Контроллеры

Введение

Одной из характерных особенностей нынешнего этапа научно-технического прогресса является все более широкое применение микроэлектроники в различных отраслях народного хозяйства. Роль микроэлектроники в развитии общественного производства определяется ее практически неограниченными возможностями в решении различных задач во всех областях народного хозяйства, глубоким влиянием на культуру и быт современного человека.

Особое внимание в настоящее время уделяется внедрению микропроцессоров, обеспечивающих решение задач автоматизации управления механизмами, приборами ' и аппаратурой. Адаптация микропроцессора к особенностям конкретной задачи осуществляется в основном путем разработки соответствующего программного обеспечения, заносимого затем в память программ. Аппаратная адаптация в большинстве случаев осуществляется путем подключения необходимых интегральных схем обрамления и организации ввода-вывода, соответствующих решаемой задаче.

В микропроцессорной технике выделился самостоятельный класс больших интегральных схем (БИС) - однокристальные микроЭВМ (ОМЭВМ), которые предназначены для “интеллектуализации” оборудования различного назначения. Архитектура однокристальных микроЭВМ - результат эволюции архитектуры микропроцессоров и микропроцессорных систем, обусловленной стремлением существенно снизить их аппаратные затраты и стоимость. Как правило, эти цели достигаются как путем повышения уровня интеграции БИС, так и за счет поиска компромисса между стоимостью, аппаратными затратами и техническими характеристиками ОМЭВМ.

ОМЭВМ представляют собой приборы, конструктивно выполненные в виде одной БИС и включающие в себя все устройства, необходимые для реализации цифровой системы управления минимальной конфигурации: процессор, запоминающее устройство данных, запоминающее устройство команд, внутренний генератор тактовых сигналов, а также программируемые интегральные схемы для связи с внешней средой. Использование ОМЭВМ в системах управления обеспечивает достижение исключительно высоких показателей эффективности при столь низкой стоимости (во многих применениях система может состоять только из одной БИС ОМЭВМ), что им, видимо, нет в ближайшем времени альтернативной элементной базы для построения управляющих и/или регулирующих систем, В настоящее время более двух третей мирового рынка микропроцессорных средств составляют именно БИС ОМЭВМ. В некоторых публикациях однокристальную микроЭВМ (ОМЭВМ) называют “микроконтроллер”. Обосновывается это тем обстоятельством, что такие микросхемы имеют незначительные емкости памяти, физическое и логическое разделение памяти программ (ПЗУ) и памяти данных (ОЗУ), упрощенную и ориентированную на задачи управления систему команд, примитивные методы адресации команд и данных. Специфическая организация ввода-вывода информации предопределяет область их применения в качестве специализированных вычислителей, включенных в контур управления объектом или процессом. Структурная организация, набор команд и аппаратно-программные средства ввода-вывода информации этих микросхем лучше всего приспособлены для решения задач управления и регулирования в приборах, устройствах и системах автоматики, а не для решения задач обработки данных. Указанные выше соображения отражают технический уровень ОМЭВМ в настоящий момент.

Анализ и классификация аппаратных и программных средств контроллеров

**Классификация контроллеров**

Ускорение научно-технического прогресса и совершенствование хозяйственного механизма тесно связаны с развитием новейшей информационной технологии. Тенденция к большей децентрализации функций управления объективно требует совершенствования непосредственно рабочих мест пользователей, вовлеченных в процесс управления, прежде всего за счет их автоматизации. Очевидно, что без конкретных потребностей (например, в повышении производительности интеллектуального уровня личного труда, выполнение работы быстрее и экономичнее) едва ли возникла бы необходимость в контроллерах. Внедрение контроллеров позволяет реализовать качественно новую информационную технологию, при которой потребитель информации получает и анализирует интересующие его данные непосредственно в процессе общения с ПЭВМ, что обеспечивает комплексную автоматизацию конкретных задач управления. Работа контроллера помогает лучшему осознанию проблемной ситуации, позволяет частично или полностью решить проблему, подталкивает к оперативному построению модели объекта управления и, тем самым, к грамотному и быстрому принятию решений.

Попытаемся классифицировать контроллеры вообще.

Будем определять контроллеры как совокупность методических,

схемотехнических и программных средств, обеспечивающих работу пользователей в некоторой предметной области. Все три компонента совершенно необходимы для функционирования контроллера. Методика проектирования контроллера не может не быть связанной с методикой его функционирования, так как функционирование развитого контроллера не предусматривает возможность его развития самими пользователями. Языковые средства микроконтроллеров являются реализацией методических средств с точки зрения конечного пользователя, и дают возможность конечному пользователю производить все необходимые действия.

В состав методических средств должны входить и средства

обучения, диапазон которых может быть очень широк: от бумажных инструкций до самообучающихся систем. Их назначение заключается в том, чтобы пользователь научился эффективно работать в условиях автоматизации полного и частного процесса. Если процесс достаточно сложный, а пользователь не имеет специализированные навыков, возможно применение контроллеров, которые позволяют постепенно ввести пользователя в предметную область его автоматизированного процесса. Чем более сложнее проектируется контроллер, тем менее функции могут быть реализованы пользователем.

Следующий аспект методического обеспечения - это реализация заданных функций контроллера, т.е. собственно его функционирования. Здесь необходимы методики определения цели текущей деятельности, информационной потребности, всевозможных сценариев для описания процессов ее реализации. Эти методические средства могут явным или неявным образом учитываться пользователем. В первом случае пользователь в значительной степени сам планирует свою деятельность.

Языковые средства контроллера необходимы, прежде всего, для однозначного смыслового соответствия действий пользователя и реакции ПЭВМ. Без них невозможен процесс обучения, организация диалога, обнаружение и исправление ошибок. Сложность разработки таких языков заключается в том, что они должны быть преимущественно непроцедурными. Если процедурный язык указывает, как выполняется задаваемое, то непроцедурный - что необходимо выполнить без детализации того, какие действия для этого требуются. Так как конечные пользователи не знают в деталях процесс реализации информационной потребности, чем выше интеллектуальность контроллера, тем больше непроцедурных возможностей должно быть предусмотрено в его языках.

Языки программирования контроллера должны быть пользователь ориентированными, в том числе и профессионально-ориентированными. Это связано с различиями в классификации пользователей. Следует учесть, что использование естественного языка, не может дать сколько-нибудь ощутимых преимуществ из-за необходимости введения через клавиатуру громоздких конструкций ради получения иногда несложных результатов. Кроме того, затраты на построение лексического анализатора для такого контроллера могут сравняться с общими затратами на все остальные составляющие.

Как и во всяком языке, основу языков контроллеров должны составлять заранее определяемые термины, а также описания способов, с помощью которых могут устанавливаться новые термины, заменяя или дополняя существующие. Указанное обстоятельство приводит к необходимости при проектировании контроллера определенным образом классифицировать терминологическую основу контроллеров, т.е. определить все основные синтаксические конструкции языка и семантические отношения между терминами и их совокупностями. В связи с этим может возникнуть необходимость в простейшей классификации контроллеров, например, по возможностям представления данных в некоторых пользовательских режимах обработки: числовые, текстовые и смешанные. Возможности языка во многом определяют и список правил, по которым пользователь может строить формальные конструкции, соответствующие реализации информационной потребности.

Резко возросшая потребность в малоразмерной авиационной техники послужила толчком для развития авиационного оборудования в направлении уменьшения его габаритно-массовых характеристик .

Современные нормы летной годности, как отечественные НЛГС-3, так и зарубежные FAR-23, предъявляют высокие требования к безопасности полетов, что напрямую связано с надежностью применяемого оборудования, резервированием основных каналов информации и предоставлением пилоту ограничительных сигналов при возможно минимальном весе оборудования. Поэтому тема дипломной работы, направленной на создание указателя скорости с цифровым выходом УСВИЦ-250, является весьма актуальной.

На основании обзора методов измерения воздушных скоростей и обзора последних разработок указателей скорости разработана конструкция указателя, в которой использованы оригинальные технические решения, примененные в указателях УСВИ-200 –Э (подвижный индекс Vм.д.), УСБС-200-Э (выдача разовой команды Vпр ≥ Vм.д.) и УСЦ-2 (наличие цифрового выхода Vпр.ц.).

**Анализ инструментальных средств**

Возвращаясь к языково-программным классификационным характеристикам, нельзя не упомянуть о средствах разработки контроллера, т.е. о языках программирования.

Все языки программирования условно можно разделить на три уровня:

- машинный код;

- автокод (язык ассемблера);

- языки высокого уровня (процедурные языки - BASIC, FORTRAN, PASCAL, C, MODULA-2, ADA; и языки исскуственного интеллекта - LISP, PROLOG, SMALLTALK, OCCAM).

Машинно-ориентированные языки

Более понятные для ЭВМ - это так называемые машинно-ориентированные языки (машинный код и язык ассемблера). Более понятные для человека именуют языками высокого уровня.

Программное обеспечение на машинно-ориентированном языке экономично в эксплуатации, однако сравнительно высокая трудоемкость и длительность разработки программного обеспечения обуславливают преимущественное применение их для создания и развития программного обеспечения драйверов и операционных систем с целью наилучшего использования аппаратных особенностей каждой конкретной ЭВМ.

Языки высокого уровня

Алгоритмические языки (языки программирования высокого

уровня общего назначения) являются машинно-независимыми,

позволяют создавать компактные обозримые программы при относительно небольших затратах времени и труда программистов. Разработка программ значительно упрощается при использовании языков высокого уровня в качестве языков программирования. Однако при этом снижается эффективность программ по быстродействию и затратам памяти в сравнении с применением языка ассемблера. Но этот недостаток с лихвой перекрывается четкостью и легкостью написания программы.

Языки высокого уровня в свою очередь подразделяются на

языки процедурного (или императивного) и эвристического

(декларативного) стиля программирования (языки исскуственного интеллекта). Наиболее популярные языки программирования

ПЭВМ высокого уровня приведены в таблице 1.2.

