Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет информационных технологий и управления

Кафедра систем управления

О. А. Чумаков, С. И. Городко

МИКРОПРОЦЕССОРЫ В СИСТЕМАХ УПРАВЛЕНИЯ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве учебно-методического пособия для специальности 1-53 01 07 «Информационные технологии и управление в технических системах»

Минск БГУИР 2016

УДК 004.31-022.53:681.5(076)
ББК 32.973.26-04я73+32.965я73
Ч-90

Р е ц е н з е н т ы:
кафедра информационных систем и технологий Белорусского национального технического университета (протокол №3 от 09.11.2015);

доцент кафедры автоматизации производственных процессов и электротехники учреждения образования «Белорусский государственный технологический университет»,
кандидат технических наук, доцент И. Ф. Кузьмицкий

Чумаков, О. А.

Ч-90 Микропроцессоры в системах управления : учеб.-метод. пособие / О. А. Чумаков, С. И. Городко. – Минск : БГУИР, 2016. – 72 с. : ил.
ISBN 978-985-543-267-9.

Приведены сведения, необходимые для синтеза цифровых систем управления на базе однокристальных микроконтроллеров и программной реализации типовых алгоритмов контроля и управления. Даны примеры и конкретные рекомендации, помогающие лучше усвоить соответствующий материал и выполнить задания по контрольным работам.

УДК 004.31-022.53:681.5(076)
ББК 32.973.26-04я73+32.965я73

ISBN 978-985-543-267-9

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Использование микропроцессоров и микроконтроллеров существенно повышает уровень автоматизации процессов управления в технических системах. Функциональная гибкость, высокая надежность, малые габариты и стоимость микропроцессорных средств обусловили целесообразность их применения в различной аппаратуре, в том числе в системах локальной автоматики. В связи с этим существенно изменился процесс проектирования цифровых систем управления. Это вызвано как использованием более сложных функциональных компонентов, так и применением новых архитектурных решений, основанных на замене некоторых аппаратных средств программными модулями. Поэтому проектирование современных средств автоматизации требует от разработчика знания вычислительной техники и программирования на качественно новом уровне с учетом специфики объектов управления в технических системах.

В то же время при проектировании цифровых систем управления на базе микроконтроллеров разработчику приходится решать задачи, многие из которых возникают и при проектировании классической вычислительной техники. Среди них – организация процессорных элементов и обеспечение их взаимодействия с памятью, построение каналов обмена информацией между микроконтроллерами и внешними устройствами, согласование функционирования элементов системы, имеющих различную скорость работы.

В данном курсе рассматриваются архитектура, принципы функционирования и обработки информации в микропроцессорных системах управления, построенных на базе микроконтроллеров. Приводятся базовые сведения о построении подсистемы памяти, организации ввода/вывода информации и системе команд восьмиразрядного микроконтроллера. В заключительном разделе раскрываются принципы программной реализации алгоритмов генерации импульсов, пропорциональных регуляторов, а также алгоритмов контурного управления. Поскольку происходящее сегодня быстрое обновление технических средств делает нецелесообразным описание конкретных устройств, то в данном учебно-методическом пособии изложены базовые принципы построения микропроцессорных систем управления.

1 ЦИФРОВЫЕ СИСТЕМЫ УПРАВЛЕНИЯ НА БАЗЕ МИКРОПРОЦЕССОРОВ И МИКРОКОНТРОЛЛЕРОВ

Микропроцессоры по функционально-структурным особенностям и областям применения можно разбить на следующие основные классы:

- универсальные микропроцессоры с CISC-архитектурой;
- универсальные микропроцессоры с RISC-архитектурой;
- специализированные микропроцессоры (DSP и ряд других);
- микроконтроллеры.

Универсальные микропроцессоры с классической CISC-архитектурой (**C**omplicated **I**nstruction **S**et **C**omputer – компьютер со сложным набором команд) применяются главным образом в персональных компьютерах и серверах. Лидером в этой области является фирма Intel, микропроцессорами которой комплектуется более 80 % выпускаемых персональных компьютеров.

Универсальные микропроцессоры с классической RISC-архитектурой (**R**educed **I**nstruction **S**et **C**omputer – компьютер с сокращенным набором команд) применяются в основном в рабочих станциях и мощных серверах. В этой области имеются несколько ведущих производителей. Необходимо также отметить транспьютеры – оригинальные RISC-микропроцессоры, разработанные фирмой Inmos для построения мультипроцессорных систем.

В классе специализированных микропроцессоров в настоящее время наиболее широко представлены DSP (**D**igital **S**ignal **P**rocessor – процессор для цифровой обработки сигналов), основными производителями которых являются фирмы Texas Instruments, Analog Devices. Кроме DSP выпускаются также микропроцессоры, специализированные для передачи информации в системах телекоммуникаций – коммуникационные контроллеры, для обработки графической информации и ряда других применений.

Микроконтроллеры являются наиболее массовым представителем микропроцессорной техники. Интегрируя на одном кристалле высокопроизводительный процессор, память и набор периферийных устройств, микроконтроллеры позволяют с минимальными затратами реализовать большую номенклатуру систем управления различными объектами и процессами. Благодаря этому микроконтроллеры находят широкое применение в системах промышленной автоматизации, контрольно-измерительной технике, аппаратуре связи, бытовой технике и др.

1.1 Принципы построения цифровых систем управления

Использование микропроцессоров, микроконтроллеров и других средств вычислительной техники в современных системах управления позволяет унифицировать аппаратные средства, а также реализовать сложные алгоритмы управления, обеспечивающие высокие качественные характеристики для широкого круга технических объектов.

Структура типовой микропроцессорной системы управления приведена на рисунке 1.1. Система содержит управляющее устройство (управляющая ЭВМ), в состав которого входят микропроцессор, память, а также устройства связи с объектом (интерфейсы), которые обеспечивают преобразование цифровых кодов на выходе управляющего устройства в сигналы, воспринимаемые объектом управления, а также преобразование выходных сигналов датчиков в двоичные коды, поступающие в управляющее устройство. Управляющая ЭВМ на основании информации о текущем состоянии объекта, а также командных сигналов рассчитывает управляющее воздействие.

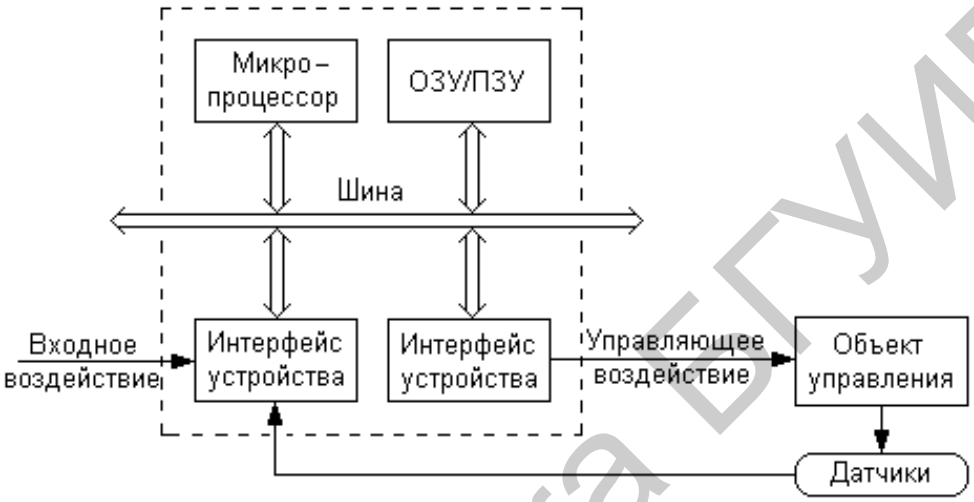

Рисунок 1.1 – Система управления на базе управляющей ЭВМ

При построении систем управления на базе микропроцессоров микроконтроллеров следует учитывать следующие особенности программной реализации алгоритмов управления:

- запаздывание, вносимое управляющей ЭВМ;
- временное и амплитудное квантование сигналов;
- возможности реализации сложных логических и вычислительных процедур, обеспечивающих адаптацию к изменениям параметров объектов и возмущающих воздействий.

Следует заметить, что цифровые системы управления на базе микропроцессоров и микроконтроллеров иногда называют устройствами на основе «программируемой» (гибкой) логики, что означает возможность их перенастройки для управления различными объектами путем изменения программы. Альтернативным методом построения цифровых систем управления является использование «жесткой» логики, при которой ядром устройства является заказная большая интегральная микросхема, реализующая алгоритмы управления для узкого круга объектов. Очевидно, что второй метод на сегодняшний день является более дорогостоящим, однако он широко используется при массовом производстве устройств управления для типовых объектов либо при создании систем управления объектами специального назначения. Для промышленного

применения наиболее рациональным является первый метод, основанный на применении микропроцессоров и микроконтроллеров.

1.2 Эволюция микроконтроллеров

С 70-х годов разработчики систем управления стали использовать вычислительные системы на базе микропроцессоров (МП), выпуск которых был освоен рядом производителей (Intel, Motorola и др.). Применение этой технологии разработки систем управления позволяло повысить скорость и эффективность проектирования новых систем на базе старых, снизить затраты на обнаружение и устранение неисправностей, а также удешевить производство.

Однако в силу своих архитектурных ограничений микропроцессоры не обладали возможностью непосредственно решать задачи управления, и для достижения поставленных целей разработчики вынуждены были снабжать их набором дополнительных устройств: памятью программ и данных, а также набором периферийных элементов: таймерами, счетчиками, аналого-цифровыми (АЦП) и цифроаналоговыми преобразователями (ЦАП), программируемыми контроллерами ввода/вывода и т. п.

Описанная структура с непринципиальными изменениями употреблялась в разработках достаточно часто, в связи с чем возникла идея интеграции наиболее часто применяемых элементов систем управления на одном кристалле.

Первые микроконтроллеры

Воплощение идеи произошло в 1976 году с выпуском фирмой Intel устройства под кодовым обозначением 8048, позднее получившего название микроконтроллер. Помимо центрального процессора на ее кристалле находились 1 Кбайт памяти программ, 64 байта памяти данных, два 8-битных таймера, генератор часов и 27 портов ввода/вывода. Микроконтроллеры семейства 8048 использовались в игровых приставках, клавиатурах первых IBM PC и в ряде других устройств.

Существует также мнение, что первым микроконтроллером была 4- разрядная однокристальная микроЭВМ TMS1000 от Texas Instruments, которая содержала 32 байта памяти данных, 1 Кбайт памяти программ, часы и поддержку ввода/вывода и выпущенная в 1972 году. Термин «микроконтроллер» вытеснил из употребления использовавшийся ранее термин «однокристальная микроЭВМ». Первый же патент на однокристальную микроЭВМ был выдан в 1971 году инженерам М. Кочрену и Г. Буну, сотрудникам Texas Instruments, предложившим разместить на одном кристалле не только микропроцессор, но и память, а также устройства ввода/вывода.

Микроконтроллер 8051

Позже, в 1980 году, компания Intel выпустила следующий микроконтроллер I8051. Удачный набор периферийных устройств, возможность гибкого выбора внешней или внутренней программной памяти и приемлемая цена обеспечили этому микроконтроллеру успех на рынке, и он стал поистине классиче-

ским образцом устройств данного класса. Этот 8-битный чип положил начало целому семейству микроконтроллеров, которые господствовали на рынке вплоть до недавнего времени.

Аналоги 8051 выпускали советские предприятия в Минске, Киеве, Воронеже, Новосибирске, на них выросло целое поколение отечественных разработчиков. На сегодняшний день существует более 200 модификаций микроконтроллеров, совместимых с I8051, выпускаемых десятками компаний. Однако 51 семейство фактически сдало свои позиции более молодым и совершенным микроконтроллерам.

Motorola и Zilog

Другими яркими представителями восьмиразрядных микроконтроллеров явились изделия компаний Motorola (68HC05, 68HC08, 68HC11) и Zilog (Z8).

Motorola длительное время не предоставляла средств, позволяющих дешево и быстро начать работать с ее контроллерами, что явно не способствовало их популярности у некорпоративных разработчиков. В нашей стране их популярность также не очень высока, возможно, еще в силу отсутствия достаточного количества доступных учебных материалов и средств разработки.

Микроконтроллеры фирмы Zilog, основанной бывшими сотрудниками Intel, еще недавно казавшиеся столь многообещающими, не выдержали гонки в стремительно развивающемся секторе рынка, и сегодня система команд Z8 выглядит достаточно устаревшей.

Microchip

Первые значительные перемены произошли с появлением PIC-контроллеров фирмы Microchip. Эти чипы предлагались по рекордно низким ценам, что позволило им в короткий срок захватить значительную часть рынка микроконтроллеров. К тому же кристаллы от Microchip оказались не уступающими, а нередко и превосходящими микроконтроллеры x51 по производительности и не требовали дорогостоящего программатора.

Эти микроконтроллеры остаются популярными в тех случаях, когда требуется создать недорогую систему, не предъявляющую высоких требований по ее управлению.

Scinex

На волне успеха PIC-контроллеров появились очень похожие на них изделия фирмы Scinex. Они обладали уже 52-мя командами против PIC-овских 33-х. Были добавлены инструкции для работы с памятью, улучшена архитектура, каждая команда выполнялась за один такт, что при прочих равных условиях было вчетверо быстрее, чем у Microchip, и к тому же их тактовая частота достигала 100 МГц.

Столь высокая скорость контроллера позволяет его создателям отказаться от различной периферии – таймеров, счетчиков, регистров сдвига в приемопередатчиках – все это рекомендуется реализовывать чисто программными средствами.

Atmel

Настоящая революция в мире микроконтроллеров произошла в 1996 году, когда корпорация Atmel представила свое семейство чипов на новом прогрессивном ядре AVR.

Более продуманная архитектура AVR, быстродействие, превосходящее контроллеры Microchip, привлекательная ценовая политика способствовали оттоку симпатий многих разработчиков от недавних претендентов на звание контроллера номер 1.

Микроконтроллеры AVR имеют более развитую систему команд, насчитывающую до 133 инструкций, производительность, приближающуюся к 1 MIPS/МГц, *F*lash ПЗУ программ с возможностью внутрисхемного перепрограммирования. AVR-архитектура оптимизирована под язык высокого уровня Си. Кроме того, все кристаллы семейства совместимы «снизу вверх».

Огромную роль сыграла доступность программного обеспечения и средств поддержки разработки. У Atmel много бесплатно распространяемых программных продуктов.

AVR практически стали еще одним индустриальным стандартом среди 8-разрядных микроконтроллеров общего назначения.

Микроконтроллеры Atmel легкодоступны в Беларуси и отличаются в среднем невысокой стоимостью, успешно конкурируя с изделиями компании Microchip.

1.3 Основные понятия и определения

Микропроцессор – программно-управляемое устройство, предназначенное для обработки цифровой информации и управления процессом этой обработки. Обычно микропроцессор (МП) состоит из одной (иногда нескольких) интегральных схем и имеет доступ к внешней памяти. Он также обеспечивает передачу информации между компонентами ЭВМ и внешней средой. МП является основой любой микроЭВМ и производится по технологии больших интегральных схем (БИС).

МикроЭВМ – вычислительное или управляющее устройство, содержащее микропроцессор, оперативное запоминающее устройство (ОЗУ), постоянное запоминающее устройство (ПЗУ), таймер, порты ввода/вывода, генератор тактовых импульсов, блок питания и другие элементы.

Однокристальный микроконтроллер – БИС, содержащая в себе МП, ОЗУ, ПЗУ небольшого объема, таймеры/счетчики, порты ввода/вывода, и ориентированная на решения задач управления. Появлению микроконтроллеров способствовало совершенствование технологии производства микроэлектроники, что позволило интегрировать на одном кристалле большинство функциональных блоков управляющей микроЭВМ. В большинстве микроконтроллеров используется восьмиразрядное вычислительное ядро с упрощенной системой команд. Память физически и логически разделена на память программ и память данных.

Подсистема ввода/вывода микроконтроллеров часто имеет аналого-цифровые и цифроаналоговые преобразователи для возможности ввода сигналов от датчиков и вывода сигналов на исполнительное устройство.

Цифровой процессор сигналов (Digital Signal Processor, DSP) – это специализированный микроконтроллер, ориентированный на решение задач цифровой фильтрации в режиме реального времени. Обычно DSP имеет мощное вычислительное ядро, спроектированное, однако, только для решения узкоспециализированных задач. По этой причине сигнальные процессоры имеют сравнительно невысокую стоимость по сравнению с обычной микроЭВМ, реализующей аналогичные функции. Применяемые, например в области телекоммуникаций DSP имеют производительность до 1,6 млрд операций/с. С архитектурной точки зрения такие процессоры могут представлять собой аналоговые функциональные преобразователи сигналов. Часто они выполняют функции аналоговых схем (например, производят генерацию колебаний, модуляцию, смещение, фильтрацию, кодирование и декодирование сигналов в реальном масштабе времени и т. д., заменяя сложные схемы, состоящие из операционных усилителей, катушек индуктивности, конденсаторов и т. д.).

2 ОСНОВЫ ОРГАНИЗАЦИИ МИКРОКОНТРОЛЛЕРОВ

2.1 Понятие микроконтроллера

С появлением однокристальных микроЭВМ произошел настоящий бум автоматизации в области управления. Именно это обстоятельство и определило используемый сегодня термин «микроконтроллер» (от англ. control – управление).

Микроконтроллер (МК) – компьютер на одной микросхеме, предназначенный для управления различными электронными устройствами и осуществления взаимодействия между ними в соответствии с заложенной в него программой.

Основной особенностью современного этапа развития микропроцессорных систем (МПС) является завершение перехода от систем, выполненных на основе нескольких больших интегральных схем, к МК. Последние отличаются от универсальных микропроцессоров (МП) тем, что на одном кристалле объединяют все основные элементы МПС: центральный процессор (ЦП), постоянное запоминающее устройство (ПЗУ), оперативное запоминающее устройство (ОЗУ), порты ввода/вывода и целый ряд специализированных электронных устройств. Поскольку встроенные устройства не требуют никаких внешних электрических цепей, они обладают повышенной надежностью.

ПЗУ служит для хранения выполняемой МК программы. ОЗУ предназначено для временного хранения переменных и организации пула внутренних регистров МК. Порты ввода/вывода обеспечивают сопряжения МК с внешними устройствами (в частности, с персональным компьютером, поскольку подавляющее число МК поддерживают стандартный интерфейс RS-232C).

К наиболее распространенным специализированным устройствам относятся:

- АЦП – аналого-цифровые преобразователи (ADC – Analog-to-Digital Converter);
- ЦАП – цифроаналоговые преобразователи (DAC – Digital-to-Analog Converter);
- цифровые потенциометры;
- различные интерфейсы связи;
- таймеры – устройства, позволяющие осуществлять их программирование на выдачу прямоугольных импульсов с заданными периодом и длительностью, а также отсчитывать интервалы времени между соседними входными импульсами.

В зависимости от конкретной модели МК часть устройств может отсутствовать либо в МК могут быть встроены уникальные устройства (например, драйверы жидкокристаллического дисплея, драйверы вакуумного флуоресцентного дисплея и т. п.).

Все специализированные устройства выполняют свои задачи под управлением микропроцессорного ядра МК.

2.2 Структурная схема микроконтроллера

МК объединяются в семейства. К одному семейству относят изделия, имеющие одинаковое микропроцессорное ядро, под которым понимают совокупность таких понятий, как система команд, циклограмма работы ЦП, организация памяти программ и памяти данных, система прерываний. Наиболее важная особенность семейства – программная совместимость на уровне двоичного кода всех входящих него МК.

Отличия между различными представителями одного семейства заключаются в основном в составе периферийных устройств и объеме памяти программ и данных.

Современные МК обладают, как правило, рядом отличительных признаков. Перечислим основные из них:

- модульная организация, при которой на базе одного процессорного ядра (центрального процессора) проектируется ряд (так называемая линейка) МК, различающихся объемом и типом памяти программ, объемом памяти данных, набором периферийных модулей, частотой синхронизации;
- использование закрытой архитектуры МК, которая характеризуется отсутствием линий магистралей адреса и данных на выводах корпуса МК. Таким образом, МК представляет собой законченную систему обработки данных, наращивание возможностей которой с использованием параллельных магистралей адреса и данных не предполагается;
- использование типовых функциональных периферийных модулей (таймеры, ЦАП, АЦП, контроллеры последовательных интерфейсов и др.), алгоритмы работы которых практически не имеют отличий в МК различных производителей;
- расширение числа режимов работы периферийных модулей, которые задаются в процессе инициализации регистров специальных функций МК.

При модульном принципе построения все МК одного семейства содержат процессорное ядро, одинаковое для всех МК данного семейства, и изменяемый функциональный блок, который различен для МК разных моделей. Структурная схема модульного МК приведена на рисунке 2.1.

Процессорное ядро включает в себя:

- центральный процессор;
- внутреннюю контроллерную магистраль (ВКМ), состоящую из шин адреса, данных и управления;
- схему синхронизации МК;
- схему управления режимами работы МК, включая поддержку режимов пониженного энергопотребления, сброса (реинициализации) и т. д.

