ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ

ПРОЕКТИРОВАНИЕ ВСТРОЕННЫХ СИСТЕМ НА МИКРОКОНТРОЛЛЕРАХ STMicroelectronics

Учебное пособие

Под редакцией В.С. Харченко, А.А. Орехова

ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ

ПРОЕКТИРОВАНИЕ ВСТРОЕННЫХ СИСТЕМ НА МИКРОКОНТРОЛЛЕРАХ STMicroelectronics

Учебное пособие

Под редакцией В.С. Харченко, А.А. Орехова

Харьков
Национальный аэрокосмический университет
им. Н.Е. Жуковского «ХАИ»
2008

УДК 627.03

Авторы: Бабешко Е.В., Желтухин А.В., Куланов В.А., Мазуренко А.В., Мпандо П., Орехов А.А., Харченко В.С., Яновский М.Э. **Проектирование встроенных систем на микроконтроллерах STMicroelectronics** / Под ред. Харченко В.С., Орехова А.А. - Министерство образования и науки Украины, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», 2007. – 197 с.
ISBN 978-966-2982-32-9

Изложены методические и инструментальные аспекты проектирования систем на 8-ми разрядных микроконтроллерах STMicroelectronics (STM). Проанализированы семейства микроконтроллеров STM. Рассмотрена архитектура микроконтроллера ST7, система команд, работа с периферийными устройствами и области его применения. Изложены вопросы разработки программного обеспечения и работы в инструментальной среде. Приводятся примеры проектирования реальных цифровых систем контроля и управления с использованием микроконтроллера ST7.

Предназначается для студентов, обучающихся по направлениям «Компьютерная инженерия», «Радиоэлектронные аппараты», «Авиация и космонавтика» и др., и является практическим руководством по разработке цифровых систем на микроконтроллерах ST7. Она может быть полезна аспирантам и преподавателям университетов, специалистам в области проектирования специализированных компьютерных систем и встроенных приложений.

Библ. – 40 наименования, рисунков – 96, таблиц – 27.

Рецензенты:

- д.т.н., профессор Хаханов Владимир Иванович, кафедра автоматизации проектирования вычислительной техники Харьковского Национального университета радиоэлектроники;
- д.т.н., профессор Фурман Илья Александрович, кафедра автоматизации и компьютерных технологий Харьковского Национального технического университета сельского хозяйства.

Книга рекомендована к изданию:
- ученым советом Национального аэрокосмического университета имени Н.Е.Жуковского «Харьковский авиационный институт» (протокол № 6 от 27 февраля 2008 года).

Содержание

СПИСОК СОКРАЩЕНИЙ

АЦП – аналого-цифровой преобразователь
БИС – большая интегральная схема
БП – блок питания
ДТ – датчик температуры
ИМС – интегральная микросхема
КК – ключевые каскады
МК – микроконтроллер
ОР – оптоэлектронное реле
ТЗ – техническое задание
ТЭН – электронагревательный элемент

ПРЕДИСЛОВИЕ

В декабре 2006 года после проведения конкурсных презентаций и предварительного анализа возможностей и перспектив развития, между университетами Украины, с одной стороны, и известной корпорацией STMicroelectronics, с другой, – были подготовлены договора о сотрудничестве. В группу таких университетов были отобраны Национальный технический университет Украины “Киевский политехнический институт”, Национальный аэрокосмический университет “Харьковский авиационный институт”, Национальный технический университет “Харьковский политехнический институт”, Харьковский национальный университет радиоэлектроники, Национальный технический университет “Львовская политехника”, Одесский национальный технический университет, а позже к ним присоединился Донецкий и Днепропетровский национальные технические университеты.

В июне 2007 года в посольстве Франции в Украине были подписаны двухсторонние договора и осуществлена передача оборудования STMicroelectronics указанным университетам. На их базе созданы учебные центры, которые сегодня оборудованы классами по изучению технологий корпорации STMicroelectronics, прежде всего, микроконтроллеров семейства ST7. В Национальном аэрокосмическом университете “ХАИ” такой центр был создан на кафедре компьютерных систем и сетей.

Данное сотрудничество имеет долговременные цели, включающие образовательную, инжиниринговую и исследовательскую компоненты. Их достижение позволит украинским университетам:

- во-первых, получить доступ к изучению и овладению передовыми технологиями в области микроэлектроники и компьютерной техники;
- во-вторых, проявить свои возможности как учебных центров, осуществляющих подготовку квалифицированных специалистов с высшим образованием и последипломную образовательную деятельность в этой динамичной области;
- в-третьих, что, на наш взгляд, особенно важно с точки зрения перспектив развития университетов, реализоваться как конструкторским и научным организациям, выполняющим инженерные проекты и прикладные исследования в содружестве с одним из мировых лидеров в разработке, производстве и продаже цифровых компонент.

Корпорация STMicroelectronics, как следует из анализа деятельности и общения с ее представителями, делает ставку в развитии и распространении своих технологий на университетские учебно-дизайнерские центры. Первые из них в Европе были созданы во Франции и Италии, следующие – развиваются в Украине.

Первый тренинг преподавателей университетов в рамках проекта был проведен инструкторами STMicroelectronics на базе Национального технического университета Украины “КПИ” в январе 2007 года и по-

священся изучению микроконтроллера ST7. Представители компании были поражены высоким уровнем украинских специалистов, за 2-3 дня успешно выполнившим программу, которая в других странах занимает от 5 до 10 дней. Это стало оптимистическим стартом данной программы в Украине. Аналогичный тренинг для новых участников проекта был организован и успешно проведен в октябре 2007 года на базе учебного центра в Национальном аэрокосмическом университете "ХАИ" специалистами кафедры компьютерных систем и сетей. Следующий тренинг по изучению 32-битных микроконтроллеров планируется провести в первой половине 2008 года.

В рамках образовательной составляющей проекта участниками проекта разработан сайт, который представляет динамично обновляемую информацию о технологиях STM и их внедрении. Ведется подготовка собственных учебных и методических материалов по микроконтроллерам ST7 для студентов и инженеров. Были переработаны учебные планы и программы дисциплин для подготовки бакалавров по направлению «Компьютерная инженерия», а также специалистов и магистров по специальности «Специализированные компьютерные системы». Сегодня в ХАИ оборудован специализированный класс, где осуществляется обучение студентов соответствующих специальностей и последипломного образования в различных формах. На кафедре разрабатываются дипломные проекты, в которых используются микроконтроллеры ST7.

Популяризации марки и продукции корпорации STMicroelectronics послужило участие ее менеджера в Украине и странах СНГ Поля Мпандо в Международной научно-технической конференции "Гарантоспособные компьютерные системы, сервисы и технологии" [11, 22]. Эта конференция была организована Национальным аэрокосмическим университетом «ХАИ» и другими организациями и состоялась в апреле 2007 года в г. Кировограде на базе Научно-производственного предприятия «Радий», одного из основных разработчиков, изготовителей и поставщиков электронного оборудования автоматизированных систем управления технологическими процессами управления и защит атомных электростанций.

Кроме того, были развернуты работы по выполнению пилотных технических проектов для получения в ближайшем будущем статуса дизайнерского центра. Презентация одного из проектов прошла во время открытия учебного центра и проведения тренинга в Национальном аэрокосмическом университете «ХАИ». Система управления и аварийного отключения автомобильного подъемника реализована на микроконтроллере ST7 и внедрена на одной из СТО в г. Харькове. Сегодня к сотрудничеству и реализации совместных проектов подключились такие промышленные предприятия и конструкторские организации, как научно-производственное объединение «СВЕТ ШАХТЕРА», ГНПП «Объединение Коммунар», НТ СКБ «ПОЛИСВИТ» и др.

На наш взгляд, форма сотрудничества, предложенная корпорацией STMicroelectronics, может стать весьма эффективной для университетского образования и науки и послужить примером взаимодействия крупного бизнеса и ВУЗов.

С текущей информацией по проекту можно ознакомиться в on-line конференции http://st.4ua.info.

Указанные обстоятельства явились побудительным мотивом для издания книги, посвященной проектированию систем на микроконтроллерах STMicroelectronics. Она имеет целью публикацию аналитических, справочных и методических материалов, а также практических разработок в области проектирования систем на микроконтроллерах STMicroelectronics, выполненных сотрудниками центра «ХАИ – STM».

В данной книге приводится обзор семейства микроконтроллеров ST7, излагаются методические и инструментальные аспекты программирования микроконтроллеров на ассемблере и языке С.

Приводятся примеры практической разработки систем на микроконтроллере ST7, начиная с технического задания на создание контроллера управления объектом и заканчивая разработкой структурной, электрической принципиальной схем, а также алгоритма функционирования объекта. Рассматриваются вопросы выбора датчиков исполнительных устройств и методы тестирования системы управления объектом. Следует подчеркнуть, что технология STMicroelectronics является удобным инструментом для создания цифровых систем контроля и управления, реализации встроенных приложений, характерных для аэрокосмической техники, мобильных систем и т.д.

Материалы книги базируются на результатах исследований и разработок, проводимых в Национальном аэрокосмическом университете «ХАИ», документации, предоставленной университету корпорацией STMicroelectronics, а также информации опубликованной на официальном сайте (http://www.st.com).

Материалы книги структурированы следующим образом. Во введении приводится информация о корпорации STMicroelectronics, ассортименте производимой продукции, областях и перспективах практического использования микроконтроллеров. Первый раздел посвящен обзору семейства ST7, анализу архитектуры микроконтроллеров данного семейства. Приводится описание системы команд и режимов адресации. Во втором разделе рассматриваются вопросы разработки программного обеспечения на языке ассемблер и языке высокого уровня “С” в инструментальной среде STVD7. Приводятся примеры программ. В третьем разделе приведен пример проектирования системы управления и аварийной остановки автомобильного подъемника. Четвертый раздел посвящен проектированию двухканального терморегулятора промышленной установки термостатирования на основе микроконтроллера ST7.

Каждый из разделов заканчивается списком контрольных вопросов и примеров для проверки усвоения материала.

Книга написана коллективом авторов. Предисловие, введение и заключение написаны Харченко В.С, Ореховым А.А., Мпандо П. Первый раздел написан Бабешко Е.В., раздел 2 - Кулановым В.А., Ореховым А.А. и Яновским М.Э., раздел 3 подготовлен совместно Желтухиным А.В., Яновским М.Э. с участием Орехова А.А., раздел 4 – Мазуренко А.В. Редактирование книги выполнено Харченко В.С., Ореховым А.А.

При разработке и реализации пилотного проекта использовались инструментальные средства программирования STVD7, безвозмездно переданные университету корпорацией STMicroelectronics в рамках программы сотрудничества.

Книга адресуется, в первую очередь, специалистам в области проектирования систем различного назначения, в том числе систем критического применения (авиационные системы, атомная энергетика, транспортные коммуникации, медицинские системы и др.). Она может быть полезна, по мнению авторов:

– студентам компьютерных специальностей и специальностей, связанных с проектированием систем управления;

– слушателям соответствующего профиля в системе последипломного образования;

– аспирантам, преподавателям, научным сотрудникам ВУЗов;

– специалистам, осуществляющим проектирование компьютерных средств и систем.

Следует подчеркнуть, что на эту работу оказали непосредственное влияние труды ведущего специалиста корпорации STMicroelectronics Мориса Левансу, в частности, его книга “STDV7 – IDE & ST7 in 10 Steps”, в переводе которой на русский и украинский языки принимали участие Белоглазов С.О., Остроумов С.Б., Прохорова Ю.Н. [30].

Авторы благодарны рецензентам - профессорам Хаханову В.И. и Фурману И.А. за рекомендации, способствовавшие улучшению качества книги, и будут признательны читателям за любые замечания и предложения по книге, которые могут быть присланы по адресу:

61070, г. Харьков, ул. Чкалова, 17, ХАИ, кафедра 503. Телефон: (057) 707-45-03. e-mail: V.Kharchenko@khai.edu

ВВЕДЕНИЕ. ОБЩАЯ ХАРАКТЕРИСТИКА ТЕХНОЛОГИЙ STMicroelectronics

B.1 Развитие корпорации STMicroelectronics

Корпорация STMicroelectronics (STM) один из всемирно признанных лидеров микроэлектроники. Она разрабатывает тысячи различных наименований продуктов. Модули памяти, электронные кредитные карточки, микроконтроллеры, цифровые микросхемы, диоды, тиристоры, силовые транзисторы – это далеко неполный перечень компонент, производимых с маркой компании.

STMicroelectronics была создана в 1987 году путем слияния итальянской фирмы SGS Microelectronica и французской Thomson Semiconducteurs с целью достижения мирового лидерства в области микроэлектроники. Новая компания преследовала стратегию эффективного развития, усиленно вкладывая капитал в научно-исследовательские и опытно-конструкторские работы, формируя стратегические объединения с солидными заказчиками и научным сообществом, широко рекламируя комплексное присутствие в ведущих экономических регионах. Со времени своего создания STM росла быстрее, чем полупроводниковая промышленность в целом, и с 1999 года стала одним из десяти ведущих мировых поставщиков полупроводников.

Вот лишь некоторые итоги развития компании: приблизительно 50000 сотрудников, 16 центров перспективных исследований и разработок, 39 проектировочных и прикладных центров, 17 основных промышленных центров и 78 отделов сбыта в 36 странах.

Штаб-квартира корпорации для работы в Европе и на развивающихся рынках, находится в Женеве. В США штаб-квартира находится в Кэрролтоне, штат Техас; для работы в Азии - в Сингапуре, в Японии – в Токио. Для недавно образовавшегося региона, который включает в себя Гонконг, Китай и Тайвань, штаб-квартира расположена в Шанхае.

С 8 декабря 1994 года, когда STM завершила свое первоначальное предложение акций, акции компании успешно продавались на Нью-Йоркской фондовой бирже. Сейчас компания имеет около 900 миллионов акций, из которых 72.4% распродаются на различных фондовых биржах. Балансом акций управляет компания STMicroelectronics Holding 2 B.V., акционерами являются Cassa Depositi e Prestiti , Finmeccanica of Italy и Areva of France.

Сегодня STM это одна из самых крупных в мире полупроводниковых компаний с оборотом 9.85 миллиардов долларов в 2006 году, более 12 миллиардов в 2007 году и лидерством во многих областях. Например, согласно последним данным, STM входит в пятерку крупнейших полупроводниковых компаний и занимает ведущую позицию по продажам интегральных схем, аналоговых специфических стандарти-

зованных продуктов. Помимо этого, STM –компания номер один в разработке камер для мобильных телефонов, номер два – дискретных компонент и аналоговых устройств и номер три – флеш-памяти. Она также занимает заметное место в применении мобильных компонентов и беспроводных технологий. STM ведущий поставщик полупроводников для компьютерных приставок к телевизору и устройств управления режимом электропитания.

B.2 Производимая продукция и производственные мощности

Компания предлагает одну из широчайших в мире номенклатур продукции, которая насчитывает более чем 3000 основных типов продуктов. Тщательно сбалансированный портфель продукции и активов включает в себя как продукты, связанные с конкретным применением, имеющие интеллектуальное содержание, так и многосегментные продукты, которые классифицируются, начиная от дискретных устройств до высокоэффективных микроконтроллеров.

Товарооборот компании четко сбалансирован между пятью высокоразвитыми промышленными секторами: системы связи (38%), вычислительный (17%), потребительский (16%), автомобильный (15%) и промышленный (14%). STM исследовала и продолжает улучшать методологию платформенного дизайна сложных интегральных схем для таких приложений, как компьютерные приставки к телевизору, защита кредитных карточек с микропроцессором, мобильная связь, с использованием различных средств информации, которые сокращают продолжительность разработки и расходы.

Подход, основанный на сбалансированном портфеле активов, позволяет компании учитывать потребности всех пользователей микроэлектроники, от всемирных стратегических заказчиков, для которых STM является предпочтительным партнером по главным проектам System-on-Chip (SoC - систем на кристалле), до промышленных предприятий, которые нуждаются в полностью завершенных универсальных устройствах.

STM имеет передовое производственное оборудование на четырех континентах. В настоящее время компания имеет 200-миллиметровые полупроводниковые пластины в Agrate Brianza (Италия), Catania (Италия), Crolles (Франция), Phoenix (США), Rousset (Франция) и Сингапуре. В производстве интегральных схем на 300-миллиметровых полупроводниковых пластинах STM применяет пневмолинию управления, называвемую Crolles2. Оборудование в Wuxi City(Китай) является совместным предприятием STM и Hynix Semiconductor по производству флеш-памяти и динамического ОЗУ. Деятельность Crolles2 привела к совместной программе между STM, Freescale и NXP для развития передовой технологии комплементарных металло-оксидных полупроводников (CMOS). Полупроводниковые

пластины дополнены эффективной сборкой и тестовым оборудованием, размещенном в Китае, Малайзии, Марокко и Сингапуре.

B.3 Исследовательские программы и результаты

Со времени своего создания STM взяло на себя серьезные обязательства по отношению к научно-исследовательской деятельности. В 2006 году компания вложила 1.667 миллиарда долларов в исследовательские проекты, что составило 16.9% выручки за год. Эти усилия принесли свои плоды в виде 607 заявок на патенты в 2006 году.

STM участвует в многочисленных научно-исследовательских проектах во всем мире и в ведущих программах Евросоюза. STM – лидер в важных промышленных инициативах, например, ENIAC(European Nanoelectronics Initiative Advisory Council) на Европейском уровне или “Poles de Competitivite” на региональном уровне.

STM развила всемирно известную сеть стратегических объединений, включая разработку продукта совместно с ведущими заказчиками, развитие технологий совместно с заказчиками и другими производителями полупроводников, объединения по развитию оборудования и автоматизированного проектирования с главными поставщиками. Это промышленное партнерство дополнено широким диапазоном исследовательских программ, которые сопровождаются ведущими университетами и исследовательскими институтами во всем мире. Вдобавок к богатым активам собственных технологий, компетенции и дополнительной экспертизе множества тщательно выбранных стратегических партнеров, STM развила способность предлагать наиболее развитые технологические решения заказчикам во всех отраслях электронной промышленности.

Многие из исследовательских программ и программ развития STM, управляются AST-организацией (системой, построенной на основе последних технических достижений). Ее миссия заключается в развитии стратегической системы знаний, которые не позднее 3-5-ти лет будут необходимы для выпуска продукции STM. Среди значительных недавних достижений STM, имеются новаторские технологии для «цифровых» потребителей, для использования сетей, для безопасности мобильных устройств и для реализации внутрикристального межсоединения.

Приверженность STM к динамичному и продуманному развитию позволило ей заслужить престижные награды по всему миру. С 1991 года компания получила более чем 70 наград в различных сферах деятельности, включая решение проблем качества, социальных проблем и защиты окружающей среды.

Взятые на себя обязательства компании по защите окружающей среды увенчались значительным сокращением потребления энергии, воды, бумаги, опасных химических веществ, увеличением повторного использования отходов и значительным влиянием на развитие озеле-

нения. STM непрерывно расширяет границы повышения ответственности каждого члена корпорации (Corporate Responsibility), добиваясь значительных достижений в ключевых отраслях, таких как профессиональная гигиена, гигиена труда и безопасность, включая аттестацию 16-ти промышленных рабочих мест и 4-х непромышленных рабочих мест по требованиям международного стандарта OHSAS 18001; использование маломощной технологии в широкой номенклатуре продукции; укрепление программы цифровой унификации, которая сопровождается компанией STMicroelectronics.

B.4 Промышленное партнерство

Со времени своего зарождения STM стала лидером в строительстве стратегических объединений и достигла успехов в развитии эффективных отношений с клиентами, поставщиками, конкурентами, университетами, исследовательскими институтами и Европейскими исследовательскими программами. Сегодня стратегические объединения и промышленное партнерство становятся все более важными факторами в достижении успеха в полупроводниковой промышленности.

STM вошла в несколько стратегических объединений с клиентами, включая Alcatel, Bosh, Hewlett-Packard, Marelli, Nokia, Nortel, Pioneer, Seagate, Siemens VDO, Thomson, Western Digital и др. Клиентские объединения снабжают STM ценными системами и технологиями ноу-хау, а также доступом к рынкам с ведущими продуктами, в то время как сама компания позволяет клиентам в той или иной степени наблюдать за развитием продукта и получать доступ к технологическим процессам и к промышленной инфраструктуре компании. Сейчас STM активно ведет работу по извлечению выгоды из своего опыта и широты своих технологических активов с тем, чтобы расширить количество клиентских объединений, нацеливаясь на изготовление оборудования в США, Европе и Азии.

В то время как компания решительно продолжает конкурировать по продажам, партнерство с другими полупроводниковыми промышленными производителями позволяет ей увеличивать свои капиталовложения в дорогостоящие исследования и разработки, промышленные ресурсы с взаимной выгодой для технологического развития. С 1992 года STM сотрудничает с Philips по совместному развитию технологии комплементарных металло-оксидных полупроводников в Crolles(Франция). В 2003 году компания начала партнерство с Freescale и Philips для совместного исследования и развития технологии комплементарных металло-оксидных полупроводников, чтобы снабдить 300мм подложку микросхемами от 90-нм до 32-нм. Объединение Crolles2, которое обычно оперирует 300-мм платой с пневмолинией управления, было создано в Crolles (Франция) с целью ускорить развитие технологий будущего и их распространение в полупроводниковой промышленности.

Компания STM – долговременный лидер в беспроводных технологиях. В 2002 году она начала взаимодействовать с Texas Instruments, чтобы определить и выдвинуть общедоступный стандарт для беспроводных технологий применительно к процессорам интерфейсов. Эта инициатива сейчас поддерживается большим числом компаний и известна, как MIPI Alliance, с STM, ARM, Nokia, Texas Instruments в качестве членов-учредителей. Сейчас альянс включает около 92 членов, которые сотрудничают как гибкие промышленные лидеры, стремящиеся определить и выдвинуть доступные стандарты для интерфейсов и мобильного применения процессоров.

Неразрушающаяся постоянная память – стратегический сектор продукции для компании STM. В этой отрасли STM в течение последних лет взаимодействует с Hynix в рамках совместной программы развития для NAND (на элементах И-НЕ) Flash – технологий и продуктов. Учреждено стратегическое объединение с корпорацией Intel по изделиям для беспроводных приложений. Кроме того, заключено соглашение с Frescale для совместной разработки микроконтроллеров со встроенной флеш-памятью в 90nm – технологическом поколении.

STM внедрила совместные программы развития с ведущими поставщиками, такими как Air Liquide, Applied Materials, ASM Lithography, Axalto, Canon, Hewlett-Packard, KLA-Tencor, LAM Research, MemC и Teradyne. STM принимает участие в совместных Европейских исследовательских программах, таких как, MEDEA+ (общеевропейская программа исследовательской работы по развитию технологий в микроэлектронике), ITEA2 (информационные технологии для европейского развития) - стратегическая программа, ориентированная, главным образом, на разработку программных систем и сервисов.

STM играет ведущую роль в двух недавно созданных Европейских Технологических Платформах: ENIAC (Европейский консультативный совет нанотехнологий), который был учрежден для обеспечения программы оперативных исследовательских работ по нанотехнологиям, и ARTEMIS (передовые исследования и технологии для встроенных систем).

Кроме того, STM сотрудничает со многими университетами во всем мире, включая Европу, США, Китай.

В 1998 году STM презентовала свою испытательную установку в Shenzhen (Китай), которая действует в рамках совместного предприятия по соглашению между STM и SHIC (высокотехнологическая промышленная компания в Shenzhen).

В 2004 году STM объявила о совместной деятельности с Hynix для строительства интерфейсного оборудования в Wuxi City (Китай). Оно базируется на расширении продукта – флеш-памяти на элементах И-НЕ. В процессе партнерства между компаниями представлена линия производства 200-миллиметровой подложки, которая была введена в конце 2006 года, и линейка продукции 300-миллиметровой платы (2007г.).

B.5 Образовательная и техническая программа в области микроэлектроники

Университетская составляющая деятельности STM направлена на развитие и использование программ, созданных в компании и основанных на принципах управления и обучения в соответствии с необходимостью повышения квалификации. Компания тщательно работает с различными учебными центрами, чтобы предложить гибкую систему обучающих программ, разработанных для того, чтобы выявлять необходимость в тренировке и обучении отдельных людей и компании в целом.

Среди тренировочных программ предлагается учебная программа для служащих компании STM, а так же для нештатных инженеров. Основная цель этой технической программы – повысить компетенцию специалистов в области технологий и управления микроэлектронной промышленности.

Эта уникальная программа является плодом партнерства между STM и двумя знаменитыми французскими школами инженеров: “L’Ecole Nationale Superieure des Mines” de Saint-Etienne (Сент-Этьен) и “l’Ecole Centrale” Marseille (Марсель). Это обеспечивает инженеров техническими знаниями и знаниями, относящимися к управлению и нацеленными на исполнение ключевой роли, в соответствии с сегодняшними потребностями микроэлектроники. Чтобы быть на уровне передовых технологий микроэлектроники, каждый год совершенствуется вся программа с помощью промышленных экспертов, преподавателей и исследователей, развиваются и улучшаются связи между теоретическими курсами и практикой, совершенствуется совместная работа промышленных экспертов и поставщиков.

Программа делится на две части:

Часть 1: Основы и практические приложения:

- Метод и технология: физическая характеристика инструментальных средств и шаги промышленного процесса.
- Развитие интегральных микросхем: проектирование инструментальных средств, тест и работа прикладной части.
- Производство и инструментальные средства управления: управление промышленным оборудованием, промышленная технология, системы качества и надежности.

Часть 2: Шестимесячные курсы непосредственно на предприятии, ориентация на специальную дисциплину, относящуюся к этой программе.

STMicroelectronics выступает за долговременное сотрудничество с университетами и видит реализацию такого сотрудничества посредством последовательности трех шагов:

- первый шаг – обучение студентов в рамках учебного центра;

- второй шаг – создание консультационного/экспертного дизайн - центра на базе университета;

- третий шаг – сотрудничество в области высоких технологий, совместная работа над проектами, обмен опытом и т.д.

На первом этапе сотрудничества организуется проведение курса по микроконтроллерам STM на базе университетов. STM предоставляет материалы курса по микроконтроллеру ST7, тренировочные наборы, проводит обучение преподавателей. Университет организует курс на базе собственных помещений и преподавателей.

Второй этап предполагает создание дизайн - центра. Университет организует группу экспертов, которая может оказывать консультации по микроконтроллерам STM, выполнять заказные разработки, проводить последипломное обучение новейшим продуктам.

Третий этап требует определения возможных путей дальнейшего сотрудничества, выбора экспертов и тем для проектов и исследований (работа над блоками, создание системных решений, совершенствование технологии и т.д.).

В 2007 году STM приступило к реализации этой программы в Украине, подписав соглашение о сотрудничестве с ведущими университетами Киева, Харькова, Львова, Одессы и др.

Сочетание производственной мощи и интеллектуальной собственности, промышленного и научного партнерства, а также одна из самых широких номенклатур продукции делают сегодня STM мировым лидером в разработке и внедрении полупроводниковых решений прикладных задач микроэлектроники.

По мере того, как новаторство продолжает продвигать электронную промышленность, комбинация известных мировых заказчиков, системы мастерства и ведущих технологий гарантирует, что компания сохранит мировые лидерские позиции в микроэлектронике.

РАЗДЕЛ 1. ОБЗОР СЕМЕЙСТВА ST7

1.1. Место ST7 в линейке микроконтроллеров

Корпорация STMicroelectronics занимает пятое место в мире по продажам полупроводниковой техники (после Intel, Samsung Electronics, Toshiba Semiconductors и Texas Instruments). Ассортимент выпускаемых микроконтроллеров достаточно широк, основные семейства приведены на рис. 1.1. (MIPS – миллион инструкций в секунду).

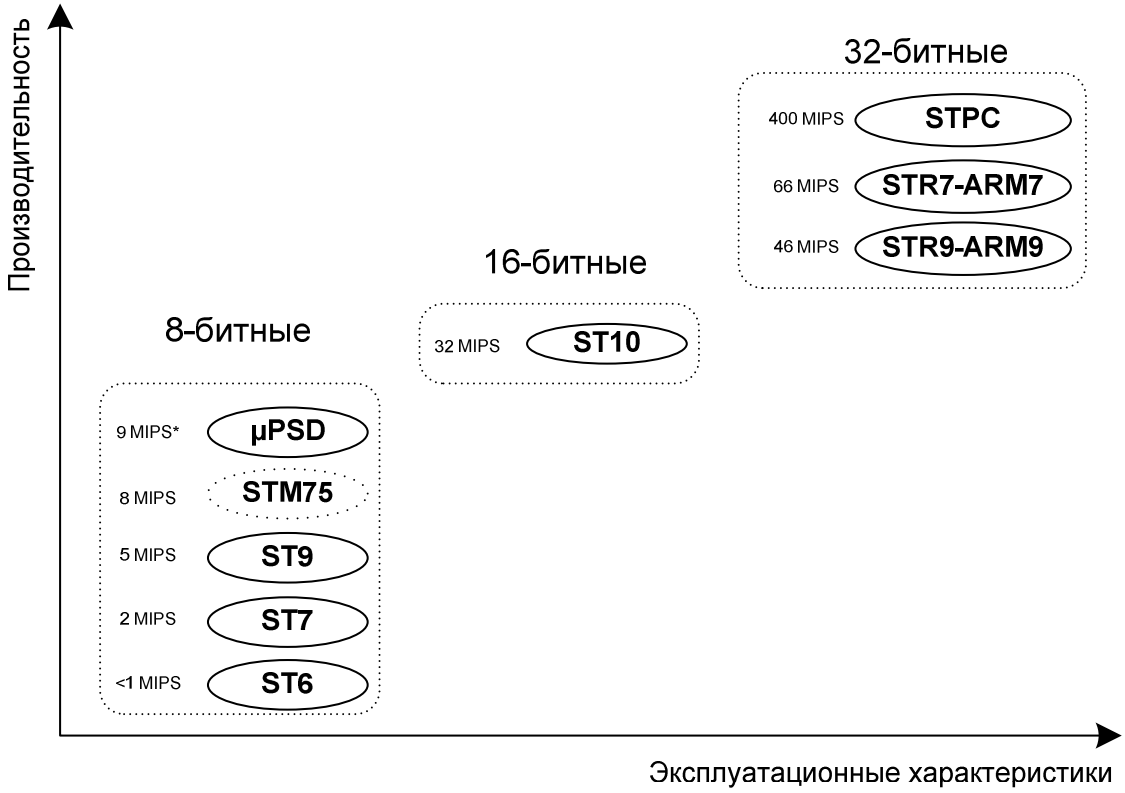


Рис. 1.1. Ассортимент микроконтроллеров STMicroelectronics

Согласно статистическим данным, 8-разрядные микроконтроллеры занимают около 40% рынка микроэлектронных устройств, предназначенных для встраиваемых систем. Несмотря на бурный рост в сфере использования 32-разрядных микроконтроллеров, их объем продаж почти вдвое меньше аналогичных показателей для 8-разрядных микроконтроллеров. Иными словами, в большинстве недорогих массовых применений 8-разрядные архитектуры по-прежнему остаются самыми востребованными.

STM выпускает три семейства 8-разрядных микроконтроллеров – ST6, ST7 и ST9 (табл. 1.1). В настоящее время идет разработка нового семейства STM75.

Микроконтроллеры ST7 являются самым многочисленным семейством STMicroelectronics.

Таблица 1.1

Семейства 8-разрядных микроконтроллеров STMicroelectronics

Тип	Характеристики
ST6	Малая потребляемая мощность, от 1.2 до 8 Кб ROM, таймер, АЦП, сторожевой таймер, 5 векторов прерывания
ST7	Система команд промышленного стандарта, от 256 до 3 Кб RAM, от 4 до 60 Кб ROM, АЦП, SPI, 16-битный таймер, 16 векторов прерывания
ST9	Большое количество внутренних регистров, различные режимы адресации, контроллер приоритета прерываний, контроллер DMA, от 16 до 128 Кб ROM, более 256 б RAM, 128 векторов прерывания

Производится более 50 моделей, которые можно разделить на следующие группы:

– базовые микроконтроллеры общего применения с 1 или 2 таймерами, интерфейсами SPI и SCI;

– микроконтроллеры, предназначенные для обработки аналоговых сигналов, которые кроме всего вышеперечисленного содержат 8-разрядный АЦП и (в ряде моделей) встроенную память EEPROM для хранения данных;

– группы микроконтроллеров, имеющие в своем составе дополнительные последовательные интерфейсы: I^2C, CAN, USB;

– специализированные микроконтроллеры.

К микроконтроллерам общего применения относятся семейства ST7Fox, ST7Lite, ST7226x, ST7232x и ST7236x (рис. 1.2).

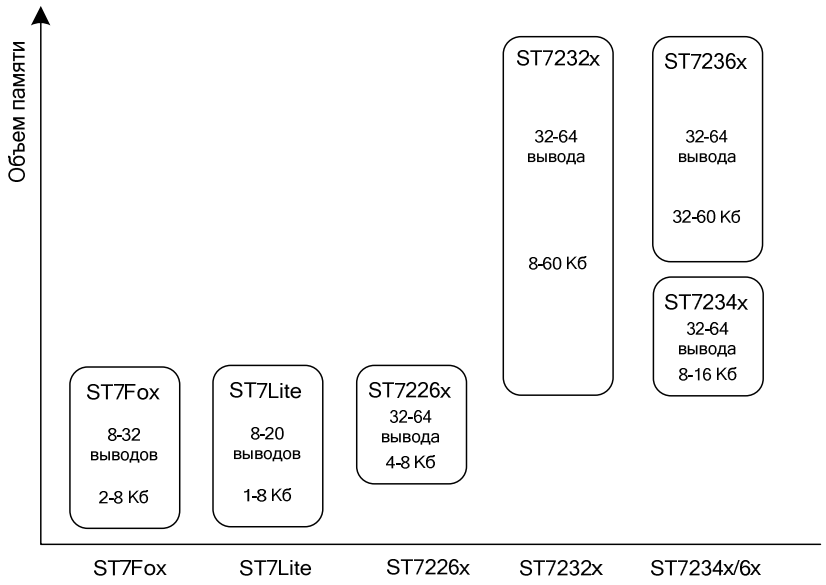


Рис. 1.2. Микроконтроллеры общего применения

Среди специализированных микроконтроллеров следует выделить семейство ST7MCx, предназначенное для управления двигателями различного типа (рис 1.3). Примерами микросхем с дополнительными интерфейсами являются семейство ST7LNBx (интерфейс DiSEqC для управления различным спутниковым оборудованием), несколько семейств с поддержкой USB различных спецификаций (ST7SCR, ST7263B, ST7265x, ST7267x, ST7268x), семейство ST7256x (интерфейс CAN).

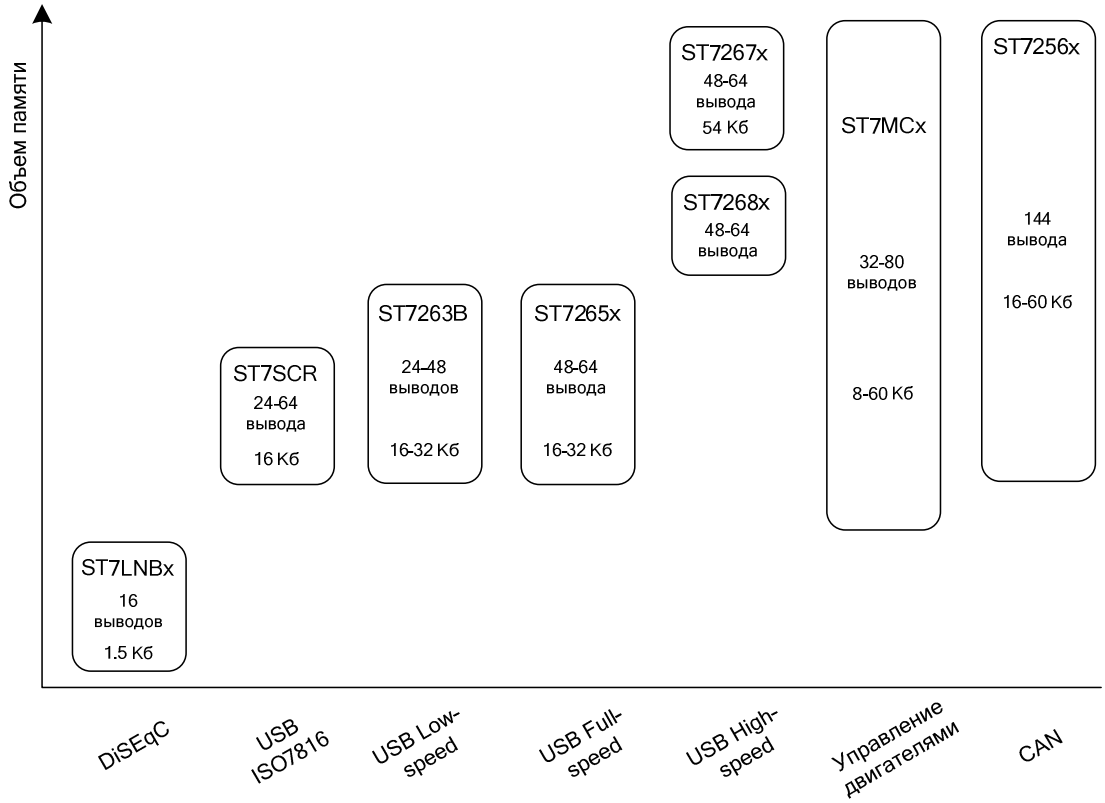


Рис. 1.3. Специализированные микроконтроллеры и микроконтроллеры с дополнительными интерфейсами

Микроконтроллеры семейства ST7 можно отнести к так называемому low-end сегменту рынка, сегменту наименее производительных микроконтроллеров. Аналогами семейства ST7 являются микроконтроллеры MC68HC05/08 фирмы Motorola, PIC16C фирмы Microchip Technology и 78K фирмы NEC.

1.2. Система обозначений ST7

Микроконтроллеры ST7 выпускаются в нескольких исполнениях, которые отличаются допустимым диапазоном рабочих температур, объемом памяти, типом корпуса и т.п. Для определения требуемого варианта используется специальная система обозначений (маркировка). Принципы маркировки микроконтроллеров ST7 показаны на рис. 1.4.

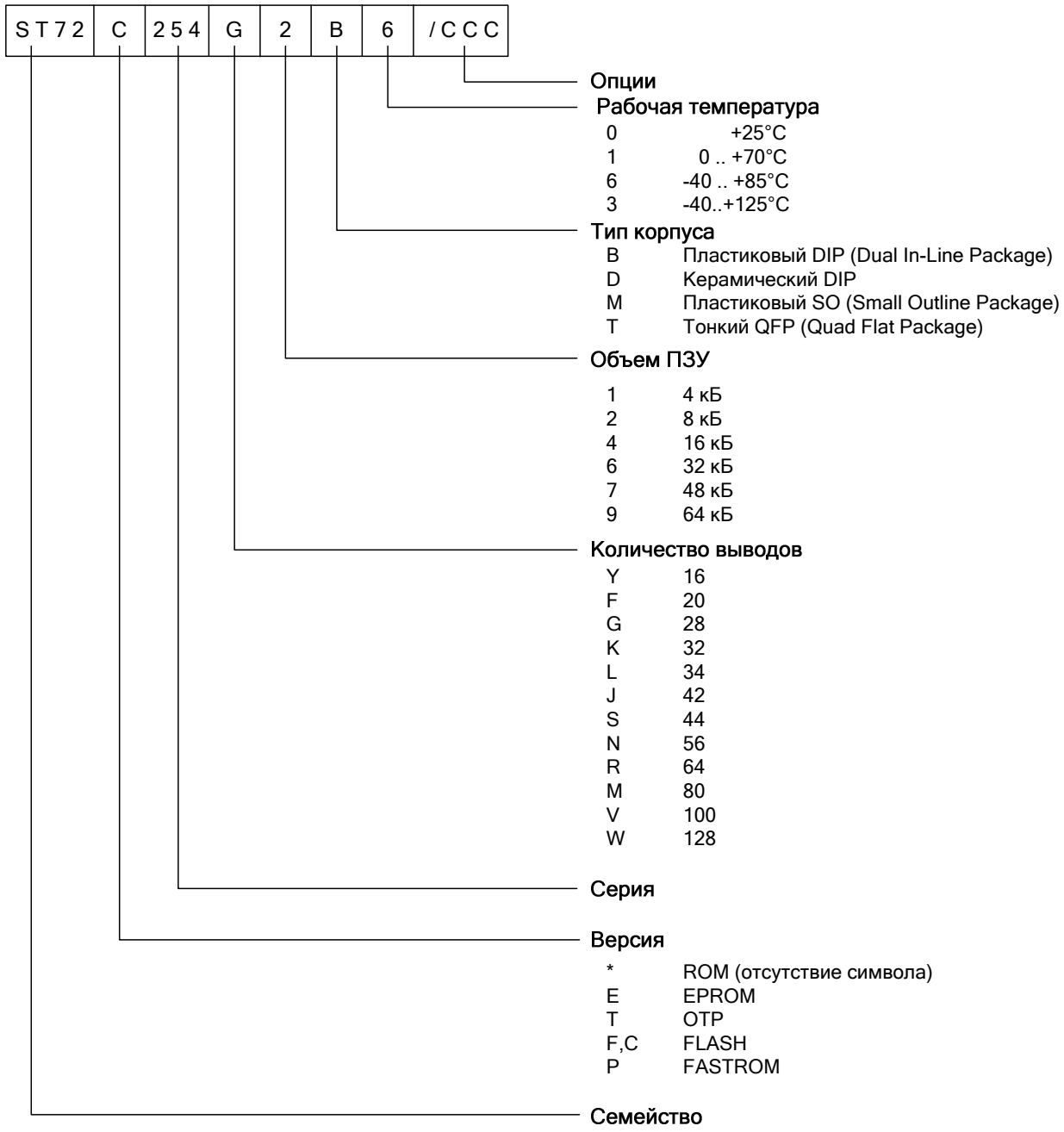


Рис. 1.4. Маркировка микроконтроллеров ST7

1.3. Архитектура ST7

1.3.1. Ядро

Семейство ST7 основывается на 8-разрядной архитектуре промышленного стандарта. Ядро имеет фон-неймановскую архитектуру, состоит из 8-битного арифметико-логического устройства (АЛУ), 6 внутренних регистров, блока управления (рис. 1.5).

Ядро взаимодействует со встроенным генератором тактовых импульсов, блоком сброса, шинами адреса и данных, периферийными устройствами и контроллером прерываний.

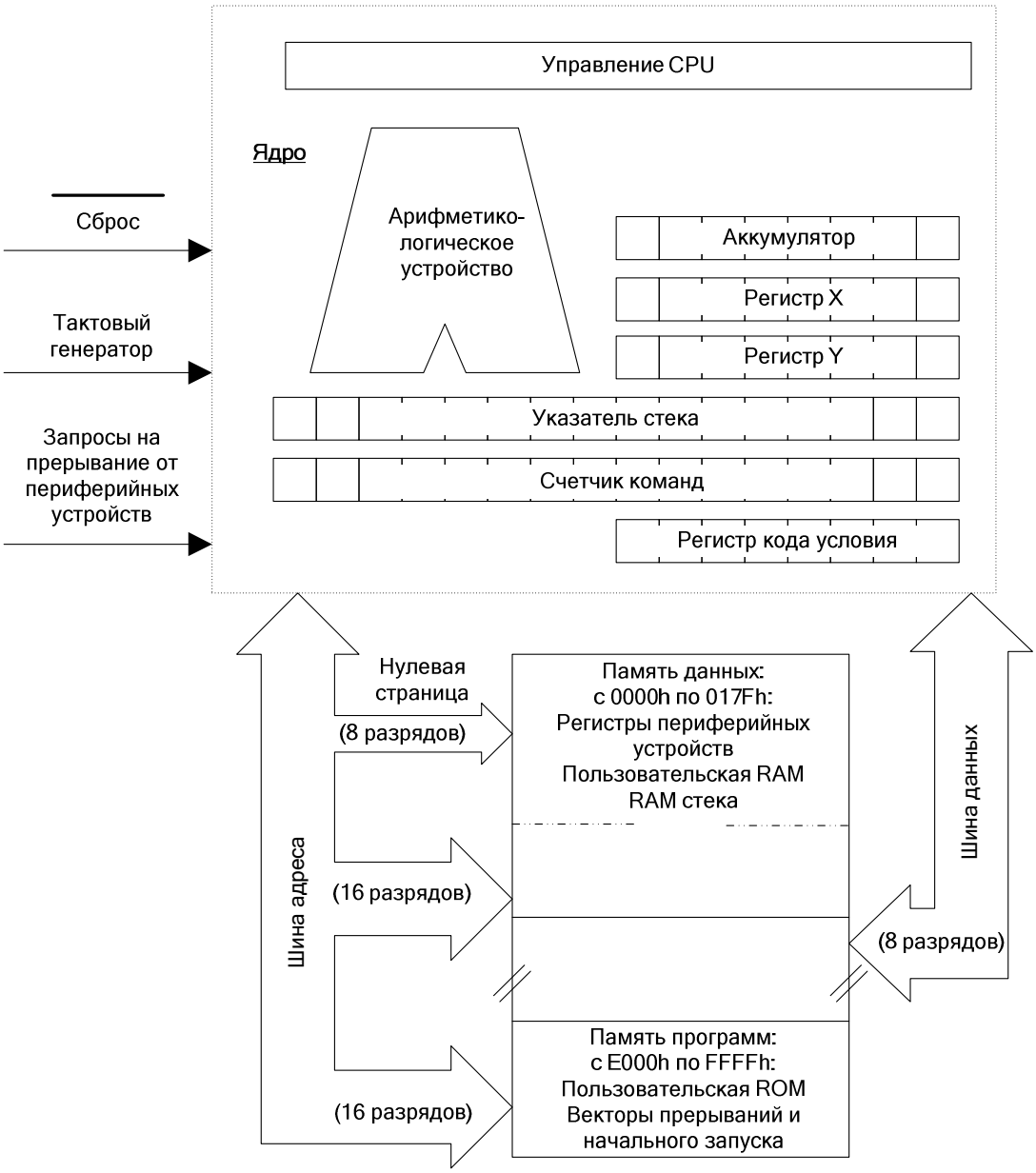


Рис. 1.5. Ядро и адресация микроконтроллера ST72251

Вход $\overline{Сброс}$ служит для подачи внешнего сигнала начального запуска микроконтроллера (низкий уровень потенциала). На этом входе также устанавливается низкий уровень потенциала, если запуск микроконтроллера вызывается внутренними причинами (включение напряжения питания, сигнал блока контроля функционирования центрального процессора и т.п.)

1.3.2. Адресное пространство

Поскольку ядро ST7 имеет фон-неймановскую архитектуру, в микроконтроллере существует единое адресное пространство, в котором располагаются программа, данные и регистры ввода/вывода.

Преимущества такой архитектуры следующие:

– доступ к любому байту программы, данных, ввода/вывода с помощью одних и тех же инструкций;

– доступ к константным данным в памяти с помощью тех же инструкций;

– упрощение программирования вследствие использования двух первых преимуществ.

16-битная шина адреса позволяет адресовать 65536 байт данных, что является достаточным для большинства приложений ST7.

Для повышения эффективности кода адресуемое пространство разделено на две части:

– адреса от 0 до 255 (0FFh) принадлежат так называемой нулевой странице памяти; для доступа к данным этой страницы достаточно 8-битного регистра адреса;

– для остальных адресов, от 256 (80h) до 65535 (0FFFFh), используется 16-битный регистр адреса.

Использование нулевой страницы предоставляет возможность значительно увеличить скорость работы, если все часто используемые данные постоянно располагаются в нулевой странице памяти. Кроме того, все входы/выходы ST7 всегда расположены в этой области памяти.

1.3.3. Внутренние регистры

Ядро ST7 содержит шесть внутренних регистров: A, X, Y, PC, SP и CC (рис. 1.6).

A – аккумулятор (accumulator) – 8-битный регистр общего назначения, используемый для хранения:

– операндов;

– результатов логических и арифметических операций.

X и **Y** – индексные регистры – два 8-битных регистра, используемые для:

– формирования адреса при индексной адресации;

– хранения временных данных.

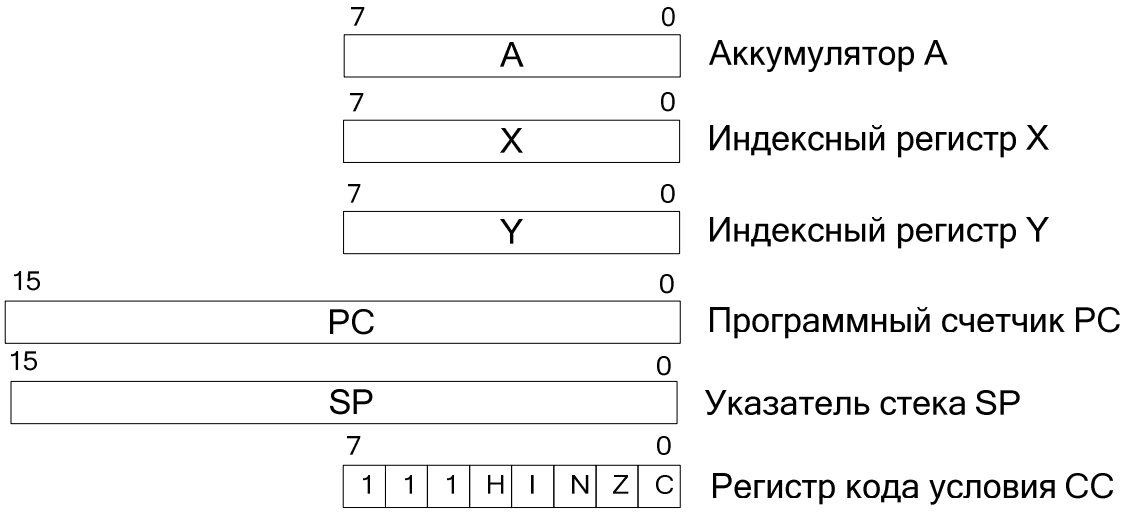


Рис. 1.6. Регистры микроконтроллеров семейства ST7

PC – счетчик команд (program counter) – 16-битный регистр, используемый для хранения адреса инструкции, которая будет выполнена следующей. Состоит из двух 8-битных регистров PCL и PCH. При запуске микроконтроллера в PC автоматически загружается адрес первой команды выполняемой программы (так называемый вектор начального запуска) из двух последних ячеек адресуемой памяти с адресами FFFE-FFFFh. Как было отмечено выше, ST7 может адресовать до 64К памяти.

SP – указатель стека (stack pointer) – 16-битный регистр. Подробное описание регистра приведено в разделе 1.3.4

CC – код условия (code condition) – это регистр, который в других микроконтроллерах обычно называется регистром состояния. В нем используются младшие 5 бит, их назначение приведено в таблице 1.2

Таблица 1.2

Назначение битов регистра кода условия

Бит	Символ	Имя	Описание
7	-	Не используется	Равен 1
6	-	Не используется	Равен 1
5	-	Не используется	Равен 1
4	H	Признак переноса для полубайтов	H=1, когда есть перенос (заем) в старший полубайт (из старшего полубайта) в результате выполнения операции сложения (вычитания)
3	I	Маска прерываний	I=1 отключает прерывания
2	N	Признак отрицательного значения	N=1, если результат последней операции отрицательный
1	Z	Признак нулевого значения	Z=1, если результат последней операции равен нулю
0	C	Признак переноса	C=1, когда результат операции сложения (вычитания) выходит за границы 8-разрядной сетки

Признак переноса **C** устанавливается, если результат операции сложения превышает значение 0FFh, или результат операции вычитания меньше 0h. Данный признак используется в арифметических операциях с 16-разрядными (и более) числами, чтобы увеличить на 1 (при сложении) или уменьшить на 1 (при вычитании) результат сложения

старших байтов, если при операции с младшими байтами результат вышел за границы 8-разрядной сетки. Признак также может устанавливаться/сбрасываться с помощью команд SCF и RCF соответственно, проверка признака осуществляется командами JRC и JRNC. Кроме того, признак может устанавливаться при выполнении команд сдвига/вращения.

Признак нулевого значения **Z** устанавливается при нулевом результате операции. Таким образом, команды «OR A, #0» и «AND A,$FF» могут быть использованы для проверки содержимого аккумулятора: в случае нулевого значения признак Z будет установлен. Проверку признака можно осуществить с помощью команд JREQ и JRNE.

Признак отрицательного значения **N** устанавливается, когда старший (7-й) бит аккумулятора равен 1. Проверка значения признака осуществляется командами JRMI и JRPL.

Маска прерываний **I** не устанавливается в результате каких-либо арифметических операций. Значение данного бита регистра CC считывается дешифратором при возникновении запроса на прерывание. Если I=1, то прерывание только фиксируется, но его обработка не выполняется. Маска управляется командами RIM (сброс), SIM (установка) и IRET (возврат из обработчика прерываний), проверка маски осуществляется командами JRM и JRNM.

Признак переноса для полубайтов **H** устанавливается при переносе (займе) в старший полубайт (из старшего полубайта) в результате выполнения операции сложения (вычитания). Данный признак обычно используется при реализации алгоритмов обработки чисел, в которых каждый разряд представляется одним байтом (например, двоично-десятичные числа). Проверка признака осуществляется командами JRH и JRNH.

Значения регистров при запуске микроконтроллера показаны на рис. 1.7.

A=*XX*h — *X*-неопределенное значение

X=*XX*h

Y=*XX*h

PC=Вектор начального запуска@FFFE-FFFFh

CC=111*X*1*XXX* (**I**=1)

SP=Старший адрес стека

Рис. 1.7. Начальные значения регистров

1.3.4. Работа со стеком

Стек предназначен для сохранения контекста центрального процессора (регистры PC, CC, A, X) во время вызова подпрограмм и обработчиков прерываний, а также для хранения пользовательских данных (с помощью инструкций PUSH и POP). Порядок сохранения регистров контекста показан на рис. 1.8.

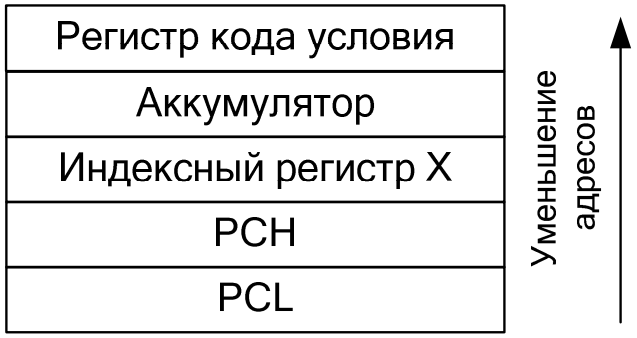


Рис. 1.8. Порядок сохранения регистров в стеке при прерывании

Следует отметить, что при автоматическом сохранении контекста содержимое регистра Y не сохраняется. Если при вызове подпрограмм и обработчиков прерываний требуется сохранить и затем восстановить содержимое регистра Y, то это необходимо сделать вручную с помощью команд PUSH и POP соответственно.

В случае переполнения указателю SP присваивается максимальный адрес стека, значение по этому адресу затирается новыми помещаемыми в стек данными. Переполнение стека при этом никак не показывается, поэтому при работе со стеком необходимо отслеживать объем помещаемых в стек данных и избегать переполнения.

Положение указателя стека можно изменить на произвольное значение в пределах допустимого диапазона, загрузив его из аккумулятора командой «LD SP, A». Кроме того, можно установить начальное значение указателя стека с помощью команды «RSP».

Размер стека и его расположение в памяти зависит от типа микропроцессора, например:

– ST72254: 128 байт (от 0100h до 017Fh);
– ST72521: 256 байт (от 0100h до 01FFh);
– ST7FoxF1: 128 байт (от 0180h до 01FFh);
– ST7Lite0: 64 байта (от 00C0h до 00FFh).

При установке микроконтроллера в начальное положение регистр SP указывает на старший адрес стека. После записи байта в эту ячейку содержимое регистра SP уменьшается на единицу, адресуя незаполненную ячейку стека. Таким образом, стек заполняется в направлении уменьшения адресов.

На рис. 1.9 показано содержимое стека и состояние указателя стека SP при последовательном выполнении различных операций.