Таблица 1.2.

|  |  |  |  |
| --- | --- | --- | --- |
| Язык | Год разработки | Разработчик | Основное применение |
| FORTRAN | 1954 | Дж. Бэкус (США) | Математические расчеты, научные исследования |
| BASIC | 1965 | Дж. Кенеми (США) | Обучение, тестовые программы |
| PASCAL | 1971 | Н.Вирт (Швейцария) | Обучение, широкое применение |
| C | 1972 | Д.М.Ричи (США) | Системное программирование |
| MODULA-2 | 1981 | Н.Вирт (Швейцария) | Разработка больших программных комплексов |
| LISP | 1960 | Дж. Маккарти (США) | Системы искусственного интеллекта |
| PROLOG | 1971 | А.Колмедауэр (Франция) | Принятие решений, логический вывод |
| SMALLTALK | Середина 1970-х | А.КейАнглия) | Системы диалога со средствами машинной графики |
| OCCAM | Начало 1980-х | Фирма INMOS(Англия) | Системы с параллельными процессами |

Кроме того, в настоящее время появились языки так называемого 4-го поколения - это языки СУБД, электронных таблиц, интегрированных систем и т.д., которые предназначены для решения узкого круга задач прикладного характера (например, обработка баз данных), но зато еще больше, по сравнению с языками общего назначения, снижают затраты времени и труда на создание выходного продукта.

Специализированные языки.

Опыт применения ПЭВМ для построения прикладных систем обработки данных показывает, что самым эффективным инструментом создания контроллера являются не универсальные языки высокого уровня, а узкоспециализированные языки - как правило языки высокого манипулирования с особенностями микропроцессора. Характеристики контроллера при этом определяются, прежде всего, принятой в

организацией данных и типом используемого транслятора.

Системы управления контроллера - это программный продукт, обеспечивающий хранение, обновление и выборку информации, представленной в формализованном виде на различных носителях.

Существует множество контроллеров, представляемых различными

производителями. Наибольшее распространение получили контроллеры на основе микрокомплекта 8051. Наибольшую популярность среди них имеет микроконтроллер фирмы INTEL 8051.

**С-51**

ICC8051 поддерживает не только 8051 (КР1816ВЕ51), но и все другие микро-ЭВМ, использующие ту же основную архитектуру и систему команд (8032, 8044, 80515, 80525 и т.д.). Может потребоваться только изменение объявлений регистров специального назначения и портов ввода/вывода.

Особенности ICC8051 V.4.00:

- два режима компиляции: стандарт ANSI и стандарт ANSI + расширение языка для 8051

- Шесть моделей памяти, обеспечивающих поддержку любых задач, от совсем простых до сложных систем, использующих переключение банков памяти

- Дополнительный атрибут для переменных, определяющий тип памяти размещения: DATA, IDATA, XDATA или CODE

- Статическое размещение локальных переменных позволяет сгенерировать компактный и быстрый код

- Переменные типа SFR (регистр специального назначения) и BIT

(прямоадресуемые биты) могут объявляться и использоваться как стандартные типы данных в языке Си

- Возможность написания на Си функций-обработчиков прерываний (включая определение векторов прерываний) без использования ассемблера

- Поставляемые вместе с компилятором обьявления SFR и BIT, специфичных для каждого члена семейства кристаллов 8051.

Компилятор Си ICC8051 поддерживает все основные элементы языка Си в стандарте ANSI. Переменные располагаются в памяти, начиная со старшего байта и заканчиваются младшим ("старший байт - по младшему адресу").

Переменные всегда плотно упакованы в памяти, т.к. архитектура 8051 не требует выравнивания данных.

[1] Специфические для 8051 типы данных sfr и bit отличаются от стандартных типов данных языка Си, т.к. размещаются в особых областях памяти.

[2] Описание "char" эквивалентно "unsigned char", но может быть изменено на "signed char" с помощью опции компилятора -с.

[3] Перечисляемые переменные ("enum") размещаются с использованием наиболее короткого возможного типа данных, определяемого диапазоном перечисляемых констант - от "char" до "int".

**Assembler (INTEL 8051)**

Система микропрограммирования является набором компактных программных продуктов для разработки программ для микропроцессоров. СМ реализована для работы на ряде компьютеров, от небольших 16-разрядных персональных машин до 32-разрядных суперминикомпьютеров.

В нем имеется ряд примеров использования как стандартных", так и имеющих особенности средств СМ. Отметим, что независимые средства ассемблера СМ очень просты и эффективны.

СМ ассемблеры - это мощные МАКРО-ассемблеры со средствами перемещения

программ, с универсальными характеристиками и применением. Хотя ассемблеры созданы на базе одного и того же основного пакета, они обладают высокой степенью совместимости с ассемблерами разработчиков микропроцессоров. Основные предметы - это способы использования ассемблера, поддержка модульного программирования и связь с языками высокого уровня.

Все ассемблеры двухпроходные, выполняются как одна программа. Во время выполнения не создается временных файлов.

Все ассемблеры, так же как и ХLINK, используют для внутренних вычислений 32-разрядные структуры, что позволяет виртуально генерировать код любого размера (т.е. не существует предела в 64 кБайт, что могло бы затруднить использование процессоров типа 68000).

Для обеспечения совместимости внутри пакета было применено несколько компромиссных решений, с учетом совместимости с ассемблерами разработчиков микропроцессоров. Особенно это касается макро-конструкций, которые сильно различаются у различных разработчиков. Во многих отношениях, однако, СМ превосходит оригинальные ассемблеры. Следующая таблица дает грубое

представление о степени совместимости этого пакета:

Совместимы по:

- машинным командам (именам и синтаксису)

- директивам определения констант (именам и синтаксису)

- директивам распределения памяти (именам и синтаксису)

- разделителям

- меткам

- основным операторам (+,-,\*,/)

- ORG и EQU

Не совместимы по:

- директивам перемещения

- расширениям операторов

- средствам условной трансляции

- опциям и командам управления ассемблером

- макросредствам.

Заметим, что средства, перечисленные в разделе "не совместимы", часто отличаются от оригинальных ассемблеров разработчиков только синтаксисом.

Наиболее популярными на сегодняшний день у программистов являются С-51 и Assembler 8051, так как оба они позволяют получать исходный код.

Постановка задачи

Результатом данного проектирования должен явиться программный продукт контроллера, обеспечивающий данными вычислитель о скорости летательного аппарата. Система должна обеспечивать считывание с датчиков и обработку данных, используя следующий набор операций: считывание, обработка, формирование слов, выдача на ЦАП приборной скорости.

Входными данными служат:

* счтанные данные с маски приборной скорости;
* счтанные данные с маски максимальной скорости;

Выходными формами служат :

* слово 206 в соответствии с РТМ;
* слово 207 в соответствии с РТМ;
* слово 271 в соответствии с РТМ;
* данные с ЦАП приборной скорости.

Кроме того, система должна включать в себя следующие подсистемы:

* контроль программного обеспечения;
* контроль о выходе из строя маски приборной скорости;
* контроль о выходе из строя маски максимальной скорости;
* контроль о превышении скорости более чем на 2 км/ч, сигнализируя о перегрузки;
* частота обновления минимальная 31 мсек по каждому значению и максимальная 60 мсек по каждому значению.

Цена младшего разряда по 206 и 207 слову 0,11575

В качестве базового языка программирования использовать

язык С-51 совместно с Assembler 8051.

Выводы

В настоящем разделе выпускной работы были проведены следующие работы:

- анализ автоматизированных контроллеров, и микропроцессоров, на основе чего произведена их классификация;

- рассмотрена и проанализирована проблема выбора инструментальных средств для разработки контроллера, на основании чего выбраны языки С-51 и Assembler;

- на основании анализа требуемого к контроллера было сформулирована задача и требования к комплексу программ деятельности контроллера.

Обобщенная структура и алгоритм функционирования контроллера

Данные в системе представляют собой информацию в виде переменных, которым жестко заданы адреса переменных для обмена между программами обслуживания контроллера.

Выбор инструментальных средств и элементной базы.

При выборе элементной базы прибора контроля учитывались следующие обстоятельства:

- построить разрабатываемую технику на современном уровне (по функциям, стоимости, энергопотреблению и габаритам) исключительно на отечественных элементах не представляется возможным;

- так как мощность, потребляемая МК, в значительной мере определяет возможности разработки экономичной аппаратуры, применения кабелей питания небольшого сечения и должна быть выбрана микро мощная элементная база. Часть схемы контроллера может быть выполнена на элементной базе с небольшим быстродействием, однако для другой части схемы должны быть применены микро мощные элементы с высоким быстродействием. Одним из, таким элементом является микроконтроллер Intel 87C51FB.

Применяемый в разработке микроконтроллер Intel 87C51FB обладает следующими основными характеристиками, которые необходимо знать для выполнения поставленной в дипломном проекте задаче.

Intel87C51FB относится к семейству n-МОП технологии и выпускается в корпусе БИС, имеющем 40 внешних выводов. Цоколевка корпуса и наименования выводов показаны на рис 3.1. Для работы Intel87C51FB требуется один источник электропитания +5 В. Через четыре программируемых порта ввода/вывода Intel87C51FB взаимодействует со средой в стандарте ТТЛ-схем с тремя состояниями выхода. Корпус Intel87C51FB имеет два вывода для подключения кварцевого резонатора, четыре вывода для сигналов, управляющих режимом работы МК, и восемь линий порта 3, которые могут быть запрограммированы пользователем на выполнение специализированных (альтернативных) функций обмена информацией со средой.

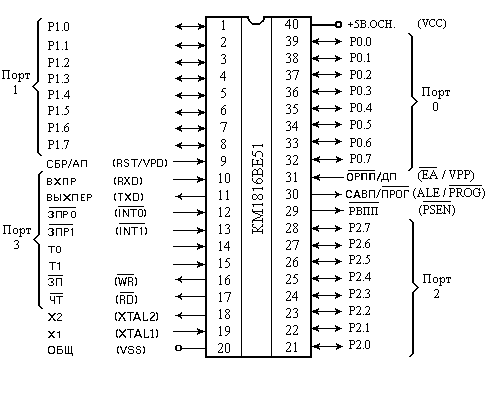


Рис 3.1.

Разработки на базе контроллеров Intel87C51FB поддерживается ассемблером, программным симулятором, внутрисхемным эмулятором фирмы Phyton и программатором.

Серия Intel87C51FB подходит для широкого спектра приложений от схем высокоскоростного управления автомобильными и электрическими двигателями до экономичных удаленных приемопередатчиков, показывающих приборов и связных процессоров. Наличие ПЗУ позволяет подстраивать параметры в прикладных программах (коды передатчика, скорости двигателя, частоты приемника и т.д.).

Малые размеры корпусов, как для обычного, так и для поверхностного монтажа, делает эту серию микроконтроллеров пригодной для портативных приложений. Низкая цена, экономичность, быстродействие, простота использования и гибкость ввода/вывода делает Intel87C51FB привлекательным даже в тех областях, где ранее не применялись микроконтроллеры. Например, таймеры, замена жесткой логики в больших системах, сопроцессоры.