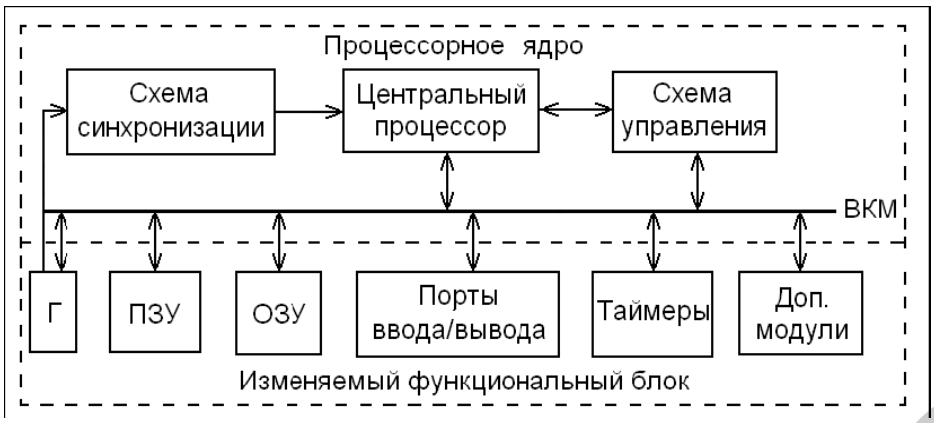

Рисунок 2.1 – Обобщенная структурная схема микроконтроллера

Изменяемый функциональный блок включает в себя модули памяти различного типа и объема, порты ввода/вывода, модули тактовых генераторов (Г), таймеры. В состав изменяемого функционального блока могут входить такие дополнительные модули, как компараторы напряжения, АЦП, ЦАП и др. Что касается модуля обработки прерываний, то в относительно простых МК он входит в состав процессорного ядра, в более сложных МК он представляет собой отдельный модуль с развитыми возможностями.

Каждый модуль проектируется для работы в составе МК с учетом конкретного протокола ВКМ. Данный подход позволяет создавать разнообразные по структуре МК в пределах одного семейства.

2.3 Классификация микроконтроллеров

Сегодня в мире выпускаются тысяч типов МК. Основным признаком для их классификации является разрядность данных, обрабатываемых арифметико-логическим устройством (АЛУ) микропроцессорного ядра. По этому признаку они делятся на:

- 4-разрядные – самые простые и дешевые МК с минимальными функциональными возможностями, предназначенные для низкостоимостных приложений;

- 8-разрядные – наиболее многочисленная группа благодаря оптимальному сочетанию цены и возможностей. Наиболее яркие представители этой группы – МК серии MCS-51 (Intel) и совместимые с ними, а также PIC (Microchip), HC68 (Motorola), Z8 (Zilog);

- 16-разрядные – класс более высокопроизводительных, чем 8-разрядные МК, устройств, но при этом и более дорогостоящих;

- 32- и 64-разрядные – МК, являющиеся обычно модификациями универсальных микропроцессоров.

В отдельную группу выделяют цифровые сигнальные процессоры (DSP – Digital Signal Processor) – специализированные МК, ориентированные на использование в системах обработки сигналов.

Сегодня наибольшая доля мирового рынка микроконтроллеров принадлежит 8-разрядным устройствам (около 50 % в стоимостном выражении), широко используемым в промышленности, бытовой и компьютерной технике. За ними следуют 16-разрядные МК и DSP (каждая из групп занимает примерно по 20 % рынка).

В то время как 8-разрядные МП общего назначения полностью вытеснены более производительными моделями, 8-разрядные МК продолжают широко использоваться. В своем развитии они прошли путь от простейших приборов с относительно слаборазвитой периферией до современных многофункциональных МК, обеспечивающих реализацию сложных алгоритмов управления в реальном масштабе времени. Причиной жизнеспособности 8-разрядных МК является их использование для управления реальными объектами, где применяются в основном алгоритмы с преобладанием логических операций, скорость обработки которых практически не зависит от разрядности процессора. Кроме того, существует большое количество применений, в которых не требуется высокая производительность, но важна низкая стоимость.

2.4 Архитектура процессорного ядра МК

Основными характеристиками, определяющими производительность процессорного ядра МК, являются:

- система команд процессора и методы адресации операндов;
- набор регистров для хранения промежуточных данных;
- организация процессов выборки и исполнения команды.

CISC- и RISC-архитектуры. С точки зрения системы команд и методов адресации операндов процессорное ядро современных 8-разрядных МК относится к одной из двух архитектур:

- CISC-архитектура (Complicated Instruction Set Computer), реализующая так называемую полную систему команд;
- RISC-архитектура (Reduced Instruction Set Computer), реализующая сокращенную систему команд.

CISC-процессоры выполняют большой набор команд с развитыми возможностями адресации, предоставляя разработчику выбор наиболее подходящей команды для выполнения необходимой операции. Такая система команд, как правило, неортогональна, т. е. не все команды могут использовать любой из способов адресации применительно к любому из регистров процессора. Выборка команды и ее выполнение осуществляются в течение нескольких циклов работы МК. К МК с CISC-архитектурой относятся МК фирмы Intel с ядром MCS-51, которые поддерживаются в настоящее время целым рядом производителей, МК семейств HC05, HC08 и HC11 фирмы Motorola и ряд других.

В процессорах с RISC-архитектурой набор исполняемых команд сокращен до минимума. Для реализации более сложных операций приходится комбинировать имеющиеся команды. При этом все команды имеют формат фиксированной длины, выборка команды из памяти и ее исполнение осуществляется за один цикл.

Система команд RISC-процессора предполагает возможность равноправного использования всех регистров процессора. Это обеспечивает дополнительную гибкость при выполнении ряда операций. К МК с RISC-процессором относятся МК AVR фирмы Atmel, МК PIC фирмы Microchip.

Основная идея RISC-архитектуры заключается в тщательном подборе таких комбинаций кодов операций, которые можно было бы выполнить за один такт тактового генератора. Выигрыш от такого подхода – резкое упрощение аппаратной реализации ЦП и возможность значительно повысить его производительность. Первоначально реализовать такой подход удавалось, лишь существенно сократив набор команд, отсюда и родилось название RISC.

Очевидно, что в общем случае одной команде в рамках CISC-архитектуры должны соответствовать несколько команд в рамках RISC-архитектуры. Однако обычно выигрыш от повышения быстродействия RISC-архитектуры перекрывает потери от менее эффективной системы команд, что приводит к более высокой эффективности RISC-систем в целом по отношению к CISC. Также с упрощением ЦП уменьшается число транзисторов, необходимых для его реализации, следовательно, уменьшается площадь кристалла. А с этим связано снижение стоимости МК и потребляемой мощности.

Фон-неймановская (принстонская) и гарвардская архитектуры. С точки зрения организации процессов выборки и исполнения команды в современных МК применяется одна из двух архитектур МПС: фон-неймановская (принстонская) или гарвардская.

Основной особенностью фон-неймановской (принстонской) архитектуры является использование общей памяти для хранения программ и данных, как показано на рисунке 2.2.

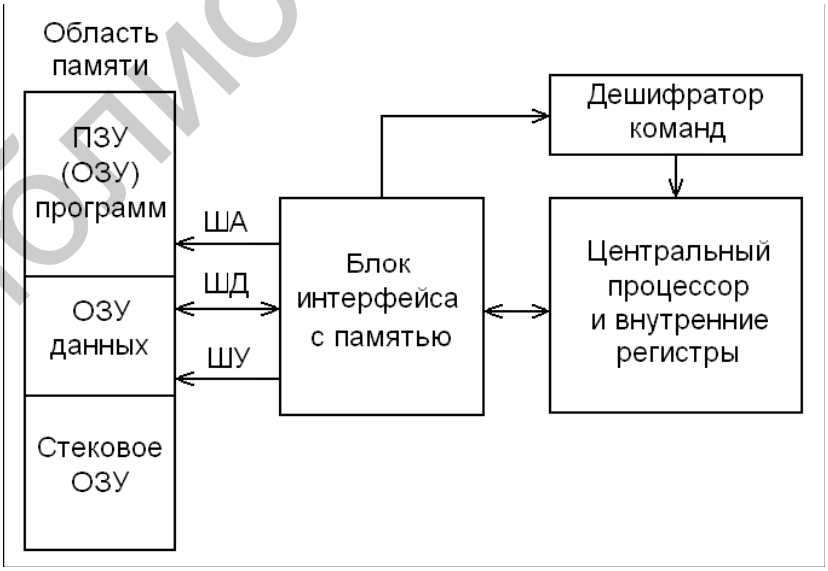

Рисунок 2.2 – Структура МПС с фон-неймановской архитектурой

Основное преимущество архитектуры фон-Неймана – упрощение устройства МПС за счет обращения только к одной общей памяти. Использование единой области памяти позволяет оперативно перераспределять ресурсы между областями программ и данных, что существенно повышает гибкость МПС с точки зрения разработчика программного обеспечения. Размещение стека в общей памяти облегчает доступ к его содержимому.

Основной особенностью гарвардской архитектуры является использование раздельных адресных пространств для хранения команд и данных, как показано на рисунке 2.3.

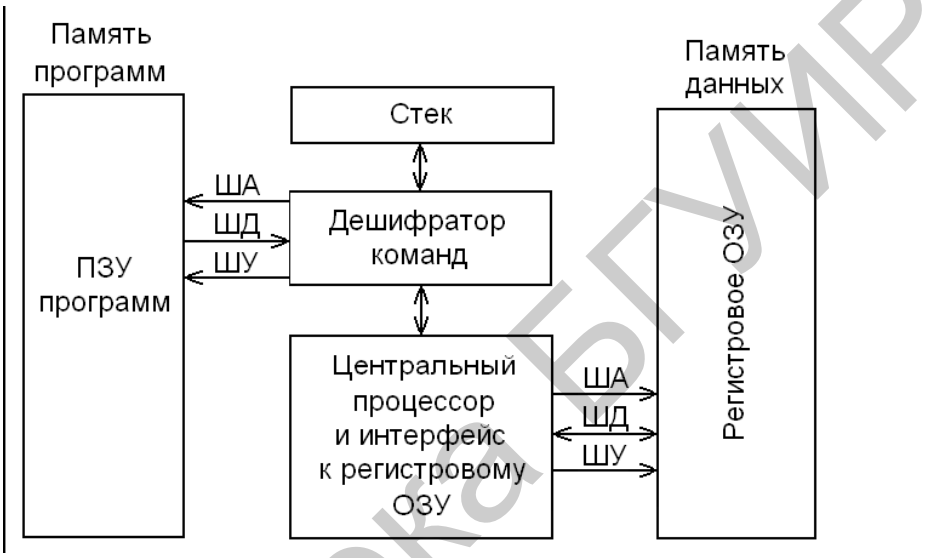

Рисунок 2.3 – Структура МПС с гарвардской архитектурой

Гарвардская архитектура почти не использовалась до конца 70-х годов, пока производители МК не поняли, что она дает определенные преимущества разработчикам автономных систем управления.

Отсюда и главное достоинство гарвардской архитектуры состоит в том, что она обеспечивает потенциально более высокую скорость выполнения программы по сравнению с фон-неймановской за счет возможности реализации параллельных операций. Благодаря разделению адресных пространств для хранения команд и данных выборка следующей команды может происходить одновременно с выполнением предыдущей, что позволяет обеспечивать выполнение различных команд за одинаковое число тактов. В связи с этим большинство производителей современных 8-разрядных МК используют именно гарвардскую архитектуру.

Сравнение МК, выполненных по разным архитектурам, является сложной задачей и ее следует проводить применительно к конкретному приложению.

Действительно объективной количественной оценкой эффективности работы МК можно считать скорость ВКМ, определяющую производительность МК, т. е. число операций, которое может быть выполнено за единицу времени.

Производительность МК измеряют в MIPS (Million Instructions per Second – миллион инструкций в секунду). Очевидно, что с ростом тактовой частоты производительность МК возрастает.

Другая количественная характеристика, описывающая экономичность работы МК, – потребляемая им мощность. Она, как правило, также возрастает с повышением тактовой частоты МК.

2.5 Особенности системы команд микроконтроллеров

Так же как и в любой микропроцессорной системе, набор команд ЦП МК включает в себя четыре основные группы команд:

- команды пересылки данных;
- арифметические команды;
- логические команды;
- команды переходов.

Для реализации возможности независимого управления разрядами портов (регистров) в большинстве современных МК предусмотрена также группа команд битового управления (так называемый булевый, или битовый процессор). Наличие команд битового процессора позволяет существенно сократить объем кода управляющих программ и время их выполнения.

В ряде МК выделяют также группу команд управления ресурсами контроллера, используемую для настройки режимов работы портов ввода/вывода, управления таймером и т. п. В большинстве же современных МК внутренние ресурсы контроллера отображаются на память данных, поэтому для целей управления ресурсами используются команды пересылки данных.

Система команд МК по сравнению с системой команд универсального МП имеет, как правило, менее развитые группы арифметических и логических команд, зато более мощные группы команд пересылки данных и управления. Эта особенность связана со спецификой применения МК, для которых прежде всего важен контроль окружающей обстановки и способность быстрой реакции с формированием управляющих воздействий.

2.6 Организация памяти микроконтроллеров

В МК используются три основных вида памяти:

- память программ представляет собой ПЗУ, предназначенное для хранения программного кода (команд) и констант. Ее содержимое в ходе выполнения программы не изменяется;
- память данных представляет собой ОЗУ и предназначена для хранения переменных в процессе выполнения программы;
- регистры МК – этот вид памяти включает в себя внутренние регистры процессора и регистры, которые служат для управления периферийными устройствами (регистры специальных функций).

Также можно выделить два дополнительных класса памяти:

- стековая память, организуемая для вызова подпрограмм и обработки прерываний;

- внешняя память – дополнительная память программ или данных, подключаемая к МК извне в тех случаях, когда встроенной памяти оказывается недостаточно.

Память программ. Основным свойством памяти программ является ее энергонезависимость, т. е. возможность хранения программы при отсутствии питания. Но при этом она может быть запрограммирована только один или же ограниченное число раз.

Выделяют следующие типы энергонезависимой памяти программ:

1) ПЗУ масочного типа – mask-ROM. Содержимое ячеек ПЗУ этого типа заносится при ее изготовлении с помощью технологических шаблонов – масок – и не может быть впоследствии изменено. Поэтому МК с таким типом памяти программ используются только после длительной опытной эксплуатации. Основным недостатком данной памяти является необходимость значительных затрат на создание комплекта фотошаблонов и их внедрение в производство. Обычно такой процесс занимает 2-3 месяца и является экономически выгодным только при выпуске свыше нескольких тысяч приборов. Главное достоинство ПЗУ масочного типа – высокая надежность, обусловленная программированием в заводских условиях с последующим контролем результата;

2) электрически программируемые ПЗУ с плавкими перемычками (ЭПЗУ), данный тип ПЗУ программируется потребителем путем пропускания импульсов тока до разрушения перемычек, соответствующих битам, которые должны стать нулевыми;

3) перепрограммируемые ПЗУ с ультрафиолетовым стиранием – EPROM (Erasable Programmable ROM). ПЗУ данного типа программируются электрическими сигналами и стираются с помощью ультрафиолетового облучения. Ячейка памяти EPROM представляет собой МОП-транзистор с «плавающим» затвором, заряд на который переносится с управляющего затвора при подаче соответствующих электрических сигналов. Для стирания содержимого ячейки она облучается ультрафиолетовым светом, который сообщает заряду на плавающем затворе энергию, достаточную для преодоления потенциального барьера и стекания на подложку. Этот процесс может занимать от нескольких секунд до нескольких минут. МК с EPROM допускают многократное программирование и выпускаются в керамическом корпусе с кварцевым окошком для доступа ультрафиолетового света. Такой корпус стоит довольно дорого, что значительно увеличивает стоимость МК;

4) однократно программируемые ПЗУ – OTPROM (One-Time Programmable ROM). Представляют собой версию EPROM, выполненную в корпусе без окошка для уменьшения стоимости МК на его основе (это уменьшение настолько значительно, что в последнее время OTPROM часто используют вместо масочных ПЗУ);

5) электрически стираемые перепрограммируемые ПЗУ – EEPROM (Electrically Erasable Programmable ROM). ПЗУ данного типа можно считать новым поколением EPROM, в которых стирание ячеек памяти производится электрическими сигналами. Применение EEPROM позволяет перепрограммировать МК, не снимая его с платы. По цене EEPROM занимают среднее положение между OTPROM и EPROM. Технология программирования памяти EEPROM допускает побайтовое стирание и программирование ячеек. Несмотря на очевидные преимущества EEPROM, только в редких моделях МК такая память используется для хранения программ. Связано это с ограниченным объемом EEPROM-памяти и с тем, что почти одновременно с ней появился похожий на EEPROM, но более дешевый тип памяти Flash-ПЗУ;

6) ПЗУ с электрическим стиранием типа Flash – Flash-ROM. Функционально *F*lash-память мало отличается от EEPROM. Основное различие состоит в способе стирания записанной информации. В памяти EEPROM стирание можно осуществлять отдельно для каждой ячейки, а во *F*lash-памяти стирание производится только целыми блоками. Если необходимо изменить содержимое одной ячейки *F*lash-памяти, потребуется перезаписать весь блок. Упрощение декодирующих схем по сравнению с EEPROM привело к тому, что МК с Flash-памятью составляют серьезную конкуренцию МК как с однократно программируемыми ПЗУ, так и с масочными ПЗУ.

Память данных. Память данных МК выполняется, как правило, на основе статического ОЗУ. Термин «статическое» означает, что содержимое ячеек ОЗУ сохраняется при снижении тактовой частоты МК до сколь угодно малых значений (с целью снижения энергопотребления). Большинство МК имеют такой параметр, как «напряжение хранения информации» – $U_{STANDBY}$. При снижении напряжения питания ниже минимально допустимого уровня U_{DDMIN}, но выше уровня $U_{STANDBY}$ работа программы МК выполняться не будет, но информация в ОЗУ сохраняется. При восстановлении напряжения питания можно будет сбросить МК и продолжить выполнение программы без потери данных. Уровень напряжения хранения составляет обычно около 1 В, что позволяет в случае необходимости перевести МК на питание от автономного источника и сохранить в этом режиме данные ОЗУ.

Объем памяти данных МК, как правило, невелик и составляет обычно десятки и сотни байт. Это обстоятельство необходимо учитывать при разработке программ для МК. Так, при программировании МК константы, если возможно, не хранятся как переменные, а заносятся в ПЗУ программ.

Регистры МК. МК имеют набор регистров, которые используются для управления его ресурсами. В число этих регистров входят обычно регистры процессора (аккумулятор, регистры состояния, индексные регистры), регистры управления (регистры управления прерываниями, таймером), регистры, обеспечивающие ввод/вывод данных (регистры данных портов, регистры управления параллельным, последовательным или аналоговым вводом/выводом). Обращение к этим регистрам может производиться по-разному.

В МК с RISC-процессором все регистры (часто и аккумулятор) располагаются по явно задаваемым адресам. Это обеспечивает более высокую гибкость при работе процессора. В некоторых МК все регистры и память данных располагаются в одном адресном пространстве. Это означает, что память данных совмещена с регистрами. Такой подход называется «отображением ресурсов МК на память».

В других МК адресное пространство устройств ввода/вывода отделено от общего пространства памяти. Отдельное пространство ввода/вывода дает некоторое преимущество процессорам с гарвардской архитектурой, обеспечивая возможность считывать команду во время обращения к регистру ввода/вывода.

Стековая память. Стековой называют память, доступ к которой организован по принципу «последним записан – первым считан» (Last Input *F*irst Output – LI*F*O). Данный тип памяти необходим для организации вызова подпрограмм и обработки прерываний. При этих операциях содержимое программного счетчика и основных регистров сохраняется в стеке, а затем восстанавливается при возврате к основной программе. Стек в МК может быть организован как аппаратно, так и программно.

Аппаратный стек представляет собой совокупность регистров, связи между которыми организованы таким образом, что при записи и считывании данных содержимое стека автоматически сдвигается. Основное достоинство аппаратного стека – высокое быстродействие, а недостаток – ограниченная емкость. В гарвардской архитектуре стековые операции производятся в специально выделенной для этой цели памяти. Это означает, что при выполнении программы вызова подпрограмм процессор с гарвардской архитектурой производит несколько действий одновременно.

Другой подход – использование для организации стека ОЗУ данных. В фон-неймановской архитектуре единая область памяти используется в том числе и для реализации стека. При этом снижается производительность устройства, так как одновременный доступ к различным видам памяти невозможен. Так при выполнении команды вызова подпрограммы выборка следующей команды осуществляется после того, как в стек будет помещено содержимое программного счетчика.

МК обеих архитектур имеют ограниченную емкость памяти для хранения данных. Если в процессоре имеется отдельный стек и объем записанных в него данных превышает его емкость, то происходит циклическое изменение содержимого указателя стека, и он начинает ссылаться на заполненную ранее ячейку стека. В результате при возврате из подпрограммы будет получен неправильный адрес возврата. Если МК использует общую область памяти для размещения данных и стека, то существует опасность, что при переполнении стека произойдет запись в область данных.

Внешняя память. Несмотря на существующую тенденцию по переходу к закрытой архитектуре МК, в некоторых случаях возникает необходимость подключения дополнительной внешней памяти программ или данных. Если МК

содержит специальные аппаратные средства для подключения внешней памяти, то эта операция производится обычным способом.