1.3.5. Память

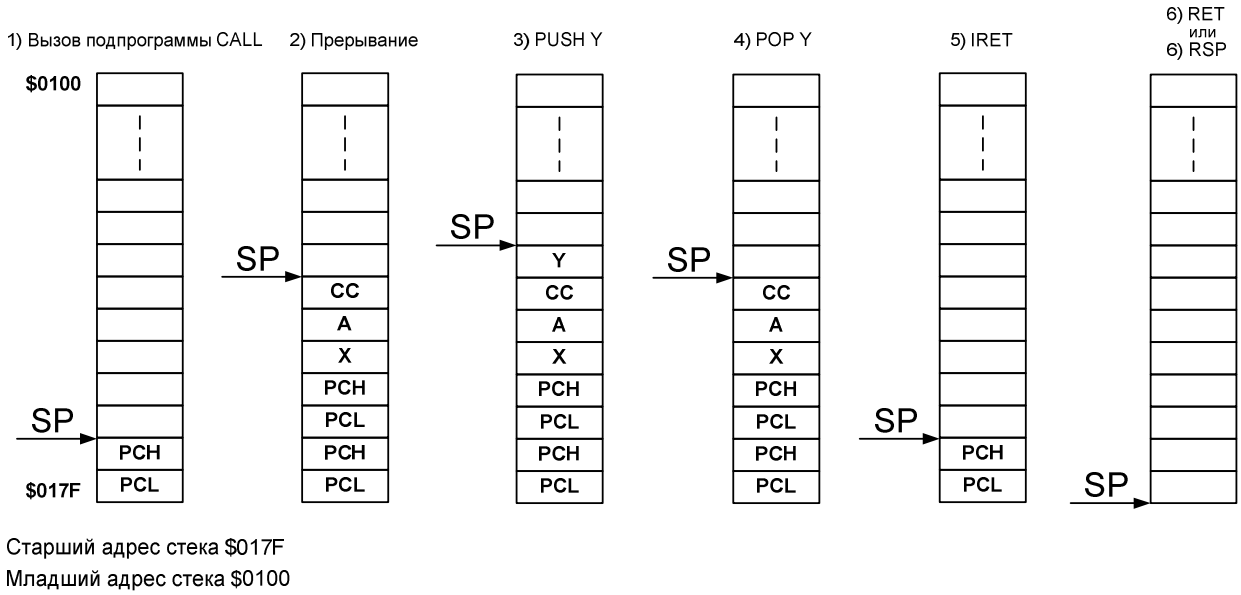


Рис. 1.9. Иллюстрация работы со стеком

Память микроконтроллеров ST7 разделена на следующие блоки (рис. 1.10):

– регистры периферийных устройств: порты ввода/вывода, таймер (TIM), АЦП (ADC), сторожевой таймер (WDG), SPI, I^2C и т. д.;

– RAM (оперативная память);

– стек: от 128 до 256 байт (в зависимости от модели);

– данные EEPROM (до 256 байт);

– память программ;

– векторы прерываний и начального запуска.

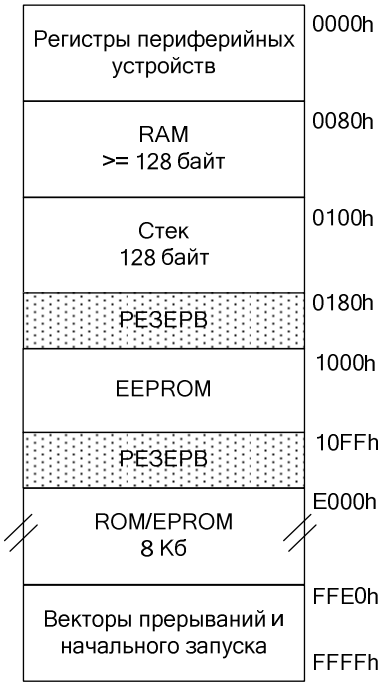


Рис. 1.10. Память микроконтроллеров ST7

Память программ микроконтроллеров семейства ST7 может быть различных типов: OTP, EPROM, FASTROM, ROM или Flash. Объем памяти программ – от 1,5 до 60 Кбайт. Наличие модификаций микроконтроллеров с памятью Flash значительно повышает скорость разработки и упрощает обновление ПО, что в конечном итоге снижает общие затраты. Такие микроконтроллеры позволяют производить программирование непосредственно в системе, для записи новой программы не требуется извлекать микроконтроллер из устройства и вставлять в разъем программатора.

Встроенное ОЗУ имеет объем от 128 байт до 6,5 Кбайт. Некоторые модели имеют встроенную EEPROM данных объемом 128 или 256 байт.

1.4. Режимы адресации

1.4.1. Перечень режимов адресации

Микроконтроллеры семейства ST7 выполняют набор операций над операндами, размещенными в аккумуляторе, индексных регистрах X и Y, памяти. Предоставляется 17 различных режимов адресации, которые можно разделить на следующие 7 групп:

– адресация кодом команды, или безадресный режим (inherent);
– непосредственная адресация (immediate);
– прямая (direct);
– индексированная (indexed);
– косвенная (indirect);
– относительная (relative);
– битовые операции (bit operations).

Для минимизации длины инструкции в байтах большинство режимов адресации имеют длинный и короткий способы. При длинном способе адресации адрес задается 16-ю битами, при коротком – 8-ю. Таким образом, длинный режим адресации позволяет получить доступ к любому байту из адресуемых 64 кб адресного пространства, а короткий – предоставляет доступ только к нулевой странице памяти (диапазон 00..FF).

Выбор способа адресации осуществляется автоматически без дополнительного участия программиста.

Инструкции CLR, CPL, NEG, BSET, BRES, BTJT, BTJF, INC, DEC, RLC, RRC, SLL, SRL, SRA, SWAP работают только с коротким способом адресации.

Ниже приведено описание режимов адресации семейства микроконтроллеров ST7.

1.4.2. Адресация кодом команды или безадресный режим

Данный режим адресации используется для команд, которые не

требуют операндов (табл. 1.3). Код операции полностью определяет, какие действия должны быть выполнены. Инструкции такого типа занимают 1 байт памяти.

Таблица 1.3

Инструкции, использующие безадресный режим адресации

Инструкция	Описание
NOP	Пустая операция
TRAP	Программное прерывание
WFI	Ждать прерывания (Wait For Interrupt)
HALT	Остановка
RET	Возврат из подпрограммы
IRET	Возврат из подпрограммы обработки прерывания
SIM	Установка маски прерывания
RIM	Сброс маски прерывания
SCF	Установка флага переноса
RCF	Сброс флага переноса
RSP	Сброс указателя стека
PUSH/POP	Протолкнуть в стек/Вытолкнуть из стека

1.4.3. Непосредственная адресация

В данном режиме адресации 8-разрядный операнд следует сразу за кодом операции (табл. 1.4). Признак режима непосредственной адресации – символ **#** перед операндом.

Таблица 1.4

Инструкции, использующие непосредственную адресацию

Инструкция	Описание	Пример
LD	Загрузка	LD A, #$3
CP	Сравнение	CP A, #%00001000
BCP	Битовое сравнение	BCP A, #%01011010
AND, OR, XOR	Логические операции	AND A, #$D6
ADC, ADD, SUB, SBC	Арифметические операции	ADD A,8

В примере, показанном на рис. 1.11, производится загрузка константы $10, указанной непосредственно, в аккумулятор.

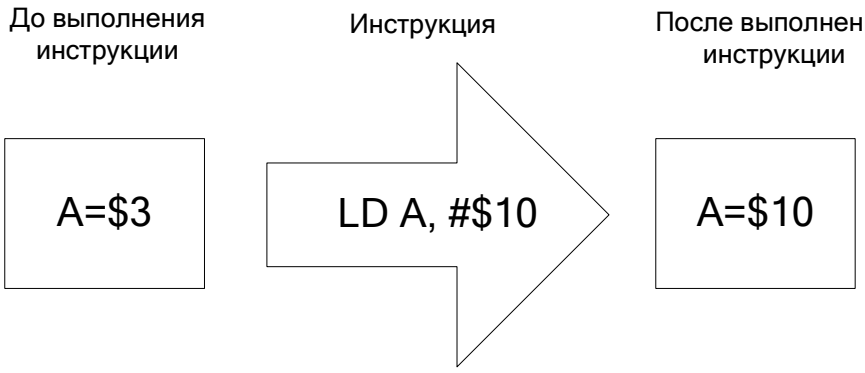


Рис. 1.11. Пример непосредственной адресации

1.4.4. Прямая адресация

При прямой адресации 8- (короткий способ) или 16-разрядный (длинный способ) адрес операнда задается сразу после кода операции. В таблице 1.5 приведены инструкции, использующие данный режим адресации с указанием допустимых способов адресации.

Таблица 1.5

Инструкции, использующие прямую адресацию

Инструкция	**Описание**	**Пример**	**Короткий способ**	**Длинный способ**
LD	Загрузка	LD A,$55	Да	Да
CP	Сравнение	CP A,$55	Да	Да
BCP	Битовое сравнение	BCP A,$55	Да	Да
AND, OR, XOR	Логические операции	OR A, $55	Да	Да
ADC, ADD, SUB, SBC	Арифметические операции	ADC A, $55	Да	Да
CLR	Очистка	CLR A	Да	Нет
INC, DEC	Инкремент/декремент	INC addr	Да	Нет
TNZ	Проверка на <=0 (Test Negative or Zero)	TNZ A	Да	Нет
CPL, NEG	Перевод в дополнительный код	CPL label	Да	Нет
BSET, BRES	Установка/сброс битов	BRES ATCSR,#CMPIE	Да	Нет

Продолжение таблицы 1.5

Инструкция	Описание	Пример	Короткий способ	Длинный способ
BTJT, BTJF	Проверка битов и переходы	BTJT variable, #3, label	Да	Нет
SLL, SRL, SRA, RLC, RRC	Операции сдвига и вращения	SRL addr,#3	Да	Нет
SWAP	Обмен полубайтов	SWAP A	Да	Нет
CALL, JP	Вызовы подпрограмм, безусловные переходы	CALL subpr	Да	Нет

В примере, показанном на рис. 1.12, производится декремент данных, размещенных по адресу $40.

Рис. 1.12. Пример прямой адресации

1.4.5. Индексированная адресация

При индексированной адресации адрес операнда образуется сложением содержимого индексного регистра (X или Y) и 8- или 16-разрядного смещения, заданного во втором и третьем байте команды. Регистр X используется по умолчанию, для указания регистра Y используется префикс. Таким образом, команды, использующие регистр Y, длиннее на 1 байт, чем команды, использующие регистр X.

Так как оба индексных регистра являются 8-битными, то без смещения можно получить доступ только к диапазону адресов 00h-FFh. Возможные варианты задания смещения приведены в таблице 1.6, а перечень инструкций – в таблице 1.7.

Таблица 1.6

Варианты смещения для индексной адресации

Тип смещения	Длина в байтах	Адресуемое пространство	Примеры
без смещения (no offset)	0	00..FF	NEG (X)
короткое смещение (short offset)	1	00..1FE	SWAP (variable, X)
длинное смещение (long offset)	2	64 кБ	OR A, (variable, Y)

Таблица 1.7

Инструкции, использующие индексированную адресацию

Инструкция	Описание	Пример	Короткий способ	Длинный способ
LD	Загрузка	LD A,(txtoff,X)	Да	Да
CP	Сравнение	CP A, (Y)	Да	Да
BCP	Битовое сравнение	BCP A, (X)	Да	Да
AND, OR, XOR	Логические операции	AND A, (Y)	Да	Да
ADC, ADD, SUB, SBC	Арифметические операции	SUB A, (X)	Да	Да
CLR	Очистка	CLR (Y)	Да	Нет
INC, DEC	Инкремент/ декремент	DEC (X)	Да	Нет
TNZ	Проверка на <=0 (Test Negative or Zero)	TNZ (Y)	Да	Нет
CPL, NEG	Перевод в дополнительный код	CPL (Y)	Да	Нет
BSET, BRES	Установка/сброс битов	BSET (X), #3	Да	Нет
BTJT, BTJF	Проверка битов и переходы	BTJF (X), #3, label	Да	Нет
SLL, SRL, SRA, RLC, RRC	Операции сдвига и вращения	RLC (variable, Y)	Да	Нет
SWAP	Обмен полубайтов	SWAP (variable, Y)	Да	Нет
CALL, JP	Вызовы подпрограмм, безусловные переходы	JP (X)	Да	Нет

На рис. 1.13 показан пример индексированной адресации без смещения. В дополнительный код переводятся данные, находящиеся по адресу, который хранится в индексном регистре Y.

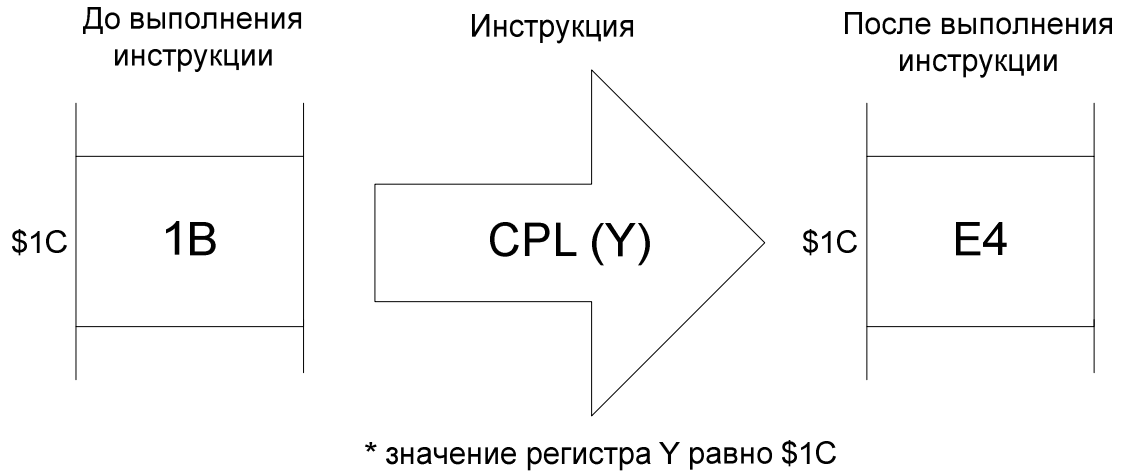


Рис. 1.13. Пример индексированной адресации без смещения

На рис. 1.14 показан пример длинного способа индексированной адресации со смещением. В аккумулятор загружаются данные, находящиеся по адресу $2AA5, вычисляющегося как сумма label=$2AA4 и значения индексного регистра X=$01.

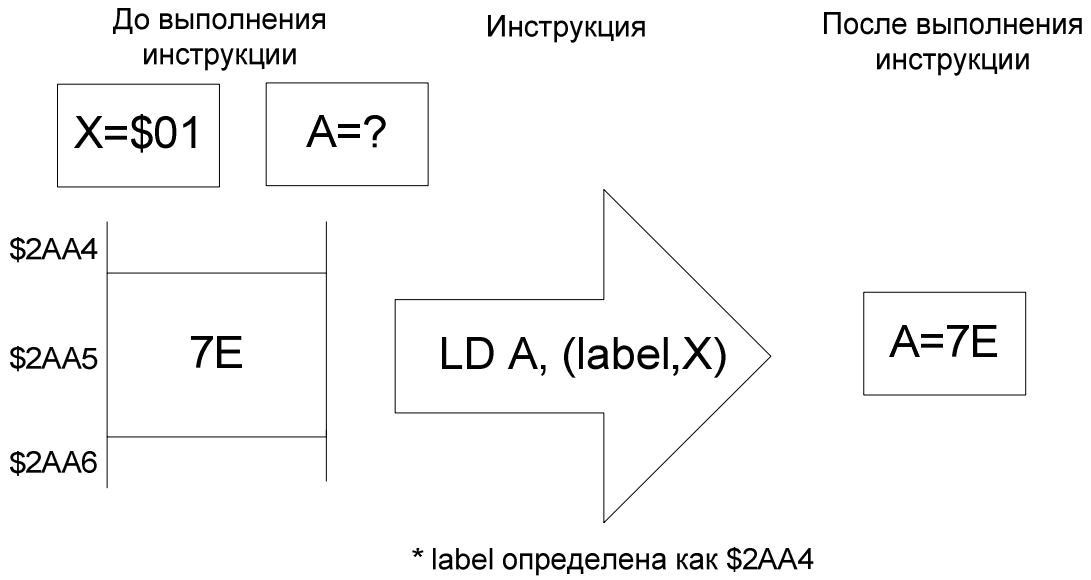


Рис. 1.14. Пример индексированной адресации со смещением

1.4.6. Косвенная адресация

Данные, используемые в операции, находятся по адресу, который хранится в памяти (указателю). Адрес указателя следует за кодом операции. Возможны короткий (указатель находится в пределах 00..FF) и длинный (указатель находится в любой ячейке из 64 кБ адресного пространства) способы адресации (табл. 1.8).

Таблица 1.8

Инструкции, позволяющие использовать косвенную адресацию

Инструкция	Описание	Пример	Короткий способ	Длинный способ
LD	Загрузка	LD A, [label.w]	Да	Да
CP	Сравнение	CP A, [label.w]	Да	Да
BCP	Битовое сравнение	BCP A, [label.w]	Да	Да
AND, OR, XOR	Логические операции	AND A, [label.w]	Да	Да
ADC, ADD, SUB, SBC	Арифметические операции	ADD A, [label.w]	Да	Да
CLR	Очистка	CLR [label]	Да	Нет
INC, DEC	Инкремент/ декремент	INC [variable]	Да	Нет
TNZ	Проверка на <=0 (Test Negative or Zero)	TNZ [variable]	Да	Нет
CPL, NEG	Перевод в дополнительный код	CPL [variable]	Да	Нет
BSET, BRES	Установка/ сброс битов	BRES [variable], #7	Да	Нет
BTJT, BTJF	Проверка битов и переходы	BTJT [variable], #3, label	Да	Нет
SLL, SRL, SRA, RLC, RRC	Операции сдвига и вращения	RLC [variable],#3	Да	Нет
SWAP	Обмен полубайтов	SWAP [label]	Да	Нет
CALL, JP	Вызовы подпрограмм, безусловные переходы	CALL [label]	Да	Нет

На рис. 1.15 косвенный режим адресации проиллюстрирован на примере команды SWAP. Данная команда меняет местами полубайты ячейки по адресу, указатель на который хранится в label=$3C.

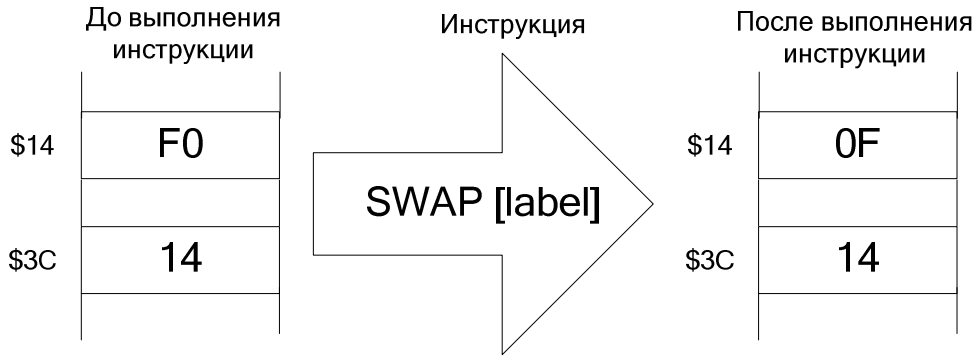


Рис. 1.15. Пример косвенной адресации

1.4.7. Относительная адресация

Относительная адресация используется только в командах ветвления. Адрес следующей команды образуется сложением текущего содержимого счетчика команд (PC) и заданного во втором байте команды 8-битного смещения. Смещение является знаковым числом и может задаваться прямым (указывается в явном виде сразу после кода операции) или непрямым (после кода операции указывается адрес, по которому хранится смещение) способами (табл. 1.9).

При относительной адресации возможен переход к командам, расположенных в пределах 127 позиций ниже или выше текущей команды программы.

Таблица 1.9

Инструкции, использующие относительную адресацию

Инструкция	Описание	Пример
JRxx	Условные переходы	JREQtimerb_exit
CALLR	Вызов подпрограммы	CALL lcd_reg_write

В таблице 1.10 сгруппированы все рассмотренные выше режимы адресации и их комбинации.

Таблица 1.10

Режимы адресации микроконтроллеров ST7

Режим адресации	Пример	Диапазон адресов	Размер указателя	Размер (в байтах)
Безадресный	NOP			+ 0
Непосредственный	LD A,#$55			+ 1

Продолжение таблицы 1.10

Режим адресации			Пример	Диапазон адресов	Размер указателя	Размер (в байтах)
Короткий	Прямой		LD A,$10	00..FF		+ 1
Длинный	Прямой		LD A,$1000	0000..FFFF		+ 2
Без смещ-ния	Прямой	Индекс.	LD A,(X)	00..FF		+ 0 (при X) + 1 (при Y)
Короткий	Прямой	Индекс.	LD A,($10,X)	00..1FE		+ 1
Длинный	Прямой	Индекс.	LD A,($1000,X)	0000..FFFF		+ 2
Короткий	Косвенный		LD A,[$10]	00..FF	Байт	+ 2
Длинный	Косвенный		LD A,[$10.W]	0000..FFFF	Слово	+ 2
Короткий	Косвенный	Индекс.	LD A,([$10],X)	00..1FE	Байт	+ 2
Длинный	Косвенный	Индекс.	LD	0000..FFFF	Слово	+ 2
Относит.	Прямой		JRNE LOOP	PC-128/ PC+127		+ 1
Относит.	Косвенный		JRNE [$10]	PC-128/ PC+127	Байт	+ 2
Битовый	Прямой		BSET $10,#7	00..FF		+ 1
Битовый	Косвенный		BSET [$10],#7	00..FF	Байт	+ 2
Битовый	Прямой	Относит.	BTJT $10,#7,LBL	00..FF		+ 2
Битовый	Косвенный	Относит.	BTJT [$10],#7,LBL	00..FF	Байт	+ 3

1.5. Система команд

Система команд микроконтроллеров ST7 включает в себя 63 инструкции длиной от 1 до 4 байт. Специальный префиксный байт, употребляемый в некоторых командах, служит для расширения стандартной 256-байтной сетки команд 8-разрядных микроконтроллеров.

Каждая команда состоит из четырех элементов (рис. 1.16):

– **метка** – идентифицирующая метка;

– **операция** – определяет выполняемое действие;

– **операнды** – параметры (числа, регистры), над которыми выполняется операция;

– **комментарий** – любой текстовый комментарий к строке.

Метка и комментарий являются необязательными элементами.

Рис. 1.16. Пример команды

Минимальное время исполнения 1-байтной команды составляет 250 нс при внутренней тактовой частоте 8 МГц.

В таблице 1.11 приведены все команды, сгруппированные по назначению. Подробное описание команд приведено в приложении 1.

Таблица 1.11

Система команд ST7

Загрузка	LD	CLR						
Работа со стеком	PUSH	POP	RSP					
Инкремент/ декремент	INC	DEC						
Сравнение	CP	TNZ	BCP					
Логические операции	AND	OR	XOR	CPL	NEG			
Битовые операции	BSET	BRES						
Проверка битов, ветвление	BTJT	BTJF						
Арифметические операции	ADC	ADD	SUB	SBC	MUL			
Сдвиги и вращения	SLL	SRL	SRA	RLC	RRC	SWAP	SLA	
Безусловный переход, вызов подпрограммы	JRA	JRT	JRF	JP	CALL	CALLR	NOP	RET
Условные переходы	JRxx							
Управление прерываниями	TRAP	WFI	HALT	IRET				
Изменение флагов регистра кода условия	SIM	RIM	SCF	RCF				

В таблице 1.12 показано, сколько различных режимов адресации могут иметь определенные группы команд.

Таблица 1.12

Количество возможных режимов адресации для команд

	Без операндов (системные)	**Умножение и ветвление**	**PUSH и POP**	**Загрузка из памяти с помощью индексных регистров**	**Загрузка и сравнение с индексными регистрами**	**Арифметические операции и операции проверки с одним операндом**	**Загрузка из памяти в аккумулятор, длинные переходы**	**Арифметические с двумя операндами**
Примеры команд	HALT	MUL	POP	LD mem, X	CP X,		CALL	ADC
	IRET	BRES	PUSH	LD mem, Y	LD X,		JP	ADD
	NOP	BSET			CP Y,		LD mem,A	AND
	RCF	BTJF			LD Y,	NEG		BCP
	RET	BTJT						CP A,
	RIM	CALLR						LD A,
	RSP	JR*				MUL		OR
	SCF					RRC		SBC
	SIM					CALL		SUB
	TRAP							XOR
	WFI							
Кол-во возможных режимов адресации	1 (безадресный режим)	2	4	9	10	11	14	15

1.6. Периферийные устройства

В группу периферийных устройств входят (рис. 1.17):
– параллельные порты ввода-вывода (порты A, B, C);
– последовательный порт SPI;
– интерфейсы SCI, CAN, I^2C, USB;
– 8- и 16-битные таймеры общего назначения;
– сторожевой таймер;
– 10-битный аналого-цифровой преобразователь (АЦП);
– блок прерываний.

Набор периферийных устройств зависит от модели микроконтроллера.

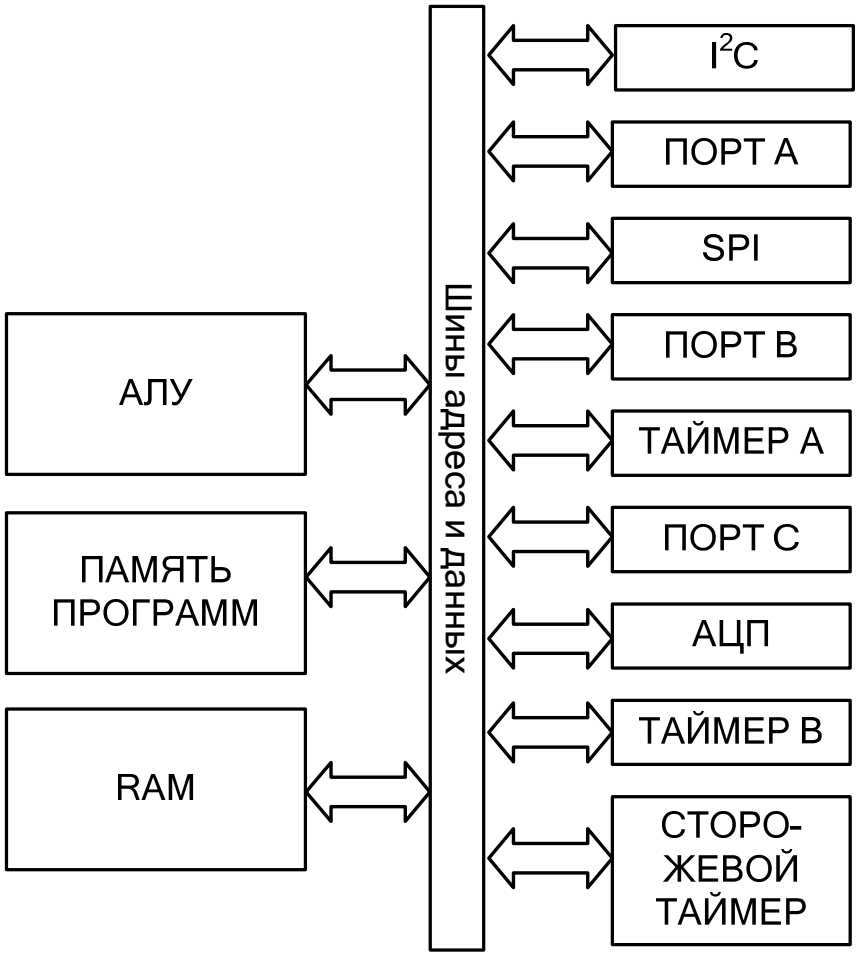


Рис. 1.17. Периферийные устройства

1.6.1. Параллельные порты ввода-вывода

Обмен данными между микроконтроллером и внешними устройствами производится с помощью портов ввода/вывода. Микроконтроллеры семейства ST7 имеют конфигурируемые порты ввода/вывода, каждый порт может быть настроен для использования либо в качестве входа (с нагрузочным резистором или без него), либо в качестве выхода (с открытым стоком либо двухтактного).

Настройка осуществляется с помощью регистров, связанных с каждым портом. В ST7 используются регистры OR (Option Register) и DDR (Data Direction Registers).

Кроме того, есть возможность использования портов в качестве входов/выходов других периферийных устройств, таких как таймеры, SPI, АЦП и др. Возможность конфигурирования портов позволяет снизить количество компонентов на принципиальной схеме и, следовательно, уменьшить размер монтажной платы.

1.6.2. Последовательный интерфейс SCI

Данный интерфейс обеспечивает стандартный асинхронный формат приема/передачи данных с одним стартовым и одним стоповым битом и длиной информационного слова в 8 или 9 бит. Поддерживается скорость от 300 до 115200 бод.

SCI – дуплексный, UART-типа. Это асинхронная система со стандартным форматом «без возврата к нулю» (NRZ) для переданного или полученного бита. Длина переданного слова 10 - 11 бит (1 старт-бит, 8 - 9 информационных разрядов, 1 стоп-бит). SCI состоит из трех модулей: приемник, передатчик и контроллер скорости пересылки данных в бодах.

1.6.3. Последовательный порт SPI

Интерфейс порта SPI (Serial Peripheral Interface) предназначен для высокоскоростного обмена между микроконтроллером ST7 и периферийными микросхемами, такими как АЦП и ЦАП, FLASH-память большой информационной емкости, часы реального времени. Данный интерфейс может быть также использован для обмена между двумя микроконтроллерами, расположенными на небольшом расстоянии. Достоинством синхронной последовательной приемо-передачи с использованием протокола SPI является полный дуплексный обмен данными, что в случае необходимости позволяет реализовать экономичную потенциальную развязку между приемником и передатчиком.

Интерфейс основан на сдвиговом регистре, который предназначен для выполнения преобразования последовательного интерфейса в параллельный, а также для обратного преобразования. Интерфейс трехпроводный, состоит из двух каналов данных и одного канала синхронизации (рис. 1.18).

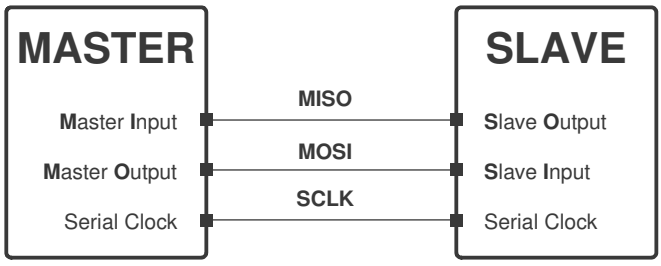


Рис. 1.18. Интерфейс порта SPI

На рис. 1.19 показано подключение к MASTER нескольких SLAVE. Выбор SLAVE, с которым MASTER будет осуществлять обмен, выполняется с помощью установки входа $\overline{SS}$.

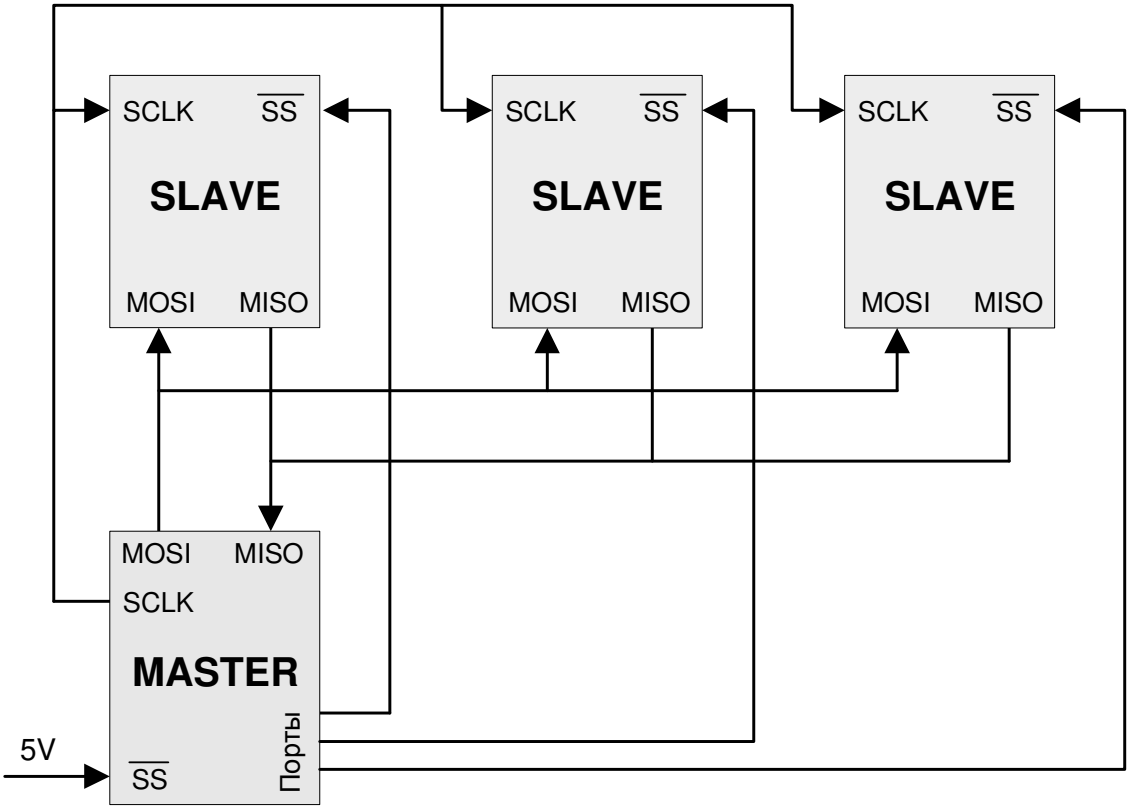


Рис. 1.19. Конфигурация с несколькими SLAVE

Настройка интерфейса SPI осуществляется с помощью регистра SPICR (SPI Control Register), описание битов данного регистра приведено ниже в табл. 1.13.

Таблица 1.13

Регистр SPICR

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
SPIE	SPE	SPR2	MSTR	CPOL	CPHA	SPR1	SPR0

Предназначение битов регистра SPICR:

SPIE – разрешение прерываний (0 – прерывания запрещены);

SPE – разрешение выхода (1 – SPI подключен к внешнему устройству);

MSTR – признак MASTER или SLAVE (1 – MASTER);

CPOL – полярность тактового генератора;

CPHA – используемый фронт тактового генератора;

SPR0, SPR1, SPR2 – выбор рабочей частоты (табл. 1.14).

Таблица 1.14

Выбор рабочей частоты

Частота	SPR2	SPR1	SPR0
F_{CPU} / 4	1	0	0
F_{CPU} / 8	0	0	0
F_{CPU} / 16	0	0	1
F_{CPU} / 32	1	1	0
F_{CPU} / 64	0	1	0
F_{CPU} / 128	0	1	1

Для просмотра статуса интерфейса SPI предназначен 8-битный регистр SPISR (SPI Status Register), описание битов данного регистра приведено ниже.

Таблица 1.15

Регистр SPISR

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
SPIF	WCOL	OVR	MODF	-	SOD	-	-

Предназначение битов регистра SPISR:
SPIF – флаг передачи данных (1 == передача данных завершена);
WCOL – статус коллизий записи;
SOD – запрещение выхода SPI (выход SPI разрешен, если SPE=1);
MODF – флаг ошибки режима;
OVR – ошибка переполнения SPI.
Биты SPIF, WCOL, MODF, OVR доступны только для чтения.
Пример инициализации порта SPI показан на рис. 1.20. В аккумулятор загружается константа, содержащая требуемые настройки порта, затем содержимое аккумулятора записывается в регистр управления портом SPICR.

```
ld A, #$5c  ; загрузить в аккумулятор значение $5C=%01011100
            ; настройки порта SPI:
            ; SPR0=SPR1=SPR2=0 (fcpu/8)
            ; CPHA=1, CPOL=1
            ; MSTR=1 (режим MASTER)
            ; SPE=1 (SPI подключен к внешнему устройству)
ld SPICR, A ; загрузить аккумулятор в регистр управления SPI
```

Рис. 1.20. Пример инициализации порта SPI

1.6.4. АЦП

Аналого-цифровые преобразователи (АЦП) обеспечивают ввод двоичных значений потенциалов, поступающих на аналоговые входы. Основные характеристики АЦП микроконтроллеров семейства ST7 следующие:

– входные значения – положительные, отрицательные напряжения не конвертируются;

– разрешение – 10 бит;

– время преобразования – 64 цикла АЦП;

– точность – 1 LSB (до 0.12%);

– включение/выключение АЦП для снижения потребляемой мощности.

Настройка параметров преобразования осуществляется с помощью регистра ADCCSR (табл. 1.16).

Таблица 1.16

Регистр ADCCSR

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
EOC	SPEED	ADON	-	-	CH2	CH1	CH0

Предназначение битов регистра ADCCSR следующее:

EOC – завершение преобразования (1 – данные преобразованы и доступны в регистрах ADCDRL и ADCDRH);

SPEED – задание частоты преобразования (1 – частота ядра, 0 – частота ядра/2);

ADON – включение АЦП (1 – АЦП включен);

CH0, CH1, CH2 – определяют контакты, которые будут прочитаны (табл. 1.17).

Таблица 1.17

Выбор входных контактов

Контакты	CH2	CH1	CH0
AIN0/PB0	0	0	0
AIN1/PB1	0	0	1
AIN2/PB2	0	1	0
AIN3/PB3	0	1	1
AIN4/PB4	1	0	0
AIN5/PB5	1	0	1
AIN6/PB6	1	1	0

Для хранения 10-битных результатов преобразования используются два 8-битных регистра – ADCDRL и ADCDRH.

Бит ADON включает/выключает АЦП. В выключенном состоянии АЦП не потребляет питания. После включения в течение первых 30 мкс результаты преобразования могут быть достаточно неточными.

Любая операция записи в регистр ADCCSR останавливает текущую операцию преобразования и запускает новую. После завершения преобразования устанавливается бит EOC и данные становятся доступны в регистрах ADCDRL и ADCDRH.

1.6.5. 16-разрядный таймер

В микроконтроллерах таймер является одним из важнейших периферийных устройств, поскольку при выполнении задач управления достаточно часто возникает необходимость приема и выдачи управляющих сигналов в заданные моменты времени. Микроконтроллеры семейства ST7 содержат 16-разрядный таймер, который может эффективно использоваться для указанных функций.

Источником тактовых импульсов для таймера может быть либо внутренний тактовый генератор, либо внешний генератор, подключенный к одному из портов. В случае использования внутреннего источника тактовых импульсов на таймер подается частота ядра, которая может быть разделена делителем частоты на 2, 4 или 8. Внешний источник занимает один порт ввод/вывода. Возможность подключения внешнего источника присутствует, например, в микроконтроллерах ST72251 and ST72311.

Основным компонентом 16-разрядного таймера является таймер свободного хода, занимающий регистры CHR и CLR. Это двоичный счетчик, который увеличивается на каждом такте. Значения регистров CHR и CLR доступны по чтению, также есть возможность сброса значения счетчика в начальное значение FFFCh.

Каждый раз при переполнении (переходе с FFFFh в 0000h) устанавливается бит TOF регистра статуса TSR. При переполнении может быть сформирован запрос на прерывание.

Кроме режима свободного хода существуют режим захвата и режим сравнения. В режиме сравнения значение таймера проверяется на соответствие некой заданной величине. В режиме захвата производится фиксирование значения таймера при возникновении определенного события – изменении сигнала на определенном входе микроконтроллера.

1.6.6. Сторожевой таймер

Сторожевой таймер представляет собой дополнительный таймер, используемый для защиты системы от ошибок, вызванных зацикливанием программы.

Управление сторожевым таймером ST7 осуществляется с помощью специального регистра WDGCR, включающего два бита управления (биты 6 и 7) и 6 времязадающих битов (рис. 1.21).

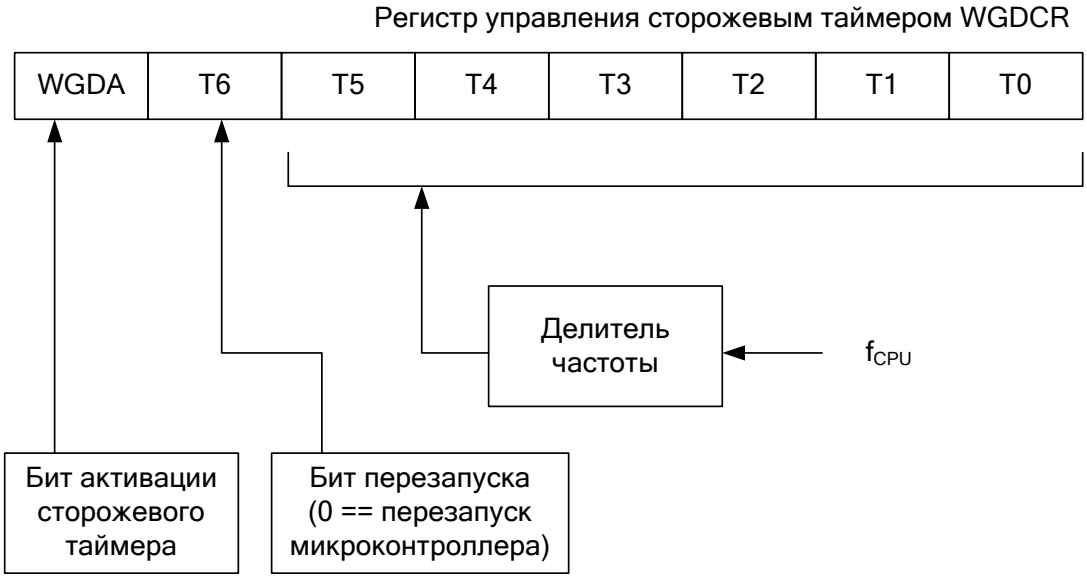


Рис. 1.21. Регистр управления сторожевым таймером

Установка бита 7 (WDGA) запускает сторожевой таймер. В большинстве микроконтроллеров семейства ST7 данный бит изначально аппаратно устанавливается в 1 и не подлежит изменению, но в некоторых моделях (например, ST72251) допускается программная установка данного бита.

Сброс бита 6 в 0 приводит к перезапуску микроконтроллера.

1.6.7. Блок обработки прерываний

Запросы на прерывания могут генерироваться несколькими внутренними (таймеры, порт SPI) или внешними источниками (порты ввода/вывода). Кроме того, в микроконтроллерах семейства ST7 поддерживаются программные прерывания (инструкция TRAP). На рис. 1.22 показаны различные возможные источники прерываний.

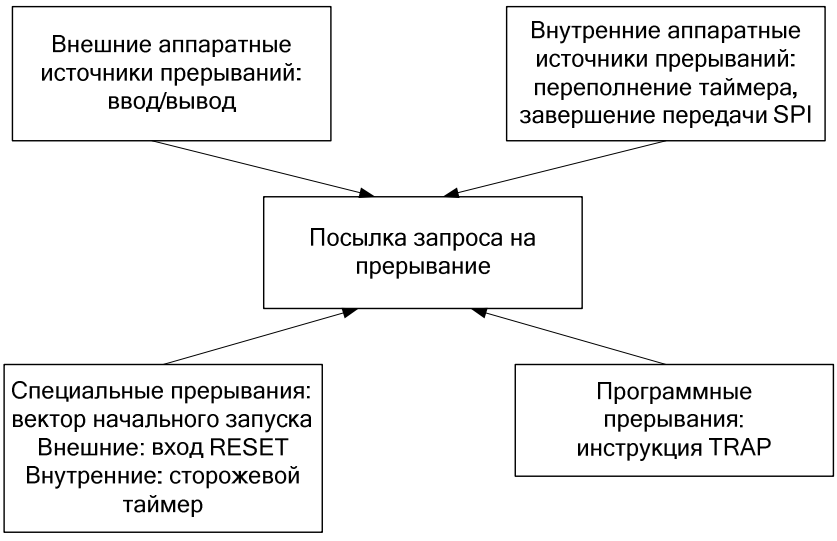


Рис. 1.22. Источники прерываний

Механизм обработки прерываний (рис. 1.23) состоит из следующих этапов:

– сохранение текущего контекста (регистры CC, A, X , PC) в стеке;

– отключение всех других прерываний путем установки бита I в регистре кода условия;

– загрузка вектора прерывания в счетчик команд;

– обработка прерывания;

– восстановление контекста;

– разрешение прерываний (сброс бита I).

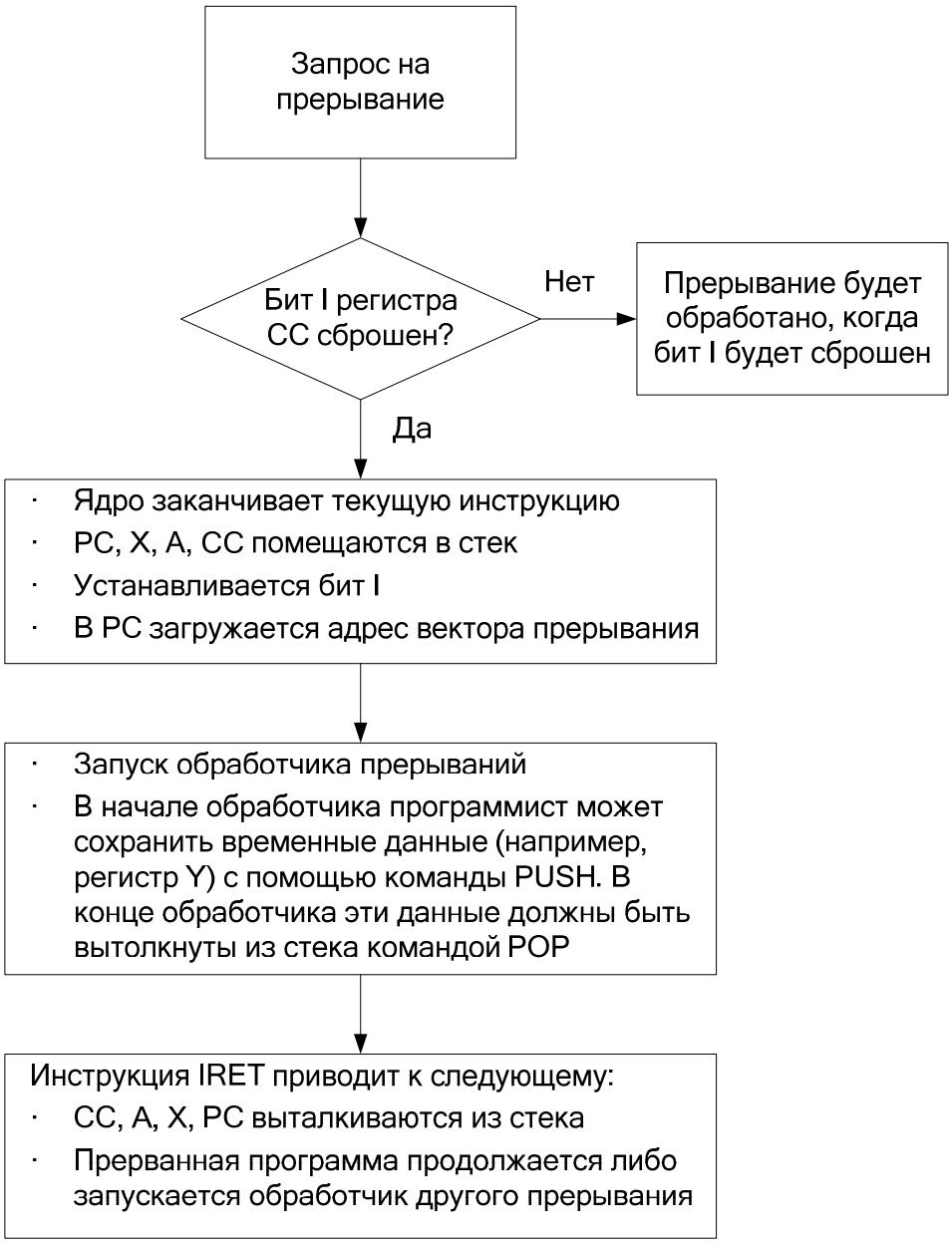


Рис. 1.23. Механизм обработки прерываний семейства ST7

Приоритет между прерываниями распределяется согласно вектору прерывания (рис. 1.24).

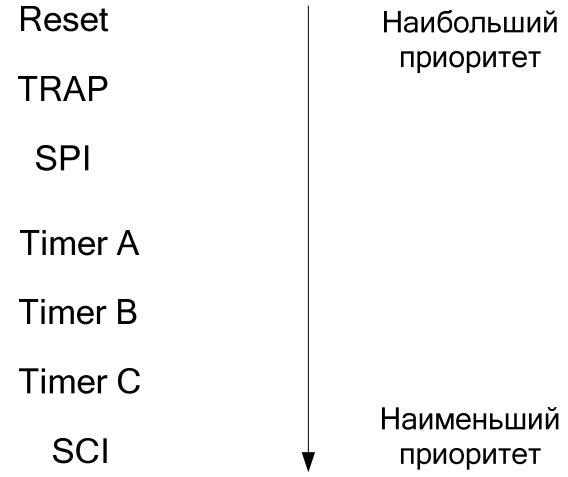


Рис. 1.24. Приоритет прерываний семейства ST7

1.6.8. Поддерживаемые интерфейсы

Микроконтроллеры семейства ST7 имеют большой набор последовательных интерфейсов – SPI, SCI, I^2C, CAN и USB. Следует отметить, что STM одной из первых предложила микроконтроллеры со встроенной поддержкой USB; при этом номенклатура таких микроконтроллеров постоянно расширяется. Так, например, ST7261, ST7262 и ST7263 – это микроконтроллеры для низкоскоростных USB-устройств, применяемые там, где требуется обмен небольшими объемами данных. ST7265 поддерживает высокоскоростной режим, что позволяет передавать большие объемы данных (цифровые изображения, звук). Такие микроконтроллеры применяются, например, в MP3-плеерах, устройствах записи/чтения Flash-карт и т.п.

В таблице 1.18 отмечены интерфейсы, поддерживаемые различными семействами микроконтроллеров ST7.

Таблица 1.18

Интерфейсы, поддерживаемые микроконтроллерами ST7

Модель	SPI	SCI	I^2C	CAN	USB
ST7FOXU0	•		•		
ST7FOXKx	•		•		
ST7FOXF1					
ST7LITEUxx					
ST7LITESxxx	•				
ST7LITE0xxx	•				
ST7LIT1xxxx	•				
ST7LITE2xxx	•				
ST7LITE3xxx	•				
ST7LITE49M			•		
ST72260Gx	•				
ST72262Gx	•				

Продолжение таблицы 1.18

Модель	SPI	SCI	I^2C	CAN	USB
ST72264Gx	•	•	•		
ST7232xKx	•	•			
ST7256xxx	•	•		•	
ST7GEME4					•
ST7SCRxxx					•
ST726xxx					•

1.7. Области применения

Микроконтроллеры ST7 имеют наиболее низкую стоимость и менее широкие функциональные возможности по сравнению с другими семействами STM. Большинство микроконтроллеров семейства ST7 ориентировано на использование в относительно несложных устройствах массового применения, однако в линейке присутствуют модели, позволяющие решать достаточно серьезные задачи. В целом номенклатура обеспечивает потребности многих отраслей промышленности в надежных, высокоэффективных и недорогих устройствах управления и контроля.

Как было отмечено выше, микроконтроллеры семейства ST7 имеют некоторое сходство с MC68HC05/08 фирмы MOTOROLA и, следовательно, могут применяться в тех же областях. Это средства беспроводной связи, телефония, локальные системы сбора информации и управления.

На мировом рынке микроконтроллеры STM занимают свою нишу в бытовой и автомобильной электронике, промышленной автоматике, устройствах сбора и предварительной обработки данных и аппаратуре связи. Примеры применения конкретных моделей микроконтроллеров приведены в табл. 1.19.

Таблица 1.19

Применение микроконтроллеров ST7

Область применения	Элемент	Примеры микроконтроллеров
Автомобили	Корпус	ST72254, ST72334, ST72511, ST72521
	Радио	ST72311
	Насосы топлива/воды	ST72141, ST72334
	Приборы	ST72389, ST72589
	Схема безопасности	ST72314, ST72215
	Вентилятор	ST72314, ST72141
	Датчики	ST7255, ST72254, ST72334

Продолжение таблицы 1.19

Область применения	Элемент	Примеры микроконтроллеров
Потребители	Цифровые проигрыватели	ST72T311, ST72254, ST72314, ST72F65
	АТС	ST72254
	Панель настроек	ST72254
Промышленность	Приборы	ST72314, ST72334, ST72311R,
	Управление бесщеточным двигателем	ST72141
	Однофазные двигатели	ST72334, ST72311R, ST72254, ST72215
	Измерения	ST72C171
	Телеметрия	ST72321R
	Промышленное управление	ST72141
	Датчики дыма	ST72334, ST72254
	Терморегуляторы	ST72334, ST72311R, ST7215, ST72216, ST72104
	Контроль температуры	ST72334, ST72311R, ST72254
	Датчики	ST7FOX
	Кондиционеры	ST7FOX
Считывающее устройство	SMARTCARD	ST72411, ST7262
Компьютер	Управление батареей (PC,GSM...)	ST72215, ST72311J
	USB	ST7261/62/63
	Игры	ST7263, ST7262
	Источники бесперебойного питания (ИБП)	ST7263, ST72215, ST72311J

Микроконтроллеры ST7 используются компаниями Logitech, Microsoft, Netac, HP, Creative при разработке различной компьютерной периферии (клавиатуры, манипуляторы «мышь», игровые джойстики, принтеры/факсы и т.п.). Компании APC, Tripplite, Cyberpower, Delta используют ST7 в блоках бесперебойного питания (UPS), блоках питания. В бытовой технике эти микроконтроллеры можно встретить у производителей Whirlpool, Invensys, Electrolux, Moulinex и др. В промышленности ST7 активно используют компании Schneider, Rockwell, Agilent и Tyco.

Примеры систем, разработанных с использованием микроконтроллеров семейства ST7, приведены в разделах 3 и 4.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Укажите состав семейств восьмиразрядных микроконтроллеров STM и их ключевые особенности.
2. Назовите элементы и порядок взаимодействия общей архитектуры микроконтроллеров семейства ST7.
3. Перечислите внутренние регистры и назовите их предназначение.
4. Каковы принципы организации памяти?
5. Дайте характеристику режимов работы и способов адресации.
6. Каковы особенности работы со стеком?
7. Перечислите стандартные периферийные устройства.
8. Перечислите основные операции работы с последовательным портом SPI.
9. Проанализируйте особенности системы прерываний.
10. Укажите области применения микроконтроллеров семейства ST7.

РАЗДЕЛ 2. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Микроконтроллер (МК) – универсальный программируемый компонент системы, основным узлом которого является программа управления. В зависимости от встроенного программного кода, разработчик имеет возможность задать необходимые функциональные характеристики системы и легко внести в них коррективы в будущем. Зачастую одно и тоже семейство микроконтроллеров может выполнять различные по сложности задачи.

Проектные решения, на базе МК, для программиста представлены определенным образом, он оперирует только с цифровыми данными, а не физическими величинами: порты ввода/вывода получают или выдают данные в бинарном виде.

Таким образом, основная задача разработчика – это корректная настройка управляющих регистров портов ввода/вывода, различных интерфейсов, периферийных устройств, а также внутренняя обработка получаемой/передаваемой информации.

Постоянное развитие новых информационных и компьютерных технологий позволило проектировщикам представлять управляющий код на различных уровнях абстракции – языках программирования (ассемблер, C, и т.д.).

Процесс написания управляющих программ МК является достаточно трудоемким и занимает длительный отрезок времени из всего этапа построения цифровой системы, поэтому наличие качественной среды разработки и средств встроенной отладки позволяет максимально упростить и ускорить написание программ.

В данном разделе рассматриваются и обсуждаются проблемы, с которыми сталкивается программист. Они касаются, во-первых, выбора соответствующего языка описания управляющей программы (ассемблер, C) для получения необходимой производительности (скорость выполнения программы на целевой системе, время ее написания, оптимальное использование внутренних ресурсов кристалла и т.д.). Во-вторых, выбора необходимой среды разработки (и работу в ней), которая не только позволит имплементировать программный код в МК, но и проводить его отладку в реальном масштабе времени.

2.1. Программирование на языке ассемблера

2.1.1. Когда надо использовать язык ассемблера

Язык ассемблера является родным языком для каждого микропроцессора. Это был единственный язык программирования для небольших микроконтроллеров, пока не появились компиляторы языков высокого уровня для различных семейств (с различной архитектурной реализацией).

Процесс написания программы на языке ассемблера требует особой осторожности и внимания, а также большого количества строк

программного кода для выполнения относительно маленькой несложной задачи. Однако язык ассемблера был и есть достаточно востребованным, когда речь идет об оптимизации используемых ресурсов (скорость выполнения программы, экономия памяти данных, памяти программ и т.д.), либо когда алгоритм является достаточно простым. В остальных случаях рекомендуется применение языков высокого уровня, таких как С. Помимо этого при написании программы разработчик получает возможность использовать «смешанный» стиль описания, когда в исходный код на языке высокого уровня вставляются ассемблерные вставки.

