в микропроцессорных системах решаются как программно, так и аппаратно. Программную часть реализации функций управления осуществляет модуль центрального процессора, аппаратную – специализированные интерфейсные модули. Эти модули: последовательный порт, контроллер прямого доступа к памяти и другие, – выполняют свои операции преобразования данных после получения команд и данных от процессора, работают параллельно с ним.

В традиционной системе на основе комплекта микропроцессорных БИС (рис. 1.1) результатом выполнения микропроцессором фрагмента программы являются данные на магистрали и модифицированное содержимое регистров и ячеек памяти данных. Ввод и вывод данных через буферы шины данных представляет собой элементарную операцию, привязанную к тактовой сетке процессора, поэтому можно говорить о чисто программной реализации функций управления. События при этом отражаются сигналами на линиях магистрали. Чисто программное управление, таким образом, реализуется там, где сигналы на линиях магистрали могут быть использованы непосредственно для управления без дальнейшего аппарат-

ного логического преобразования, например при работе с другим микропроцессором.

Часть типовых операций управления, необходимых для связи с реальными объектами, реализованы в традиционной системе аппаратно в виде специализированных интерфейсных БИС. Эти операции не являются тривиальными, поскольку реализуются сложными самостоятельно функционирующими цифровыми автоматами, преобразующими данные. Такие автоматы, как последовательный порт, тактируются некоторым синхросигналом и проходят в процессе работы ряд состояний. Они самостоятельно воспринимают внешние события (сигналы) и обрабатывают их, привязка к сетке процессора производится по прерываниям. То же касается и формирования выходных сигналов управления. При отсутствии интерфейсной БИС с нужной операцией прежде приходилось создавать требуемый автомат из схем малой степени интеграции.

Очевидно, что в общем случае микропроцессорная система управляет объектами, используя комбинированную аппаратно-программную реализацию функций управления.

Микроконтроллер отличается от микропроцессора тем, что аппаратные операции, наиболее часто используемые в микропроцессорных системах управления, выполняются внутренними модулями, интегрированными на кристалл вместе с процессорным ядром. Это, кроме памяти программ и данных, таймеры/счетчики, последовательные порты, модули АЦП и ЦАП, модули управления электродвигателями и т.п.

Модуль процессора в микроконтроллере способен выполнять с разделением времени несколько функций управления. При этом функции управления реализуются как процессы на временной сетке с некоторым элементарным интервалом времени ΔT_N, и можно говорить о *квазипараллельных процессах*.

Организацию и взаимодействие квазипараллельных процессов управления рассмотрим на примере работы ведущего микроконтроллера системы управления рентгенотелевизионного аппарата. Связи платы ведущего микроконтроллера с панелью управления и другими объектами видны из рис. 1.3.

При включении аппарата процедура инициализации ведущего микроконтроллера устанавливает режим автоматического просвечивания, на программируемое устройство питания рентгеновского излучателя (РИ) подаются логические сигналы управления и аналоговые сигналы уставок анодного напряжения КС и анодного тока МС. Формируются также сигналы для блока диафрагм, усилителя рентгеновского изображения (УРИ) и системы обработки изображений (СОИ). Аппарат переходит в состояние готовности текущего режима, тип и параметры режима могут изменяться с панели управления, но высокое анодное напряжение на РИ подается толь-

ко при нажатии педали или ручного включателя. Включение высокого напряжения сопровождается формированием логических сигналов в соответствии с определенной циклограммой.

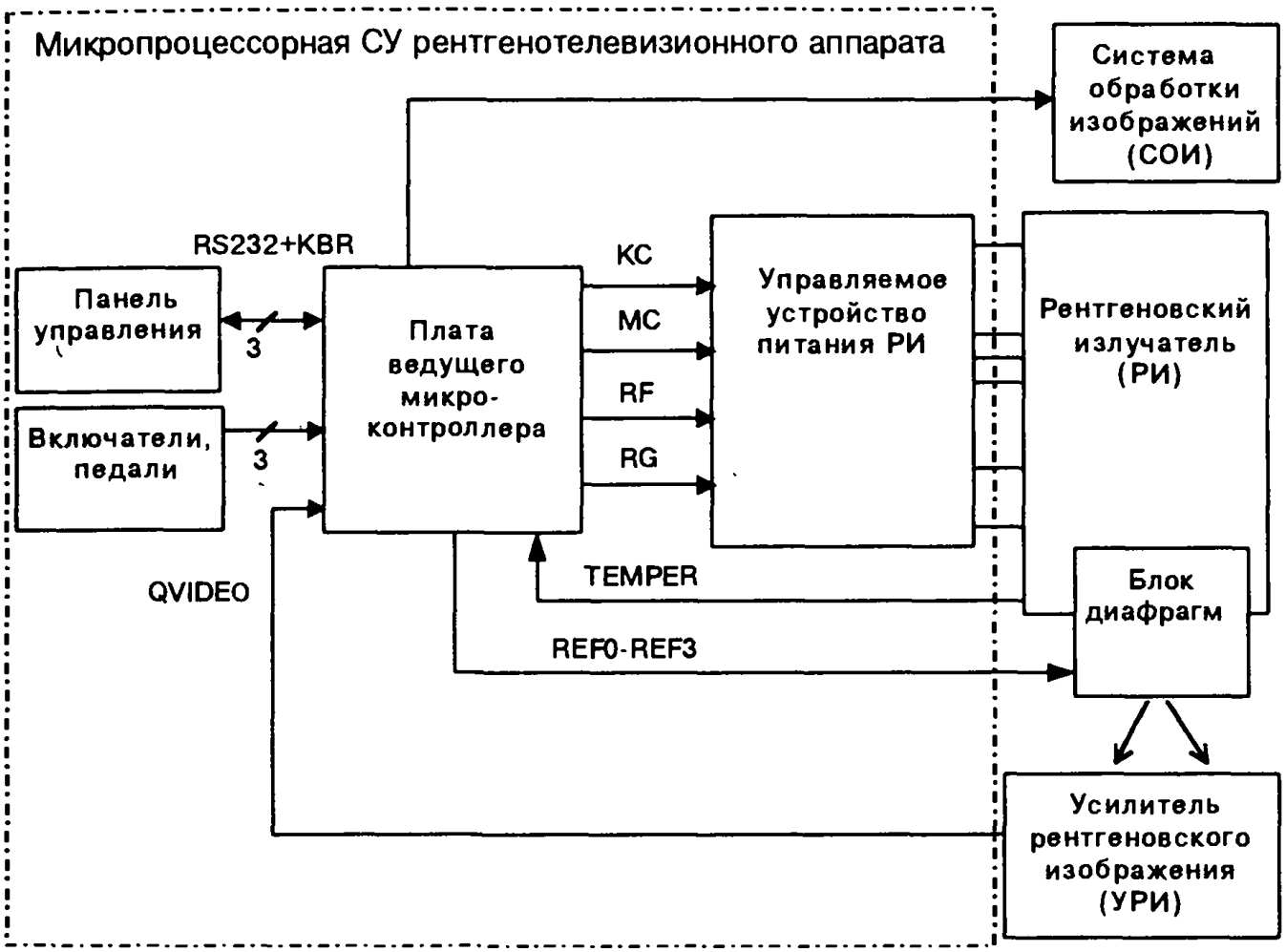

Рис. 1.3. Микропроцессорная СУ рентгенотелевизионного аппарата и связи с объектами управления

При включенном РИ ведущий микроконтроллер обслуживает процесс автоматического регулирования мощности дозы. При этом он получает цифровой сигнал обратной связи, частота которого зависит от величины рассогласования аналогового сигнала ОС относительно опорного значения, и формирует новое значение кода уставки, из которого далее образуется аналоговый сигнал КС. Каждый импульс цифрового сигнала ОС вызывает прерывание, процедура которого инкрементирует или декрементирует внутренний счетчик микроконтроллера, являющийся источником кода уставки. Между прерываниями анализируется посылка от панели управления на предмет изменения раскрыва диафрагм. Отпускание педали или включателя приводит к снятию высокого напряжения с РИ, аппарат вновь переходит в состояние готовности текущего режима. Система обеспечения безопасности функционирования формирует общий сигнал неготовности и код ошибки. Сигнал неготовности приводит к снятию высокого напряжения с РИ и выдаче кода ошибки на панель оператора.

Из вышеизложенного можно заключить, что в сложных системах управления ведущий микроконтроллер должен обеспечить реализацию следующих процессов:

- формирование временной сетки с заданным элементарным интервалом;
- обслуживание подсистемы безопасности;
- формирование циклограмм сигналов управления объектами;
- регулирование параметров непрерывных процессов в аппарате на основе анализа сигналов обратных связей.
- взаимодействие с оператором через панель управления и отдельные включатели/педали, переключение режимов работы.

Организация параллельных процессов управления на микроконтроллерах требует предварительного (статического) планирования их выполнения: расчета времени, механизмов взаимодействия и необходимых ресурсов. Рассмотрим временную диаграмму реализации перечисленных выше процессов в СУ рентгенотелевизионного аппарата, представленную на рис. 1.4.

Каждый процесс в СУ на микроконтроллерах реализуется как последовательность операторов на сетке системного времени. Операторы выполняются с использованием процедур. Процедуры могут прерывать друг друга, но операторы определенных процессов должны быть выполнены к определенному времени.

Рассмотрим каждый из перечисленных выше процессов. При этом мы не будем подробнее, чем это сделано в комментариях, анализировать текст программы. Нашей целью является анализ структуры программы и определение времени выполнения операторов каждого из процессов. Текст фрагментов программы служит для демонстрации того, что в развитой системе программная реализация функций управления требует значительного времени и ресурсов из-за необходимости реализовать взаимодействие квазипараллельных процессов.

Формирование временной сетки. Для взаимодействия процессов вначале необходимо определить величину элементарного интервала времени и сформировать временную сетку. Сетка формируется с использованием выделенного аппаратного модуля микроконтроллера - таймера/счетчика, работающего в режиме счета внутренних импульсов синхронизации. Таймер при переполнении формирует запрос прерывания, процедура обслуживания которого вызывает его перезагрузку.

Прерывание таймера системного времени должно иметь наивысший приоритет. Поскольку процесс формирования циклограмм сигналов управления использует интервалы системного времени и это является основой программного управления, временная сетка должна быть непрерывной и равномерной $\Delta T_N = T_{N+1} - T_N = const = \Delta T_{системы}$. Важной задачей является определение длительности элементарного интервала системного

времени. Слишком длительный интервал вызывает недозагрузку ведущего микроконтроллера, ухудшение параметра «реального времени». Слишком короткий интервал может привести к нарушению циклограмм сигналов управления и аварийной ситуации.