Приведем краткий обзор характеристик МК:

- 111 базовых команд;

- Большинство команд (94) имеют формат один или два байта и выполняются за один или два машинных цикла;

- рабочая частота 12 МГц

- -байтное АЛУ и схемы аппаратурной реализации команд умножения и деления;

- 14 - битовые команды;

- двунаправленная 8-битная шина;

- стираемое программируемое ПЗУ программ емкостью 64 Кбайт;

- ОЗУ данных емкостью 128 байта;

- два 16-битных таймера/счетчика;

- программируемые схемы ввода/вывода (32 линий);

- блок двухуровневого векторного прерывания от пяти источников;

- асинхронный канал дуплексного последовательного ввода/вывода информации со скоростью до 375 Кбит/с;

- генератор, схему синхронизации и управления;

- 8 х 4 регистров общего использования;

- 128 программно-управляемых флагов;

- прямая, косвенная и относительная адресация данных и команд;

- четыре источника прерывания:

Структурная схема микропроцессора

Рассмотрим структурную схему микроконтроллера Intel87C51FB рисунок 3.2.

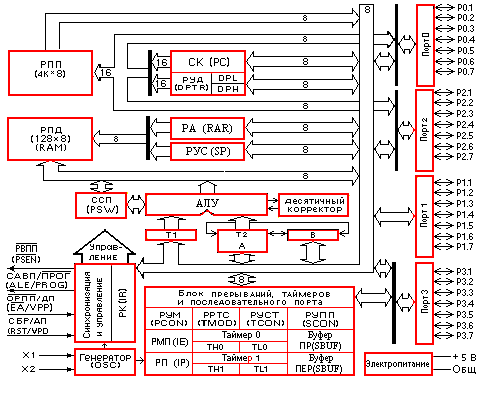


Рис 3.2.

Основу структурной схемы образует внутренняя двунаправленная 8-битная шина, которая связывает между собой все основные узлы и устройства: резидентную память, АЛУ, блок регистров специальных функций, устройство управления и порты ввода/вывода. Рассмотрим основные элементы структуры и особенности организации вычислительного процесса.

**Арифметическо-логическое устройство**

8-битное АЛУ может выполнять арифметические операции сложения, вычитания, умножения и деления; логические операции И, ИЛИ, исключающее ИЛИ, а также операции циклического сдвига, сброса, инвертирования и т.п. В АЛУ имеются программно недоступные регистры T1 и T2, предназначенные для временного хранения операндов, схема десятичной коррекции и схема формирования признаков.

Простейшая операция сложения используется в АЛУ для инкрементирования содержимого регистров, продвижения регистра-указателя данных и автоматического вычисления следующего адреса РПП. Простейшая операция вычитания используется в АЛУ для декрементирования регистров и сравнения переменных.

Простейшие операции автоматически образуют "тандемы" для выполнения в АЛУ таких операций, как, например, инкрементирование 16-битных регистровых пар. В АЛУ реализуется механизм каскадного выполнения простейших операций для реализации сложных команд. Так, например, при выполнении одной из команд условной передачи правления по результату сравнения в АЛУ трижды инкрементируется СК, дважды производится чтение из РПД, выполняется арифметическое сравнение двух переменных, формируется 16-битный адрес перехода и принимается решение о том, делать или не делать переход по программе. Все перечисленные операции выполняются в АЛУ всего лишь за 2 мкс.

Важной особенностью АЛУ является его способность оперировать не только байтами, но и битами. Отдельные программно-доступные биты могут быть установлены, сброшены, инвертированы, переданы, проверены и использованы в логических операциях. Эта способность АЛУ, оперировать битами, столь важна, что во многих описаниях Intel87C51FB говорится о наличии в нем "булевского процессора". Для управления объектами часто применяются алгоритмы, содержащие операции над входными и выходными булевскими переменными (истина/ложь), реализация которых средствами обычных микропроцессоров сопряжена с определенными трудностями.

Таким образом, АЛУ может оперировать четырьмя типами информационных объектов: булевскими (1 бит), цифровыми (4 бита), байтными (8 бит) и адресными (16 бит). В АЛУ выполняется 51 различная операция пересылки или преобразования этих данных. Так как используется 11 режимов адресации (7 для данных и 4 для адресов), то путем комбинирования "операция/ режим адресации" базовое число команд 111 расширяется до 255 из 256 возможных при однобайтном коде операции.

**Резидентная память**

Память программ и память данных, размещенные на кристалле физически и логически разделены, имеют различные механизмы адресации, работают под управлением различных сигналов и выполняют разные функции.

Память программ (ПЗУ или СППЗУ) имеет емкость 64 Кбайта и предназначена для хранения команд, констант, управляющих слов инициализации, таблиц перекодировки входных и выходных сменных и т.п. РПП имеет 16-битную шину адреса, через которую обеспечивается доступ из счетчика команд или из регистра-указателя данных. Последний выполняет функции базового регистра при косвенных переходах по программе или используется в командах, оперирующих с таблицами.

Память данных (ОЗУ) предназначена для хранения переменных в процессе выполнения прикладной программы, адресуется одним байтом и имеет емкость 128 байт. Кроме того, к адресному пространству РПД примыкают адреса регистров специальных функций (РСФ), которые перечислены в табл. 3.1.

Память программ, так же как и память данных, может быть расширена путем подключения внешних БИС.

Аккумулятор и ССП. Аккумулятор является источником операнда и местом фиксации результата при выполнении арифметических, логических операций и ряда операций передачи данных. Кроме того, только с использованием аккумулятора могут быть выполнены операции сдвигов, проверка на нуль, формирование флага паритета и т.п.

**Таблица 3.1.**

|  |  |  |
| --- | --- | --- |
| **Символ** | **Наименование** | **Адрес** |
| **\*** ACC | Аккумулятор | 0E0H |
| **\*** B | Регистр-расширитель аккумулятора | 0F0H |
| **\*** PSW | Слово состояния программы | 0D0H |
| SP | Регистр-указатель стека | 81H |
| DPTR | Регистр-указатель данных (DPH)  (DPL) | 83H |
| 82H |
| **\*** P0 | Порт 0 | 80H |
| **\*** P1 | Порт 1 | 90H |
| **\*** P2 | Порт 2 | 0A0H |
| **\*** P3 | Порт 3 | 0B0H |
| **\*** IP | Регистр приоритетов | 0B8H |
| **\*** IE | Регистр маски прерываний | 0A8H |
| TMOD | Регистр режима таймера/счетчика | 89H |
| **\*** TCON | Регистр управления/статус таймера | 88H |
| TH0 | Таймер 0 (старший байт) | 8CH |
| TL0 | Таймер 0 (младший байт) | 8AH |
| TH1 | Таймер 1 (старший байт) | 8DH |
| TL1 | Таймер 1 (младший байт) | 8BH |
| **\*** SCON | Регистр управления приемопередатчиком | 98H |
| SBUF | Буфер приемопередатчика | 99H |
| PCON | Регистр управления мощностью | 87H |
| **Примечание.** Регистры, имена которых отмечены знаком (**\***), допускают адресацию отдельных бит. | | |

При выполнении многих команд в АЛУ формируется ряд признаков операции (флагов), которые фиксируются в регистре ССП. В табл. 3.2. приводится перечень флагов ССП, даются их символические имена и описываются условия их формирования.

**Таблица 3.2.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Символ** | **Позиция** | **Имя и назначение** | | | |
| C | PSW.7 | Флаг переноса. Устанавливается и сбрасывается аппаратурными средствами или программой при выполнении арифметических и логических операций | | | |
| AC | PSW.6 | Флаг вспомогательного переноса. Устанавливается и сбрасывается только аппаратурными средствами при выполнении команд сложения и вычитания и сигнализирует о переносе или заем в бите 3 | | | |
| F0 | PSW.5 | Флаг 0. Может быть установлен, сброшен или проверен программой как флаг, специфицируемый пользователем | | | |
| RS1  RS0 | PSW.4  PSW.3 | Выбор банка регистров. Устанавливается и сбрасывается программой для выбора рабочего банка регистров (см. примечание) | | | |
| OV | PSW.2 | Флаг переполнения. Устанавливается и сбрасывается аппаратно при выполнении арифметических операций | | | |
| **\_** | PSW.1 | Не используется | | | |
| P | PSW.0 | Флаг паритета. Устанавливается и сбрасывается аппаратуры в каждом цикле команды и фиксирует нечетное/четное число единичных бит в аккумуляторе, т.е. выполняет контроль по четности | | | |
| **Примечание.** | | RS1 | RS0 | Банк | Границы адресов |
| 0 | 0 | 0 | 00H-07H |
| 0 | 1 | 1 | 08H-0FH |
| 1 | 0 | 2 | 10H-17H |
| 1 | 1 | 3 | 18H-1FH |

Наиболее "активным" флагом ССП является флаг переноса, которые принимает участие и модифицируется в процессе выполнения множеств операций, включая сложение, вычитание и сдвиги. Кроме того, флаг переноса (C) выполняет функции "булевого аккумулятора" в командах, манипулирующих с битами. Флаг переполнения (OV) фиксирует арифметическое переполнение при операциях над целыми числами со знаком и делает возможным использование арифметики в дополнительных кодах. АЛУ не управляет флагами селекции банка регистров (RS0, RS1), и их значение полностью определяется прикладной программой и используется для выбора одного из четырех регистровых банков.

Широкое распространение получило представление о том, что в микропроцессорах, архитектура которых опирается на аккумулятор, большинство команд работают с ним, используя адресацию "по умолчанию" (неявную). В Intel87C51FB дело обстоит иначе. Хотя процессор имеет в своей основе аккумулятор, однако, он может выполнять множество команд и без участия аккумулятора. Например, данные могут быть переданы из любой ячейки РПД в любой регистр, любой регистр может быть загружен непосредственным операндом и т.д. Многие логические операции могут быть выполнены без участия аккумулятора. Кроме того, переменные могут быть инкрементированы, декрементированы и проверены (test) без использования аккумулятора. Флаги и управляющие биты могут быть проверены и изменены аналогично.

Регистры-указатели.

8-битный указатель стека (РУС) может адресовать любую область РПД. Его содержимое инкрементируется прежде, чем данные будут запомнены в стеке в ходе выполнения команд PUSH и CALL. Содержимое РУС декрементируется после выполнения команд POP и RET. Подобный способ адресации элементов стека называют прединкрементным/постдекрементным. В процессе инициализации Intel87C51FB после сигнала СБР в РУС автоматически загружается код 07H. Это значит, что если прикладная программа не переопределяет стек, то первый элемент данных в стеке будет располагаться в ячейке РПД с адресом 08Н.