Приведем в качестве примера ряд задач (ту часть кода), где ассемблер (ассемблерные вставки) является распространенным решением.

1. Инициализационная часть программы. Все языки высокого уровня позволяют проводить настройку ядра и памяти МК. Однако, базовая организация памяти (адрес ROM и RAM, векторов прерывания, сброса и т.д.) подаются в качестве ассемблерных вставок (шаблонов-инициализаторов), которые должны в обязательном порядке быть включены в головной проект.
2. Некоторые подпрограммы обработки прерываний, когда использование ассемблера полезно для оптимизации (минимизации) их времени выполнения.
3. Некоторые часто используемые функции (подпрограммы), к которым происходит периодическое обращение. В этом случае может быть обеспечена оптимизация скорости выполнения программного кода.

2.1.2. Процесс написания программного кода на языке ассемблера

Разработка программы, в общем случае, состоит из трех основных этапов: анализ, написание кода (кодирование), отладка.

На первом этапе программист должен понять, что программа должна делать. Особенность данного этапа заключается в том, что в большей степени это бумажная работа (представление программы на уровне блочной структуры, составление алгоритмов, определение параметров и типов входных/выходных данных, приблизительный анализ необходимого объема памяти и т.д.).

На втором этапе происходит процесс трансляции результатов первого этапа, результатом которой является исходный код и несколько служебных (проектных) файлов, управляющих утилитами программирования.

Третий этап состоит из операций, необходимых для корректной трансляции исходного кода – удаления программных ошибок, которые могут появиться на предыдущих этапах разработки (логические ошибки, управления данными и т.д.). Данный этап является наиболее тру-

доемким и занимает наибольшую часть времени из всех вышеперечисленных.

Когда программа функционально закончена и отлажена, она сохраняется в EPROM, которая может быть как внешней, так и внутренней (ST7) по отношению к МК.

2.1.3. Язык ассемблера

Язык ассемблера – это набор мнемоник, которые просто дублируют каждую инструкцию ядра МК в более читабельной форме. Данный набор формирует определенный алгоритм работы всей системы (операции над данными, настройка периферийных узлов и т.д.).

```
    ld X,#255     ; загрузим в регистр X значение 255
decr1:
    dec X         ; уменьшим на 1 значение регистра X,
     jrne decr1   ; пока X не равен 0
```

В данном программном коде происходит загрузка числа 255 в регистр **X** с последующим его декрементом.

Следует отметить, что язык ассемблера не является стандартизированным по нескольким причинам:

– набор инструкций разных процессоров (микроконтроллеров) отличается;

– для каждого процессора или микроконтроллера, программные средства разработки и написания кода существенно отличаются по синтаксису.

Общим, при написании кода на ассемблере, является изучение синтаксиса и набора инструкций конкретного семейства микропроцессорного ядра, а также среды разработки, в которой будет реализован алгоритм.

2.1.3.1. Ассемблер

Следует также различать понятие «ассемблер» и «язык ассемблера», так как под ассемблером понимают не только программный код, но и набор утилит и средств для интерпретации программы в машинный код, с последующей имплементацией его в микроконтроллер.

Выходным результатом программного кода на языке ассемблера является (рис. 2.1):

– объектный файл, содержащий двоичный код (*.obj);

– листинг файл – файл отчета, который включает в себя исходный код на ассемблере, а также его цифровую интерпретацию (*.lst);

– другие файлы, количество и тип которых зависит от конкретного типа ассемблера: файлы отчетов об ошибках (*.err), файлы символьных таблиц (*.sym), типах переменных и т.д.

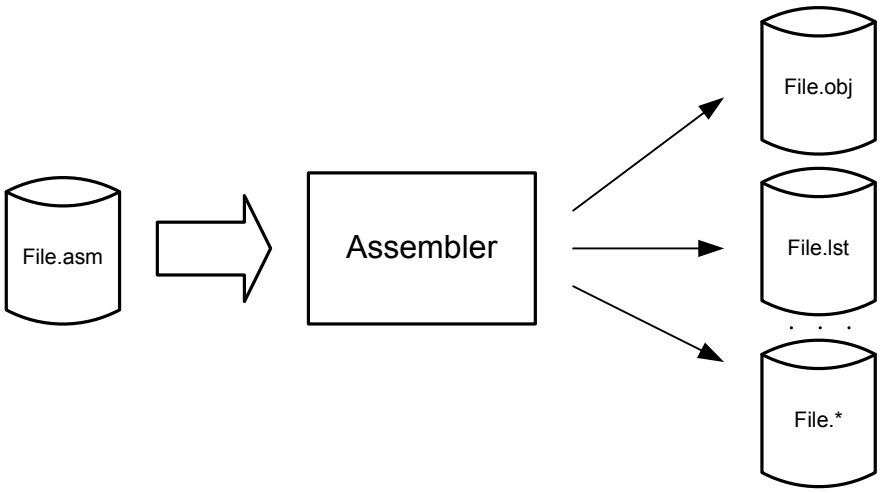


Рис. 2.1. Выходные файлы проекта на языке ассемблер

2.1.3.2. Компоновщик (Linker)

При незначительном размере программного кода (один файл), ассемблер интерпретирует его в объектный файл (*.obj), который содержит все необходимые машинные инструкции, готовые к использованию. Данная процедура называется абсолютным ассемблированием (работа с абсолютными адресами).

В связи с ростом сложности приложений, выполняемых на микроконтроллерах во встроенных системах, суммарное количество строк одного проекта может быть слишком велико и размещение их всех в одном файле не представляется возможным по ряду причин.

Во-первых, снижается удобочитаемость всего программного кода, во-вторых, при минимальном внесении корректив в данный проект, приходилось бы перекомпилировать весь файл, размер которого чрезвычайно огромен и соответственно время его трансляции выросло бы в несколько раз.

Однако в настоящее время большинство приложений строится из множества файлов проекта (библиотеки и т.д.), каждый из которых содержит сотни и тысячи строк ассемблерного кода. Для организации механизма разработки иерархических проектов ассемблер ST7 фирмы STMicroelectronics использует ряд дополнительных ключевых слов и инструкций EXTERN и PUBLIC.

Ключевое слово EXTERN для компоновщика означает, что конкретная метка (функция, переменная) объявлена во внешнем файле. Ключевое слово PUBLIC означает, что данная переменная (функцию) может использоваться в других внешних файлах проекта.

Общая структура проекта и схема взаимодействия между файлами представлена на рисунке 2.2 и 2.3 соответственно.

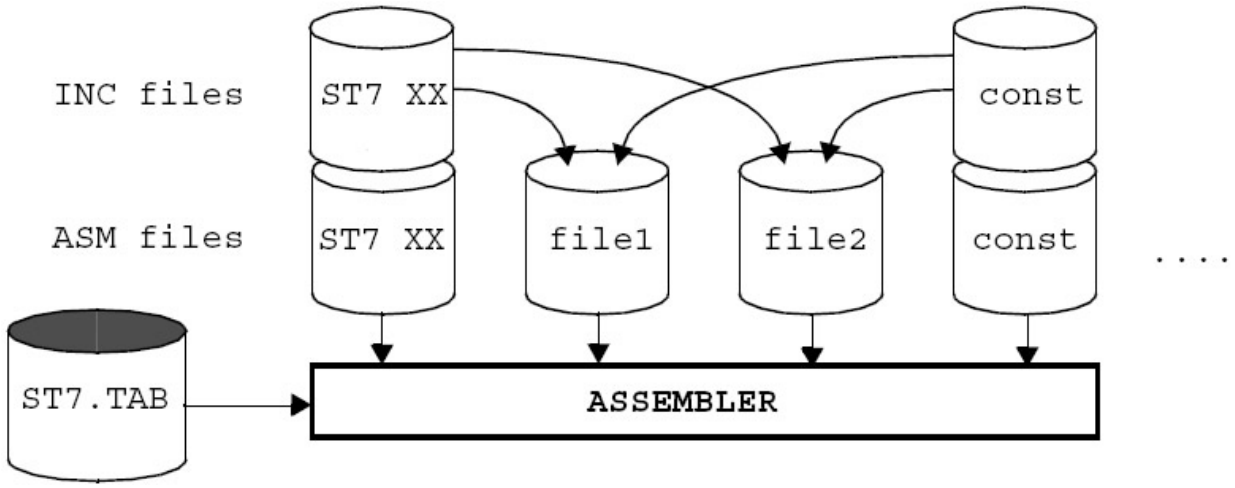


Рис. 2.2. Общая структура проекта – ассемблер ST7

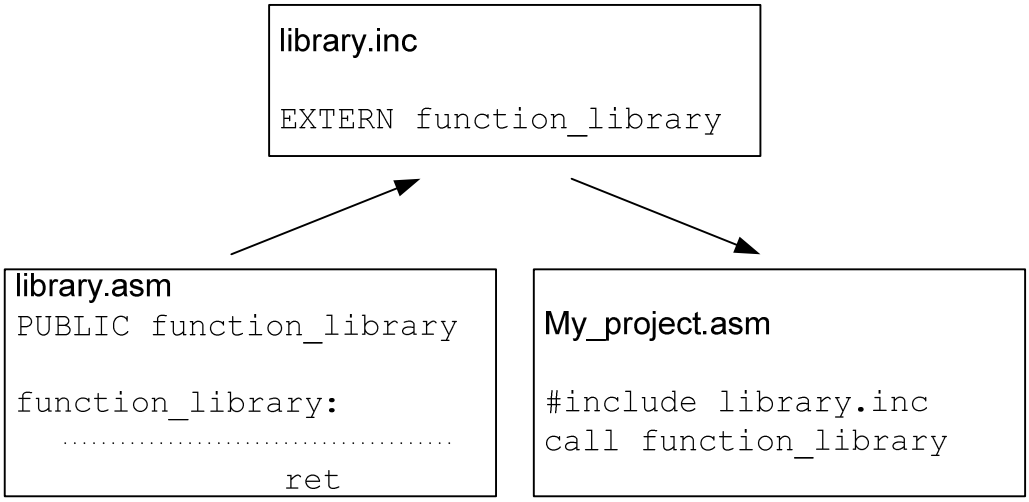


Рис. 2.3. Схема взаимодействия между файлами проекта

В основной файл проекта My_project.asm (рис. 2.3) подключается заголовочный файл библиотеки (library.inc), которые содержит прототип функции (`function_library`). Непосредственная реализация данной функции (подпрограммы) выполняется в исходном файле library.asm.

Таким образом, обязательным атрибутом любого ассемблера является наличие компоновщика (linker), в задачи которого входит:

– объединение объектных фалов (*.obj) в один;

– корректировка значений адресов во всех операндах команд, которые ссылаются на объект, чье расположение в памяти установлено либо изменено в процессе конкатенации.

Для выполнения подобных действий необходим контрольный файл, в котором бы оговаривались правила последовательности объединения объектов (*.obj), их перечень, а также абсолютные адреса, к которым будет произведена привязка при занесении их в память микроконтроллера.

Как только программа скомпонована, на выходе получаем абсолютный объектный файл (*.abs или *.cod), который может использоваться как для отладки в эмуляторе, так и для прошивки EPROM микроконтроллера.

Результатом работы компоновщика являются (рис. 2.4):

– абсолютный объектный файл (*.cod или *.abs);
– файл таблицы имен (*.sym);
– файл распределения (*.map).

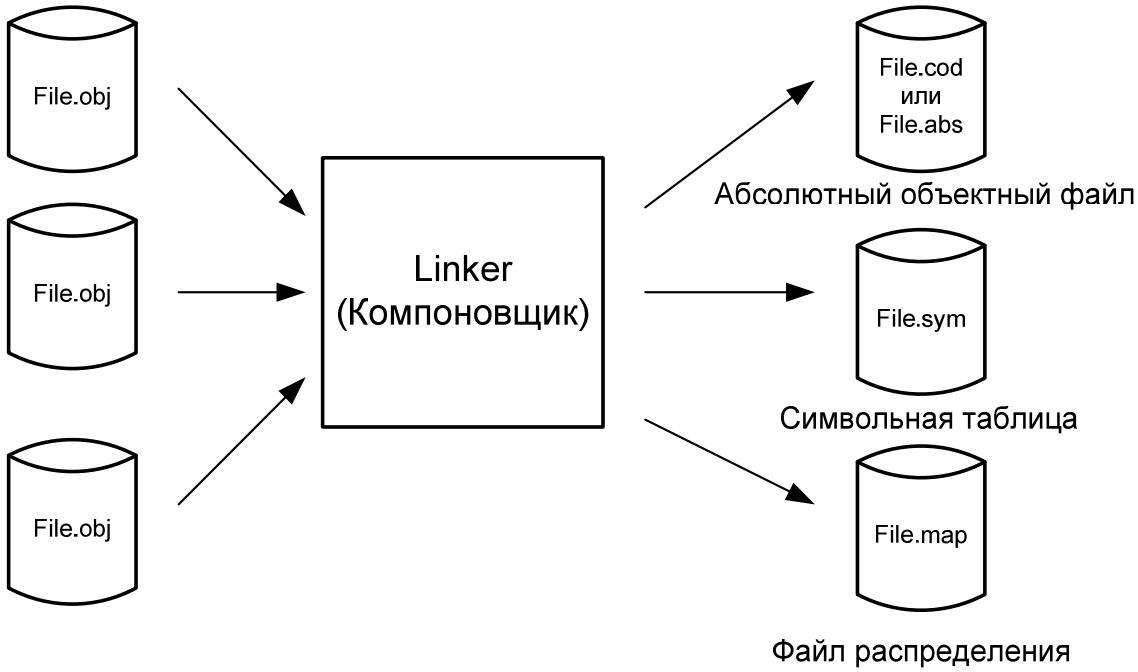


Рис. 2.4. Выходной результаты работы компоновщика

Файл таблицы имен (*.sym) содержит используемые в программе идентификаторы (метки, имена подпрограмм и переменных) и их атрибуты (признаки); таблица имен создается во время работы транслятором и компоновщиком; используется на стадиях семантического анализа и генерации промежуточного кода, а также для компоновки программы. Она может быть сохранена соответственно в объектном или загрузочном модуле для целей отладки.

Файл распределения (*.map) содержит информацию о компонуемых файлах и внешних (public) символах.

2.1.4. Ассемблер ST7

ST7 Ассемблер – кросс-ассемблер (формирует машинный код для процессорного ядра другого типа, чем тот, на котором он исполняется). Обычно такие средства используется для бортовых и встраиваемых компьютеров, а также для микроконтроллеров.

2.1.4.1. Исходные файлы проекта

Программный код, написанный на языке ассемблера ST7 и сохраненный в текстовом ASCII формате, называется исходным файлом проекта. Исходный файл проекта имеет расширение *.asm.

2.1.4.2. Формат файла проекта на языке ассемблера

Как и для любого ассемблера, исходный файл проекта для микроконтроллеров ST7 имеет строгий формат и синтаксис (см. приложение 4).

Первая строка данного файла зарезервирована для указания *.tab файла, т.е. целевого процессора, под который пишется программа (в нашем случае это st7.tab). Допускается также не указывать *.tab файл, а просто прописать полный путь к нему, т.к. ассемблер ищет его по умолчанию:

```
c:\program files\st7tools\asm\st7\
```

При отсутствии указанной строки поиск производится в текущей директории проекта. Если файл не найден, будет выдана ошибка и трансляция программы прекратится.

Исходный файл проекта на языке ассемблере ST7 обязательно должен заканчиваться ключевым словом `END` (приложение 4), в противном случае компилятор выдаст ошибку.

Остальной формат исходного кода имеет следующий вид:

```
[метка[:]]<space>[КОП]<space>[операнд]<space>[;комментарий]
```

где `<space>` – это символ клавиши «SPACE» ($20) или «TAB» ($09);

КОП – код операции.

Все четыре поля могут быть пустыми, однако поле `<space>` является обязательным, если:

– вся строка пустая;

– строка начинается с комментария;

– строка заканчивается перед оставшимся полем.

Пример кода на ассемблере ST7 представлен на рисунке 2.5.

Рис. 2.5. Пример кода на ассемблере ST7

Примечание: не ставьте END в конце файла включения (*.inc), иначе ассемблерный код (*.asm) никогда не будет откомпилирован.

Метки, мнемоники и директивы

Меткой может быть имя переменной или адрес в коде, для того чтобы упростить доступ к памяти и работу с инструкцией JUMP соответственно. Метки следует всегда начинать с первой колонки, и наоборот, каждое выражение, начинающееся в первой колонке, будет рассматриваться компилятором как метка.

Мнемоники или код операции (рис. 2.5) получают имя, для того чтобы упростить понимание кода программы и заменить машинный код на более интуитивно понятный. Они никогда не могут находиться в первой колонке.

Директивы дают указание ассемблеру или компоновщику для управления процессом трансляции и компоновки исходного текста в файл прошивки:

```
     #INCLUDE "ST7Lite2.INC" ; Директива ассемблера
     WORDS                   ; Директива ассемблера
     segment 'rom'           ; Директива компоновщика
     ...                     ;
.NEXT                        ; Метка
     LD (Table, X), A        ; Мнемоника
     DEC X                   ; Мнемоника
     JRPL NEXT               ; Мнемоника
```

Раздел объявления файлов включения (*.inc)

Следующим в структуре файла исходного кода (Приложение 4) идет раздел объявления файлов включения (*.inc), в которых хранятся прототипы импортируемых функций и переменных, распределение регистров и памяти конкретного семейства микроконтроллеров фирмы STM.

Например, файл ST7Lite2.inc содержит прототипы предопределенных переменных (распределение регистров и памяти) для МК ST7FLite29. Общая структура проекта для МК ST7FLite29 представлена на рисунке 2.6.

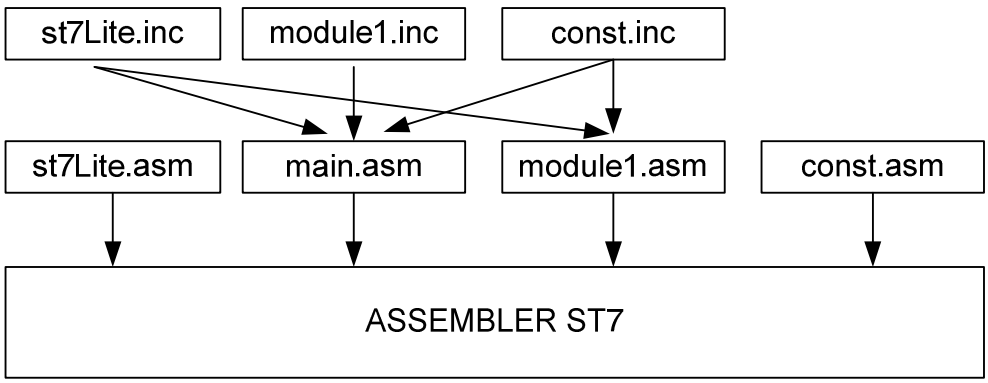


Рис. 2.6. Структура проекта для МК ST7FLite29

Данная структура не является строгой, но, файлы ST7Lite2.asm и ST7Lite2.inc должны быть включены в проект обязательно.

Разработчику рекомендуется выносить константы и переменные всего проекта в отдельные файлы – например, const.asm и variable.asm и делать глобальными только те из них, которые он сам считает необходимым, описывая их в файле const.inc и variable.inc соответственно.

Ключевые слова PUBLIC, LOCAL и EXTERN

По умолчанию все переменные, константы и т.д. в исходном коде являются локальными, т.е. недоступными извне. Если же предполагается экспортировать эти данные в другой модуль, для этого необходимо объявить их глобальными.

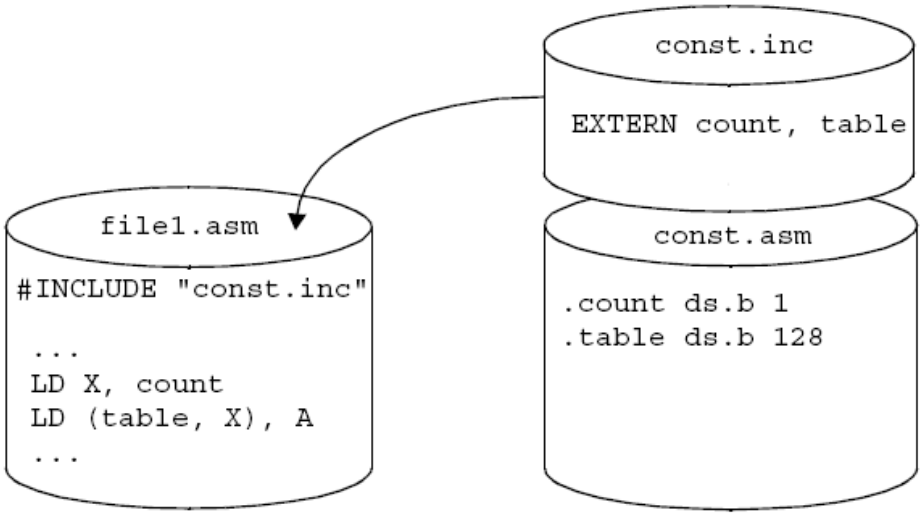


Рис. 2.7. Схема взаимодействия между файлами проекта. Использование директив PUBLIC, EXTREN.

Сделать это можно несколькими способами – поставить точку перед меткой (например, `.var_exp`) или использовать ключевое слово PUBLIC (см. приложение 2) перед списком глобальных переменных в первом модуле. Второй шаг – размещение ключевого слова EXTERN в файле включения (*.inc) перед необходимым списком внешних переменных и подключение этого файл к другому модулю (рис. 2.7).

Раздел объявления символов

Символы, как и константные типы данных, определяются в исходном файле и не могут изменяться во время выполнения программы. Основным отличием является то, что их значения доступны только на время сборки (времени ассемблирования), а не во время выполнения программы. Особую ценность символы представляют для генерации константных данных. Символические значения определяются при помощи оператора `EQU`. Например:

```
A_DOZEN EQU 12
```

После этого, слово A_DOZEN может быть там, где есть необходимость использовать число 12. Данный подход особенно полезен, если определенное значение предполагается применять в нескольких частях программного кода. Например, программа, которая отображает символы на ЖКИ-индикаторе, обязана «знать» количество символов, которые она может показать:

```
DISPLAY_WIDTH EQU 16
```

Таким образом, если DISPLAY_WIDTH используется несколько раз в программе, при смене числа 16 на 24 нет необходимости пересматривать код, а просто сделать замену в одном месте программы.

Второе применение символов – выражения:

```
DISPLAY_WIDTH EQU 24
DISPLAY_HEIGHT EQU 4
Total_Chars: DC.b (DISPLAY_WIDTH * DISPLAY_HEIGHT)
```

В данном случае два символа DISPLAY_WIDTH и DISPLAY_HEIGHT не занимают никакой памяти, когда константа Total_Chars является результирующей и занимает 1 байт в памяти МК.

Таким образом, если значение данных константного типа будет изменяться в зависимости от требований к программе или по другим причинам, имеет смысл использовать символы.

Раздел объявления констант и переменных

Перед тем как в программном коде использовать переменные и константы, они должны быть изначально объявлены. Для выделения части оперативной памяти RAM (для переменных) необходимо использовать ключевые слова DS.b и DS.w для определения переменной типа байт (byte - 8 бит) и типа слово (word - 16 бит) соответственно:

```
aByte: DS.B 1    ; переменная типа байта
aWord: DS.W 1    ; переменная типа word – 2 байта
Array1: DS.B 20  ; массив 20 однобайтовых переменных
Array2: DS.W 40  ; массив 20 двухбайтовых переменных
```

Константы в языке ассемблер ST7 могут быть двух типов: константные данные и данные символьного типа.

Для объявления констант необходимо выделить часть памяти (ROM), для этого используются следующие директивы – DC.b (byte), DC.w (word). Использование ключевого слова WORD вместо DC.w позволяет выделить память, начиная с младшего байта, однако BYTE и DC.b работают одинаково:

```
.PowerOf2 DC.b 1,2,4,8,16,32,64,128  ; степень 2
```

Следующая инструкция позволяет считать один байт из таблицы `.PowerOf2` в зависимости от значения индекса в регистре **`X`**. Результат возведения числа в степень 2 помещается в аккумулятор (**`A`**):

```
LD A, PowerOf2,(X)
```

Ключевое слово `STRING` используется для объявления строки символов, а также любая последовательность байтов может быть представлена подобным образом:

```
Message: STRING "Hello"; строки Message и Message2 идентичны
Message2: STRING 48h, 45h, 4Ch, 4Ch, 4Fh
```

Примечание: необходимо использовать ключевое слово `DS.` всегда, когда надо выделить память под переменную, вместо ключевого слова `EQU`, которое не резервирует никакой памяти.

Примечание: для оптимизации работы программы рекомендуется размещать часто используемые переменные на «нулевой странице» (‘ram0’, т.е. от 0 до 0FFh).

BYTES, WORDS и понятие segment

Директивы `BYTES`, `WORDS` определяют 8 или 16 битовый адрес для меток, следующих за ними. Следовательно, ключевое слово `BYTES` необходимо располагать перед определением аппаратных регистров и переменных в оперативной памяти (RAM) на «нулевой странице» (0..0FFh). Директиву `WORDS` следует использовать во всех остальных случаях, например, при определении переменных в RAM на других страницах, EEPROM переменных, для программного кода, констант и векторов прерывания в памяти ROM.

Ключевое слово `segment` является директивой компоновщика и используется для определения границ памяти. Данная директива опрашивается каждый раз, когда вы хотите поместить переменную или программный код в разных границах памяти.

Существует несколько групп сегментов, называемых классом. Понятие класс не имеет никаких свойств. Их концепция была введена для того, чтобы помочь разработчику сформировать адресуемое пространство в зависимости от характеристик конкретных областей памяти. В большинстве случаев стандартным является набор классов, представленных в таблице 2.1.

Для блока исходного кода, директива `segment` может использоваться только после безусловного перехода или инструкции выхода из подпрограммы из-за того, что два сегмента являются независимыми объектами, которые могут располагаться в различных частях памяти. Т.е. два сегмента, которые следует друг за другом в исходном

файле, могут быть размещены в разных частях памяти в процессе трансляции исходного кода. Разделение программы на сегменты позволяет упростить процесс ее размещения.

Таблица 2.1

Набор стандартных классов сегментов

Имя класса (примеры)	Тип класса и область применения
‘ROM’	в ПЗУ (ROM) МК, для макрокоманд
‘RAM0’	в ОЗУ (RAM) МК, «нулевая страница» (0..0FFh)
‘RAM’	в ОЗУ (RAM) МК, используется расширенная адресация
‘STACK’	в ОЗУ (RAM), доступна указателю стека
‘IO’	для регистров ввода/вывода (всегда на «нулевой странице»)

Примечание: ассемблер не может определить длину (разрядность) метки по ключевому слову `segment`, которое распознает только компоновщик, вот почему необходимо указывать длину метки, используя ключевые слова `BYTES` и `WORDS`.

Примечание: `WORDS` является значением по умолчанию. Рекомендуется размещать его в начале *.asm - файла и в конце файла *.inc.

Примечание: `BYTES` и `WORDS` определяют не длину переменной, а длину адреса переменной. Возможна ситуация, когда 16-битовые данные (слово) размещено на «нулевой странице» и байт (8-битовая переменная) хранятся после адреса $100. Таким образом, нельзя смешивать в программном коде директивы `BYTES` и `WORDS` с `BYTE` и `WORD`:

```
BYTES
segment 'ram0'
.count ds.w 1        ; зарезервировать на 0-ой странице RAM
                     ; переменную count типа word (2 байта)
WORDS
segment 'ram1'
.step ds.b 3         ; зарезервировать 3 байта для переменной
step
segment 'rom'
.rate dc.w 9600 ; зарезервировать константу rate типа word в ROM
.tab dc.b $AA,%01010101 ; зарезервировать константу tab в
                        ; два байта
```

Раздел объявления подпрограмм

Большинство современных программ имеют нелинейную структуру. Зачастую при написании исходного кода разработчик сталкивается с необходимостью выполнения микроконтроллером одних и тех же часто повторяющихся операций, например: обмен данными с перифе-

рийными устройствами (опрос датчиков и исполнительных устройств), работа с индикацией, обработка прерываний и т.д.

Для уменьшения количества строк и структуризации исходного кода программы используется модульно-процедурный подход программирования, где основной минимальной структурной единицей является подпрограмма (функция или процедура).

Ассемблер ST7 позволяет разработчику делить основную программу на процедуры и функции и, в зависимости от выполняемой задачи, вызывать необходимые.

Ниже приведен пример подпрограммы формирования задержки:

```
wait:
     ld X,#255
decr2:
     ld Y,#255
decr1:
     dec Y
     jrne decr1
     dec X
     jrne decr2
     ret
```

Обязательными параметрами является пара – метка (имя подпрограммы) и ключевое слово **ret** (возврат из подпрограммы).

Для вызова процедуры в исходном коде главной программы используется инструкция **CALL** имя_подпрограммы. Например:

```
call init_port
call wait
```

После выполнения директивы **CALL** адрес следующей за ней инструкции сохраняется в стеке и только после этого осуществляется переход в подпрограмму. Специальная команда **ret** осуществляет выход из подпрограммы – загружает в счетчик команд (PC) хранимое значение адреса следующей команды ассемблера основной программы из стека. Все подпрограммы должны быть определены перед тем, как будут использованы в исходном коде (перед вызовом **CALL**). Как уже было сказано выше, разработчик также получает возможность их экспортировать в другие файлы проекта (использовать в качестве библиотечных функций), для этого используются ключевые слова PUBLIC и EXTERN.

Раздел описания основной части программы

Выполнение программы на микроконтроллере ST7FLite29 начинается с вызова вектора прерывания RESET, которому может быть назначена подпрограмма, в разделе описания векторов, обрабатывающая его. Она и является основным телом программы, с которой начинается выполнение всего программного кода. В нашем случае (см.

Приложение 4), вектор прерывания `RESET` обрабатывает подпрограмма `main`, которая, в свою очередь, является точкой входа в основную программу (см. рис. 3.7).

Раздел объявления подпрограмм обработки прерываний

Микроконтроллер – программируемый компонент, который выполняет только одну инструкцию в определенный промежуток времени. Однако, большинство, если не все приложения, требуют управления многими задачами одновременно. Обычно для повышения эффективности и упрощения систем на микроконтроллере разработчики пытаются реализовать большее количество функциональных возможностей на одном чипе. Решением подобной проблемы находится в двух сферах – программной и аппаратной. Аппаратный подход – применение механизма прерываний, программный – многозадачность.

Прерывание в компьютерной терминологии – механизм, который позволяет приостановить текущее выполнение программы в случае возникновения какого-либо события с более высоким приоритетом. После этого события вызывается подпрограмма обработки прерывания, а затем происходит возврат в исходную точку главной программы (рис. 2.7).

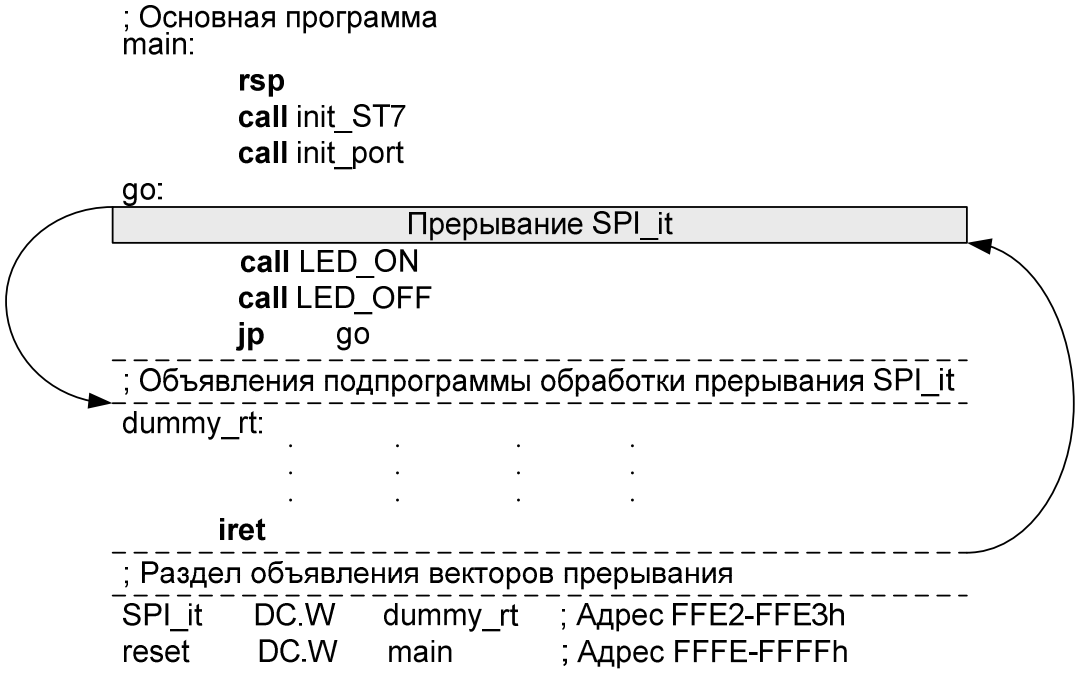


Рис. 2.7. Пример выполнения прерывания

При появлении запроса на прерывание от внешнего источника, первое, что делает микропроцессорное ядро (после окончания выполнения текущей инструкции) – сохранение состояния внутренних регистров, чтобы по возврату из подпрограммы обработки прерывания продолжить работу без потери данных. Данная процедура выполняет-

ся следующим образом: значения всех регистров ядра сохраняются в стеке (счетчик команд, регистр **X**, аккумулятор, регистр признака) кроме значения регистра **Y**, т.к. архитектура ST7 была наследована от архитектуры микроконтроллера, у которого он отсутствовал. Если существует необходимость, то значение регистра **Y** может быть принудительно помещено в стек в начале выполнения подпрограммы обработки прерывания (команда **PUSH Y**).

Подпрограмма обработки прерывания всегда представлена в виде пары: метка – имя и ключевое слово **IRET** – выход из подпрограммы (рис. 2.7). После выхода из подпрограммы обработки прерывания происходит возврат в основную программу, при этом из стека восстанавливаются предыдущие значения регистров.

В том случае, если программист до этого сохранял значение регистра Y, его также необходимо восстановить. Для этого используется директива ассемблера **POP Y**.

Чтобы предостеречь выполнение подпрограммы обработки прерывания от срабатывания другого прерывания, в регистре признака результата автоматически устанавливается 5-ый бит (флаг **I**). Некоторые периферийные узлы МК автоматически снимают его (устанавливают в 0) после окончания работы процедуры. В этом случае никаких особых действий выполнять не надо.

Другие же модули (например, таймер) не снимают флаг **I** даже после завершения процедуры обработки. Поэтому его обязательно необходимо сбрасывать вручную в теле подпрограммы обработки прерывания до ключевого слова **IRET**, в противном случае она будет вызвана повторно (войдет в бесконечный цикл) и основная программа будет заблокирована. Еще один случай, когда может понадобиться ручной сброс флага прерывания **I** – это необходимость приостановки текущего выполнения обработчика (например, время его работы достаточно большое). Однако прибегать к подобным процедурам желательно только при необходимости, т.к. размер стека ограничен.

Раздел объявления векторов прерывания

Последним разделом в структуре главного файла проекта идет область объявления векторов прерывания.

Основным назначением инициализации векторов прерывания является необходимость указания микропроцессорному ядру адреса кода, который должен выполняться при возникновении того или иного внешнего события (формировании запроса).

Вектора прерывания представляют собой таблицу 16-битовых слов в памяти программ, которые содержат начальный адрес различных подпрограмм обработки прерывания (см. приложение 4).

В зависимости от источника прерывания (ввод/вывод, таймер и т.д.) ядро выбирает из заранее определенной области адрес подпрограммы обработки прерывания. Вектора всегда размещаются в конце адресного пространства микроконтроллера. Для каждого источника

прерывания существует один вектор, плюс еще один для события RESET. Когда происходит прерывание или наступает событие RESET, выбирается необходимый вектор для выполнения подпрограммы обработки прерываний или запуска основной программы.

2.2. Введение в интегрированную среду разработки ST7 Visual Developer

2.2.1. Установка среды

Для разработки и отладки проектов под микроконтроллеры ST7 необходимо наличие программной среды. Фирма STM предлагает интегрированный пакет (Toolset), который включает в себя два средства:

– ST7 Visual Developer (STVD7) с возможностью внутрисхемной отладки проектных решений;

– ST7 Visual Programmer (STVP7) – средство для программирования МК.

Перед установкой программного обеспечения ST7 Visual Developer (STVD7 IDE), необходимо учитывать следующие требования, предъявляемые к системе:

– операционная система Windows 98, Windows Me, Windows 2000 или Windows XP;

– минимальный объем 64 Мб оперативной памяти (RAM) и 40 Мб памяти доступной на жестком диске;

– USB, COM или LPT – порты (в зависимости от программатора).

Обязательным требованием является наличие прав администратора при установке программного обеспечения.

Процесс установки состоит из нескольких этапов:

– выбор необходимого пакета установки (рис. 2.8);

– настройка параметров установки.

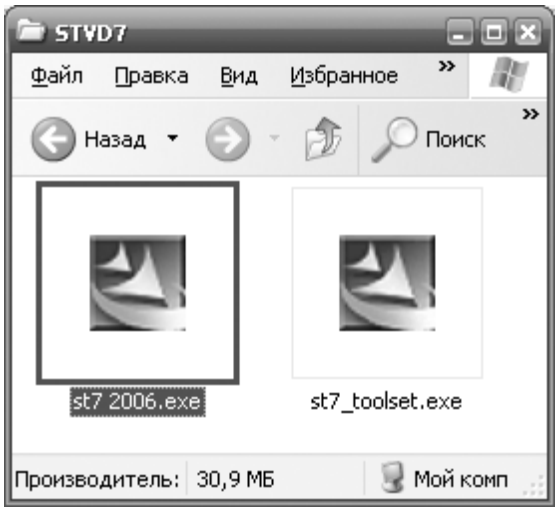


Рис. 2.8. Выбор пакета установки

Выбор пакета установки зависит от типа решаемой задачи, а также наличия наборов необходимых аппаратных средств – комплектов разработки (Development Kits).

Процесс инсталляции пакета представляет собой стандартную процедуру установки приложений с указанием места, куда производится установка, а также необходимости создания ярлыков на рабочем столе.

Работа с отладочным комплектом “ST7/ST5 training board” требует следующих программных компонент:

- кросс-ассемблер (asm);
- компоновщик (lyn);
- средства форматирования (obsend);
- библиотека (lib).

Пакет разработки (полная библиотека ST7, STVD7 и ST макроассемблер) распространяется бесплатно и может быть взята с официального сайта фирмы STM (http://stm.com/mcu).

Процесс установки пакета ST7 Toolset (Version 3.11) фирмы SofTec Microsystems состоит из нескольких основных этапов.

1. Выбор места установки пакета на жестком диске (рис. 2.9).

Рис. 2.9. Установка пакета ST7 Toolset (Version 3.11)

2. Выбор параметров установки (рис. 2.10).

Для работы с комплектом разработки “ST7/ST5 training board” наличие установленного компонента inDART-STX является обязательным.

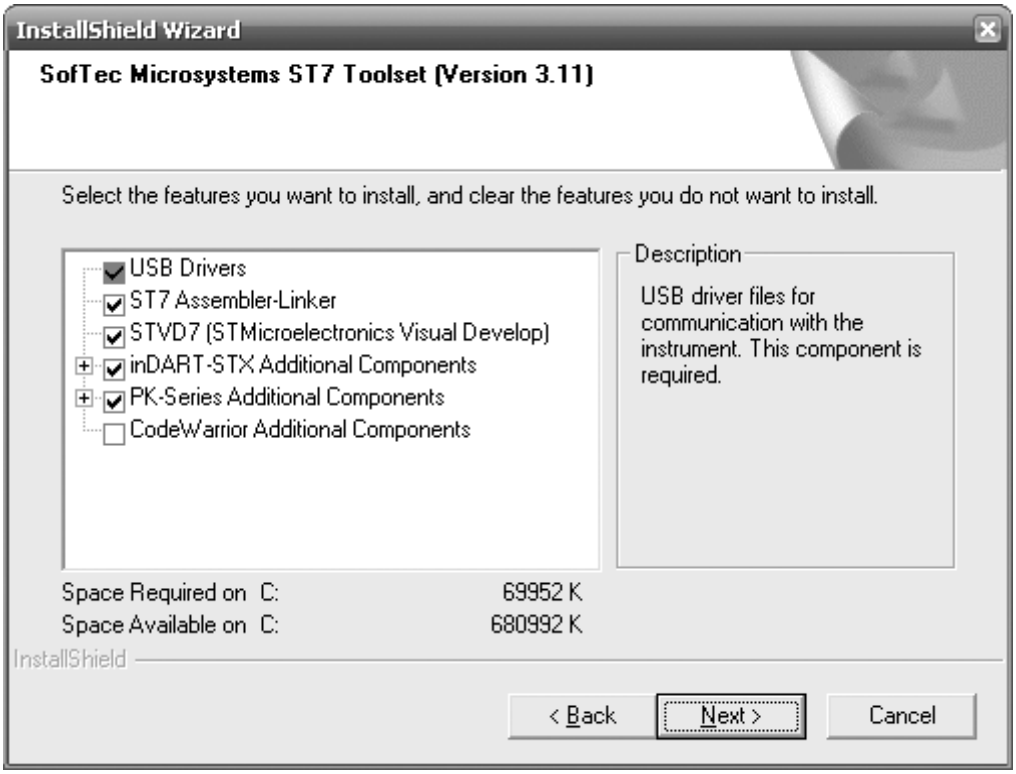


Рис. 2.10. Выбор параметров установки пакета ST7 toolset

3. Выбор имени папки программы для размещения ярлыков пакета в меню быстрого запуска (рис. 2.11).

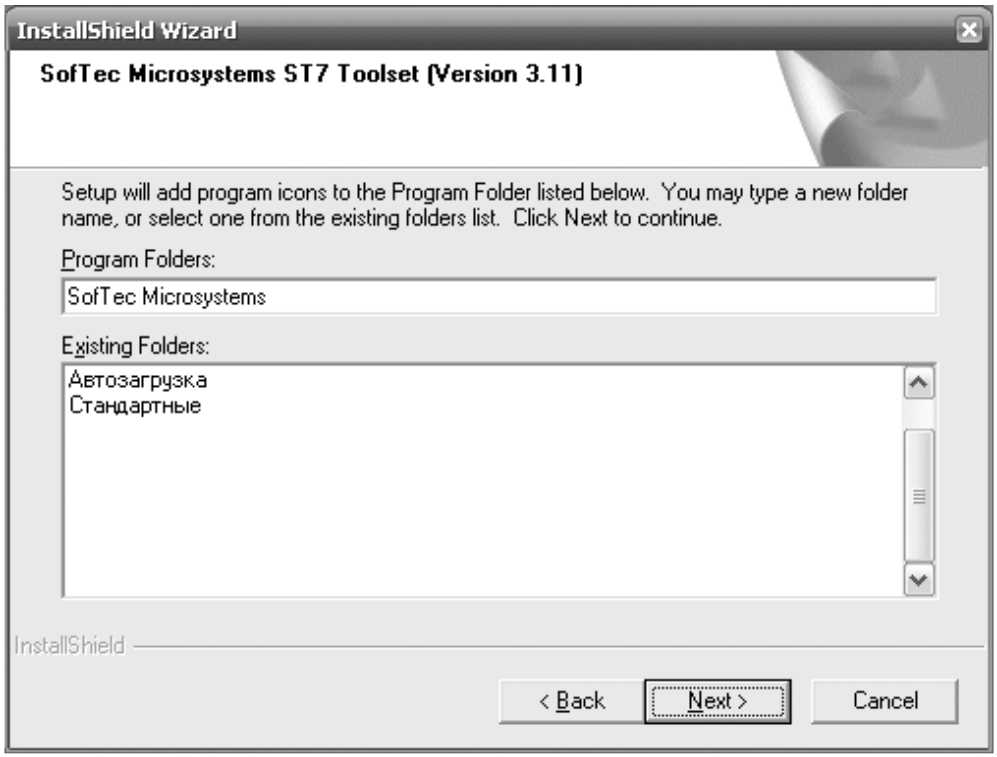


Рис. 2.11. Выбор имени папки программы для размещения ярлыков пакета в меню быстрого запуска

4. Установка пакета с необходимыми параметрами (рис. 2.12).

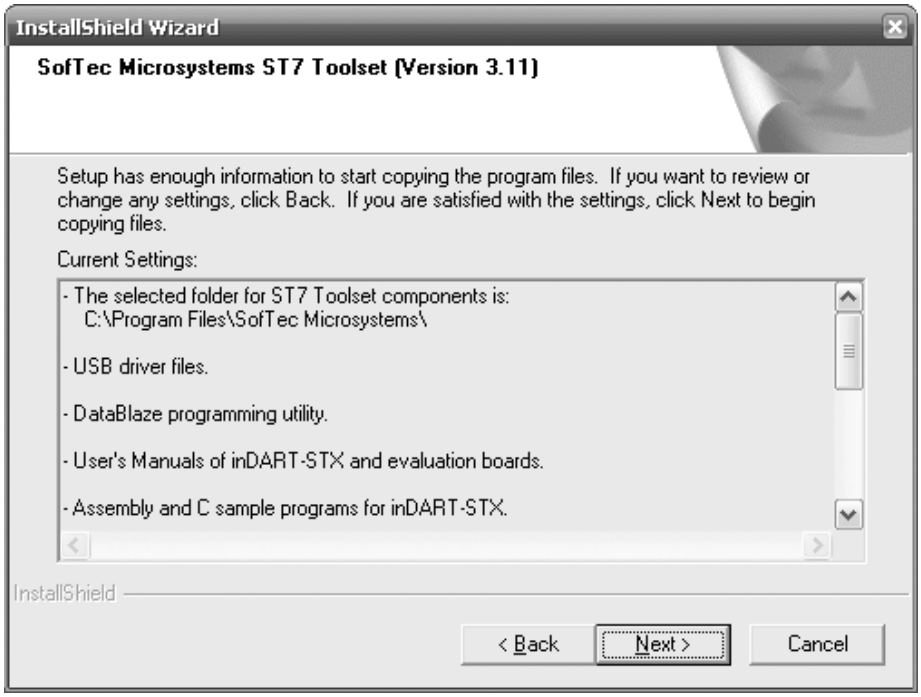


Рис. 2.12. Установка пакета ST7 Toolset фирмы SofTec Microsystems

Во время установки программа проводник скопирует необходимые файлы на жесткий диск.

2.2.2. Общая характеристика среды

ST7 Visual Developer IDE представляет собой среду разработки проектов под микроконтроллеры фирмы STM (рис 2.13).

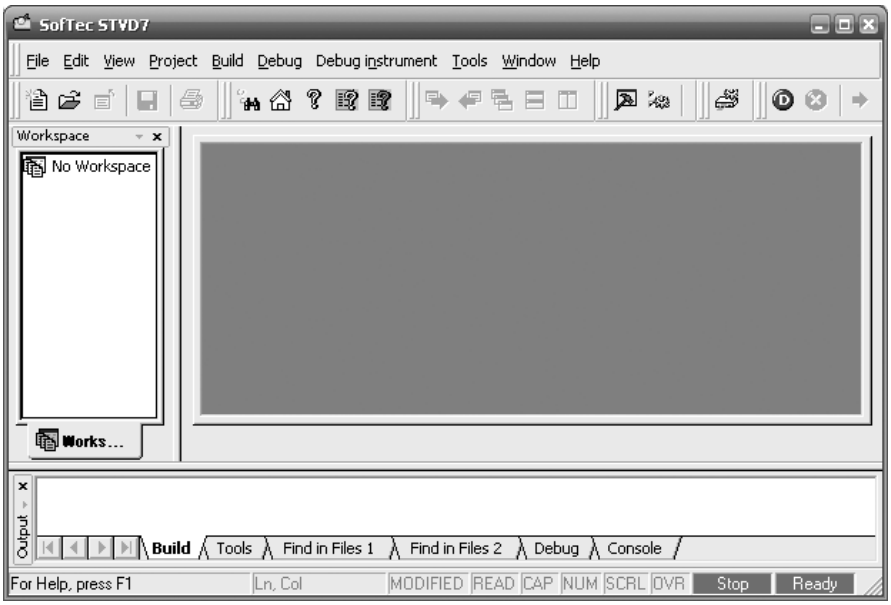


Рис. 2.13. Общий вид среды разработки STVD7 IDE

Близкие по назначению проекты могут объединяться в наборы проектов – рабочую среду (Workspace). В начале разработки программы необходимо создать рабочую среду, после чего добавить проект.

2.2.3. Запуск среды на выполнение и создание проекта

Для создания проекта необходимо:

1. Запустить ST7 Visual Developer IDE на выполнение (рис. 2.14).

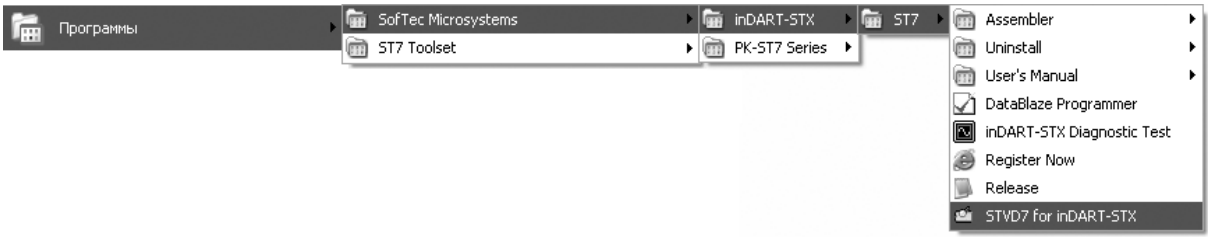


Рис. 2.14. Запуск среды STVD7 на выполнение

Для этого в меню
Пуск->Программы->SofTecMicrosystems->inDART-STX->ST7 выбрать ярлык “STVD7 for inDART-STX” как показано на рис. 2.14, после чего на экране появится основное окно рабочей среды (2.15)

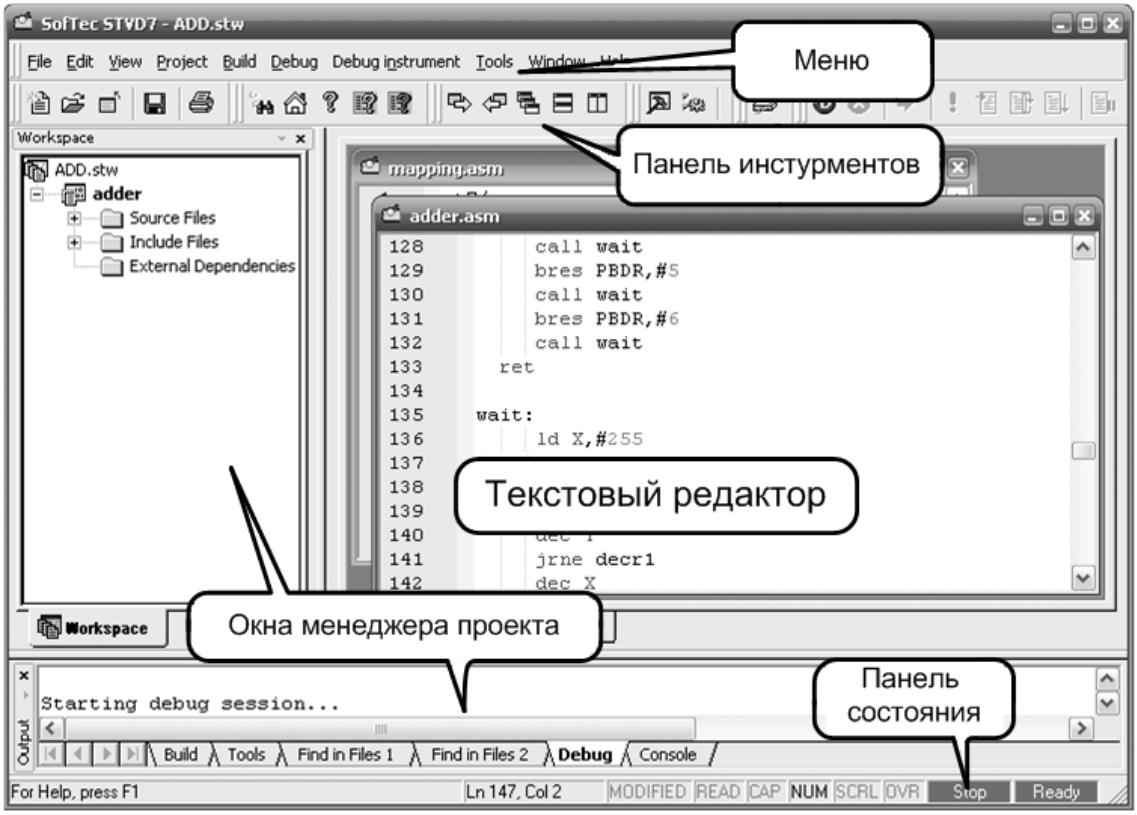


Рис. 2.15. Интегрированная среда разработки STVD7 IDE

2. Для создания новой рабочей среды (проекта) в меню “File” необходимо выбрать “New Workspace...” (рис. 2.15).

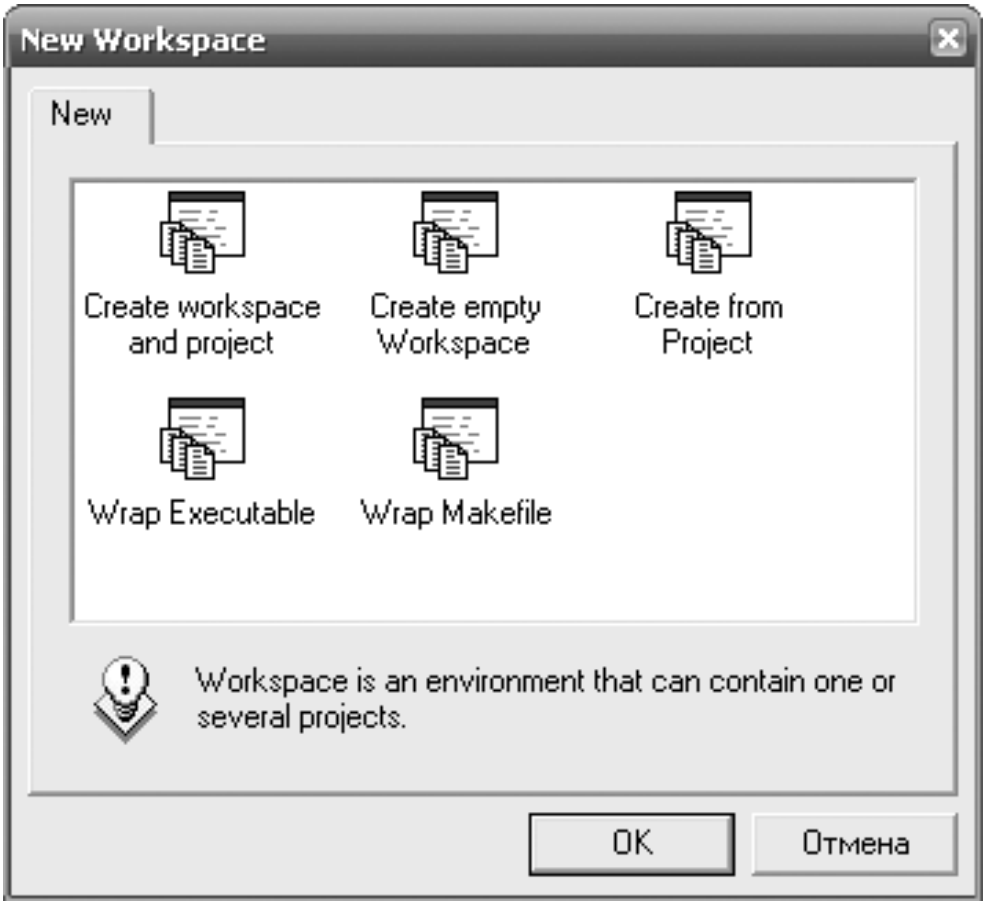


Рис. 2.16. Диалоговое окно создания рабочей среды

В появившемся диалоговом окне New Workspace (рис. 2.16) существуют следующие варианты выбора:

– создать рабочую среду вместе с новым проектом (Create workspace and project);
– создать пустую рабочую среду (Create empty Workspace);
– создать из проекта (Create from Project);
– создать обертку для исполняемого файла (Wrap Executable);
– создать обертку для сборочного файла проекта (Wrap Makefile).