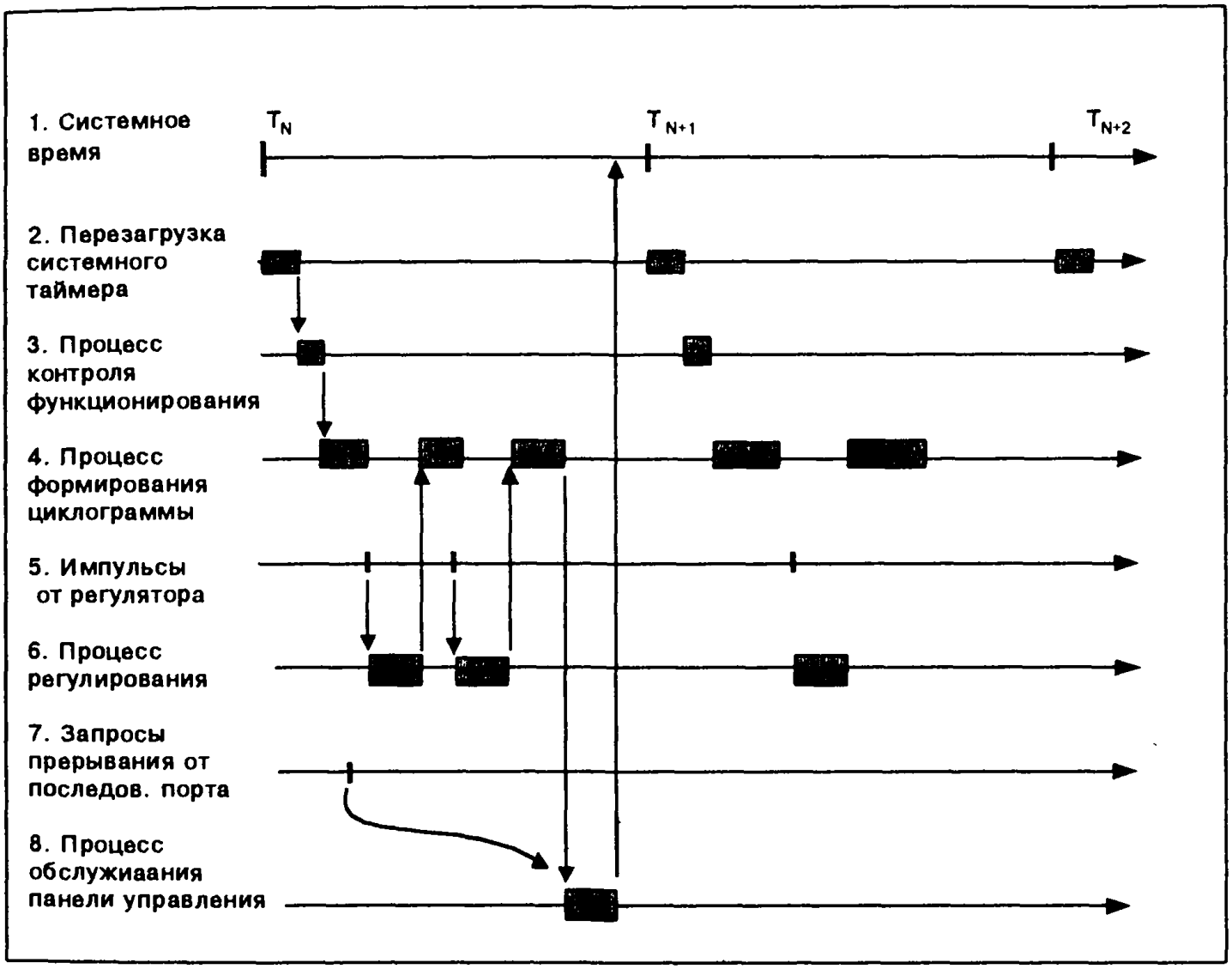

Рис. 1.4. Диаграмма квазипараллельных процессов в системе управления

Если в качестве ведущего используется микроконтроллер семейства MCS-51, то таймер (например T0) считает машинные циклы ($T_ц = 12/F_{осц}$). Перезагрузка таймера системного времени производится процедурой в две команды по два цикла, еще два цикла тратится на вход в процедуру. Следовательно, перезагрузка выполняется за 6 машинных циклов. Чтобы не накапливалась ошибка, это время должно быть учтено коррекцией начального значения, например следующим образом:

```
IntT0:
        mov THO,  #High (TORLD+6)  ; перезагрузка ст. байта T0
        mov TLO,  #Low (TORLD+6)   ; перезагрузка мл. байта T0
```

Таким образом, процесс формирования временной сетки включает один оператор перезагрузки таймера, который выполняется после завершения текущего интервала сетки. Нас интересует длительность выполнения этого оператора, представляющая собой сумму машинных циклов его команд ΣQ_i . Как указано выше, команды занимают 6 машинных циклов микроконтроллера. При тактовой частоте $F_{осц}$ = 12 МГц оператор формирования временной сетки нашего примера выполняется за $\Delta t_{св}$ = 6,0 мксек.

Обслуживание подсистемы безопасности. Микроконтроллер может обслуживать подсистему безопасности, самостоятельно анализируя значения сигналов с некоторых датчиков, а также обрабатывая запрос прерывания от внешней схемы контроля функционирования. Самостоятельно анализируемые сигналы с датчиков отражают состояние медленно меняющихся параметров аппарата – например, температуры. Результатом анализа может быть подача сигналов предупреждения – например мигание индикатора на панели управления. Запрос прерывания является результатом работы схемы контроля функционирования, которая следит за быстро изменяющимися параметрами – например, основной частотой синхронизации. При пропадании этого сигнала запрос прерывания приводит к снятию высокого напряжения, отображению сообщения об ошибке функционирования.

Если анализируемые сигналы с датчиков имеют логическую форму, то процедура, реализующая оператор рассматриваемого процесса, может иметь следующий вид:

```
TCntr:   mov    OBRP5, P5          ; * ввод значения из порта в регистр-образ порта
         mov    C, IDTP            ; * значение линии копируем
         mov    IdTank, C          ; * в бит светодиода IdTank
         jb     TP2, KTP3          ; *  если TP2 пассивный ( активный «0»), выходим
         mov    A, BLINK           ; * иначе мигаем
         jz     KTP1               ; *  если счетчик мигания обнулился, перезагружаем
         djnz   BLINK, KTP2        ;    если нет
KTP1:    mov    BLINK, #TankRLD    ; * перезагружаем счетчик мигания
         cpl    Tblink             ; * и инвертируем флаг мигания
KTP2:
         mov    C,Tblink           ;* флаг мигания копируем во флаг индикации
         mov    IdTank, C          ; * перегрева
KTP3:
         setb TankSh               ; * установка бита отображения температуры РИ
         ret
```

Эта процедура читает данные с порта P5 (микроконтроллер 80C552) и копирует данные в регистр-образ порта OBRP5, второй и третий биты которого хранят информацию о значениях сигналов с датчиков температуры. Если сигналы образуются на выходах компараторов напряжения, то температурный диапазон разбивается на три участка. Текущая температура показывается оператору светодиодом на панели.

T<T1	T1≤T<T2	T≥T2
IDTP=1, TP2=1	IDTP=0, TP2=1	IDTP=0, TP2=0
Светодиод не горит	Светодиод горит	Светодиод мигает

Мигание светодиода реализуется посредством программного счетчика BLINK, константа перезагрузки которого TankRLD определяет частоту вспышек. Результом работы этого фрагмента программы и процесса является установленный флаг TankSh и значение бита ldTank (активное значение «0» управляет зажиганием светодиода).

Команды последовательного порта на зажигание и гашение светодиода формируются далее процедурами TankOn и TankOFF. Эти процедуры уже относятся к процессу взаимодействия с оператором через последовательный порт.

Рассматриваемый процесс использует следующие ресурсы внутреннего ОЗУ микроконтроллера:

								BLINK Счетчик мигания ldTank
			Tblink	ldTANK	TankSh			**FLAG** Байт дополнительных флагов
				IDTP	KTP2			**OBRP5** Образ P5

Таким образом, процесс обслуживания подсистемы безопасности включает один оператор, реализуется одной процедурой, выполняемой на каждом интервале системной сетки.

Последовательность выполняемых команд в процедуре зависит от входных условий (значений IDTP и TP2). Наиболее длинная (в машинных циклах) последовательность команд $max\ (\Sigma\ Q_i)$ реализуется при переходе на метку KTP2 и занимает 19 циклов микроконтроллера. При $F_{осц}$ = 12 МГц время обслуживания подсистемы безопасности нашего примера составляет $\Delta t_{6п}$=19,0 мксек.

Формирование временных диаграмм логических сигналов управления. Для формирования временных диаграмм внутренних и внешних сигналов при программном управлении используются программные счетчики, временная сетка системы и последовательный опрос флагов, которые характеризуют активность сигналов. При этом в течение каждого элементарного интервала системной сетки процедура формирования циклограммы производит опрос каждого флага, при его активности инкрементирует или декрементирует регистр счетчика сигнала, при наличии переполнения сбрасывает флаг и изменяет значение сигнала. Таким образом, циклограмма сигналов управления формируется с точностью не хуже элементарного интервала системной сетки. Для реализации указанного механизма под

буферы счета и флаги для каждого сигнала в памяти данных микроконтроллера должны быть выделены ресурсы. Модель части ресурсов для СУ рентгенотелевизионного аппарата может выглядеть так:

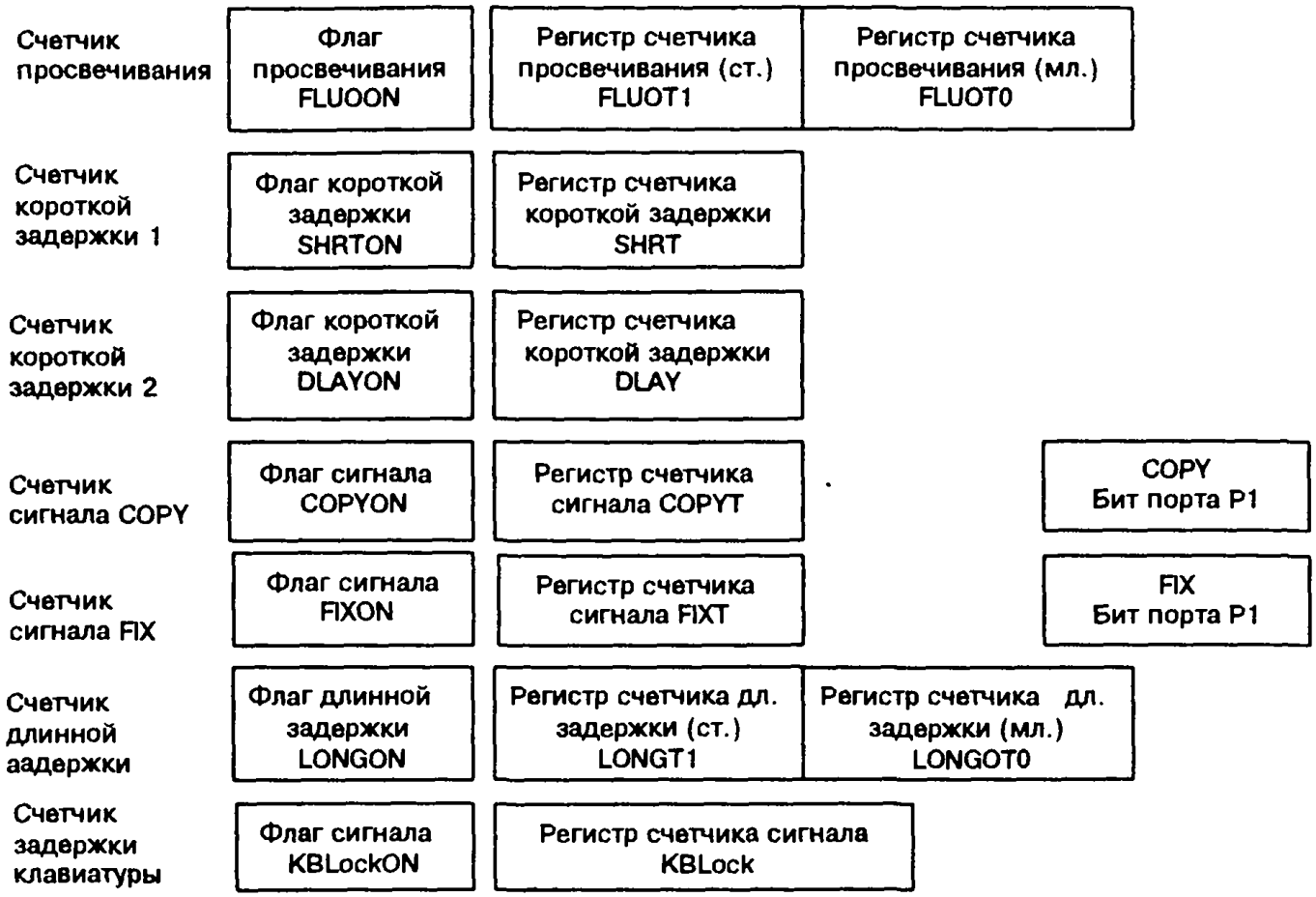

Рис. 1.5. Модель ресурсов для обслуживания циклограммы сигналов

В модели имеются счетчики, обслуживающие циклограмму внешних сигналов (FLUO – включение высокого напряжения, COPY, FIX – сигналы управления СОИ) и счетчики, обслуживающие внутренние протоколы СУ (KBLock – обслуживания клавиатуры панели управления, SHRT, DLAY, LONG – формирования интервалов времени). Флаги событий устанавливаются в соответствии с командами от панели управления. Отметим, что флаги и счетчики реализуются программно и могут быть привязаны к любой ячейке памяти данных микроконтроллера. Определенные выше ресурсы обслуживаются программой в следующей последовательности:

- сохранение в стеке аккумулятора и регистра флагов;
- перезагрузка сторожевого таймера;
- проверка флага просвечивания FLUO, наращивание счетчика FLUOT1, FLUOT0, проверка значения счетчика и включение звукового сигнала при значениях времени 5 мин. и 10 мин.;
- проверка флага короткой задержки SHRTON, наращивание счетчика SHRT, проверка значения счетчика и сброс флага при переполнении;

- проверка флага короткой задержки DLAYON, наращивание счетчика DLAY, проверка значения счетчика и сброс флага при переполнении;
- проверка флага FIXON, наращивание счетчика FIXT, проверка значения счетчика, сброс флага и снятие сигнала FIX при переполнении, гашение индикатора;
- проверка флага COPYON, наращивание счетчика COPYT, проверка значения счетчика, сброс флага и снятие сигнала COPY при переполнении, гашение индикатора;
- проверка флага длинной задержки LONGON, наращивание счетчика LONGT1, LONGT0, проверка значения счетчика и сброс флага при переполнении;
- проверка флага KBLockON, наращивание счетчика KBLock, проверка значения счетчика и сброс флага при переполнении;

Текст процедуры формирования сигналов управления может быть следующим:

```
;*****************************Циклограмма***********************************
IsRdy:    push  ACC
          push  PSW

          orl   PCON, #10h              ; перезагрузка сторожевого таймера
          mov   T3, WATCHT

          jnb   FLUOON, OUTET           ; если установлен флаг просвечивания
          inc   FLUOTO                  ; увеличим мл.байт счетчика на 1
          mov   A,FLUOTO
          jnz   FT1
          inc   FLUOT1                  ; если ноль, прибавим перенос к ст. байту
          mov   A,LastFL                ; сравним текущее время с отображаемым
          xrl   A,FLUOT1
          jz    FT4                     ; если не совпадает,
          setb  TOSOW                   ; установим флаг вывода на дисплей
FT4:      mov   LastFL,FLUOT1           ; переустановим отображаемое время
FT1:      mov   A,FLUOT1                ; проверим текущее время просв-ния
          cjne  A,#T5MIN,FT2            ; если оно равно 5 мин.
          setb  BELL                    ; включим звуковой сигнал
FT2:      cjne  A,#T10MIN,FT3           ; если оно равно 10 мин.
FT3:      jc    OUTFT
          setb  BELL                    ; включим звуковой сигнал
          setb  FLTMNR                  ; установим флаги ошибки просвечивания
          setb  NTRDERR                 ; установим флаг общей ошибки
          ajmp  OUTLG
OUTFT:
          jnb   SHRTON, OUTSH           ; если уст. флаг короткой задержки 1
          mov   A,SHRT                  ; проверим его счетчик
          jz    STPSH ;
          djnz  SHRT, OUTSH             ; если не ноль, инкрементируем его
STPSH:    clr   SHRTON                  ; если ноль, сбросим флаг
OUTSH:
          jnb   DLAYON,OUTDL            ; если уст. флаг короткой задержки 2
          mov   A,DLAY                  ; проверим его счетчик
```

```
        jz      STPDL   ;
        djnz    DLAY, OUTDL             ; если не ноль, инкрементируем его
STPDL:  clr     DLAYON                  ; если ноль, сбросим флаг
OUTDL:
        jnb     FIXON,OUTFX             ; если уст. флаг FIX
        mov     A,FIXT                  ; проверим его счетчик
        jz      STPFX   ;
        djnz    FIXT, OUTFX             ; если не ноль, инкрементируем его
STPFX:  clr     FIXON                   ; если ноль, сбросим флаг
        setb    FIX                     ; уст.пассивное значение линии
        setb    IdFIX                   ; и пассивное значение индикатора
        setb    LEDOUT                  ; уст.флаг обновления светодиодов
OUTFX:
        jnb     COPYON,OUTCP            ; если уст. флаг COPY
        mov     A,COPYT                 ; проверим его счетчик
        jz      STPCP   ;
        djnz    COPYT, OUTCP            ; если не ноль, инкрементируем его
STPCP:  clr     COPYON                  ; если ноль, сбросим флаг
        setb    COPY                    ; уст. пассивное значение линии
        setb    IdCOPY                  ; и пассивное значение индикатора
        setb    LEDOUT                  ; уст.флаг обновления светодиодов
OUTCP:
        jnb     LONGON,OUTLG            ; если уст.  флаг длинной задержки
        clr     C                       ;
        mov     A,LONG0                 ; декремент мл.  байта счетчика
        subb    A,#1                    ;
        mov     LONG0,A ;
        mov     A,LONG1                 ; декремент ст.  байта счетчика
        subb    A,#0    ;
        mov     LONG1,A ;
        jnz     OUTLG                   ; проверим ст. байт на 0
        mov     A,LONG0
        jnz     OUTLG                   ; проверим мл. байт на
        clr     LONGON                  ; счетчик обнулен, сбросим флаг
OUTLG:
        jnb     KBlockON,OUTKB          ; если уст. флаг задержки  клавиатуры
        jb      KBRQ,STPKB              ; и нажата клавиша ;
        mov     A,Kblock                ; проверим счетчик
        jz      STPKB
        djnz    Kblock,OUTKB            ; если не ноль, инкрементируем его
        mov     A,Kblock                ; проверим счетчик
        jz      STPKB   ;
        djnz    Kblock,OUTKB            ; если не ноль, инкрементируем его
STPKB:  clr     KblockON                ; очистка флага
        jb      KBRQ,NORMKB ;
        setb    KblockFast
        ;если не нажата клавиша, уст. "быстрый режим
        sjmp    OUTKB                   ; и уйдем
NORMKB: clr     KblockFast              ; если нажата клавиша,
                                        ; сбросим "быстрый режим"

        pop     PSW
        pop     ACC
        ret
;*****************************************************************
```

Процедура формирования циклограммы сигналов управления может следовать непосредственно за процедурой обслуживания подсистемы безопасности. Последовательность выполняемых команд в этом фрагменте программы зависит от входных условий (значений флагов FLUOON, SHRTON, DLAYON, COPYON, FIXON, LONGON, и KBLockON). Наиболее длинная последовательность команд $max\ (\sum Q_i)$ включает 114 машинных циклов микроконтроллера и при тактовой частоте $F_{осц}$ = 12 МГц занимает $\Delta t_{цикл}$=114,0 мксек.

Регулирование параметров непрерывных процессов. Регулирование целесообразно выполнять с использованием процедуры прерывания, запросом может служить сигнал от компаратора при рассогласовании, величина которого превышает интервал нечувствительности. Кроме величины рассогласования необходимо знать его знак.

В рассматриваемом примере, рентгенотелевизионном аппарате, сигнал рассогласования преобразуется в последовательность импульсов, частота которых пропорциональна напряжению рассогласования. Каждый импульс вызывает прерывание, а значение логического сигнала UP_Auto говорит о необходимости увеличивать или уменьшать значение уставок анодного напряжения КС и тока МС. Их аналоговые значения формируются в два этапа: вначале процедура прерывания анализирует сигнал UP_Auto, а также определяет, не достигнуты ли границы регулирования (LOWLIM,UPLIM). Если границы не достигнуты, формируются новые значения кодов уставок. Далее вызывается процедура LDDAC, которая записывает эти коды по 8-разрядной шине в регистры двухканального ЦАП. Затем формируются сигналы управления LDAC и EA, по которым на выходе ЦАП появляются аналоговые значения КС и МС.

```
;********************** Регулирование по прерыванию Int0**************
Int0:
            push    ACC                     ; *
            push    00                      ; *
            push    PSW                     ; *
            mov     R0, #KVCNTH             ; *
            jb      UP_AUTO,IUP             ; * проверяем линию Up_Auto
    ;Уменьшение
            cjne    @R0,#High (LOWLIM+8),  IDN2     ;если High=Low предел
            dec     R0
            cjne    @R0,#LOW (LOWLIM+8) , IDN2      ;Low<Low  предел, то
IDN2:       jnc     IDN1
Noldec:
            mov     KVCNTL, #Low (LOWLIM)
            mov     KVCNTL, #High (LOWLIM)
            sjmp    IPASS                           ;регулировать не будем
IDN1:                                               ;иначе уменьшаем KVCNT
            clr     C
            mov     A,KVCNTL
            subb    A, #8
            mov     KVCNTL,A
```

```
        mov     A,KVCNTL
        subb    A, #0
        mov     KVCNTL,A
        sjmp    TODAC
IUP:                                    ;Увеличение
        cjne    @R0,#High (UPLIM-8) , IUP1      ; * если High=Up предел
        dec     R0                              ; *
        cjne    @R0,#Low (UPLIM-8) , IUP2       ; * и  Low+8>Up предел
IUP2:   jc      IUP1                            ; *
        mov     KVCNTL, #Low (UPLIM)
        mov     KVCNTL, #High (UPLIM)
        sjmp    IPASS                           ;регулировать не будем
IUP1:                                           ;иначе  увеличиваем
        mov     A, #8                           ; *
        add     A,KVCNTL                        ; *
        mov     KVCNTL,A                        ; *
        mov     A,KVCNTH                        ; *
        addc    A, #0                           ; *
        mov     KVCNTH,A                        ; *
TODAC:                                  ; Загрузка ЦАП
        lcall   LDDAC                           ; *
IPASS:
        pop     PSW                             ; *
        pop     00                              ; *
        pop     ACC                             ; *
IOUT:
        clr     IE0                             ; *
        reti                                    ; *
;*****************************************************************
```

Последовательность выполняемых команд в этом фрагменте программы включает две независимых ветви (наращивание и уменьшение счетчика KVCNT) с одинаковым количеством команд. После проверки значения на линии UP_AUTO выполняется одна из них. Наиболее длинная последовательность команд имеет место, если пределы регулирования не достигнуты. Такая последовательность команд (отмечена символом *) включает 33 машинных цикла микроконтроллера. Еще 54 цикла занимают процедуры извлечения кодов значений напряжения и тока из таблиц (по значению указателя KVCNT) и загрузки их в регистры двухканального ЦАП. Эти процедуры представляют собой линейную последовательность команд и из-за недостатка места мы их приводить не будем.

Таким образом, оператор регулирования реализуется процедурой прерывания, которая включает две процедуры второго уровня (процедура LDDAC вызывает процедуру GetKVWord). Максимальная длина команд набора процедур регулирования $max\ (\sum Q_i)$ включает 87 машинных циклов микроконтроллера и при тактовой частоте $F_{осц}$ = 12 МГц занимает $\Delta t_{рег}$ = 87,0 мксек.

Взаимодействие с оператором через панель управления. Взаимодействие с оператором заключается в приеме кода нажатых на клавиатуре панели клавиш, переключении режимов работы аппарата и установке пара-

метров текущего режима, передаче параметров текущего режима для отображения на дисплеях и индикаторах панели управления.

Электрофизические установки типа рентгенотелевизионного аппарата могут иметь достаточно сложные панели управления, с многочисленными клавишами и переключателями, отдельными индикаторами и дисплеями. Рассматриваемый в качестве примера аппарат имеет следующие устройства на панели управления:

Устройства панели управления	Кол-во устройств
Клавиши	36
Светодиоды	18
Дисплеи	1
Включатели и педали	4

В случае возникновения ошибки на панель должно быть передано соответствующее сообщение, код ошибки и, желательно, диагностическая информация. В целях диагностики аппарат должен оставаться в состоянии готовности текущего режима (высокое напряжение снимается), поэтому временная сетка не должна нарушаться.