Второй более универсальный способ заключается в том, чтобы использовать порты ввода/вывода для подключения внешней памяти и реализовать обращение к памяти программными средствами. Такой способ позволяет задействовать простые устройства ввода/вывода без реализации сложных шинных интерфейсов, однако приводит к снижению быстродействия системы при обращении к внешней памяти.

2.7 Организация связи МК с внешней средой и временем

Порты ввода/вывода. Каждый МК имеет некоторое количество линий ввода/вывода, которые объединены в многоразрядные (чаще 8-разрядные) параллельные порты ввода/вывода. Порты выполняют роль устройств временного согласования функционирования МК и объекта управления, которые в общем случае работают асинхронно.

В памяти МК каждому порту ввода/вывода соответствует свой адрес регистра данных. Обращение к регистру данных порта ввода/вывода производится теми же командами, что и обращение к памяти данных. Кроме того, во многих МК отдельные разряды портов могут быть опрошены или установлены командами битового процессора.

Различают следующие типы параллельных портов в зависимости от их функций:

- однонаправленные порты, предназначенные только для ввода или только для вывода информации;
- двунаправленные порты, направление передачи которых (ввод или вывод) определяется в процессе инициализации МК;
- порты с альтернативной функцией (мультиплексированные порты). Отдельные линии этих портов используются совместно со встроенными периферийными устройствами МК, такими как таймеры, АЦП, контроллеры последовательных интерфейсов;
- порты с программно управляемой схемотехникой входного/выходного буфера.

Различают три типа алгоритмов обмена информацией между МК и внешним устройством через параллельные порты ввода/вывода:

- режим простого программного ввода/вывода;
- режим ввода/вывода со стробированием;
- режим ввода/вывода с полным набором сигналов подтверждения обмена.

Таймеры и процессоры событий. Большинство задач управления, которые реализуются с помощью МК, требуют исполнения их в реальном времени. Под этим понимается способность системы получить информацию о состоянии управляемого объекта, выполнить необходимые расчетные процедуры и выдать

управляющие воздействия в течение достаточно короткого интервала времени, достаточного для желаемого изменения состояния объекта.

Возлагать функции формирования управления в реальном масштабе времени только на ЦП неэффективно, так как это занимает ресурсы, необходимые для расчетных процедур. Поэтому в большинстве современных МК используется аппаратная поддержка работы в реальном времени с использованием таймеров.

Модули таймеров служат для приема информации о времени наступления тех или иных событий от внешних датчиков событий, а также для формирования управляющих воздействий во времени.

Модуль «классического» таймера 8-разрядного МК представляет собой 8- или 16-разрядный счетчик со специальной схемой управления. Схемотехникой МК обычно предусматривается возможность использования таймера как в режиме формирования последовательности импульсов, так и в режиме счетчика внешних событий, поэтому его часто называют таймером/счетчиком.

Направление счета – только прямое, т. е. при поступлении входных импульсов содержимое счетчика инкрементируется.

В зависимости от настройки счетчик может использовать один из источников входных сигналов:

- импульсную последовательность с выхода управляемого делителя частоты;

- сигналы внешних событий, поступающие на вход МК.

В первом случае говорят, что счетчик работает в режиме таймера, во втором – в режиме счетчика событий. При переполнении счетчика устанавливается в «единицу» триггер переполнения, который генерирует запрос на прерывание, если прерывания от таймера разрешены. Пуск и останов таймера, а также установка или чтение текущего состояния счетчика выполняются программным способом.

Рассмотренный «классический» модуль таймера/счетчика широко применяется в различных моделях относительно простых МК. Он может использоваться для измерения временных интервалов и формирования последовательности импульсов. Основными недостатками такого таймера/счетчика являются:

- потери времени на выполнение команд пуска и останова таймера, приводящие к появлению ошибки при измерении временных интервалов и ограничивающие минимальную длительность измеряемых интервалов времени;

- сложности при формировании временных интервалов (меток времени), отличных от периода полного коэффициента счета;

- невозможность одновременного обслуживания сразу нескольких каналов.

Первые из двух перечисленных недостатков были устранены в усовершенствованном модуле таймера/счетчика, используемом в МК семейства MCS-51 (Intel) посредством введения дополнительной логики счетного входа. Такое решение повышает точность измерения временных интервалов, так как пуск и останов таймера производится уже аппаратно. Также в усовершенствованном таймере реализован режим перезагрузки счетчика произвольным кодом

в момент переполнения. Это позволяет формировать временные последовательности с периодом, отличным от периода полного коэффициента счета.

Однако эти усовершенствования не устраняют главного недостатка модуля «классического» таймера – одноканального режима работы. Совершенствование подсистемы реального времени МК в данной области ведется по следующим направлениям:

- увеличение числа модулей таймеров/счетчиков. Этот путь характерен для фирм, выпускающих МК со структурой MCS-51, а также для МК компаний Mitsubishi и Hitachi;

- модификация структуры модуля таймера/счетчика, при которой увеличение числа каналов достигается не за счет увеличения числа счетчиков, а за счет введения дополнительных аппаратных средств входного захвата (input capture – IC) и выходного сравнения (output compare – OC). Такой подход используется, в частности, в МК компании Motorola.

Принцип действия канала входного захвата таймера/счетчика состоит в следующем. Схема детектора события «наблюдает» за уровнем напряжения на одном из входов МК. При изменении уровня логического сигнала вырабатывается импульс записи, и текущее состояние счетчика таймера записывается в 16-разрядный регистр входного захвата. Описанное действие в микропроцессорной технике называют событием захвата. Предусмотрена возможность выбора типа сигнала на входе, воспринимаемого как событие:

- положительный (передний) фронт сигнала;
- отрицательный (задний) фронт сигнала;
- любое изменение логического уровня сигнала.

Выбор типа события захвата устанавливается в процессе инициализации таймера и может неоднократно изменяться в ходе выполнения программы. Каждое событие захвата приводит к установке в «1» триггера входного захвата и появлению на его выходе флага входного захвата. Состояние триггера входного захвата может быть считано программно, а если прерывания по событию захвата разрешены – формируется запрос на прерывание.

Использование режима входного захвата позволяет исключить ошибки измерения входного интервала времени, связанные со временем перехода к подпрограмме обработки прерывания, так как копирование текущего состояния счетчика осуществляется аппаратными, а не программными средствами. Однако время перехода на подпрограмму обработки прерывания накладывает ограничение на длительность измеряемого интервала времени, так как предполагается, что второе событие захвата произойдет позже, чем будет считан код первого события.

Аппаратные средства канала выходного сравнения построены следующим образом. Цифровой компаратор непрерывно сравнивает текущий код счетчика таймера с кодом, который записан в 16-разрядном регистре выходного сравнения. В момент равенства кодов на одном из выходов МК устанавливается заданный уровень логического сигнала. Обычно предусмотрено три типа изменения сигнала на выходе в момент события выходного сравнения:

- установка высокого логического уровня;
- установка низкого логического уровня;
- инвертирование сигнала на выходе.

При наступлении события сравнения устанавливаются в «1» триггер выходного сравнения и соответствующий ему признак выходного сравнения. Аналогично режиму входного захвата состояние триггера выходного сравнения может быть считано программно, а если прерывания по событию сравнения разрешены – формируется запрос на прерывание.

Режим выходного сравнения предназначен прежде всего для формирования временных интервалов заданной длительности. Длительность сформированного временного интервала при этом определяется только разностью кодов, последовательно загружаемых в регистр выходного сравнения, и не зависит от программного обеспечения МК. Минимальную длительность формируемого временного интервала ограничивает только время, необходимое для записи нового значения кода в регистр канала сравнения.

Аппаратные средства усовершенствованного таймера позволяют решить многие задачи управления в реальном времени. Однако с усложнением алгоритмов управления отчетливо проявляются ограничения модулей усовершенствованного таймера, а именно:

- недостаточное число каналов захвата и сравнения, принадлежащих одному счетчику временной базы. Это не позволяет сформировать синхронизированные между собой многоканальные импульсные последовательности;
- однозначно определенная конфигурация канала (или захват или сравнение) часто не удовлетворяет потребностям решаемой задачи;
- формирование сигналов по методу широтно-импульсной модуляции (ШИМ) требует программной поддержки, что снижает максимально достижимую частоту выходного сигнала.

Поэтому следующим этапом развития модулей подсистемы реального времени МК стали модули процессоров событий, которые являются достаточно сложными устройствами и при изучении которых следует обращаться к специальной литературе.

2.8 Система прерываний

Обработка прерываний в МК происходит в соответствии с общими принципами обработки прерываний в МПС. Модуль прерываний принимает запросы прерывания и организует переход к выполнению подпрограммы обработки данного прерывания. Запросы прерывания могут поступать как от внешних источников, так и от источников, расположенных в различных внутренних модулях МК. В качестве входов для приема запросов от внешних источников чаще всего используются выводы параллельных портов ввода/вывода, для которых эта функция является альтернативной. Источниками запросов внешних прерываний также могут быть любые изменения внешних сигналов на некоторых специально выделенных линиях портов ввода/вывода.

Источниками внутренних запросов прерываний могут служить, например, следующие события:

- переполнение таймеров/счетчиков;
- сигналы от каналов входного захвата и выходного сравнения таймеров/счетчиков или от процессора событий;
- готовность памяти EEPROM;
- сигналы прерывания от дополнительных модулей МК, включая завершение передачи или приема информации по одному из последовательных портов и др.

Любой запрос прерывания поступает на обработку, если прерывания в МК разрешены и разрешено прерывание по данному запросу. Адрес, который загружается в программный счетчик при переходе к обработке прерывания, называется «вектором прерывания». В зависимости от организации модуля прерываний конкретного МК различные источники прерываний могут иметь разные векторы или использовать некоторые из них совместно.

Вопрос о приоритетах при одновременном поступлении нескольких запросов на прерывание решается в различных МК по-разному. Есть МК с одноуровневой системой приоритетов, в рамках которой все запросы равноценны, многоуровневой системой с фиксированными приоритетами и многоуровневой программируемой системой приоритетов.

Особенно важны аппаратные прерывания, связанные с включением питания, подачей сигнала «сброс» и переполнением сторожевого таймера. Они имеют немаскируемый характер и чаще всего разделяют один общий вектор прерывания. Это вполне логично, поскольку результатом каждого из событий является сброс МК.

2.9 Режимы энергопотребления

Малый уровень энергопотребления является зачастую определяющим фактором при выборе способа реализации цифровой управляющей системы. Современные МК предоставляют пользователю большие возможности в плане экономии энергопотребления и имеют, как правило, следующие основные режимы работы:

- активный режим (Run mode) – основной режим работы МК. В этом режиме МК исполняет рабочую программу, и все его ресурсы доступны. Потребляемая мощность имеет максимальное значение P_{RUN}. Большинство современных МК выполнено по КМОП-технологии, поэтому мощность потребления в активном режиме сильно зависит от тактовой частоты;
- режим ожидания (Wait mode, Idle mode или Halt mode). В этом режиме прекращает работу центральный процессор, но продолжают функционировать периферийные модули, которые контролируют состояние объекта управления. При необходимости сигналы от периферийных модулей переводят МК в активный режим, и рабочая программа формирует необходимые управляющие воздействия. Перевод МК из режима ожидания в рабочий режим осуществляется

по прерываниям от внешних источников или периферийных модулей, либо при сбросе МК. В режиме ожидания мощность потребления МК P_{WAIT} снижается по сравнению с активным режимом в 5–10 раз;

- режим останова (Stop mode, Sleep mode или Power Down mode). В этом режиме прекращает работу как центральный процессор, так и большинство периферийных модулей. Переход МК из состояния останова в рабочий режим возможен, как правило, только по прерываниям от внешних источников или после подачи сигнала сброса. В режиме останова мощность потребления МК P_{STOP} снижается по сравнению с активным режимом примерно на три порядка и составляет единицы микроватт.

Два последних режима называют режимами пониженного энергопотребления. Минимизация энергопотребления системы на МК в целом достигается за счет оптимизации мощности потребления МК в активном режиме, а также использования режимов пониженного энергопотребления. При этом необходимо иметь в виду, что режимы ожидания и останова существенно отличаются временем перехода из режима пониженного энергопотребления в активный режим. Выход из режима ожидания обычно происходит в течение 3–5 периодов синхронизации МК, в то время как задержка выхода из режима останова составляет несколько тысяч периодов синхронизации. Кроме снижения динамики работы системы значительное время перехода в активный режим является причиной дополнительного расхода энергии.

Мощность потребления МК в активном режиме является одной из важнейших характеристик контроллера. Она в значительной степени зависит от напряжения питания МК и частоты тактирования. Поскольку зависимость тока потребления от напряжения питания МК почти прямо пропорциональная, снижение напряжения питания весьма существенно понижает мощность потребления МК. Необходимо, однако, иметь в виду, что для многих типов МК с понижением напряжения питания уменьшается максимально допустимая частота тактирования, т. е. выигрыш в потребляемой мощности сопровождается снижением производительности системы.

2.10 Тактовые генераторы

Современные МК содержат встроенные тактовые генераторы, которые требуют минимального числа внешних времязадающих элементов. На практике используются три основных способа задания тактовой частоты генератора: с помощью кварцевого резонатора, керамического резонатора и внешней RC-цепи.

Кварцевый или керамический резонатор подключается к входу и выходу инвертирующего усилителя, встроенного в МК. Обычно для стабильной работы генератора требуется также подключение двух дополнительных конденсаторов, емкости которых определяются производителем МК для конкретной частоты резонатора.

Использование кварцевого резонатора позволяет обеспечить высокую точность и стабильность тактовой частоты (разброс частот кварцевого резонатора обычно составляет менее 0,01 %). Такой уровень точности требуется для обеспечения точного хода часов реального времени или организации интерфейса с другими устройствами. Основными недостатками кварцевого резонатора являются его низкая механическая прочность (высокая хрупкость) и относительно высокая стоимость.

При менее жестких требованиях к стабильности тактовой частоты возможно использование более стойких к ударной нагрузке керамических резонаторов. Многие керамические резонаторы уже имеют встроенные конденсаторы, что позволяет уменьшить количество внешних подключаемых элементов с трех до одного. Керамические резонаторы имеют разброс частот обычно порядка 0,5 %.

Самым дешевым способом задания тактовой частоты МК является использование внешней RC-цепи. Последняя не обеспечивает высокой точности задания тактовой частоты (разброс частот может доходить до десятков процентов), что неприемлемо для многих приложений, где требуется точный подсчет времени. Однако имеется масса практических задач, где точность задания тактовой частоты не имеет большого значения.

Зависимость тактовой частоты МК от номиналов RC-цепи зависит от конкретной реализации внутреннего генератора и приводится в руководстве по применению контроллера.

Практически все МК допускают работу от внешнего источника тактового сигнала, который подключается ко входу внутреннего инвертирующего усилителя. При помощи внешнего тактового генератора можно задать любую тактовую частоту МК в пределах его рабочего диапазона и обеспечить синхронную работу сразу нескольких устройств.

Некоторые современные МК содержат встроенные RC-генераторы, которые позволяют контроллеру работать вообще без внешних цепей синхронизации. Работа внутреннего генератора обычно разрешается путем программирования соответствующего бита регистра конфигурации МК.

Ряд семейств МК (например HC08 фирмы Motorola) имеют в своем составе схему тактирования, основанную на принципе синтезатора частоты с контуром фазовой автоподстройки (PLL – phase loop lock). Такая схема работает как умножитель частоты и позволяет задавать тактовую частоту с помощью низкочастотного кварцевого резонатора, что снижает уровень электромагнитного излучения МК. Коэффициенты деления контура PLL могут быть изменены программным путем, что позволяет снизить тактовую частоту (и соответственно потребляемую мощность) в промежутки времени, когда высокое быстродействие не требуется.

В некоторых МК семейства AVR фирмы Atmel тактовая частота контроллера, задаваемая внутренней RC-цепью, также может изменяться программными средствами.

2.11 Аппаратные средства обеспечения надежной работы

Прикладная программа, записанная в память программ МК, должна обеспечивать его надежную работу при любых комбинациях входных сигналов. Однако в результате электромагнитных помех, колебаний напряжения питания и других внешних факторов предусмотренный разработчиком ход выполнения программы может быть нарушен. С целью обеспечения надежного запуска, контроля работы МК и восстановления работоспособности системы в отсутствие оператора все современные МК снабжаются аппаратными средствами обеспечения надежной работы. К ним относятся:

- схема формирования сигнала сброса МК;
- модуль мониторинга напряжения питания;
- сторожевой таймер.

Схема формирования сигнала сброса МК. При включении напряжения питания МК должен начать выполнять записанную в памяти программу работы. На этапе нарастания напряжения питания МК принудительно переводится в начальное состояние, которое называют состоянием сброса. При этом устанавливаются в исходное состояние внутренние магистрали МК, сигналы управления и регистры специальных функций.

С целью обеспечения надежного запуска от любых источников питания с различной динамикой нарастания напряжения большинство современных МК содержат встроенный детектор напряжения питания (схема Power-On-Reset – POR), который формирует сигнал сброса при нарастании напряжения питания.

Сразу после выхода из состояния сброса МК выполняет следующие действия:

- запускает генератор синхронизации МК. Для стабилизации частоты тактирования внутренними средствами формируется задержка времени;
- считывает, если необходимо, энергонезависимые регистры конфигурации в соответствующие регистры ОЗУ;
- загружает в счетчик команд адрес начала рабочей программы;
- производит выборку первой программы из памяти программ и приступает к выполнению программы.

Адрес ячейки памяти, в которой хранится код первой исполняемой команды, называют вектором начального запуска, или вектором сброса. В некоторых МК этот адрес однозначно определен и приведен в техническом описании. Про такие МК говорят, что они имеют фиксированный вектор сброса. В других МК вектор сброса может быть произвольно определен пользователем. На этапе программирования МК необходимый вектор начального запуска записывается в ячейки с фиксированными адресами, и при выходе МК из сброса автоматически загружается в счетчик команд. О таких МК говорят, что они имеют загружаемый вектор сброса.

Для перевода МК в состояние сброса при установившемся напряжении питания достаточно подать сигнал высокого или низкого уровня (в соответ-

ствии со спецификацией МК) на вход сброса (RESET). Обычно для формирования сигнала сброса при включении напряжения питания и нажатии кнопки сброса используют RC-цепь.

МК может перейти в состояние сброса также по сигналам устройств контроля состояния, которые имеются в составе контроллера. В этом случае говорят, что МК находится в состоянии внутреннего сброса. При этом буфер линии RESET устанавливается в состояние вывода с низким логическим уровнем на выходе. Данный сигнал может быть использован для установки в начальное состояние периферийных интегральных схем.

Блок детектирования пониженного напряжения питания. В реальных условиях эксплуатации может сложиться такая ситуация, при которой напряжение питания МК опустится ниже минимально допустимого, но не достигнет порога отпускания схемы POR. В этих условиях МК может «зависнуть»: при восстановлении напряжения питания до номинального значения МК останется неработоспособным.

Для восстановления работоспособности системы после «просадки» напряжения питания МК необходимо снова сбросить. Для этой цели в современных МК реализован дополнительный блок детектирования пониженного напряжения питания. Рассматриваемый модуль генерирует сигнал внутреннего сброса при снижении напряжения питания до уровня чуть ниже минимально допустимого. Уровень срабатывания данного блока значительно превышает напряжение сохранения данных в ОЗУ МК. Событие сброса по сигналу блока пониженного напряжения питания отмечается специальным битом в одном из регистров МК. Следовательно, программно анализируя этот бит после сброса МК, можно установить, что данные целы, и продолжить выполнение программы.

2.12 Сторожевой таймер

Если, несмотря на все принятые меры, МК все же «завис», то на случай выхода из этого состояния все современные МК имеют встроенный модуль сторожевого таймера.

Основу сторожевого таймера составляет многоразрядный счетчик. При сбросе МК счетчик обнуляется. После перехода МК в активный режим работы значение счетчика начинает увеличиваться независимо от выполняемой программы. При достижении счетчиком максимального кода генерируется сигнал внутреннего сброса, и МК начинает выполнять рабочую программу сначала.

Принцип действия сторожевого таймера схематично показан на рисунке 2.4.

Для исключения сброса по переполнению сторожевого таймера рабочая программа МК должна периодически сбрасывать счетчик путем исполнения специальной команды или посредством записи некоторого кода в один из регистров специальных функций. Тогда при нормальном порядке выполнения рабочей программы переполнения счетчика сторожевого таймера не происходит, и

он не оказывает влияния на работу МК. Однако, если исполнение рабочей программы было нарушено, например, вследствие «зависания», то велика вероятность того, что счетчик не будет сброшен вовремя. Тогда произойдет сброс по переполнению сторожевого таймера, и нормальный ход выполнения рабочей программы будет восстановлен.