3. Выбираем первый пункт “Create workspace and project”.

В появившемся диалоговом окне (рис. 2.17) необходимо ввести имя рабочей среды (Workspace filename) и путь к каталогу, где будут размещаться файлы среды (Workspace location).

Следует отметить, что для корректной работы приложения полный путь и имена файлов рабочей среды по возможности не должны содержать кириллицу.

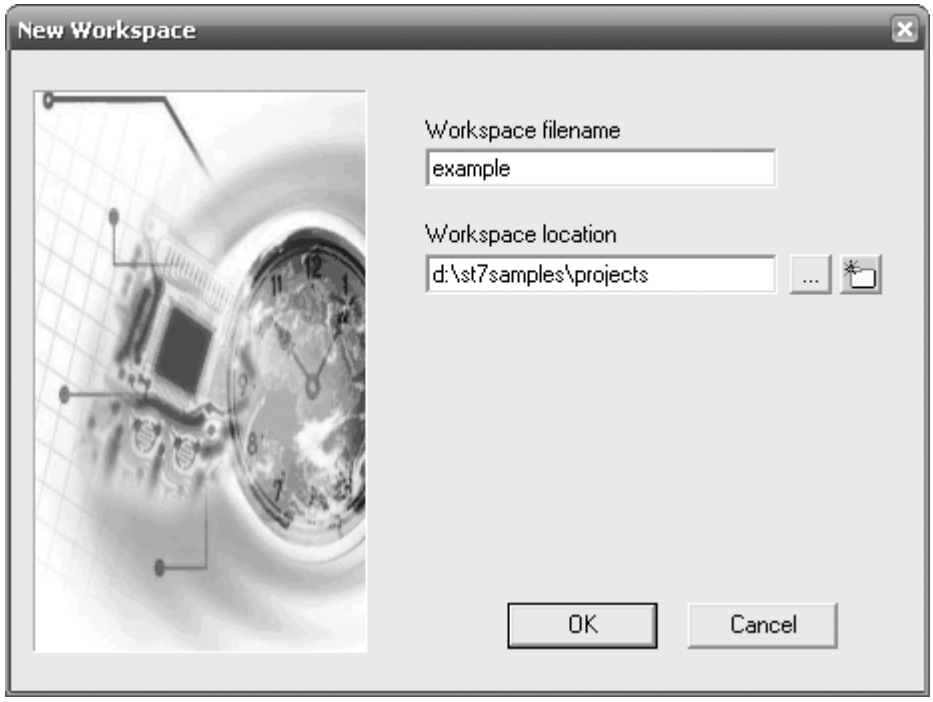


Рис. 2.17. Диалоговое окно создания рабочей среды

4. После создания рабочей среды необходимо указать имя проекта, полный путь к папке проекта, а также требуемый компоновщик (в нашем случае “ST7 Assembler Linker”) (рис. 2.18).

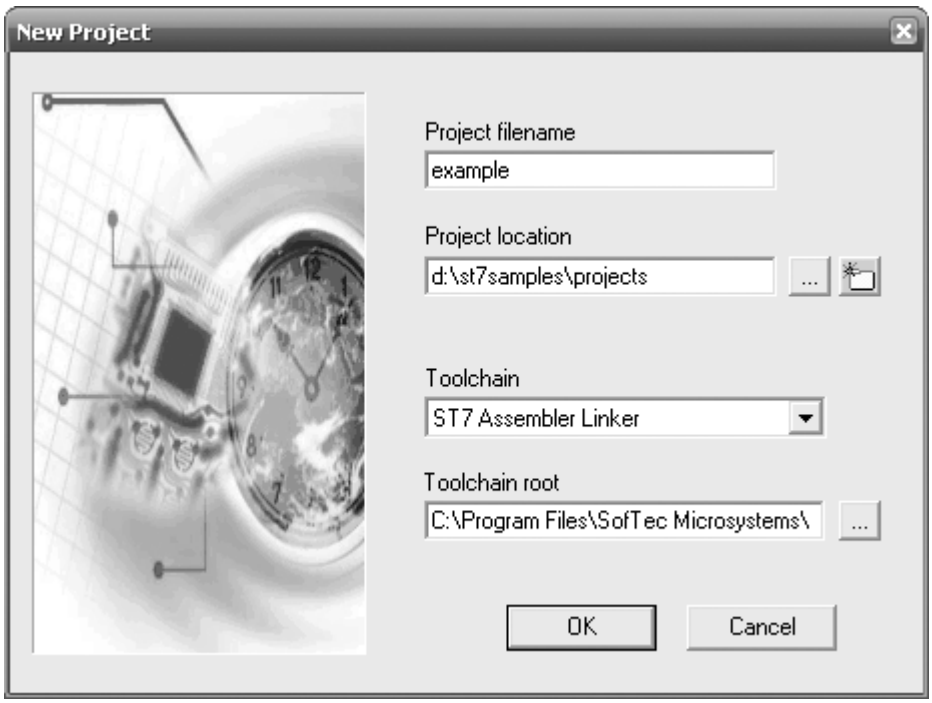


Рис. 2.18. Диалоговое окно создания проекта

5. Последним этапом при создании проекта является выбор семейства микроконтроллера, под который будет разрабатываться проект (рис. 2.19).

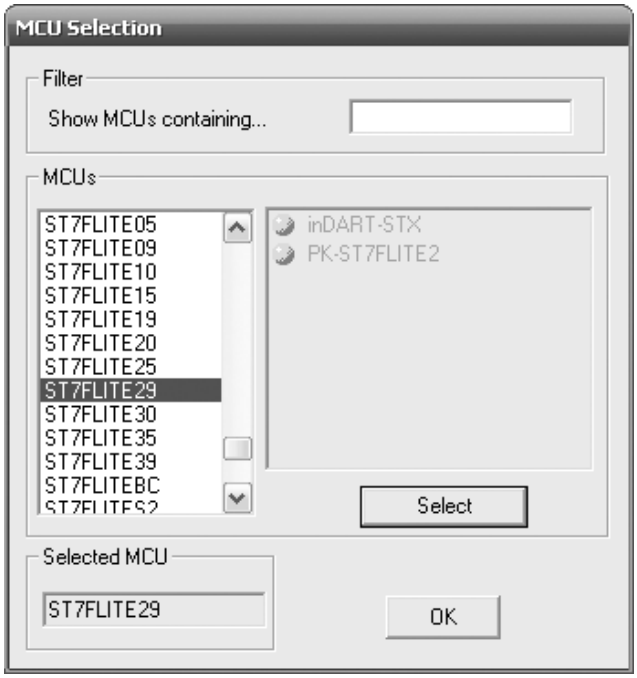


Рис. 2.19. Диалоговое окно выбора типа микроконтроллера

По завершении всех перечисленных действий необходимо нажать кнопку OK.

2.2.4. Тестирование и отладка программ

Среда разработки STVD7 поддерживает механизм внутрисхемной отладки проектных решений (рис. 2.20).

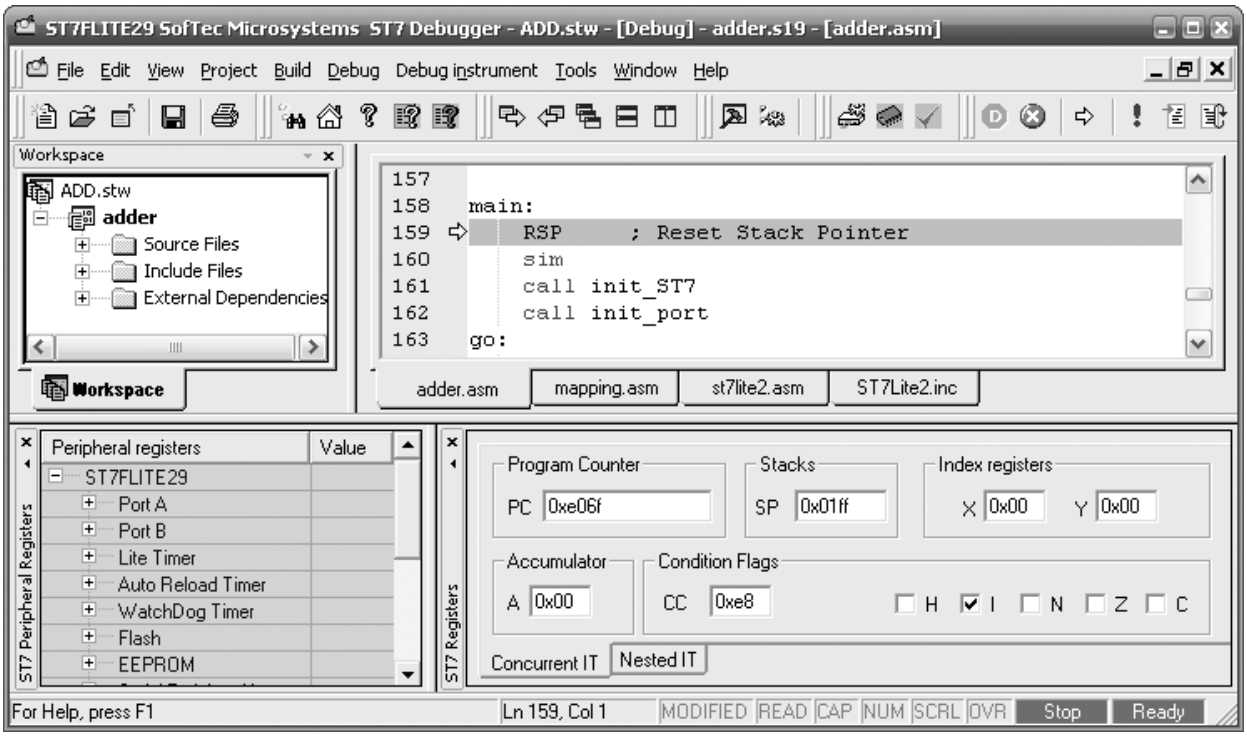


Рис. 2.20. Отладка в среде STVD7

Основным преимуществами использования STVD7 является возможность получения расширенных результатов работы приложения (значения регистров периферийных модулей, значения внутренних регистров ядра, стека, регистра признака, состояние переменных в памяти ROM, RAM и т.д.) в реальной масштабе времени с использованием режима внутрисхемной отладки.

2.2.5. Знакомство с меню

2.2.5.1. File (Файл)

Данное меню предоставляет стандартные команды для управления рабочими областями и текстовыми файлами, включает следующие пункты: New, Open, Close, Close All, Save, Save as, Save All, Print и Recent (рис 2.21).

Команды для работы с рабочими областями New Workspace и Open Workspace достаточно важны, поскольку перед созданием проектов, разработкой и отладкой ваших приложений вы должны создать (или открыть существующую) рабочую область.

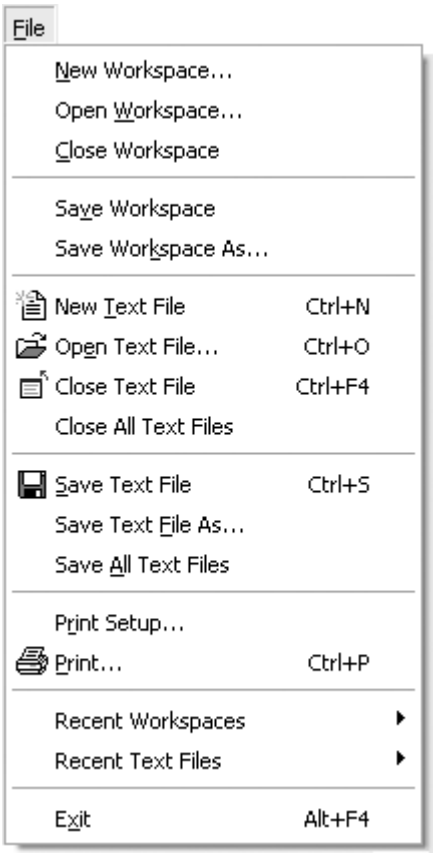


Рис. 2.21. Меню «Файл»

2.2.5.2. Edit (Правка)

Данное меню включает команды редактирования файла и поиска текста. Эти команды доступны, когда открыто окно редактора в режимах создания и отладки. Дополнительно это меню предоставляет команды точек останова и доступ к окну *QuickWatch*, которое использу-

ется при отладке. Ниже приведена информация о командах этого меню (рис 2.22).

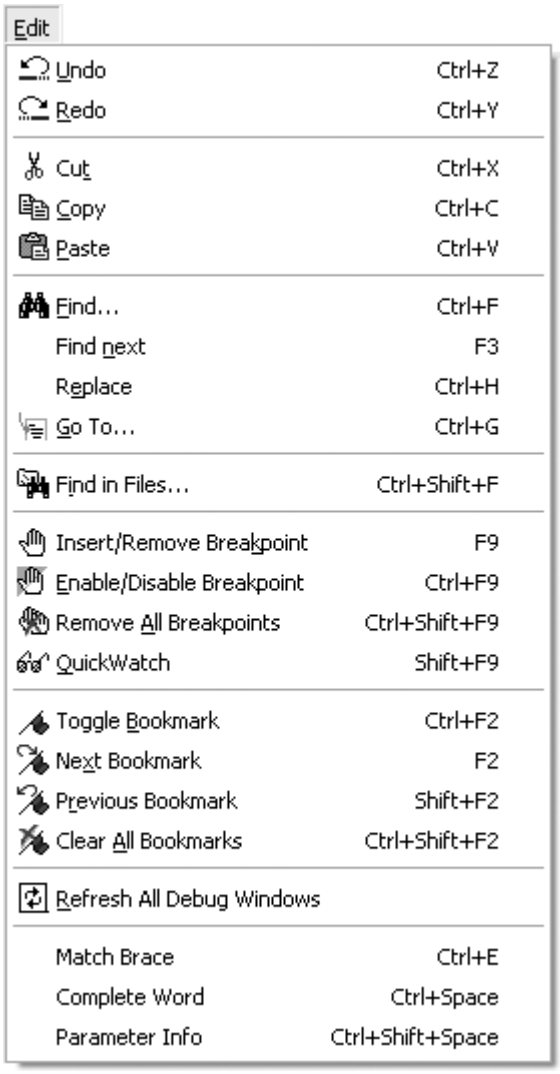


Рис. 2.22. Меню «Правка»

Undo/Redo

Данная команда отменяет последнюю команду редактора или возвращает последнюю отмененную команду.

Cut, Copy, Paste

Стандартные операции с буфером обмена: вырезать, копировать, вставить.

Find…, Find Next и Replace

Данные команды предназначены для поиска и/или замены строки в редактируемом тексте. Команда *Find* также доступна в контекстном меню.

Go to

Позволяет осуществить переход к указанной строке, адресу или функции.

Find in Files

Находит строку в любом из файлов в заданной директории.

Breakpoints

Позволяет добавить точки останова в текст программы в активном окне редактора, после чего программа будет выполняться с учетом этих точек.

Bookmarks

Вставка / удаление закладок, а также навигация по ним в пределах текущего файла.

Quick Watch

Данная команда открывает окно *Quick Watch*, предназначенное для быстрого доступа к функциям просмотра значений переменных.

Refresh

Регенерирует все окна.

Match Brace

С помощью данной команды можно перейти к скобке, которая соответствует скобке, подсвеченной в окне редактора.

Complete Word

Данная команда выводит список возможных завершений введенного слова.

Parameter Info

Открывает информацию о синтаксисе вводимой инструкции ассемблера ST7.

2.2.5.3. View (Вид)

Команды данного меню позволяют открывать такие окна, как Workspace, Output и Instruction Breakpoint в режиме создания или отладки. В режиме отладки это меню также позволяет открывать различные окна для просмотра исходного кода программы, состояния регистров, памяти, стека и переменных (рис. 2.23).

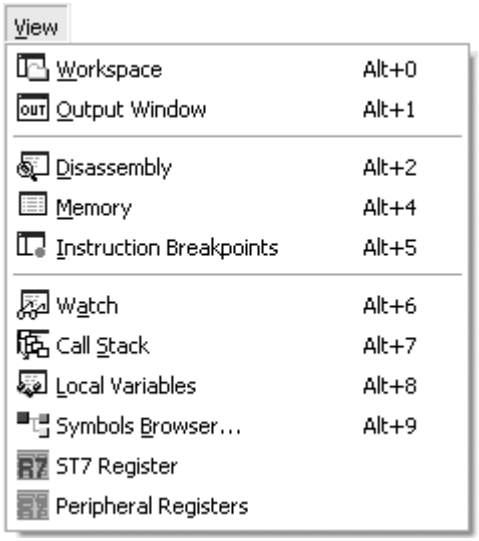


Рис. 2.23. Меню «Вид»

2.2.5.4. Project (Проект)

Данное меню предоставляет доступ к настройкам проекта, просмотру зависимостей и т.п. (2.24).

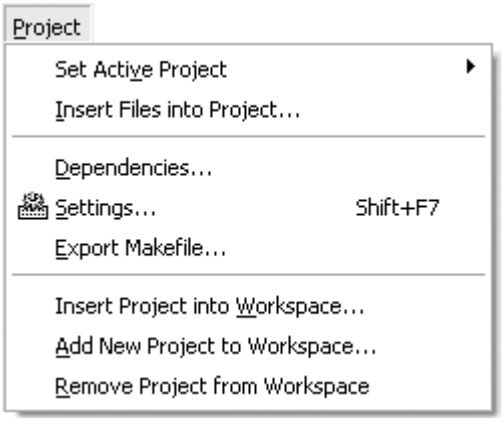


Рис. 2.24. Меню «Проект»

Set Active Project

Позволяет выбрать проект, который вы хотите сделать активным для изменения или конфигурирования.

Insert Files into Project

Открывает окно, в котором существующий файл можно вставить в активный проект.

Dependencies...

Позволяет устанавливать зависимости среди проектов в рамках текущей рабочей области.

Settings...
Открывает окно настроек проекта.

Export Makefile...
Генерирует makefile-скрипт, который может быть запущен на выполнение с помощью утилиты gmake вне среды STVD7.

Insert Project into Workspace
Открывает окно, в котором можно указать путь к существующему проекту и добавить его в текущую рабочую область.

Add New Project to Workspace
Позволяет создать новый проект и добавить его в текущую рабочую область.

Remove Project from Workspace
Удаляет активный проект из рабочей области.

2.2.5.5. Build (Сборка)

Содержит команды, которые позволяют конфигурировать, запускать и останавливать создание вашего приложения. Эти команды, за исключением Compile и Batch Build, обращаются к активному проекту в текущей рабочей области. Compile применяется к файлу программы, который Вы выбрали в окне рабочей области. Batch Build применяется к проектам, которые выделены в окне Batch Build. Следует отметить, что доступа к командам в этом меню нет до тех пор, пока не будут созданы рабочая область с проектом (рис. 2.25).

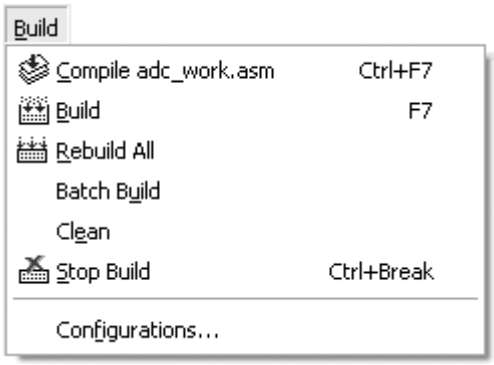


Рис. 2.25. Меню «Сборка»

2.2.5.6. Debug (Отладка)

Данное меню предоставляет доступ к командам запуска и останова загруженной программы (Run, Restart, Continue, Run to Cursor и Stop), пошагового запуска (Step Into, Step Over, Step Out) и командам Go To PC и Set PC (рис. 2.26). Эти команды предназначены для работы с программой при отладке.

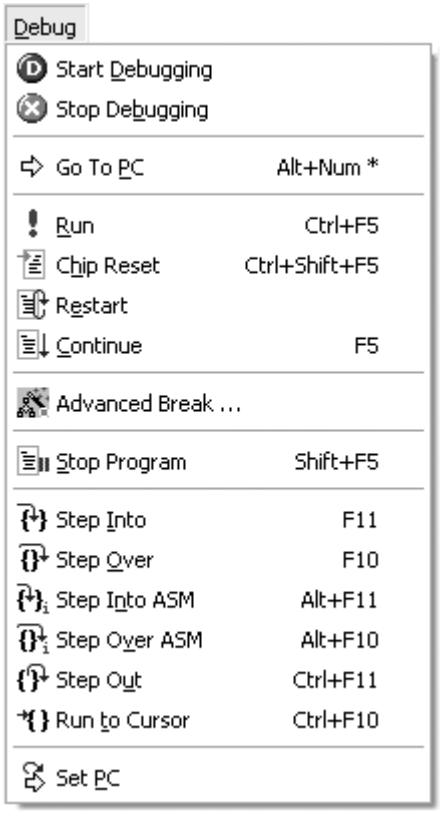


Рис. 2.26. Меню «Отладка»

2.2.5.7. Debug instrument (Средства отладки)

Данное меню предоставляет доступ к опциям, которые предназначены для отладки аппаратных средств. Содержание меню меняется в зависимости от выбранных инструментов отладки. Прежде чем будет выбран инструмент отладки, данное меню представит следующие команды:

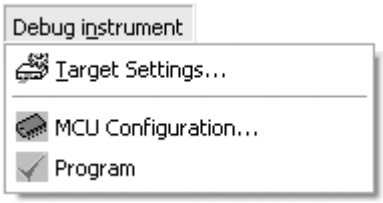


Рис. 2.27. Меню «Средства отладки»

Target Settings

Открывает окно настроек, позволяющее выбрать инструмент отладки и конфигурировать подключение через USB или Ethernet.

2.2.5.8. Tools (Инструменты)

Позволяет настроить внешний вид и расположение окон среды STVD7 и установить различные пользовательские опции (рис. 2.28).

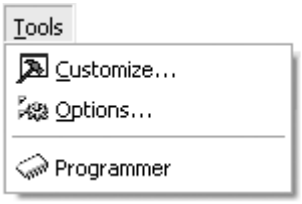


Рис. 2.28. Меню «Инструменты»

Customize

Позволяет выбрать функции для добавления в меню *Tools*.

Options

Позволяет настроить различные пользовательские опции, включая:

- панели инструментов;
- команды;
- опции редактирования/отладки;
- рабочие области.

Programmer

Открывает интерфейс программирования, предназначенный для загрузки программы в микроконтроллер.

2.2.5.9. Windows (Окна)

Данное меню предоставляет доступ к командам упорядочивания открытых окон редактора и навигации по ним (рис. 2.29).

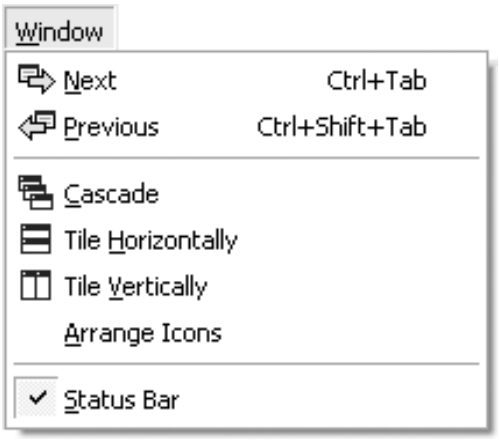


Рис. 2.29. Меню «Окна»

Next/Previous

С помощью данной команды можно переходить по очереди к открытым окнам редактора.

Cascade/Tile
Упорядочивает все открытые окна в пределах главной области приложения согласно выбранной опции. На независимые и связанные окна действие команды не распространяется.

Arrange Icons
Упорядочивает пиктограммы всех свернутых окон.

Status Bar
Управляет представлением строки состояния.

List of opened files
Создает список всех файлов в главном окне. Активное окно помечено специальной отметкой. Если щелкнуть левой кнопкой мыши на любом окне в списке, то это окно становится активным. Двойной щелчок на имени файла открывает его для просмотра.

2.2.5.10. Help (Помощь)

Данное меню предоставляет доступ к командам справки (рис. 2.30).

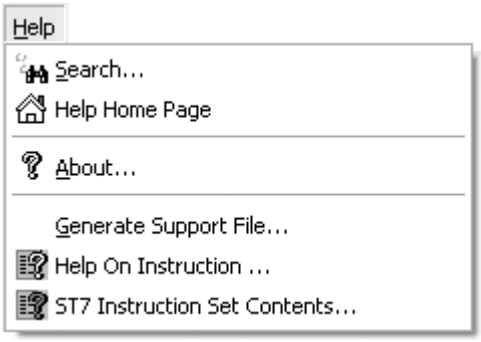


Рис. 2.30. Меню «Помощь»

Search
Осуществляет поиск в online-справке.

Help Home Page
Открывает домашнюю страницу online-справки.

About...
Выводит информацию об STVD7 и инструменте отладки.

Help On Instruction...
Открывает список команд ST7.

ST7 Instruction Set Contents...
Открывает оглавление системы команд ST7.

Generate Support File

Позволяет сгенерировать log-файлы, которые Вы можете отправлять службе поддержки для получения справки.

Контекстное меню

STVD7 предоставляет контекстные меню в окнах просмотра, редактора и рабочей области. Контекстные меню содержат команды, зависящие от окна. В некоторых случаях команды доступны только в контекстном меню. Получить доступ к контекстному меню окна можно щелчком правой клавиши мыши в пределах окна.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В каких случаях оправдано использование языка ассемблера?
2. Перечислите основные этапы разработки проектов на микроконтроллере фимы STM.
3. Укажите выходные файлы проекта при программировании на языке ассемблера ST7.
4. Какое основное назначение компоновщика (linker)?
5. Какова общая структура проекта? Схема взаимодействия между файлами проекта.
6. Перечислите основные разделы файлов проекта.
7. Что такое метки, мнемоники, директивы?
8. Для чего нужен раздел объявления файлов включения? Основные файлы включения при работе с МК ST7FLite29.
9. Основное назначение ключевых слов PUBLIC, LOCAL и EXTERN.
10. Поясните ряд понятий: раздел объявления символов, констант и переменных, сегментация памяти, ключевое слово segment.

РАЗДЕЛ 3. ПРОЕКТИРОВАНИЕ УСТРОЙСТВА КОНТРОЛЯ СОСТОЯНИЯ АВТОМОБИЛЬНОГО ПОДЪЕМНИКА НА ОСНОВЕ МИКРОКОНТРОЛЛЕРА СЕРИИ ST7

3.1. Описание устройства

Общее описание разрабатываемого изделия. Устройство предназначено для контроля состояния опорных стоек автомобильного подъемника, для отслеживания и предотвращения аварийной ситуации перекоса подымаемого транспортного средства. Объект контроля приведен на рис. 3.1. Разрабатываемое устройство предназначено для контроля автомобильного подъемника, оборудованного двумя стойками, не связанными между собой.

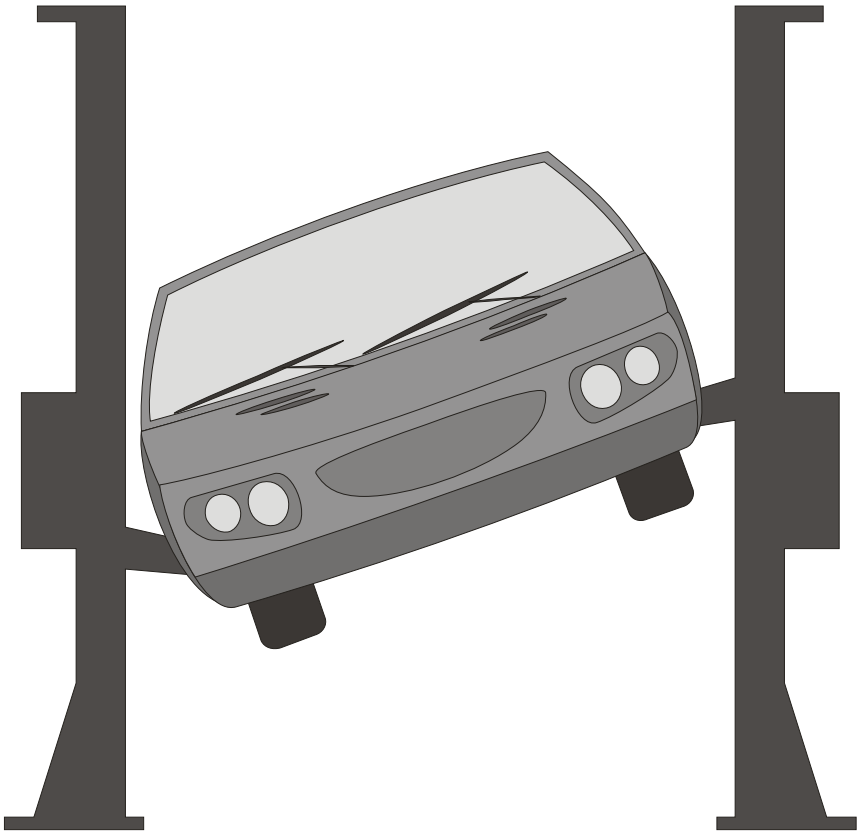


Рис. 3.1. Объект контроля

Каждая стойка автомобильного подъемника оборудована автономным электроприводом. Электропривод состоит из трехфазного асинхронного электродвигателя и механического редуктора. Скорость подъема каждой стойки подъемника зависит от частоты вращения вала электродвигателя, передаточного числа механического редуктора и шага резьбы винтового подъемника. В идеальном случае все стойки должны выполнять перемещение синхронно, потому что винтовые стойки, механические редукторы и электродвигатели на всех стойках одинаковые.

В реальной ситуации механические редукторы и винтовые стойки осуществляют подъем автомобиля синхронно при одинаковой частоте

вращения электродвигателей с погрешностью изготовления винтовых стоек, которой можно пренебречь. Следовательно, неравномерность подъема может возникнуть только из-за различной частоты вращения электродвигателей привода. В соответствии с известными теоретическими положениями, частота вращения трехфазного асинхронного двигателя зависит только от его конструкции, (количества полюсов) и от частоты переменного тока, питающего двигатель. На практике же частота вращения в некоторой степени зависит также от величины напряжения питания электродвигателя и от нагрузки на валу двигателя. Зависимость от этих параметров нелинейная и для каждого двигателя индивидуальная. Нагрузка на вал электродвигателя изменяется при каждом подъеме автомобиля и определяется его положением и распределением его массы. Вследствие вышеперечисленного может возникнуть ситуация при которой стойки автомобильного подъемника будут двигаться неравномерно, что может привести к недопустимому перекосу стоек автомобильного подъемника с последующей деформацией кузова подымаемого автомобиля, или даже опрокидыванием последнего. Для того чтобы избежать этого, необходимо на ранней стадии обнаружить перекос, остановить систему и сообщить оператору о возможной опасности.

3.2. Техническое задание на разработку устройства контроля состояния автомобильного подъемника

3.2.1. Наименование работы и основания для выполнения работы

3.2.1.1. Проводимой по настоящему техническому заданию работе (ОКР) присваивается наименование: «Устройство контроля состояния автомобильного подъемника».

3.2.1.2. Настоящая работа выполняется на основе разработки пилотного проекта.

3.2.2. Цель выполнения работы. Наименование и назначение образца

3.2.2.1. Целью выполнения работы является создание (разработка) в соответствии с требованиями настоящего Т.З. «устройства контроля».

3.2.2.2. Разрабатываемому устройству присваивается наименование «Устройство контроля состояния автомобильного подъемника».

3.2.2.3. Разрабатываемое «Устройство контроля состояния автомобильного подъемника» предназначено для детектирования и сигнализации перекоса опорных стоек автомобильного подъемника.

3.2.3. Тактико-технические требования к устройству и программному обеспечению

3.2.3.1. Состав образца

3.2.3.1.1. Разрабатываемое устройство должно содержать в своем составе:
- устройство;
- источник питания;
- комплект датчиков вращения.

3.2.3.1.2. Комплект документации должен содержать:
- Техническое условие на «Устройство контроля состояния автомобильного подъемника»;
- Техническое описание изделия;
- Инструкция по монтажу устройства;
- Комплект конструктивной документации в соответствии с ЕСКД;
- Комплект ПО в соответствии с ЕСПД.

3.2.3.2. Требования по назначению

3.2.3.2.1. Устройство должно выполнять следующие функции:
- детектировать перекос опорных стоек автоподъемника;
- подача звукового, светового сигналов и останов подъемника при детектировании перекоса;
- возможность продолжения движения даже при обнаружении перекоса;
- прекращение движения при срабатывании датчиков конечного положения;
- работа в режиме двух стоек и в режиме четырех стоек.

3.2.3.2.2. Устройство должно принимать и обрабатывать информацию по нескольким каналам с параметрами сигналов:
- канал1 - цифровой, датчики вращения;
- канал2 - цифровой, датчики конечного положения;
- канал3 - цифровой, управляющие кнопки;
- канал4 - цифровой, управление электромагнитным реле.

3.2.3.2.3. Устройство должно формировать управляющие

сигналы на исполнительные механизмы по (количество каналов) со следующими характеристиками:
  – выходное напряжение (5В);
  – максимальные выходные токи (10мА).

3.2.3.2.4. Предельная масса изделия ≤ 0,5кг.

3.2.3.2.5. Предельные габариты изделия ≤ 120x120x120мм.

3.2.3.2.6. Предельная потребляемая мощность ≤ 15Вт.

3.2.3.3. Требования к электропитанию, радиоэлектронной защите и помехоустойчивости

3.2.3.3.1. Устройство должно запитываться от сети переменного тока 220В +/- 20% 50Гц.

3.2.3.3.2. Резервирование электропитания не предусматривается.

3.2.3.3.3. Информационные цепи различного назначения и цепи электропитания должны быть разнесены на разные разъемы.

3.2.3.3.4. Управляющие сигналы и датчики конечного положения должны иметь гальваническую развязку.

3.2.3.3.5. Устройство не должно создавать помех, нарушающих нормальную работу смежной.

3.2.3.4. Требования по живучести и стойкости к внешним воздействиям

3.2.3.4.1. Разрабатываемое устройство должно нормально функционировать в следующих условиях:
  – Диапазон рабочих температур 0 - +50℃.
  – Влажность 90% при температуре +20℃.

3.2.3.5. Требования по надежности

3.2.3.5.1. Срок эксплуатации устройства не менее 5 лет.

3.2.3.5.2. Вероятность безотказной работы в течение срока эксплуатации – не менее 0,9.

3.2.3.6. Требования по эргономике и технической эстетике

3.2.3.6.1. Органы управления устройства должны предусматривать защиту от случайной активации.

3.2.3.7. Требования по эксплуатации, удобству технического обслуживания, ремонта и хранения

3.2.3.7.1. Устройство должно нормально функционировать и сохранять свои характеристики на протяжении срока эксплуатации в условиях, соответствующих требованиям пунктов настоящего Т.3.

3.2.3.7.2. При проведении испытаний устройства не должно требоваться другой аппаратуры помимо контрольно-проверочной аппаратуры.

3.2.3.7.3. Подготовка устройства к эксплуатации требует проведения дополнительных монтажных и регулировочных работ.

3.2.3.7.4. Устройство должно допускать эксплуатацию неквалифицированным пользователем.

3.2.3.7.5. Комплект эксплуатационной документации должен содержать:
- Техническое описание устройства;
- Инструкцию по эксплуатации;
- Инструкцию по проверке.

3.2.3.7.6. Ремонт устройства должен осуществляться на сервисных центрах подготовленными специалистами.

3.2.3.7.7. Срок гарантии должен быть не менее трех лет.

3.2.3.7.8. Устройство должно допускать хранение в условиях отапливаемых складских помещений в штатной упаковке не менее десяти лет. Штатная упаковка пылевлагонепроницаемая.

3.2.3.8. Требования по транспортабельности

3.2.3.8.1. Устройство в штатной упаковке должно допускать транспортирование при температуре окружающей среды -20 - +50°С следующими видами транспорта:
- железнодорожным – на любые расстояния, со скоростями, принятыми для этого вида транспорта в специальной упаковке.
- воздушным – на любые расстояния с любыми скоростями в специальной упаковке.
- автомобильным – на любые расстояния, с любыми скоростями, по шоссейным дорогам, в специальной упаковке.

Примечание. Штатная упаковка – устройства пылевлагонепроницаемая и противоударная.

3.2.3.9. Требования по безопасности

3.2.3.9.1. Устройство должно отвечать ГОСТ (ГОСТ 12.1.004-91, ГОСТ 12.1.044-81) на пожаробезопасность.
3.2.3.9.2. Устройство должно отвечать требованиям ГОСТ (ГОСТ 12.2.007.0.-75) на электробезопасность.
3.2.3.9.3. Устройство должно быть травмобезопасным.
3.2.3.9.4. Устройство и его составные части не должны содержать токсичных материалов, выделяющих вредные газы и другие вещества при горении и при штатной работе.

3.2.3.10. Требования по стандартизации и унификации

3.2.3.10.1. Разработка устройства должна вестись с учетом технически и экономически обоснованных унификации, стандартизации и взаимозаменяемости деталей, узлов, блоков.
3.2.3.10.2. Подбор параметров комплектующих изделий (отбор изделий) при сборке устройства не допускается.

3.2.3.11. Требование по технологичности

3.2.3.11.1. Конструкция компонентов устройства и КПА должна быть технологичной.
3.2.3.11.2. Производство устройства – единичное.
3.2.3.11.3. Схемотехническая реализация устройства должна исключать ручную регулировку.
3.2.3.11.4. Конструкция узлов, кабелей устройства должны обеспечивать взаимозаменяемость.

3.2.3.12. Конструктивные требования

3.2.3.12.1. Масса устройства – не более 0.5 кг.
3.2.3.12.2. Габаритные размеры – не более 120x120x120мм.
3.2.3.12.3. Потребляемая мощность не более 15Вт.
3.2.3.12.4. Установочные размеры 120x120x120мм.
3.2.3.12.5. Конструкция блоков должна обеспечивать удобный доступ к разъемам, удобство их стыковки и расстыковки с применением стандартного инструмента.
3.2.3.12.6. Кабели, обеспечивающие межблочные соедине-

ния устройства, должны входить в комплект поставки.

3.2.3.12.7. Материалы и способы покрытия внешних поверхностей определяются разработчиком и согласовываются с заказчиком.

3.2.3.12.8. Блоки устройства, требующие защиты от эксплуатационных факторов, должны быть снабжены защитными крышками.

3.2.3.13. Требование к электрическим цепям

3.2.3.13.1. Устройство должно нормально функционировать при питании от источника переменного тока 100-240В, ~500мА, 50-60Гц.

3.2.3.13.2. Устройство не должно отказывать при аварийном включении/выключении источника питания.

3.2.3.13.3. После подачи питания или включения устройства оно автоматически должно приводиться в исходное состояние.

3.2.3.13.4. В устройстве должны быть предусмотрены меры, исключающие возникновение короткого замыкания, реализованы цепи обеспечивающие отключение устройства от источника питания в случае возникновения короткого замыкания.

3.2.3.13.5. В устройстве должны быть предусмотрены меры исключающие включение устройства при неправильной подстыковке блоков и кабелей.

3.2.3.13.6. Коэффициент нагрузки всех входящих в устройство элементов должен быть не более 50%.

3.2.3.13.7. В устройстве должна быть предусмотрена гальваническая развязка цепей источников сигналов.

3.2.3.13.8. Должны быть предусмотрены меры по защите устройства от статического электричества.

3.2.3.13.9. Реализация цепей включения электропитания должна быть согласована с Заказчиком.

3.2.3.13.10. Перечень команд управления должен быть согласован с Заказчиком.

3.2.3.14. Требования к программному обеспечению

3.2.3.14.1. Рабочая программа для управляющего микроконтроллера должна быть написана на языке ассемблер для ST7 Microelectronics в среде разработки SofTec STVD7 .

3.2.3.14.2. Максимальный объем, занимаемый программой

в памяти устройства, не должен не превышать 8Кб.

3.2.3.14.3. Программа для управляющего МК должна разрабатываться и отлаживаться на комплексе разработки SofTec STVD7 SofTec Microsystems Toolset.

3.2.3.14.4. Наборы входных данных для тестирования программы должны согласовываться с Заказчиком.

3.2.3.14.5. При поставке устройства программа работы должны быть на внутреннем ПЗУ микроконтроллер с установленными битами защиты.

3.2.4. Требования к материалам и комплектующим изделиям

3.2.4.1. Устройство должно разрабатываться на основе использования отечественных и зарубежных материалов и комплектующих.

3.2.4.2. Рекомендуется применение элементной базы с ресурсом менее 5 лет.

3.2.5. Этапы выполнения работы

3.2.5.1. Разработка, экспериментальная отработка и изготовление устройства должны соответствовать требованиям ЕСКД и ГОСТ (ГОСТ 2.004-88).

3.2.5.2. Разработка устройства должна проводиться по следующим стадиям:

- эскизное проектирование.
- разработка рабочей документации на опытный образец.
- изготовление опытного образца.
- автономное испытание опытного образца.
- Корректировка рабочей документации по результатам испытаний.
- изготовление и поставка опытных образцов устройства для комплексных испытаний.
- комплексные испытания устройства.
- корректировка рабочей документации по результатам испытаний.
- изготовление и поставка первого штатного образца изделия.

3.3. Разработка структурной схемы и алгоритма функционирования устройства контроля состояния автомобильного подъемника. Выбор типов датчиков и исполнительных устройств

Основное назначение разрабатываемого устройства – детектирование перекоса опорных стоек автомобильного подъемника, аварийная остановка подъемника и сообщение оператору об аварийной ситуации световой и звуковой сигнализацией. Разрабатываемое устройство представляет собой электронную систему контроля, которая, как и большинство систем контроля, является системой замкнутого типа. Она должна включать следующие элементы:

- элемент аварийного управления – электромеханическое реле управляющее пускателем;
- датчики вращения;
- датчики концевых положений стоек;
- управляющее устройство (УУ).

Принцип действия устройства контроля состояния автомобильного подъемника заключается в определении направления перемещения стоек автомобильного подъемника, подсчете количества оборотов стоек, полученных с датчиков вращения, подключенных к винтовому валу подъемника, определении разницы в ходе каждой стойки подъемника и аварийной остановке подъемника при превышении максимально допустимой разницы в движении одной или нескольких стоек.

Допустимый перекос определяется разницей в количестве оборотов между различными стойками подъемника.

После определения перекоса необходимо сформировать управляющее воздействие на автоподъемник для аварийной остановки и на устройства световой и акустической сигнализации.

Исходя из вышесказанного, можно предложить структуру системы контроля автомобильного подъемника, изображенную на рис. 3.2.

Структурная схема устройства контроля состояния автомобильного подъемника содержит следующие блоки:

- блок питания системы, состоящий из понижающего трансформатора 220В на 28В (входит в комплект стандартного автомобильного подъемника), двухполупериодного выпрямителя, стабилизатора напряжения (на +12В), предназначенного для питания управляющей обмотки электромагнитного реле, стабилизатора напряжения (на +5В) предназначенного для питания всех остальных блоков системы контроля;
- датчики вращения, формирующие информацию о направлении перемещения стоек автомобильного подъемника и о величине перемещения;
- датчики концевых положений стоек автомобильного подъемника;

- триггеры Шмитта, предназначенные для формирования сигналов ТТЛ уровня, приходящих с датчиков вращения и датчиков концевых положений стоек подъемника на входы микроконтроллера;

- устройство управления, выполненное на однокристальном микроконтроллере ST7FLITE29;

- кнопку сброс аварийного состояния;

- дешифратор индикации аварийной остановки, предназначенной для указания стойки подъемника, совершившей наименьшее количество оборотов;

- блок индикации, состоящий из двух светодиодов, индицирующих состояние каждой стойки автомобильного подъемника;

- блок усилителя акустической сигнализации аварийной остановки автомобильного подъемника;

- блок акустической сигнализации аварийной остановки автомобильного подъемника;

- блок усилителя управления электромеханическим реле цепи аварийной защиты автомобильного подъемника;

- электромеханическое реле цепи аварийной защиты автомобильного подъемника.

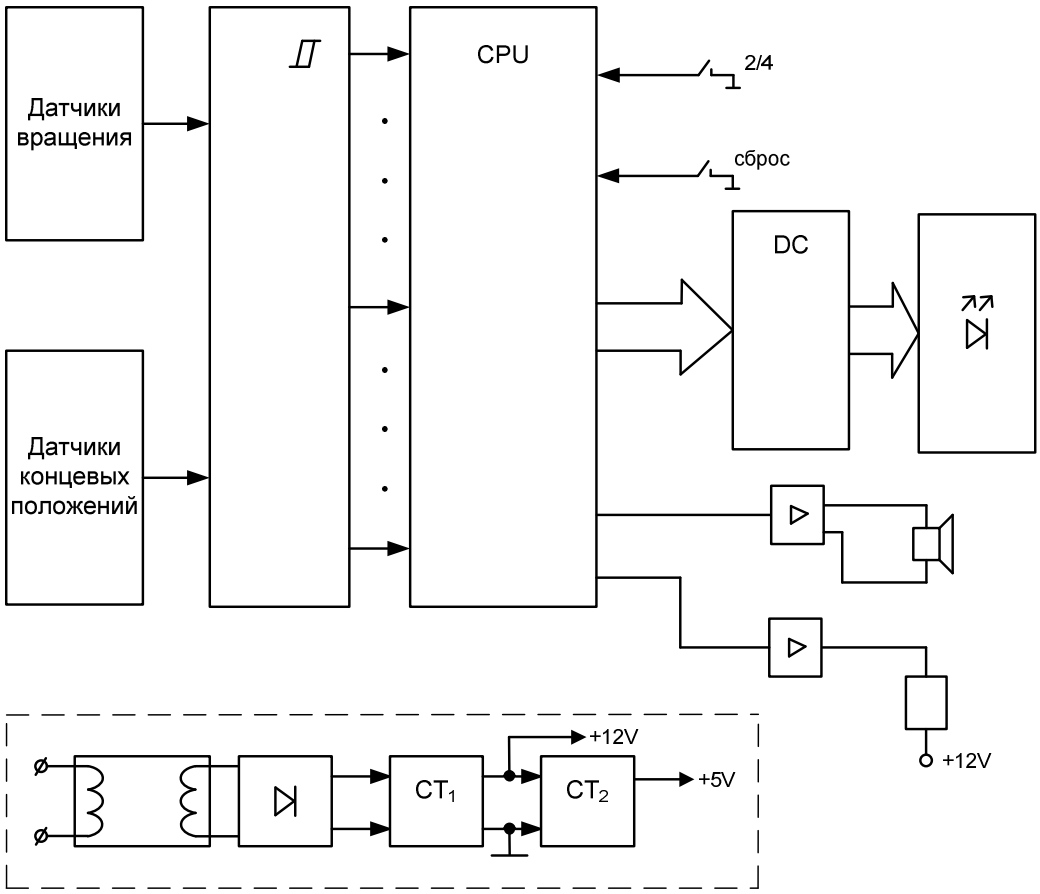


Рис. 3.2. Структура устройства контроля состояния автомобильного подъемника

В качестве датчиков вращения предполагается использовать сдвоенный оптический датчик с открытым оптическим каналом, который позволяет определить направление вращения стоек подъемника и позволяет подсчитывать количество оборотов стойки.

В качестве датчиков концевого положения стоек автомобильного подъемника используются датчики типа “сухой контакт”, через которые запитываются управляющие обмотки магнитных пускателей электроприводов стоек подъемника. Для определения состояния этих датчиков целесообразно использовать оптроны, которые обеспечат гальваническую развязку между управляющей схемой и силовой частью системы управления подъемника.

Для индикации аварийной ситуации целесообразно использовать дискретные светодиоды красного цвета и динамик для звуковой сигнализации.

Для управления цепью аварийной защиты подъемника необходимо применить электромеханическое реле, обеспечивающее как гальваническую развязку между управляющей схемой и силовой частью системы управления подъемника, так и необходимый коммутируемый ток управления автомобильным подъемником.

На основе полученной структуры устройства был разработан алгоритм управления (рис. 3.3), который включает в себя конфигурировавния системы после включения ее оператором и укрупненную схему функционирования.

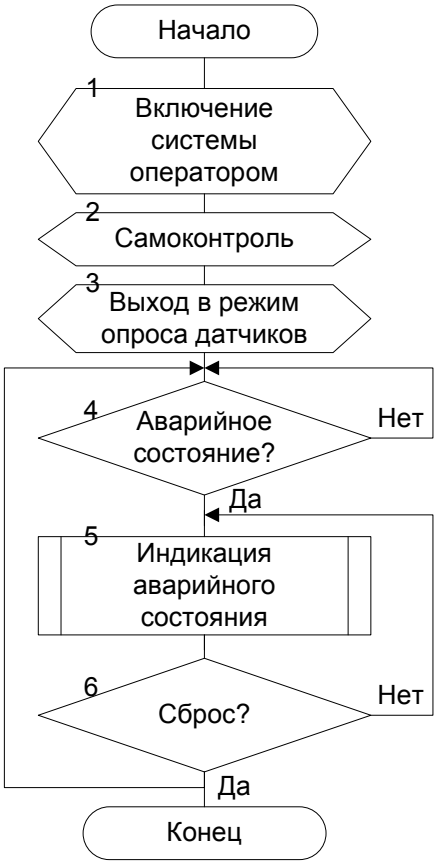


Рис. 3.3. Алгоритм управления

3.4. Разработка функциональной схемы устройства контроля состояния автомобильного подъемника.

Для питания устройства контроля состояния автомобильного подъемника используем имеющийся в системе блока управления подъемника понижающий трансформатор с напряжением на вторичной обмотке 28В 50Гц. После выпрямления двухполупериодным выпрямителем переменного тока с вторичной обмотки трансформатора и сглаживания пульсаций напряжения электролитическим конденсатором фильтра получим постоянное напряжение амплитудой +40В. Из полученного напряжения необходимо получить стабилизированное напряжение +12В, необходимое для питания управляющей обмотки защитного электромеханического реле, и стабилизированное напряжение +5В, необходимое для питания всех остальных блоков системы контроля.

В связи с тем, что вся система контроля обладает невысоким током потребления, а к точности и стабильности питающего напряжения не предъявляются высокие требования, целесообразно в качестве стабилизаторов напряжения применить недорогие интегральные линейные стабилизаторы с фиксированным выходным напряжением +12В и +5В соответственно. К сожалению, вышеупомянутые стабилизаторы не допускают входного напряжения выше, чем +30В и +20В соответственно. Поэтому следует поставить входной стабилизатор, формирующий напряжение +24В (допускающий входное напряжение +45В), с выхода которого запитать стабилизатор на напряжение +12В, а стабилизатор формирующий напряжение +5В, запитать с выхода стабилизатора на +12В, как показано на схеме, изображенной на рис.3.4.

Выход датчика вращения с открытым оптическим каналом необходимо подключить ко входам микроконтроллера. Подключать необходимо через триггер Шмитта, который выполняет функцию подавления “дребезга контактов”, вызываемого механической вибрацией датчиков и стоек подъемника, и формирования выходного сигнала с датчика вращения до уровня ТТЛ.

Датчики концевых положений стоек подъемника подключаются через оптрон светодиод – фототранзистор с закрытым оптическим каналом. Оптрон обеспечивает гальваническую развязку между концевыми выключателями, коммутирующими переменное напряжение 220В и управляющей электроникой системы контроля. Переменное напряжение 220В, подаваемое на светодиод оптрона, гасится на пленочном конденсаторе 10нF х 400В. Светодиод оптрона от обратного напряжения защищается кремниевым диодом, включенным, встречно-параллельно. Выходы оптронов соединены по схеме монтажного И. Выходной сигнал с оптронов интегрируется на конденсаторе и поступает на вход триггера Шмитта, который служит для формирования сигнала, совместимого с уровнями ТТЛ для последующей подачи на вход микроконтроллера.

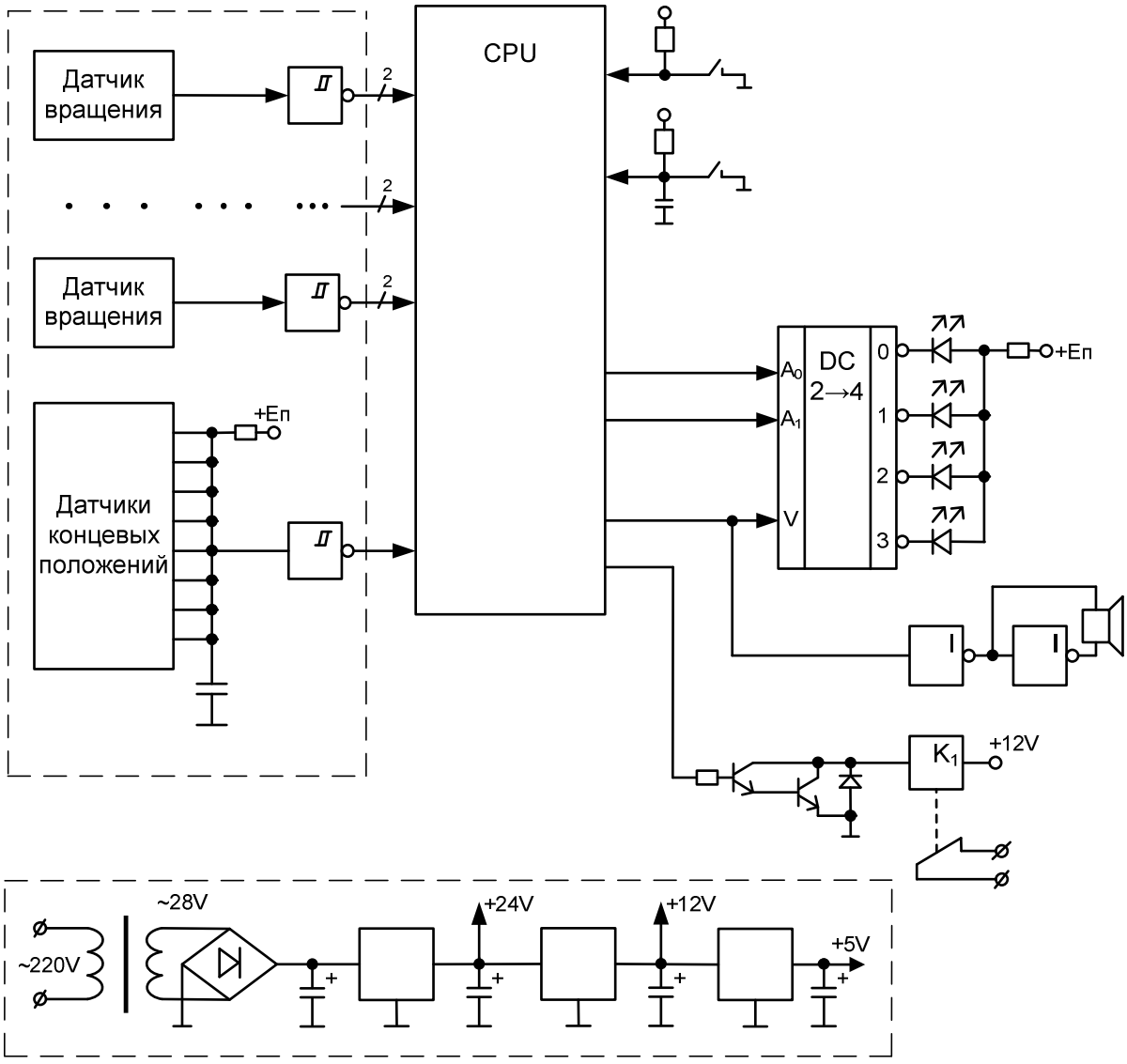


Рис. 3.4. Функциональная схема устройства контроля состояния автомобильного подъемника

Кнопка сброса через интегрирующую RC цепочку подключается непосредственно к входу микроконтроллера. Нормально разомкнутая кнопка работает на замыкание. Опрос состояния кнопки осуществляется программно.

Индикация стойки подъемника, совершившей наименьшее количество оборотов, осуществляется на светодиодном индикаторе выполненном из дискретных элементов, подключенных через дешифратор “2 на 4” к микроконтроллеру. Два выходных сигнала микроконтроллера в двоичном коде указывают номер стойки, совершившей наименьшее количество оборотов, а третий выход микроконтроллера включает световую и звуковую сигнализацию аварийной ситуации на подъемнике. Для звуковой сигнализации используется дополнительный усилитель, построенный на логических элементах НЕ.

Управляющая обмотка электромеханического реле аварийного отключения автомобильного подъемника через усилитель, выполненный на составном транзисторе, включенном по схеме Дарлингтона, подключена к выходу микроконтроллера. Выходной транзистор усилителя от выброса напряжения обратной полярности защищается обратно включенным диодом параллельно переходу К-Э силового транзистора.