Для сокращения количества линий связи меду панелью и платой ведущего микроконтроллера, а также увеличения помехоустойчивости обмена, передача обычно ведется по последовательному каналу. В рассматриваемом примере используется интерфейс RS232, при его использовании желательно сформировать систему команд обмена, например такую:

Код (КОП)	Мнемоника	Действие
02	readkey	чтение кода клавиши
23	light_led	засветить светодиод №
24	dark_led	погасить светодиод №
86	show_data2	высветить данные (2 байта)
87	show_data3	высветить данные (3 байта)

Обмен при использовании подобной системы команд обмен строится по принципу Master – Slave. Устройством Master является плата ведущего микроконтроллера, устройством Slave – микроконтролер панели управления. Все информационные посылки между платами начинаются со специального кода – EscByte. Ведущее устройство посылает команду ведомому в посылке:

EscByte, КОП, B1 ... Bn

где B1.. Bn – байты данных (n= 0.. 4, определяется кодом команды).

Если при получении посылки нет ошибок (первый байт равен EscByte, правильный КОП, верное количество байтов данных), то по окончании приема посылки ведомый выдает ответ «понял» в формате

EscByte, cmOk

и выполняет команду. По окончании выполнения команды ведущему устройству высылается байт готовности в формате

EscByte, ReadyByte.

Использование системы команд увеличивает надежность последовательного обмена, но увеличивает время на обслуживание соответствующего процесса.

При работе через последовательный порт время реакции складывается из двух компонент:

- $\Delta t_{пданные}$ – времени обслуживания процессором квазипараллельного процесса подготовки передаваемых и обработки принятых данных;
- $t_{ппорт}$ – времени работы процедуры прерывания последовательного порта и собственно обмена по линии.

$$t_{панель} = \Delta t_{пданные} + t_{ппорт}$$

Процедура обработки прерывания последовательного порта весьма короткая, при передаче она, в основном, заключается в загрузке байта в регистр данных порта (SBUF в архитектуре MCS-51). Далее последовательный порт работает автономно, а процессор освобождается для обработки программно реализуемых процессов. Поэтому сейчас нас интересует интервал времени $t_{пданные}$.

Фрагмент программы нашего примера, отвечающий за процесс взаимодействия с панелью управления, может иметь следующий вид:

```
Work:
        jb      KBRQ,WNoKey             ; * если нажата клавиша,
WKB:
        lcall   GETKBD                  ; * чтение клавиатуры
        jnb     F0,WNoKey               ; * если байт принят
        setb    TOSHOW                  ; *
        cjne    A,#kbAUTO,W1            ; * проверяем код клавиши
        lcall   InitA                   ;установим автоматический режим
        sjmp    WNoKey

W1:     cjne    A,#kbMNL,W2             ; *
        jb      GRFMOD,WNoKey           ;если не съемка,
        lcall   InitM                   ;установим ручной режим
        sjmp    WNoKey
```

```
W2:     cjne  A,#kbPLS,W3          ; *
        jb    GRFMOD,WNoKey        ;если не съемка,
        lcall InitP                ;установим импульсный режим
        sjmp  WNoKey

W3:     cjne  A,#kbGRF,W5          ; *
        lcall InitG                ;установим режим съемки
        sjmp  WNoKey

W5:
        lcall VICTRL          ; * обработка клавиш упр. видео и диафрагмами

WNoKey:                            ; нажата педаль/кнопка (F1=true/false)
        lcall FTHDTP               ; *
        jnb   F1, WNS              ; *
        jnb   GRFMOD, Wrun         ; *
        mov   OBRP5, P5            ; *
        jb    HDSW, WNS            ; *
        lcall GRFSW                ; *
        sjmp  WNS                  ; *
Wrun:
        lcall RunFluo              ;Fluo автоматический/ручной(импульсный)

WNS:                               ; * No GRF, Fluo
        ajmp  Work                 ; *
```

В начале программы анализируется линия KBRQ, активный низкий уровень на которой свидетельствует о нажатой клавише. Если это имеет место, то вызывается процедура обслуживания последовательного порта GETKBD, которая читает код нажатой клавиши. Далее этот код сравнивается с эталонами и при совпадении вызывается процедура переключения в указанный режим. Если совпадения с основными эталонами нет, то процедурой VICTRL проверяются клавиши второго уровня, устанавливающие параметры текущего режима (положение диафрагм, вид взаимодействия с СОИ и т.д.). Эта процедура длинная (62 машинных цикла), по структуре похожа на опрос счетчиков в процессе формирования циклограммы. При отсутствии нажатых клавиш проверяются включатели и педали (активность характеризует значение флага F1=1) и при наличии сигнала вызываются процедуры RunFluo и GRFSW подачи высокого напряжения при просвечивании и съемке, соответственно.

Наибольшую длительность этот процесс имеет при анализе клавиш второго уровня (клавиши параметров текущего режима в процедуре VICTRL). Длина последовательности команд при этом $max\ (\sum Q_i)$ составляет 105 машинных циклов микроконтроллера и при тактовой частоте $F_{осц}$ = 12 МГц занимает $\Delta t_{пданные}$ = 105,0 мксек.

Критерии планирования квазипараллельных процессов. Основные цели планирования квазипараллельных процессов в системе управления на микроконтроллерах состоят в следующем:

- определить возможность реализация заданного в ТЗ алгоритма управления на данном типе микроконтроллера;
- организовать реализацию алгоритма управления преимущественно программным образом, с минимальным привлечением внешних (относительно микроконтроллера) аппаратных средств.

Следующие критерии, последовательно применяемые при отборе вариантов, могут привести к нужному решению (рис. 1.4):

- Циклограммы сигналов должны формироваться с необходимой точностью, поэтому элементарный интервал времени системной сетки не должен превышать минимального допуска временной диаграммы сигналов управления:

$$\Delta T_{системы} \le min\ \{\Delta T_{сигналов}\} \qquad (W_1)$$

Это условие является обязательным. Оно же дает первое приближение («сверху») величины интервала системной сетки.

- Все операции процессов формирования системной временной сетки, контроля функционирования, формирования циклограмм внутренних и внешних сигналов должны быть выполнены в течении элементарного интервала времени сетки:

$$max\ (\Sigma\Sigma\ Q_{ij}) \le \Delta T_{системы} \qquad (W_2)$$

Это условие также является обязательным, его нарушение может вызвать аварийную ситуацию.

Применение этого условия дает второе приближение величины («снизу») интервала системной сетки.

- Времени, оставшегося в интервалах системной сетки после выполнения обязательных операторов, должно хватить на выполнение нужного количества операторов регулирования параметров технологических процессов за определенное время (должно обеспечиваться необходимое быстродействие аппарата):

$$\sum^{K} \{\Delta T_{системы} - max(\sum^{R}\sum^{P} Q_{ij})\}k \ge H * \Delta t_{рег}, \qquad (W_3)$$

$$H =] F / \Delta U_{рег} [$$
$$K =] F / v * \Delta T_{системы} [$$

где P – количество команд i-го оператора предшествующего процесса в последовательности наибольшей длины;

R – количество предшествующих обязательных (фиксированных) процессов;

H – количество операторов регулирования, посредством которых достигается изменение значения сигнала на указанную величину F;

K – количество интервалов системного времени, за которое сигнал должен измениться на заданную величину F при заданном времени нарастания v.

В нашем примере это означает, что при требуемой скорости нарастания сигнала на выходе KV равной v = 10,0 В/с и дискрете регулирования $\Delta U_{рег}$ = 0,25 В (в одном операторе) при максимальном разбалансе за десять 10-миллисекундных циклов должно выполняться четыре оператора регулирования. Величина изменения напряжения сигнала KV принята равной F = 1,0 В.

Это третье приближение величины («снизу») интервала системной сетки.

- При развитой панели управления и сложном протоколе обмена с ней ведущего микроконтроллера времени, оставшегося в интервалах системной сетки после выполнения обязательных операторов и оператора регулирования, должно хватить на выполнение нужного количества операторов обмена с панелью за определенное время.

$$\sum^{M} \{\Delta T_{системы} - max(\sum^{R+S} \sum^{P} Qij)\}k \le \Delta T_{оператора}, \qquad (W_4)$$

где $R+S$ – количество предшествующих фиксированных и плавающих процессов;

M – количество интервалов системного времени, за которые должен быть выполнен обмен с панелью;

$\Delta T_{оператора}$ – допустимое время ожидания оператором отклика системы.

Это четвертое приближение величины («снизу») интервала системной сетки.

1.4. Спецификация сигналов управления

В предыдущем параграфе были рассмотрены программные квазипараллельные процессы, посредством которых в системе на микроконтроллерах реализуются программные компоненты функций управления.

Анализ сложных СУ на микроконтроллерах, в том числе рассматриваемого примера, показывает, что каждая функция управления может характеризоваться следующими параметрами:

- количеством линий входных и выходных сигналов. В набор сигналов должны быть включены как внешние, так и внутренние сигналы (флаги, регистры микроконтроллера), служащие для взаимодействия процессов;
- типом выполняемой операции преобразования данных. Особенностью микропроцессорной техники является использование типовых операций типа AD- и DA-преобразований, счетчиков и т.д., которые реализованы в виде модулей ввода/вывода микроконтроллеров или отдельных микросхем. В этом случае может быть указано имя операции и разрядность, например ADC-8. При реализации произвольной цифровой операции ее имя может быть связано с описанием на одном из языков типа HDL;
- временем выполнения операции преобразования данных. Имеется в виду полное время аппаратно-программной реализации, входящее в параметр «реальное время» всей системы управления;
- временем программной реализации оператора квазипараллельного процесса;
- объемом используемых внутренних ресурсов микроконтроллера;
- объемом аппаратуры, внешней относительно ведущего микроконтроллера и необходимой для аппаратной реализации операций управления.

С учетом использования в микропроцессорной технике типовых модулей ввода/вывода и интерфейсных БИС можно сделать заключение, что задача проектирования структуры аппаратных средств системы управления на микроконтроллерах представляет собой задачу анализа возможных (применяемых) решений и отбора вариантов на основе системы критериев.

Критерии взаимодействия квазипараллельных процессов мы обсудили в первую очередь, поскольку этот механизм является основой процесса управления. Теперь рассмотрим топологический аспект структурных вариантов – спецификацию сигналов управления. Определим набор сигналов и линий обмена нашего примера – СУ рентгенотелевизионного аппарата, в форме, не зависящей от варианта реализации. Рассмотрим также трансформацию этого набора во входные и выходные сигналы микроконтроллера при традиционном магистрально-модульном подходе и преимущественно параллельном обмене с внешними схемами.

Взаимодействие с оператором. Взаимодействие с оператором осуществляется в основном через панель управления, отдельные команды (например, включения РИ) подаются посредством выносных ручных включателей и педалей. Панель оператора представляется для системы управления как клавиатура, набор индикаторов (например, светодиодов) и дисплеев для отображения символьной информации. Светодиоды индицируют активность выбранного клавишей режима, а также подают сигналы преду-

преждения (наличие излучения, нагрев излучателя выше нормы и т.п.). На дисплеях отображается информация о режиме работы установки и его текущих параметрах.

Количество клавиш, светодиодов, дисплеев нашего примера, а также линий для их обслуживания системой управления приведено в следующей таблице.

Устройства и служебные сигналы	Кол-во устройств	Кол-во линий при обслуживании клавиатуры как матрицы	Кол-во линий при обслуживании панели отдельным контроллером
Клавиши	32	12	-
Светодиоды	18	18	-
Дисплеи	1	11	-
Включатели и педали	4	3	3
Служебные сигналы		0-N	4
Общее кол-во линий		≥44	7

Общее количество устройств взаимодействия с оператором определяется формулой:

$$N_{ОП} = N_{КЛ} + N_{ИНД} + N_{ДИСП} + N_{В}$$

где $N_{КЛ}$, $N_{ИНД}$, $N_{ДИСП}$, $N_{В}$ – количество клавиш, индикаторов, дисплеев и выносных устройств (включателей, педалей). Обычно клавиатуру организуют как матрицу, число сигналов от устройств взаимодействия с оператором, обслуживаемых микропроцессорной системой при этом равно:

$$L_{ОП} = L_{МКЛ} + L_{ИНД} + \Sigma L_{ДИСП} + L_{В}$$

где $L_{МКЛ}$ – сумма строк и столбцов в матрице клавиатуры, $L_{ИНД}$ – количество сигналов обслуживания индикаторов, $\Sigma L_{ДИСП}$ – общее количество сигналов для обслуживания дисплеев, $N_{В}$ – количество сигналов от выносных устройств. Во втором столбце таблицы приведено количество сигналов панели нашего примера при одном дисплее и организации клавиатуры в виде матрицы 4×8.