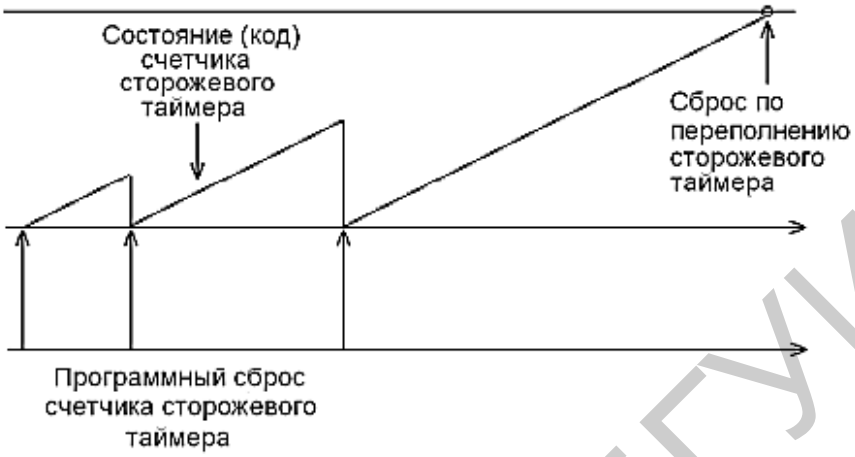

Рисунок 2.4 – Принцип действия сторожевого таймера

Модули сторожевых таймеров конкретных МК могут иметь различные особенности:

- в ряде МК векторы внешнего сброса и сброса по переполнению сторожевого таймера совпадают. Это не позволяет выявить причину сброса программным путем и затрудняет написание рабочей программы. Более высокоуровневые МК имеют либо различные векторы сброса, либо отмечают событие сброса по переполнению сторожевого таймера установкой специального бита в одном из регистров специальных функций;

- в некоторых МК при переходе в один из режимов пониженного энергопотребления, когда рабочая программа не выполняется, автоматически приостанавливается работа сторожевого таймера. В других МК сторожевой таймер имеет независимый тактовый генератор, который продолжает функционировать и в режиме ожидания. В этом случае необходимо периодически выводить МК из состояния ожидания для сброса сторожевого таймера.

Использование сторожевого таймера существенно повышает способность к самовосстановлению системы на основе МК.

2.13 Дополнительные периферийные модули МК

Описанные выше модули составляют так называемый базовый комплект МК и входят в состав любого современного МК. Очевидна необходимость включения в состав МК дополнительных модулей, состав и возможности которых определяются конкретной решаемой задачей. Среди таких дополнительных модулей следует прежде всего отметить:

- модули последовательного ввода/вывода данных;
- модули аналогового ввода/вывода.

Модули последовательного ввода/вывода. Наличие в составе 8-разрядного МК модуля контроллера последовательного ввода/вывода стало в последнее время обычным явлением. Задачи, которые решаются средствами модуля контроллера последовательного ввода/вывода, можно разделить на три основные группы:

- связь встроенной микроконтроллерной системы с системой управления верхнего уровня, например, с персональным компьютером. Чаще всего для этой цели используются интерфейсы RS-232C и RS-485;
- связь с внешними по отношению к МК периферийными интегральными схемами, а также с датчиками физических величин с последовательным выходом. Для этих целей используются интерфейсы I2C, SPI, а также нестандартные протоколы обмена;
- интерфейс связи с локальной сетью в мультимикроконтроллерных системах. В системах с числом МК до пяти обычно используются сети на основе интерфейсов I2C, RS-232C и RS-485 с собственными сетевыми протоколами высокого уровня. В более сложных системах все более популярным становится протокол CAN.

С точки зрения организации обмена информацией упомянутые типы интерфейсов последовательной связи отличаются:

- режимом передачи данных (синхронный или асинхронный);
- форматом кадра (число бит в посылке при передаче байта полезной информации);
- временными диаграммами сигналов на линиях (уровни сигналов и положение фронтов при переключениях).

Число линий, по которым происходит передача в последовательном коде, обычно равно двум (I2C, RS-232C, RS-485) или трем (SPI и некоторые нестандартные протоколы). Данное обстоятельство позволяет спроектировать модули контроллеров последовательного обмена таким образом, чтобы с их помощью на аппаратном уровне можно было реализовать несколько типов последовательных интерфейсов. При этом режим передачи (синхронный или асинхронный) и формат кадра поддерживаются на уровне логических сигналов, а реальные физические уровни сигналов для каждого интерфейса получают с помощью специальных интегральных схем, которые называют приемопередатчиками, конвертерами, трансиверами.

Среди различных типов встроенных контроллеров последовательного обмена, которые входят в состав тех или иных 8-разрядных МК, сложился стандарт «де-факто» – модуль UART (Universal Asynchronous Receiver and Transmitter) – универсальный асинхронный приемопередатчик. Однако большинство модулей UART, кроме асинхронного режима обмена, способны также реализовать и режим синхронной передачи данных. Не все производители МК

используют термин UART для обозначения типа модуля контроллера последовательного обмена. Иногда его называют SCI (Serial Communication Interface).

Модули типа UART в асинхронном режиме работы позволяют реализовать протокол обмена для интерфейсов RS-232C, RS-422A, RS-485, в синхронном режиме – нестандартные синхронные протоколы обмена, и в некоторых моделях – SPI.

Протоколы интерфейсов локальных сетей на основе МК (I2C и CAN) отличает более сложная логика работы. Поэтому контроллеры CAN интерфейса всегда выполняются в виде самостоятельного модуля. Интерфейс I2C с возможностью работы как в ведущем, так и ведомом режиме, также обычно поддерживается специальным модулем.

В последнее время появилось большое количество МК со встроенными модулями контроллеров CAN и модулями универсального последовательного интерфейса периферийных устройств USB (Universal Serial Bus). Каждый из этих интерфейсов имеет достаточно сложные протоколы обмена, для ознакомления с которыми следует обращаться к специальной литературе.

Модули аналогового ввода/вывода. Необходимость приема и формирования аналоговых сигналов требует наличия в МК модулей аналогового ввода/вывода.

Простейшим устройством аналогового ввода в МК является встроенный компаратор напряжения. Компаратор сравнивает входное аналоговое напряжение с опорным потенциалом V_{REF} и устанавливает на выходе логическую единицу, если входное напряжение больше опорного. Компараторы удобнее всего использовать для контроля определенного значения входного напряжения, например в термостатах.

Более широкие возможности для работы с аналоговыми сигналами дает АЦП, встроенный в МК. Чаще всего он реализуется в виде модуля многоканального АЦП, предназначенного для ввода в МК аналоговых сигналов с датчиков физических величин и преобразования этих сигналов в двоичный код.

Для подключения одного из источников аналоговых сигналов ко входу АЦП служит специальный многоканальный аналоговый коммутатор. При этом выбор источника сигнала для преобразования осуществляется посредством записи номера канала коммутатора в соответствующие разряды регистра управления АЦП.

Обычно аналого-цифровой преобразователь выполнен по методу последовательного приближения. Практически во всех моделях 8-разрядных МК разрядность АЦП также составляет 8 разрядов. Соответственно формат представления результатов измерения АЦП – однобайтовый. Исключение составляют лишь модули АЦП микроконтроллеров для управления преобразователями частоты для электроприводов, разрешающая способность которых равна 10 разрядам.

Время преобразования для типовых модулей АЦП микроконтроллеров составляет от единиц до десятков микросекунд. Момент завершения каждого

цикла преобразования отмечается установкой триггера готовности данных. Если прерывания от модуля АЦП разрешены, то генерируется запрос на прерывания. Как правило, чтение регистра результата сбрасывает триггер готовности.

Большинство модулей АЦП имеют только режим программного запуска: установка одного из битов регистра режима запускает очередное измерение. Наиболее универсальные модули АЦП имеют также режим автоматического запуска, при котором после завершения одного цикла преобразования немедленно начинается следующий. Однако данные измерения каждого цикла должны быть считаны программным способом.

Цифроаналоговые преобразователи в составе МК в отличие от АЦП встречаются гораздо реже. Функция цифроаналогового преобразователя реализуется средствами модуля программируемого таймера в режиме ШИМ. На одном из выводов МК формируется высокочастотная импульсная последовательность с регулируемой длительностью импульса. Полученный сигнал сглаживается фильтром нижних частот на операционном усилителе. Разрешающая способность такого ЦАП определяется дискретностью регулирования коэффициента заполнения в режиме ШИМ.

3 ПРИМЕРЫ РАЗРАБОТОК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ МИКРОПРОЦЕССОРНЫХ СИСТЕМ УПРАВЛЕНИЯ

Все множество задач, решаемых микропроцессорными системами управления, можно разделить на три подгруппы: задачи ввода информации, задачи арифметико-логических преобразований информации и задачи вывода информации.

3.1 Ввод информации в микропроцессорную систему

К задачам ввода относятся как задачи получения от оператора информации о требуемых режимах и настройках системы, так и задачи выделения информации, поступающей в виде сигналов (амплитудных, частотных и др.) с датчиков объекта управления.

Опрос пользовательского пульта. В большинстве встраиваемых систем управления необходимо организовывать связь с оператором для указания параметров работы системы.

Рассмотрим задачу определения нажатой клавиши на шестнадцатиклавишном пульте, схема подключения которого к микроконтроллеру Infineon 80C515 показана на рисунке 3.1.

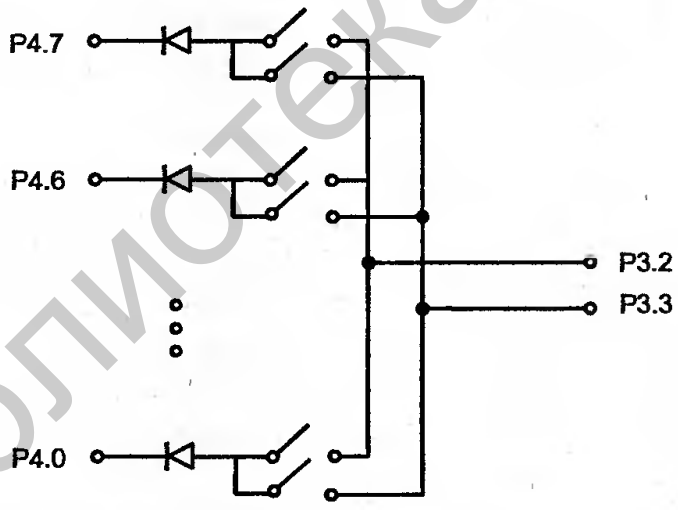

Рисунок 3.1– Схема подключения пользовательского пульта

Клавиши объединены попарно в матрицу размерностью 8×2. Столбцы матрицы клавиатуры соединены с линиями порта P4. В реализованной схеме низкий уровень на одной из выходных линий порта P4 обеспечивает опрос соответствующего столбца матрицы. Строки матрицы подключены к выводам P3.3 и P3.2 порта P3. Низкий уровень на этих линиях идентифицирует нажатые клавиши, опрашиваемые в текущий момент времени.

Для определения состояния клавиатуры необходимо сформировать входные сигналы опроса клавиатуры и считать ее состояние. Опрос клавиатуры удобно выполнять, организовав выдачу последовательности «бегущий нуль» по столбцам клавиатуры, считывая при этом значение строк клавиатуры в каждый такт формирования нового входного воздействия. Совокупность значений строк матрицы клавиатуры назовем картой состояния.

Схема алгоритма программы приведена на рисунке 3.2, а ее листинг – на рисунке 3.3.

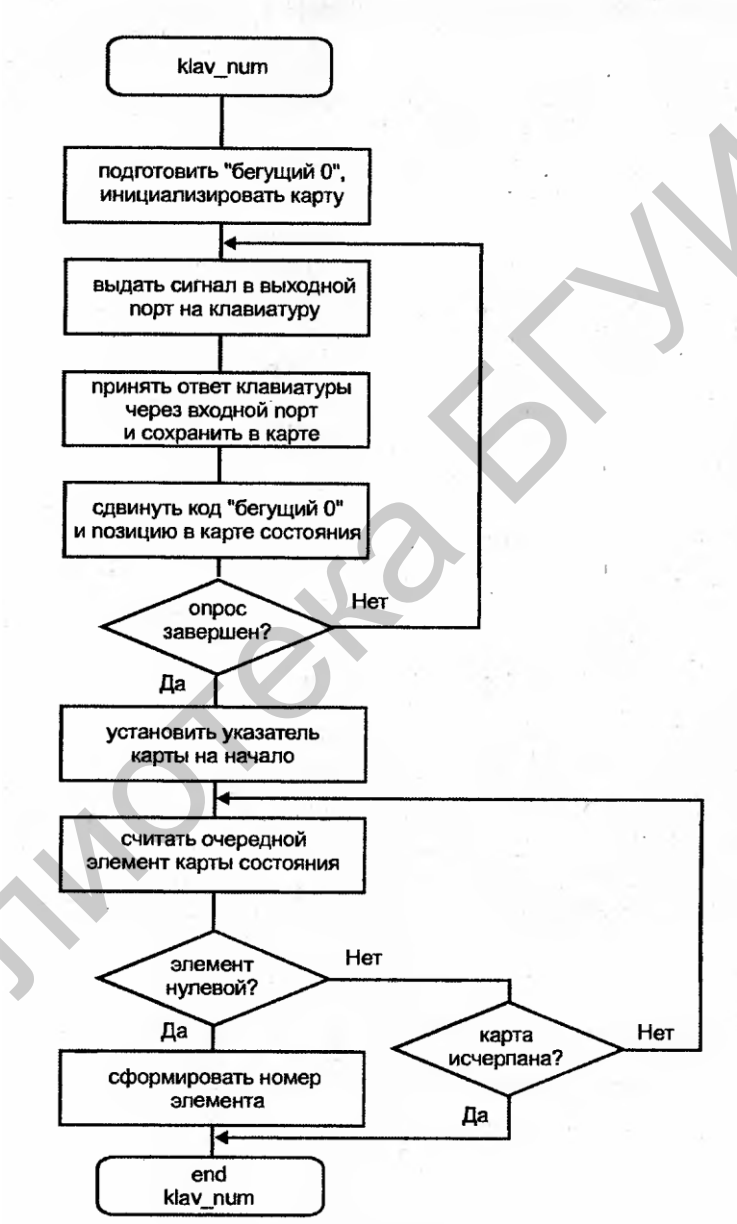

Рисунок 3.2 – Схема алгоритма программы опроса пульта

Номер нажатой клавиши определяется как (X - start)*2 + 1 для клавиш, отображаемых на разряд P3.3, и как (X - start)*2 + 2 для клавиш, отображаемых на разряд P3.2, где X – номер нулевого элемента карты состояния, start – адрес начала карты.

```
num:        equ 40 h                ; номер нажатой клавиши
map_start:  equ 30h                 ;начало области хранения карты
                                    ;клавиатуры
klav:       push psw                ;сохранение используемых
            push 0                  ;ресурсов МК
            push a

                                    ; чтение карты в память МК
            mov a, #07Fh            ; подготовка «бегущего нуля»
            mov r0,#map_start       ;адрес начала карты состояния
                                    ;клавиатуры
$scan:      mov 0E8h,a              ;выдача сигнала в порт 4
            setb p3.3
            setb p3.2
            mov @r0, P3             ;чтение ответа из порта 3
            inc r0                  ;в очередную позицию карты
            setb c
            rrc a                   ;подготовка нового опроса
            cjne a,#11111111b, $scan

                                    ;дешифрация карты
            mov r0,#map_start
$check:     mov a, @r0
            jnb acc.3, $odd         ; выявление нажатой клавиши
                                    ;с нечетным номером

            jnb acc.2, $even        ; выявление нажатой клавиши
                                    ;с четным номером
            inc r0                  ;просмотр карты далее
            cjne r0, #(map_start+8), $check

            mov a, #0               ;нажатых клавиш нет
            ljump klav_end
$odd:       mov a, r0               ;вычисление номера нажатой клавиши
            clr c                   ;для нечетных позиций
            subb a,# map_start
            mov b, #2
            mul ab
            add a,#1
            ljump klav_end
$even:      mov a, r0               ;вычисление номера нажатой клавиши
            clr c                   ;для четных позиций
            subb a,# map_start
            mov b, #2
            mul ab
            add a,#2
klav_end:   mov num, a              ;запись в результирующую ячейку
            pop a                   ;восстановление используемых
            pop 0                   ;ресурсов МК
            pop psw
            ret
```

Рисунок 3.3 – Листинг программы опроса пульта

Опрос датчиков аналоговых величин. В системах управления одной из типовых задач является задача определения показаний датчиков объекта, представляющих информацию в виде аналогового сигнала – уровня напряжения.

Перед разработчиком в данном случае возникают две задачи: масштабирования сигналов и аналого-цифрового преобразования.

Первая задача решается применением схемотехнических решений (внешних по отношению к микроконтроллеру узлов масштабирования, выполненных, например, на базе операционных усилителей). Вторая задача решается либо схемотехнически (в случае отсутствия встроенного в микроконтроллер АЦП), либо программно (при наличии данного периферийного узла).

Рассмотрим подсистему измерения уровня освещенности, использующую в качестве датчика фотодиод. Принципиальная схема устройства сопряжения с микроконтроллером 80C515 показана на рисунке 3.4.

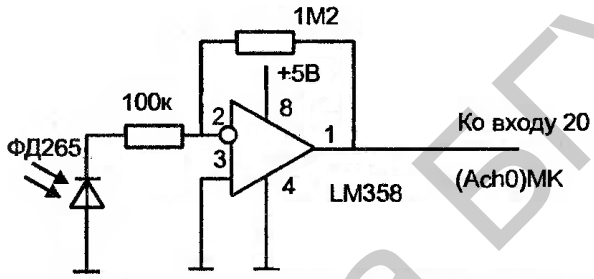

Рисунок 3.4 – Сопряжение датчика с микроконтроллером

Блок-схема программы приведена на рисунке 3.5, а ее листинг – на рисунке 3.6.

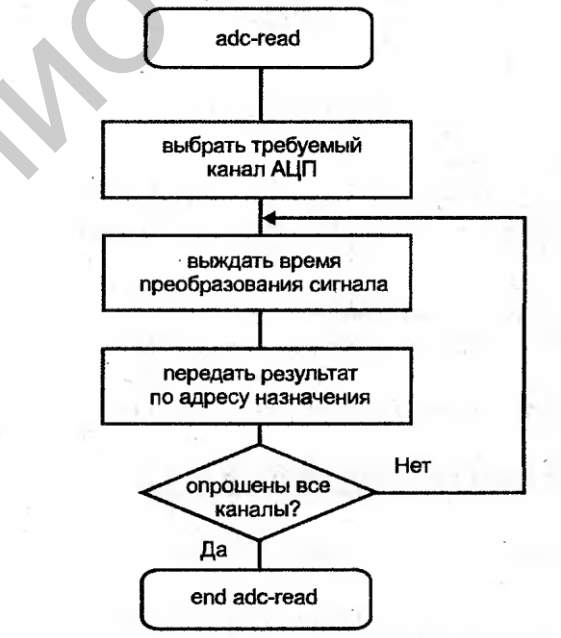

Рисунок 3.5 – Схема алгоритма программы опроса датчиков аналоговых сигналов

В данном случае проводится опрос датчика, подключенного к нулевому каналу АЦП. Диапазон масштабированных операционным усилителем входных сигналов составляет 0–5 В.

```
U1:          equ 40h                ;напряжение с датчика

U1_read:     push 0                 ;процедура опроса датчиков
             mov r0 ,#0             ;выбор нулевого канала
             lcall adc_switch
             mov 0DAh, #0h          ;задание диапазона
             lcall adc_delay        ;задержка
             mov U1, 0D9h           ;сохранение в U1
             pop 0
             ret

adc_switch:  mov a,0D8h             ;настройка АЦП:
             anl a, #11111000b      ; непрерывное преобразование
             orl a, #00001000b
             orl a, r0              ; выбор канала
             mov 0D8h, a
             ret

adc_delay:   push 0                 ; выдержка паузы для преобразования
             mov r0, #20            ; в АЦП
             djnz r0, $
             pop 0
             ret
```

Рисунок 3.6 – Листинг программы опроса датчиков аналоговых сигналов

Определение длительности временных интервалов. Важной задачей при построении управляющих систем является задача определения времени продолжительности события, представленного двоичным сигналом или совокупностью двоичных сигналов на входах микроконтроллера.

Рассмотрим задачу определения времени продолжительности события, начало которого характеризуется переходом из состояния логической единицы в состояние логического нуля двоичного сигнала U1, а конец – переходом из состояния логической единицы в состояние логического нуля двоичного сигнала U2.

Для удобства решения задачи подадим сигнал U1 на вход первого запроса прерывания микроконтроллера, а сигнал U2 – на вход нулевого запроса прерывания (рисунок 3.7).