Реле аварийной защиты автомобильного подъемника включается при включении питания системы после успешного прохождения самодиагностики и отключается в случае возникновения аварийной ситуации. Повторное включение реле возможно после подачи сигнала сброс оператором путем нажатия и отпускания кнопки сброс на пульте управления.

3.5. Выбор элементной базы и разработка схемы электрической принципиальной устройства контроля состояния автомобильного подъемника.

Функционально устройство контроля состояния автомобильного подъемника состоит из четырех функциональных блоков: блока питания, датчиков вращения стоек подъемника, датчиков концевых положений подъемника и блока управления. Общий вид расположения всех блоков на печатной плате показан на рис. 3.5.

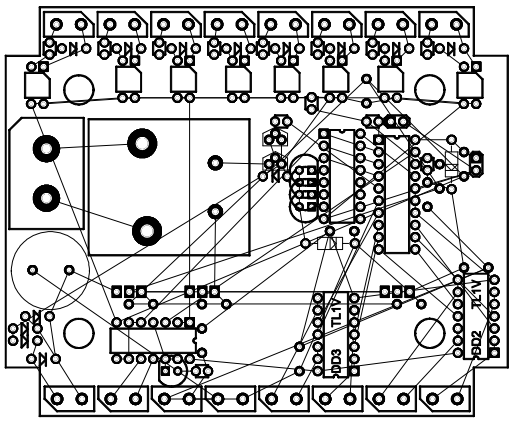


Рис. 3.5. Расположения элементов на печатной плате

Блок питания имеет в качестве входного напряжения переменное напряжение 28В, 50Гц, используемое в схеме управления автомобильного подъемника. После выпрямления и сглаживания на конденсаторе фильтра получим постоянное напряжение около 40В. Интегральные стабилизаторы серии LM7805 LM7812 не допускают входного напряжения более 35В. Поэтому необходимо использовать дополнительный стабилизатор LM7824 для того, чтобы понизить входное напряжение до 24В. Все три стабилизатора включаем последовательно. Схема блока питания представлена на рис. 3.6.

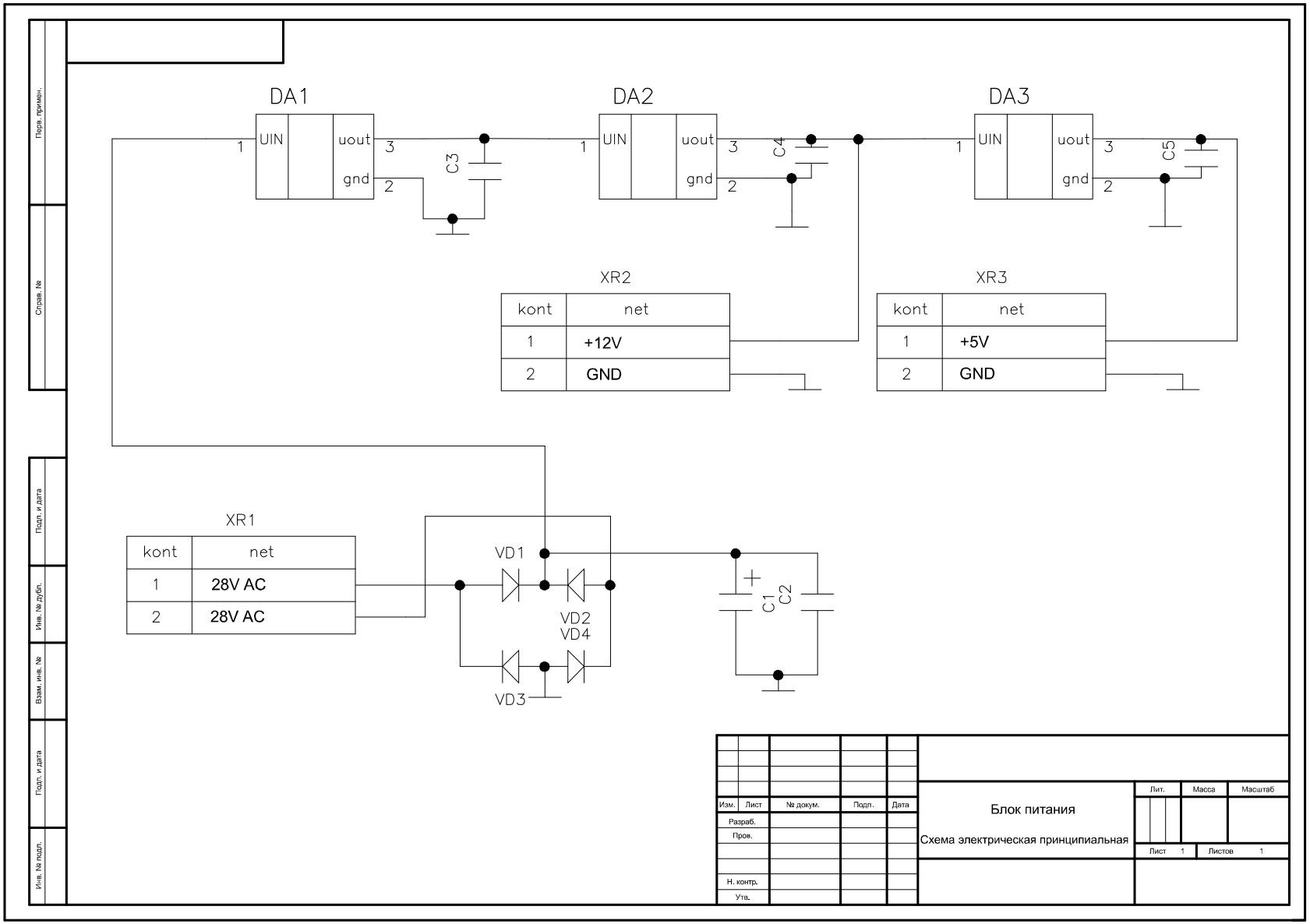


Рис. 3.6. Схема электрическая принципиальная блока питания

Датчик вращения должен формировать сигналы на блок управления, по которым можно определить направление движения стоек подъемника и величину перемещения стоек подъемника. В качестве датчика целесообразно использовать инфракрасный датчик с открытым каналом. Для определения направления движения стоек необходимо установить на каждую стойку по два датчика со смещением. Светодиоды датчиков вращения включены в цепь питания последовательно через резистор, обеспечивающий ток на уровне 20 мА. Формирование выходного сигнала осуществляется на триггере Шмитта, (микросхема К1561ТЛ1). Схему датчика вращения представлена на рис. 3.7.

В качестве датчика первичной информации был выбран инфракрасный датчик движения фирмы HoneyWell, представляющий собой разнесенную инфракрасную пару светодиод – фототранзистор с открытым каналом. Расстояние между излучателем и приемником составляет около 5 мм. Этого зазора будет вполне достаточно что бы через него проходила непрозрачная пластина укрепленная на вращающейся стойке автомобильного подъемника. Для повышения точности контроля синхронности подъема автомобиля различными стойками, необходимо установить на каждой стойке по четыре пластины под углом 90 градусов.

Направление движения подъемника, подъем или опускание, можно определять установив два инфракрасных датчика со смещением 5 - 10 мм, по ходу движения, перекрывающей оптический канал датчиков, непрозрачной пластины. Направление движения стойки подъемника определяется по разности фаз импульсов формируемых датчиками движения.

Учитывая невысокую скорость движения подъемника все сигналы, формируемые датчиками движения можно опрашивать программно. Частота сигналов формируемых датчиком движения при движении автомобильного подъемника не превышает в нашем случае 40Гц. При этом скорость движения подъемника будет составлять 0,1 м/сек.

Триггер Шмитта необходим для формирования выходного сигнала с крутым фронтом и спадом импульса, а также для подавления эффекта “дребезга контактов” который может возникнуть с течением времени при эксплуатации устройства вследствии загрязнения открытого канала инфракрасного датчика.

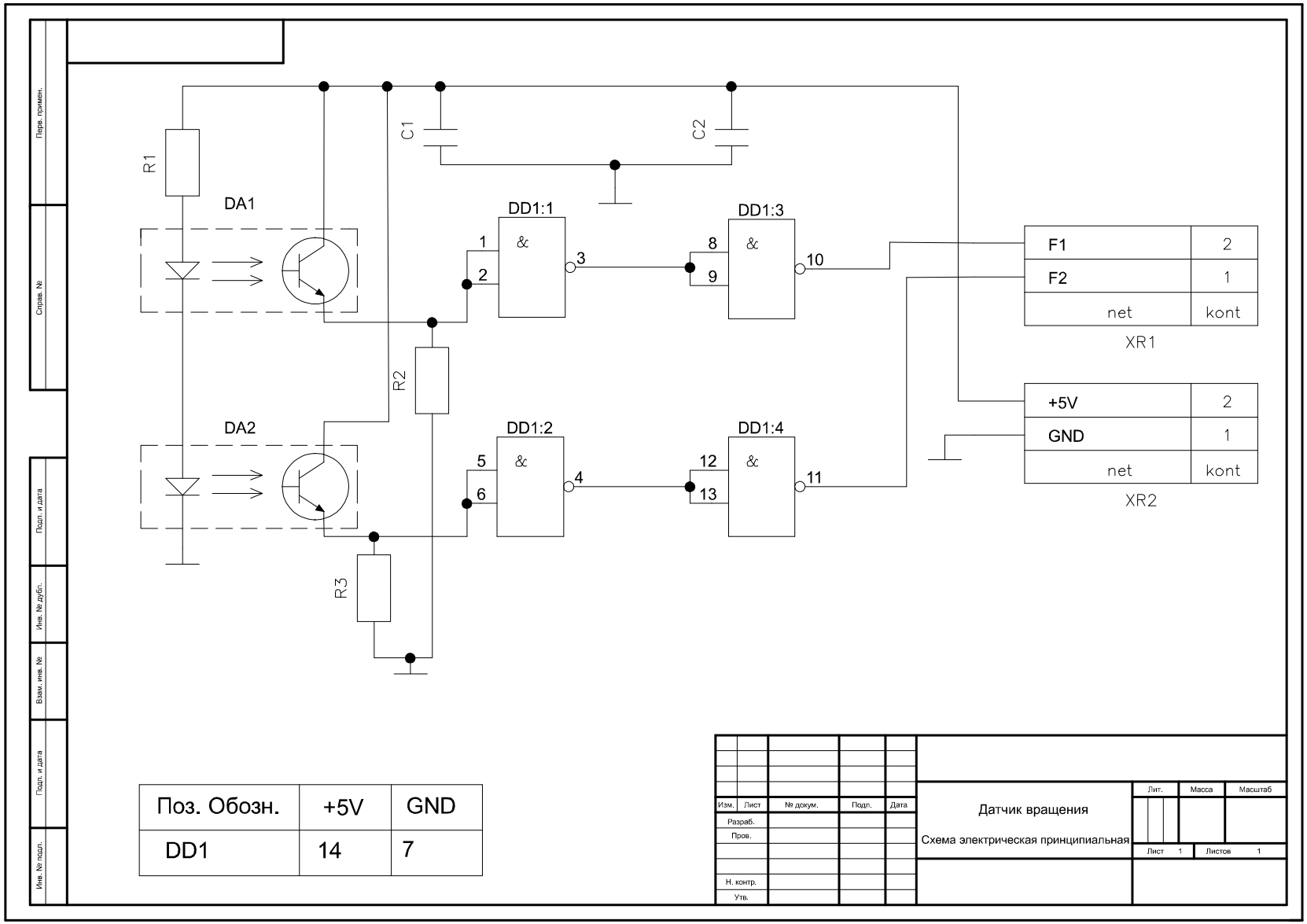


Рис. 3.7. Электрическая принципиальная датчика вращения

Концевики представляют собой пару механических контактов, работающих на размыкание. На разомкнутых контактах образуется переменное напряжение 220В (50 Гц). Датчик концевых положений автомобильного подъемника представляет собой электрическую цепь, состоящую из гасящего конденсатора и светодиода закрытой оптопары. Параллельно светодиоду оптопары включен защитный диод, предназначенный для ограничения обратного напряжения на светодиоде. Выходы оптопар всех датчиков концевых положений соединены параллельно, образуя схему монтажного И. Схема датчика концевых положений представлена на рис. 3.8.

В качестве оптопары была выбрана недорогая оптопара L817 выполненная в корпусе DIP4, состоящая из пары: светодиод- фототранзистор, обеспечивающая гальваническую развязку 500В между входом и выходом и способная формировать выходной импульс со скоростью наростания – спада менее 1 мс. Соответственно задержка срабатывания датчика концевого положения обусловленная временем срабатывания оптопары не превысит 1 мс. Учитывая то что, датчик концевого положения представляет собой пару механических контактов через которые протекает переменный ток частотой 50 Гц необходимый для удерживания контактов магнитного пускателя, следовательно, возможная задержка сигнала обусловленная работой системы может составлять время не менее 10 мс (пол периода) и небольшая дополнительная задержка в оптроне на работе системы сказаться не сможет.

Гальваническая развязка обеспечиваемая оптроном необходима для защиты блока управления, системы контроля автомобильного подъемника от высокого напряжения управления магнитными пускателями коммутирующими силовые приводы.

Выход транзисторов всех оптронов включен параллельно образуя схему монтажного И. Все транзисторы при срабатывании соответствующего датчика конечного положения разряжают конденсатор интегрирующей RC-цепочки. В результате этого, на входе микроконтроллера будет сформирован сигнал извещающий его о том, что, одна из стоек достигла конечного положения и движение подъемника прекращено. При этом фактическая разность движения стоек принимается равной нулю.

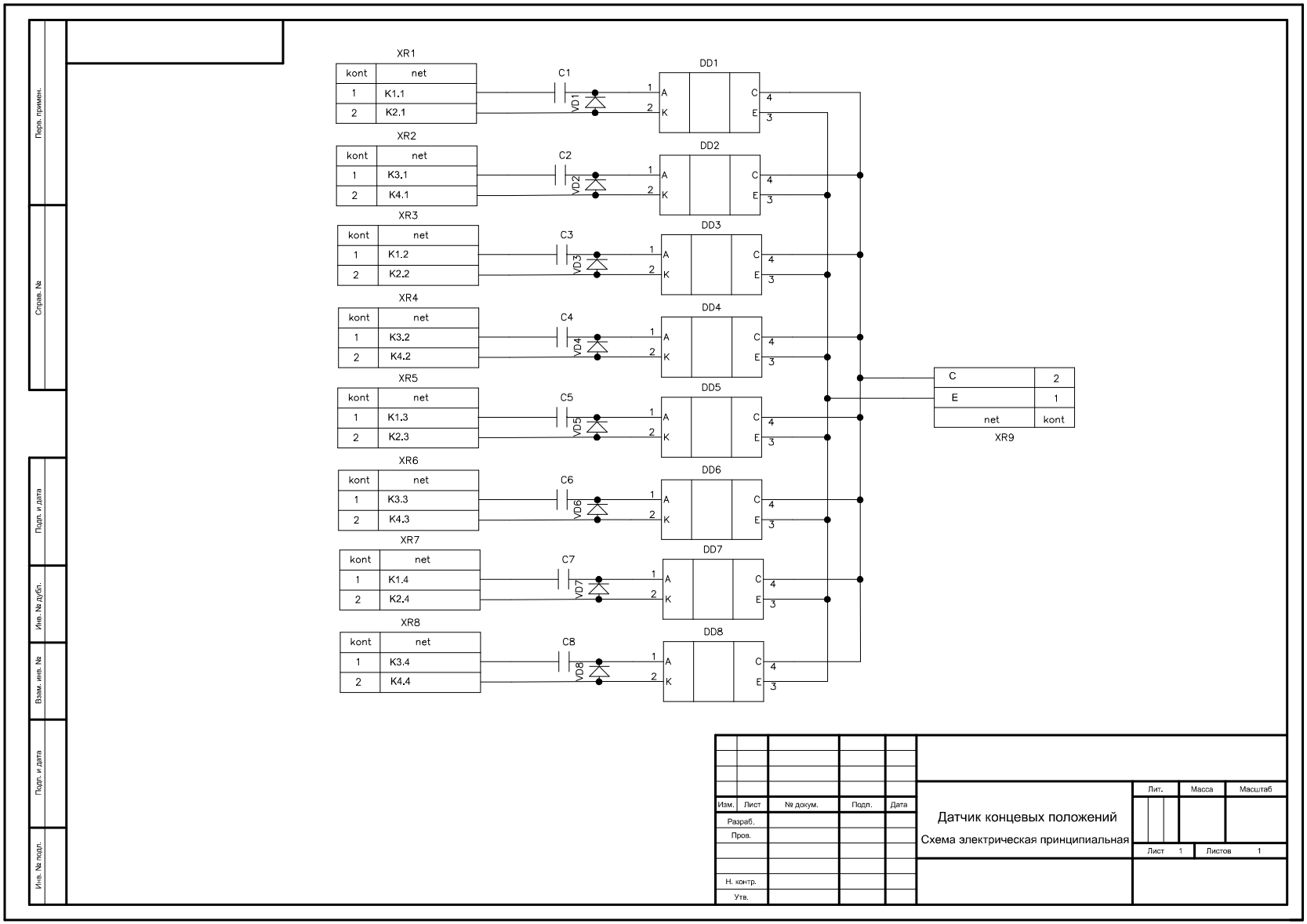


Рис. 3.8. Принципиальная схема датчика концевых положений

Блок управления принимает информацию от датчиков вращения по двум линиям связи для каждой стойки. Следовательно, с учетом четырех стоек мы имеем 8 информационных линий, которые через формирователи, выполненные на триггерах Шмитта, необходимо подключить к программно опрашиваемым выводам микроконтроллера. Сигнал с датчиков концевых положений поступает на интегрирующую RC цепочку и через формирователь, выполненный на триггере Шмитта, подается на программно опрашиваемый вывод микроконтроллера.

Информационный сигнал с кнопки сброса через интегрирующую RC цепочку подается на программно опрашиваемый вывод микроконтроллера. Сигнал переключателя определяющего тип подъемника, подается непосредственно на программно опрашиваемый вывод микроконтроллера. Микроконтроллер подает сигнал включения подъемника со своего выхода на усилитель, выполненный на транзисторах для включения электромеханического реле.

Еще три выхода микроконтроллера используются для световой и звуковой сигнализации аварийного состояния подъемника. Световая сигнализация выполнена на дешифраторе и четырех светодиодах. Звуковая сигнализация выполнена на пьезоэлектрическом излучателе который подключается к мостовому усилителю выполненному на логических элементах. Для этого используются три триггера Шмитта. Схема блока управления положений показана на рис. 3.9.

Перечень элементов всех блоков устройства контроля автомобильного подъемника: блока питания, датчика вращения, датчиков концевых положений и блока управления представлены в таблицах 3.1 – 3.4 соответственно.

Информация, приходящая на блок управления от датчиков вращения, поступает с частотой не более 40 Гц, что дает возможность опрашивать эти сигналы программно. Разность фаз сигналов, формируемых датчиками вращения, зависит от скорости вращения стоек и расстояния между оптопарами и составляет в нашем случае время около 0,5 мс, при максимальной скорости вращения, что также делает возможным программный опрос всех датчиков.

При нормальной работе подъемника осуществлятся программный опрос датчиков движения и вычисление разности, а также опрос датчиков концевых положений подъемника. При аварийной остановке подъемника производится только программный опрос кнопки сброс.

На рабочей тактовой частоте микроконтроллер за минимальное время 0,5 мс может выполнить около 1000 команд, что вполне достаточно для опроса всех датчиков и вычисления разности в движении стоек автомобильного подъемника.

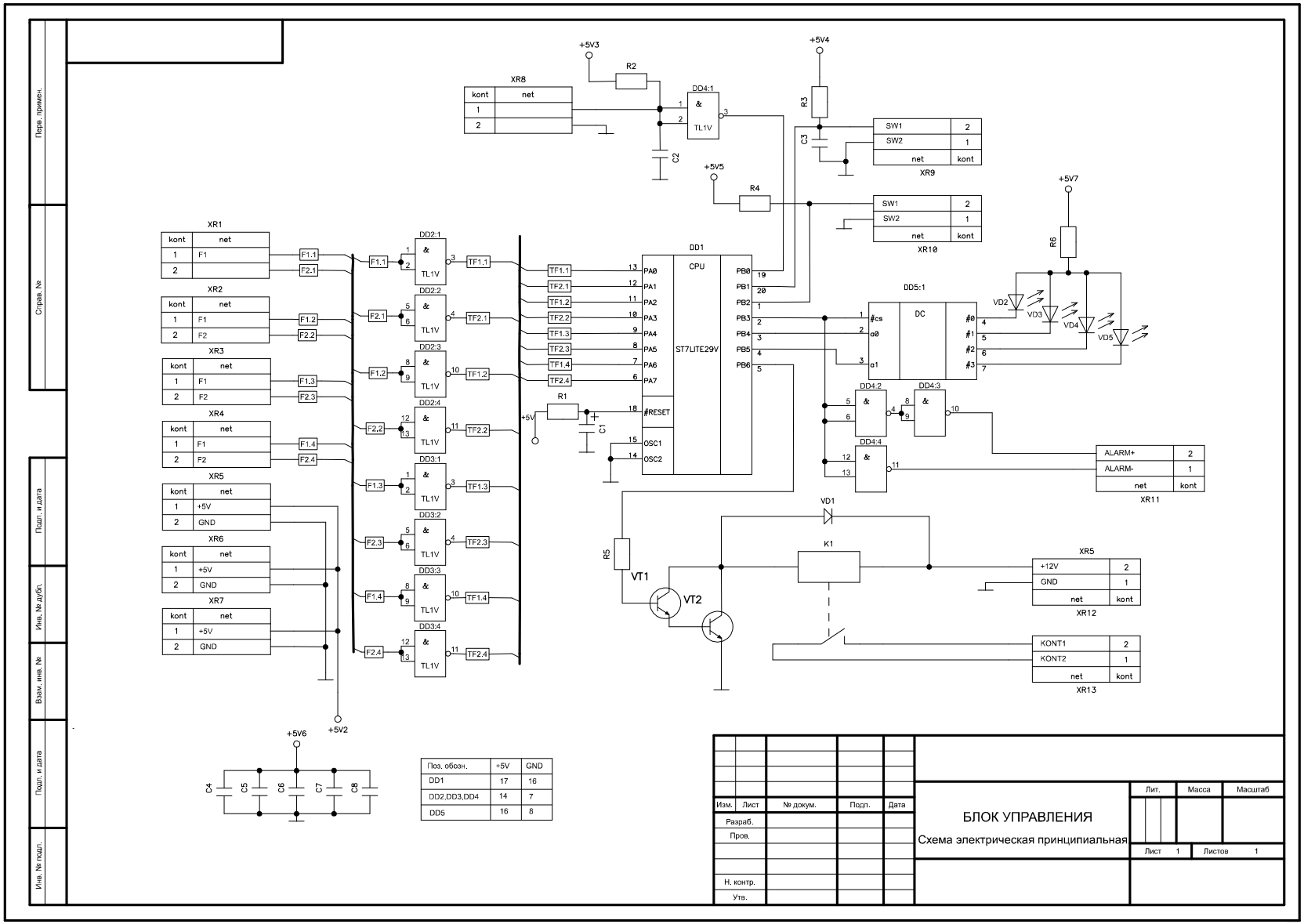


Рис. 3.9. Принципиальная схема блока управления

Таблица 3.1

Поз. обозначение	Наименование	Кол.	Примечание
	Конденсаторы		
C1	К50-35 2200 Мкф +20% -40% 50В	1	
C2-C5	SMD 0805 0.1 Мкф +20% -40% 50В	4	
	Диоды		
VD1-VD4	IRL205	4	
	Микросхемы		
DA1	LM7824	1	
DA2	LM7812	1	
DA3	LM7805	1	
	Разъемы		
XR1-XR3	TB5-2	3	

						Лит.	Масса	Масштаб
Изм.	Лист	№ докум.	Подп.	Дата	Блок питания			
Разраб.								
Пров.					Перечень элементов			
						Лист 1	Листов 1	
Н. контр.								
Утв.								

Таблица 3.2

Поз. обозначение	Наименование	Кол.	Примечание
	Резисторы SMD 0805 +/- 5%		
R1	100 Ом	1	
R2,R3	4,7 кОм	2	
	Конденсаторы SMD 0805 +20% -40% 50В		
C1,C2	0.1 Мкф	2	
	Микросхемы		
DD1	К1561ТЛ1	1	
DA2,DA3	НОА 1875	2	
	Разъемы		
XR1,XR2	ТВ5-2	2	

					Датчик вращения	Лит.	Масса	Масштаб
Изм.	Лист	№ докум.	Подп.	Дата				
Разраб.					Перечень элементов			
Пров.						Лист 1	Листов 1	
Н. контр.								
Утв.								

Таблица 3.3

Поз. обозначение	Наименование	Кол.	Примечание
	Конденсаторы К10-73		
C1-C8	10 Нф 400В +20% -40%	8	
	Диоды		
VD1-VD8	КД521А	8	
	Микросхемы		
DD1-DD8	FOD817A	8	
	Разъемы		
XR1-XR8	ТВ5-2	8	

						Лит.	Масса	Масштаб
Изм.	Лист	№ докум.	Подп.	Дата	Датчик концевых положений			
Разраб.						Лист 1	Листов 1	
Пров.					Перечень элементов			
Н. контр.								
Утв.								

Таблица 3.4

Поз. обозначение	Наименование	Кол.	Примечание
	Резисторы SMD 0805 +/- 5%		
R1,R2	10 кОм	2	
R3	51 кОм	1	
R4	10 кОм	1	
R5	2 кОм	1	
R6	240 Ом	1	
	Конденсаторы		
C1	К50-35 10 Мкф +20% -40% 16В	1	
C2-C8	SMD 0805 +20% -40% 50В	7	
	Диоды		
VD1	IRL205	1	
VD2-VD5	АЛ310А	4	
	Транзисторы		
VT1,VT2	КТ3117А	2	
	Микросхемы		
DD1	ST7FLITE29	1	
DD2-DD4	К1561ТЛ1	3	
DD5	К1554 ИД14	1	
	РЕЛЕ		
К1	BS-902AS 12V	1	
	Разъемы		
XR1-XR12	ТВ5-2	12	
XR13	ТВ10-2	1	

Изм.	Лист	№ докум.	Подп.	Дата	Блок управления Перечень элементов	Лит.	Масса	Масштаб
Разраб.								
Пров.						Лист 1	Листов 1	
Н. контр.								
Утв.								

3.6. Разработка управляющей программы микроконтроллера

Разработка управляющей программы, ее трансляция и отладка в устройстве существенно зависит от используемых системных средств. Ресурсы 8-разрядного контроллера ST7 достаточны для поддержки программирования на языках высокого уровня. Это позволяет использовать все преимущества структурного программирования, разрабатывать программное обеспечение с использованием раздельно транслируемых модулей. Однако, несмотря на это, сохраняется возможность поддержки языками низкого уровня типа ассемблера. Это особенно важно при необходимости обеспечения контролируемых интервалов времени, а так же в случаях разработки управляющих программ с простым алгоритмом функционирования.

В данном случае алгоритм функционирования управляющей программы довольно прост и при этом нет необходимости в построении сложного структурного проекта с использованием библиотечных функций. Несмотря на простоту алгоритма сама по себе программа будет критична ко времени, так как ей необходимо постоянно опрашивать несколько датчиков, кнопки и состояние концевиков. Поэтому для разработки управляющей программы был выбран язык низкого уровня – ассемблер.

Для разработки управляющей программы на ассемблере используется стандартный программный комплекс. Это инструментальная среда ST7 Visual Developer, представляющая собой специализированную систему разработки и имеющая в своем составе менеджер проектов, текстовый редактор, симулятор. Она так же позволяет в автоматическом режиме создавать управляющие программы для различных микроконтроллеров семейства ST7. Выбор конкретного типа контроллера среди моделей семейства обеспечивает соответствующая опция меню конфигурации проекта. Загрузив программу в оперативную память контроллера, пользователь имеет возможность запускать ее в пошаговом или непрерывном режимах, задавать условные или безусловные точки останова, контролировать содержимое ячеек памяти и регистров.

Таким образом, используя данное средство разработки, был реализован основной алгоритм функционирования (рис. 3.10) и основная подпрограмма (рис. 3.11). Она реализует детектирования перекоса опорных стояк автомобильного подъемника, основывается на хранении и постоянном подсчете разницы количества оборотов между стойками подъемника (приложение 5).

Из временных диаграмм (рис. 3.12 и рис. 3.13) видно, что при движении вверх или вниз стоек автомобильного подъемника сигнал с каждого из датчиков приходит со смещением во времени. Это означает, что можно определить, в какую сторону происходит движение по тому, на какой порт микроконтроллера, подключенного к датчикам, пришел сигнал раньше.

Рис. 3.10. Блок схема основной подпрограммы

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Описание: Основная подпрограмма. Основной программный цикл.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
main:
    rsp                             ; сброс указателя стека
    sim                             ; установка маски прерываний
    clr   MCCSR                     ; инициализация
                                    ; микроконтроллера в
                                    ; нормальном режиме
    ld    A,              #$f0      ; формирование константы для
                                    ; настройка порта A
    ld    PADDR,          A         ; загрузка настройки порта A в
                                    ; регистр направления данных
    ld    PAOR,           A         ; загрузка настройки порта A
                                    ; в регистр
                                    ; конфигураций
    ld    A,              #$78      ; формирование константы
                                    ; для настройка порта B
    ld    PBDDR,          A         ; загрузка настройки порта B в
                                    ; регистр направления данных
    ld    PBOR,           A         ; загрузка настройки порта B в
                                    ; регистр конфигураций
    call  values_nulling            ; обнуление всех переменных
    ld    A,              #20       ; формирование константы
```