Двухбайтный регистр-указатель данных (РУД) обычно используется для фиксации 16-битного адреса в операциях с обращением к внешней памяти. Командами МК51 регистр-указатель данных может быть использован или как 16-битный регистр, или как два независимых 8-битных регистра (DPH и DPL).

Таймер/счетчик.

В составе средств Intel87C51FB имеются регистровые пары с символическими именами ТНО, TLO и THI, TLI, на основе которых функционируют два независимых программно-управляемых 16-битных таймера/счетчика событий.

Буфер последовательного порта. Регистр с символическим именем SBUF представляет собой два независимых регистра - буфер приемника и буфер передатчика. Загрузка байта в SBUF немедленно вызывает начало процесса передачи через последовательный порт. Когда байт считывается из SBUF, это значит, что его источником является приемник последовательного порта.

Регистры специальных функций. Регистры с символическими именами IP, IE, TMOD, TCON, SCON и PCON используются для фиксации и программного изменения управляющих бит и бит состояния схемы прерывания, таймера/счетчика, приемопередатчика последовательного порта и для управления мощностью электропитания Intel87C51FB.

Все четыре порта предназначены для ввода или вывода информации побайтно. Каждый порт содержит управляемые регистр-защелку, входной буфер и выходной драйвер. Все выводы порта 3 могут быть использованы для реализации альтернативных функций, перечисленных в табл. 3.3.. Альтернативные функции могут быть задействованы путем записи 1 в соответствующие биты регистра-защелки (Р3.0-Р3.7) порта 3.

**Таблица 3.3.**

|  |  |  |
| --- | --- | --- |
| **Символ** | **Позиция** | **Имя и назначение** |
| **¬RD** | **P3.7** | **Чтение. Активный сигнал низкого уровня формируется аппаратурно при обращении к ВПД** |
| **¬WR** | **P3.6** | **Запись. Активный сигнал низкого уровня формируется аппаратурно при обращении к ВПД** |
| **T1** | **P3.5** | **Вход таймера/счетчика 1 или тест-вход** |
| **T0** | **P3.4** | **Вход таймера/счетчика 0 или тест-вход** |
| **¬INT1** | **P3.3** | **Вход запроса прерывания 1. Воспринимается сигнал низкого уровня или срез** |
| **¬INT0** | **P3.2** | **Вход запроса прерывания 0. Воспринимается сигнал низкого уровня или срез** |
| **TXD** | **P3.1** | **Выход передатчика последовательного порта в режиме УАПП. Выход синхронизации в режиме сдвигающего регистра** |
| **RXD** | **P3.0** | **Вход приемника последовательного порта в режиме УАПП. Ввод/вывод данных в режиме сдвигающего регистра** |

Порт 0 является двунаправленным, а порты 1, 2 и З - квазидвунаправленными. Каждая линия портов может быть использована независимо для ввода или вывода информации. Для того чтобы некоторая линия порта использовалась для ввода, в D-триггер регистра-защелки порта должна быть записана 1, которая закрывает МОП-транзистор выходной цепи.

По сигналу СБР в регистры-защелки всех портов автоматически записываются единицы, настраивающие их тем самым на режим ввода.

Все порты могут быть использованы для организации ввода/вывода информации по двунаправленным линиям передачи. Однако порты 0 и 2 не могут быть использованы для этой цели в случае, если МК-система имеет внешнюю память, связь с которой организуется через общую разделяемую шину адреса/данных, работающую в режиме временного мультиплексирования.

Запись в порт.

При выполнении команды, которая изменяет содержимое регистра-защелки порта, новое значение фиксируется в регистре в момент S6P2 последнего цикла команды. Однако опрос содержимого регистра-защелки выходной схемой осуществляется во время фазы Р1 и, следовательно, новое содержимое регистра-защелки появляется на выходных контактах порта только в момент S1P1 следующего машинного цикла.

Нагрузочная способность портов. Выходные линии портов 1, 2 и 3 могут работать на одну ТТЛ-схему. Линии порта 0 могут быть нагружены на два входа ТТЛ-схем каждая. Линии порта 0 могут работать и на n-МОП-схемы, однако при этом их необходимо подключать на источник электропитания через внешние нагрузочные резисторы за исключением случая, когда шина порта 0 используется в качестве шины адреса/данных внешней памяти.

Входные сигналы для МК51 могут формироваться ТТЛ-схемами или n-МОП-схемами. Допустимо использование в качестве источников сигналов для Intel87C51FB схем с открытым коллектором или открытым стоком. Однако при этом время изменения входного сигнала при переходе из 0 в 1 окажется сильно затянутым.

Особенности работы портов.

Обращение к портам ввода/вывода возможно с использованием команд, оперирующих с байтом, отдельным битом и произвольной комбинацией бит. При этом в тех случаях, когда порт является одновременно операндом и местом назначения результата, устройство управления автоматически реализует специальный режим, который называется "чтение-модификация-запись". Этот режим обращения предполагает ввод сигналов не с внешних выводов порта, а из его регистра-защелки, что позволяет исключить неправильное считывание ранее выведенной информации.

Обзор типов команд

Большинство команд (94) имеют формат один или два байта и выполняются за один или два машинных цикла. При тактовой частоте 2 МГц длительность машинного цикла составляет 1 мкс. Обычно первый байт команды любых типа и формата всегда содержит код операции (КОП). Второй и третий байты содержат либо адреса операндов, либо непосредственные операнды.

Типы операндов.

Состав операндов Intel87C51FB включает в себя операнды четырех типов: биты, 4-битные цифры, байты и 16-битные слова. Пример рис 3.3.

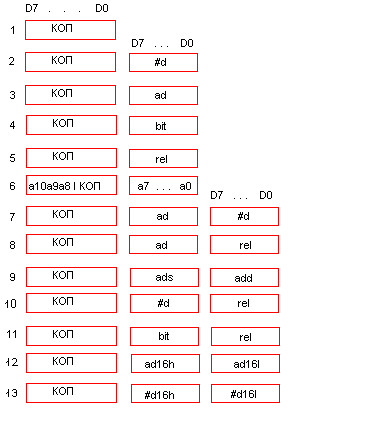


Рис 3.3.

Intel87C51FB имеет 128 программно-управляемых флагов пользователя. Имеется также возможность адресации отдельных бит блока регистров специальных функций и портов. Для адресации бит используется прямой 8-битный адрес (bit). Косвенная адресация бит невозможна. Карты адресов отдельных бит представлены на рис и рис . Четырехбитные операнды используются только при операциях обмена (команды SWAP и XCHD). Восьмибитным операндом может быть ячейка памяти программ или данных (резидентной или внешней), константа (непосредственный операнд), регистры специальных функций (РСФ), а также порты ввода/вывода. Порты и РСФ адресуются только прямым способом. Байты памяти могут адресоваться также и косвенным образом через адресные регистры (R0, R1, DPTR и PC). Двухбайтные операнды - это константы и прямые адреса, для представления которых используются второй и третий байты команды.

Способы адресации данных.

В Intel87C51FB используются такие способы адресации данных: прямая, непосредственная, косвенная и неявная. Следует отметить, что при косвенном способе адресации РПД используются все восемь бит адресных регистров R0 и R1.

Флаги результата.

Слово состояния программы (PSW) включает в себя четыре флага: C - перенос, AC - вспомогательный перенос, O - переполнение и P - паритет.

Флаг паритета напрямую зависит от текущего значения аккумулятора. Если число единичных бит аккумулятора нечетное, то флаг P устанавливается, а если четное - сбрасывается попытки изменить флаг Р, присваивая ему новое значение, будут безуспешными, если содержимое аккумулятора при этом останется неизменным.

Флаг АС устанавливается в случае, если при выполнении операции сложения/вычитания между тетрадами байта возник перенос/заем.

Флаг C устанавливается, если в старшем бите результата возникает перенос или заем. При выполнении операций умножения и деления флаг C сбрасывается. Флаг OV устанавливается, если результат операции сложения/вычитания не укладывается в семи битах и старший (восьмой) бит результата не может интерпретироваться как знаковый. При выполнении операции деления флаг OV сбрасывается, а в случае деления на нуль устанавливается. При умножении флаг OV устанавливается, если результат больше 255.

В табл. 3.5 перечисляются команды, при выполнении которых модифицируются флаги результата. В таблице отсутствует флаг паритета, так как его значение изменяется всеми командами, которые изменяют содержимое аккумулятора. Кроме команд, приведенных в таблице, флаги модифицируются командами, в которых местом назначения результата определены PSW или его отдельные биты, а также командами операций над битами.

**Таблица 3.5.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Команды** | **Флаги** |  | **Команды** | **Флаги** |
| ADD | C, OV, AC | CLR C | C = 0 |
| ADDC | C, OV, AC | CPL C | C = **¬**C |
| SUBB | C, OV, AC | ANL C, b | C |
| MUL | C = 0, OV | ANL C, /b | C |
| DIV | C = 0, OV | ORL C, b | C |
| DA | C | ORL C, /b | C |
| RRC | C | MOV C, b | C |
| RLC | C | CJNE | C |
| SETB C | C = 1 |  |  |

Символическая адресация.

При использовании ассемблера ASM51 для получения объектных кодов программ допускается применение в программах символических имен регистров специальных функций (РСФ), портов и их отдельных бит рис. 3.4., рис 3.5.

Для адресации отдельных бит РСФ и портов (такая возможность имеется не у всех РСФ) можно использовать символическое имя бита следующей структуры:

**< имя РСФ или порта > . <номер бита>**

Например, символическое имя пятого бита аккумулятора будет следующим: АСС.5. Символические имена РСФ, портов и их бит являются зарезервированными словами для Intel87C51FB, и их не надо определять с помощью директив ассемблера.

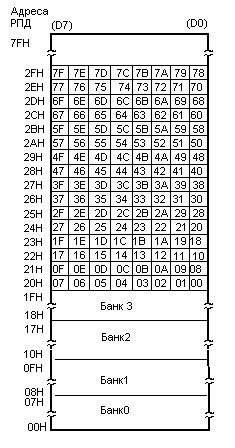


Рис 3.4.

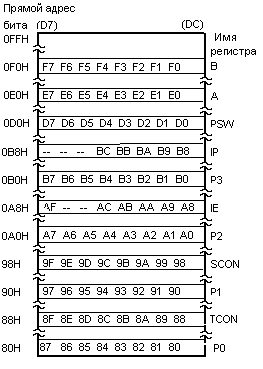


Рис 3.5.