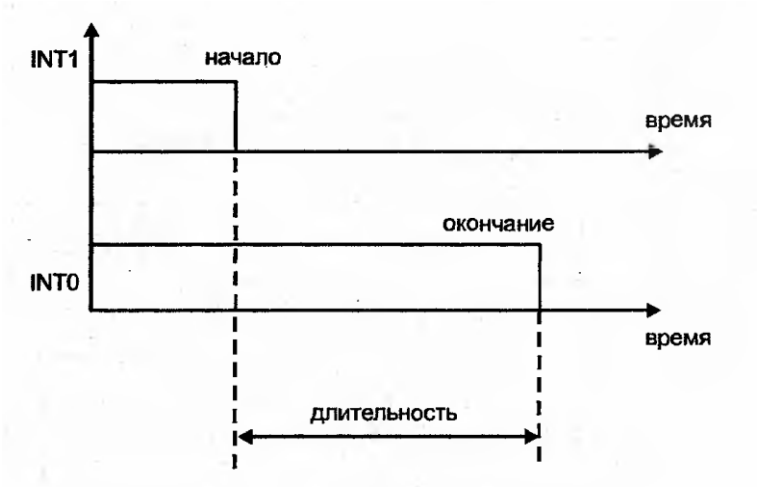

Рисунок 3.7 – Сигналы начала и окончания события

Для снижения влияния подпрограмм определения времени события на прочие процессы, реализуемые в системе, следует решать задачу с использованием системы прерываний. По срезу сигнала на входе INT1 возникает прерывание, обработчик которого запускает таймер, отсчитывающий кванты времени протяженности события, по срезу сигнала на входе INT0 возникает прерывание, обработчик которого прекращает счет, блокируя таймер. Для повышения точности определения времени в ситуации возможных многократных смен состояний входов последующие прерывания от источников стартового и стопового импульсов блокируются до завершения текущего цикла подсчета времени.

Листинг программы представлен на рисунке 3.8, а схема программного комплекса в виде совокупности трех подпрограмм-обработчиков прерываний приведена на рисунке 3.9.

```
cnt_hi:      equ 30h          ; счетчик текущего времени
cnt_lo:      equ 31h
last_tim_h:  equ 40h          ; длительность предыдущего цикла
last_tim_l:  equ 41h

main:        lcall init       ; основная программа
loop:        …
             sjmp loop

init:                         ; инициализация переменных и устройств
             mov cnt_hi, #0   ; обнуление счетчиков
             mov cnt_lo, #0
             mov last_tim_h, #0
             mov last_tim_l, #0
```

Рисунок 3.8 – Листинг программы определения длительности события, лист 1

```
                        mov a, TMOD             ; таймер Т0 в режиме перезагрузки
                        anl a, #11110000b       ; будет отмерять интервалы
                        orl a, #00000001b       ; времени в 100 мкс
                        mov TMOD, a
                        mov TH0, #155
                        mov TL0, #155
                        setb et0                ; разрешение работы
                        setb ex1                ; обработчиков необходимых
                        setb ea                 ; типов прерываний
                        ret
; обработка события старт-импульса
int1:                   mov cnt_hi, #0          ;начало нового цикла измерений
                        mov cnt_lo, #0
                        mov TL0,#155
                        setb tr0                ; запуск таймера
                        setb ex0                ; подготовка к распознаванию
                                                ;стоп-импульса
                        clr ex1                 ; блокировка прерывания данного
                                                ;типа
                        reti
; обработка события стоп-импульса
int0:                   clr tr0                 ;запрет дальнейшего счета
                        clr ex0                 ;блокировка прерывания данного
                                                ;типа
                        mov last_tim_h, cnt_hi  ;сохранение измеренного времени
                        mov last_tim_l, cnt_lo
                        setb ex1                ;подготовка к распознаванию
                                                ;очередного старт-импульса
int0_ex:                reti
;-------------------------------------------------------------------------------------
tim0:                   push psw
                        push a
                        inc cnt_lo              ;увеличение значения длительности
                        mov a, cnt_lo           ;импульса
                        jnz tim0_ex
                        inc cnt_hi
tim0_ex:                pop a
                        pop psw
                        reti
; переходы к обработчикам необходимых типов прерываний
                        org 0003h
                        ljmp int0
                        org 000bh
                        ljmp int0
                        org 0013h
                        ljmp int1
```

Рисунок 3.8 , лист 2

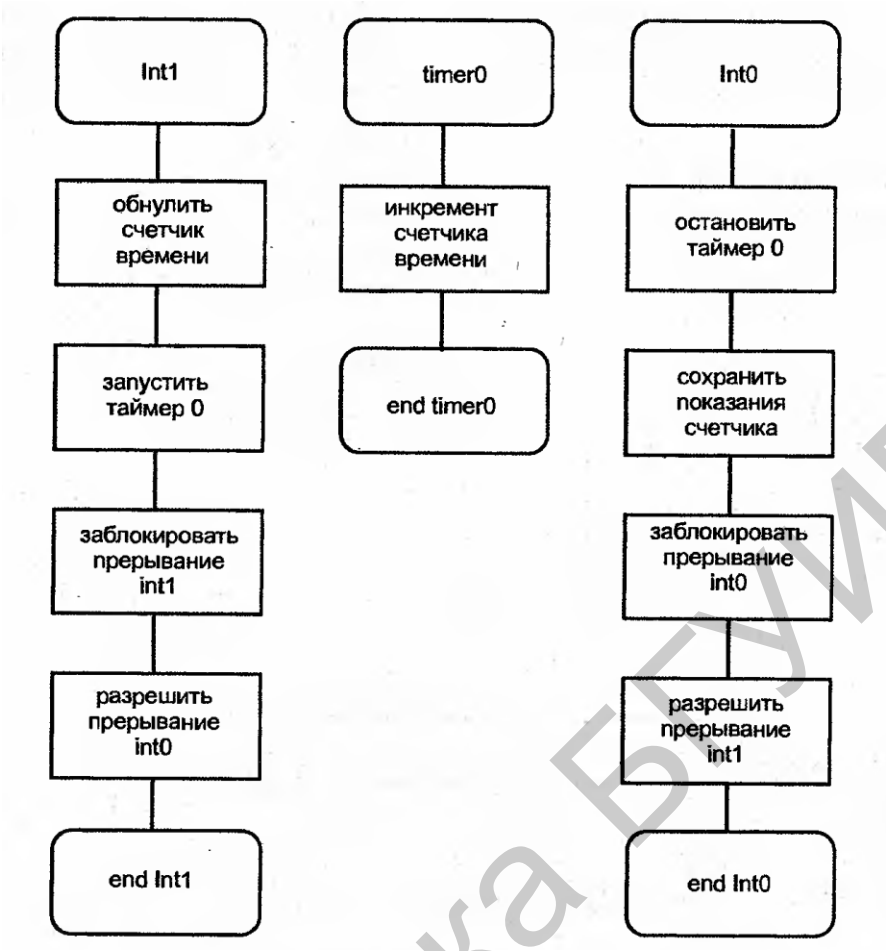

Рисунок 3.9 – Схемы алгоритмов программ определения длительности событий

В переменных *last_tim_h* и *last_tim_l* хранятся соответственно старший и младший байты длительности последнего завершившегося события в квантах времени по 100 мкс.

3.2 Вывод информации из микропроцессорной системы управления

В номенклатуру задач вывода информации входят как задачи вывода управляющей информации, поступающей в виде различных сигналов (амплитудных, частотных и др.), на объект управления, так и организация вывода информации на пульт оператора.

Вывод цифровых кодовых последовательностей. В большинстве встраиваемых систем управления необходимо организовывать связь с оператором для вывода информации о параметрах работы системы.

Рассмотрим задачу вывода алфавитно-цифровой информации на жидкокристаллический алфавитно-цифровой индикатор (ЖКИ) модели DM2021 фирмы Sanyo. Данная модель ЖКИ имеет поле вывода информации размером две строки по двадцать символов в каждой. ЖКИ содержит видеопамять, в которой хранятся отображаемые символы, а также обладает собственной системой

управления жидкокристаллической панелью. Наличие указанных узлов существенно упрощает работу с ЖКИ, т. к. собственно управление отображением точек – элементов изображения – производится автоматически в соответствии с поданной на ЖКИ командой.

ЖКИ содержит восьмиразрядную шину команд-данных и шину управления, в состав которой входят одноразрядные линии разрешения программирования (E), выбора типа посылки «команда-данные» (RS) и выбора направления передачи данных «чтение-запись» (RW). Подключение ЖКИ к микроконтроллеру 80C515 показано на рисунке 3.10. Порт P4 предназначен для организации шины команд-данных, а старшие три бита порта P1 предназначены для организации шины управления. Переменный резистор необходим для регулировки контрастности изображения.

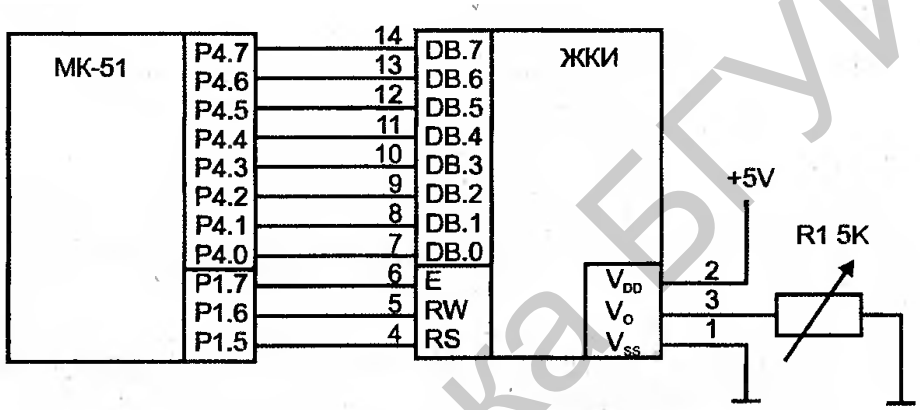

Рисунок 3.10 – Сопряжение ЖКИ с микроконтроллером

Процедура записи в ЖКИ выполняется в три этапа: на шине DB устанавливается информация, отражающая подаваемые в ЖКИ команды-данные, затем устанавливаются необходимые значения на линиях RW и RS и, наконец, на входе E формируется переход от высокого логического уровня к низкому. Для возврата системы управления ЖКИ в исходное состояние следует перевести вход E в состояние логической единицы.

Не приводя всей системы команд ЖКИ, остановимся на следующих из них:

38h – установка восьмибитного режима обмена с ЖКИ, использование для вывода обеих строк с размером символа 5x7 точек; 0Ch – активизация всех знакомест ЖКИ в режиме погашенного курсора; 80h – установка адреса, начиная с которого записываемые в ЖКИ данные будут последовательно располагаться в видеопамяти.

Схема алгоритма и текст программы, осуществляющей вывод на ЖКИ сорока символов, коды которых расположены во внешней памяти данных микроконтроллера, начиная с адреса 0*FF*D0h, приведены соответственно на рисунках 3.11 и 3.12.

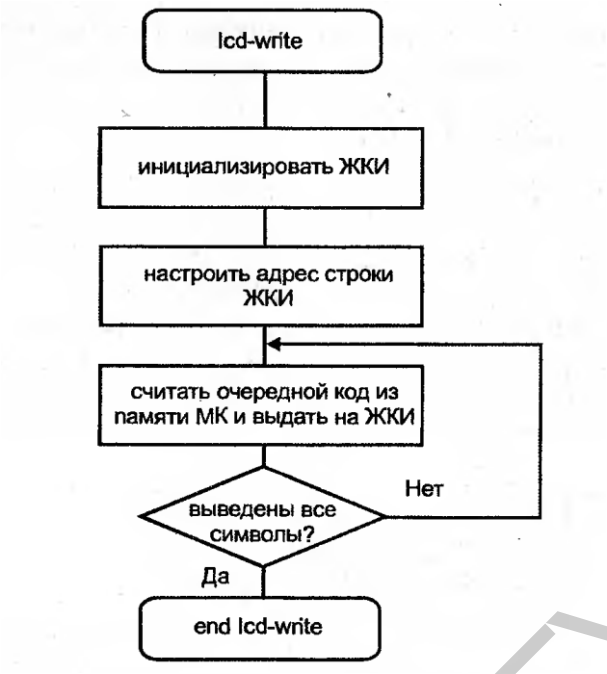

Рисунок 3.11 – Схема алгоритма программы вывода информации на ЖКИ

```
indic:
indic_init:         mov indic_w1, #0            ;инициализация ЖКИ
                    mov indic_w0, #38h
                    lcall indic_wr
                    mov indic_w0, #0Ch
                    lcall indic_wr
                    mov indic_w0, #80h          ;установка адреса первого
                    lcall indic_wr              ;символа первой строки
                    mov indic_w1, #1
                    mov dptr, #0FFD0h
indic_data_wr_1:    movx a, @dptr               ;вывод символа первой строки
                    mov indic_w0, a
                    lcall indic_wr
                    inc dptr
                    mov a, dpl
                    cjne a,#0E4h,indic_data_wr_1
                    mov indic_w1,#0
                    mov indic_w0, #0C0h         ;установка адреса первого
                    lcall indic_wr              ;символа второй строки
                    mov indic_w1, #1
indic_data_wr_2:    movx a, @dptr               ;вывод символа второй строки
                    mov indic_w0, a
                    lcall indic_wr
                    inc dptr
                    mov a, dpl
```

Рисунок 3.12 – Листинг программы вывода данных на ЖКИ, лист 1

```
                cjne a,#0F8h,indic_data_wr_2
indic_exit:     ret
indic_wr:       mov 0e8h, indic_w0          ;процедура записи в ЖКИ
                setb p1.7
                clr p1.6
                mov a, indic_w1
                mov c,acc.0
                mov p1.5, c
                lcall indic_delay
                clr p1.7
                lcall indic_delay
                setb p1.7
                ret
indic_delay:    nop                         ; процедура программной
                nop                         ;задержки
                nop
                nop
                nop
                nop
                nop
                nop
                ret
                org 0FFD0h
Data:           db 'It is test example.'    ;пример организации
                db '0123456789ABCDEF!@#$'   ;выводимых данных
```

Рисунок 3.12, лист 2

Вывод ШИМ-сигналов. В ряде задач управления возникает необходимость выдачи на объект управляющих сигналов определенной интенсивности (либо заданной мощности за некоторый период). Как правило, для объектов с аналоговым управлением применяется формирование управляющих сигналов с изменяемой амплитудой.

Один из вариантов реализации такого управления заключается в использовании широтно-импульсной модуляции (ШИМ). В данном варианте на одном из разрядов выходного порта микроконтроллера формируются периодические импульсные сигналы с постоянной частотой следования и изменяемым отношением времени длительности импульса ко времени длительности паузы, которое определяет интенсивность управляющего сигнала.

Рассмотрим реализацию ШИМ на примере управления частотой вращения двигателя постоянного тока. Схема устройства сопряжения микроконтроллера 80С515 с двигателем показана на рисунке 3.13. Схема алгоритма программы формирования ШИМ-сигналов приведена на рисунке 3.14, а ее текст – на рисунке 3.15.

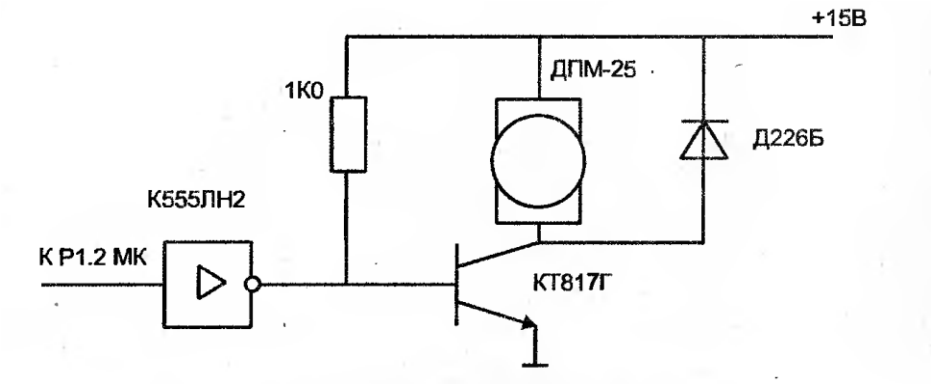

Рисунок 3.13 – Схема устройства сопряжения

Рисунок 3.14 – Схема алгоритма программы ШИМ-управления

```
main:   lcall init_pwm          ; инициализация ШИМ
loop:   …
        lcall pwm_contr         ; задание ШИМ
        sjmp loop

init_pwm: mov old_pwm, #255     ;предыдущая константа перезагрузки
        orl p1, #00001000b      ;вывод через третий регистр меток
                                ;– через разряд P1.3
        mov cbh, #0e0h          ;константа перезагрузки соответствует
        mov cah, #080h          ;частоте ШИМ в 124 Гц
        mov c8h, #00010001b     ;таймер 2 в режиме неуправляемого счета,
                                ;с выводом ШИМ и автоперезагрузкой
        mov c1h, #10000000b     ;разрешен выв. через третий регистр меток
```

Рисунок 3.15 – Листинг программы ШИМ-управления, лист 1

```
                mov c6h,#80h            ;младший байт третьего регистра меток
                ret
pwm_contr:      mov a, pwm_var          ;прием управления
                cpl a                   ;расчет константы времени
                mov b, #8
                div ab
                cjne a,old_pwm,pwm_ch
                ljmp pwm_exit
pwm_ch:         mov old_pwm,a           ;в случае изменившегося управления
                add a,#224
                mov c7h, a              ;вывод в старший байт регистра
pwm_exit:       ret
```

Рисунок 3.15, лист 2

Вывод сигналов с временным сдвигом. Достаточно часто возникает задача формирования временного интервала с управляемой длительностью, начало которого определяется внешним старт-сигналом (задача синхронизации).

Требуемая временная диаграмма приведена на рисунке 3.16.

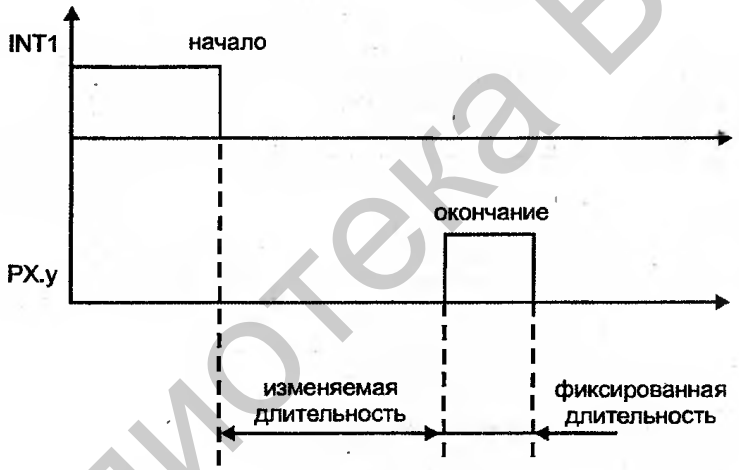

Рисунок 3.16 – Формируемый временной интервал

Для возможности гибкого использования программного комплекса, реализующего данный вариант управления, целесообразно организовать его с использованием возможностей, предоставляемых системой прерываний. При возникновении запускающего события обработчик прерывания INT0 запускает таймер 0, отсчитывающий заданный основной программой временной интервал, по истечении которого обработчик прерывания таймера 0 на определенном разряде выходного порта формирует высокий логический уровень. После этого обработчик перезапускает таймер с такими настройками, чтобы выдержать фиксированный интервал длительности импульса. Выполняясь вторично, обработчик прерывания таймера 0 на заданной выходной линии формирует низкий логический уровень.

Схема алгоритма управляющей программы приведена на рисунке 3.17, а листинг программы – на рисунке 3.18.

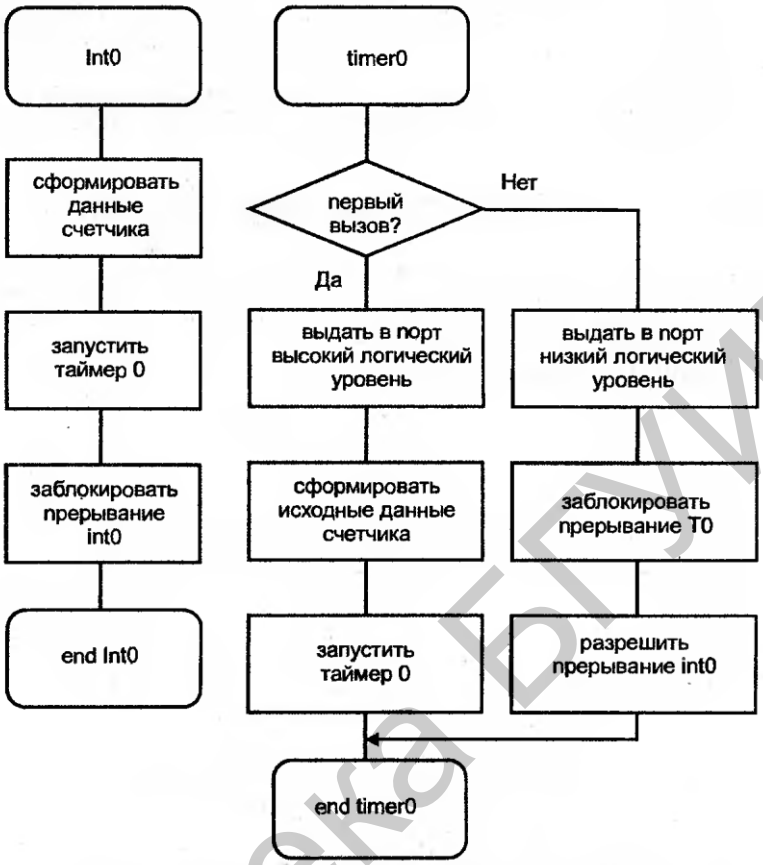

Рисунок 3.17 – Схема алгоритма программы выдержки временного интервала

```
kupr:       equ 30h

main:       lcall init
loop:       …
            sjmp loop
init:       mov a, tmod          ;таймер 0 – 16-битный счетчик
            anl a, #11110000b
            orl a, #00000001b
            mov tmod, a
            clr P1.2             ;обнуление выходной линии
            clr F0               ;номер прохода: 0 – первый, 1 – второй
            setb it0             ;настройка системы прерываний
            setb ex0
            clr et0
            clr tr0
```

Рисунок 3.18 – Листинг программы выдержки временного интервала, лист 1

```
                setb ea
                ret
;подпрограмма обработки прерывания INT0, синхронизация по старт-импульсу
sync_impulse:   push 0
                push acc
                push b
                push psw

                mov a,kupr          ;определение временной выдержки
                mov b,#8            ;максимальное время – 10 мс
                div ab
                add a, #7           ;минимальное время – 2 мс
                mov r0,a
                mov a,#255
                clr c
                subb a,r0
                mov th0,a           ;установка начальных показаний
                mov tl0,#030h       ;таймера
                setb et0            ;настройка системы прерываний
                setb tr0
                clr ex0

                pop psw
                pop b
                pop acc
                pop 0
                reti
;подпрограмма обработки прерывания T0
switch:         jb F0, off          ;анализ номера вхождения в обработчик
on:                                 ;выдать импульс на 240 мкс
                setb F0
                setb P1.2
                mov th0,#255
                mov tl0,#15
                sjmp switch_exit
off:                                ;завершить импульс
                clr F0
                clr P1.2
                setb ex0            ;подготовка к новому циклу
                clr et0
                clr tr0
switch_exit:    reti
;переход к обработчикам прерываний
                org 0003h
                ljmp sync_impulse
                org 000bh
                ljmp switch
```

Рисунок 3.18, лист 2

Программа формирует импульс длительностью 240 мкс, выдаваемый спустя 2... 10 мс относительно начала запускающего перепада на входе INT0 в зависимости от величины кода *kupr*.