```
                                    ; максимальной
                                    ; разницы счетчиков
    ld    diff_counter,  A          ; загрузка константы
                                    ; возникновение
                                    ; аварийной ситуации
    ld    A,             #$48       ; формирование константы
                                    ; для начала
                                    ; движения подъемника
    ld    PBDR,          A          ; загрузка константы в порт B
    clr   err                       ; сброс состояния ошибки
    clr   f_oper                    ; сброс флагов операций
LBL_MAIN_LOOP:                      ; начало основного цикла
                                    ; программы
    call  watchdog_disable          ; отключение сброса от
                                    ; сторожевого таймера
    call  check_state               ; проверка состояния
                                    ; счетчиков,
                                    ; результат находится в A
    cp    A,             #0         ; если A равен 0 - подъемник
                                    ; работает нормально
    jreq  LBL_MAIN_LOOP             ; продолжаем цикл проверки
                                    ; состояния счетчиков
    call  relay_off                 ; если A не равно
                                    ; 0 - остановить подъемник
    call  indication_error          ; подача светового и звукового
                                    ; сигнала,
                                    ; пока не нажат сброс
    ld    A,             #$48       ; формирование константы
                                    ; для продолжения
                                    ; движения подъемника
    ld    PBDR,          A          ; загрузка константы в порт B
    jp    LBL_MAIN_LOOP             ; возврат на начало основного
                                    ; цикла программы
  ret ; конец подпрограммы main
```

Рис. 3.11. Управляющая программа

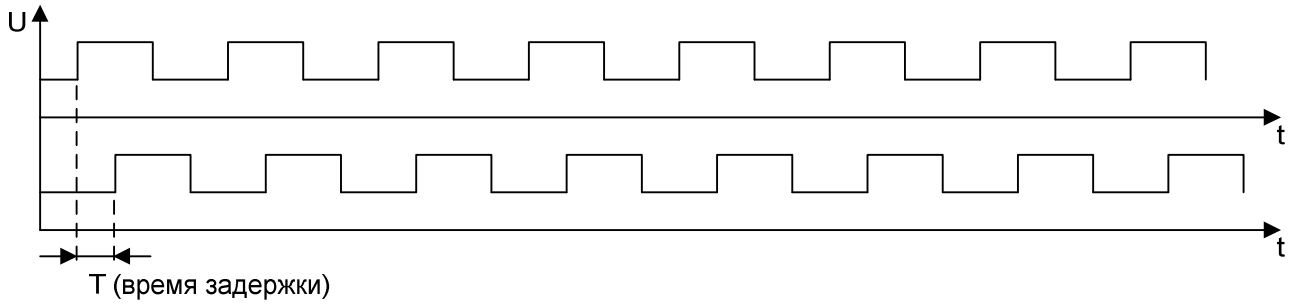


Рис. 3.12. Временные диаграммы датчика движения при подъеме

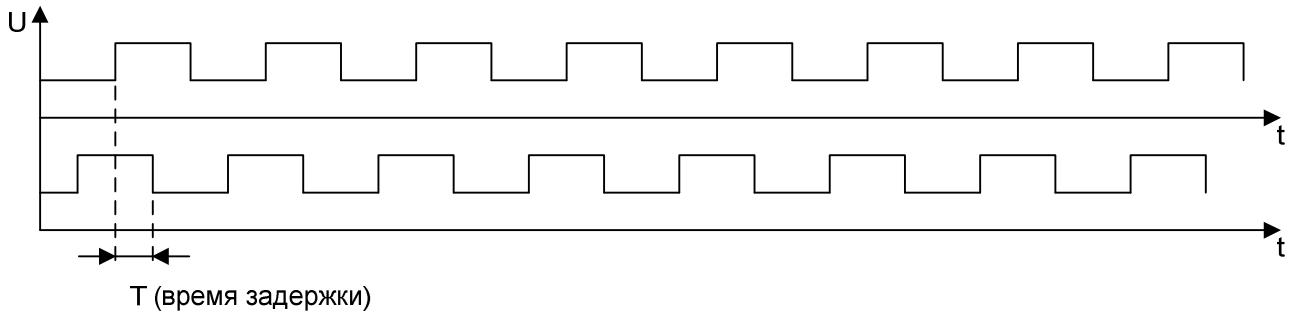


Рис. 3.13. Временные диаграммы датчика движения при опускании

Поскольку микроконтроллер, который используется в проекте, - 8-битный, количество оборотов значительно больше, чем 256, а с 16-битными словами работать не очень удобно, был разработан алгоритм, при реализации которого необходимо хранить разницу, а не количество оборотов. При этом для хранения разницы достаточно 1 байта.

Кроме того, необходимо определять направление перемещения, чтобы обнаружить перекос. Для этого нужно постоянно опрашивает датчики вращения, состояния концевых выключателей и управляющие кнопки. В случае, если сработает концевой выключатель, или разница между стойками подъемника будет недопустимой, программа должна сформировать управляющий сигнал останова на электромеханическое реле, управляющее пускателем.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. От чего зависит частота вращения асинхронного двигателя?
2. Каким образом определяется направление перемещения подъемника (по временным диаграммам)?
3. Зачем нужен стабилизатор напряжения +24V?
4. Каким образом указывается стойка которая запаздывает?
5. Для чего в схеме используется триггер Шмитта?
6. Зачем нужна оптронная развязка между концевиком и блоком управления?
7. Как можно обойтись байтовым счетчиком для подсчета количества оборотов стоек?
8. Как следует настроить порты ST7FLITE29?
9. Как определить аварийное состояние системы? Чем определяется аварийное состояние системы?
10. Что происходит с системой контроля при нажатии кнопки RESET?

РАЗДЕЛ 4. ПРОЕКТИРОВАНИЕ ДВУХКАНАЛЬНОЙ СИСТЕМЫ ТЕРМОРЕГУЛИРОВАНИЯ НА ОСНОВЕ МИКРОКОНТРОЛЛЕРА СЕРИИ ST7

4.1. Техническое задание на разработку двухканального терморегулятора

Общее описание разрабатываемого изделия. Устройство предназначено для измерения и поддержания в заданном диапазоне температуры в нагревательной камере промышленной установки термостатирования. Функциональная схема установки термостатирования приведена на рис.4.1. Камера оборудована двумя электронагревательными элементами (ТЭН1, ТЭН2) переключаемыми с помощью электромеханических реле переменного тока (Р1, Р2) и двумя датчиками температуры (ДТ1, ДТ2). Регулирование температуры осуществляется путем нагрева, до заданной температуры, воздуха нагревательной камеры с помощью электронагревательных элементов, которые включаются подачей напряжения питания, и последующего естественного его охлаждения. Скорость изменения температуры воздушной среды нагревательной камеры составляет от 10 до 15 °С/мин.

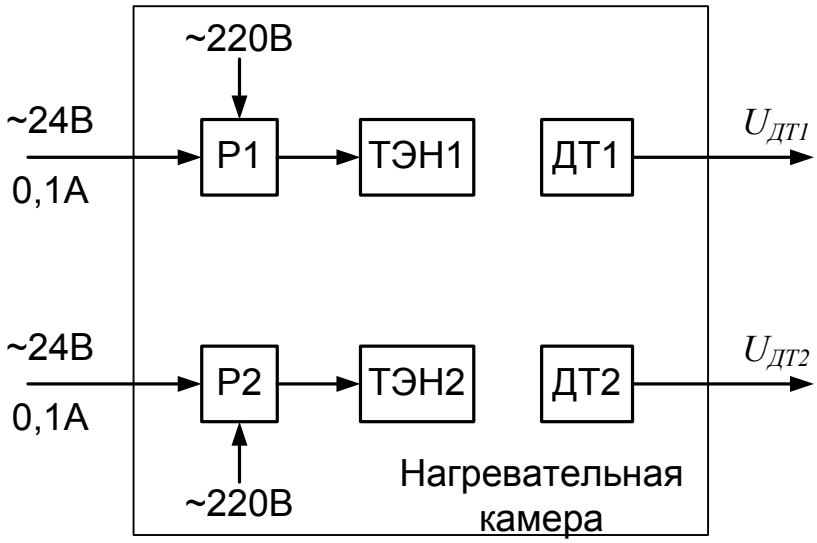


Рис. 4.1. Функциональная схема установки термостатирования

Характеристики сигнала управления электромеханическими реле:
- номинальное напряжение (действующее значение) – ~24 В;
- частота сигнала управления – 50 Гц;
- номинальный ток включения – 0,1 А.

Требования к техническим характеристикам терморегулятора:
- диапазон рабочих температур – от 0 до +300˚С;
- количество каналов измерения и регулирования – 2;
- точность измерения температуры – ± 1˚С;
- точность поддержания заданной температуры – ± 3˚С;
- дискретность установки заданной температуры – 1˚С;

Требования к функциональным характеристикам терморегулятора. Изделие должно выполнять следующие функции:

- измерение и цифровую индикацию двух каналов текущей температуры в нагревательной камере;
- независимую цифровую установку двух значений заданной температуры;
- хранение значений установленной температуры;
- индикацию включения электронагревательных элементов;
- звуковую сигнализацию входа\выхода в режим установки заданной температуры;
- индикацию включения устройства (наличие напряжения питания).

Требования к конструкции. Устройство должно представлять собой моноблок с разъемами для подключения внешних датчиков температуры и электромеханических реле управления электронагревательными элементами. Габаритные размеры устройства не должны превышать 70 х 70 х 130 мм.

Требования к питанию. Питание терморегулятора осуществляется от сети переменного тока общего пользования ~220В.

Условия эксплуатации. Устройство предназначено для работы в диапазоне рабочих температур окружающей среды – от +0 до +50˚С, при относительной влажности воздуха до 85% (при температуре +25˚С).

Специальные требования. Необходимо предусмотреть гальваническую развязку между управляющей частью и исполнительными органами системы регулирования температуры.

4.2. Разработка структурной схемы и алгоритма функционирования терморегулятора. Выбор типов датчиков и исполнительных устройств

Основное назначение разрабатываемого устройства - регулирование температуры нагревательной камеры установки термостатирования. Разрабатываемая система представляет собой электронную систему регулирования (управления), которая, как и большинство систем управления является системой замкнутого типа (рис.4.2). Она включает объекты управления - электронагревательные элементы (ТЭН1, ТЭН2), исполнительные органы – ключевые каскады (КК1, КК2), датчики информационных параметров – датчики температуры (ДТ1, ДТ2) и управляющее устройство (УУ). Особенностью данной системы является наличие двух независимых каналов управления.

Принцип действия системы регулирования заключается в следующем. Датчики температуры ДТ1 и ДТ2 измеряют текущую температуру в определенных областях нагревательной камеры. Сигнал с ДТ1 и

ДТ2 поступает на управляющее устройство, которое сравнивает текущие значения с заданным интервалом температур. При этом должно выполняться условие:

$$(T_i^{уст} - \Delta T) \le T_i \le (T_i^{уст} + \Delta T), \qquad (4.1)$$

где T_i - текущая температура ($i = \overline{1,2}$);

$T_i^{уст}$ - значения установленной температуры;

ΔT - точность поддержания температуры.

Если текущее значение по какому-либо каналу измерения ниже значения $(T_i^{уст} - \Delta T)$ управляющее устройство вырабатывает сигнал, открывающий соответствующий ключевой каскад, в результате чего напряжение питания подается на электронагревательный элемент. Происходит нагрев воздушной среды нагревательной камеры до тех пор, пока не выполниться условие $T_i \ge (T_i^{уст} + \Delta T)$. При выполнении указанного условия, управляющее устройство вырабатывает сигнал запирания ключевого каскада и осуществляется естественное охлаждение нагревательной камеры.

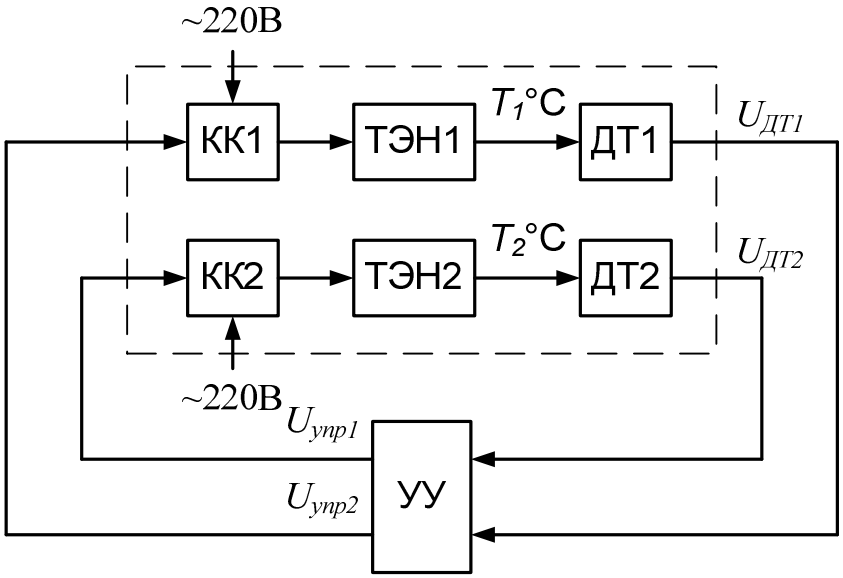


Рис. 4.2. Обобщенная структура системы регулирования температуры нагревательной камеры

Датчики температуры по материалу, из которого они изготавливаются, делятся на металлические и полупроводниковые. По принципу действия металлические датчики делятся на терморезистивные датчики и термопары [6, 10, 14].

В основе работы терморезистивных датчиков лежит зависимость электрического сопротивления от температуры. В разных диапазонах рабочих температур эта зависимость аппроксимируется интерполяционными уравнениями различной степени [14], но в первом приближении ее можно считать линейной:

$$R_T = R_0[\,1+\alpha(T-T_0)\,], \qquad (4.2)$$

где T - текущая температура;

R_0 - значения электрического сопротивления при температуре $T_0 = 0°\text{C}\ (273\text{K})$;

R_T - значения сопротивления при температуре $T°\text{C}$;

α - температурный коэффициент сопротивления (ТКС).

Прецизионные терморезистивные датчики обычно изготавливают из никеля ($\text{ТКС}=5{,}4\cdot10^{-3}\ \text{K}^{-1}$) или платины ($\text{ТКС}=3.9\cdot10^{-3}\ \text{K}^{-1}$). Их применяют в диапазоне температур от -260°С до +1300°С [14].

Для измерения сопротивления, а соответственно, и температуры на основе терморезистивных датчиков, широко применяют измерительную схему называемую мостом Уитсона (рис. 4.3), точнее его четвертьмостовую конфигурацию.

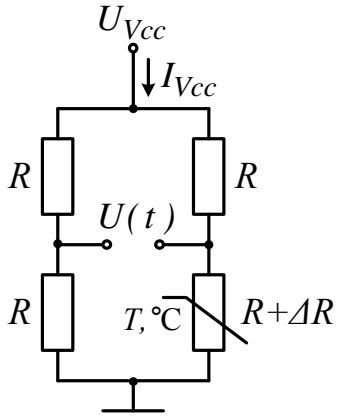


Рис. 4.3. Мост Уитсона

При $\Delta R = 0$ мост находится в нулевом (сбалансированном) состоянии $U(t)=0$. Выходное напряжение моста связано с изменением сопротивления ΔR нелинейно:

$$U(t) = \frac{U_{Vcc}}{4}\left[\frac{\Delta R}{R+\frac{\Delta R}{2}}\right]. \qquad (4.3)$$

Нелинейность указанной схемы составляет 0,05% от абсолютного значения. Следует отметить, что данная нелинейность относится только к схеме измерения, и поскольку она описывается аналитически, то может быть компенсирована при цифровой обработке сигнала измерения.

Дополнительная погрешность измерений вносится за счет саморазогрева сопротивления, вследствие чего ток запитки моста I_{Vcc} должен быть очень мал.

Таким образом, недостатками метода измерения температуры на основе терморезистивных датчиков являются нелинейность измере-

ний, погрешность за счет саморазогрева терморезистора, зависимость выходного сигнала от напряжения запитки моста.

Кроме того, как показывает практика, в силу конструктивных особенностей изготовления терморезисторов, они обладают большей инерционностью, чем термопары.

Термопары относятся к датчикам генераторного типа. В основе их работы лежит эффект генерации термо-ЭДС в точке контакта двух разнородных металлов под действием температуры. В зависимости от вида металлов термо-ЭДС может меняться от 7мкв/°С до 75мкв/°С. Диапазон рабочих температур составляет от 0°С до +2300°С [6, 14].

При подключении датчика на основе термопары появляется дополнительная паразитная термо-ЭДС, которая может быть компенсирована путем реализации полного датчика [10]. Полный датчик образуют две термопары (рис. 4.4), одна из которых постоянно находится при температуре 0°С (температура таяния льда).

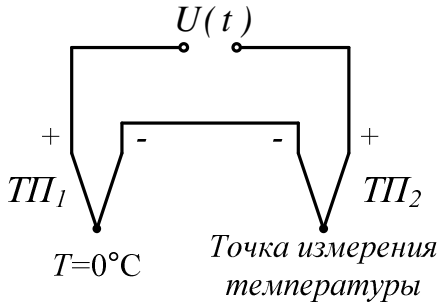


Рис. 4.4. Схема измерения температуры с использованием термопар

Преимущество метода измерения температуры с использованием термопар – высокая точность и разрешающая способность. Недостаток низкая повторяемость и возникновение паразитных термо-ЭДС, вследствие чего возникает необходимость настройки прибора.

Полупроводниковые датчики температуры основаны на зависимости сопротивления кристалла от температуры. ТКС таких датчиков в 3÷5 раз больше ТКС металлических датчиков, но поскольку их диапазон рабочих температур составляет от -50°С до +180°С, что не удовлетворяет требований технического задания (ТЗ), то рассматривать возможность применения датчиков указанного типа в разрабатываемой системе терморегулирования не имеет смысла.

Основываясь на приведенном выше анализе, можно сделать вывод, что наиболее подходящими датчиками температуры (ДТ1,ДТ2) для разрабатываемой системы являются датчики на основе термопар.

Ключевые каскады системы терморегулирования представляют собой реле переменного тока, коммутирующие напряжение запитки (~220 В) к нагревательным элементам (ТЭН1, ТЭН2). Специальным требованием при их реализации, согласно ТЗ является гальваническая развязка с управляющим устройством.

Гальваническая развязка между управляющими и силовыми цепями традиционно осуществляется с использованием электромеханических приборов (трансформаторов и реле). Тот же эффект может быть достигнут использованием приборов с оптической связью (оптоэлектронных реле или оптронов) [8,12]. Принцип действия приборов с оптической связью основан на двойном преобразовании энергии. В передатчике электрический сигнал преобразуется в световой пучок, а в приемнике осуществляется обратное преобразование, в результате чего приемник и передатчик связаны только оптической связью, что дает возможность осуществлять гальваническую развязку приемника и передатчика.

Оптоэлектронные приборы превосходят электромагнитные аналоги по надежности, долговечности, переходным и частотным характеристикам.

Можно отметить следующие достоинства этих приборов:

– возможность реализации бесконтактного оптического управления электронными объектами и обусловленные этим разнообразие и гибкость конструкторских решений управляющих цепей;

– однонаправленность распространения информации по оптическому каналу, отсутствие обратной реакции приемника на излучатель;

– широкая частотная полоса пропускания оптрона, отсутствие ограничения со стороны низких частот (что свойственно импульсным трансформаторам); возможность передачи по оптической линии как импульсного сигнала, так и постоянной составляющей;

– невосприимчивость оптических каналов связи к воздействию электромагнитных полей, что обусловливает их защищенность от помех;

– для оптронов не существует принципиальных физических или конструктивных ограничений по достижению высоких напряжений и сопротивлений развязки, а также малой проходной емкости.

Использование оптронов не требует специальных средств согласования с цифровыми ИМС.

Согласно ТЗ система терморегулирования должна осуществлять цифровую индикацию текущей и цифровую установку заданной температуры, очевидно, что в таком случае предпочтителен и цифровой вариант реализации управляющего устройства. При этом для отображения значений текущей температуры естественным решением, на наш взгляд, будет применение семисегментных светодиодных индикаторов (по три десятичных разряда на канал), поскольку отображение текстовой информации не требуется.

Управляющее устройство (контроллер) системы терморегулирования, как цифровое вычислительное устройство, может быть реализовано как на основе наборов ИС цифровой логики, так и на основе однокристальных решений (на основе однокристального микроконтрол-

лера, например). Второй вариант является более предпочтительным, поскольку позволяет минимизировать габариты разрабатываемого устройства, сократить сроки разработки и оставляет возможность реконфигурирования системы.

Резюмируя вышесказанное, приходим к окончательному виду структурной схемы разрабатываемой системы термостабилизации (рис. 4.5).

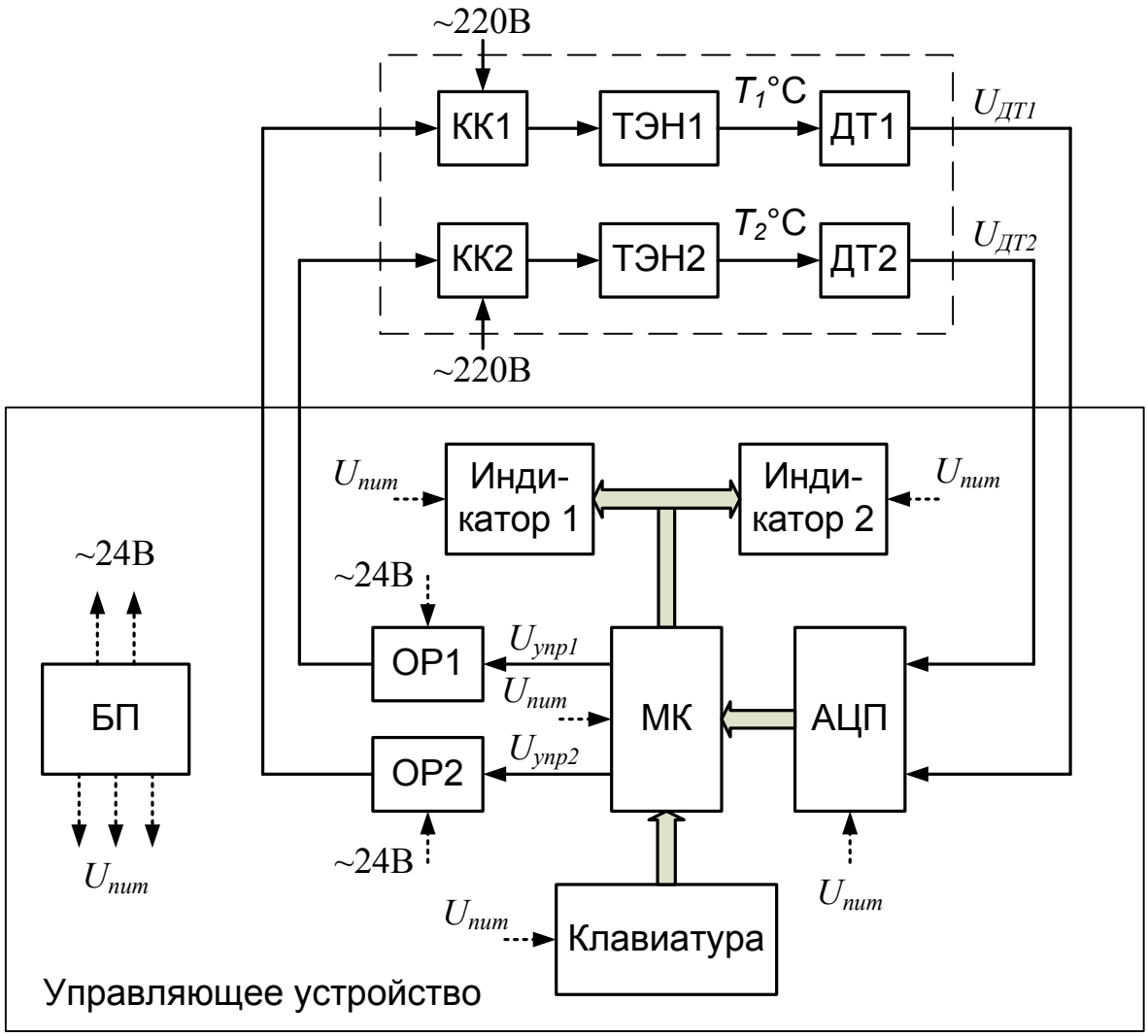


Рис. 4.5. Структурная схема разрабатываемой системы терморегулирования

Для преобразования сигналов аналоговых датчиков температуры к форме удобной для обработки в вычислительных устройствах в схеме предусмотрен двухканальный аналого-цифровой преобразователь (АЦП). Индикаторы 1 и 2 служат для непрерывной цифровой индикации текущей и задаваемой температуры нагревательной камеры. Клавиатура предназначена для установки заданной температуры стабилизации. Оптоэлектронные реле ОР1 и ОР2, как уже упоминалось выше, осуществляют гальваническую развязку управляющей части и исполнительных органов (силовой части) системы терморегулирования.

Все активные устройства управляющего контроллера запитываются от блока питания управляющего устройства (БП), на вход которого подается напряжение питания сети переменного тока общего пользования ~220 В.

4.3. Выбор элементной базы и разработка схемы электрической принципиальной терморегулятора

Синтез принципиальной схемы начнем с выбора основных элементов системы.

Датчики температуры. В качестве датчиков температуры выбраны датчики на основе термоэлектрического эффекта (термопары). В зависимости от диапазона измеряемых температур их подразделяют на низкотемпературные (до 300 °С), среднетемпературные (до 600 °С) и высокотемпературные (свыше 1800 °С), а в зависимости от способа применения - на погружаемые в рабочую среду и поверхностные [14]. Погружаемые термопары используют для измерения температуры в газообразных и жидких неагрессивных и агрессивных средах. Очевидно, что наиболее полно условиям ТЗ удовлетворяют среднетемпературные погружаемые термопары, поскольку для низкотемпературных верхняя граница диапазона рабочих температур соизмерима с указанным в ТЗ значением. Среди промышленно выпускаемых термопар, указанных типов, наибольшего распространения получили хромель-алюмелевые (ХА) или хромель-копелевые (ХК) термопары. Основные характеристики этих термопар [10] приведены в табл. 4.1.

Таблица 4.1.

Основные характеристики ХА и ХК термопар

Материал термопары	Тип	Диапазон измеряемых температур, °С	Значение термо-ЭДС, мВ, при температуре рабочего спая, °С						
			0	50	100	150	200	250	300
Хромель-алюмель	ТХА	-200…+1000	0	2,44	4,10	6,14	8,14	10,15	12,21
Хромель-копель	ТХК	-50…+600	0	4,10	6,90	10,62	14,57	18,69	22,88

Как видно из таблицы, значение термо-ЭДС термопары хромель-копель (ТХК) примерно в два раза выше, чем у термопары хромель-алюмель (ТХА) в одном и том же диапазоне рабочих температур, хотя диапазон измеряемых температур у ТХА шире. Поскольку диапазон измеряемых температур ТХК удовлетворяет заданному в ТЗ значению, в качестве датчиков температуры разрабатываемой системы терморегулирования выберем термопары типа ТХК.

Аналого-цифровой преобразователь. Аналого-цифровой преобразователь – устройство, предназначенное для преобразования входного аналогового сигнала в выходной цифровой код. Микросхемы АЦП выпускаются рядом производителей: Analog Devices, Maxim, Burr-Brown, STMicroelectronics, Hitachi, National Semiconductor, Texas Instruments и др. Основными характеристиками АЦП являются разрядность, быстродействие, диапазон входных напряжений и значение опорного напряжения, тип интерфейса, характеристики питания, тип корпуса.

Разрядность АЦП определяет разрешающую способность (точность) преобразования. В настоящее время выпускаются АЦП с разрядностью от 8 до 24. Разрешающую способность АЦП можно оценить, зная значимость младшего разряда, как величину обратную 2^n, где n - количество разрядов АЦП. Следует отметить, что реальная погрешность преобразования несколько выше значимости младшего разряда, за счет нелинейности характеристики АЦП, а также зависит от возможного «пропуска кодов».

В некоторых приложениях очень важное значение имеет быстродействие АЦП. По быстродействию современные АЦП можно условно разделить на три группы: АЦП низкого быстродействия (до 100 ksps), АЦП среднего быстродействия (от 100 до 500 ksps), АЦП высокого быстродействия (более 500 ksps). Для технологических контролеров чаще всего используются АЦП среднего и низкого быстродействия.

Опорное напряжение АЦП определяет диапазон возможных входных значений напряжений. Входному напряжению величина которого равна значению опорного напряжения соответствует максимальный цифровой код, представимый с помощью n-разрядов регистра результата АЦП. Опорное напряжение задается специальным высокостабильным источником опорного напряжения, который может быть, как внешним по отношению к АЦП, так и внутренним (встроенным).

В случае, когда АЦП реализован в виде отдельной ИМС, решающее значение может иметь скорость обмена АЦП с вычислительным устройством. Она зависит от используемого интерфейса передачи данных. Как правило, в АЦП реализован один из унифицированных последовательных или параллельных интерфейсов.

Вместе с тем во многих моделях современных микроконтроллеров АЦП присутствует в виде периферийного устройства.

В разрабатываемой системе терморегулирования может быть использован как АЦП, реализованный в виде отдельной ИМС, так и интегрированный в микроконтроллер. Как правило, АЦП реализованные в виде отдельной ИМС обладают более высокими показателями разрешающей способности и быстродействия, но при этом усложняется реализация обмена данными между АЦП и вычислительным устройством (требуется использование программной или аппаратной реализации последовательного или параллельного интерфейсов обмена

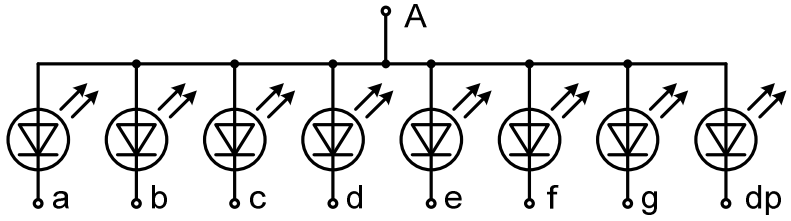
данными). Как показывает опыт практической реализации систем подобного типа, для измерения температуры с заданной в ТЗ точностью вполне достаточно интегрированного АЦП. Поэтому наличие АЦП будет являться одним из критериев выбора микроконтроллера управляющего устройства.

Индикатор. Использование семисегментных светодиодных индикаторов во встроенных системах позволяет отображать десятичные цифры, некоторые буквы русского и латинского алфавитов, а также некоторые специальные знаки. Семисегментные индикаторы по схеме соединения в них светодиодных сегментов делятся на два вида:

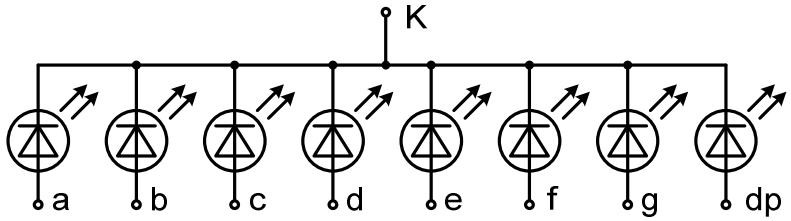
1) с общим анодом (рис. 4.6, а);

2) сообщим катодом (рис. 4.6, б).

«Зажигание» светодиодного сегмента для индикатора первого типа осуществляется путем подачи логического «0», для второго – логической «1» на соответствующий вывод индикатора.



а)



б)

Рис. 4.6. Типы соединений сегментов в индикаторах

Для отображения цифры на семисегментном индикаторе, ее машинное представление преобразуется в байт индикации (двоично-десятичный код), который непосредственно подается на выводы индикатора.

Для отображения многосимвольной информации отдельные семисегментные индикаторы объединяются в линейные (однострочные) дисплеи. Существует два способа организации интерфейса МК с линейным дисплеем: статический и динамический. Первый требует наличия на входах каждого индикатора специальных буферных регистров для хранения кодов выводимых символов. Это увеличивает число

дополнительных микросхем, а также повышает потребляемую мощность.

Динамический основан на том, что человеческий глаз является инерционным звеном; отображаемая на дисплее информация, если ее обновлять с частотой не менее 20 раз в секунду, представляется неизменной. Динамический способ индикации требует значительно меньших аппаратных затрат, но более сложного программного обеспечения. Именно этот способ нашел преимущественное распространение в встроенных системах.

При динамической индикации байт индикации поступает одновременно на входы всех семисегментных индикаторов, образующих линейный дисплей, а выбор знакоместа осуществляется байтом выборки. Для обеспечения яркой и ровной (немигающей) индикации необходимо обеспечить: во-первых, запрет выборки знакомест на время изменения байта индикации (бланкирование), во-вторых, регенерацию изображения на каждом знакоместе с частотой не менее 20 Гц.

В настоящее время промышленностью выпускаются специализированные ИМС, реализующие не только аппаратное преобразование числа в двоично-десятичный код, но и осуществляющие управление линейным семисегментным индикатором. Примером такого индикаторного контроллера (драйвера) служит серийно выпускаемые ИМС MAX7219/MAX7221 фирмы MAXIM [34]. Драйверы MAX7219/MAX7221 используются для управления линейным 8-разрядным (до 8 десятичных разрядов числа) семисегментным индикатором с общим катодом. Они содержат в себе аппаратный декодер, осуществляющий преобразование из двоичного в двоично-десятичный код и ОЗУ для хранения восьми байтов индикации. Драйвер MAX7221 кроме прочего имеет встроенный аппаратный интерфейс SPI для обмена данными с процессорным устройством, что облегчает программную реализацию обмена данными.

Таким образом, в случае использования MAX7221, в качестве индикаторов могут быть выбраны любые 3-х разрядные семисегментные индикаторы с общим катодом, например, BT-N325RD [40]. При этом схема подключения индикаторов системы терморегулирования будет иметь вид представленный на рис. 4.7.

Резистор R позволяет задать ток для всех сегментов одновременно. Минимальному значению $R = 9{,}53$ кОм соответствует ток разряда равный 40 мА. Средний ток, а, следовательно, и средняя рассеиваемая разрядом мощность, определяется не только током разряда, но и частотой регенерации изображения на нем. Выберем $R = 10$ кОм. Методика расчета рассеиваемой мощности приведена в [34].

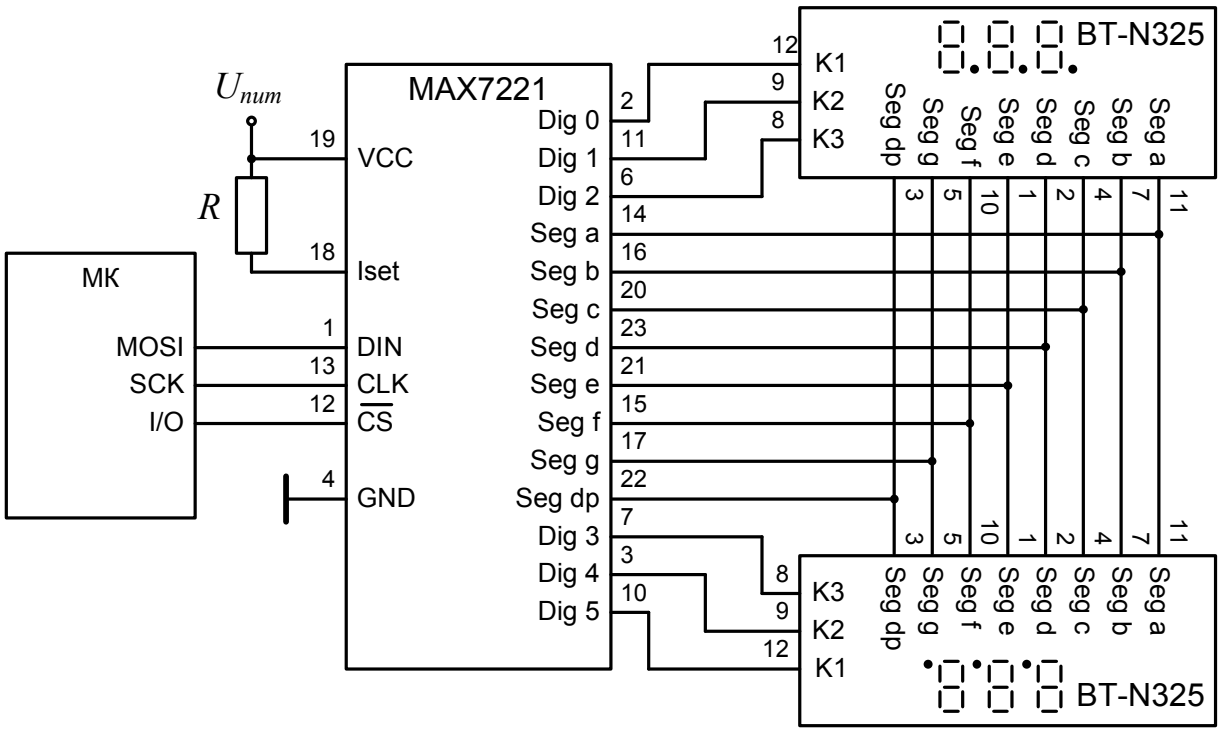


Рис. 4.7. Схема подключения индикаторов температуры

При выборе микроконтроллера следует обратить внимание на наличие в нем встроенного интерфейса SPI.

Клавиатура. Для установки заданной температуры и выбора режимов работы системы терморегулирования используем трехпозиционную клавиатуру. Назначение клавиш будет детальнее описано в подразделе касающемся разработки программного обеспечения микроконтроллера.

При работе микроконтроллера с датчиками, имеющими механические или электромеханические контакты (кнопки, клавиши, реле, клавиатуры и т.д.), возникает явление, называемое дребезгом контактов. Оно заключается в том, что при замыкании контактов происходит их отскок, под действием сил упругости, обратное соединение, повторный отскок и т.д. Это приводит к переходному процессу, при котором сигнал с контакта может быть прочитан как случайная последовательность нулей и единиц (рис. 4.8). Подавить это нежелательное явление можно аппаратными либо программными способами.

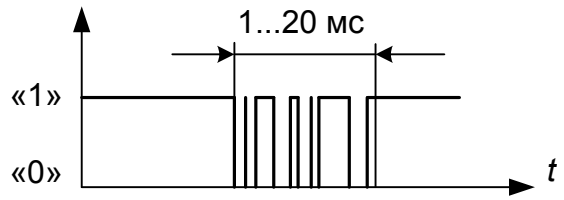
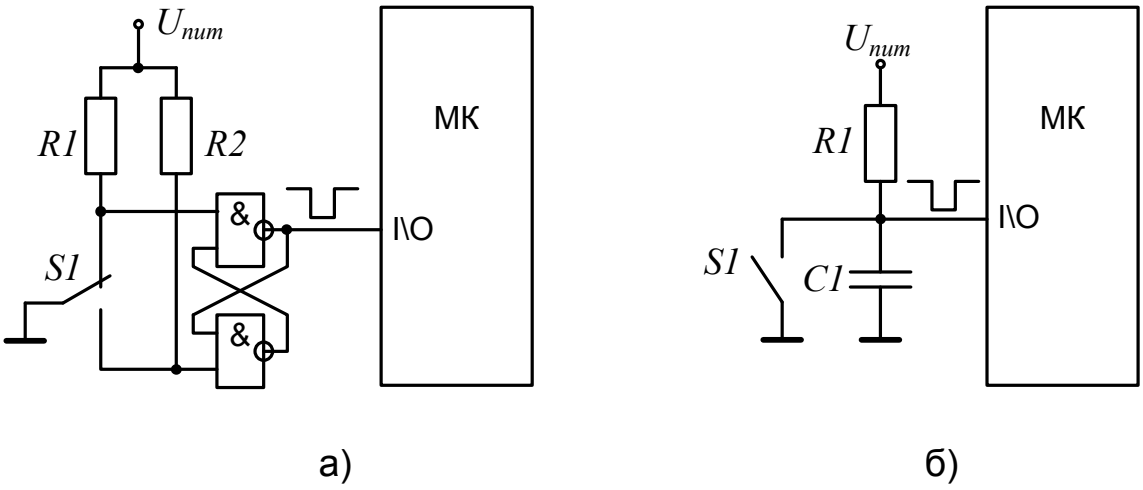


Рис. 4.8. К пояснению эффекта дребезга контактов

Аппаратные способы заключаются в использовании схемы одновибратора (рис. 4.9, а), или фильтра нижних частот (рис. 4.9, б) для сглаживания пульсаций, следующих с периодом меньшим, чем постоянная времени фильтра (RC-цепочки) $\tau = RC$.



а) б)

Рис. 4.9. Способы аппаратного подавления дребезга контактов

Среди программных способов наибольшее распространение получили два:

1) подсчет заданного числа совпадающих значений сигнала;

2) временная задержка.

Суть первого способа заключается в многократном считывании сигнала с контакта. Подсчет удачных опросов (т.е. опросов, обнаруживших, что контакт замкнут) ведется программным счетчиком. Если после серии удачных опросов встречается неудачный, то подсчет начинается сначала. Контакт считается устойчиво замкнутым (дребезг устранен), если последовало N удачных опросов. Число N подбирается экспериментально для каждого типа используемых датчиков и лежит в пределах от 10 до 100.

Устранение дребезга контактов путем введения временной задержки заключается в том, что программа, обнаружив замыкание контакта, запрещает опрос состояния этого контакта на время, заведомо большее длительности переходного процесса. Время задержки подбирается экспериментально для каждого типа датчиков в пределах 1…20 мс и реализуется обычно с помощью одного из таймеров\счетчиков.

При проектировании системы терморегулирования остановимся на способе подавления дребезга контактов с использованием RC-цепочки, так как он требует минимальных аппаратных затрат и не усложняет программной реализации процедур обработки нажатия клавиш. Электрическая принципиальная схема подключения клавиатуры к микроконтроллеру приведена на рис. 4.10. Рассчитаем параметры

RC-цепочек. Для всех RC-цепочек на рис. 4.10 параметры установим одинаковыми.

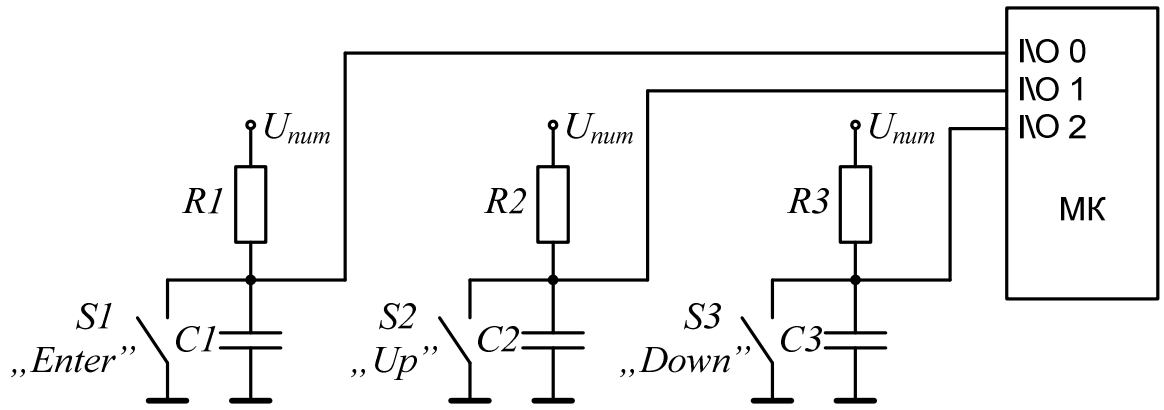


Рис. 4.10. Схема подключения клавиатуры к микроконтроллеру управляющего устройства

Пусть время переходных процессов (время дребезга контактов) в используемых клавишных переключателях $S1...S3$ равно $t_{др} = 1$мс. Условием стабильной фиксации логического уровня на входе микроконтроллера при устранении дребезга контактов с помощью RC-цепочки является неравенство

$$t_{др} \le \tau_{RC}, \qquad (4.4)$$

где $\tau_{RC} = RC$ – постоянная времени RC-цепочки.

Зададимся значением резисторов $R1 = R2 = R3 = 10$кОм. Тогда согласно (4.4) $C1 = C2 = C3 \ge 1 \cdot 10^{-7}$ Ф. Выберем значения из стандартного ряда номиналов $C1 = C2 = C3 = 0{,}1$мкФ.

Выбор микроконтроллера системы регулирования температуры. В качестве микроконтроллера системы регулирования может быть использован один из 8-битных микроконтроллеров, широко применяемых в настоящее время для решения подобного рода задач.

Выбор микроконтроллера для реализации управляющих функций разрабатываемой системы регулирования является вопросом неоднозначным, имеющим несколько вариантов решения. Как правило, задача решается эвристическими методами. Основными характеристиками, которые при этом принимаются во внимание, являются:

- производительность, MIPS (MIPS – *Million Instructions per Second* – миллион инструкций в секунду);
- количество линий ввода/вывода;
- состав периферийных блоков;
- тактовая частота, МГц;
- энергопотребление, мВт/МГц;
- цена.

Важность того или иного параметра определяется в каждом конкретном случае индивидуально. Например, первый параметр – производительность (количество выполняемых операций в единицу времени). Точное значение требуемой производительности можно определить, зная количество операций, требуемое для реализации алгоритма работы устройства с помощью данного конкретного микроконтроллера. Определить это значение можно в том случае, если однозначно установлен алгоритм работы программного обеспечения микроконтроллера, а алгоритм может быть однозначно определен только тогда, когда выбран конкретный микроконтроллер, разработана принципиальная схема, определены используемые периферийные узлы и т.д. Получается замкнутый круг.

Еще сложнее решать задачу выбора микроконтроллера при разработке принципиально нового устройства, не имеющего аналогов, поскольку в процессе разработки алгоритм работы устройства может претерпевать существенных изменений.

Несколько сократить область выбора могут ограничения по другим, приведенным выше критериям.

Рассчитаем требуемое количество линий ввода\вывода, а также определим требуемые периферийные узлы микроконтроллера, для реализации управляющего устройства системы терморегулирования.

Для подключения клавиатуры используются три линии ввода\вывода. Две линии используются для управления исполнительными механизмами. Еще две линии ввода\вывода используются для отображения состояния ТЭН1 и ТЭН2. Одна линия – для управления звуковой сигнализацией выбора режимов работы устройства.

Таким образом, микроконтроллер должен иметь как минимум 8 линий ввода\вывода общего назначения. Кроме того, для упрощения обмена данными с индикаторами (см. выше), требуется наличие интерфейса SPI; для оцифровки сигналов, поступающих с аналоговых датчиков температуры – АЦП, а для хранения установленных значений температуры – энергонезависимая память данных (EEPROM - Electrically Erasable Programmable Read-Only Memory - электрически стираемое программируемое постоянное запоминающее устройство - ЭСППЗУ), емкостью не менее 4 байт (по 2 байта для каждого значения температуры).

Рассчитаем разрядность АЦП микроконтроллера.

Согласно ТЗ диапазон измеряемых температур составляет $T_{\Delta} = 300^{\circ}\mathrm{C}$, а точность измерения - $\delta T = 1^{\circ}\mathrm{C}$. Для обеспечения заданной точности, требуемое количество уровней квантования электриче-

ского сигнала, соответствующего измеряемой физической величине, должно составлять:

$$N_{кв} = \frac{T_\Delta}{\delta T}. \qquad (4.5)$$

Тогда требуемое количество разрядов АЦП [2]:

$$n = E\left[\log_2(N_{кв})\right], \qquad (4.6)$$

где $E(x)$ - оператор округления до ближайшего большего целого.

Для рассматриваемого случая согласно (4.5), (4.6) n=9, следовательно, для аналого-цифрового преобразования сигналов внутренний АЦП микроконтроллера должен иметь разрядность не менее 9.

Из всего разнообразия моделей 8-битных микроконтроллеров фирмы STMicroelectronics оптимальным по приведенным выше критериям является микроконтроллер STF7LITE19.

Его основные характеристики [39]:

- объем памяти программ – 4 кБ;
- объем ОЗУ – 256 байт;
- объем ЭСППЗУ – 128 байт;
- количество линий ввода\вывода – 15;
- встроенный 10-разрядный АЦП (7 входных каналов);
- встроенный интерфейс SPI;
- один 8-битный и один 12-битный таймер\счетчик;
- тактовая частота ЦПУ – до 8 МГц;
- напряжение питания – от 2,7 В до 5,5 В.

Разработка принципиальной схемы. Напряжение питания драйвера MAX7221 и микроконтроллера составляет +5 В, поэтому аналогичное напряжение питания может быть выбрано для остальных блоков системы терморегуляции (см. рис. 4.5).

Поскольку питание терморегулятора согласно ТЗ должно осуществляться от сети переменного тока общего пользования ~220В, для получения $U_{пит} = +5\ \text{В}$ нужно использовать вторичный источник питания.

Рассчитаем энергетические показатели терморегулятора.

В случае, когда одновременно нажаты все (три) клавиши, ток потребления клавиатуры составляет

$$I_{кл}^{max} = 3 \times \frac{U_{пит}}{R} = 3 \times \frac{5\ \text{В}}{10 \cdot 10^3\ \text{Ом}} = 1{,}5\ \text{мА}.$$

В качестве оптронов системы терморегулирования могут быть выбраны любые оптроны для коммутации переменного тока не менее 0,1 мА, с коммутируемым напряжением не менее ~24 В (действующее значение) и током управления не более 20 мА (максимальный выходной ток вывода микроконтроллера). Например, PVT312 [38]. Схема подключения оптронов к микроконтроллеру приведена на рис. 4.11.

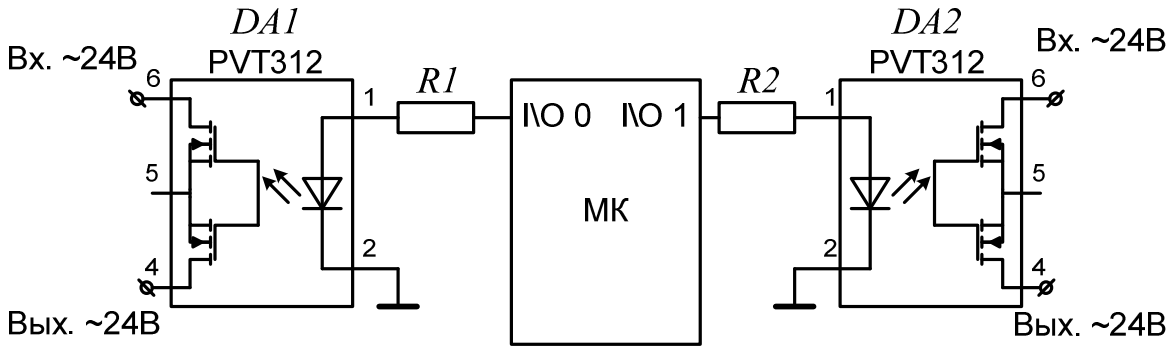


Рис. 4.11. Схема подключения оптронов

Сопротивления $R1$ и $R2$ устанавливают величину управляющего тока оптронов. Их значения рассчитывают следующим образом.

Минимальное значение управляющего тока оптрона – $I_{упр}^{min} = 2$ мА [38]. Для надежного срабатывания номинальный ток управления оптронов выбирается равным

$$I_{упр}^{ном} = K_з \cdot I_{упр}^{min}, \qquad (4.7)$$

где $K_з$ - коэффициент запаса ($K_з = 1{,}5 \div 3$). Зададим $K_з = 2$, тогда согласно (4.7) $I_{упр}^{ном} = 5$ мА.

По вольтамперной характеристике оптрона [38], при выбранном управляющем токе, определяем падение напряжения на диоде - $U_{VD} = 1{,}2$ В.

Напряжение высокого логического уровня на выходе микроконтроллера (при $U_{пит} = +5$ В) составляет $U_{лог."1"}^{min} \geq 3{,}4$ В. Таким образом, на ограничительном сопротивлении $R1$ ($R2$) должно падать напряжение $U_{R1(R2)} = 2{,}2...3{,}8$ В. Соответственно $R1$ ($R2$) для обеспечения заданного управляющего тока должны находится в пределах $R1 = R2 = \frac{U_{R1(R2)}}{I_{упр}^{ном}} = 440...760$ Ом. Выберем номинальное значение сопротивлений $R1 = R2 = 560$ Ом.

Когда одновременно включены оба ТЭНа (то есть, поданы управляющие сигналы на оба оптрона), максимальный ток потребления оптронов составляет $I_{опт}^{мах} = 2 \times \frac{U_{лог."1"}^{max} - U_{VD}}{R} = 2 \times \frac{5 \text{ В} - 1{,}2 \text{ В}}{560 \text{ Ом}} = 13{,}6$ мА.

Максимальный ток потребления микроконтроллера - $I_{МК}^{max} = 100$ мА [39], а максимальный ток потребления драйвера совместно с индикаторами - $I_{дрв+инд}^{max} = 500$ мА [34].

Таким образом, максимальный ток, потребляемый управляющей частью устройства

$$I_{max} = I_{кл}^{max} + I_{опт}^{мах} + I_{МК}^{max} + I_{дрв+инд}^{max} = 1{,}5 \text{ мА} + 13{,}6 \text{ мА} + 100 \text{ мА} + 500 \text{ мА} = 615{,}1 \text{ мА}.$$

Принципиальная схема вторичного источника питания приведена на рис. 4.12. Он включает в себя понижающий трансформатор, диодный мост и стабилизатор напряжения. В качестве последнего используется интегральный стабилизатор L7805 [37] или его аналог.

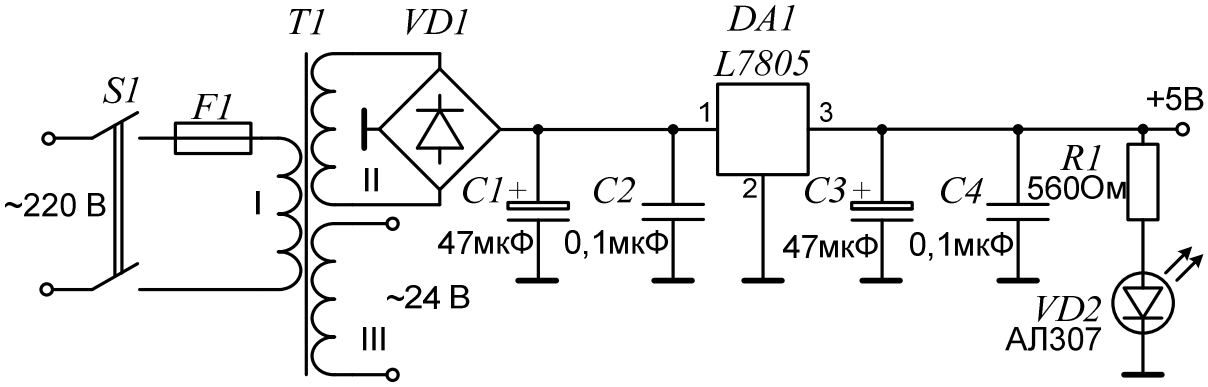


Рис. 4.12. Схема источника питания терморегулятора

Вторичная обмотка трансформатора II от которой запитывается управляющая часть устройства должна быть рассчитана на выходной ток $I_{вых.II} = K_з \cdot I_{max} = 2 \cdot 615{,}1 \text{ мА} = 1230{,}2 \text{ мА}$ и напряжение $U_{вых.II} = \sim 7{,}5 \text{ В}$ (амплитудное значение). То есть выходная мощность обмотки II трансформатора должна составлять $P_{вых.II} = U_{вых.II} \cdot I_{вых.II} = 7{,}5 \text{ В} \cdot 1230{,}2 \text{ мА} = 9{,}23 \text{ В} \cdot \text{А}$. Обмотка III используется для управления электромеханическими реле (исполнительными органами установки) ее выходная мощность должна составлять $P_{вых.III} = U_{вых.III} \cdot K_з \cdot I_{вых.III} = 24 \text{ В} \cdot 2 \cdot 0{,}2 \text{ А} = 9{,}6 \text{ В} \cdot \text{А}$.

Выбор диодного моста определяется максимальным выходным током вторичной обмотки II трансформатора $I_{вых.II}$. В качестве такового может быть использован SB201, рассчитанный на максимальный ток 2 А и входное напряжение 35 В (действующее значение) [36].

Номинальные значения емкости блокировочных конденсаторов С1–С4 выбраны согласно рекомендациям производителя стабилизатора L7805 [37].

Резистор $R1$ ограничивает ток, протекающий через светодиод VD2, который сигнализирует наличие питания прибора (индикацию включения устройства). Его величина выбирается исходя из следующих соображений. Номинальный ток, при котором свечение светодиода достигает номинальной яркости, для большинства выпускаемых светодиодов составляет примерно $I_{VD} = 10 \text{ мА}$. При этом падение напряжения на светодиоде, определяемое по его вольтамперной характеристике, составляет порядка 1,8 - 1,9 В. Таким образом, на ограничивающем резисторе должно падать напряжение

$U_{R1} = U_{пит} - (1{,}8 \div 1{,}9)$ В. При номинальном токе $I_{VD} = 10$ мА и напряжении питания $U_{пит} = +5$ В, согласно закона Ома, сопротивление $R1 = 310 \div 320$ Ом. Как показывает практический опыт, без заметного снижения яркости, сопротивление $R1$ может быть увеличено в 1,5-3 раза (особенно для так называемых суперярких светодиодов). Из соображений унификации выберем номинальное значение $R1 = 560$ Ом таким же, как и для ограничительных резисторов оптронов.

Точность аналого-цифрового преобразования определяется не только разрядностью АЦП, но и динамическим диапазоном входного аналогового сигнала.

Рассчитаем величину уровня квантования сигнала температурного датчика. Из табл. 4.1 при температуре 300ºС (верхняя граница диапазона измеряемых температур) величина напряжения на выходе термопары равна $U_{max} = 22{,}88$ мВ. Абсолютная величина напряжения, соответствующего уровню квантования, при количестве разрядов АЦП n, равна $\Delta U = \dfrac{U_{max}}{2^n} = \dfrac{22{,}88 \text{ мВ}}{2^{10}} = 22{,}3$ мкВ. Чем больше ΔU по сравнению с дисперсией шума на входе АЦП, тем выше точность измерения уровня сигнала (меньше сказывается влияние шумов на точность измерения). Абсолютная величина напряжения, соответствующая уровню квантования ΔU пропорциональна величине U_{max}, которая определяет динамический диапазон выходных сигналов термодатчиков.

Очевидно, что максимальная точность будет достигнута, когда динамический диапазон сигнала будет равен динамическому диапазону АЦП, который определяется величиной опорного напряжения. Величина опорного напряжения АЦП микроконтроллера ST7FLITE19 составляет $U_{пит} = +5$ В.

Для увеличения динамического диапазона сигнала температурных датчиков системы терморегулирования в каждом канале измерения использованы масштабирующие усилители (рис. 4.13).

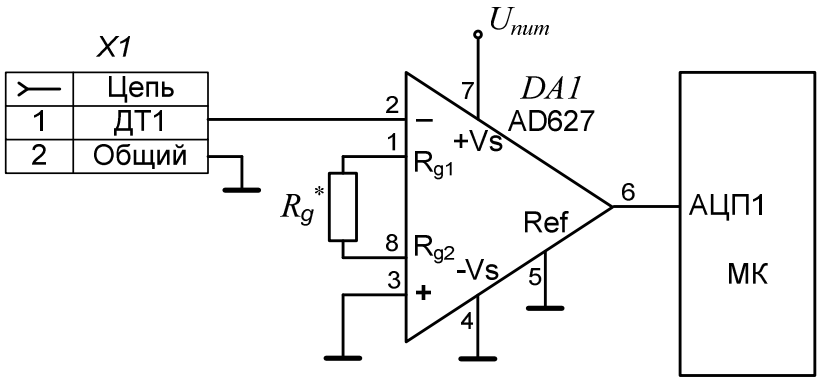


Рис. 4.13. Схема входного усилителя терморегулятора

Схема разрабатываемого устройства представленный на рис. 4.14.

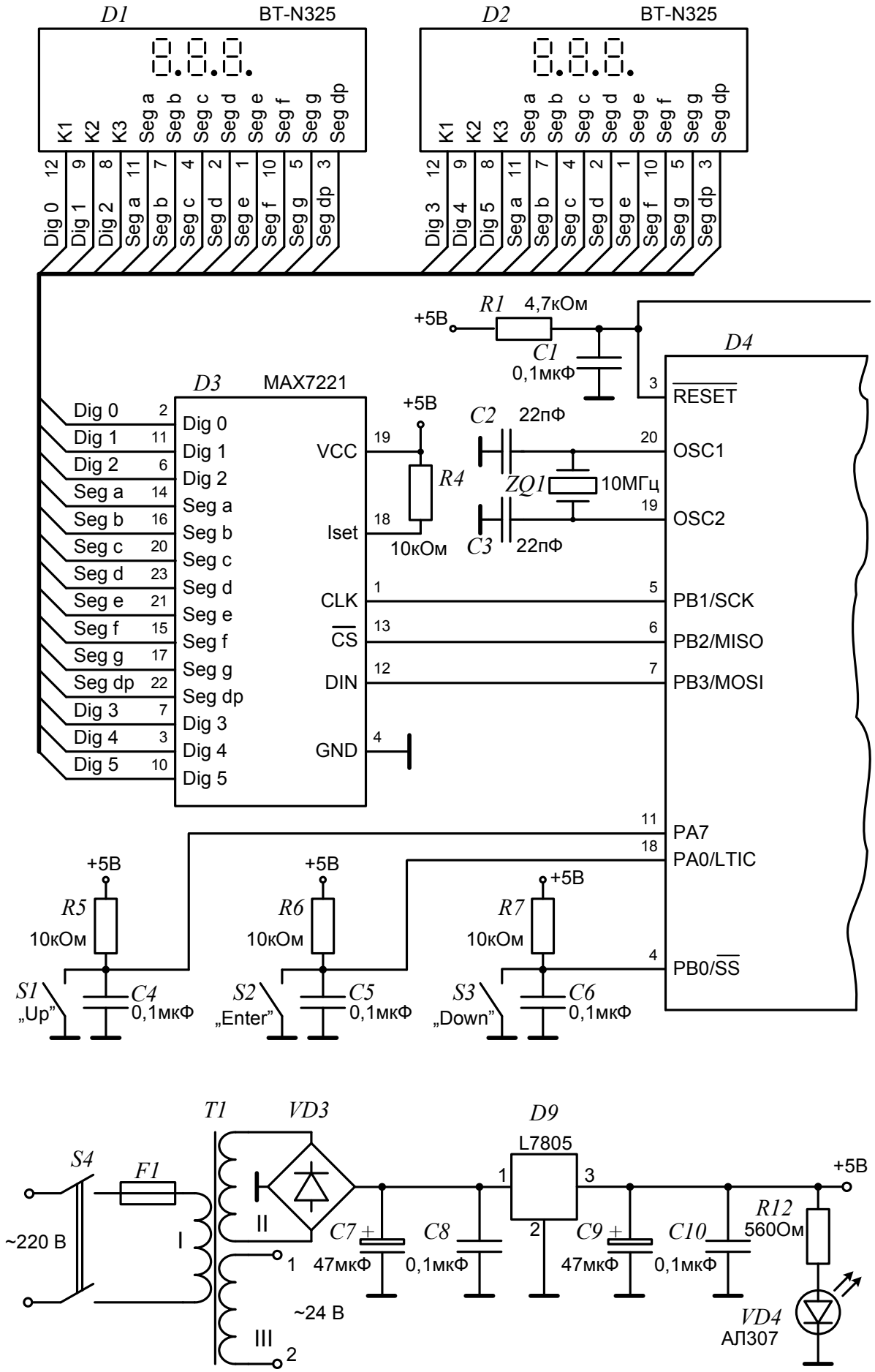
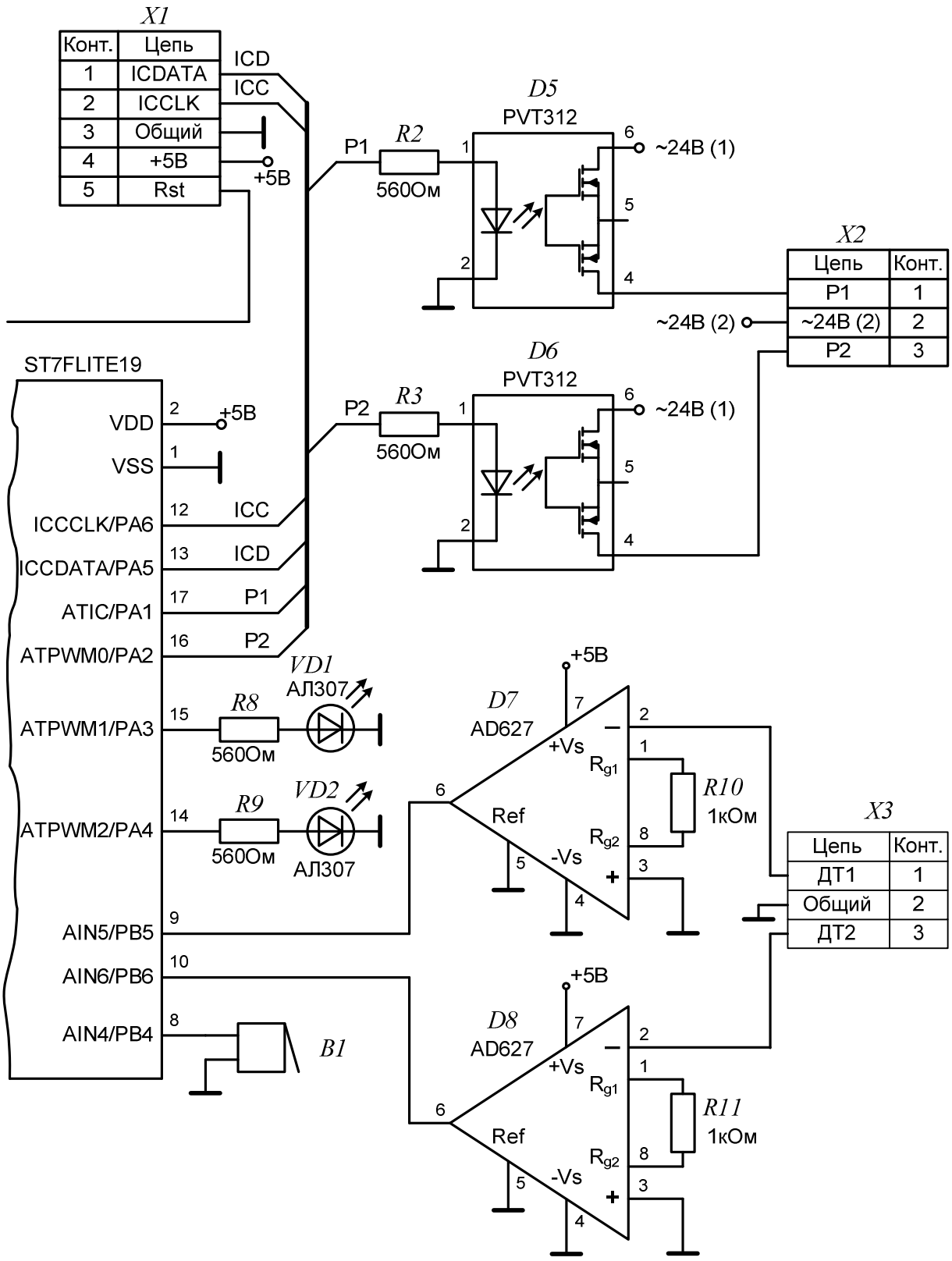


Рис. 4.14. Схема электрическая



принципиальная терморегулятора

Для усиления малых дифференциальных сигналов (сигналы с выхода термопар), к которым могут быть примешаны большие синфазные помехи применены инструментальные усилители AD621 [35]. Термином «инструментальный усилитель» («измерительный усилитель») обозначают дифференциальный усилитель со связями по постоянному току, высоким коэффициентом усиления, высоким входным полным сопротивлением и большим коэффициентом подавления синфазного сигнала [23].

Требуемый коэффициент усиления определяется соотношением:

$$K_{yc} \le \frac{U_{on}}{U_{max}} = \frac{5\ \text{В}}{22{,}88 \cdot 10^{-3}\ \text{В}} = 218{,}5. \qquad (4.8)$$

Для AD621 коэффициент усиления задается внешним резистором *Rg*, сопротивление которого определяется [35]:

$$Rg = \frac{200\ \text{кОм}}{K_{yc} - 5}. \qquad (4.9)$$

Согласно (4.9) для $K_{yc} = 218{,}5$, сопротивление $Rg = 937\ \text{Ом}$. Выберем ближайшее номинальное значение $Rg = 1\ \text{кОм}$. Тогда реальный коэффициент усиления составит $K_{yc} = 205$, что удовлетворяет условию (4.8).

Для звуковой сигнализации входа\выхода в режим установки заданной температуры использован пьезоэлектрический излучатель $B1$.

Ограничительные резисторы $R8, R9$ рассчитываются по той же методике что и $R12$.

Предохранитель $F1$ должен быть рассчитан на ток $I_{F1} = K_з \cdot \frac{P_{вых.II} + P_{вых.III}}{220B} = 10 \cdot \frac{9{,}23\ \text{В} \cdot \text{А} + 9{,}6\ \text{В} \cdot \text{А}}{220B} = 0{,}85\ \text{А}$.

Разъем $X1$ служит для подключения внутрисхемного программатора\отладчика, $X2$, $X3$ - исполнительных органов (реле) и датчиков температуры, соответственно.

Для обеспечения производительности микроконтроллера близкой к максимальной, выбран кварцевый резонатор *ZQ1*, рассчитанный на частоту 10 МГц.

4.4. Разработка управляющей программы микроконтроллера

В соответствии с логикой работы, описанной выше, программное обеспечение микроконтроллера должно реализовать два основных режима работы устройства – режим индикации текущей температуры и режим установки заданной температуры. В обоих режимах устройство должно выполнять основную функцию – регулирование температуры путем включения\выключения ТЭН1, ТЭН2.

Программное обеспечение разрабатывается с учетом системы команд микроконтроллера ST7FLITE19 на языке Ассемблер.

На рис. 4.15 представлена блок-схема алгоритма работы основной программы. Фрагменты исходного кода программы приведены в приложении П.6.

После инициализации системы тактирования и периферийных узлов микроконтроллера, из ЭСППЗУ считываются значения температуры $T_{уст1}$ (T1H:T1L), $T_{уст2}$ (T2H:T2L), установленной (заданной) для каждого из каналов в предыдущем сеансе работы.

В подпрограмме вычисления пороговых значений (рис. 4.16) вычисляется нижнее значение температуры в нагревательной камере ($T_{вкл.1} = T_{уст1} - \Delta T$), при достижении которого происходит включение термоэлектронагревателя, и верхнее значение, при котором он выключается ($T_{выкл.1} = T_{уст1} + \Delta T$). Поскольку установленные температуры для первого и второго каналов могут отличаться, то могут отличаться и пороговые значения для ТЭН1 и ТЭН2. Алгоритм вычисления пороговых значений второго канала аналогичен. Величина точности поддержания заданной температуры – постоянная и, согласно ТЗ, составляет $\Delta T = \pm 3$ °C.

Основная функция устройства – регулирование температуры будет реализована с использованием прерывания таймера\счетчика. В обработчике этого прерывания будет также индицироваться текущая температура, поэтому в основной программе после установки глобального разрешения прерываний, осуществляется переход на опрос клавиатуры. Нажатие клавиши $S2$ "Enter" (см. рис. 4.14) приводит к переходу в режим установки заданной температуры. В этом режиме индикация текущей температуры временно прекращается, индицируются значения устанавливаемой температуры. После установки заданной температуры устройство возвращается к опросу клавиши $S2$ "Enter" и возобновляет индикацию текущей температуры.

Рис. 4.15. Блок-схема основной программы микроконтроллера

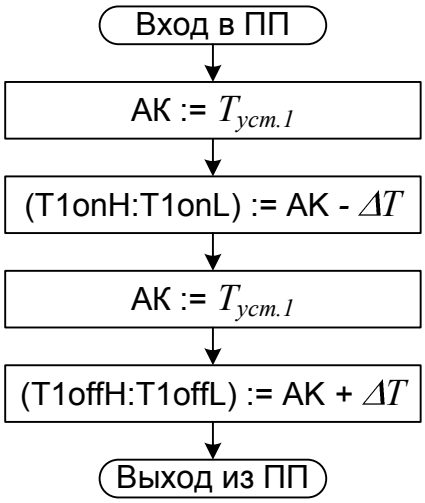


Рис. 4.16. Блок-схема подпрограммы вычисления пороговых значений первого канала

Определим временной интервал между прерываниями таймера\счетчика.

Согласно теоремы отсчетов Котельникова (теоремы Найквиста) интервал дискретизации сигналов ДТ1, ДТ2 должен составлять:

$$T_{\partial} \le \frac{1}{2F_{в}}, \qquad (4.10)$$

где $F_{в}$ - верхняя граничная частота в спектре сигнала.

Скорость изменения температуры в нагревательной камере, заданная в ТЗ, составляет от 10 до 15 °С/мин. Следовательно, верхняя граничная частота в спектре сигнала температурного датчика будет равна $F_{в} = \frac{15}{60}\ c^{-1} = 0{,}25\ \text{Гц}$, а интервал дискретизации (интервал отсчитываемый таймером), согласно (4.10) - $T_{\partial} \le \frac{1}{2 \cdot 0{,}25\ \text{Гц}} = 2\ \text{с}$.

Поскольку в подпрограмме-обработчике прерывания также должна происходить индикация значений текущей температуры, осуществляемая динамическим способом, то нижнее значение временного интервала будет определяться частотой обновления разрядов семисегментного индикатора. Для драйвера MAX7221 частота регенерации изображения разрядов равна:

$$F_{рег} = \frac{8 \cdot f_{osc}}{N}, \qquad (4.11)$$

где f_{osc} - частота сканирования разрядов MAX7221 ($f_{osc} = 800\ \text{Гц}$ [34]); N - количество разрядов индикации.

При требуемых $N = 6$ разрядах индикации, согласно (4.11): $F_{рег} \approx 1067\ \text{Гц}$ $\left(T_{рег} = \frac{1}{F_{рег}} \approx 0{,}94\ \text{мс}\right)$.

Таким образом, период дискретизации сигналов датчиков температуры (временной интервал, отсчитываемый таймером) должен находится в пределах

$$0{,}94 \text{ мс} \le T_{\partial} \le 2 \text{ с}, \qquad (4.12)$$

Выберем значение T_{∂} на порядок больше интервала регенерации: $T_{\partial} = 10$ мс.

Для формирования временного интервала может быть использован один из таймеров: автозагружаемый таймер AT2 или таймер LT2 (включающий в себя два счетчика).

Источником счетных импульсов таймера AT2 может быть ГТИ ($F_{ГТИ} = 5$ МГц при частоте кварцевого резонатора 10 МГц [39]), выход счетчика 1 таймера LT2 ($F_{LT2} = 1$ кГц) и генератор импульсов с частотой $F_{32M} = 32$ МГц. В первом случае количество счетных импульсов, отсчитываемых таймером для формирования указанного временного интервала равно $N = T_{\partial} \cdot F_{ГТИ} = 0{,}01 \cdot 5 \cdot 10^6$ Гц=50000. Очевидно, что такое количество импульсов 12-битный таймер AT2 подсчитать не может. При $F_{32M} = 32$ МГц N получается еще большим.

Рассмотрим вариант использования таймера LT2 для формирования требуемого временного интервала.

Счетными импульсами обоих счетчиков таймера LT2 выступают импульсы с частотой $\frac{F_{ГТИ}}{32}$. 8-битный счетчик 2 таймера является автозагружаемым. При $F_{ГТИ} = 5$ МГц максимальный временной интервал, который он может сформировать, равен $t_{в.и.} = \frac{32}{F_{ГТИ}} \cdot (2^8 - 1) = 1{,}6$ мс, что меньше требуемого.

Таким образом, при выбранной частоте кварцевого резонатора для формирования временного интервала $T_{\partial} = 10$ мс может быть использован только таймер AT2, в качестве счетных импульсов которого служит сигнал счетчика 1 таймера LT2 ($T_{LT2} = 1$ мс). Количество счетных импульсов, которое должен подсчитать таймер равно 10, следовательно в регистр автозагрузки нужно записать число $2^{12} - 10 = 4086 = 0FF6h$.

Возможен также вариант уменьшения частоты кварцевого резонатора. Но это приведет к уменьшению вычислительной мощности ЦПУ, а поскольку алгоритм работы программного обеспечения микроконтроллера пока еще не определен, то есть не известно реальное требуемое быстродействие микроконтроллера, то этот вариант менее желателен.

Подпрограмма обработки прерывания по переполнению таймера AT2 представлена на рис. 4.17 - 4.18.

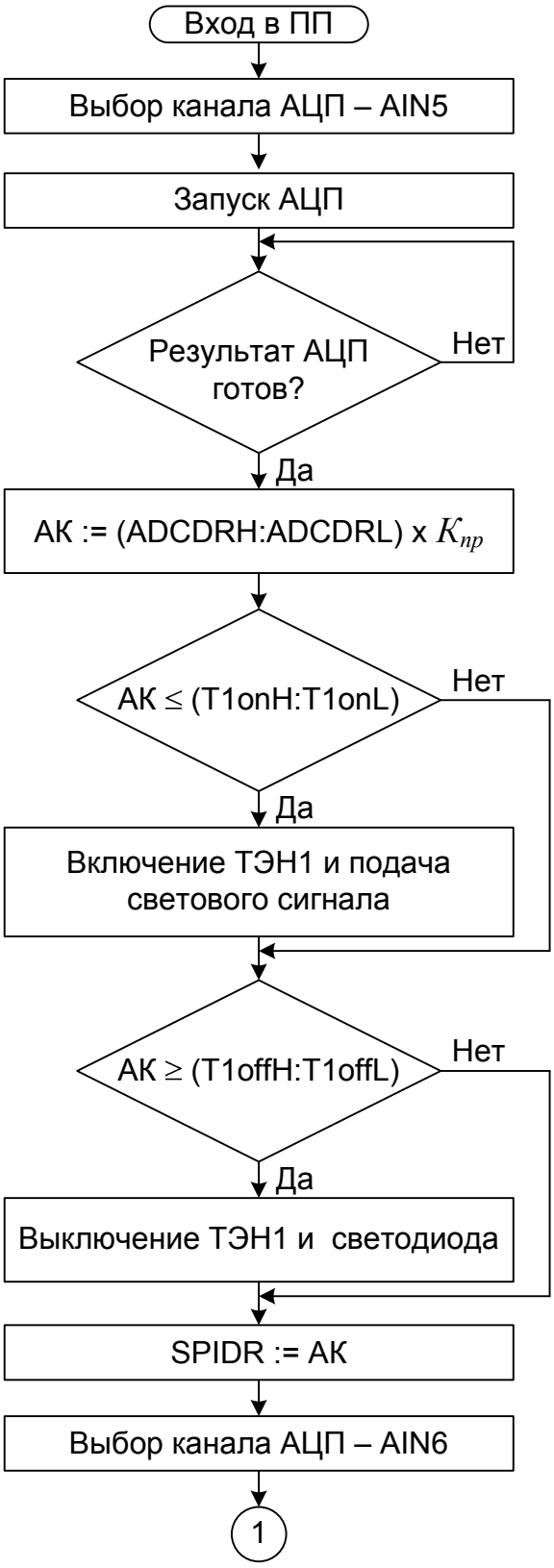


Рис. 4.17. Блок-схема подпрограммы-обработчика прерывания по переполнению таймера AT2 (начало)

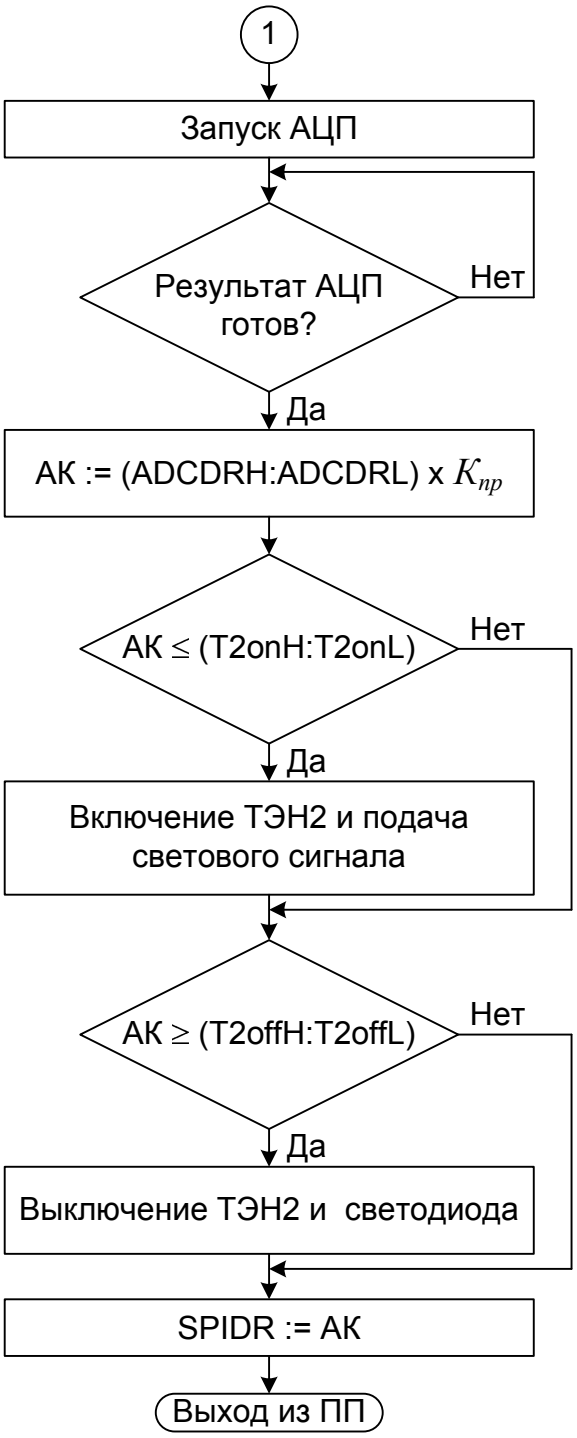


Рис. 4.18. Блок-схема подпрограммы-обработчика прерывания по переполнению таймера AT2 (окончание)

Величина коэффициента преобразования напряжения датчика в температуру $K_{пр}$ определяется следующим образом. Напряжение на выходе датчика при температуре 300 ℃ составляет 22,88 мВ (см. табл. 4.1). С учетом коэффициента усиления входного усилителя

$$K_{пр} = \frac{300°C}{K_{ус} \cdot 22{,}8\ \text{мВ}} = \frac{300°C}{205 \cdot 22{,}8\ \text{мВ}} = 0{,}064\ {}^{\circ}\text{C}/\text{мВ} = 64\ {}^{\circ}\text{C}/\text{В}.$$

В основной программе при нажатии клавиши $S2$ "Enter" осуществляется переход в пользовательское меню (рис. 4.19).

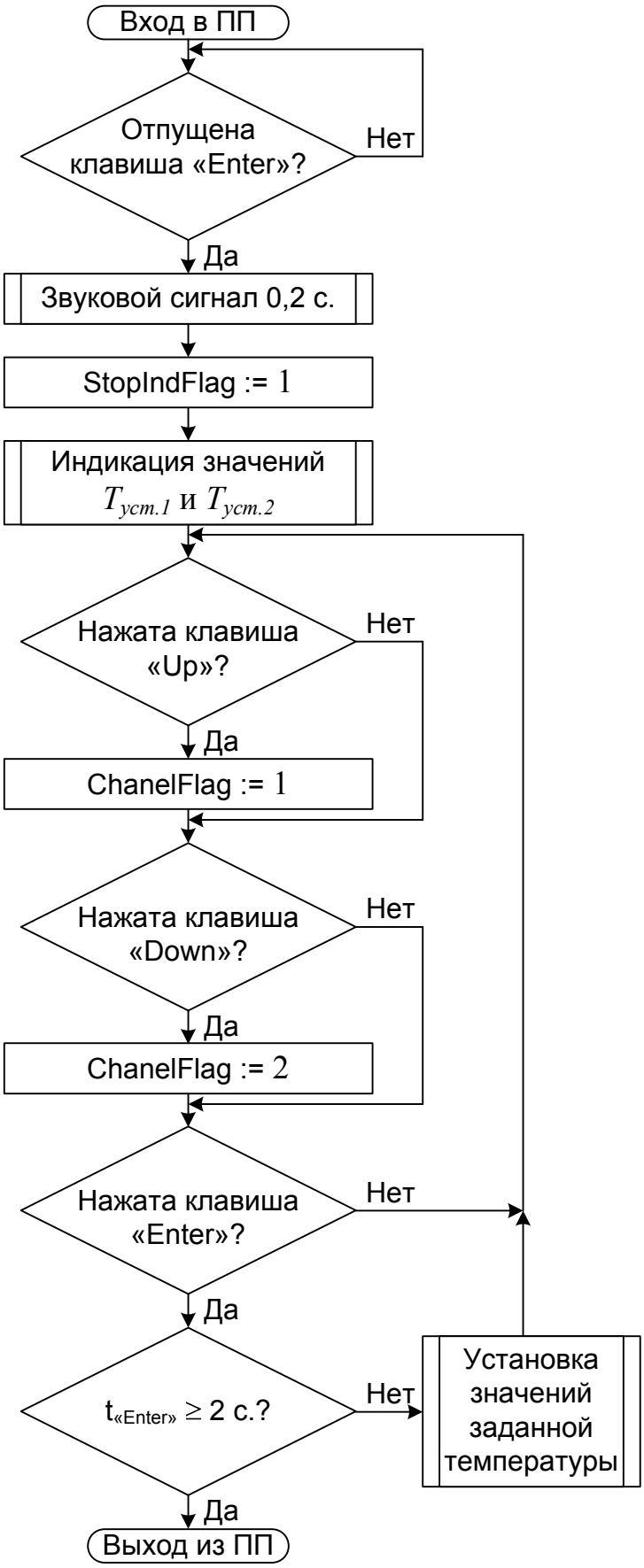


Рис. 4.19. Блок-схема подпрограммы пользовательского меню

В меню, с помощью клавиш *S1* "Up" и *S3* "Down" сначала выбирается канал, для которого устанавливается температура, а затем по нажатию клавиши *S2* "Enter" осуществляется переход в режим установки заданной температуры выбранного канала (рис. 4.20 – 4.21).

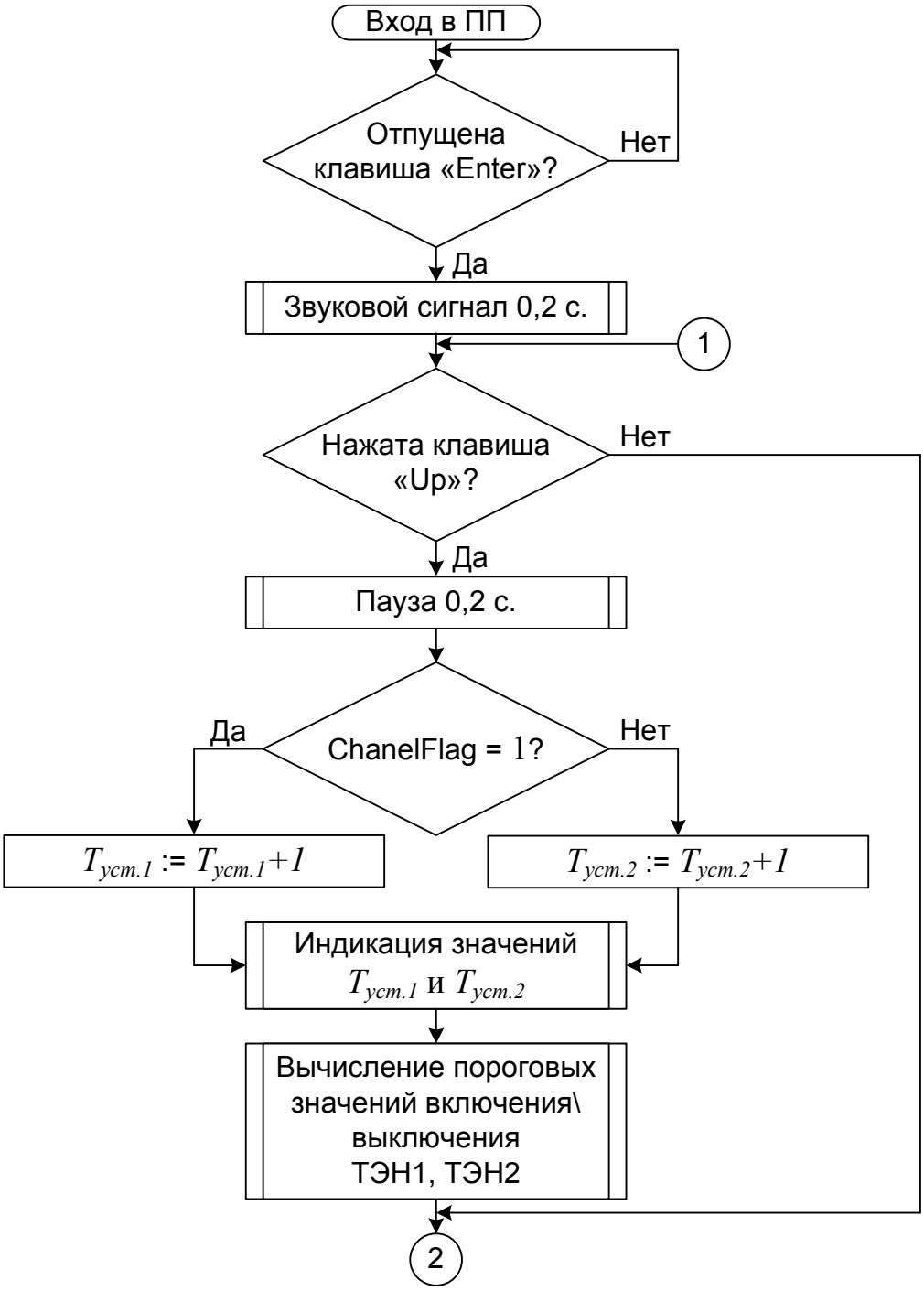


Рис. 4.20. Блок-схема подпрограммы установки заданной температуры (начало)

В этом режиме, как уже было сказано, индикация текущей температуры временно прекращается, индицируются значения устанавливаемой температуры по первому и второму каналам. Для запрета индикации текущей температуры используется пользовательский флаг.

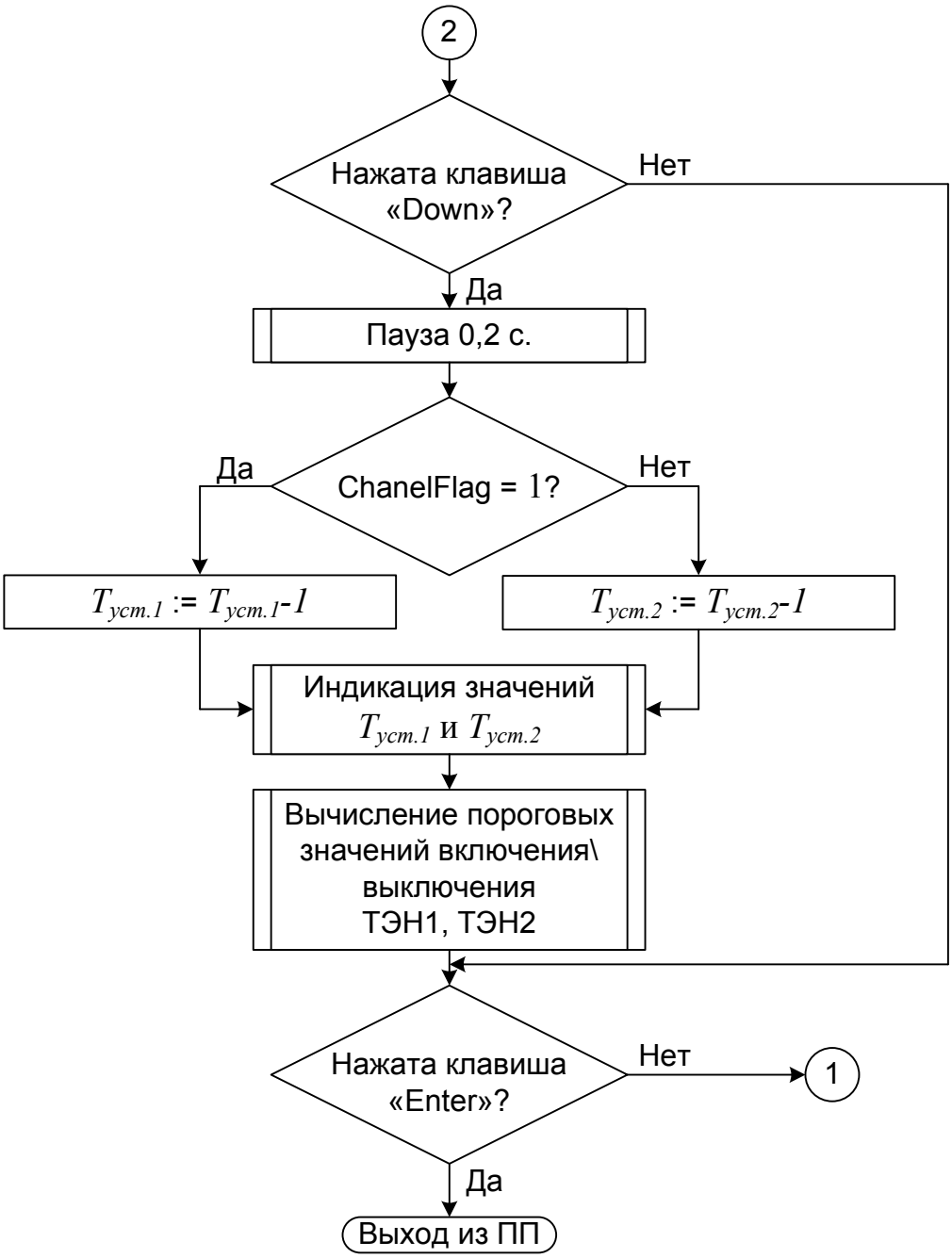


Рис. 4.21. Блок-схема подпрограммы установки заданной температуры (окончание)

Установка температуры выбранного канала осуществляется с помощью все тех же клавиш *S1* "Up" и *S3* "Down". При нажатии клавиши *S2* "Enter" устройство возвращается в пользовательское меню, выход из которого в основной режим возможен только при удерживании клавиши в течении 2 с. Этот временной интервал формируется методом программных циклов [2]. Переключение между режимами сопровождается звуковой сигнализацией пьезоизлучателя *B1*.

4.5. Тестирование и настройка прибора

Как уже упоминалось, главными недостатками при использовании в качестве температурных датчиков термопар являются их низкая повторяемость и возникновение паразитных термо-ЭДС, что приводит к необходимости настройки разработанного устройства (экспериментального определения и корректировки K_{np}). Кроме того, так как величина сигнала на выходе термопары составляет всего лишь десятки милливольт, а длина подводящих проводников - единицы метров, то в условиях воздействия промышленных помех имеет место существенное снижение отношения сигнал/шум на входе АЦП.

4.5.1. Выбор типа и параметров цифрового фильтра

Как показали испытания в реальных условиях, сигнал на входе АЦП микроконтроллера (рис. 4.22), записанный с помощью цифрового осциллографа, представляет собой смесь полезного сигнала и шума. Это приводит к «скачкам» в показаниях устройства, то есть к неправильному отображению текущей температуры нагревательной камеры, «дребезгу» исполнительных органов в области пороговых значений и соответственно к снижению ресурса, как электромеханических реле, так и термоэлектронагревателей.

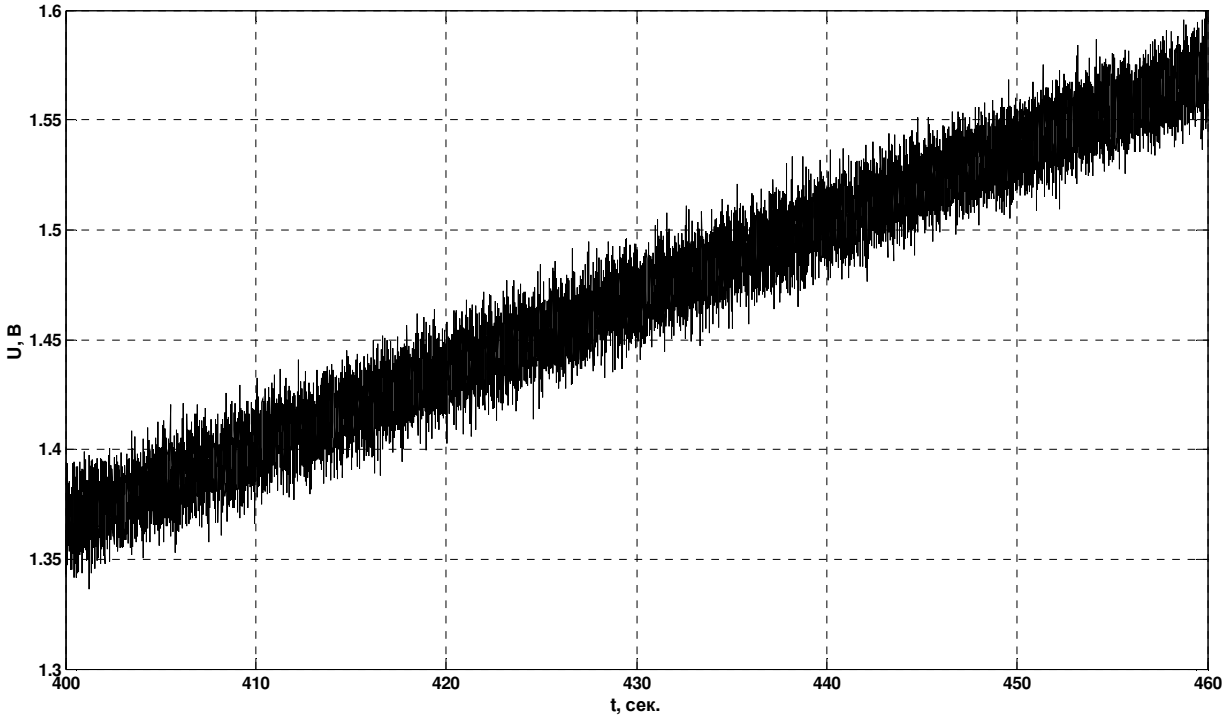


Рис. 4.22. Сигнал на входе одного из каналов АЦП микроконтроллера

Сигнал (рис. 4.22) соответствует диапазону температур нагревательной камеры от 100 °С до 115 °С (температура в нагревательной камере измерялась контрольным прибором).

Анализ полученного сигнала проведем в среде моделирования Matlab, с помощью программы (рис. 4.23).

Программа спектрального анализа и фильтрации сигнала температурного датчика терморегулятора