Даже при матричной организации клавиатуры число линий часто превышает возможности одного микроконтроллера по обслуживанию панели одновременно с другими объектами управления. Для сокращения числа линий и увеличения помехоустойчивости целесообразно организовать обмен между панелью управления и ведущим микроконтроллером по последовательному каналу. При этом панель управления должна иметь собственную плату ведомого микроконтроллера (рис. 1.6).

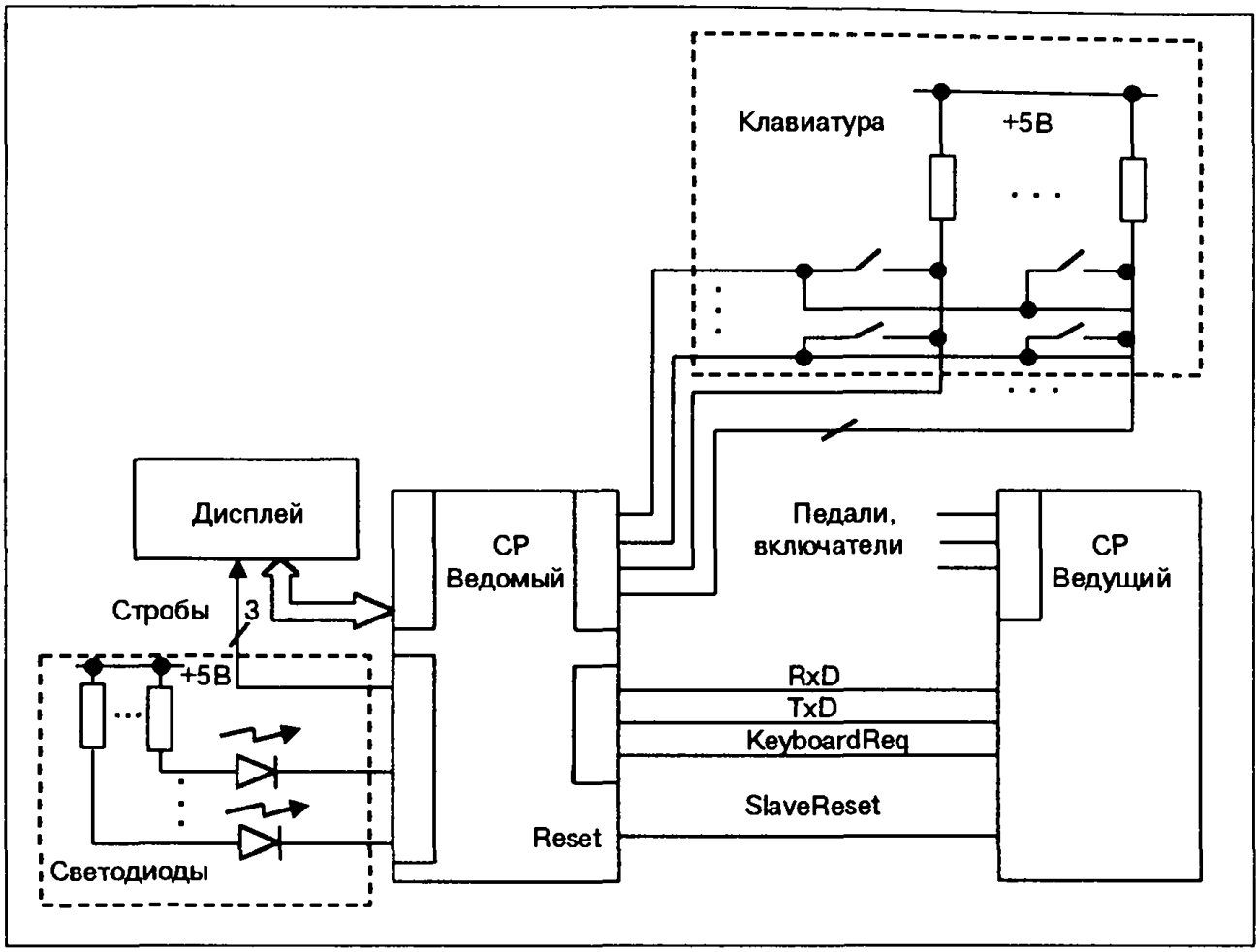

Рис. 1.6. Обслуживание панели управления ведомым микроконтроллером

Для ведущего микроконтроллера число сигналов и линий взаимодействия с оператором при последовательном обмене с платой ведомого микроконтроллера равно:

$$L'_{ОП} = L_{СТ} + L_{П\text{-}ИНДИВИД} + L_{В}$$

где $L_{СТ}$ и $L_{П\text{-}ИНДИВИД}$ – число соответственно стандартных линий канала и линий индивидуальных сигналов. При дуплексном канале ($L_{СТ}$ =2) индивидуальными целесообразно сделать сигналы активности клавиатуры и инициализации процессора панели.

Управление рентгеновским излучателем. Устройство питания РИ получает от системы управления два аналоговых сигнала уставок напряжения и тока. Операция цифро-аналогового преобразования может быть 12-разрядной (DAC-12). Логических сигналов четыре: подачи высокого напряжения на РИ при просвечивании, идентификации режима съемки, включения цепи преднакала РИ, подачи высокого напряжения на РИ при съемке. Таким образом, система управления должна обеспечить формирование двух аналоговых сигналов управления $L_{РИА}$ и четырех цифровых $L_{РИЦ}$. Общее число линий системы для управления РИ равно:

$$L_{РИ} = L_{РИА} + L_{РИЦ}$$

При использовании двухканального 12-разрядного ЦАП с параллельной загрузкой (рис. 1.7) для микроконтроллера два аналоговых сигнала представляются 11 цифровыми сигналами и четырьмя служебными.

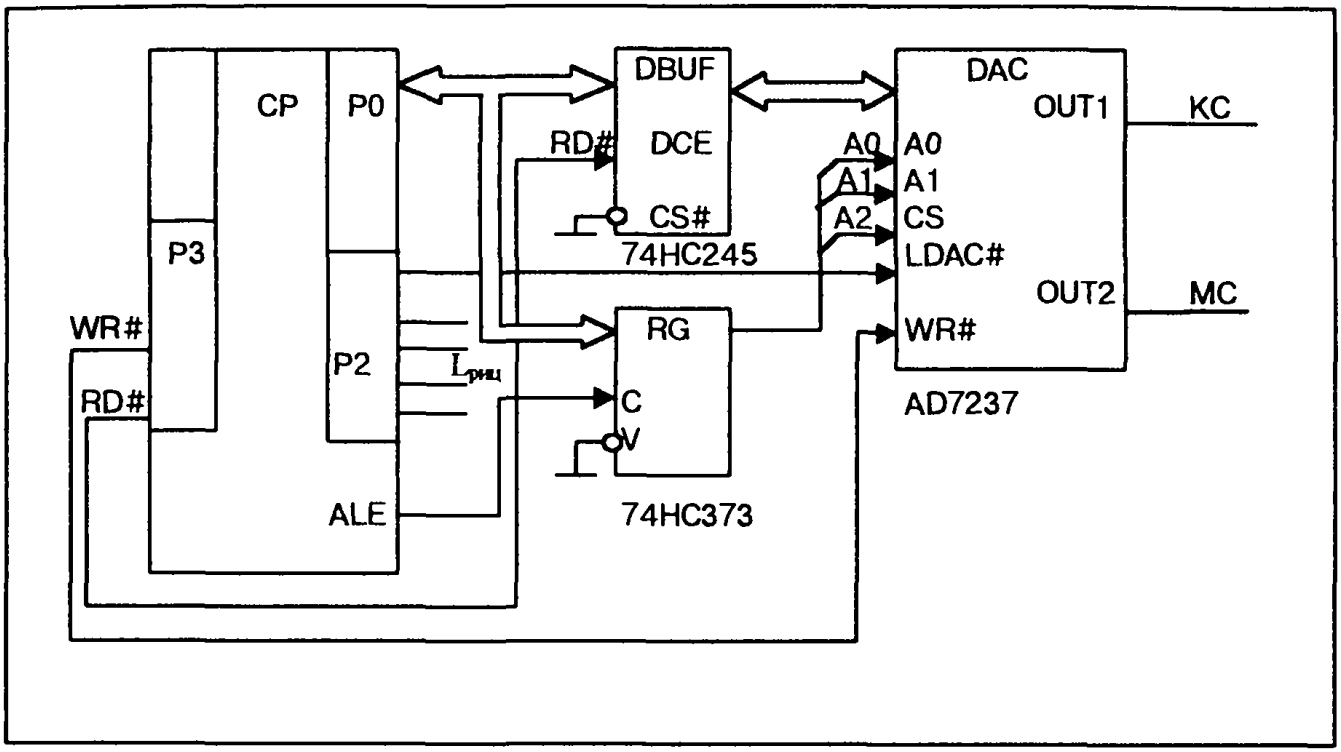

Рис. 1.7. Формирование сигналов управления РИ

Для микроконтроллера число линий обслуживания РИ равно

$$L'_{РИ} = L_Д + L_{РИ\text{-}АДР} + L_{РИ\text{-}СЛ} + L_{РИЦ}$$

где $L_Д$ – разрядность шины данных, $L_{РИ\text{-}АДР}$ и $L_{РИ\text{-}СЛ}$ – число соответственно необходимых линий адреса и необходимых служебных сигналов (записи/чтения, выборки микросхемы и т.п.). В целом $L'_{РИ}$=19, но при мультиплексированной шине адрес/данные $L'_{РИ}$=16.

Управление диафрагмами. Привод диафрагмы представляет собой электромеханическую систему, включающую электродвигатель постоянного тока и потенциометр обратной связи. Таких приводов три: раскрыва ирисовой диафрагмы, раскрыва шторной диафрагмы, поворота шторной диафрагмы. Драйвер управления диафрагмами получает сигналы уставок для каждого привода, с потенциометров приводов получает сигналы обратной связи и формирует биполярные сигналы необходимой мощности для питания электродвигателей. В современных цифровых системах управления в качестве уставок используются коды, которые после цифро-аналогового преобразования образуют сигналы управления драйверами электродвигателей. При такой структуре система управления с помощью ЦАП (операция, например, DAC-8) должна сформировать три аналоговых сигнала для управления электромоторами диафрагм (рис. 1.8).

$$L_{ДФ} = L_{ДА}$$

Микроконтроллер должен обеспечить загрузку кодов уставок в три ЦАП. Общее количество сигналов ведущего микроконтроллера для управления диафрагмами при параллельной загрузке можно определить по формуле:

$$L'_{ДФ} = L_{Д} + L_{ДФ\text{-}АДР} + L_{ДФ\text{-}СЛ}$$

где $L_{Д}$ – разрядность шины данных, $L_{ДФ\text{-}АДР}$ и $L_{ДФ\text{-}СЛ}$ – число соответственно необходимых линий адреса и служебных сигналов. При 8-разрядной шине данных, трех линиях адреса и двух сигналах чтения/записи $L_{ДФ} = 13$.