Выбор Инструментальных средств

При выборе инструментальных средств разработки было необходимо обеспечить удобство и быстроту создания и отладки программного обеспечения микроконтроллера. Программа отлаживалась в интегрированной среде PICE-51, которую можно охарактеризовать следующим образом. PICE под Windows основная интегрирующая среда разработки для технологии микрокристалла включающей INTEL87C51FB семейств микроконтроллера. PICE позволяет пользователю писать, отлаживать, и оптимизировать многие процессоры для разработки микропрограмм. PICE включает текстовый редактор, имитатор, и руководитель проекта. PICE также имеет внутрисхемный эмулятор, программирующие устройства, и другие системы разработки инструментальных средств микрокристалла. Инструментальные средства PICE реализуют для пользователя следующие функции:

Отладка его исходной программы;

Эмуляция процессоров:

**Intel**: 80C31/32, 8XC51/52/54/58, 8XL/C51FA/FB/FC, 8XC51RA/RB/RC;

**Philips**: 80C31/32, 8XC51/52/54/58, 8XC51FA/FB/FC; 8XC51RA+/RB+/RC+/RD+, 8XC524, 8XC528, 8XC550, 8XC652, 8XC654, 8XC575, 8XC576, 8XC851, 89C535,89C536, 89C538;

**Atmel**: 89C51, 89C52, 89C55, 89C4051, 89C2051, 89C1051;

**Siemens**: SAB501;

**MHS**: 80C31, 8XC51, 80C32, 8XC52, 8XC154;

**Oki**: 80C31, 8XC51, 8XC154;

**AMD**: 80C31, 8XC51, 8XC52;

Автоматически располагает ошибки в исходных файлах для редактирования;

Отладка с контрольными точками, базировавшимися на внутренних величинах регистра;

· Часы ход программы с имитатором или часами в реальном времени с внутрисхемным эмулятором POD51-RX.

· Делать синхронизацию размеров со стоповыми часами;

· Видеть переменные в специальных окнах;

· Находить быстрые ответы на вопросы с PICE онлайновой подсказки;

Встроенная разработка среды Project-51 позволяет пользователю быстро перемещаться между другими способами разработки/отладки.

Встроенная разработка среды Project-51 включает инструментальные средства следующей разработки:

Руководитель проекта позволяет пользователю создавать проект, и работать со специфическими файлами связанными с проектом. Project-51 руководитель проекта также позволяет пользователю формировать приложение и загружать программный код в эмулятор или имитатор с нажатием мыши.

Project-51 это:

редактор позволяет программисту писать и редактировать исходные файлы микропрограмм или другие текстовые файлы для INTEL87C51FB микроконтроллеров;

универсальный ассемблер микроконтроллеров позволяет пользователю получать программный код;

программный имитатор имитирует выполнение инструкции и I/O микроконтроллеров.

Внутрисхемный эмулятор POD51-RX использует аппаратные средства, чтобы эмулировать INTEL87C51FB микроконтроллеров в реальном времени. Аппаратные средства POD51-RX эмулятора продаются отдельно.

После установки и компиляции проектов в Project-51, программист захочет увидеть как сделаны его кодовые прогоны. Если у вас есть программатор устройства, вы можете запрограммировать устройство микроконтроллера и подключите программируемое устройство в вашем фактическом приложении, чтобы проверить, что приложение работает как ожидается. Обычно, приложение не будет выполняться правильно сначала, и программист должен отладить код. Отладчик может использовать Project-51, чтобы с имитировать программный код или может использовать эмулятор POD51-RX, чтобы выполнять созданные микропрограммы в фактическом приложении пока программист производит отлаживание созданной программы. Во время отладки, пользователь может использовать прерывание и точки останова, когда выполняется код его программы. Можно наблюдать величины регистра в окне регистра или в специальном функциональном окне регистра, чтобы увидеть состояние процессора и следовательно выполняемый код и пошаговый режим.

Внутрисхемный эмулятор POD51-RX выполняет код в фактической скорости выполнения (реальное время) в разрабатываемых целевых аппаратных средствах, останавливаясь только в определенных контрольных точках. Project-51 имитирует выполнение любого процессора и имитирует условия I/O в скоростях, которые зависят от скорости вашего PC.

Все эти функции используют информацию из Project-51 проекта. Строка помечается в исходной программе, символических позициях в памяти, и функциональные имена из кода могут использоваться, чтобы установить прерывание и точки останова и изучать и модифицировать регистры.

Исходя из всего выше сказанного, выбор элементной базы был остановлен на уже перечисленных микросхемах. Выбор инструментальных средств разработки обоснован необходимостью применения интегрированной среды Project-51 для отладки ассемблерных программ контроллеров.

Разработка структуры программного обеспечения подсистемы.

В процессе дипломного проектирования необходимо разработать подсистему контроля вскрытия персонального компьютера обеспечивающую постоянный контроль поставленных на охрану объектов и выдачу информации о их состоянии на АРМ СБ.

Программа контроллера УСВИЦ-250 должна выполнять следующие основные функции:

Осуществлять считывание с масок кодов скоростей;

Осуществлять перевод этих кодов в двоичный код;

Осуществлять формирование 32х-разрядного слова по ARING-427;

Осуществлять вывод в магистраль выходного слова;

Осуществлять вывод аналогового сигнала о приборной скорости.

Результатом анализа и осмысления, поставленных при разработке данной подсистемы требований явилась следующая структура подпрограмм в основной программе, обеспечивающей выполнения поставленных требований:

1. Подпрограмма, взаимодействующая масок с МК должна выполнять следующие возложенные на нее функции:

- принимать коды грея;

2. Подрограмма, осуществляющая перевод кодов по формуле:

Код\_Двоичный4=Код\_Грея4;

Код\_Двоичный3=Код\_Грея3 ⊕ Код\_Двоичный4;

Код\_Двоичный2=Код\_Грея2 ⊕ Код\_Двоичный3;

Код\_Двоичный1=Код\_Грея1 ⊕ Код\_Двоичный2;

Код\_Двоичный0=Код\_Грея0 ⊕ Код\_Двоичный1;

3. Подрограмма, формирование 32х-разрядного слова по ARING-427 должна выполнять следующие функции:

- привести значение скорости к цене младшего разряда;

занести по адресам 8-9 в слове индификатор от УСВИЦ-250;

занести по адресам 30-29 в слове о достоверности слова;

подсчитать четность слова и занести по адресу 31 в слове.

Занести по адресам 0-8 в слове адрес слова;

4. Подпрограмма, осуществляющая выдачу слова в магистраль с частотой 12.5КГц, должна выполнять следующие возложенные на нее функции:

- обеспечивать частоту вывода 12.5КГц;

обеспечивать выдачу последовательно 206, 207 и 271 слова;

Подпрограмма, осуществляющая выдачу приборной скорости в аналоговом виде и должна выполнять следующие возложенные на нее функции:

записи в регистр ЦАП приборной скорости;

сделать паузу для записи в регистр;

установить флаг в ЦАП о записи в регистр.

Файловая структура подсистемы представляет собой совокупность шести файлов, один из которых содержит основной текст программы SPEED.C , со всеми вышеперечисленными основными и вспомогательными подпрограммами. Другие файлы - содержат подпрограммы написаны на ассемблере используемых в основной программе. При помощи интегрированной среды Project-51 программа компилируется и, в случае, успешного результата обработки, создает файл SPEED.HEX, содержащий исходный код программы в шестнадцатиричном виде. Полученный исполняемый программный код при помощи программатора, подключенного к компьютеру, заносится в микроконтроллер для последующего выполнения и тестирования, обнаружения возможных допущенных при программировании ошибок.

Для отладки созданного программного кода, внутрисхемный эмулятор устанавливался на опытный образец печатной платы с напаянными на ней всеми необходимыми для корректной работы микросхемами и навесными элементами (резисторы, конденсаторы). Все микросхемы, установленные на плате, в совокупности с микроконтроллером и его программным обеспечением, составляют разрабатываемый контроллер. Подобная отладка, с применением осциллографа, контрольного пульта ПК-10, дает вполне ясную картину программно генерируемых сигналов с микроконтроллера. Осциллограф позволяет в удобной форме проследить периодичность генерации сигналов и длительность процессов, которые они вызывают в схеме, а ПК-10 достоверность выходного слова. Также довольно легко наблюдать принимаемые микроконтроллером на обработку сигналы, поступающие с других микросхем. Применение осциллографа и ПК-10, при отладке программы на реальной аппаратуре, необходимо, и безусловно, значительно облегчает отладку и доработку исходного текста программы.

Описание структуры полей программы

Переменные для временных обработак и константы с их адресами

const int speed=172 - служит для добавления при наземном контролле контроллера;

const int test\_speed=2 - служит для добавления при проверки масок приборной и максимальной скорости;

static int a @ 0x18 - временная преременная;

static int flag @ 0x18 - временная преременная;

static int flag1 @ 0x1a - временная преременная;

Переменные для выходного слова адрес 206 по РТМ с их адресами

static unsigned long SlVmd @ 0x10;

static unsigned char SlVmd\_1 @ 0x10;

static unsigned char SlVmd\_2 @ 0x11;

static unsigned char SlVmd\_3 @ 0x12;

static unsigned char SlVmd\_4 @ 0x13;

Переменные для выходного слова адрес 207 по РТМ с их адресами

static unsigned long SlVpr @ 0x3c;

static unsigned char SlVpr\_1 @ 0x3c;

static unsigned char SlVpr\_2 @ 0x3d;

static unsigned char SlVpr\_3 @ 0x3e;

static unsigned char SlVpr\_4 @ 0x3f;

Переменные для выходного слова адрес 271 по РТМ с их адресами

static unsigned long SlDs @ 0x38;

static unsigned char SlDs\_1 @ 0x38;

static unsigned char SlDs\_2 @ 0x39;

static unsigned char SlDs\_3 @ 0x3a;

static unsigned char SlDs\_4 @ 0x3b;

Переменная для выдачи на ЦАП с его адресом

static unsigned int CapSlowo @ 0x14;

Переменные для рабочих областей стеков с их адресами

static unsigned long Stec @ 0x28;

static unsigned int Stec1 @ 0x28;

static unsigned int Stec2 @ 0x2a;

Переменные для сохранение переведенных скоростей с их адресами

static unsigned int StecVprd @ 0x30;

static unsigned int StecVmdd @ 0x32;