```
S=hex2dec('sig_AIN5.txt');    % чтение отсчетов сигнала
Кус=205;                      % коэффициент усиления усилителя
N=8;                          % порядок фильтра
fs=200;                       % частота дискретизации сигнала
t=400:1/fs:460;          % формирование отсчетов времени
f=0:fs/length(t):fs/2;        % формирование частотных отсчетов

%---------------------------------------------------------------
% Графическое представление сигнала
%---------------------------------------------------------------
plot(t,S,'Color','black','LineWidth',1.0);
axis([400 460 1.3 1.6])
grid on;
title('Сигнал на входе AIN5');
xlabel('t, сек.');
ylabel('U, В');

%---------------------------------------------------------------
% Графическое представление спектра сигнала
%---------------------------------------------------------------
Sf=fft(S,length(f));
Asf=Sf.*conj(Sf)/length(f);
f1=fs*(0:length(Sf)/2-1)/length(Sf);
figure;
plot(f1,Asf(1:length(f1)),'Color','black','LineWidth',1.5);
axis([0,110,0,0.025]);
grid on;
title('Спектр сигнала на входе AIN5');
xlabel('f, Гц');
ylabel('S(f), В/Гц');

%---------------------------------------------------------------
% Графическое представление спектра сигнала и АЧХ фильтра
%---------------------------------------------------------------

w=f./fs;
H=0.025*1/N*abs(sin(N*pi*w)./sin(pi*w));
hold on;
plot(f,H);
axis([0,110,0,0.025]);
grid on;
title('Спектр сигнала и АЧХ фильтра');

%---------------------------------------------------------------
% Фильтрация сигнала и вывод результата
%---------------------------------------------------------------

Num(1:N)=1/N;
```

```
Den=[1];
Y=filter(Num,Den,S);
figure;
plot(t,Y,'Color','black','LineWidth',1.0);
axis([400,460,1.3,1.6]);
grid on;
title('Сигнал после фильтрации');
xlabel('t, сек.');
ylabel('U, В');
```

Рис. 4.23. Программа спектрального анализа и фильтрации сигнала

Спектр указанного сигнала, построенный с использованием среды моделирования Matlab представлен на рис. 4.24.

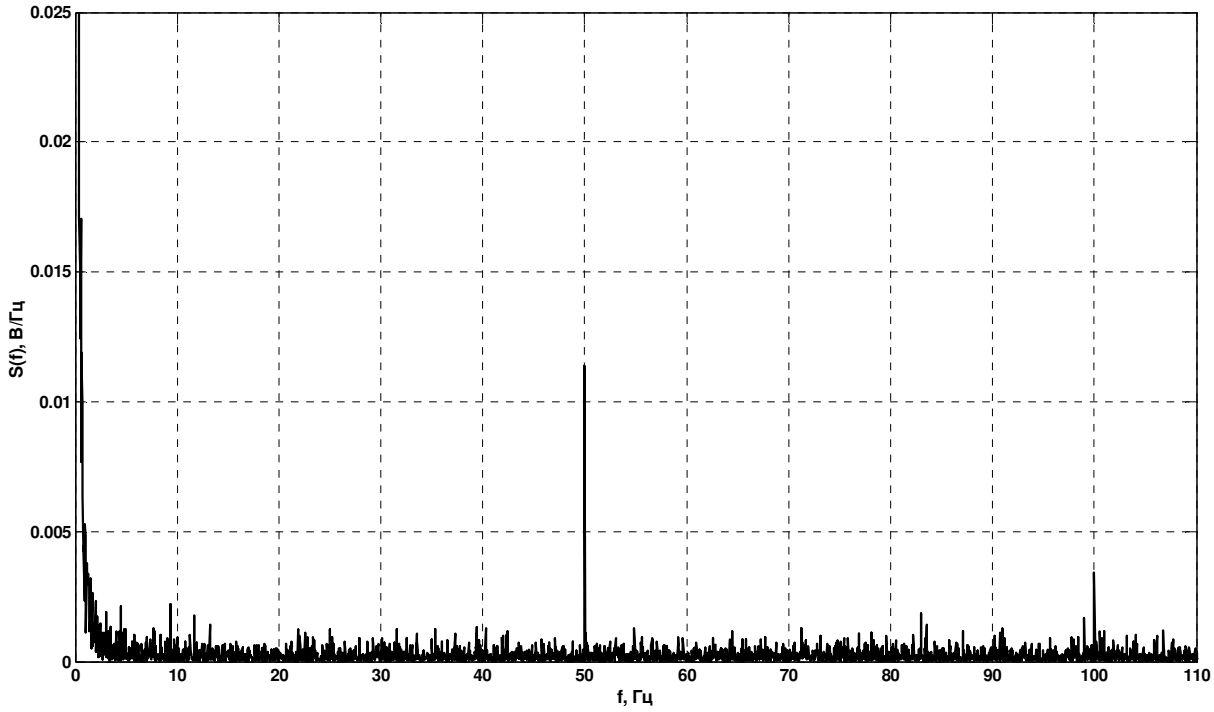


Рис. 4.24. Спектр сигнала на входе АЦП

Как видно из рис. 4.24, кроме полезной составляющей (начальная область спектра), несущей информацию об изменении температуры, спектр сигнала содержит также шумовую составляющую и составляющие на частотах 50 и 100 Гц, которые являются сетевыми помехами промышленных установок. Оптимальным путем устранения этих помех может быть использование цифрового усредняющего фильтра [2].

Этот фильтр представляет собой подвижное окно, которое сдвигается вдоль массива данных на один элемент с приходом каждого следующего отсчета сигнала (рис. 4.25).

Рис. 4.25. Пример 8-элементного усредняющего оконного фильтра

Центральный элемент заменяется средним значением всех элементов этого окна.

Амплитудно-частотная характеристика (АЧХ) такого фильтра описывается выражением:

$$H(f)=\frac{1}{N}\left|\frac{sin\left(\frac{N\pi f}{F_д}\right)}{\frac{\pi f}{F_д}}\right|, \qquad (4.13)$$

где f - текущее значение частоты;

N - порядок фильтра (количество элементов окна);

$F_д$ - частота дискретизации сигнала.

На рис. 4.26 представлена АЧХ 8-элементного ($N=8$) усредняющего оконного фильтра. Как видно из рисунка, подбирая параметры $F_д$ и N можно добиваться подавления гармоник на частотах $\frac{nF_д}{N}$ ($n=1,2,...,\frac{N}{2}$ - номер гармоники).

Достоинством оконного усредняющего фильтра является то, что для его реализации не требуются операции умножения. Для реализации оконного усредняющего фильтра с размером окна, равным N, на каждом такте требуется $N-1$ операция сложения и одна операция деления на N [2]. Уменьшить количество операций, а следовательно, повысить быстродействие фильтра можно, если использовать следующую особенность: для сдвига окна на одну позицию достаточно вычесть из текущей суммы значение N-элемента и прибавить значение нового элемента окна с последующим делением, вновь образовавшейся, суммы на N. Таким образом, вместо $N-1$ операций сложения на каждом такте можно использовать одну операцию вычитания и одну операцию сложения. Если N - четное, процедура деления может быть эффективно заменена операцией сдвига вправо.

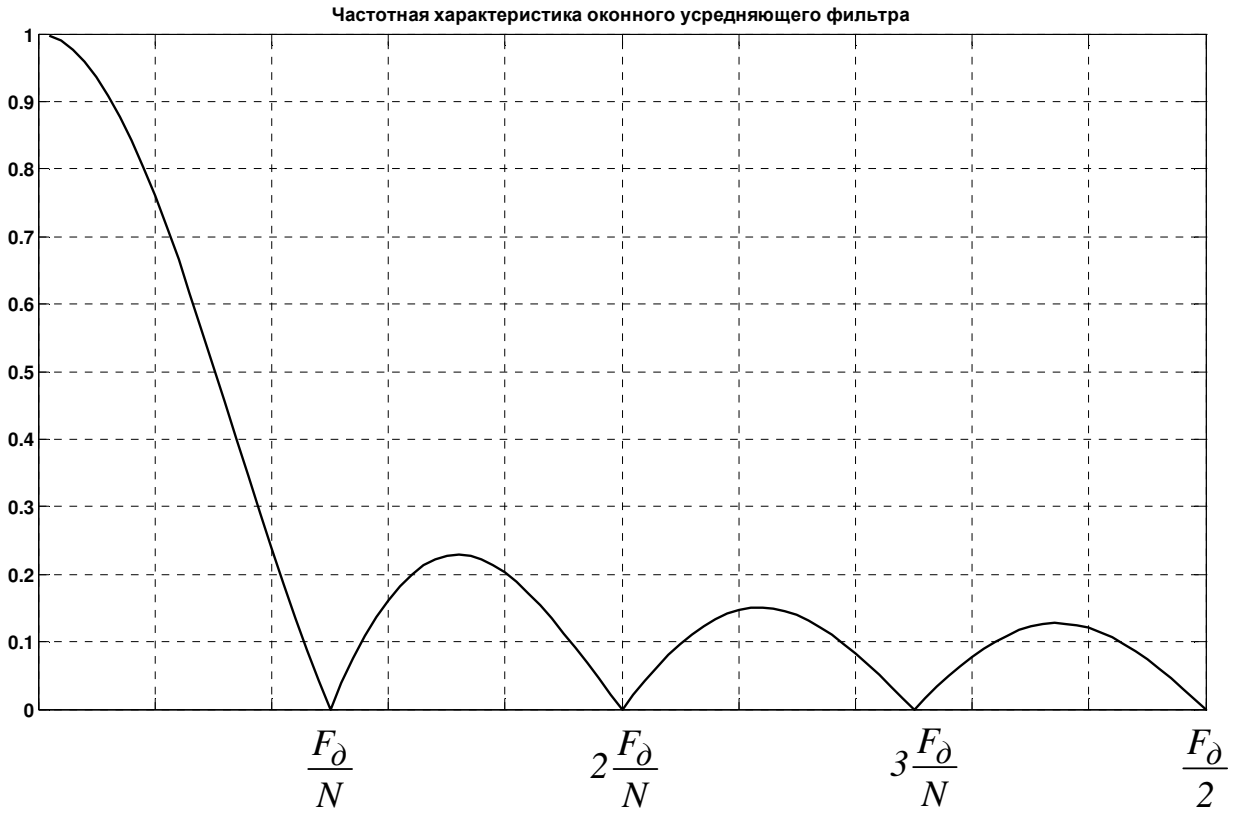


Рис. 4.26. АЧХ 8-элементного усредняющего оконного фильтра

Основываясь на приведенном выше, выберем $F_д = \frac{1}{T_д} = 200\,\text{Гц}$ и $N = 8$. Период дискретизации должен быть несколько скорректирован в сторону уменьшения - $T_д = 5\,\text{мс}$. Это значение удовлетворяет условию (4.12), но теперь количество счетных импульсов, которое должен подсчитать таймер, для формирования временного интервала $T_д$, равно 5, следовательно, в его регистр автозагрузки нужно записать число $2^{12} - 5 = 4091 = 0FFBh$.

АЧХ фильтра для выбранных параметров и результат моделирования процесса фильтрации сигнала (рис. 4.22) представлены на рис. 4.27 и рис. 4.28, соответственно.

Очевидно, что дисперсия шумов после фильтрации сигнала значительно меньше, чем у исходного сигнала (рис. 4.22).

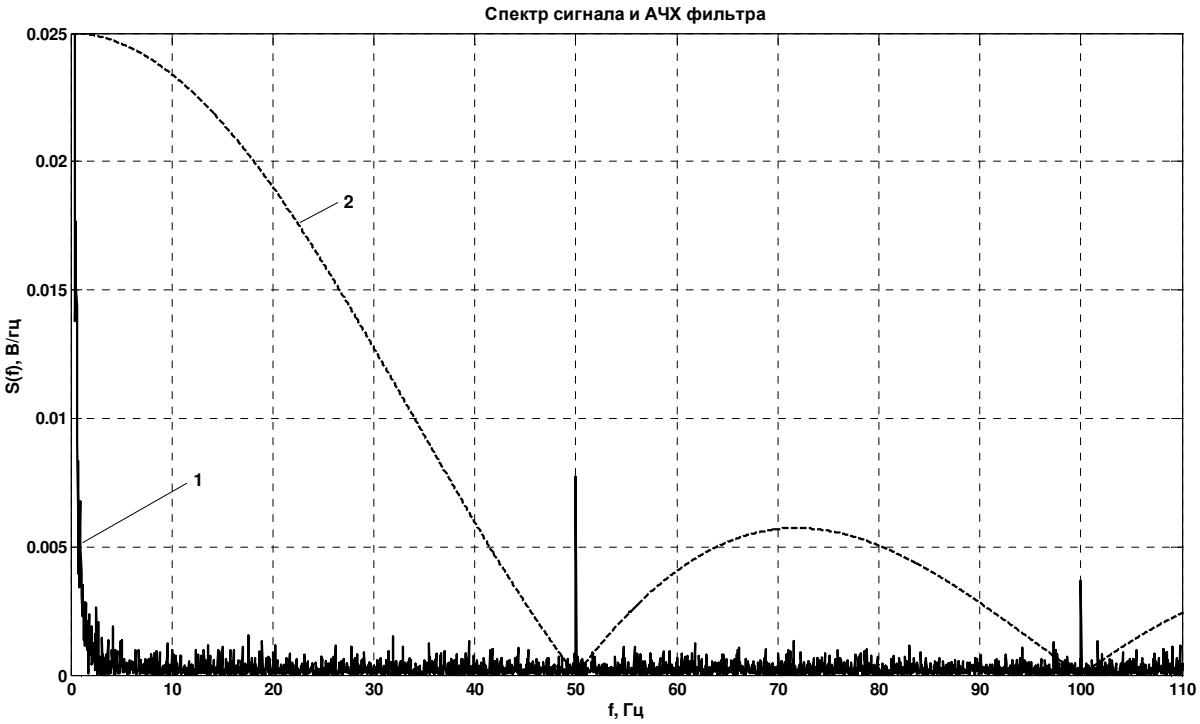


Рис. 4.27. Спектр сигнала (1) и АЧХ 8-элементного усредняющего оконного фильтра (2)

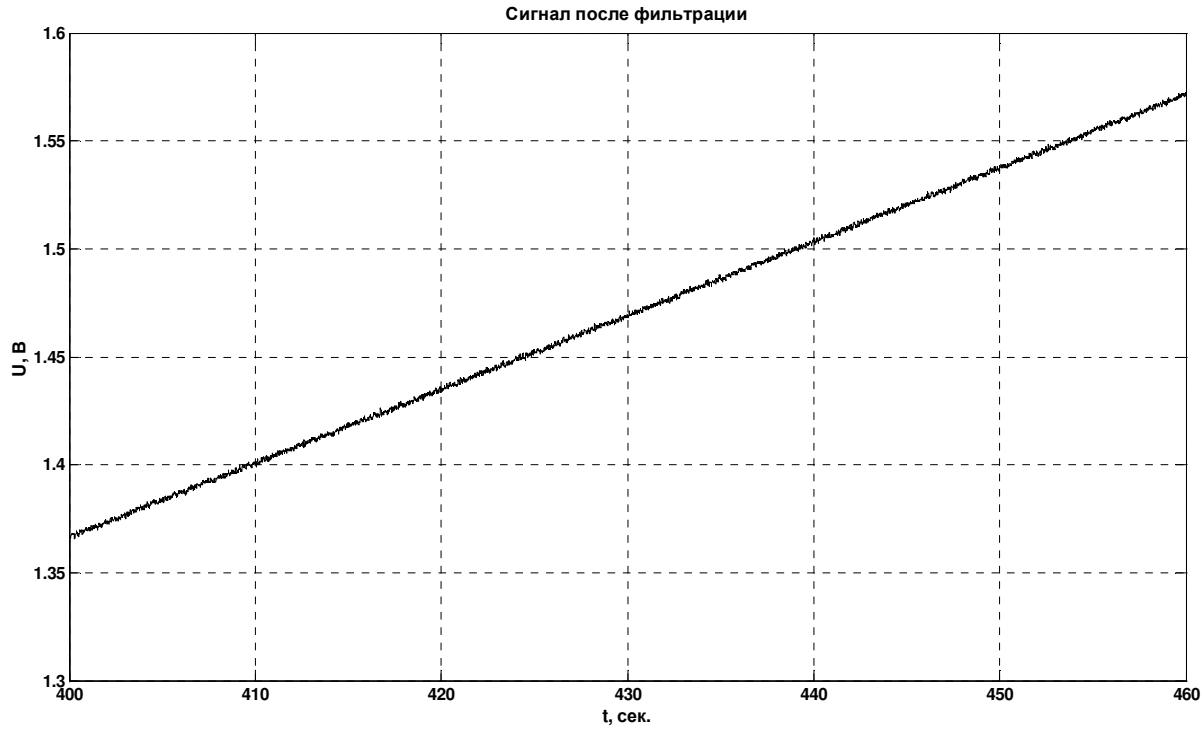


Рис. 4.28. Сигнал после фильтрации оконным фильтром

4.5.2. Модификация управляющей программы микроконтроллера

Применение цифрового фильтра требует корректировки программного обеспечения микроконтроллера. В блок-схеме (рис. 4.18) после блока готовности результата АЦП добавляется подпрограмма фильтрации сигнала температурного датчика.

Операция фильтрации может быть реализована согласно блок-схеме приведенной на рис. 4.29. Работа фильтра основана на циклическом буфере, который представляет собой набор шестнадцати ячеек ОЗУ (по две ячейки на один отсчет сигнала) содержащих текущие данные усредняющего окна. Для доступа к элементам окна используется регистр индексной адресации X.

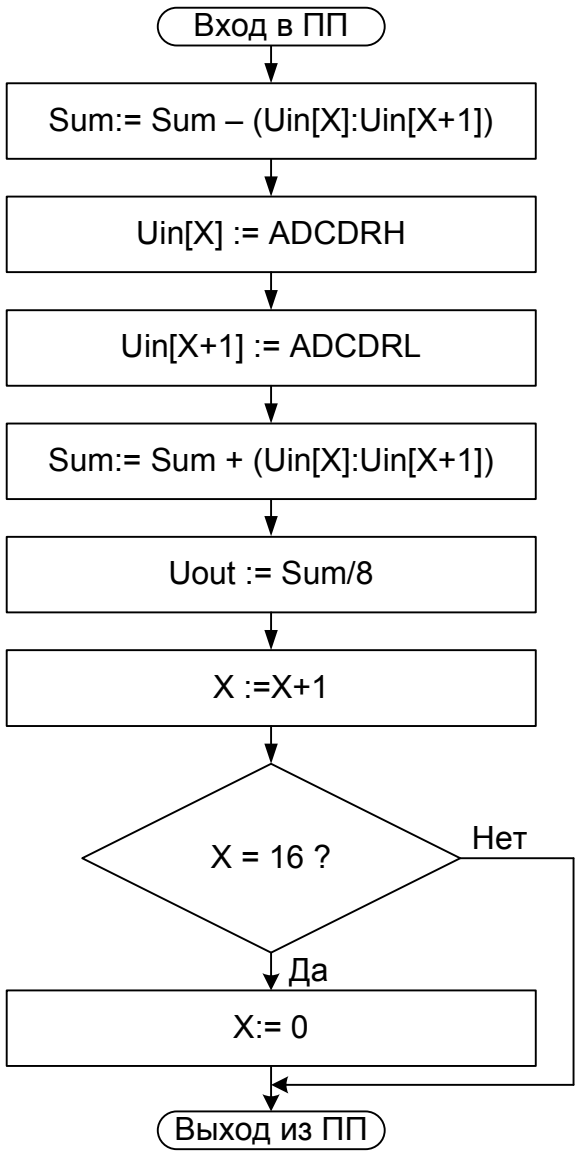


Рис. 4.29. Блок-схема подпрограммы фильтрации сигнала

Поскольку технологический разброс при изготовлении термопар и АЦП может приводить к искажению коэффициента преобразования

напряжения датчика в температуру, нужно определить его действительное значение $K_{пр\ действ.}$, с учетом действия цифрового фильтра. Напомним, что расчетный коэффициент преобразования равен $K_{пр} = 64\ {}^{\circ C}\!/_{\text{В}}$.

Сигнал после фильтрации (рис. 4.28) как и сигнал до фильтрации (рис. 4.22) соответствует диапазону температур от 100 ℃ до 115 ℃. Таким образом, реальное значение коэффициента преобразования равно $K_{пр\ действ.} = \dfrac{115°C - 100°C}{1{,}57\text{В} - 1{,}37\text{В}} = 75\ {}^{\circ C}\!/_{\text{В}}$.

Поскольку результат АЦП и соответственно результат фильтрации измеряется не в вольтах, а в уровнях квантования относительно опорного напряжения $U_{оп} = U_{пит} = 5B$, то в программе при умножении результата АЦП следует использовать значение коэффициента преобразования в этих единицах измерения:

$$K_{пр} = \frac{K_{пр\ действ.} \cdot U_{оп}}{2^{10}} = \frac{75\ {}^{\circ C}\!/_{\text{В}} \cdot 5B}{2^{10}} = 0{,}3662\ {}^{\circ C}\!/_{\text{ур. кв.}}.$$

После корректировки программы осуществляем проверку показаний разработанного устройства и их сравнение с показаниями контрольных приборов. Они не должны отличаться более, чем на ± 1˚С – величину точности измерения температуры заданную в ТЗ.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Укажите критерии выбора микроконтроллера при разработке электронного устройства.
2. Для чего используется регистр индексной адресации X?
3. В чем разница между статическим и динамическим режимами индикации? Поясните достоинства и недостатки этих режимов.
4. Объясните принцип действия цифрового усредняющего фильтра.
5. Поясните принцип действия и назначение таймеров-счетчиков микроконтроллера ST7FLITE19.

Заключение

Данная книга является первым изданием в Украине, где в систематизированном виде изложены возможности микроконтроллерных технологий STMicroelectronics и приведены результаты применения микроконтроллера ST7 для проектирования и отладки цифровых систем контроля и управления.

В соответствии с целью, которая была поставлена, в пособии дан обзор семейства микроконтроллеров ST7, описаны особенности разработки программного обеспечения и, что наиболее важно, по нашему мнению, приведены примеры создания реальных систем, начиная от анализа технического задания и заканчивая некоторыми вопросами тестирования и настройки приборов.

Следует подчеркнуть, что авторы не ставили перед собой задачу описать все аспекты разработки профессионального проекта системы. Часть традиционных и обязательных вопросов, относящихся к такой разработке, остались не раскрытыми. Это касается детальной оценки технических характеристик (энергопотребления, быстродействия, надежности и др.), отладки и испытаний программных средств и системы в целом и др. Указанные вопросы изложены в других изданиях, которыми может быть дополнен список литературы. В материалах [2,4,5,10] есть ответы на часть из них.

8-разрядный микроконтроллер ST7 открывает линейку решений, которая к началу написания книги пополнилась семейством 32-разрядных микроконтроллеров на базе ядра ARM Cortex M3. Это решение позволяет значительно повысить производительность и уменьшить энергопотребление. При этом цена такого микроконтроллера сопоставима со стоимостью 16-разрядных микроконтроллеров. Разработчикам предоставляется широкий набор программных и аппаратных отладочных средств, что, несомненно, позволит им уже в ближайшее время использовать в своих проектах новую платформу.

Авторы планируют посвятить этому микроконтроллеру и примерам его использования следующую работу в рамках сотрудничества STM – ХАИ и других университетов Украины.

Знания и опыт, которые получают участники этой перспективной программы, послужат успешному международному образовательному и научно-техническому сотрудничеству университетов и крупнейших компаний – лидеров мировой компьютерной индустрии. На повестке дня - создание университетского технопарка, одним из базовых компонентов которого мог бы стать центр «STM – ХАИ».

ПРИЛОЖЕНИЕ 1.
СИСТЕМА КОМАНД МИКРОКОНТРОЛЛЕРОВ СЕРИИ ST7

Общее описание. Система команд ST7 предоставляет большие возможности обработки данных, реализацию арифметических, логических операций, команды вызова подпрограмм и возврата, безусловного и условного переходов, команды обработки отдельных бит и пересылки данных, а также обеспечивает управление в режиме реального времени. ST7 имеют некоторое сходство с микроконтроллерами фирмы MOTOROLA (например MC68HC05/08) и могут быть ориентированы на те же области применения.

Процессор ST7 содержит регистр-аккумулятор, два индексных регистра X и Y, регистр состояния процессора, счетчик команд, адресующий 64 Кбайт памяти, 16-разрядный указатель стека, в котором доступен только младший байт (остальные 8 бит зарезервированы и аппаратно устанавливаются в “1”). В зависимости от версии микроконтроллера, максимальный размер стека составляет 64 или 256 байт.

Система команд включает в себя 63 инструкции длиной от 1 до 4 байт. Специальный префиксный байт, употребляемый в некоторых командах, служит для расширения стандартной 256-байт сетки команд 8-разрядных МК. Минимальное время исполнения 1-байтной команды составляет 250 нс при внутренней тактовой частоте 8 МГц.

Ниже в описании машинных команд микроконтроллера ST7 будут использоваться некоторые условные обозначения. Перечислим их:

Режимы адресации:

1 – безадресный режим (inherent);
2 – непосредственный (immediate);
3 – прямой (direct);
4 – индексированный (indexed);
5 – косвенный (indirect);
6 – относительный (relative);
s – короткий (short);
w – длинный (long);

Флаги:

H – полуперенос (half carry bit);
I – маска прерываний (interrupt mask);
N – отрицательный (negative);
Z – ноль (zero);
C – перенос (carry/borrow);

Переменные:

d	–	приемник (destination);
s	–	источник (source);
SP	–	указатель стека (stack pointer);
PC	–	счетчик команд (program counter);
CC	–	регистр кодов признаков (conditional code register).

Алфавитный перечень команд:

ADC

Название: Add With Carry
Синтаксис: **adc** d, s
Операция: d <= d + s + C
Источник: память
Приемник: аккумулятор
Влияние на флаги: H, N, Z, C
Режимы адресации: 2, 3, 4, 5, s, w
Описание: содержимое регистра “s” и флаг переноса прибавляется к содержимому регистра “d”

ADD

Название: Add Without Carry
Синтаксис: **add** d, s
Операция: d <= d + s
Источник: память
Приемник: аккумулятор
Влияние на флаги: H, N, Z, C
Режимы адресации: 2, 3, 4, 5, s, w
Тактовых циклов: 1
Описание: содержимое регистра “s” прибавляется к содержимому регистра “d”

AND

Название: Logical AND
Синтаксис: **and** d, s
Операция: d <= d AND s
Источник: память
Приемник: аккумулятор
Влияние на флаги: N, Z
Режимы адресации: 2, 3, 4, 5, s, w
Описание: связывание логической операцией “И” содержимого регистров “d” и “s” с последующим сохранением результата в “d”

BCP

Название: Bit compare A, mem
Синтаксис: **bcp** s, d
Операция: (N, Z) <= s AND d
Источник: память
Приемник: аккумулятор
Влияние на флаги: N, Z
Режимы адресации: 2, 3, 4, 5, s, w
Описание: битовое сравнение

BRES

Название: Bit Reset in d
Синтаксис: **bres** d, b
Операция: d <= d AND (2^b)
Источник: константа
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 4, 5, s
Описание: сброс разряда “b” в “d”

BSET

Название: Bit Set in d
Синтаксис: **bset** d, b
Операция: d <= d OR (2^b)
Источник: константа
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 4, 5, s
Описание: установка разряда “b” в “d”

BTJF

Название: Bit Test Jump if bit is False (0)
Синтаксис: **btjf** d, b, rel
Операция: IF (d AND (2^b)) = 0, то PC <= PC + rel
Источник: константа
Приемник: память
Влияние на флаги: C
Режимы адресации: 3, 4, 5, s
Описание: переход, если разряд “b” в “d” равен “0”

BTJT

Название: Bit Test Jump if bit is True (1)
Синтаксис: **btjt** d, b, rel
Операция: IF (d AND (2^b)) ≠ 0, то PC <= PC + rel
Источник: константа

Приемник: память
Влияние на флаги: C
Режимы адресации: 3, 4, 5, s
Описание: переход, если разряд “b” в “d” не равен “0”

CALL

Название: Direct Subroutine Call
Синтаксис: **call** d
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 4, 5, s, w
Описание: вызов подпрограммы по абсолютному адресу “d”; адрес возврата – команды, следующей после команды call – сохраняется в стеке

CALLR

Название: Call Subroutine Relative
Синтаксис: **callr** d
Источник: адрес
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 5, 6, s
Описание: вызов подпрограммы по короткому (short) адресу “d”

CLR

Название: Clear d
Синтаксис: **clr** d
Операция: d <= 0
Источник: регистр, память
Приемник: регистр, память
Влияние на флаги: N = 0, Z = 1
Режимы адресации: 1, 3, 4, 5, s
Описание: обнуление “d”

CP

Название: Arithmetic Compare
Синтаксис: **cp** d, s
Операция: { N, Z, C } = TEST(d - s)
Источник: память
Приемник: регистр
Влияние на флаги: N, Z, C
Режимы адресации: 2, 3, 4, 5, s, w

Описание: сравнение “d” и “s” при этом “d” вычисляется из “s” без сохранения результата (содержимое “d” не изменяется)

CPL

Название: Logical Complement of d
Синтаксис: **cpl** d
Операция: d <= d XOR FF
Источник: регистр, память
Приемник: регистр, память
Влияние на флаги: N, Z, C = 1
Режимы адресации: 1, 3, 4, 5, s
Описание: побитовая инверсия “d”

DEC

Название: Decrement d
Синтаксис: **dec** d
Операция: d <= d – 1
Источник: регистр, память
Приемник: регистр, память
Влияние на флаги: N, Z
Режимы адресации: 1, 3, 4, 5, s
Описание: из содержимого “d” вычитает “1”; флаг переноса не изменяется, благодаря чему “d” может использовать в качестве счетчика цикла при выполнении арифметических операций над несколькими байтами

HALT

Название: Halt
Синтаксис: **halt**
Влияние на флаги: I = 0
Режимы адресации: 1
Описание: останов программы и переход в “спящий” режим энергосбережения

INC

Название: Increment d
Синтаксис: **inc** d
Операция: d <= d + 1
Источник: регистр, память
Приемник: регистр, память
Влияние на флаги: N, Z
Режимы адресации: 1, 3, 4, 5, s

Описание: к содержимому “d” прибавляется “1”; флаг переноса не изменяется, благодаря чему “d” может использовать в качестве счетчика цикла при выполнении арифметических операций над несколькими байтами

IRET

Название: Interrupt routine Return
Синтаксис: **iret**
Операция: POP CC, A, X, PC
Влияние на флаги: H, I, N, Z, C
Режимы адресации: 1
Описание: возврат из подпрограммы обработки прерывания

JP

Название: Absolute Jump
Синтаксис: **jp** d
Операция: PC <= d
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 4, 5, s, w
Описание: безусловный переход по абсолютному адресу “d”

JRA

Название: Jump Relative Always
Синтаксис: **jra** d
Операция: PC <= PC + d
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 5, 6
Описание: безусловный переход по относительному адресу текущего счетчика команд + “d”

JRT

Название: Jump Relative if True
Синтаксис: **jrt** d
Операция: PC <= PC + d
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 5, 6
Описание: условный переход по относительному адресу

JRF

Название: Jump Relative if False (never jump)
Синтаксис: **jrf** d
Операция: condition false
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 5, 6
Описание: никогда не переходит

JRIH

Название: Jump if Port INT pin = 1
Синтаксис: **jrih** d
Операция: (no port interrupts)
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 5, 6
Описание: переход по адресу “b” если порт INT pin = 1

JRIL

Название: Jump if Port INT pin = 0
Синтаксис: **jril** d
Операция: (port interrupt)
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 5, 6
Описание: переход по адресу “b” если порт INT pin = 0

JRH

Название: Jump if H = 1
Синтаксис: **jrh** d
Операция: IF H = 1 ?
Источник: Отсутствует
Приемник: Память
Влияние на флаги: Нет
Режимы адресации: 3, 5, 6
Описание: переход, если флаг “H” равен “1”

JRNH

Название: Jump if H = 0
Синтаксис: **jrnh** d
Операция: IF H = 0 ?
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 5, 6

Описание: переход, если флаг “H” равен “0”

JRM

Название: Jump if I = 1
Синтаксис: **jrm** d
Операция: IF I = 1 ?
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 5, 6
Описание: переход, если флаг “I” равен “1”

JRNM

Название: Jump if I = 0
Синтаксис: **jrnm** d
Операция: IF I = 0 ?
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 5, 6
Описание: переход, если флаг “I” равен “0”

JRMI

Название: Jump if N = 1
Синтаксис: **jrmi** d
Операция: IF N = 1 ? (minus)
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 5, 6
Описание: переход, если флаг “N” равен “1”

JRPL

Название: Jump if N = 0
Синтаксис: **jrpl** d
Операция: IF N = 0 ? (plus)
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 5, 6
Описание: переход, если флаг “N” равен “0”

JREQ

Название: Jump if Z = 1
Синтаксис: **jreq** d
Операция: IF Z = 1 ? (equal)
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 5, 6
Описание: переход, если флаг “Z” равен “1”

JRNE

Название: Jump if Z = 0
Синтаксис: **jrne** d
Операция: IF Z = 0 ? (not equal)
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 5, 6
Описание: переход, если флаг “Z” равен “0”

JRC

Название: Jump if C = 1
Синтаксис: **jrc** d
Операция: IF C = 1 ?
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 5, 6
Описание: переход, если флаг “C” равен “1”

JRNC

Название: Jump if C = 0
Синтаксис: **jrnc** d
Операция: IF C = 0 ?
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 5, 6
Описание: переход, если флаг “C” равен “0”

JRULT

Название: Jump if C = 1
Синтаксис: **jrult** d
Операция: Jmp IF unsigned <
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 5, 6
Описание: переход, если меньше (беззнаковое)

JRUGE

Название: Jump if C = 0
Синтаксис: **jruge** d
Операция: Jmp IF unsigned ≥
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 5, 6
Описание: переход, если больше или равно (беззнаковое)

JRUGT

Название: Jump if (C + Z = 0)
Синтаксис: **jrugt** d
Операция: Jmp IF unsigned >
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 5, 6
Описание: переход, если больше (беззнаковое)

JRULE

Название: Jump if (C + Z = 1)
Синтаксис: **jrule** d
Операция: Jmp IF unsigned ≤
Приемник: память
Влияние на флаги: нет
Режимы адресации: 3, 5, 6
Описание: переход, если меньше или равно (беззнаковое)

LD

Название: Load s in d
Синтаксис: **ld** d, s
Операция: d <= s
Источник: память, регистр
Приемник: регистр, память
Влияние на флаги: N, Z
Режимы адресации: 1, 2, 3, 4, 5, s, w
Описание: команда пересылки байта из ячейки памяти в аккумулятор и наоборот

MUL

Название: Multiply d by s
Синтаксис: **mul** d, s
Операция: d:s <= d * s
Источник: X, Y, A (аккумулятор)
Приемник: A (аккумулятор), X, Y
Влияние на флаги: H = 0, C = 0
Режимы адресации: 1
Описание: перемножение множителя “s” и множителя “d”

NEG

Название: Negate d (logical 2-complement)
Синтаксис: **neg** d
Операция: d <= (d XOR FF) + 1, или d <= 0 - d
Приемник: регистр, память

Влияние на флаги: N, Z, C
Режимы адресации: 1, 3, 4, 5, s
Описание: изменение знака содержимого “d”

NOP

Название: No operation
Синтаксис: **nop**
Операция: нет
Влияние на флаги: нет
Режимы адресации: 1
Описание: пустая команда для задержки на один такт программы

OR

Название: Logical OR
Синтаксис: **or** d, s
Операция: d <= d OR s
Источник: память
Приемник: аккумулятор
Влияние на флаги: N, Z
Режимы адресации: 2, 3, 4, 5, 6, s, w
Описание: объединение логической операцией “ИЛИ” содержимого регистров “d” и “s” с последующим сохранением результата в “d”

POP

Название: Pop from the Stack
Синтаксис: **pop** d
Операция: d <= (++SP)
Приемник: регистр, CC
Влияние на флаги: H, I, N, Z, C
Режимы адресации: 1
Описание: указатель стека увеличивается на “1”, после чего значение в памяти, на которое указывает этот указатель, записывается в “d”

PUSH

Название: Push onto the Stack
Синтаксис: **push** d
Операция: (SP--) <= d
Источник: регистр, CC
Влияние на флаги: нет
Режимы адресации: 1

Описание: содержимое “d” сохраняется в ячейки памяти, адресуемой с помощью указателя стека, после чего этот указатель уменьшается на “1”

RCF

Название: Reset carry flag
Синтаксис: **rcf**
Операция: C = 0
Влияние на флаги: C = 0
Режимы адресации: 1
Описание: обнуление флага переноса

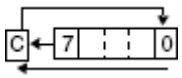
RET

Название: Subroutine return
Синтаксис: **ret**
Операция: MSB(PC) <= (++SP)
LSB(PC) <= (++SP)
Влияние на флаги: нет
Режимы адресации: 1
Описание: вызов подпрограммы, адрес возврата которой должен находиться в стеке

RIM

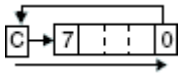
Название: Reset interrupt mask
Синтаксис: **rim**
Операция: I = 0
Влияние на флаги: I = 0
Режимы адресации: 1
Описание: сброс маски прерывания

RLC

Название: Rotate Left through Carry
Синтаксис: **rlc** d
Операция: 

Приемник: регистр, память
Влияние на флаги: N, Z, C
Режимы адресации: 1, 3, 4, 5, s
Описание: все разряды “d” сдвигаются на одну позицию влево; флаг “C” копируется в младший разряд, а старший разряд – в “C”

RRC

Название: Rotate Right through Carry
Синтаксис: **rrc** d
Операция: 

Приемник: регистр, память
Влияние на флаги: N, Z, C
Режимы адресации: 1, 3, 4, 5, s
Описание: все разряды “d” сдвигаются на одну позицию вправо; “C” копируется в старший разряд, а младший разряд – в “C”

RSP

Название: Reset Stack pointer
Синтаксис: **rsp**
Операция: SP <= Reset Value
Влияние на флаги: нет
Режимы адресации: 1
Описание: сброс указателя стека

SBC

Название: Subtract s from d with carry
Синтаксис: **sbc** d, s
Операция: d <= d – s - C
Источник: память
Приемник: аккумулятор
Влияние на флаги: N, Z, C
Режимы адресации: 2, 3, 4, 5, s, w
Описание: содержимое “s” и флаг переноса вычитаются из содержимого “d”

SCF

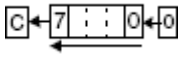
Название: Set carry flag
Синтаксис: **scf**
Операция: C = 1
Влияние на флаги: C = 1
Режимы адресации: 1
Описание: установка флага переноса

SIM

Название: Set interrupt mask
Синтаксис: **sim**
Операция: I = 1
Влияние на флаги: I = 1
Режимы адресации: 1

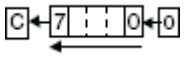
Описание: установка маски прерывания

SLA

Название: Shift left arithmetic (equal to SLL d = 1)
Синтаксис: **sla** d
Операция: 

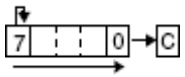
Источник: отсутствует
Приемник: регистр, память
Влияние на флаги: N, Z, C
Режимы адресации: 1, 3, 4, 5, s
Описание: все разряды “d” сдвигаются на одну позицию влево; старший разряд копируется в “C”, а младший разряд заполняется “0”

SLL

Название: Shift left logical
Синтаксис: **sll** d
Операция: 

Источник: отсутствует
Приемник: регистр, память
Влияние на флаги: N, Z, C
Режимы адресации: 1, 3, 4, 5, s
Описание: все разряды “d” сдвигаются на одну позицию влево; старший разряд копируется в “C”, а младший разряд заполняется “0”

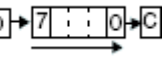
SRA

Название: Shift right arithmetic (equal to SLL one)
Синтаксис: **sra** d
Операция: 

Источник: отсутствует
Приемник: регистр, память
Влияние на флаги: N, Z, C
Режимы адресации: 1, 3, 4, 5, s
Описание: все разряды “d” сдвигаются на одну позицию вправо; старший разряд не меняет свое значение, а младший разряд копируется в “C”

SRL

Название: Shift right logical
Синтаксис: **srl** d

Операция: 

Источник: отсутствует
Приемник: регистр, память
Влияние на флаги: N = 0, Z, C
Режимы адресации: 1, 3, 4, 5, s
Описание: все разряды “d” сдвигаются на одну позицию вправо; старший разряд заполняется “0”, а младший разряд копируется в “C”

SUB

Название: Subtract s from d
Синтаксис: **sub** d, s
Операция: d <= d – s
Источник: память
Приемник: аккумулятор
Влияние на флаги: N, Z, C
Режимы адресации: 2, 3, 4, 5, s, w
Описание: содержимое “s” вычитается из содержимого “d”

SWAP

Название: Swap nibbles
Синтаксис: **swap** d
Операция: d(7:4) <=> d(3:0)
Приемник: регистр, память
Влияние на флаги: N, Z
Режимы адресации: 1, 3, 4, 5, s
Описание: старший и младший полубайты “d” меняются местами

TNZ

Название: Test for Neg & Zero
Синтаксис: **tnz** d
Операция: { N, Z } = TEST(d)
Приемник: регистр, память
Влияние на флаги: N, Z
Режимы адресации: 1, 3, 4, 5, s
Описание: проверяет, является ли содержимое “d” нулевым или отрицательным; само содержимое “d” остается неизменным

TRAP

Название: Software trap

Синтаксис: `trap`
Влияние на флаги: I = 1
Режимы адресации: 1
Описание: программное прерывание

WFI

Название: Wait for interrupt
Синтаксис: `wfi`
Влияние на флаги: I = 0
Режимы адресации: 1
Описание: ожидает прерывания

XOR

Название: Exclusive OR (d with s)
Синтаксис: `xor d, s`
Операция: d <= d XOR s
Источник: память
Приемник: аккумулятор
Влияние на флаги: N, Z
Режимы адресации: 2, 3, 4, 5, s, w
Описание: поразрядное объединение содержимого “s” и “d” логической операцией “ИСКЛЮЧАЮЩЕЕ ИЛИ” с сохранением результата в “d”

ПРИЛОЖЕНИЕ 2.

ДИРЕКТИВЫ ПРЕПРОЦЕССОРА

Общее описание. Директивы препроцессора – это инструкции препроцессору (табл. П2.1). Препроцессор – это текстовый процессор, который манипулирует текстом исходного файла на первой фазе компиляции. Директивы препроцессора обычно используются для облегчения внесения изменений в исходные программы, а так же для настройки и определения параметров среды. Расположенные в исходном файле директивы заставляют препроцессор выполнять конкретные действия. Например, препроцессор может заменить лексемы в тексте, вставить содержимое других файлов в исходный файл или подавить компиляцию части файла, удаляя сегменты текста.

Таблица П2.1

ТАБЛИЦА ДИРЕКТИВ ПРЕПРОЦЕССОРА

Директива	Синтаксис	Описание
.BELL	**.BELL**	звуковой сигнал на консоли
BYTE	**BYTE** <выражение или “строка”>, [,<выражение или “строка”>...]	определить байт в объектном коде
BYTES	**BYTES**	определение типа метки: тип = байт
CEQU	метка **CEQU** <выражение>	приравнять уже существующую метку к выражению
.CTRL	**.CTRL** <код>[,<код>]...	послать контрольные коды на принтер
DATE	**DATE**	определить 12-байтовую дату ASCII в объектный код
DC.B	**DC.B** <выражение или “строка”>, [,<выражение или “строка”>]	определить байт(ы) в объектном коде
DC.W	**DC.W** <выражние> [,<выражение>...]	определить слово(а) в объектном коде
DC.L	**DC.L** <выражение> [,<выражение>...]	определить длинные слова в объектном коде

Директива	Синтаксис	Описание
#DEFINE	**#DEFINE** <ИМЯ_КОНСТАНТЫ> <ЗНАЧЕНИЕ>	определить именованную константу
DS.B	**DS.B** [опционально количество байт]	определить в объектном коде пространство размером byte
DS.W	**DS.W** [опционально количество слов]	определить в объектном коде пространство размером word = 2 байта
DS.L	**DS.L** [опционально количество длинных слов]	определить в объектном коде пространство размером long = 4 байта
END	**END**	конец исходного кода
EQU	метка **EQU** <ВЫРОЖЕНИЯ>	приравнять метку к выражению
EXTERN	**EXTERN**	определить внешние метки
#ELSE	**#ELSE**	условное ELSE
#ENDIF	**#ENDIF**	признак конца условия
FCS	**FCS** <“строка”> |<число> [<“строка”> |<число>]...	составить константную строку
.FORM	**.FORM** <выражение>	установить длину строки листинга
GROUP	**GROUP** <выражение>	область имен исходного кода
#IF	**#IF** <выражение>	начать условную сборку
#IF1	**#IF1**	условие IF, которое должно выполняться в проходе #1, чтобы быть истинным
#IF2	**#IF2**	условие IF, которое должно выполняться в проходе #2, чтобы быть истинным
#IFB	**#IFB** <аргумент>	условный оператор - пустой параметр

Директива	Синтаксис	Описание
#IFIDN	**#IFIDN** <аргумент-1> <аргумент-2>	условный оператор - идентичные параметры
#IFDEF	**#IFDEF** <выражение>	условный оператор - определяемый параметр
#IFLAB	**#IFLAB** <аргумент>	условный оператор - параметр-метка
#INCLUDE	**#INCLUDE** “<имя_файла>”	вставить внешний файл с исходным кодом
INTEL	**INTEL**	установить указатель основания в стиле Intel
.LALL	**.LALL**	включить в листинг полное тело макросов
.LIST	**.LIST**	включить листинг (по умолчанию)
#LOAD	**#LOAD** “путь\имя_файла”	загрузить именованный объектный файл во время компоновки
LOCAL	**LOCAL** <аргумент>	определить метки как локальные по отношению к макросу
LONG	**LONG** <выражение> [,<выражение>...]	определить длинное слово в объектном коде
LONGS	**LONGS**	определить длину новой метки по умолчанию как long
MACRO	<macro> **MACRO** [параметр-1] [,параметр-2]...	определить шаблон макроса
MEND	**MEND**	конец объявления макроса
MOTOROLA	**MOTOROLA**	установить указатель основания в стиле Motorola
.NOCHANGE	**.NOCHANGE**	включить в листинг исходные строки #DEFINE
.NOLIST	**.NOLIST**	выключить листинг
%OUT	**%OUT** строка	вывести строку на консоль

Директива	Синтаксис	Описание
.PAGE	**.PAGE**	выполнить прогон листа
PUBLIC	**PUBLIC** <аргумент>	сделать метки открытыми
REPEAT	**REPEAT**	инициализатор цикла времени ассемблирования
.SALL	**.SALL**	подавить все тело вызванного макроса
SEGMENT	|<имя>|**SEGMENT** <выравнивание> <объединение> '<class>' [cod]	начало нового сегмента
.SETDP	**.SETDP** <базовый адрес>	установить базовый адрес для прямой страницы
SKIP	**SKIP** <количество байт>,<переменная>	вставляет указанное количество байтов с начальным значением
STRING	**STRING** <выражение или "строка">,[,<выражение или "строка">...]	определить байтовый уровень строки
SUBTTL	**SUBTTL** "<строка подзаголовка>"	определить подзаголовок для заголовка листинга
.TAB	**.TAB** <метка>, <Opcode>, <операнд>, <комментарий>	установить длины полей в листинге
TEXAS	**TEXAS**	указатель основания в стиле Texas Instruments
TITLE	**TITLE** "<строка заголовка>"	определить главный заголовок для листинга
UNTIL	**UNTIL** <выражение>	ограничитель цикла времени ассемблирования
WORD	**WORD** <выражение> [, <выражение>...]	определить слово в объектном коде
WORDS	**WORDS**	определить длину новой метки по умолчанию как word

Директива	Синтаксис	Описание
.XALL	**.XALL**	включить в листинг только макросы, генерирующие код
ZILOG	**ZILOG**	указатель основания в стиле Zilog

ПРИЛОЖЕНИЕ 3.
ОСНОВНЫЕ ХАРАКТЕРИСТИКИ МИКРОКОНТРОЛЛЕРОВ ST7

Шифр	Тип памяти программ		Память программ, Кб	RAM, Б	E²PROM данных	Входы	Таймеры			Последовательный интерфейс	Напряжение питания, В
	Flash	ROM					12/ 16 бит	8-бит	Другое		
1	2	3	4	5	6	7	8	9	10	11	12
ST7LITEUS2	•		1	128			1x12 бит	1	WDG, RTC		2.4..5.5
ST7LITEUS5	•		1	128		5x10 бит		1	WDG, RTC		2.4..5.5
ST7LITEU05	•		2	128		5x10 бит		1	WDG, RTC		2.4..5.5
ST7LITEU09	•		2	128	128	5x10 бит		1	WDG, RTC		2.4..5.5
ST7LITES2Y0	•		1	128				1	WDG, RTC	SPI	2.4..5.5
ST7LITES5Y0	•		1	128		5x8 бит		1	WDG, RTC	SPI	2.4..5.5
ST7LITE02Y0	•		1.5	128				1	WDG, RTC	SPI	2.4..5.5
ST7LITE05Y0	•		1.5	128		5x8 бит	1x12 бит	1	WDG, RTC	SPI	2.4..5.5
ST7LITE09Y0	•		1.5	128	128	5x8 бит		1	WDG, RTC	SPI	2.4..5.5
ST7LIT10BF0	•		2	256		7x10 бит		2	WDG, RTC	SPI	2.7..5.5
ST7LIT10BY0	•		2	256		7x10 бит		2	WDG, RTC	SPI	2.7..5.5
ST7LIT15BF0	•		2	256		7x10 бит		2	WDG, RTC	SPI	2.7..5.5
ST7LIT15BY0	•		2	256		7x10 бит		2	WDG, RTC	SPI	2.7..5.5
ST7LIT19BF0	•		2	256	128	7x10 бит		2	WDG, RTC	SPI	2.7..5.5
ST7LIT19BY0	•		2	256	128	7x10 бит		2	WDG, RTC	SPI	2.7..5.5
ST7LIT10BF1	•		4	256		7x10 бит	2x12 бит	2	WDG, RTC	SPI	2.7..5.5
ST7LIT10BY1	•		4	256		7x10 бит		2	WDG, RTC	SPI	2.7..5.5
ST7LIT15BF1	•		4	256		7x10 бит		2	WDG, RTC	SPI	2.7..5.5
ST7LIT15BY1	•		4	256		7x10 бит		2	WDG, RTC	SPI	2.7..5.5
ST7LIT19BF1	•		4	256	128	7x10 бит		2	WDG, RTC	SPI	2.7..5.5
ST7LIT19BY1	•		4	256	128	7x10 бит		2	WDG, RTC	SPI	2.7..5.5
ST7DALIF2	•		8	384	256	7x10 бит	1x12 бит	2	WDG, RTC	SPI/ DALI	2.4..5.5
ST7LITE30F2	•		8	384		7x10 бит	2x12 бит	2	WDG, RTC	SPI/ LINSCI	2.7..5.5
ST7LITE35F2	•		8	384		7x10		2	WDG,	SPI/	2.7..5.5

1	2	3	4	5	6	7	8	9	10	11	12
						бит			RTC	LINSCI	
ST7LITE39F2	•		8	384	256	7x10 бит		1	WDG, RTC	SPI/ LINSCI	2.7..5.5
ST72260G1	•	•	4	256			2x16 бит		WDG, RTC	SPI	2.7..5.5
ST72262G1	•	•	4	256		6x10 бит			WDG, RTC	SPI	2.7..5.5
ST72264G1	•	•	4	256		6x10 бит			WDG, RTC	SPI/ SCI/ I^2C	2.7..5.5
ST7232AK1	•	•	4	384		8x10 бит			WDG, RTC	SPI/ SCI	3.8..5.5
ST72262G2	•	•	8	256		6x10 бит			WDG, RTC	SPI	2.7..5.5
ST72264G2	•	•	8	256		6x10 бит			WDG, RTC	SPI/ SCI/ I^2C	2.7..5.5
ST72324BK2	•	•	8	384		8x10 бит			WDG, RTC	SPI/ SCI	3.8..5.5
ST72324LK2	•	•	8	384		8x10 бит			WDG, RTC	SPI/ SCI	2.85..5.5
ST7232AK2	•	•	8	384		8x10 бит			WDG, RTC	SPI/ SCI	3.8..5.5
ST72340K2	•		8	512	256				WWDG, RTC	SPI/ SCI	2.7..5.5
ST72344K2	•		8	512	256	8x10 бит			WWDG, RTC	SPI/ SCI/ I^2C	2.7..5.5
ST72324BK4	•	•	16	512		8x10 бит			WDG, RTC	SPI/ SCI	3.8..5.5
ST72324LK4	•	•	16	512		8x10 бит			WDG, RTC	SPI/ SCI	2.85..3.6
ST72325K4	•	•	16	512		8x10 бит	2x16 бит	1	CSS, WDG, RTC	SPI/ SCI/ I^2C	3.8..5.5
ST72340K4	•		16	1K	256				WWDG, RTC	SPI/ SCI	2.7..5.5
ST72344K4	•	•	16	1K	256	8x10 бит			WWDG, RTC	SPI/ SCI/ I^2C	2.7..5.5
ST72321BK6	•		32	1K		8x10 бит			WWDG, RTC	SPI/ SCI/ I^2C	3.8..5.5
ST72324BK6	•	•	32	1K		8x10 бит			WDG, RTC	SPI/ SCI	3.8..5.5
ST72324LK6	•		32	1K		8x10 бит			WDG, RTC	SPI/ SCI	2.85..3.6
ST72325K6	•	•	32	1K		8x10 бит		1	WDG, RTC	SPI/ SCI/ I^2C	3.8..5.5
ST72361K6	•	•	32	1K		6x10 бит	1x16 бит	1	CSS, WDG, RTC	SPI/ 2xSCI	4.5..5.5
ST72361K7	•	•	48	1K		6x10 бит		1	WWDG, RTC	SPI/ 2xSCI	4.5..5.5
ST72361K9	•	•	60	2K		12x10 бит		1	WWDG, RTC	SPI/ 2xSCI	4.5..5.5
ST7232AJ1	•	•	4	384		12x10 бит	2x16 бит		WDG, RTC	SPI/ SCI	3.8..5.5
ST72324BJ2	•	•	8	384		12x10 бит			WDG, RTC	SPI/ SCI	3.8..5.5
ST72324LJ2	•	•	8	384		12x10 бит			WDG, RTC	SPI/ SCI	2.85..3.6
ST72324LS2	•		8	384		12x10 бит			WDG, RTC	SPI/ SCI	2.85..3.6

ПРИЛОЖЕНИЕ 4.
ШАБЛОН ОСНОВНОЙ ПРОГРАММЫ МИКРОКОНТРОЛЛЕРА ST7