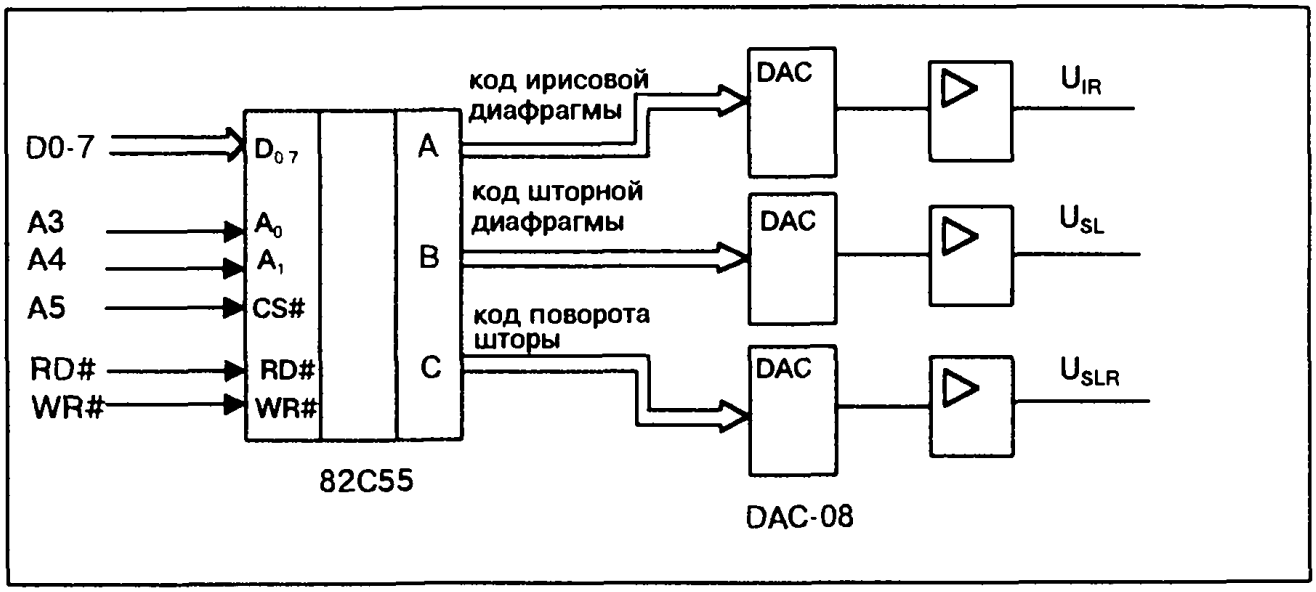

Рис. 1.8. Управление диафрагмами

Взаимодействие с усилителем рентгеновского изображения (УРИ). Система управления должна формировать для УРИ цифровые сигналы переключения полей РЭОП и включения высокого напряжения при просвечивании, а также принимать и обрабатывать аналоговый сигнал управления мощностью дозы.

Количество линий аналоговых и цифровых сигналов между УРИ и системой управления равно

$$L_{УРИ} = L_{УРИА} + L_{УРИЦ}$$

Внешняя относительно микроконтроллера схема обработки входного аналогового сигнала управления мощностью дозы представляет собой преобразователь напряжение-частота. Микроконтроллер получает с этой схемы три цифровых сигнала: запрос прерывания для вызова процедуры регулирования (F_KC), логический сигнал «напряжение больше/меньше»

(Up_Down), логический сигнал разрешения регулирования (F_Enable). Таким образом, для микроконтроллера функция взаимодействия с УРИ представляется пятью цифровыми сигналами

$$L'_{УРИ} = L'_{УРИЦ}$$

Взаимодействие с системой обработки изображений (СОИ). Система управления выдает на СОИ шесть командных логических сигналов, таких как вращение изображения вправо, вращение изображения влево и т.п. Кроме того, система управления и СОИ должны обеспечить взаимодействие при включении и снятии рентгеновского излучения для правильного формирования видеоизображения. Для этого необходимы пять логических сигналов: наличие режима просвечивания; запрос на подготовку СОИ; ответ СОИ; наличие режима импульсного просвечивания; наличие режима одноимпульсного просвечивания. Таким образом, число цифровых сигналов и линий связи между СУ (ведущим микроконтроллером) и СОИ $L_{СОИ} = L'_{УСОИ} = 11$.

Контроль функционирования аппарата и обеспечение безопасности. Подсистемы контроля функционирования и безопасности сложных установок и приборов строятся по иерархическому принципу. Нижний уровень составляют локальные схемы защиты мощных цепей. Средний уровень представляет собой схему анализа параметров важнейших сигналов и формирования флагов предупреждения. Верхний уровень в микроконтроллерных системах реализуется системой прерываний, процедуры которых выполняют аварийный сброс системы и собирают информацию о состоянии флагов предупреждений. Отображение этой информации входит в функцию взаимодействия с оператором.

СУ рентгенотелевизионного аппарата должна обеспечить :

1. Контроль температуры излучателя РИ.
2. Индикацию на панели оператора наличия высокого напряжения на РИ (наличия рентгеновского излучения).
3. Формирование звукового сигнала при просвечивании более 5,0 мин. и 10,0 мин.
4. Снятие высокого напряжения с РИ при превышении временем съемки значения 4,0 сек.

Кроме того, целесообразно дополнительно контролировать следующие параметры: значения напряжений низковольтных цепей питания; напряжение и ток источника питания РИ; значение частоты основного системного синхросигнала; готовность диафрагм; значение логического сигнала «СОИ занята».

Аналоговые значения напряжений с терморезистора РИ, высоковольтного преобразователя РИ (после делителя), ряд напряжений низковольтного питания могут анализироваться внешними аналоговыми компараторами или внутренним блоком АЦП микроконтроллера. Сигнал от таймера просвечивания является внутренним для микроконтроллера. Схемы контроля частоты синхронизации и ограничения времени съемки должны быть внешними из соображений безопасности. Функционирование цепи накала РИ, готовность диафрагм, тока высоковольтного источника питания контролируются локально в соответствующих цепях и на микроконтроллер приходят логические сигналы аварийного состояния.

Возникновение любого аварийного сигнала приводит к формированию сигнала неготовности, по которому ведущий микроконтроллер снимает высокое напряжение с РИ и отображает код ошибки на панели управления. Для определения кода ошибки он должен прочитать состояние всех флагов ошибок.

Количество линий сигналов контроля системы управления можно определить по формуле:

$$L_К = L_{КА} + L_{КЦ} + L_{КСПЕЦ}$$

где $L_{КА}$ – число аналоговых контролируемых сигналов, $L_{КЦ}$ – число цифровых сигналов предупреждения, $L_{КСПЕЦ}$ – количество специальных сигналов, контролируемых СУ (синхросигнал).

Для микроконтроллера функция контроля и обеспечения безопасности представляется сигналами

$$L'_К = L'_{КА} + L'_{КЦ}$$

где $L'_{КА}$ - число аналоговых сигналов (контролируемых микроконтроллером), $L'_{КЦ}$ – число цифровых сигналов предупреждения.

При большом количестве внешних сигналов предупреждения их целесообразно фиксировать во внешних регистрах и читать флаги через шину данных (рис. 1.9). При этом число линий микропроцессора, необходимых для реализации функции безопасности, равно:

$$L'_К = L'_{КА} + L_Д + L_{К\text{-}АДР} + L_{К\text{-}СЛ} + L_{К\text{-}ИНДИВИД}$$

где $L_{К\text{-}АДР}$, $L_{К\text{-}СЛ}$ и $L_{К\text{-}ИНДИВИД}$ – число соответственно необходимых линий адреса, служебных сигналов и индивидуально обслуживаемых сигналов системы контроля.

При 8-разрядной шине данных и двух линиях адреса, которые одновременно служат стробами чтения, трех индивидуальных сигналах общее количество линий микроконтроллера для обслуживания подсистемы контроля $L'_К = 13$.

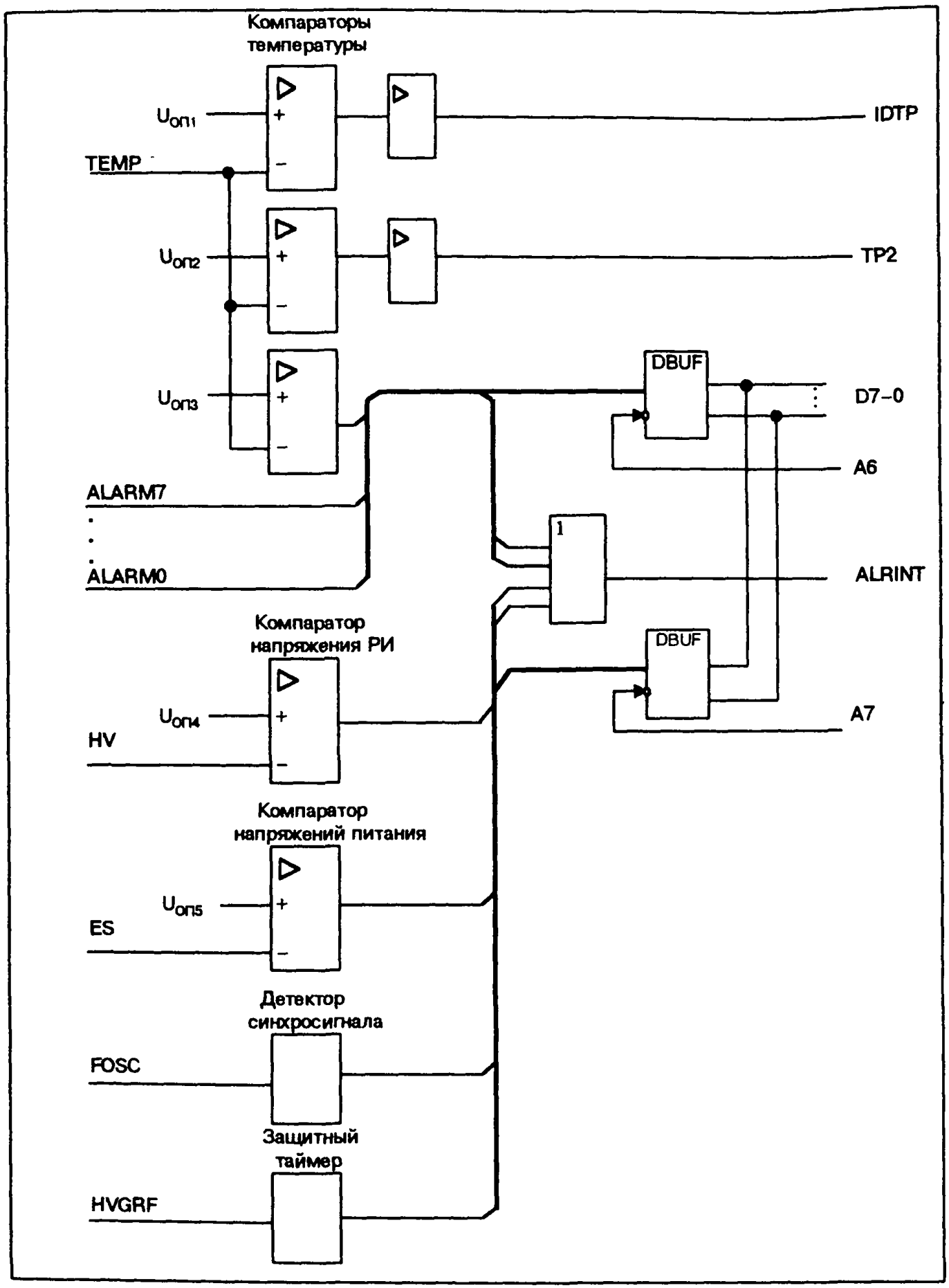

Рис. 1.9. Внешняя схема контроля функционирования установки

Итоговая таблица сигналов, которые должны быть обработаны системой управления и ведущим микроконтроллером, имеет следующий вид:

Объекты управления	Система управления				Ведущий микроконтроллер	
	Линии цифровых сигналов		Линии аналоговых сигналов		Линии цифровых сигналов	Линии аналоговых сигналов
	Входы	Выходы	Входы	Выходы	Входы/Выходы	Входы
Панель управления	5(15*)	2(29*)			7	
Излучатель РИ		4		2	16**(19)	
Блок диафрагм				3	0*** (13)	
УРИ		2	1		5	
СОИ	1	10			11	
Устройство контроля	9		4		3(13)	
Итого	15(25)	18(45)	5	5	42(68)	

* При обслуживании клавиатуры ведущим микроконтроллером
** С учетом мультиплексированного вывода данных и младшего байта адреса через один порт.
*** С учетом того, что шины данных и адреса уже сформированы

Данные таблицы о номенклатуре аналоговых и цифровых сигналов системы управления определяются типом объектов управления и протоколами обмена с ними. Эти данные являются исходными при разработке системы управления. Набор сигналов ведущего микроконтроллера определяется вариантом структуры системы управления. В данном случае специфицирован вариант с преимущественно параллельным обменом через магистраль, аналоговые входы микроконтроллера не используются.