3.3 Программная реализация алгоритмов управления

Позиционное и контурное управление. Рассматривая построение автоматизированных систем, можно выделить два метода формирования управляющего воздействия. В простейших случаях это может быть реализовано с помощью реле, которое формирует скачкообразное управляющее воздействие, например включение/выключение нагревателя или привода какого-либо механизма, при этом выходом является одна координата. Такой закон управления называют позиционным. В более сложных случаях управляющее воздействие изменяется плавно, по нескольким координатам, с учетом возможностей системы. Это может быть, к примеру, управление пером графопостроителя (рисунок 3.19). Такой закон управления называется контурным.

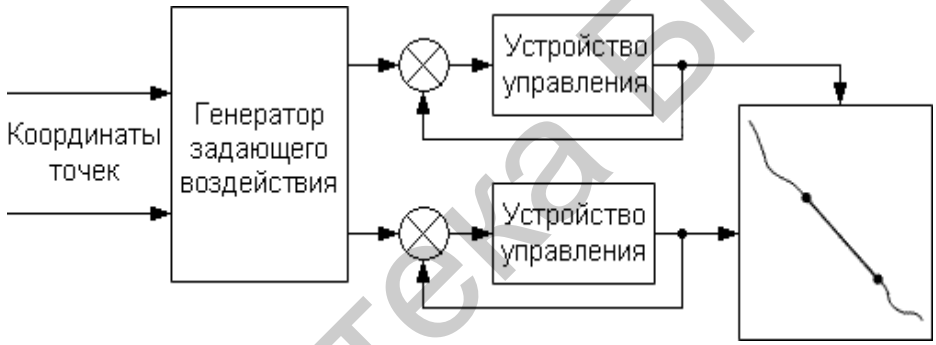

Рисунок 3.19 – Контурное управление

Позиционное и контурное управление имеет большое значение при управлении различными электромеханическими системами, например роботами. При контурном управлении необходимо обеспечить не только переход в конечную точку, но и движение по заданной траектории.

Программная реализация алгоритмов линейной интерполяции. Алгоритмы линейной интерполяции используются для согласованного управления несколькими координатами (x, y, z) в станках с ЧПУ и роботах. Для этого используется два способа:

1) при помощи параметрического представления прямой;

2) на основе оценочной функции.

Рассмотрим *первый* способ (рисунок 3.20).

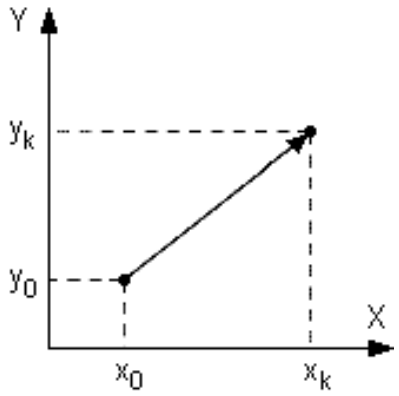

Рисунок 3.20 – Параметрическое представлении прямой

Для движения по прямой из точки (x_0, y_0) в точку (x_k, y_k) можно использовать следующий алгоритм:

for $t = 0 : \Delta t : T$
$x(t) = x_0 + V_x \cdot t$
$y(t) = y_0 + V_y \cdot t$
out (x, y)
end

К достоинству алгоритма можно отнести возможность изменения скорости в процессе движения, которая может иметь треугольный либо трапецеидальный профиль. Однако его программная реализация для микроконтроллеров, у которых отсутствует команда умножения, сопряжена с определенными трудностями.

Идея *второго* способа заключается в дискретном изменении координат X и Y и вычислении на каждом шаге оценочной функции f. За начальное положение примем начало координат. Пусть интерполируемый наклонный отрезок лежит в первом квадранте. Тогда его уравнение имеет вид $y = (\Delta y/\Delta x) \cdot x$. Оценочная функция $f = y - (\Delta y/\Delta x) \cdot x$. На интерполируемом отрезке прямой функция $f = 0$, выше этой прямой – $f > 0$, ниже – $f < 0$ (рисунок 3.21), поэтому направление движения определяется знаком оценочной функции, которая пересчитывается на каждом шаге. Так как координаты x и y могут меняться только на дискретное значение, то будем измерять x и y в целых числах.

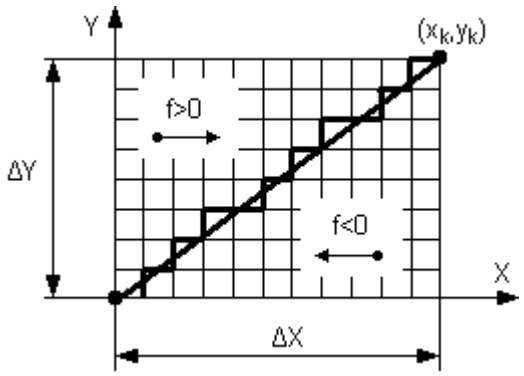

Рисунок 3.21 – Представление прямой методом оценочной функции

Алгоритм, основанный на вычислении оценочной функции, состоит в следующем:

– для текущих значений x_i и y_i вычисляется оценочная функция $f(x_i, y_i) = y_i - (\Delta y / \Delta x) \cdot x_i$;

– если функция $f(x_i, y_i) < 0$, то шаг делается по координате Y: $y_j + 1 = y_j + 1$;

– если функция $f(x_i, y_i) \geq 0$, то шаг делается по координате X: $x_j + 1 = x_j + 1$;

– вычисляется новое значение оценочной функции, и процесс повторяется. Вычисление этих операций заканчивается при попадании в точку Δx, Δy.

Поскольку в рассмотренном алгоритме важно не знание оценочной функции, а ее знак, то можно использовать функцию

$$f'(x_i, y_i) = y_i \Delta x - x_i \Delta y.$$

Пусть на j-м шаге алгоритма

$$f'(x_i, y_i) = y_j \Delta x - x_j \Delta y.$$

При шаге по оси Y

$$x_{j+1} = x_j; \;\; y_{j+1} = y_j + 1; \tag{3.1}$$

$$f'_{j+1}(x_{j+1}, y_{j+1}) = (y_{j+1} + 1)\Delta x - x_j \Delta y = f'_j(x_j, y_j) + \Delta x.$$

При шаге по оси X

$$x_{j+1} = x_j + 1; \;\; y_{j+1} = y_j; \tag{3.2}$$

$$f'_{j+1}(x_{j+1}, y_{j+1}) = y_j \Delta x - (x_j + 1)\Delta y = f'_j(x_j, y_j) - \Delta y.$$

Формулы (3.1) и (3.2) позволяют вычислить оценочную функцию на каждом шаге алгоритма с использованием только операций сложения и вычитания. В программе предполагается, что отрезок прямой располагается в первом квадранте и имеет начальное значение $f(0, 0) = 0$. При интерполяции отрезков прямых, лежащих в других квадрантах, алгоритм остается тем же. Для его полной реализации необходимо учитывать номер квадранта, в котором лежит отрезок, увеличивая или уменьшая при этом соответствующие координаты *x* и *y*. Заметим, что интерполяция осуществляется в локальной системе координат. В этой же системе задаются координаты конечной точки, поэтому перед выводом необходимо преобразовать координаты из локальной системы в глобальную.

Программная реализация алгоритмов круговой интерполяции. При интерполяции дуг окружностей можно использовать два способа:

1) при помощи параметрических уравнений;

2) метод оценочной функции.

Первый способ основан на математическом представлении дуги с помощью тригонометрических функций:

$$\begin{cases} X = x_0 + R \cdot \cos(\omega t), \\ Y = y_0 + R \cdot \sin(\omega t). \end{cases}$$

Программная реализация такого метода с помощью МК потребует не только использования операции умножения, но и вычисления функций sin и cos, что достаточно проблематично.

Метод оценочной функции (рисунок 3.22) лишен этого недостатка.

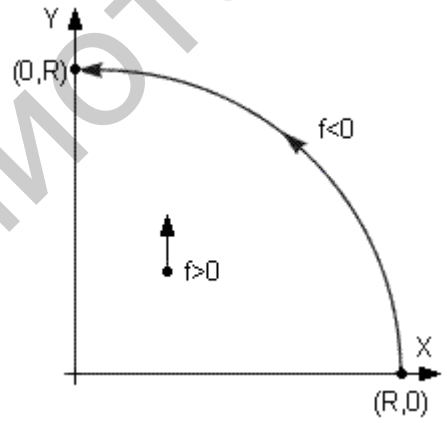

Рисунок 3.22– Представление дуги методом оценочной функции

Этот метод использует оценочную функцию вида $f(x, y) = R^2 - (x^2 + y^2)$, где *R* – радиус дуги; *x*, *y* – расстояние от центра до текущей точки дуги вдоль осей *X* и *Y*. На линии окружности $f = 0$, внутри окружности $f < 0$, вне окружности – $f > 0$. Алгоритм интерполяции аналогичен алгоритму линейной интерполяции, изложенному выше, *X* и *Y* изменяются дискретно, и на каждом шаге вычисляется оценочная функция *f*; если $f > 0$ осуществляется шаг по *X*, если $f < 0$ – шаг по *Y*. При этом:

– если $f \geq 0$, то $y_{i+1} = y_i + 1$,

$f_{i+1} = R^2 - x_i^2 - (y_i + 1)^2 = f_i - 2y_i - 1$;

– если $f < 0$, то $x_{i+1} = x_i - 1$

$f_{i+1} = f_i + 2x_i - 1$.

В общем случае необходимо осуществлять интерполяцию в локальной системе координат, которая расположена в центре дуги окружности (не в начальной точке и/или в другом квадранте). Задача интерполяции для каждой дуги решается в локальной системе координат, а перед выводом необходимо перейти к глобальной системе.

Реализация алгоритмов пропорционального управления. В реальных технических системах отработка объектом выдаваемых на него управляющих воздействий выполняется принципиально неидеально вследствие действия ряда возмущающих факторов. В электромеханических системах к таким факторам относятся непостоянство нагрузки (момента сопротивления) во времени, искажения управляющих сигналов под воздействием электромагнитных помех и др. В связи с этим обеспечение качественного выполнения системой заданных эволюции возможно лишь при наблюдении за состоянием объекта и определении степени отклонения значений наблюдаемых параметров от заданных значений (уставок) с целью коррекции подаваемых на объект управляющих воздействий для обеспечения требуемого качества отработки заданной траектории. Решение задач такого рода возлагается на регуляторы.

Типовой вариант организации программных компонентов встраиваемой системы управления с применением программных регуляторов показан на рисунке 3.23.

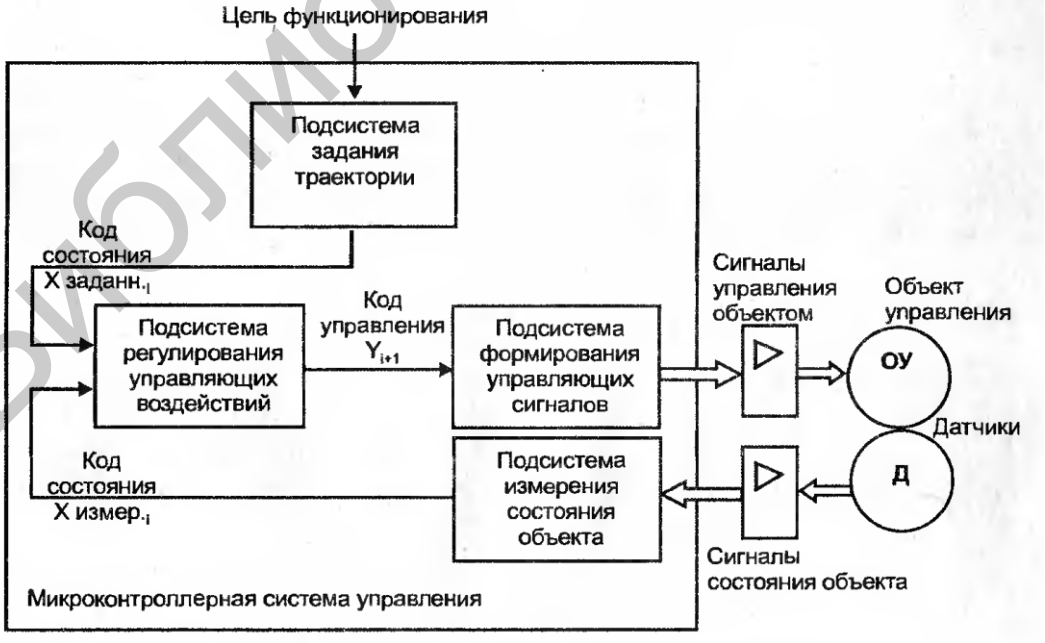

Рисунок 3.23 – Использование программных регуляторов в МК-системе

В данной схеме подсистема задания траектории в соответствии со сформированной извне (на более высоком иерархическом уровне управления) целью функционирования системы в целом через каждый очередной квант времени генерирует требуемый вектор состояния объекта $X_{задан}$ (в виде соответствующей кодовой комбинации). Подсистема измерения выделяет из физических сигналов, поступающих с датчиков, информацию о векторе состояния объекта $X_{измер}$ (также в виде кодовой комбинации). Задачей регулятора является вычисление такого управления Y (в виде кодового слова, преобразуемого далее формирователем в физические управляющие сигналы U), которое бы переводило объект в новое состояние X с наперед заданной степенью близости к $X_{задан}$. Следует отметить, что это управляющее воздействие объект также отработает неидеально. Таким образом, функционирование такой системы управления заключается в постоянном слежении за состоянием объекта в целях минимизации отклонения от заданной траектории.

Итак, регулятор вычисляет некоторую функцию $Y(X_{задан}, X_{измер}, t,\ldots)$, где помимо информации, описывающей текущую ситуацию (это аргументы $X_{задан}$ и $X_{измер}$), в качестве дополнительных параметров могут присутствовать аргументы, описывающие поведение системы на некотором отрезке времени – это либо значение системного времени t, либо производные и/или интегралы мгновенных значений $X_{задан}$, $X_{измер}$ и соотношений между ними.

Простейшим способом управления с непрерывным изменением управляющего воздействия является пропорциональное управление, которое реализуется на базе П, ПИ, ПИД регуляторов:

а) П-регулятор; $U = k \cdot \varepsilon$;

б) ПИ-регулятор: $U = k_1\varepsilon + k_2\int_0^t \varepsilon(t)dt$;

в) ПИД-регулятор: $U = k_1\varepsilon + k_2\int_0^t \varepsilon(t)dt + k_3\dot{\varepsilon}$,

где ε – разница температур.

Из теории автоматического управления известно, что П-регулятор является простейшим, однако он имеет ошибку в установившемся режиме. Астатическую ошибку можно устранить с помощью ПИ-регулятора, но он обеспечивает слабую устойчивость. Поэтому для повышения устойчивости и качества переходных процессов используют пропорционально-интегрально-дифференциальное регулирование: ПИД-регулятор, который дополнительно имеет дифференциальную составляющую. В микропроцессорных системах ПИД-регуляторы реализуются при помощи рекуррентных соотношений. При этом принимают

$$\dot{\varepsilon}_k \approx \frac{\varepsilon_k - \varepsilon_{k-1}}{\Delta T}, \quad \int_0^t \varepsilon(t)dt \approx \sum_{i=1}^{k} \varepsilon_i \cdot \Delta t = \sum_{i=1}^{k-1} \varepsilon_i \Delta t + \varepsilon_k \Delta t .$$

Тогда алгоритмы, реализующие регуляторы, можно записать в виде

а) П-регулятор; $U_k = k \cdot \varepsilon$;

б) ПИ-регулятор: $U_k = A_1\varepsilon_k + A_2\varepsilon_{k-1}$;

в) ПИД-регулятор: $U_k = U_{k-1} + A_1\varepsilon_k + A_2\varepsilon_{k-1}$.

При программной реализации этого выражения возникают такие проблемы, как переполнение разрядной сетки при умножении; получение нулевого результата при вычитании близких чисел, особенно при вычислении скорости; интегральная составляющая зависит от периода квантования, откуда следует, что коэффициенты A_1, A_2 должны подстраиваться в соответствии с параметром Δt. Также следует учитывать, что программа зациклена, это приводит к необходимости ее вызова по таймеру.

Опрос датчиков аналоговых величин. При построении замкнутых систем управления электромеханическими объектами, позволяющих отработать уставки оператора путем определения величины управляющего воздействия как функции от заданного и текущего состояния объекта, необходимо измерение этого состояния путем использования датчиков обратной связи: по частоте вращения, по моменту сопротивления, по положению и др. Рассмотрим подсистему измерения значений первых двух из указанных параметров.

Аппаратная реализация подсистемы ввода показана на рисунке 3.24.

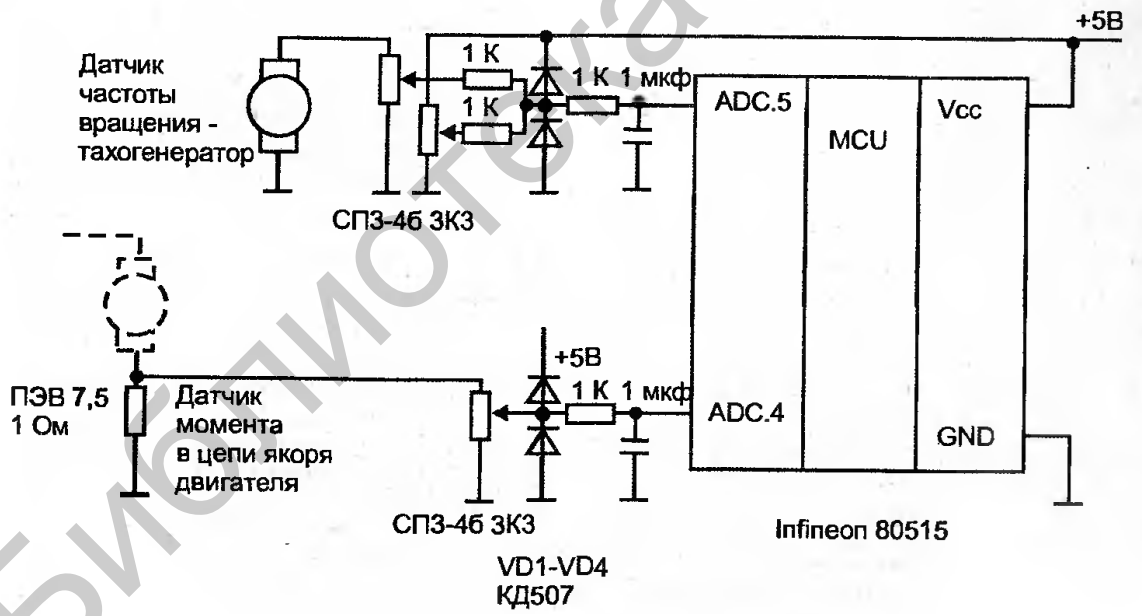

Рисунок 3.24 – Схема устройства сопряжения с датчиками

В качестве датчика частоты вращения использован тахогенератор (здесь будем говорить о некотором абстрактном тахогенераторе; в каждом конкретном случае целесообразно выбирать наиболее полно соответствующий параметрам электромеханической системы, в частности, по значению максимальной частоты вращения). В простейших случаях в качестве тахогенератора может быть использован двигатель постоянного тока, включенный в режиме ге-

нератора. В качестве датчика момента используется образцовый резистор, падение напряжения на котором пропорционально току якорной обмотки двигателя (исполнительного или нагрузочного); сила этого тока в свою очередь пропорциональна величине момента на валу ротора. В связи с тем, что значения уровней выходных напряжений этих датчиков достаточно велики, необходимое масштабирование сигналов, как правило, сводится к пропорциональному снижению их амплитуды резистивными делителями до диапазона 0...5 В, применяемого в АЦП.