Переменные вводимых скорости с приборной скорости маски с их адресами

static unsigned int Vprg @ 0x20;

static unsigned int Vmdg @ 0x24;

Переменные вводимых скорости с максимальной скорости маски с их адресами

static unsigned int Vprd @ 0x22;

static unsigned int Vmdd @ 0x26;

Переменные рабочее слово SlRab расписаны в битовой резидентной памяти по битам и словам с их адресами

static unsigned long SlRab @ 0x2c;

static unsigned char SlRab\_4 @ 0x2c;

static unsigned char SlRab\_3 @ 0x2d;

static unsigned char SlRab\_2 @ 0x2e;

static unsigned char SlRab\_1 @ 0x2f;

static unsigned char Adress @ 0x2f - по этому адресу находится адрес выходного слова;

static bit unsigned char SlRab\_bit31 @ 0x67;

static bit unsigned char SlRab\_bit30 @ 0x66;

static bit unsigned char SlRab\_bit29 @ 0x65;

static bit unsigned char SlRab\_bit28 @ 0x64;

static bit unsigned char SlRab\_bit27 @ 0x63;

static bit unsigned char SlRab\_bit26 @ 0x62;

static bit unsigned char SlRab\_bit25 @ 0x61;

static bit unsigned char SlRab\_bit24 @ 0x60;

static bit unsigned char SlRab\_bit23 @ 0x6f;

static bit unsigned char SlRab\_bit22 @ 0x6e;

static bit unsigned char SlRab\_bit21 @ 0x6d;

static bit unsigned char SlRab\_bit20 @ 0x6c;

static bit unsigned char SlRab\_bit19 @ 0x6b;

static bit unsigned char SlRab\_bit18 @ 0x6a;

static bit unsigned char SlRab\_bit17 @ 0x69;

static bit unsigned char SlRab\_bit16 @ 0x68;

static bit unsigned char SlRab\_bit15 @ 0x77;

static bit unsigned char SlRab\_bit14 @ 0x76;

static bit unsigned char SlRab\_bit13 @ 0x75;

static bit unsigned char SlRab\_bit12 @ 0x74;

static bit unsigned char SlRab\_bit11 @ 0x73;

static bit unsigned char SlRab\_bit10 @ 0x72;

static bit unsigned char SlRab\_bit9 @ 0x71;

static bit unsigned char SlRab\_bit8 @ 0x70;

static bit unsigned char SlRab\_bit7 @ 0x7f;

static bit unsigned char SlRab\_bit6 @ 0x7e;

static bit unsigned char SlRab\_bit5 @ 0x7d;

static bit unsigned char SlRab\_bit4 @ 0x7c;

static bit unsigned char SlRab\_bit3 @ 0x7b;

static bit unsigned char SlRab\_bit2 @ 0x7a;

static bit unsigned char SlRab\_bit1 @ 0x79;

static bit unsigned char SlRab\_bit0 @ 0x78;

Переменные приборной скорости в коде Грея расписаны в битовой резидентной памяти по битам с их адресами

static bit unsigned char Vprg\_bit9 @ 0x01;

static bit unsigned char Vprg\_bit8 @ 0x00;

static bit unsigned char Vprg\_bit7 @ 0x0f;

static bit unsigned char Vprg\_bit6 @ 0x0e;

static bit unsigned char Vprg\_bit5 @ 0x0d;

static bit unsigned char Vprg\_bit4 @ 0x0c;

static bit unsigned char Vprg\_bit3 @ 0x0b;

static bit unsigned char Vprg\_bit2 @ 0x0a;

static bit unsigned char Vprg\_bit1 @ 0x09;

static bit unsigned char Vprg\_bit0 @ 0x08;

Переменные приборной скорости в двоичном коде расписаны в битовой резидентной памяти по битам с их адресами

static bit unsigned char Vprd\_bit9 @ 0x11;

static bit unsigned char Vprd\_bit8 @ 0x10;

static bit unsigned char Vprd\_bit7 @ 0x1f;

static bit unsigned char Vprd\_bit6 @ 0x1e;

static bit unsigned char Vprd\_bit5 @ 0x1d;

static bit unsigned char Vprd\_bit4 @ 0x1c;

static bit unsigned char Vprd\_bit3 @ 0x1b;

static bit unsigned char Vprd\_bit2 @ 0x1a;

static bit unsigned char Vprd\_bit1 @ 0x19;

static bit unsigned char Vprd\_bit0 @ 0x18;

Переменные максимальной скорости в коде Грея расписаны в битовой резидентной памяти по битам с их адресами

static bit unsigned char Vmdg\_bit9 @ 0x21;

static bit unsigned char Vmdg\_bit8 @ 0x20;

static bit unsigned char Vmdg\_bit7 @ 0x2f;

static bit unsigned char Vmdg\_bit6 @ 0x2e;

static bit unsigned char Vmdg\_bit5 @ 0x2d;

static bit unsigned char Vmdg\_bit4 @ 0x2c;

static bit unsigned char Vmdg\_bit3 @ 0x2b;

static bit unsigned char Vmdg\_bit2 @ 0x2a;

static bit unsigned char Vmdg\_bit1 @ 0x29;

static bit unsigned char Vmdg\_bit0 @ 0x28;

Переменные максимальной скорости в двоичном коде расписаны в битовой резидентной памяти по битам с их адресами

static bit unsigned char Vmdd\_bit9 @ 0x31;

static bit unsigned char Vmdd\_bit8 @ 0x30;

static bit unsigned char Vmdd\_bit7 @ 0x3f;

static bit unsigned char Vmdd\_bit6 @ 0x3e;

static bit unsigned char Vmdd\_bit5 @ 0x3d;

static bit unsigned char Vmdd\_bit4 @ 0x3c;

static bit unsigned char Vmdd\_bit3 @ 0x3b;

static bit unsigned char Vmdd\_bit2 @ 0x3a;

static bit unsigned char Vmdd\_bit1 @ 0x39;

static bit unsigned char Vmdd\_bit0 @ 0x38;

Формат слова 206 и 207.

Форматы 206, 207 слова состоят из 32 разрядов который показан в таблице 2.2 .

Таблица 2.2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27-11 | 10 | 9 | 8 | 7-0 |
| CRC | состояние | | Знак | данные | Точность | Идентификатор | | адрес |

Матрица состояния информации параметров.

|  |  |  |
| --- | --- | --- |
| Разряды в слове | | Наименование функции |
| 31 | 30 |
| 0 | 0 | Предупреждение об отказе |
| 0 | 1 | Нет вычисленных данных |
| 1 | 0 | Функциональный тест |
| 1 | 1 | Нормальная работа |

Алгоритм функционирования контроллера

Контроллер включает в себя следующие функции :

- Сброс сторожевого таймера;

- Очстка всех областей переменных;

- Подготовка вывода слова 206 с признаком "нет вычисленных данных";

- Подготовка вывода слова 207 с признаком "нет вычисленных данных";

- Заполнение сохраненного значения приборной скорости;

- Заполнение сохраненного значения максимальной скорости;

- Цикл проверки на отказ маски приборной скорости;

- Цикл проверки на отказ маски максимальной скорости;

- Проверка на наличие разовой команды тест контроля;

- Маштабирование приборной скорости для выдачи в ЦАП;

- Тестирование ПО;

- Формирование слова адрес 271 в соответствии с признаками об тестирования ПО;

- Формирование выходного слова адрес 206 в соответствии с РТМ, изм,3;

- Формирование выходного слова адрес 207 в соответствии с РТМ, изм,3;

- Сброс сторожевого таймера.

Выводы

В настоящем разделе выпускной работы были проведены

следующие работы:

- был проведен анализ структуры программы контроллера;

- был рассмотрен алгоритм функционирования контроллера.

Реализация

Описание функций основных модулей системы.

Основной файл системы является проэктный файл - SPEED.PRJ, в которой: указывается путь к файлам системы, установка адресов ПЗУ, ОЗУ, стека, векторов прерывания и параметров контроллера, список компилируемых файлов.

Основная программа

void main(void)

{

/\*---------- Запуск таймера на вывод --------------------------------------\*/

set\_vector(TIMER0,Timer);

Zapusk();

/\*-------- Сброс сторожевого таймера --------------------------------------\*/

TXD=0;

TXD=1;

/\*----------------- Обнуление данных --------------------------------------\*/

Clear();

/\*------------------------ Вывод на ЦАП -----------------------------------\*/

Machtab(StecVprd);

VivodCapSl();

/\*--------- Подготовка вывода 206 слова -----------------------------------\*/

SlRab=0;

Adress=0x61;

Priznak\_Usvic();/\* признак от УСВИЦ-250 \*/

Priznak\_Work(0,1);/\* признак нет вычисленных данных \*/

CalcChetSl();/\* Подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlVmd\_1=SlRab\_1;

SlVmd\_2=SlRab\_2;

SlVmd\_3=SlRab\_3;

SlVmd\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*--------- Подготовка вывода 207 слова -----------------------------------\*/

SlRab=0;

Adress=0xe1;

Priznak\_Usvic();/\* признак от УСВИЦ-250 \*/

Priznak\_Work(0,1);/\* признак нет вычисленных данных \*/

CalcChetSl();/\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlVpr\_1=SlRab\_1;

SlVpr\_2=SlRab\_2;

SlVpr\_3=SlRab\_3;

SlVpr\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*--------- Заполнение начальными данными Vpr и Vmd -----------------------\*/

a=3;

do {

/\* vvodvmd(); \*/

ImVxodMd(); /\* имитация входа Vmd \*/

Perevod\_Md(); /\* перевод из грея в двоичную \*/

StecVmdd=Vmdd; /\* Сохранение Vmd.\*/

/\* vvodvpr();\*/

ImVxodPr(); /\* имитация входа Vpr \*/

Perevod\_Pr(); /\* перевод из грея в двоичную \*/

StecVprd=Vprd; /\*Сохранение Vpr \*/

a--; }

while (a>0);

Nach: {

/\*-------- Сброс сторожевого таймера --------------------------------------\*/

TXD=0;

TXD=1;

/\*-------------------- Обновление данных Vpr и Vmd -----------------------\*/

/\* vvodvmd();\*/

ImVxodMd(); /\* имитация входа Vmd \*/

Perevod\_Md(); /\* перевод из грея в двоичную \*/

StecVmdd=Vmdd; /\*Сохранение Vм.д.\*/

/\* vvodvpr();\*/

ImVxodPr(); /\* имитация входа Vpr \*/

Perevod\_Pr(); /\* перевод из грея в двоичную \*/

StecVprd=Vprd; /\*Сохранение Vпр.д.\*/

/\*----------------- Цикл проверки на отказ Vmd ----------------------------\*/

Cikl\_Otkaz\_Vmd();