В таблице учтено, что при обмене с устройством управления диафрагмами и устройством контроля по 8-разрядным мультиплексированным шинам данных и адреса общее количество цифровых сигналов сокращается с 68 до 42.

На основе проведенного анализа функций, квазипараллельных процессов и сигналов управления объектами можно произвести выбор типа ведущего микроконтроллера и варианта структуры системы управления.

Выбор ведущего микроконтроллера и структурного варианта системы управления. Выбор типа микроконтроллера и структуры СУ является итерационным процессом. В целом сущность отбора можно сформулировать как распределение и перераспределение между ведущим микроконтроллером и схемами его обрамления программных операторов и аппаратных операций, в совокупности реализующих необходимые функции управления. Основной целью является минимизация схем обрамления и выводов корпуса микроконтроллера, реализация функций управления

преимущественно программным образом или аппаратно-программным образом с использованием внутренних модулей микроконтроллера.

Последовательность отбора вариантов может быть следующей.

- В качестве исходного проверяется вариант структуры СУ, состоящий только из микроконтроллера, возможно с обрамлением в виде драйверов, не выполняющих операций преобразования данных. Выбирается микроконтроллер, у которого количество выводов портов и служебных линий обслуживания обмена минимально превышает количество линий системы управления. Этот микроконтроллер должен быть обеспечен инструментальными средствами (система программирования, схемный эмулятор). Для этого варианта выполняется планирование и расчет времени квазипараллельных процессов на основе критериев, приведенных в предыдущем параграфе.
- Если не найден микроконтроллер, способный обеспечить непосредственное формирование нужного числа сигналов системы управления, или внутренние модули ввода-вывода не могут обеспечить нужного набора операций, или квазипараллельные процессы с привлечением внутренних модулей не обеспечивают нужного значения параметра реального времени τ, то нужно использовать внешние схемы обрамления. При сложной панели управления в первую очередь целесообразно рассмотреть вариант, в котором панель обслуживается ведомым микроконтроллером и связь с ведущим микроконтроллером осуществляется через последовательный канал.
- При использовании внешних схем обрамления важнейшим моментом является определение типа обмена их с микроконтроллером: параллельный или последовательный. Критерием является время выполнения функций управления при выполнении операций преобразования данных внешними схемами. Часто это время определяется скоростью операций аналого-цифрового и цифро-аналогового преобразований. В системах на микроконтроллерах в настоящее время характерной скоростью таких операций является значение 1,5 – 2,0 Мбит/сек. Эту скорость может обеспечить последовательный интерфейс SPI при тактовой частоте микроконтроллера 6 – 8 МГц. Если скорость выше и выбран параллельный обмен, то большинство внешних схем должно поддерживать этот протокол, поскольку уже существующая магистраль позволяет сократить количество линий обмена (см. предыдущую таблицу). Использование последовательных интерфейсов существенно сокращает объем аппаратуры и позволяет вынести внешние схемы к объектам управления.
- При использовании внешних схем обрамления могут высвободиться выводы портов микроконтроллера. При наличии у микроконтролера внутреннего модуля АЦП, входы которого мультиплексированы с ли-

ниями цифровых портов, следует освободить именно эти выводы и использовать их для выполнения функций контроля или регулирования. При выборе варианта следует обратиться к циклограммам квазипараллельных процессов, откуда видно, что оператор контроля занимает фиксированное время в каждом системном интервале, а оператор регулирования распределен по нескольким интервалам и не предъявляет жестких требований к величине интервала системного времени.

- При освобождении линий цифровых портов после добавления к структуре системы внешней схемы в качестве очередного варианта следует по циклограммам квазипараллельных процессов провести расчет возможности программной реализации всех остальных цифровых сигналов (расширение набора сигналов процесса формирования циклограмм).
- В завершение отбора вариантов при наличии свободных выводов следует проверить возможность применения микроконтроллера выбранной архитектуры с меньшим количеством выводов корпуса.

Исходя из предыдущей таблицы и с учетом приведенных критериев в качестве ведущего микроконтроллера рассматриваемого примера может быть выбран микроконтроллер типа MCS-51, у которого пользователю предоставлены 6 портов, например Siemens SAB 80C537 или Philips 87C552.

1.5. Особенности систем управления на микроконтроллерах и ПЛИС

Комбинированное применение микроконтроллеров и ПЛИС в системах управления может быть осуществлено в следующих формах.

- Описание функций микроконтроллера известной архитектуры (или его части) на языке HDL, описание функций внешних специализированных модулей, компоновка иерархического проекта и загрузка файла, описывающего всю систему управления, в одну СБИС программируемой логики. Основными проблемами такого подхода являются: верификация проекта, тестопригодность системы, необходимость использования дорогостоящего оборудования и печатных плат с очень высокими технологическими нормами.
- Использование СБИС, объединяющей на кристалле микроконтроллерное ядро и матрицу программируемой логики. Это могут быть упомянутые выше изделия фирм Triscend и Atmel. Такой подход позволяет выполнить предварительное макетирование частей системы, но вопросы верификации всего проекта и отладки системы управления совместно с объектами остаются сложными.

- Совместное использование БИС микроконтроллеров и ПЛИС, связанных преимущественно через последовательные интерфейсы. Один из микроконтроллеров может выполнять функции ведущего и по командам оператора осуществлять общее управление (переключать режимы, управлять памятью данных и т.п.). ПЛИС способна быстро выполнить операции преобразования данных, формально описанные на одном из языков HDL, причем в одну микросхему могут быть упакованы операции различных функций управления. Преимуществом этого подхода являются: доступность линий связи между компонентами и тестопригодность всей системы управления, невысокая стоимость разработки и производства системы.

В качестве примера рассмотрим вариант системы управления рентгенотелевизионного аппарата, в котором ведущий микроконтроллер и ПЛИС связаны по интерфейсу SPI, функционируют параллельно и выполняют общие распределенные функции управления.

В данном примере ПЛИС выполняет роль расширителя портов и существенно сокращает число линий ввода-вывода микроконтроллера, а также осуществляет контроль частоты синхронизации системы управления и формирует общий сигнал неготовности. Схема связи микроконтроллера с ПЛИС через интерфейс SPI приведена на рис. 1.10.

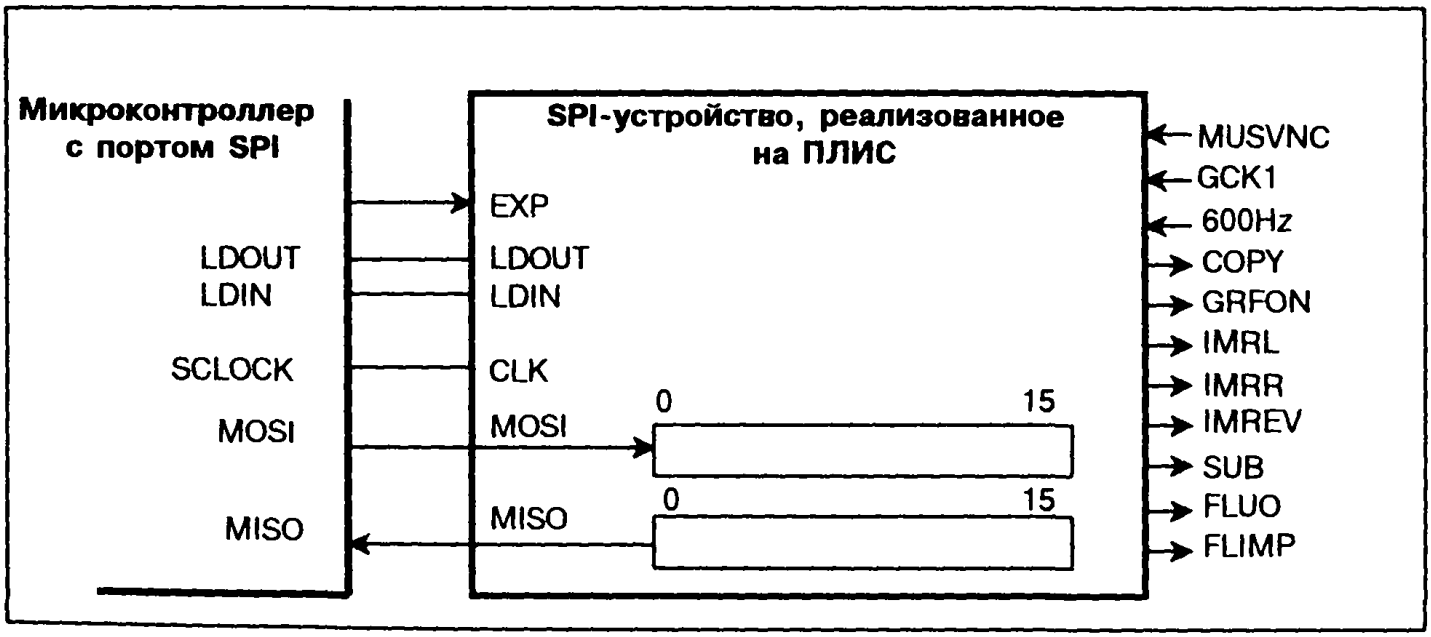

Рис. 1.10. Связь микроконтроллера и ПЛИС через интерфейс SPI

В данном случае SPI-устройство, реализованное на ПЛИС, имеет два 16-разрядных сдвиговых регистра. Один из них работает только на вывод данных из микроконтроллера, другой – только на ввод.

Устройство, реализованное на ПЛИС, содержит схемы обрамления, которые реализуют следующие операции:

- формирование сдвигового регистра вывода данных из микроконтроллера в ПЛИС;
- формирование сигнала копирования кадра COPY;
- формирование сигнала съемки GRFON;
- формирование сигнала аварии по времени съемки TIMAL;
- формирование сигналов: IMRL (вращение изображения влево), IMRR (вращение изображения вправо), IMREV (инверсия изображения), SUB (вычитание кадров), FLUO (управление СОИ), FLIMP (режим импульсного просвечивания);
- формирование сигнала аварии синхронизации CLAL;
- формирование сдвигового регистра ввода данных в микроконтроллер из ПЛИС.

Здесь мы рассмотрим только регистр вывода и связанные с ним схемы. Регистр ввода формируется аналогично и в данной системе управления позволяет передать микроконтроллеру флаги ошибок.

Формирование сдвигового регистра вывода данных из микроконтроллера в ПЛИС. Регистр имеет следующий формат

15	14	13	12	11	10	9	8
IMRL	IMRR	IMREV	COPY	0	SUB	FLUO	FLIM
7	6	5	4	3	2	1	0
0	CERR	VIDERR	TPAL	SAL	0	0	0

Биты регистра вывода:

- IMRL – для выработки внешнего сигнала вращения изображения влево;
- IMRR – для выработки внешнего сигнала вращения изображения вправо;
- IMREV – для выработки внешнего сигнала инверсии изображения;
- COPY – для выработки внешнего сигнала копирования кадра;
- SUB – для выработки внешнего сигнала вычитания кадров;
- FLUO – для выработки внешнего сигнала управления СОИ;
- FLIM – для выработки внешнего сигнала импульсного просвечивания;
- CERR – сигнал ошибки: нет ответного сигнала СОИ;
- VIDERR – сигнал ошибки: нет стабилизации при автоматическом просвечивании;
- TPAL – сигнал ошибки: недопустимая температура блока РИ;
- SAL – сигнал ошибки: источник питания не готов.

Формирование сигнала COPY. Этот сигнал формируется (активный уровень – логический ноль), если нажата соответствующая клавиша на

панели оператора. Сигнал импульсный, его длительность равна 0,5 сек. В ПЛИС приходит 16-разрядное слово от микроконтроллера, в котором находится флаг этого сигнала (если нажата клавиша – логический ноль).

Формирование сигнала GRFON. Этот сигнал команды на включение высокого напряжения в режиме съемки становится активным (уровень логической единицы), если сигнал на входе EXP имеет активный высокий уровень.

Формирование сигнала TIMAL. Этот сигнал ошибки по превышению времени съемки формируется, если длительность сигнала GRFON превышает 5 сек. Активный уровень сигнала ошибки TIMAL – логическая единица.