Требуемый коэффициент деления определяется при настройке подбором положения ползунка реостата (в случае необходимости сужения диапазона измерений можно применить соответствующие настройки АЦП). Так как ротор исполнительного двигателя может вращаться в обоих направлениях, выходное напряжение тахогенератора в штатном режиме может быть отрицательным, что недопустимо для входов АЦП МК. В схеме обработки сигналов канала АЦП №5 применен пассивный сумматор, «поднимающий» масштабированное напряжение тахогенератора на величину +2,5 В (половина полного диапазона измеряемых напряжений), точно задаваемую с помощью потенциометра. Напряжение тахогенератора должно быть смасштабировано так, чтобы в рабочем диапазоне частот $-W_{max}...+W_{max}$ оно находилось в пределе –2,5…2,5 В. При этом $U_{вх}$ АЦП будет находиться в диапазоне 0...5 В. Для защиты микроконтроллера от повреждений при возможных аварийных режимах (резкие увеличения значений частоты и/или тока, приводящие к превышению выхода напряжения $U_{вх}$ АЦП за границы диапазона 0...5 В) используются диоды, открывающиеся при снижении входного напряжения ниже величины –0,4 В (нижний диод пары) и при превышении входным напряжением величины +5,4 В (верхний диод пары). Возможные помехи, наводимые электромагнитными полями электрических машин, отфильтровываются *RC*-цепями.

Схема алгоритма управляющей программы приведена на рисунке 3.25, а листинг программы – на рисунке 3.26. Для определенности примера положим, что $U_{вх}5$ находится в диапазоне 0...5 В, а $U_{вх}4$ – в диапазоне 0...1 В (т. е. $I_я$ – ток якоря двигателя, который лежит в диапазоне 0...1 А).

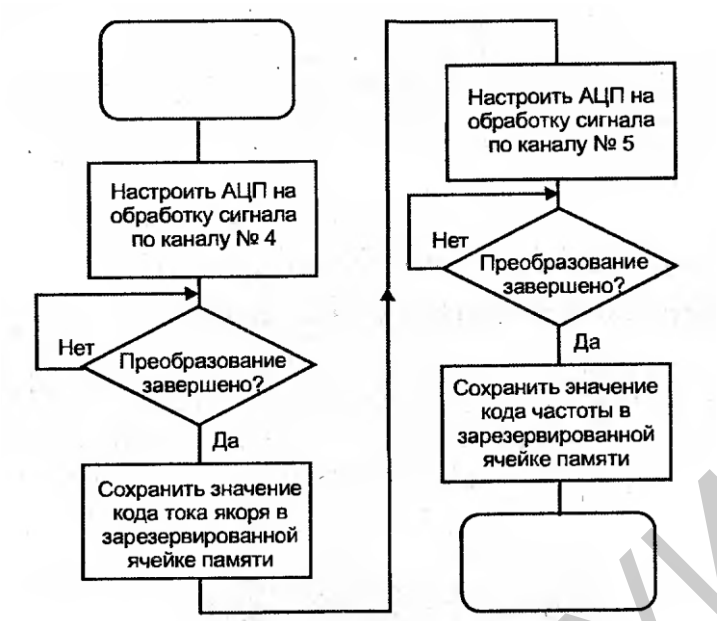

Рисунок 3.25 – Схема алгоритма опроса датчиков частоты и момента

```
; подпрограмма wm_read – опрос датчиков частоты вращения и момента
                ; Символические определения
W:              equ 36h                 ;код частоты вращения
I:              equ 37h                 ;код тока якоря

wm_read:        push 0                  ;сохранение значения регистра
                mov r0,#4
                call $adc_switch        ;активизация канала 4 (ток якоря)
                mov 0DAh, #040h         ;диапазон измерения 0-1,25 В – это
                                        ;ближайший больший для диапазона
                                        ; 0-1 В
                jb 0d8.h4,$             ;ожидание готовности АЦП
                mov I, 0D9h             ;сохранение кода тока якоря
                inc r0
                call $adc_switch        ;активизация канала 5 (частота вращен.)
                mov 0DAh, #0h           ;полный диапазон измерений
                jb 0d8.h4,$             ;ожидание готовности АЦП
                mov W, 0D9h             ;сохранение кода частоты вращения
                pop 0                   ;восстановление значения регистра
                ret
$adc_switch:    mov a,0d8h              ;процедура настройки канала АЦП
                anl a,#1111100b
                orl a,#00001000b
                orl a,r0                ;номер канала хранится в R0
                mov 0d8h, a
wm_read_end:    ret
```

Рисунок 3.26 – Листинг программы опроса датчиков

Управление двигателями постоянного тока. Одним из наиболее применяемых в автоматике типов электродвигателей является двигатель постоянного тока (ДПТ), характеризующийся простотой управления, удобством аналитической модели, линейностью характеристик в широком диапазоне работы. Рассмотрим реализацию управления ДПТ, обеспечивающую управление частотой вращения двигателя с возможностью смены направления вращения (реверсом).

Достаточно высокая (по сравнению с быстродействием управляющего микроконтроллера) инерционность электродвигателей, в основном определяемая механическими и конструктивными особенностями и составляющая, как правило, десятки (и более) миллисекунд, позволяет использовать для управления не аналоговый сигнал, от амплитуды которого линейно зависит желаемая частота вращения *w*, а широтно-импульсно-модулированный сигнал, в котором для изменения *w* соответствующим образом изменяется соотношение T импульса/T паузы. В сущности, изменение этого соотношения (соответственно и мощности, передаваемой на двигатель за период действия ШИМ-сигнала) эквивалентно изменению уровня аналогового напряжения управления (рисунок 3.27). Сглаживание импульсов ШИМ происходит за счет инерционности объекта управления.

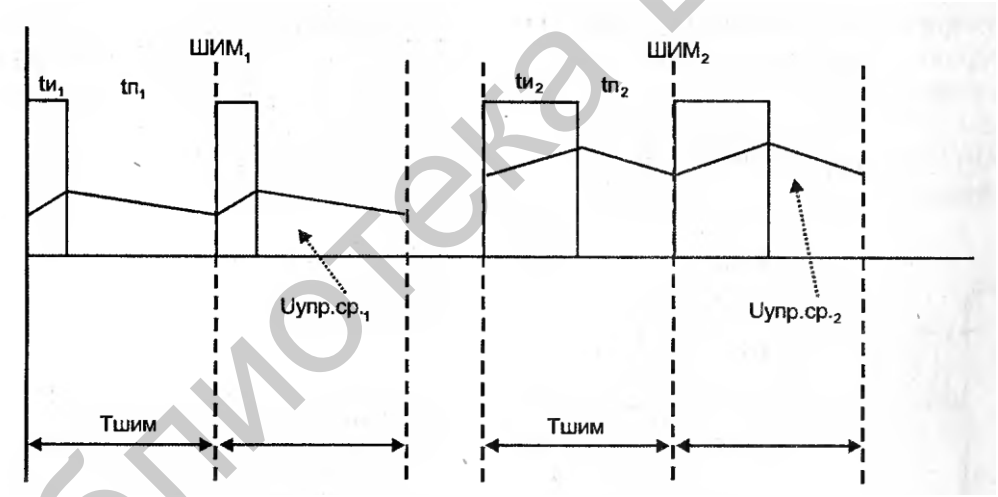

Рисунок 3.27 – Сигналы ШИМ-управления

Схемотехническая реализация устройства сопряжения микроконтроллера с ДПТ показана на рисунке 3.28.

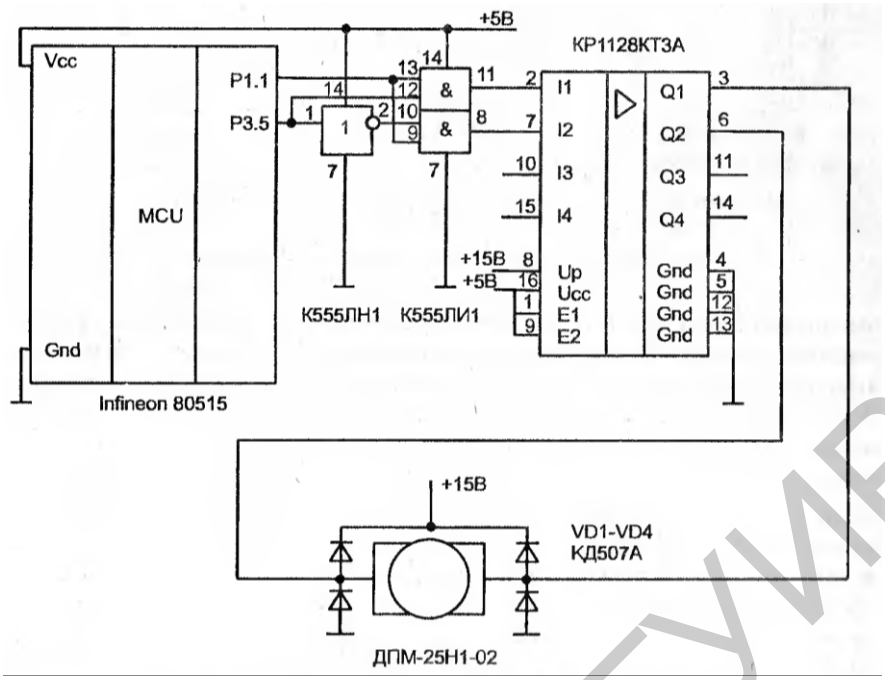

Рисунок 3.28 – Схемотехническая реализация устройства сопряжения с ДПТ

ШИМ-сигнал необходимой скважности снимается с линии P1.1, представляющего собой выход канала блока быстрого ввода-вывода, обеспечивающего при соответствующей программной настройке формирование ШИМ. Знак, определяющий направление протекания тока через обмотку ДПТ, задается линией P3.5 микроконтроллера. На микросхемах «НЕ» и «2И» реализован узел, обеспечивающий наложение знака на ШИМ-сигнал, а микросхема КР1128КТ3А предназначена для усиления мощности управляющих сигналов.

Временные диаграммы работы узла показаны на рисунке 3.29.

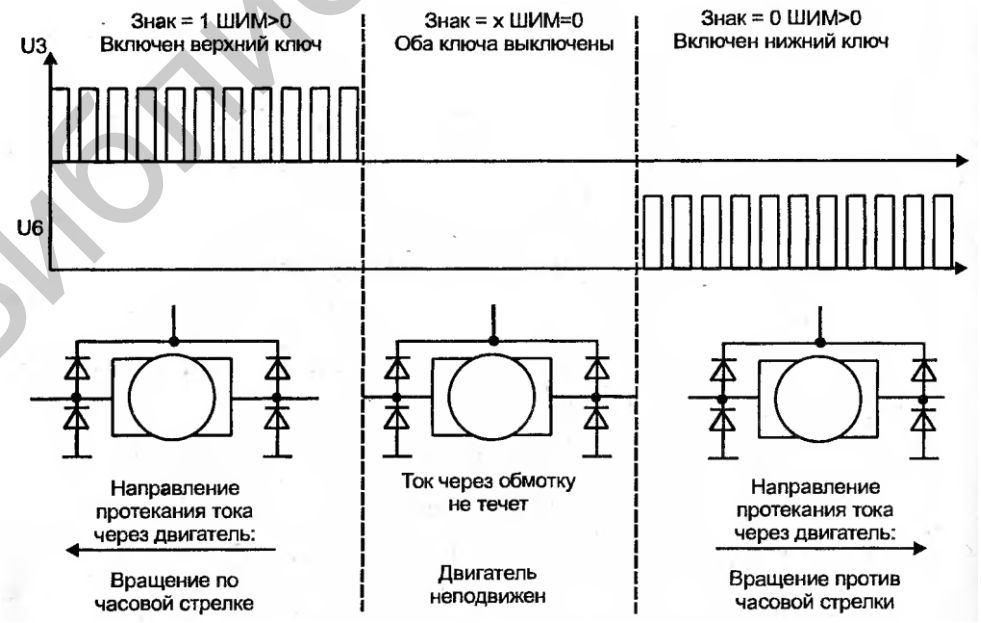

Рисунок 3.29 – Временные диаграммы работы устройства сопряжения с ДПТ

Микросхема КР1128КТ3А коммутирует большие токи (до 1 А постоянного тока на фазу) и требует при подключении индуктивной нагрузки применения внешних диодов, что и выполнено в примере.

Отметим, что возможен вариант аппаратной реализации, в котором для управления используются не один, а два канала ШИМ, при этом операции наложения знака на ШИМ-сигнал выполняются программно: на одном канале осуществляется генерация ШИМ, на другом такая генерация блокируется (формируется логический ноль). Такая организация позволяет исключить из схемы микросхемы «НЕ» и «2И», однако расходует важный ресурс МК– каналы ШИМ. Степень оптимальности вариантов решений должна определяться в каждом случае применения индивидуально.

Схема алгоритма модуля управляющей программы приведена на рисунке 3.30, а листинг программы – на рисунке 3.31.

Рисунок 3.30 – Схема алгоритма модуля управляющей программы

```
; подпрограмма dpt_drv  – управление ДПТ
k_upr:      equ 30h          ;величина кода управления:
                             ;7-разряд – знак, разряды 6-0 – модуль
main:       lcall init       ;настройка переменных и устройств

cycle:      ……               ;действия, предусмотренные в
                             ;главном цикле фоновой программы,
                             ;в том числе – выдача управления на
            lcall pwm        ; ДПТ
            sjmp cycle
```

Рисунок 3.31 – Листинг подпрограммы управления ДПТ, лист 1

```
init:   mov c8h, #00010001b   ;задать режим Т2: неуправл. счет
                              ;с автоперезагрузкой
        mov cbh, #FFh         ; константа перезагрузки: старший байт
        mov cah,#80h          ;младший байт (частота ШИМ  7,8 кГц)
        orl p1,#00000010b     ;вывод через канал 1 (разряд Р1.1)
        mov c1h, #00001000b   ;тип сигналов – сигналы ШИМ
        mov c3h, cbh          ;установка минимального кода ШИМ
        mov c2h, cah
        mov k_upr, #0
        ret

pwm:    mov a,k_upr           ;прием кода управления
        mov c, acc.7          ;выделение знака
        mov p3.5, c           ;и выдача его на соответ. линию
        cpl a                 ;больше код – шире импульс
        orl a, #80h           ;учет диапазона изменения кода
        mov c2h, a            ;выдача в мл. байт регистра канала
        ret
```

Рисунок 3.31 – Лист 2

Положим, что переменная управления, формируемая извне (например, подпрограммой-регулятором), определена в следующем формате: младшие 7 бит переменной управления кодируют интенсивность ШИМ-сигнала, а старший – его знак.

Поскольку ограничение сверху на частоту ШИМ-сигнала не накладывается (имеется лишь ограничение снизу: важно, чтобы она была на порядок выше собственный частоты объекта управления), выберем частоту максимальной из удобных. Для этого проинициализируем регистр перезагрузки таймера кодом *FF*80h. Таким образом, для корректной отработки ШИМ-сигналов регистр данных канала должен содержать значения в диапазоне *FF*80h – *FFFF*h, т. е. управление скважностью производится значениями младшими семью битами младшего байта регистра данных (и только ими). Удобство реализации очевидно – отделив из кода управления знаковый разряд и передав его на разряд Р3.5, оставшиеся 7 бит (кодируют диапазон 00 – 7*F*h) необходимо добавить к коду 80h (минимальное значение младшего байта регистра канала) и занести результирующее значение в младший байт регистра данных канала.

Управление асинхронными трехфазными двигателями. Асинхронные трехфазные двигатели (АТД) обладают высоким КПД и низкой стоимостью, а также высокой надежностью, что позволяет рекомендовать их в ряде задач автоматики в качестве исполнительных двигателей.

При управлении АТД, как правило, применяется амплитудно-частотное управление, характеризующееся пропорциональной зависимостью амплитуды питающего напряжения от частоты его изменения.

Для АТД необходимо сформировать вращающееся электромагнитное поле, управляя набором ключей *К*1 – *К*6, преобразующих постоянное напряжение $U_{пит}$ в трехфазное напряжение переменного тока (рисунок 3.32). Здесь показаны уровни напряжений между началами обмоток.

Изменение частоты вращения ротора АТД достигается изменением частоты коммутации ключей *К*1 – *К*6 с одновременным изменением $U_{пит}$ так, чтобы соотношение между амплитудой и частотой напряжения оставалось постоянным. Изменение амплитуды $U_{пит}$ обеспечивается с помощью ШИМ (рисунок 3.33).

Схема алгоритмов программ управления приведены на рисунке 3.34, а листинг программы – на рисунке 3.35 соответственно.