```
ST7/
;*****************************************************************
; ЗАГОЛОВОК:
; АВТОР:
; ОПИСАНИЕ:
;*****************************************************************

     TITLE "TEMPLATE.ASM"

; MOTOROLA формат, другие доступные форматы Intel, Zilog
; или Texas.

     MOTOROLA

;*****************************************************************
; Объявление файлов включения *.INC
;*****************************************************************
; Файлы включения с прототипами импортируемых
; переменных и функций
; ST7Lite2.INC - содержит переменные, определенные
; для МК ST7Lite2

     #include "ST7Lite2.INC"

;----------------------------------------------------------------

;*****************************************************************
; Объявление глобальных функций, переменных и констант
;*****************************************************************

;*****************************************************************
; Объявление символов
;*****************************************************************

;----------------------------------------------------------------
     BYTES
     segment byte 'ram0'

;*****************************************************************
; Раздел объявления переменных в 'ram0' части памяти
;*****************************************************************

;----------------------------------------------------------------
     WORDS
     segment byte 'rom'

;*****************************************************************
; Раздел объявления констант в 'rom' части памяти
;*****************************************************************
```

```
;-----------------------------------------------------------------

;*****************************************************************
; Раздел объявления подпрограмм
;*****************************************************************

;-----------------------------------------------------------------

;*****************************************************************
; Основная часть программы
;*****************************************************************

main:
    rsp                             ; сброс указателя стека
    sim                             ; установка маски прерываний
    clr   MCCSR                     ; инициализация
LBL_MAIN_LOOP:                      ; начало основного цикла
                                    ; программы
    jp    LBL_MAIN_LOOP             ; возврат на начало основного
                                    ; цикла программы
  ret ; конец подпрограммы main

;*****************************************************************
; Раздел объявления подпрограммы прерывания
;*****************************************************************

dummy_rt:  IRET ; Пустая процедура для возврата в основную
                ; программу.

;*****************************************************************
; Объявление векторов прерывания
;*****************************************************************
     segment 'vectit'    DC.W dummy_rt  ; Адрес FFE0-FFE1h
     SPI_it              DC.W dummy_rt  ; Адрес FFE2-FFE3h
     lt_RTC1_it          DC.W dummy_rt  ; Адрес FFE4-FFE5h
     lt_IC_it            DC.W dummy_rt  ; Адрес FFE6-FFE7h
     at_timerover_it     DC.W dummy_rt  ; Адрес FFE8-FFE9h
     at_timerOC_it       DC.W dummy_rt  ; Адрес FFEA-FFEBh
     AVD_it              DC.W dummy_rt  ; Адрес FFEC-FFEDh
                         DC.W dummy_rt  ; Адрес FFEE-FFEFh
     lt_RTC2_it          DC.W dummy_rt  ; Адрес FFF0-FFF1h
     ext3_it             DC.W dummy_rt  ; Адрес FFF2-FFF3h
     ext2_it             DC.W dummy_rt  ; Адрес FFF4-FFF5h
     ext1_it             DC.W dummy_rt  ; Адрес FFF6-FFF7h
     ext0_it             DC.W dummy_rt  ; Адрес FFF8-FFF9h
     AWU_it              DC.W dummy_rt  ; Адрес FFFA-FFFBh
     softit              DC.W dummy_rt  ; Адрес FFFC-FFFDh
     reset               DC.W main      ; Адрес FFFE-FFFFh

     END
;*****************************************************************
```

ПРИЛОЖЕНИЕ 5.
ЛИСТИНГ УПРАВЛЯЮЩЕЙ ПРОГРАММЫ УСТРОЙСТВА КОНТРОЛЯ СОСТОЯНИЯ АВТОМОБИЛЬНОГО ПОДЪЕМНИКА

```
st7/

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; НАЗВАНИЕ:         control.asm
;; РАЗРАБОТЧИКИ:     А.В.Желтухин, М.Э.Яновский
;;                   Национальный аэрокосмический университет
;;                   "Харьковский авиационный институт"
;; ОПИСАНИЕ:         Управляющая программа устройства контроля
;;                   автомобильным подъемником
;;                   (пилотный проект на ST7)
;; ВЕРСИЯ:           1.0
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

TITLE "control.asm "
MOTOROLA

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Объявление подключаемых файлов  (*.INC)
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  #include "st7lite2.inc" ; описание переменных

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Объявление глобальных функций
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    PUBLIC watchdog_disable
    PUBLIC relay_on
    PUBLIC relay_off
    PUBLIC values_nulling
    PUBLIC wait
    PUBLIC indication_on
    PUBLIC indication_off
    PUBLIC indication_error

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Объявление констант
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  ;; PORT A A0-A7 - данные с датчика вращения
  ;; Биты PORT B:
  #define PORT_TRAILER          #0  ;  чтение (датчик концевых
                                    ;  положений)
```

```
  #define PORT_RESET            #1  ;  чтение (кнопка сброса)
  #define PORT_INDICATION       #3  ;  запись (индикация)
  #define PORT_LED_ERROR_LOW    #4  ;  запись (младший бит
                                    ;  индикации ошибки)
  #define PORT_LED_ERROR_HIGH   #5  ;  запись (старший бит
                                    ;  индикации ошибки)
  #define PORT_RELAY            #6  ;  запись (реле)

  ;; Ошибки
  #define NO_ERROR              #0  ;  ошибок нет
  #define SHIFT_ERROR           #1  ;  перекос опорных стоек
  #define UP_DOWN_ERROR         #2  ;  движения в разные
                                    ;  стороны стоек

  BYTES

  segment byte 'ram0'


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Объявление переменных
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

f_oper        ds.b  1  ; флаг операций = 1 байт
f_reset       ds.b  1  ; флаг сброса = 1 байт
diff_counter  ds.b  1  ; разница между счетчиками = 1 байт
counter1      ds.b  1  ; счетчик первой стойки = 1 байт
counter2      ds.b  1  ; счетчик второй стойки = 1 байт
err           ds.b  1  ; флаг ошибки = 1 байт


        WORDS

  segment byte 'rom'

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Реализация подпрограмм
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Описание: Отключение сторожевого таймера
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
watchdog_disable:
    ld  A,      #$fe ; загрузка в аккумулятор константы
                     ; отключения
    ld  WDGCR,  A    ; загрузка содержимого аккумулятора в
                     ; регистр
  ret ; конец подпрограммы watchdog_disable

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;; Описание: Включение реле. Начать движения стоек.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
relay_on:
    bset PBDR,  PORT_RELAY  ; relay -> 1
  ret ; конец подпрограммы raley_on

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Описание: Выключение реле. Остановить стойки.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
relay_off:
    bres PBDR,  PORT_RELAY  ; relay -> 0
  ret ; конец подпрограммы relay_off

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Описание: Обнуление переменных
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
values_nulling:
    clr  counter1  ; очистка всех битов счетчика первой стойки
    clr  counter2  ; очистка всех битов счетчика второй стойки
    clr  f_reset   ; очистка флага сброса
    clr  err       ; очистка флага ошибок
  ret ; конец подпрограммы values_nulling

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Описание: Ожидание (задержка)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
wait:
    ld    X,  #255          ; загрузка в регистр X -> 255
LBL_X_NOT_ZERO:             ;
    ld    Y,  #255          ; загрузка в регистр Y -> 255
LBL_ Y_NOT_ZERO:            ;
    dec   Y                 ; декремент Y
    jrne  LBL_ X_NOT_ZERO   ; если Y не равен 0 перейти на метку
    dec   X                 ; декремент X
    jrne  LBL_ X_NOT_ZERO   ; если X не равен 0 перейти на метку
  ret ; конец подпрограммы wait

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Описание: Включение индикации
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
indication_on:
    bres  PBDR,    PORT_INDICATION  ; indication -> 1
  ret ; конец подпрограммы indication_on

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Описание: Выключение индикации
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
indication_off:
    bset  PBDR,    PORT_INDICATION  ; indication -> 0
  ret ; конец подпрограммы indication_off

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Описание: Индикация ошибки
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
indication_error:
    bres  PBDR,    PORT_LED_ERROR_HIGH    ; сброс старшего бита
                                          ; индикации
LBL_LOOP:                                 ; основной цикл
                                          ; индикации ошибки
    ld    A,       err                    ; загрузить в А
                                          ; значение ошибки
    cp    A,       UP_DOWN_ERROR          ; если ошибка
                                          ; движение в разные
                                          ; стороны стоек
    jrne  LBL_SELECT_COUNTER_ERR          ; иначе переходим
                                          ; по метке
    ld    A,       #$e7                   ; загрузка в А
                                          ; константы обнуления
                                          ; третьего и
                                          ; четвертого битов
    and   A,       PBDR                   ; обнуление
    ld    PBDR,    A                      ; загрузка значение в
                                          ; порт В
    call  wait                            ; вызов подпрограммы
                                          ; ожидания
    ld    A,       #$10                   ; загрузка в А
                                          ; константы установки
    or    A,       PBDR                   ; установка четвертого
                                          ; бита
    ld    PBDR,    A                      ; загрузка значения в
                                          ; порт В
    jp    LBL_END_INDICATION              ; перейти по метке
LBL_SELECT_COUNTER_ERR:                   ; определение из-за
                                          ; какой именно стойки
                                          ; возникла ошибка
    ld    A,       counter1               ; загрузка в А
                                          ; значения первого
                                          ; счетчика
    cp    A,       counter2               ; сравнение первого со
                                          ; вторым счетчиком
    jrult LBL_LED1                        ; если счетчик один
                                          ; меньше счетчика два
                                          ; перейти по метке
    bset  PBDR,    PORT_LED_ERROR_LOW     ; установка младшего
                                          ; бита ошибки, что
                                          ; соответствует
                                          ; разрешению индикации
                                          ; ошибки произошедшей
                                          ; из-за первой стойки
                                          ; подъемника
    jp    LBL_SET_LED_ERROR_END           ; переход на индикацию
LBL_LED1:                                 ; счетчик один меньше
                                          ; счетчика два
    bres  PBDR,    PORT_LED_ERROR_LOW     ; сбрасываем младшего
                                          ; бита ошибки, что
                                          ; соответствует
                                          ; разрешению индикации
                                          ; ошибки произошедшей
```

```
                                        ; из-за второй стойки
                                        ; подъемника
LBL_SET_LED_ERROR_END:                  ; световая и звуковая
                                        ; индикация ошибки
    call  indication_on                 ; включение индикации
    call  wait                          ; задержка
    call  indication_off                ; выключение индикации
LBL_END_INDICATION:                     ; конец индикации
    call  wait                          ; задержка
    btjt  PBDR,    PORT_RESET,  LBL_RESET ; если отпущен сброс
                                        ; перейти по метке
    bset  f_reset,      #0              ; установить флаг
                                        ; нажатия сброса
    jp    LBL_LOOP                      ; перейти на начало
                                        ; основного цикла
LBL_RESET:                              ; если отпущен сброс
    btjf  f_reset,      #0,  LBL_LOOP   ; если флаг сброса не
                                        ; установлен перейти
                                        ; на начало основного
                                        ; цикла
    bres  f_reset,      #0              ; обнуление флага
                                        ; сброса
    ld    A,            err             ; загрузка в A ошибки
    cp    A,            UP_DOWN_ERROR   ; если ошибка движение
                                        ; в разные стороны
                                        ; стоек
    jreq  LBL_INDICATION_ERROR          ; перейти по метке
    ld    A,            diff_counter    ; загрузить в A
                                        ; разницу счетчиков
    add   A,            #4              ; добавить максимально
                                        ; допустимое значение
                                        ; при перекосе
    ld    diff_counter, A               ; загрузить в разницу
LBL_INDICATION_ERROR:                   ; перейти на индикацию
    clr   err                           ; сброс флага ошибки
    clr   f_oper                        ; сброс флага операций
  ret ; конец подпрограммы indication_error

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Описание: Проверка состояния
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
check_state:
    btjf  PBDR,  PORT_TRAILER,  LBL_NTRAILER  ; если концевик не
                                              ; сработал
                                              ; перейти по метке
    call  values_nulling                      ; вызов
                                              ; подпрограммы
                                              ; обнуления
                                              ; переменных
LBL_NTRAILER:                                 ; концевик не
                                              ; сработал
    ld    A,          counter1                ; загрузка в A
                                              ; первого счетчика
    cp    A,          counter2                ; сравнение со
```

```
                                                ; вторым счетчиком
    jrult LBL_LESS_COUNTER1                     ; если первый
                                                ; счетчик меньше
                                                ; второго перейти
                                                ; по метке
    sub   A,          counter2                  ; вычитание из
                                                ; первого счетчика
                                                ; значение второго
    jp    LBL_DIFF                              ; перейти на
                                                ; подсчет разницы
LBL_LESS_COUNTER1:                              ; первый счетчик
                                                ; меньше
    ld    A,          counter2                  ; загрузка в A
                                                ; второго счетчика
    sub   A,          counter1                  ; вычитание из
                                                ; второго счетчика
                                                ; значение первого
LBL_DIFF:                                       ; подсчет разницы
    cp    A,          diff_counter              ; если сохраненная
                                                ; разница
                                                ; полученная на
                                                ; предыдущем шаге
    jrule  LBL_READ_COUNTER1                    ; меньше или равна
                                                ; разницы взятой
                                                ; по модулю на
                                                ; текущем шаге
                                                ; перейти по метке
    ld    A,          SHIFT_ERROR               ; вернуть ошибку
                                                ; перекоса
    jp    LBL_CHECK_STATE_END_ERROR             ; перейти на
                                                ; проверку
                                                ; состояния и
                                                ; возврата ошибки
LBL_READ_COUNTER1:                              ; чтение состояния
                                                ; первой стойки
    btjt  PADR,       #0,   LBL_COUNTER1        ; если первый
                                                ; счетчик
                                                ; перейти по метке
    btjf  f_oper,     #1,   LBL_FOPER1          ; если флаг 1
                                                ; не установлен
                                                ; перейти по метке
    bset  f_oper,     #3                        ; установить
                                                ; флаг 3
    jp    LBL_READ_COUNTER2                     ; перейти на
                                                ; чтение значения
                                                ; второго счетчика
LBL_FOPER1:                                     ; флаг 1
    bset  f_oper,     #0                        ; установить
                                                ; флаг 0
    jp    LBL_READ_COUNTER2                     ; переход на
                                                ; чтение значения
                                                ; второго счетчика
LBL_COUNTER1:                                   ; сработал первый
                                                ; счетчик
```

```
    btjt  PADR,      #1,  LBL_COUNTER2        ; если второй
                                              ; перейти по метке
    btjf  f_oper,    #0,  LBL_FOPER2          ; если флаг 0 не
                                              ; установлен
                                              ; перейти по метке
    bset  f_oper,    #2                       ; установить
                                              ; флаг 2
    jp    LBL_READ_COUNTER2                   ; переход на
                                              ; чтение значения
                                              ; второго счетчика
LBL_FOPER2:                                   ; флаг 2
    bset  f_oper,    #1                       ; установить
                                              ; флаг 1
    jp    LBL_READ_COUNTER2                   ; переход на
                                              ; чтение значения
                                              ; второго счетчика
LBL_COUNTER2:                                 ; сработал второй
    ld    A,         f_oper                   ; загрузка в A
                                              ; флагов операций
    and   A,         #$fc                     ; сброс флага 0
                                              ; и флага 1
    ld    f_oper,    A                        ; загрузка
                                              ; значения из A
    btjf  f_oper,    #2,  LBL_OPER3           ; если флаг 2
                                              ; не установлен
                                              ; перейти по метке
    inc   counter1                            ; инкремент
                                              ; первого счетчика
    jp    LBL_ONE_BYTE                        ; переход по метке
LBL_OPER3:                                    ; флаг 3
    btjf  f_oper,    #3,  LBL_READ_COUNTER2   ; если флаг 3
                                              ; не установлен
                                              ; перейти по метке
    dec   counter1                            ; декремент
                                              ; первого счетчика
    jp    LBL_ONE_BYTE                        ; переход на
                                              ; АЛГАРИТМ
LBL_READ_COUNTER2:                            ; чтение значения
                                              ; второго счетчика
    btjt  PADR,      #2,  LBL_COUNTER3        ; если третий
                                              ; счетчика
                                              ; перейти по метке
    btjf  f_oper,    #5,  LBL_FOPER4          ; если флаг 5
                                              ; не установлен
                                              ; перейти по метке
    bset  f_oper,    #7                       ; установить
                                              ; флаг 7
    jp    LBL_CHECK_STATE_END                 ; перейти на
                                              ; проверку
                                              ; состояний
LBL_FOPER4:                                   ; флаг 4
    bset  f_oper,    #4                       ; установить
                                              ; флаг 4
    jp    LBL_CHECK_STATE_END                 ; перейти на
```

```
                                                ; проверку
                                                ; состояний
LBL_COUNTER3:                                   ; сработал третий
                                                ; счетчик
    btjt  PADR,       #3,  LBL_COUNTER4         ; если четвертый
                                                ; счетчика
                                                ; перейти по метке
    btjf  f_oper,     #4,  LBL_FOPER5           ; если не
                                                ; установлен
                                                ; флаг 4
                                                ; перейти по метке
    bset  f_oper,     #6                        ; установить
                                                ; флаг 6
    jp    LBL_CHECK_STATE_END                   ; перейти на
                                                ; проверку
                                                ; состояний
LBL_FOPER5:                                     ; флаг 5
    bset  f_oper,     #5                        ; установить
                                                ; флаг 5
    jp    LBL_CHECK_STATE_END                   ; перейти на
                                                ; проверку
                                                ; состояний
LBL_COUNTER4:                                   ; сработал
                                                ; четвертый
                                                ; счетчик
    ld    A,          f_oper                    ; загрузка в A
                                                ; флагов операций
    and   A,          #$cf                      ; сброс флага 4
                                                ; и флага 5
    ld    f_oper,     A                         ; загрузка
                                                ; значения из A
    btjf  f_oper,     #6,  LBL_OPER6            ; если флаг 6
                                                ; не установлен
                                                ; перейти по метке
    inc   counter2                              ; инкрементировать
                                                ; счетчик 2
    jp    LBL_ONE_BYTE                          ; перейти по метке
LBL_OPER6:                                      ; флаг 6
    btjf  f_oper,     #7,  LBL_CHECK_STATE_END ; если флаг 7
                                                ; не установлен
                                                ; перейти по метке
    dec   counter2                              ; декремент
                                                ; первого счетчика
LBL_ONE_BYTE:                                   ; расчет разницы
                                                ; счетчиков с
                                                ; учетом флагов
                                                ; операций
    btjf  f_oper,     #2,  LBL_CHECK_F3         ; если флаг 2
                                                ; не установлен
                                                ; перейти по метке
    btjf  f_oper,     #7,  LBL_CHECK_COUNTERS   ; если флаг 7
                                                ; не установлен
                                                ; перейти по метке
    ld    A,          UP_DOWN_ERROR             ; вернуть ошибку
```

```
                                        ; движение в
                                        ; разные стороны
    jp    LBL_CHECK_STATE_END_ERROR     ; перейти на
                                        ; проверку
                                        ; состояния и
                                        ; возврата ошибки
LBL_CHECK_COUNTERS:                     ; проверка
                                        ; счетчиков
    ld    A,          counter1          ; загрузка в A
                                        ; первого счетчика
    cp    A,          counter2          ; сравнение со
                                        ; вторым
    jrult LBL_LESS_ST1                  ; переход если
                                        ; первый счетчик
                                        ; меньше второго
    sub   A,          counter2          ; вычесть из
                                        ; первого счетчика
                                        ; второй
    ld    counter1,   A                 ; загрузить в
                                        ; первый счетчик
                                        ; разницы
    clr   counter2                      ; очистка второго
                                        ; счетчика
    jp    LBL_RESET_FLAGS               ; переход на сброс
                                        ; флагов
LBL_LESS_ST1:                           ; первый счетчик
                                        ; меньше второго
    ld    A,          counter2          ; загрузить в A
                                        ; второй счетчик
    sub   A,          counter1          ; вычесть из
                                        ; второго счетчика
                                        ; первый
    ld    counter2,   A                 ; загрузить
                                        ; разницу во
                                        ; второй счетчик
    clr               counter1          ; очистка первого
                                        ; счетчика
LBL_RESET_FLAGS:                        ; сброс флагов
    ld    A,          f_oper            ; загрузка в A
                                        ; флагов операций
    and   A,          #$33              ; сброс флага 2
                                        ; флага 3
    ld    f_oper,     A                 ; загрузка
                                        ; значения во флаг
                                        ; операций
    jp    LBL_CHECK_STATE_END           ; перейти на
                                        ; проверку
                                        ; состояний
LBL_CHECK_F3:                           ; проверка
                                        ; третьего флага
    btjf  f_oper,     #3,  LBL_CHECK_F6 ; если флаг 3
                                        ; перейти по метке
    btjf  f_oper,     #6,  LBL_INC_COUNTERS ; если флаг 6
                                        ; перейти по метке
```

```
    ld    A,         UP_DOWN_ERROR          ; вернуть ошибку
                                            ; движение в
                                            ; разные стороны
    jp    LBL_CHECK_STATE_END_ERROR          ; перейти на
                                            ; проверку
                                            ; состояния и
                                            ; возврата ошибки
LBL_INC_COUNTERS:                           ; инкремент
                                            ; счетчиков
    inc   counter1                          ; инкремент
                                            ; первого счетчика
    inc   counter2                          ; инкремент
                                            ; второго счетчика
    jp    LBL_CHECK_COUNTERS                ; перейти на
                                            ; проверку
                                            ; счетчиков
LBL_CHECK_F6:                               ; проверка шестого
                                            ; флага
    btjt  f_oper,    #6, LBL_CHECK_COUNTERS ; если флаг 6
                                            ; перейти по метке
    btjt  f_oper,    #7, LBL_INC_COUNTERS   ; если флаг 7
                                            ; перейти по метке
    jp    LBL_CHECK_STATE_END               ; перейти на
                                            ; проверку
                                            ; состояний
LBL_CHECK_STATE_END:                        ; проверка
                                            ; состояний
    ld    A,         NO_ERROR               ; нет ошибок
LBL_CHECK_STATE_END_ERROR:                  ; возврат ошибки
    ld    err,       A                      ; загрузка в A
; состояния ошибки
  ret ; конец подпрограммы check_state

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Описание: Основная подпрограмма. Основной программный цикл.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
main:
    rsp                                 ; сброс указателя стека
    sim                                 ; установка маски прерываний
    clr   MCCSR                         ; инициализация
                                        ; микроконтроллера в
                                        ; нормальном режиме
    ld    A,           #$f0             ; формирование константы для
                                        ; настройка порта A
    ld    PADDR,       A                ; загрузка настройки порта A в
                                        ; регистр направления данных
    ld    PAOR,        A                ; загрузка настройки порта A
                                        ; в регистр
                                        ; конфигураций
    ld    A,           #$78             ; формирование константы
                                        ; для настройка порта B
    ld    PBDDR,       A                ; загрузка настройки порта B в
                                        ; регистр направления данных
    ld    PBOR,        A                ; загрузка настройки порта B в
```

```
                                    ; регистр конфигураций
    call  values_nulling            ; обнуление всех переменных
    ld    A,              #20       ; формирование константы
                                    ; максимальной
                                    ; разницы счетчиков
    ld    diff_counter,   A         ; загрузка константы
                                    ; возникновение
                                    ; аварийной ситуации
    ld    A,              #$48      ; формирование константы
                                    ; для начала
                                    ; движения подъемника
    ld    PBDR,           A         ; загрузка константы в порт B
    clr   err                       ; сброс состояния ошибки
    clr   f_oper                    ; сброс флагов операций
LBL_MAIN_LOOP:                      ; начало основного цикла
                                    ; программы
    call  watchdog_disable          ; отключение сброса от
                                    ; сторожевого таймера
    call  check_state               ; проверка состояния
                                    ; счетчиков,
                                    ; результат находится в A
    cp    A,              #0        ; если A равен 0 - подъемник
                                    ; работает нормально
    jreq  LBL_MAIN_LOOP             ; продолжаем цикл проверки
                                    ; состояния счетчиков
    call  relay_off                 ; если A не равно
                                    ; 0 - остановить подъемник
    call  indication_error          ; подача светового и звукового
                                    ; сигнала,
                                    ; пока не нажат сброс
    ld    A,              #$48      ; формирование константы
                                    ; для продолжения
                                    ; движения подъемника
    ld    PBDR,           A         ; загрузка константы в порт B
    jp    LBL_MAIN_LOOP             ; возврат на начало основного
                                    ; цикла программы
  ret ; конец подпрограммы main

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Определение подпрограмм-обработчиков прерываний
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

dummy_rt:  IRET  ; Пустая процедура для возврата
                 ; в основную программу

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Объявление таблицы векторов прерываний
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

  segment 'vectit'
```

```
    DC.W  dummy_rt                ; Адрес FFE0-FFE1h
lt_RTC1_it        DC.W  dummy_rt  ; Адрес FFE4-FFE5h
lt_IC_it          DC.W  dummy_rt  ; Адрес FFE6-FFE7h
at_timerover_it   DC.W  dummy_rt  ; Адрес e FFE8-FFE9h
at_timerOC_it     DC.W  dummy_rt  ; Адрес FFEA-FFEBh
AVD_it            DC.W  dummy_rt  ; Адрес FFEC-FFEDh
    DC.W  dummy_rt                ; Адрес FFEE-FFEFh
lt_RTC2_it  DC.W  dummy_rt        ; Адрес FFF0-FFF1h
ext3_it     DC.W  dummy_rt        ; Адрес FFF2-FFF3h
ext2_it     DC.W  dummy_rt        ; Адрес FFF4-FFF5h
ext1_it     DC.W  dummy_rt        ; Адрес FFF6-FFF7h
ext0_it     DC.W  dummy_rt        ; Адрес FFF8-FFF9h
AWU_it      DC.W  dummy_rt        ; Адрес FFFA-FFFBh
softit      DC.W  dummy_rt        ; Адрес FFFC-FFFDh
reset       DC.W  main            ; Адрес FFFE-FFFFh

  END
```

ПРИЛОЖЕНИЕ 6.
ЛИСТИНГ УПРАВЛЯЮЩЕЙ ПРОГРАММЫ МИКРОКОНТРОЛЛЕРА ТЕРМОРЕГУЛЯТОРА

```
ST7/
;*************************************************************
; НАЗВАНИЕ:     Main.asm
; РАЗРАБОТЧКИ: А.В.Мазуренко, (almaz@xai.edu.ua)
;               Национальный аэрокосмический университет
;               "Харьковский авиационный институт"
; ОПИСАНИЕ:     ПО микроконтроллера терморегулятора
;               (пилотный проект на ST7)
; ВЕРСИЯ:       1.0
;*************************************************************

 TITLE "Main.asm"
 MOTOROLA

;*************************************************************
;Объявление подключаемых файлов  (*.INC)
;*************************************************************
 #include "ST7FLITE19.INC"     ;описание переменных
                               ; для ST7FLITE19
 #include "MAX7221.INC"        ;прототипы импортируемых
                               ; переменных и процедур
                               ;  для MAX7221

;*************************************************************
; Объявление глобальных функций, переменных и констант
;*************************************************************


;*************************************************************
; Объявление символов
;*************************************************************


;*************************************************************
; Объявление переменных в ОЗУ
;*************************************************************
 BYTES
 segment byte 'ram0'
zero      ds.b 1    ; ячейка нулевого значения
dT        ds.b 1    ; погрешность поддержания температуры dT
KeyFlags  ds.b 1    ; флаги нажатия клавиш
StpIndFlg ds.b 1    ; флаг индикации текущей температуры
T1L       ds.b 1    ; Туст1 (младший байт)
T1H       ds.b 1    ; Туст1 (старший байт)
T2L       ds.b 1    ; Туст2 (младший байт)
T2H       ds.b 1    ; Туст2 (старший байт)
T1onLds.b 1    ; Твкл.1 (младший байт)
T1onHds.b 1    ; Твкл.1 (старший байт)
T2onLds.b 1    ; Твкл.2 (младший байт)
T2onHds.b 1    ; Твкл.2 (старший байт)
```

```
T1offL    ds.b 1    ; Твыкл.1 (младший байт)
T1offH    ds.b 1    ; Твыкл.1 (старший байт)
T2offL    ds.b 1    ; Твыкл.2 (младший байт)
T2offH    ds.b 1    ; Твыкл.2 (старший байт)
Uin       ds.b 1    ; Uin1 (старший байт)
     ds.b 1    ; Uin1 (младший байт)
     ds.b 1    ; Uin2 (старший байт)
     ds.b 1    ; Uin2 (младший байт)
     ds.b 1    ; Uin3 (старший байт)
     ds.b 1    ; Uin3 (младший байт)
     ds.b 1    ; Uin4 (старший байт)
     ds.b 1    ; Uin4 (младший байт)
     ds.b 1    ; Uin5 (старший байт)
     ds.b 1    ; Uin5 (младший байт)
     ds.b 1    ; Uin6 (старший байт)
     ds.b 1    ; Uin6 (младший байт)
     ds.b 1    ; Uin7 (старший байт)
     ds.b 1    ; Uin7 (младший байт)
     ds.b 1    ; Uin8 (старший байт)
     ds.b 1    ; Uin8 (младший байт)

;************************************************************
; Объявление констант в ПЗУ (ПП)
;************************************************************
     WORDS
segment byte 'rom'

;************************************************************
; Определение подпрограмм
;************************************************************
;------------------------------------------------------------
; Инициализация переменных
;------------------------------------------------------------
init_val:
clr  A              ; формирование константы для zero
ld   zero,A         ; инициализация переменной zero
ld   StpIndFlg,A    ; инициализация переменной StpIndFl
ld   A,#3           ; формирование константы для dT
ld   dT,A           ; инициализации переменной dT
ret
;------------------------------------------------------------
; Инициализация портов МК
;------------------------------------------------------------
init_port:
ld   A,#%00011110   ; настройка портов A и B:
ld   PADDR,A        ; 1,2,3,4 выв. A – вых., ост. входы
ld   PAOR,A         ; вых. – двухтакт., вх. – плавающие
ld   PBDDR,A        ; 1,2,3,4 выв. B – вых., ост. входы
ld   PBOR,A         ; вых. – двухтакт., вх. – плавающие
clr  A              ; константа для иниция.-ции выходов
ld   PADR,A         ; инициализация выходов порта A
ld   PBDR,A         ; инициализация выходов порта B
ret
```

```
;-------------------------------------------------------------
; Считывание из EEPROM установленных значений температуры
;-------------------------------------------------------------
read_T:
ld   A,$1000        ; адрес Туст.1 (мл. байт)
ld   T1L,A          ; считывание Туст.1 (мл. байт)
ld   A,$1001        ; адрес Туст.1 (ст. байт)
ld   T1H,A          ; считывание Туст.1 (мл. байт)
ld   A,$1002        ; адрес Туст.2 (мл. байт)
ld   T2L,A          ; считывание Туст.2 (мл. байт)
ld   A,$1003        ; адрес Туст.2 (ст. байт)
ld   T2H,A          ; считывание Туст.1 (ст. байт)
ret
;-------------------------------------------------------------
; Вычисление пороговых значений для канала 1
;-------------------------------------------------------------
thrshld1:
rcf                 ; сброс флага переноса (C=0)
ld   A,T1L          ;\;
sub  A,dT           ; |
ld   T1onL,A        ; |Твкл.1=Туст1-dT
ld   A,T1H          ; |
sbc  A,dT           ; |
ld   T1onH,A        ;/;
rcf                 ; сброс флага переноса (C=0)
ld   A,T1L          ;\;
add  A,dT           ; |
ld   T1offL,A       ; |Твыкл.1=Туст1+dT
ld   A,T1H          ; |
adc  A,dT           ; |
ld   T1offH,A       ;/;
ret
;-------------------------------------------------------------
; Вычисление пороговых значений для канала 2
;-------------------------------------------------------------
thrshld2:
rcf                 ; сброс флага переноса (C=0)
ld   A,T2L          ;\;
sub  A,dT           ; |
ld   T2onL,A        ; |Твкл.2=Туст1-dT
ld   A,T2H          ; |
sbc  A,dT           ; |
ld   T2onH,A        ;/;
rcf                 ; сброс флага переноса (C=0)
ld   A,T2L          ;\;
add  A,dT           ; |
ld   T2offL,A       ; |Твыкл.2=Туст1+dT
ld   A,T2H          ; |
adc  A,dT           ; |
ld   T2offH,A       ;/;
ret


;*************************************************************
```

```
; ОСНОВНАЯ ЧАСТЬ ПРОГРАММЫ
;*************************************************************
main:
 rsp                  ; сброс указателя стека
 clr  MCCSR           ; нормальный режим работы ГТИ
 clr  KeyFlags        ; сброс ячейки KeyFlags
 call init_val        ; инициализация переменных
 call init_port       ; инициализация портов
 ld   A,#%00101010    ; константа для настройки AT2
 ld   ATCSR,A         ; Тси=1мс, прерывание по переполн.
 ld   A,#$0f          ; константа 1 для инициализации AT2
 ld   ATRH,A          ; автозагружаемый регистр (ст.байт)
 ld   A,#$f6          ; константа 2 для инициализации AT2
 ld   ATRL,A          ; автозагружаемый регистр (мл.байт)
 ld   A,#%01100101    ; константа для настройки АЦП
 ld   ADCCSR,A        ; fАЦП=fЦПУ, вкл.АЦП, выбор AIN5
 ld   A,#%01110000    ; константа для настройки SPI
 ld   SPICR,A         ; fSPI=fЦПУ/4, вкл.SPI, реж. МАСТЕР
 call read_T          ; считать Туст1 и Туст2 из ЭСППЗУ
 call thrshld1        ; вычисление порогов для канала 1
 call thrshld2        ; вычисление порогов для канала 2
 rim                  ; разрешение прерываний
work:
 btjt KeyFlags,#1,menu    ; если "Enter" нажата -
                          ;  переход в меню
 jp   work                ; возврат на проверку

menu:
 ;--- ... ---; выбор канала для установки температуры,
 ;--- ... ---; установка заданной температуры
 jp   work   ; возврат в основную программу

;*************************************************************
; Определение подпрограмм-обработчиков прерываний
;*************************************************************
;-------------------------------------------------------------
; Обработчик прерывания по переполнению AT2-таймера
;-------------------------------------------------------------
ATOvrFlow:
 ;--- ... ---; выбор канала АЦП, запуск преобразования,
 ;--- ... ---; ADCDRxКпреобр, Ton<ADCDRxКпреобр<Toff,
 ;--- ... ---; вкл.\выкл. ТЭН, световая индикация
 iret
;-------------------------------------------------------------
; Обработчик прерывания по нажатию клавиши "Enter"
;-------------------------------------------------------------
key1scan: bset KeyFlags,#1
 iret
;-------------------------------------------------------------
; Обработчик прерывания по нажатию клавиши "Up"
;-------------------------------------------------------------
key2scan: bset KeyFlags,#2
 iret
;-------------------------------------------------------------
```

```
; Обработчик прерывания по нажатию клавиши "Down"
;-------------------------------------------------------------
key3scan: bset KeyFlags,#3
 iret
;-------------------------------------------------------------
; Пустая процедура для возврата в основную программу
;-------------------------------------------------------------
dummy_rt: iret

;*************************************************************
; Объявление таблицы векторов прерываний
;*************************************************************
 segment 'vectit'
unused_int13        DC.W dummy_rt  ; Адрес FFE0-FFE1h
SPI_int             DC.W dummy_rt  ; Адрес FFE2-FFE3h
LT_RTC1_int         DC.W dummy_rt  ; Адрес FFE4-FFE5h
LT_IntComp_int      DC.W dummy_rt  ; Адрес FFE6-FFE7h
AT_Ovrflow_int DC.W ATOvrFlow      ; Адрес FFE8-FFE9h
AT_OutComp_int      DC.W dummy_rt  ; Адрес FFEA-FFEBh
AVD_int             DC.W dummy_rt  ; Адрес FFEC-FFEDh
unused_int6         DC.W dummy_rt  ; Адрес FFEE-FFEFh
LT_RTC2_int         DC.W dummy_rt  ; Адрес FFF0-FFF1h
Ext3_int            DC.W key3scan  ; Адрес FFF2-FFF3h
Ext2_int            DC.W dummy_rt  ; Адрес FFF4-FFF5h
Ext1_int            DC.W key2scan  ; Адрес FFF6-FFF7h
Ext0_int            DC.W key1scan  ; Адрес FFF8-FFF9h
AWU_int             DC.W dummy_rt  ; Адрес FFFA-FFFBh
Soft_int            DC.W dummy_rt  ; Адрес FFFC-FFFDh
Reset          DC.W main           ; Адрес FFFE-FFFFh
 END
```

Литература

1. Барбаш І.П., Благодарний М.П., Харченко В.С. та інш. Основи цифрових систем // За ред. Благодарного М.П., Харченка В.С. – Харків: Нац. аэрокосмічний ун-т «Харьк. авіац. ін-т». – 2002.– 672 с.
2. Барышев И.В., Мазуренко А.В., Горбуненко О.А. Прикладные вопросы цифровой обработки информации. Часть 1. Применение микроконтроллеров в РТС сбора, обработки и передачи информации, Харьков, ХАИ, 2006. – 112 с.
3. Бочарников И. Микроконтроллеры ARM компании STMicroelectronics. // Новости электроники. – 2007. – №6.
4. Галькевич А.А., Желтухин А.В., Куланов В.А. Специализированные компьютерные системы. Разработка технического задания. – Харьков: Нац. аэрокосмический ун-т «Харьк. авиац. ин-т», 2005.– 23 с.
5. Галькевич А.А., Желтухин А.В., Куланов В.А. и др. Интерфейсы. Интерфейс ISA (PC104) : Учебное пособие. – Харьков: Нац. аэрокосмический ун-т «Харьк. авиац. ин-т», 2007.– 71 с.
6. Датчики: Справочник // Под ред. Готра З.Ю., Чайковского О.И. Львов: Каменяр, 1995. – 312 с.
7. Желтухин А.В., Галькевич А.А. и др. Периферийные устройства: Учебное пособие. – Харьков: Нац. аэрокосмический ун-т «Харьк. авиац. ин-т». – 2005.– 127 с.
8. Иванов В.И., Аксенов А.И., Юшин А.М. Полупроводниковые оптоэлектронные приборы. // Справочник. – М.: Энергоатомиздат, 1984. – 311 с.
9. Киселев Д. Семейство STR910F. Новые 32-бит ARM-микроконтроллеры компании STMICROELECTRONICS // Электроника: НТБ. – 2006. – №6.
10. Левшина Т.С., Новицкий П.В. Электрические измерения физических величин: Измерительные преобразователи. – Л.: Энергоатомиздат. Ленингр. отд-ние, 1983. – 320 с.
11. Мпандо П. STMicroelectronics: научное и промышленное партнерство. // Інформаційні інфраструктури та технології, 2007. – №7. – С. 27-31.
12. Носов Ю.Р., Сидоров А.С. Оптроны и их применение – М.: Радио и связь, 1981. – 230 с.
13. Олейник Б.Н., Лаздина С.И., Лаздин В.П., Жагулло О.М. Приборы и методы температурных измерений – М.: Изд-во стандартов, 1987. – 296 с.
14. Олейник В. 16-разрядные микроконтроллеры семейства ST10 с ядром C166 от STMicroelectronics. // Радиокомпоненты, 2007. – №5. – С. 19-22.

15. Остроумов С.Б., Ушаков А.А., Бабешко Е.В. Разработка проектов на основе программируемых логических контроллеров. – Харьков: Нац. аэрокосмический ун-т «Харьк. авиац. ин-т», 2007.– 36 с.
16. Предко М. Руководство по микроконтроллерам. Том I. М: Постмаркет, 2001. - 416 с.
17. Предко М. Руководство по микроконтроллерам. Том II. М: Постмаркет, 2001. - 488 с.
18. Ридико Л. Микроконтроллеры фирмы STMicroelectronics. Часть 1. // Электронные компоненты, 2002. – №5. – С. 74-78.
19. Ридико Л. Микроконтроллеры фирмы STMicroelectronics. Часть 2. // Электронные компоненты, 2002. – №7. – С. 98-100.
20. Севбо В., Титов М. Микроконтроллеры фирмы STMicroelectronics // CHIP News, 2001. – №6.
21. Харченко В.С. Теоретические основы дефектоустойчивых цифровых систем с версионной избыточностью: Монография. – МО Украины, 1998. – 503 с.
22. Харченко В.С., Орехов А.А. Университеты Украины и компания STMicroelectronics: старт сотрудничества. // Інформаційні інфраструктури та технології, 2007. – №7. – С. 26.
23. Хоровиц П.,Хилл У. Искусство схемотехники. Издание 5-е, переработанное. М.: Мир, 1998. – 704 с.
24. Чепурин И. Перспективы 8-разрядных микроконтроллеров. // Компоненты и технологии, № 4, 2006. – С. 98-101.
25. Шагурин И.И. Микропроцессоры и микроконтроллеры фирмы Motorola: Справ. пособие. – М., 1998. – 560 с.
26. Юдин А. Микроконтроллеры компании STMICROELECTRONICS с ядром ARM // Компоненты и технологии, 2006. – №3.
27. Юдин А. Новое семейство микроконтроллеров компании STMICROELECTRONICS с ядром ARM // Компоненты и технологии. – 2006. – №6.
28. Юдин А. Решения STMICROELECTRONICS для информационных табло на светодиодах // Компоненты и технологии. – 2006. – №2.
29. Edouard-Alexandre Sabatier ST7 Microcontroller Training. Тренинг STM, Киев, 2007. – 101 с.
30. Maurice Levansuu STDV7 – IDE & ST7 in 10 Steps. Тренинг STM, Киев, 2007. – 189 с.
31. http://www.st.com - ST7 8-bit MCU Family. User Guide
32. http://www.st.com - ST7 Family. Programming Manual
33. http://www.st.com - STMicroelectronics Company Presentation. August, 2007
34. http://datasheets.maxim-ic.com - MAX7219/MAX7221. Serially Interfaced, 8-Digit LED Display Drivers. Datasheet. Rev. 4; 7/03. – Maxim corp. – 16 p.
35. http://www.analog.com - AD627. Micropower, Single- and Dual-Supply, Rail-to-Rail Instrumentation Amplifier. Datasheet. Rev. D, 11/07. – Analog Devices, Inc. – 16 p.

36. http://www.diotec-usa.com - 2 AMPS Silicon bridge rectifiers. Datasheet. No. BRSB-200-1C. – Diotec Electronics corp. – 2 p.
37. http://www.fairchildsemi.com - LM78XX/LM78XXA. 3-Terminal 1A Positive Voltage Regulator. Datasheet. Rev. 1.0.1; 6/06. – Fairchild Semiconductor corp. – 28 p.
38. http://www.irf.com - Series PVT312. Microelectronic Power IC. Datasheet. No. PD 10038D; 8/00. – International Rectifier. – 5 p.
39. http://www.st.com - ST7LITE1x. Datasheet. Rev. 2.0; 12/04. – STMicroelectronics. – 131 p.
40. http://www.ystone.com.tw - Three Digit LED Displays. Datasheet. – Yellow Stone corp. – 1 p.