Формирование сигналов IMRL, IMRR, IMREV, SUB, FLUO, FLIMP. Эти сигналы формируются на соответствующих выводах ПЛИС исходя из значений битов регистра вывода, сформированных микроконтроллером. Активным значением сигналов IMRL, IMRR, IMREV, SUB является логический ноль, а сигналов FLUO и FLIMP логическая единица. Активным значением всех битов регистра вывода является логический ноль.

Формирование сигнала CLAL. Это сигнал ошибки с активным уровнем логической единицы. Он формируется, если на вход ПЛИС «600Hz» подается частота меньше 500 Гц или больше 700 Гц. Используется вспомогательный сигнал GCK1 = 32,768 кГц.

Если рассматриваемое SPI-устройство реализуется на ПЛИС фирмы Altera, то текстовый файл его описания на языке AHDL может иметь следующий вид:

```
SUBDESIGN masha
(
----входы----
EXP_H       :INPUT;    -- сигнал команды на включение таймера и сигнала
                          GRFON при съемке
MNSYNC      :INPUT;    -- синхронизирующий сигнал 50Гц
GCK1        :INPUT;    -- тактовые импульсы 32,768кГц
600HZ       :INPUT;    -- системная частота синхронизации 600Гц
MOS         :INPUT;    -- линия ввода данных в ПЛИС
CLK         :INPUT;    -- синхросигнал обмена
nLDIN       :INPUT;    -- строб ввода из ПЛИС в микроконтроллер
nLDOUT      :INPUT;    -- строб вывода в ПЛИС из микроконтроллера

-----выходы------
nIMRL       :OUTPUT;  -- сигнал вращения изображения влево
nIMRR       :OUTPUT;  -- сигнал вращения изображения вправо
nIMREV      :OUTPUT;  -- сигнал инверсии изображения
nFLUO       :OUTPUT;  -- сигнал режима просвечивания
nCOPY       :OUTPUT;  -- сигнал для СОИ
nSUB        :OUTPUT;  -- сигнал для СОИ
FLIM        :OUTPUT;  -- сигнал импульсного просвечивания
```

```
GRFON      :OUTPUT; -- сигнал команды на включение высокого напряжения при съемке
)

VARIABLE                -- триггеры
TR_grfon   :DFFE;       -- триггер для формирования сигнала RADON
TR_sub     :DFFE;       -- триггер для формирования сигнала SUB
TR_imrr    :DFFE;       -- триггер для формирования сигнала IMRR
TR_imrl    :DFFE;       -- триггер для формирования сигнала IMRL
TR_imrev   :DFFE;       -- триггер для формирования сигнала IDSRH0
TR_flim    :DFFE;       -- триггер для формирования сигнала IDFLIT
TR_fluo    :DFFE;       -- триггер для формирования сигнала COFL
TR_clal    :DFFE;       -- триггер для формирования сигнала CLAL
TR_copy    :DFFE;       -- триггер для формирования сигнала COPY
TR_600hz   :DFFE;       -- триггер для формирования сигнала CLAL
EnaB       :DFFE;       -- триггер, работающий по !LDOUT
Count_sec[6..0]    :DFFE;   -- триггеры 0.5 и 2 секунд для сигнала COPY
Count_5sec[7..0]   :DFFE;   -- счетчик для форирования сигнала ошибки TIMAL
Count_600hz[5..0]  :DFFE;   -- счетчик для формирования сигнала ошибки CLAL-H
Reg_out[15..0]     :DFFE;   -- регистр вывода данных в ПЛИС из микроконтроллера

  -- временные переменные, сигналы ошибок
TIMAL      :NODE;       -- сигнал ошибки: авария по времени съемки
CLAL       :NODE;       -- сигнал ошибки: авария синхронизации
CLALtemp   :NODE;       -- временная переменная CLAL
COPYtemp   :NODE;       -- временная переменная COPY
COPY_L     :NODE;       -- временная переменная COPY
FLUO_L     :NODE;       -- временная переменная COFL
IMREV_L    :NODE;       -- временная переменная IDSRH0
IMRR_L     :NODE;       -- временная переменная IMRR
IMRL_L     :NODE;       -- временная переменная IMRL
FLIM_H     :NODE;       -- временная переменная IDFLIT
SUB_L      :NODE;       -- временная переменная SUB
CERR       :NODE;       -- временная переменная CERR
VIDERR     :NODE;       -- временная переменная VIDERR
TPAL       :NODE;       -- временная переменная TPAL
nSAL       :NODE;       -- временная переменная SAL

BEGIN                   -- начало программы
DEFAULTS                -- значения, присвоенные по умолчанию
  SUB_L=VCC;            -- временная переменная SUB
  FLIM_H=GND;           -- временная переменная IDFLIT
  COPY_L=VCC;           -- временная переменная COPY
  CLUO_L=VCC;           -- временная переменная COFL
  IMREV_L=VCC;          -- временная переменная IDSRH0
  IMRR_L=VCC;           -- временная переменная IMRR
  IMRL_L=VCC;           -- временная переменная IMRL
  CLAL=GND;             -- сигнал ошибки: авария синхронизации
END DEFAULTS;

-- формирование сигналов GRFON, SUB, FLIM, FLUO, IMREV, IMRR, IMRL--

  TR_grfon.D=EXP_H;     -- вход триггера, есть сигнал RADON
  TR_grfon.clk=GCK1;    -- вход тактовых импульсов, 32,768 кГц
  GRFON=TR_grfon.Q;     -- сигнал RADON - активный уровень

  TR_sub.D=SUB_L;       -- вход триггера, временная переменная SUB
```

```
TR_sub.clk=GCK1;      -- вход тактовых импульсов, 32,768 кГц
nSUB=TR_sub.Q;        -- сигнал вычитания кадров

TR_flim.D=FLIM_H;     -- вход триггера, временная переменная IDFLIT
TR_flim.clk=GCK1;     -- вход тактовых импульсов, 32,768 кГц
FLIM=TR_flim;         -- сигнал импульсного просвечивания

TR_fluo.D=FLUO_L;     -- вход триггера, временная переменная FLUO
TR_fluo.clk=GCK1;     -- вход тактовых импульсов, 32,768 кГц
nFLUO=TR_fluo;        -- сигнал управление видеопроцессором

TR_imrev.D=IMREV_L;--вход триггера, временная переменная IMREV
TR_imrev.clk=GCK1;    -- вход тактовых импульсов, 32,768 кГц
nIMREV=TR_imrev;      -- сигнал инверсии изображения

TR_imrr.D=IMRR_L;     -- вход триггера, временная переменная IMRR
TR_imrr.clk=GCK1;     -- вход тактовых импульсов, 32,768 кГц
nIMRR=TR_imrr;        -- сигнал вращения изображения вправо

TR_imrl.D=IMRL_L;     -- вход триггера, временная переменная IMRL
TR_imrl.clk=GCK1;     -- вход тактовых импульсов, 32,768 кГц
nIMRL=TR_imrl;        -- сигнал вращения изображения влево

------ формирование сигнала COPYtemp длительностью 0.5 секунды, активный низкий уро
вень ----
 IF (COPY_L==GND)&(Count_sec[].Q<25) THEN    -- длительность им-
                                                пульса COPY < 0.5 секунды
   Count_sec[].ena=VCC;                      -- сигнал разрешения
   COPYtemp=GND                              -- временная переменная COPY,  активный уровень
 ELSIF (COPY_L==GND)&(Count_sec[].Q>25) THEN       -- длительность
                                                   пульса COPY > 0.5 секунды
   COPYtemp=VCC;                             -- временная переменная COPY, начальный уровень
   Count_sec[].ena=GND;                      -- сигнал, запрещающий работу
 ELSE                                        -- если сигнал COPY=VCC
   COPYtemp=VCC;                             -- временная переменная COPY,начальный уровень
 END IF;

------- счетчик для сигнала COPY -------
 Count_sec[].clrn=nLDOUT;              -- сигнал, обнуляющий счетчик
 Count_sec[].clk=MNSYNC;               -- вход тактовых импульсов
 Count_sec[].d=(Count_sec[].Q+1)& nLDOUT; -- счетчик,который на-
                                             чинает работать при !LDOUT

------------ формирование сигнала COPY ---------
 TR_copy.d=COPYtemp;       -- вход триггера
 TR_copy.clk=GCK1;         -- вход тактовых импульсов
 nCOPY=TR_copy.Q;          -- выход триггера, сигнал COPY

------ формирование сигнала ошибки TIMAL, 5сек. продолжительность сигнала GRFON,---
   --
IF (Count_5sec[].Q<250)&(EXP_H==VCC) THEN    -- длительность  сигнала   GRFON< 5 сек
  Count_5sec[].clk= MNSYNC;                   -- вход тактовых импульсов
  Count_5sec[].d=Count_5sec[].Q+1;            -- счетчик тактовых импульсов
  TIMAL= GND;                                    -- нет сигнала ошибки
 ELSIF (EXP_H==GND) THEN                         -- нет сигнала GRFON
   TIMAL= GND;                                   -- нет сигнала ошибки
```

```
ELSE                                          -- длительность сигнала GRFON > 5 секунд
  TIMAL= VCC;                                 -- есть сигнал ошибки
END IF;

------ формирование сигнала ошибки CLAL, на вход 600HZ подается <500 Гц или >700 Гц
       ------
TR_600hz.d=600HZ;                             -- вход триггера, 600 Гц
TR_600hz.clk= GCK1;                           -- вход тактовых импульсов
IF (TR_600hz.Q==1) THEN                       -- 1/2 периода импульса 600 Гц
 Count_600hz[].clk = GCK1;                    -- вход тактовых импульсов
 Count_600hz[].d=Count_600hz[].Q+1;           -- счетчик такт. импульсов
 Count_600hz[].clrn=TR_600hz;                 -- очистка счетчика
END IF;
IF (Count_600hz[]>=33)#(Count_600hz[]<23) THEN -- если число импульсов >=33 или <23
  CLALtemp=VCC;                               -- есть ошибка
ELSE                                          -- если число импульсов <33 или >=23
  CLALtemp=GND;                               -- нет ошибки
END IF;
TR_clal.d=CLALtemp;
TR_clal.clc=!600HZ;
CLAL=TR_clal.Q;                               -- сигнал ошибки CLAL

------ ввод данных в ПЛИС из микроконтроллера ------
Reg_out[].clc = CLK;                          -- вход тактовых импульсов
Reg_out[].ena = !nLDOUT;                      -- сигнал разрешения
Reg_out[].d = (Reg_out[14..0].Q,MOSI);        -- сдвиговый регистр

------ выработка внешних сигналов ------
EnaB.clk= nLDOUT;                             -- вспомогательный триггер
EnaB.d= VCC;                                  --
EnaB.clrn=nLDOUT;
IF EnaB.Q==VCC THEN-- если выход триггера=VCC,  т.е. сдвиговый регистр не работает
  IMRL_L=Reg_in[15];                          -- временная переменная nIMRL
  IMRR_L=Reg_in[14];                          -- временная переменная nIMRR
  IMREV_L=Reg_in[13];                         -- временная переменная nIMRV
  COPY_L=Reg_in[12];                          -- временная переменная nCOPY
  SUB_L=Reg_in[10];                           -- временная переменная nSUB
  FLUO_L=Reg_in[9];                           -- временная переменная nFLUO
  FLIM_H=Reg_in[8];                           -- временная переменная FLIM
  CERR=Reg_in[6];                             -- сигнал ошибки: нет ответа CMXRSC
  VIDERR=Reg_in[5]; -- сигнал ошибки: нет стабилизации при автомат. просвечивании
  TPAL=Reg_in[4];    -- сигнал ошибки: недопустимая температура блока РИ
  nSUL=Reg_in[3];    -- сигнал ошибки SUL-L: источник питания не готов
END IF;

END; -- конец программы
```

Синтаксические конструкции языка AHDL описаны в главе 5. Там же приведены примеры реализации программируемого счетчика/таймера и регистрового АЛУ RISC микроконтроллера, тексты которых, кроме комментариев, снабжены дополнительными комментариями.