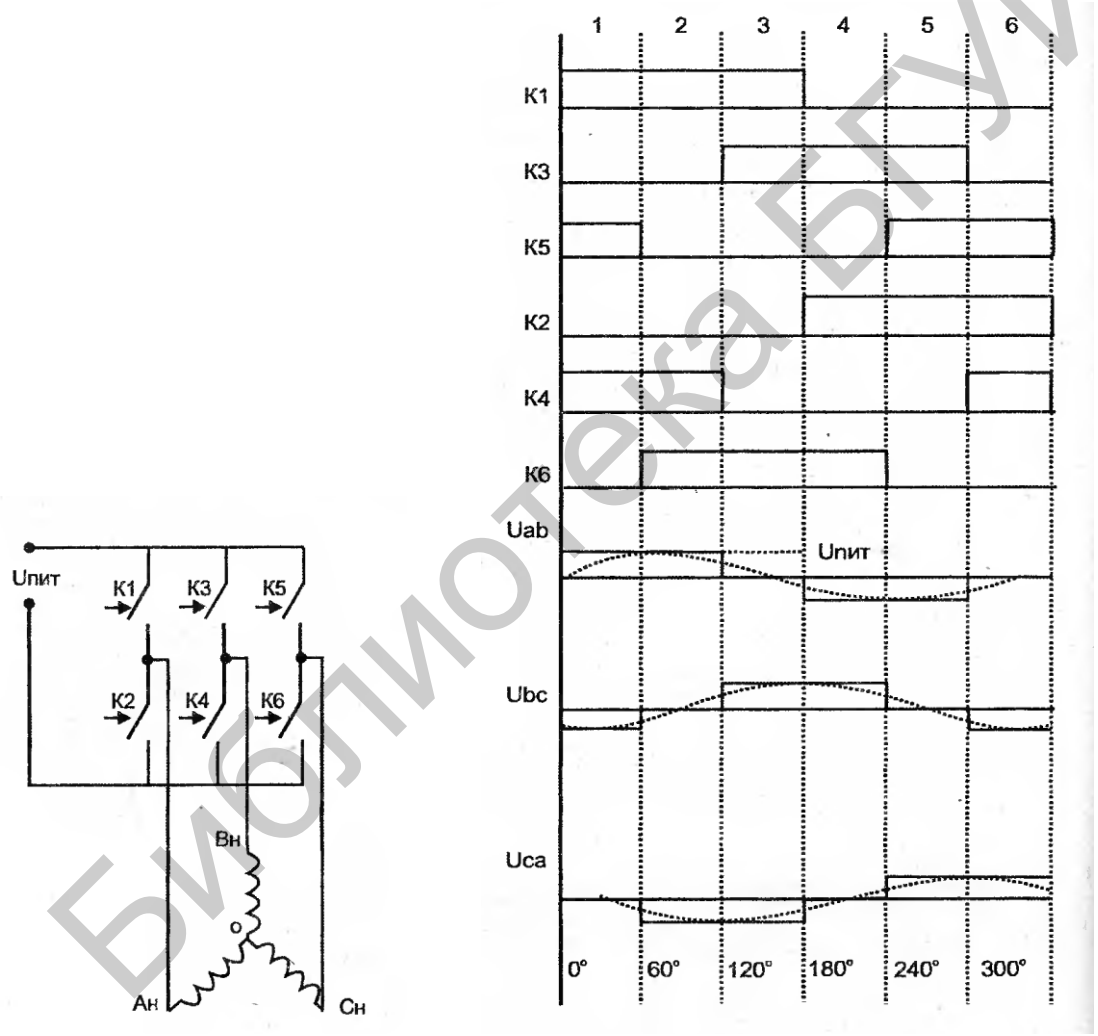

Рисунок 3.32 – Структура и временные диаграммы работы блока ключей

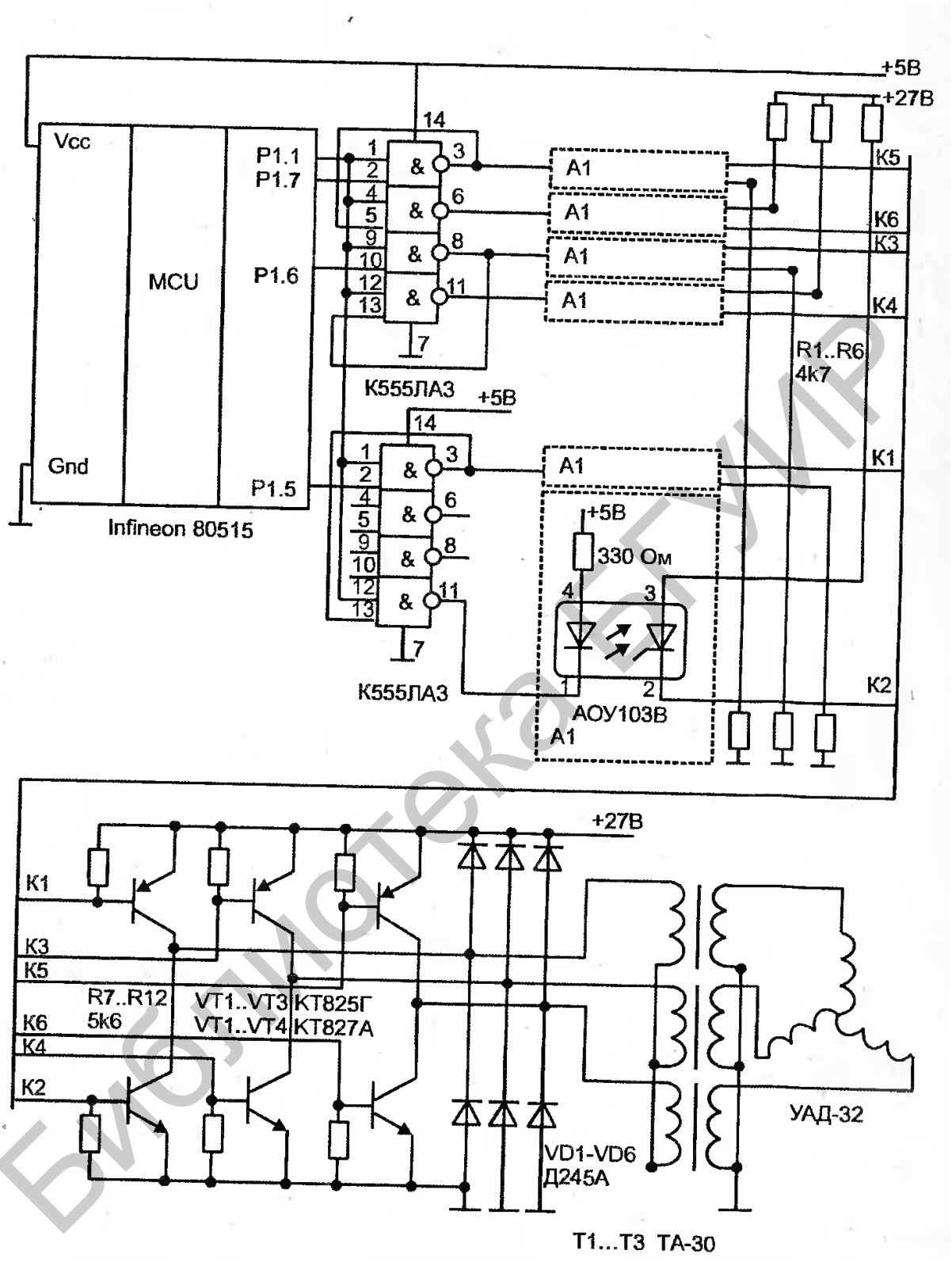

Рисунок 3.33 – Схемотехническая реализация устройства сопряжения с АТД

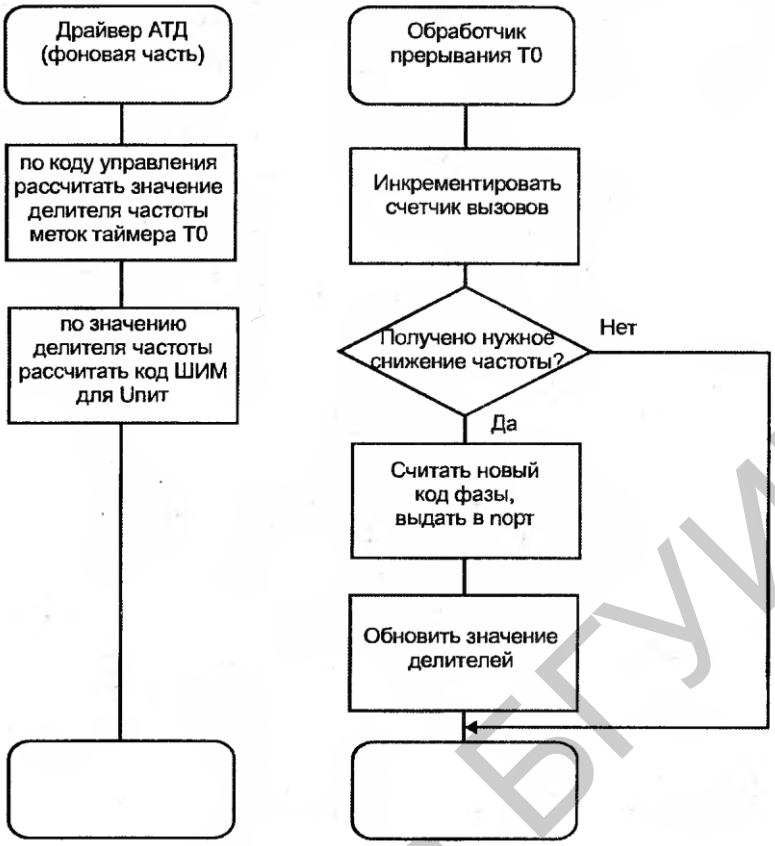

Рисунок 3.34 – Схемы алгоритмов программ управления

```
;подпрограмма atd_drv – управление АТД
atd_cntr:     equ 3Ah              ;код управления АТД
oldatd:       equ 39h              ;значение ШИМ предыдущего цикла
n_cycle:      equ 42h              ;счетчик фаз
count_sec:    equ 43h              ;делители частоты
newcntsec:    equ 44h
num_sec:      equ 45h
main:         lcall init_atd       ;настройка переменных и устройств,
                                   ;подсистемы управления АТД
cycle:        ....
              lcall main_count_sec ;расчет амплитуды Uпит
              lcall atdcontr       ;формирование Uпит посредством ШИМ
              sjmp cycle
init_atd:     mov a,tmod           ;настройка таймера в 16-битный режим
              anl a,#11110000b
              orl a,#00000001b
              mov tmod,a
              mov th0,#FCh         ;частота перезагрузок таймера 1 кГц
              mov tl0,#18h
              mov newcntsec, #18
              mov n_cycle,#0
              mov oldatd,#255
```

Рисунок 3.35 – Листинг подпрограммы управления АТД, лист 1

```
                mov num_sec, #0
                setb P1.7               ;начальное положение фаз
                clr P1.6
                setb P1.5
                setb tr0                ;пуск таймера 0
                mov 0c8h,#00010001b ;настройка T2 для обеспечения ШИМ
                mov 0c1h,#00001000b ;выделение 1 канала для ШИМ (P1.1)
                orl p1,#00000010b
                mov cah,#00h
                mov cbh,#0feh           ;f ШИМ 2 кГц
                mov c2h,#00h            ;начальное задание ШИМ
                mov c3h,#0feh
                setb et0                ;разрешение прерываний от T0
                setb ea
                ret
int_t0:         mov tl0,#18h            ;обработчик запросов прерывания от T0
                mov th0,#FCh
                push acc
                push psw
                push dph
                push dpl
                inc num_sec             ;АТД переключается на очередную фазу
                mov a,num_sec           ;через каждые newcntsec миллисекунд
                cjne a,newcntsec,not_eqv
                mov newcntsec,count_sec ;обновление значения делителя частой
                lcall key
                mov num_sec,#0
not_eqv:        pop dpl
                pop dph
                pop psw
                pop acc
                reti
;подпрограмма формирования импульсов для управления ключами
key:            inc n_cycle             ;переход на очередную фазу
                mov a,n_cycle           ;нумерация фаз от 0 до 5
                cjne a,#6,not_6
                mov n_cycle,#0
                mov a,n_cycle
not_6:          mov dptr, #table        ;считать код коммутации ключей К1-К6
                move a,@a + dptr
                anl p1,#0Fh
                orl p1,a                ;и выдать их в старшие разряды порта P1
                ret
table:          db 160, 32, 96, 64, 192, 128 ;таблица кодов коммутаций в
                                        ;разрядах 5-7 содержит триады:
                                        ;101 - 100 - 110 - 010 - 011 - 001
main_count_sec:                         ;по коду управления рассчитывается частота
                mov a,atd_cntr          ;при максимальном коде управления
                cpl a                   ;count_sec равен 4
```

Рисунок 3.35 – Лист 2

```
                mov b,#3h           ;при минимальном коде управления
                div ab              ;count_sec равен 18
                add a,#19h
                mov b,#6
                div ab
                mov count_sec,a
                ret

atdcontr:       mov a,newcntsec     ;по частоте управления рассчитывается код ШИМ
                mov b,#14
                mul ab              ;Кшим изменяется от 56 до 252
                cjne a, oldatd, ch_atd ;прямо пропорционально частоте вращения
                jmp exit_atd
ch_atd:         mov oldatd,a        ;в случае изменившегося управления -
                mov c2h,a           ;обновление младш. байта рег. меток канала 1
exit_atd:       ret
                ;размещение точек входа
                ;в обработчики соответствующих запросов прерываний
                org 000bh
                ljmp t0_ int        ;по отсчету временных интервалов
```

Рисунок 3.35 – Лист 3

Организация межмодульного программного взаимодействия. В реальных микропроцессорных системах управления организована совместная работа подсистем опроса пользовательского пульта, измерения показаний датчиков и вывода информации на пользовательский пульт с отображением информации на ЖКИ.

Пусть информация выводится на ЖКИ в формате, представленном на рисунке 3.36.

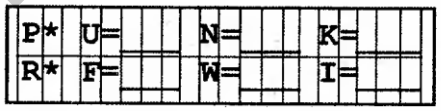

Рисунок 3.36 – Формат отображения информации

Введем следующие обозначения: *P** – код заданной пользователем программы работы (например *P*2), *R** – код заданного пользователем режима работы (например *R*l), *U* – код заданной пользователем уставки №1 (например *U* = 140), *N*, *K*, *F* – значения уставок №2, №3 и №4 соответственно, *W* – код измеренной частоты вращения, *I* – код измеренного тока якоря.

Задача совместной схемотехнической реализации данных подсистем не представляет сложности, т. к. является простой композицией рассмотренных ранее решений, поэтому не рассматривается.

Рассмотрим задачу объединения соответствующих подпрограмм. Так как считывать информацию с источников и выводить ее в указанном формате на

ЖКИ предполагается неограниченно долго, то программу, выполняющую данные действия, целесообразно реализовать в виде двух фрагментов: в подготовительной части целесообразно сформировать во внешней памяти данных МК заданный шаблон отображения и вывести его на ЖКИ, а в циклической – считывать коды соответствующих величин, преобразовывать их в *ASCII*-представление и записывать в соответствующие позиции относительно начала буфера, хранящего шаблон. Суть *ASCII*-преобразования информации сводится к разбиению байта данных на цифры и замене каждой цифры кодом соответствующего символа цифры. В соответствии с кодировкой *ASCII*, цифра 0 имеет код 30h, цифра 1–31h и т. д., т. е. для осуществления такого преобразования необходимо к значению кода цифры добавить код 30h.

Схема программного комплекса приведена на рис. 3.37, а его текст – на рисунке 3.38.

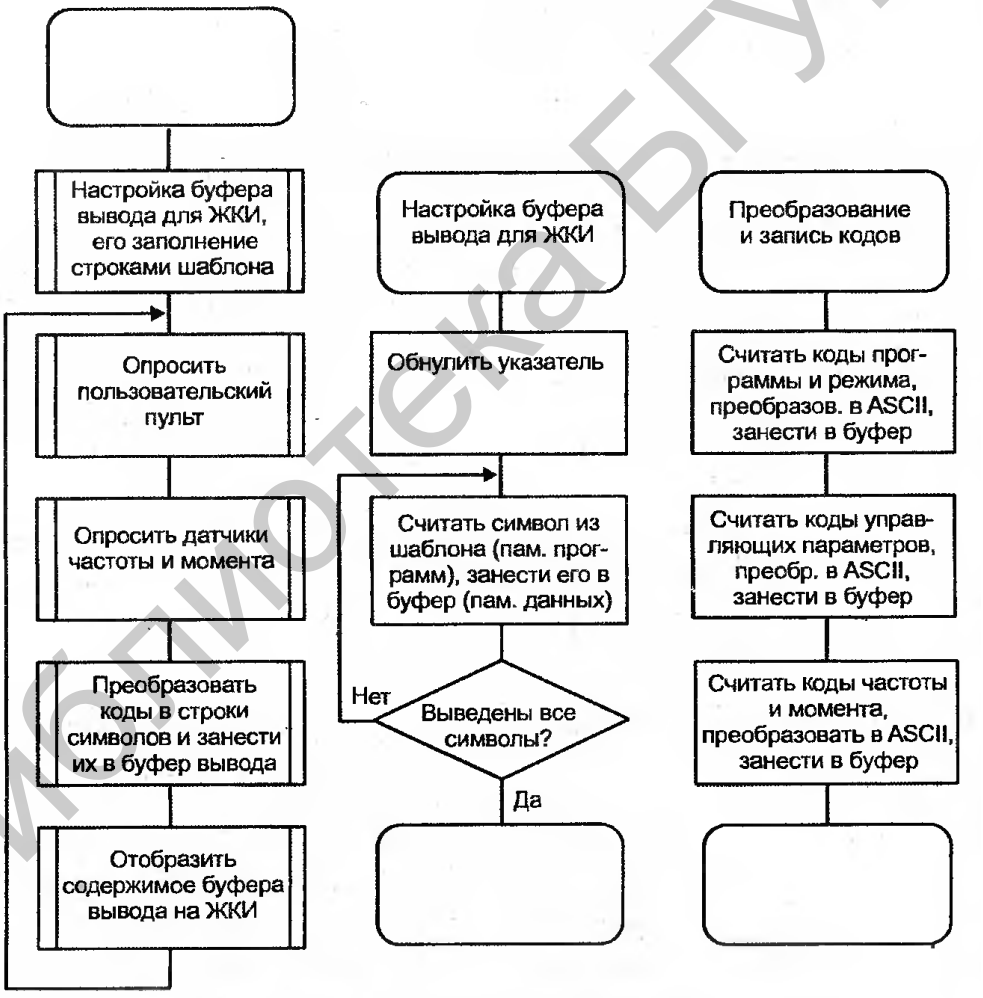

Рисунок 3.37 – Схема взаимодействия программного комплекса

```
;Демонстрация взаимодействия программных компонент
                org 0000h
                ljmp main
                org 0300h
main:           lcall init
main_cycle:     lcall scan
                lcall wm_read
                lcall prep_data
                lcall indic
                ljmp main_cycle
                                ;подключаемые процедуры библиотеки
                include asms\su\init.asm
                include asms\su\scan.asm
                include asms\su\wm_read.asm
                include asms\su\prep_dat.asm
                include asms\su\decimal.asm
                include asms\su\indic.asm
;подпрограмма настройки программного комплекса
init:           push 0
                mov r0,#0
$init_1:        mov dptr,#init_stringl          ;вычисление смещения
                mov a,dpl
                add a,r0
                mov dpl,a
                jnc $init_2
                inc dph
$init_2:        clr a                           ;и чтение очередного символа строки,
                movc a,@a+dptr                  ;расположенной в памяти программ
                mov dph,a
                mov a,#0D0h
                add a,r0
                mov dpl,a
                mov a,dph
                mov dph,#0FFh                   ;с занесением его в буфер для
                movx @dptr,a                    ;вывода на ЖКИ с тем же смещением
                inc r0
                cjne r0,#40,$init_1             ;объем буфера равен сорока символам
init_end:       pop 0
                ret
init_string1:   db 'P* U=___ N=___ K=___ ;строка-шаблон
init_string2:   db 'R* F=___ V=___ I=___

;подпрограмма преобразования кодов параметров в их
;строковое представление
prep_data:      mov dptr,#0FFD1h                ;адресация знакоместа цифры
                mov a,Kprog                     ;кода программы (0...5), чтение
                add a,#30h                      ;и преобразование этой цифры в символ
                movx @dptr,a                    ;занесение символа в буфер для вывода
                mov dptr,#0FEE5h                ;аналогичные преобразования
```

Рисунок 3.38 – Листинг программного комплекса, лист 1

```
                mov a,Kreg              ;для кода режима
                add a,#30h
                movx @dptr,a
                mov dptr,#0FFD5h        ;т. к. коды управления – байты, то для них
                mov a,Kupr1             ;выполняется процедура преобразования
                call decimal            ;в три символа и запись этих символов
                mov dptr,#0FFDBh
                mov a, Kupr2            ;аналогично для Kupr2
                call decimal
                mov dptr,#0FFE1h
                mov a, Kupr3            ;аналогично для Кирr3
                call decimal
                mov dptr,#0FFE9h
                mov a, Kupr4            ;аналогично для Kupr4
                call decimal
                mov dptr,#0FFEFh
                mov a,W                 ;аналогично для частоты вращения
                call decimal
                mov a,I                 ;аналогично для тока якоря
                mov dptr,#0FFF5h
                call decimal
                ret

;подпрограмма преобразования байта, хранящегося в аккумуляторе,
;в трехсимвольную строку, с ее записью во внешнюю память,
;начиная с адреса из DPTR
decimal:        mov b,#100
                div ab                  ;выделение цифры сотен
                add a,#30h              ;ее преобразование в символ цифры
                movx @dptr,a            ;и запись во внешнюю память данных
                inc dptr                ;перемещение указателя
                mov a,b
                mov b,#10
                div ab                  ;аналогично для цифры десятков
                add a,#30h
                movx @dptr, a
                inc dptr
                mov a,b                 ;аналогично для цифры единиц
                add a,#30h
                movx @dptr,a
                inc dptr                ;указатель – на следующую ячейку
                ret
```

Рисунок 3.38 – Лист 2

ЗАКЛЮЧЕНИЕ

Пособие охватывает наиболее важные вопросы проектирования микропроцессорных систем управления.

Основное внимание уделяется схемотехнической и программной реализации основных узлов систем управления электромеханическими объектами. Рассмотрены типовые формы регуляторов и методы их программной реализации. Эти методы до сих пор остаются основными, несмотря на множество других методов, имеющих преимущества для систем конкретного применения.

Надеемся, что весьма сжатые сведения в учебном пособии являются достаточными для того, чтобы без затруднений перейти к более глубокому изучению соответствующих тем в опубликованных источниках, например, перечисленных ниже в списке литературы.

Пособие может быть использовано для индивидуального изучения и самостоятельного выполнения индивидуальных заданий по контрольным работам. В то же время оно будет полезно и в ходе дипломного проектирования студентам специальности 1-53 01 07 «Информационные технологии и управление в технических системах».

ЛИТЕРАТУРА

1 Новиков, Ю. В. Основы микропроцессорной техники : курс лекций для вузов / Ю. В. Новиков, П. К. Скоробогатов. – М. : ИНТУИТ РУ, 2006.

2 Микропроцессорные системы: учеб. пособие для вузов / Е. К. Александров [и др.] ; под общ. ред. Д. В. Пузанкова. – СПб. : Политехника, 2002.

3 Васильев, А. Е. Микроконтроллеры. Разработка встраиваемых приложений : учеб. пособие / А. Е. Васильев. – СПб. : БХВ-Петербург, 2008.

4 Белов, А. В. Самоучитель разработчика устройств на микроконтроллерах AVR / А. В. Белов. – СПб. : Наука и техника, 2008.

5 Предко, М. Руководство по микроконтроллерам. В 2 т. / М. Предко. – М. : Постмаркет, 2001.

6 Бродин, В. Б. Микроконтроллеры. Архитектура, интерфейс / В. Б. Бродин, И. И. Шагурин. – М. : ЭКОМ, 1999.

7 Сташин, В. В. Проектирование цифровых устройств на однокристальных микроконтроллерах / В. В. Сташин, А. В. Урусов, О. Ф. Мологонцева. – М. : Энергоатомиздат, 1990.

8 Петров, И. В. Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования / И. В. Петров. – М.: СОЛОН-Пресс, 2004.

9 Тавернье, К. PIC-микроконтроллеры. Практика применения / К. Тавернье ; пер. с фр. – М. : ДМК-Пресс, 2002.

10 Современные микроконтроллеры : архитектура, средства проектирования, примеры применения / под ред. И. В. Коршуна. – М. : Аким, 1998.

11 Цифровые и аналоговые интегральные микросхемы : справочник / под ред. С. В. Якубовского. – М. : Радио и связь, 1989.

12 Ульрих, В. А. Микроконтроллеры PIC16X7XX : семейство МК с АЦП / В. А. Ульрих. – М. : СОЛОН, 2005.

Св. план 2013, поз. 19

Учебное издание

Чумаков Олег Анатольевич
Городко Сергей Иванович

МИКРОПРОЦЕССОРЫ В СИСТЕМАХ УПРАВЛЕНИЯ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *В. М. Задоля*

Подписано в печать 20.10.2016. Формат 60×84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 4,3. Уч.-изд. л. 4,0. Тираж 100 экз. Заказ 9.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014,
ЛП №02330/264 от 14.04.2014.
220013, Минск, П. Бровки, 6