/\*----------------- Цикл проверки на отказ Vpr ----------------------------\*/

Cikl\_Otkaz\_Vpr();

/\*----------------- Тест на наличие РК "ТК" -------------------------------\*/

{

if(INT0==1) Test\_Kontrol();

}

while(INT0==1)

/\*----------------- Тест на наличие Vmd >= Vpr ----------------------------\*/

if(Vprd>=Vmdd) { RXD=0; PK\_Vpr=1; }

else { RXD=1; PK\_Vpr=0; }

/\*----------------- Маштабирование Vprd -----------------------------------\*/

Machtab(StecVprd);

VivodCapSl();

/\*------------- копируем в поле 271 функциональный тест ------------------\*/

SlRab=0;

Priznak\_Work(1,0);/\* признак функциональный тест \*/

CalcChetSl(); /\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlDs\_1=SlRab\_1;

SlDs\_2=SlRab\_2;

SlDs\_3=SlRab\_3;

SlDs\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*------------- копируем в поле 206 функциональный тест ------------------\*/

SlRab=0;

Priznak\_Work(1,0);/\* признак функциональный тест \*/

CalcChetSl(); /\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlVpr\_1=SlRab\_1;

SlVpr\_2=SlRab\_2;

SlVpr\_3=SlRab\_3;

SlVpr\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*------------- копируем в поле 207 функциональный тест ------------------\*/

SlRab=0;

Priznak\_Work(1,0);/\* признак функциональный тест \*/

CalcChetSl(); /\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlVmd\_1=SlRab\_1;

SlVmd\_2=SlRab\_2;

SlVmd\_3=SlRab\_3;

SlVmd\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*----------------- Тест Контроль программы -------------------------------\*/

TestControlPO();

/\*-------- Формирование дискретного 271 слова ----------------------------\*/

SlRab=0;

Adress=0x9d;/\* адрес слова 271 \*/

Priznak\_Test();/\* растановка признаков тестирования в слове \*/

Priznak\_Usvic();/\* признак от УСВИЦ-250 \*/

CalcChetSl();/\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlDs\_1=SlRab\_1;

SlDs\_2=SlRab\_2;

SlDs\_3=SlRab\_3;

SlDs\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*------ Формирование 206 слова в соответствии с РТМ, изм,3 для Vpr ------\*/

SlRab=0;

FormPTM(StecVprd);

Adress=0x61;/\* адрес слова 206 \*/

Priznak\_Usvic();/\* признак от УСВИЦ-250 \*/

Priznak\_Work(1,1);/\* признак нормальной работы \*/

If\_Speed(Vprd);/\* проверка на < 50км \*/

if ((Vpr\_Ok==1)||(PO\_Ok==1)) Priznak\_Work(0,0);/\* отказ \*/

CalcChetSl();/\* подсчет четности \*/

/\*------------- Копирование параметров в область вывода -------------------\*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlVpr\_1=SlRab\_1;

SlVpr\_2=SlRab\_2;

SlVpr\_3=SlRab\_3;

SlVpr\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*----- Формирование 207 слова в соответствии с РТМ, изм,3 для Vmd -------\*/

SlRab=0;

FormPTM(StecVmdd);

Adress=0xe1;/\* адрес слова 207 \*/

Priznak\_Usvic();/\* признак от УСВИЦ-250 \*/

Priznak\_Work(1,1);/\* признак нормальной работы \*/

If\_Speed(Vmdd); /\* проверка на < 50км \*/

if ((Vmd\_Ok==1)||(PO\_Ok==1)) Priznak\_Work(0,0);/\* отказ \*/

CalcChetSl();/\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlVmd\_1=SlRab\_1;

SlVmd\_2=SlRab\_2;

SlVmd\_3=SlRab\_3;

SlVmd\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

SlRab=0;

/\*-------------------- G O T O -> N A C H ---------------------------\*/

}

goto Nach;

}

Процедура установки приоритетов прерываний.

void Zapusk(void){

/\*------------- Запуск таймера --------------------------\*/

asm(" mov TCON,#1 ");

asm(" anl IP,#0 ");

asm(" orl IP,#2 ");

asm(" mov TMOD,#01 ");

asm(" mov a,#0eah "); /\* число для загрузки таймера \*/

asm(" mov TH0,a "); /\* 65536-20\*36-685+67, где старшая часть \*/

asm(" mov a,#0d3h "); /\* равна eah, а младшая - d3h \*/

asm(" mov TL0,a ");

asm(" setb TCON.4 ");

asm(" anl IE,#0 ");

asm(" orl IE,#82h ");

}

Назначение: запуск таймера с установки приоритетов прерываний.

Вход: нет.

Выход: установленный счетчик таймера.

Процедура обнуления переменных.

void Clear(void){

StecVmdd=StecVprd=0;

Vprd=Vmdd=Vprg=Vmdg=0;

SlRab=SlVmd=SlVpr=SlDs=CapSlowo=0;

SlRab=0;

}

Назначение: обнуление областей переменных в начале работы программы от мусора при включения питания.

Вход: нет.

Выход: нет.

Процедура считывания с кодов грея с максимальной допустимой маски.

\_vvodvmd:

push 0e0h

push 1

push 2

push 3

clr tcon.4

orl p0,#1 ;запрещение работы D1

anl p0,#1 ;запрещение работы D1

orl p1,#14h ;запрещение D2 и разрешение D4

anl p1,#14h ;запрещение D2 и разрешение D4

;считываем для Vmd

call Read\_P\_1\_5

mov bitvmd0,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvmd1,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvmd2,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvmd3,c

и т.д.

Назначение: считывание кодов с максимальной маски

Вход: нет.

Выход: заполнены биты максимальной маски.

Процедура считывания с кодов грея с приборной маски.

\_vvodvpr:

push 0e0h

push 1

push 2

push 3

clr tcon.4

orl p0,#1 ;запрещение работы D1

anl p0,#1 ;запрещение работы D1

orl p1,#1 ;запрещение D4 и разрешение D2

anl p1,#1 ;запрещение D4 и разрешение D2

;считываем для Vpr

call Read\_P\_1\_5

mov bitvpr0,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvpr1,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvpr2,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvpr3,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvpr4,c

и т.д.

Назначение: считывание кодов с приборной маски

Вход: нет.

Выход: заполнены биты приборной маски.

Процедура вывода в магистраль 32х-разрядного слова с частотой 12.5 КГц.

Vivod206:

PUSH 0D0H

PUSH 0E0H

PUSH 0

PUSH 1

PUSH 2

PUSH 3

CLR PSW.3

CLR PSW.4

ORL P0,#0C0H

MOV R3,#4

MOV R0,#Sl206

Bait206:

MOV A,@R0

MOV R2,#7

Bit206:

RRC A

ORL P0,#0ffh

MOV Bit6V,C

MOV Bit7V,C

CPL Bit7V

MOV R1,#15

Pauza206\_0: DJNZ R1,Pauza206\_0

NOP

NOP

MOV P0,BaitV

MOV R1,#18

Pauza206\_1: DJNZ R1,Pauza206\_1

DJNZ R2,Bit206

RRC A

ORL P0,#0ffh

MOV Bit6V,C

MOV Bit7V,C

CPL Bit7V

MOV R1,#16

Pauza206\_2: DJNZ R1,Pauza206\_2

MOV P0,BaitV

MOV R1,#15

Pauza206\_3: DJNZ R1,Pauza206\_3

INC R0

DJNZ R3,Bait206

NOP

NOP

NOP

ORL P0,#0C0H

POP 3

POP 2

POP 1

POP 0

POP 0E0H

POP 0D0H

RET

Назначение: выдача побитно в магистраль 32х-разрядного слова с частотой 12.5 КГц.

Вход: номер выводимого слова (номера 0 соответствует выводимому слову 206, и соответственно 2=207,3=271).

Выход: нет.

Процедура аналогового вывода приборной скорости на ЦАП.

\_VivodCapSl:

push 0e0h

push 1

push 2

push 3

clr tcon.4

mov a,CapSlowo0

mov p0,a

mov a,CapSlowo1

mov p2,a

setb p0.3

nop ;задержка чтобы ЦАП успел принять данные

nop

nop

nop

nop

clr p0.3

setb tcon.4

pop 3

pop 2

pop 1

pop 0e0h

ret

Назначение: аналоговый вывод на ЦАП приборной скорости .

Вход: приборная скорость.

Выход: на шине данных ЦАП высота с приведенным коэффициентом.

Процедура установки признака от УСВИЦ-250.

void Priznak\_Usvic(viod){

SlRab\_bit8=0;/\* Признак \*/

SlRab\_bit9=1;/\* от УСВИЦ-250 \*/

}

Назначение: в биты 8, 9 выходного слова ставится 0 и 1 соответственно для обозначения, что слово исходит от УСВИЦ-250.

Вход: нет.

Выход: SlRab\_bit8=0; SlRab\_bit9=1;

Процедура установки признаков работоспособности УСВИЦ-250.

void Priznak\_Work(bit unsigned char a, bit unsigned char b){

SlRab\_bit30=a;

SlRab\_bit29=b;

}

Назначение: в 30 и 29 биты ставятся признаки работоспособность УСВИЦ-250.

Вход: bit unsigned char a, bit unsigned char b.

Выход: SlRab\_bit30=a; SlRab\_bit29=b;

Процедура наземного контроля.

void Test\_Kontrol(void){

int V=0;

vvodvmd();

Perevod\_Md(); /\* перевод из грея в двоичную \*/

vvodvpr();

Perevod\_Pr(); /\* перевод из грея в двоичную \*/

/\*----------------- Сохраняем тестовое значение ---------------------------\*/

V=Vprd;

/\*----------------- Прабавляем тестовое значение --------------------------\*/

Vprd+=166\*4;

/\*----------------- Тест на наличие Vmd >= Vpr ----------------------------\*/

if(Vprd>=Vmdd) { RXD=0; PK\_Vpr=1; }

else { RXD=1; PK\_Vpr=0; }

/\*----------------- Отнимаем тестовое значение ----------------------------\*/

Vprd-=166\*4;

/\*----------------- Востанавливаем тестовое значение ----------------------\*/

Vprd=V;

/\*----------------- Маштабирование Vprd -----------------------------------\*/

Machtab(StecVprd);

VivodCapSl();

/\*------------- копируем в поле 271 функциональный тест ------------------\*/

SlRab=0;

Priznak\_Work(1,0);/\* признак функциональный тест \*/

CalcChetSl(); /\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlDs\_1=SlRab\_1;

SlDs\_2=SlRab\_2;

SlDs\_3=SlRab\_3;

SlDs\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*------------- копируем в поле 206 функциональный тест ------------------\*/

SlRab=0;

Priznak\_Work(1,0);/\* признак функциональный тест \*/

CalcChetSl(); /\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlVpr\_1=SlRab\_1;

SlVpr\_2=SlRab\_2;

SlVpr\_3=SlRab\_3;

SlVpr\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*------------- копируем в поле 207 функциональный тест ------------------\*/

SlRab=0;

Priznak\_Work(1,0);/\* признак функциональный тест \*/

CalcChetSl(); /\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlVmd\_1=SlRab\_1;

SlVmd\_2=SlRab\_2;

SlVmd\_3=SlRab\_3;

SlVmd\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*----------------- Тест Контроль программы -------------------------------\*/

TestControlPO();

/\*-------- Формирование дискретного 271 слова ----------------------------\*/

SlRab=0;

Adress=0x9d;/\* адрес слова 271 \*/

Priznak\_Test();/\* растановка признаков тестирования в слове \*/

Priznak\_Usvic();/\* признак от УСВИЦ-250 \*/

CalcChetSl();/\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlDs\_1=SlRab\_1;

SlDs\_2=SlRab\_2;

SlDs\_3=SlRab\_3;

SlDs\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*------ Формирование 206 слова в соответствии с РТМ, изм,3 для Vpr ------\*/

SlRab=0;

FormPTM(StecVprd);

Adress=0x61;/\* адрес слова 206 \*/

Priznak\_Usvic();/\* признак от УСВИЦ-250 \*/

Priznak\_Work(1,1);/\* признак нормальной работы \*/

If\_Speed(Vprd);/\* проверка на < 50км \*/

if ((Vpr\_Ok==1)||(PO\_Ok==1)) Priznak\_Work(0,0);/\* отказе \*/

CalcChetSl();/\* подсчет четности \*/

/\*------------- Копирование параметров в область вывода -------------------\*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlVpr\_1=SlRab\_1;

SlVpr\_2=SlRab\_2;

SlVpr\_3=SlRab\_3;

SlVpr\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*----- Формирование 207 слова в соответствии с РТМ, изм,3 для Vmd -------\*/

SlRab=0;

FormPTM(StecVmdd);

Adress=0xe1;/\* адрес слова 207 \*/

Priznak\_Usvic();/\* признак от УСВИЦ-250 \*/

Priznak\_Work(1,1);/\* признак нормальной работы \*/

If\_Speed(Vmdd); /\* проверка на < 50км \*/

if ((Vmd\_Ok==1)||(PO\_Ok==1)) Priznak\_Work(0,0);/\* отказе \*/

CalcChetSl();/\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlVmd\_1=SlRab\_1;

SlVmd\_2=SlRab\_2;

SlVmd\_3=SlRab\_3;

SlVmd\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

SlRab=0;

/\*-------- Сброс сторожевого таймера --------------------------------------\*/

TXD=0;

TXD=1;

}

Назначение: Пилотом на земле включает тумблер тест контроля состояния прибора. Максимальная скорость индексом устанавливается на максимальную скорость (250) и к приборной скорости прибавляем тестовое значение скорости. При не совпадении скоростей выводим на индикацию световую и звуковую, сигнализируя о неисправности прибора. После чего проходит весь цикл программы функционирования прибора.

Вход: тестовое значение.

Выход: индикация.

Процедура подсчета и записи бита четности 32х-разрядного рабочего слова.

void CalcChetSl(void){

SlRab\_bit31=1;

SlRab\_bit31=SlRab\_bit31⊕SlRab\_bit0⊕SlRab\_bit1⊕SlRab\_bit2;

SlRab\_bit31=SlRab\_bit31⊕SlRab\_bit3⊕SlRab\_bit4⊕SlRab\_bit5;

SlRab\_bit31=SlRab\_bit31⊕SlRab\_bit6⊕SlRab\_bit7⊕SlRab\_bit8;

SlRab\_bit31=SlRab\_bit31⊕SlRab\_bit9⊕SlRab\_bit10⊕SlRab\_bit11;

SlRab\_bit31=SlRab\_bit31⊕SlRab\_bit12⊕SlRab\_bit13⊕SlRab\_bit14;

SlRab\_bit31=SlRab\_bit31⊕SlRab\_bit15⊕SlRab\_bit16⊕SlRab\_bit17;

SlRab\_bit31=SlRab\_bit31⊕SlRab\_bit18⊕SlRab\_bit19⊕SlRab\_bit20;

SlRab\_bit31=SlRab\_bit31⊕SlRab\_bit21⊕SlRab\_bit22⊕SlRab\_bit23;

SlRab\_bit31=SlRab\_bit31⊕SlRab\_bit24⊕SlRab\_bit25⊕SlRab\_bit26;

SlRab\_bit31=SlRab\_bit31⊕SlRab\_bit27⊕SlRab\_bit28⊕SlRab\_bit29;

SlRab\_bit31=SlRab\_bit31⊕SlRab\_bit30;

}

Назначение: подсчет четности слова и выставления знака четности в 31 разряде подсчет четности производиться по модулю 2.

Вход: нет.

Выход: SlRab\_bit31- знак четности.

Процедура перевода из кодов грея в двоичный код максимально допустимую скорость.

void Perevod\_Md(void){

Vmdd=0;

Vmdd\_bit9=Vmdg\_bit9;

Vmdd\_bit8=Vmdg\_bit8⊕Vmdd\_bit9;

Vmdd\_bit7=Vmdg\_bit7⊕Vmdd\_bit8;

Vmdd\_bit6=Vmdg\_bit6⊕Vmdd\_bit7;

Vmdd\_bit5=Vmdg\_bit5⊕Vmdd\_bit6;

Vmdd\_bit4=Vmdg\_bit4⊕Vmdd\_bit5;

Vmdd\_bit3=Vmdg\_bit3⊕Vmdd\_bit4;

Vmdd\_bit2=Vmdg\_bit2⊕Vmdd\_bit3;

Vmdd\_bit1=Vmdg\_bit1⊕Vmdd\_bit2;

Vmdd\_bit0=Vmdg\_bit0⊕Vmdd\_bit1;

}

Назначение: перевод кодов грея в двоичный код.

Вход: Коды грея.

Выход: двоичный код.

Процедура перевода из кодов грея в двоичный код приборную скорость.

void Perevod\_Pr(void){

Vprd=0;

Vprd\_bit9=Vprg\_bit9;

Vprd\_bit8=Vprg\_bit8⊕Vprd\_bit9;

Vprd\_bit7=Vprg\_bit7⊕Vprd\_bit8;

Vprd\_bit6=Vprg\_bit6⊕Vprd\_bit7;

Vprd\_bit5=Vprg\_bit5⊕Vprd\_bit6;

Vprd\_bit4=Vprg\_bit4⊕Vprd\_bit5;

Vprd\_bit3=Vprg\_bit3⊕Vprd\_bit4;

Vprd\_bit2=Vprg\_bit2⊕Vprd\_bit3;

Vprd\_bit1=Vprg\_bit1⊕Vprd\_bit2;

Vprd\_bit0=Vprg\_bit0⊕Vprd\_bit1;

}

Назначение: перевод кодов грея в двоичный код.

Вход: Коды грея.

Выход: двоичный код.

Процедура проверки на отказ маски максимальной скорости.

void Cikl\_Otkaz\_Vmd(void){

int c=3,St;

Vmd\_Ok=0;

T1=0;

do

{

vvodvmd();

Perevod\_Md();/\* перевод из грея в двоичную \*/

St=StecVmdd;

if ((St+test\_speed>Vmdd)&&(St-test\_speed<Vmdd))

{

StecVmdd=Vmdd;

break;

}

if (c==0)

{

Vmd\_Ok=1;

T1=1;

break;

}

c--;

}

while(c>(-1));

}

Назначение: проверка маски максимальной скорости путем проверки предыдущих значений с полученными значениями и допуск не должен превышать тестовой единицы. Если превышение возникло, то проверка проходит еще 3 раза, но если после этого превышение не исчезло. Выдается световая сигнализация на борт панель пилоту и записывается в ячейку служебного слова (271).

Вход: нет.

Выход: нет.

Процедура проверки на отказ маски приборной скорости.

void Cikl\_Otkaz\_Vpr(void){

int b=3,St;

Vpr\_Ok=0;

RXD=0;

do

{

vvodvpr();

Perevod\_Pr();/\* перевод из грея в двоичную \*/

St=StecVprd;

if ((St+test\_speed>Vprd)&&(St-test\_speed<Vprd))

{

StecVprd=Vprd;

break;

}

if (b==0)

{

Vpr\_Ok=1;

RXD=1;

break;

}

b--;

}

while(b>(-1));

}

Назначение: проверка маски приборной скорости путем проверки предыдущих значений с полученными значениями и допуск не должен превышать тестовой единицы. Если превышение возникло, то проверка проходит еще 3 раза, но если после этого превышение не исчезло. Выдается световая сигнализация на борт панель пилоту и записывается в ячейку служебного слова (271).

Вход: нет.

Выход: нет.

Процедура масштабирование для вывода на ЦАП.

void Machtab(unsigned int V1){

double rab1;

rab1=(double)V1;

rab1\*=0.63;

CapSlowo=0;

CapSlowo=(unsigned int)rab1;

CapSlowo0<<=6;

}

Назначение: масштабирование необходимо потому, что по ТУ выдача аналогового сигнала должна соответветствовать всем единицам на входе ЦАП, на аналоговом выходе 0.63 В.

Вход: приборная скорость.

Выход: нет.

Процедура формирование 32х-разрядного слова с соответствием с РТМ

void FormPTM(unsigned int V1){

double rab2;

SlRab=0;

rab2=(double)V1;

rab2/=0.11575;

SlRab=(unsigned long)rab2;

rab2-=(double)SlRab;

SlRab<<=12;

}

Назначение: для формирование 32х-разрядного слова с соответствием с РТМ необходима данные поместить с 11 по 27 разряд с ценой младшего разряда.

Вход: приборная скорость.

Выход: данные помещены в слове на свое место.

Процедура тест контроля программного обеспечения

void TestControlPO(void){

int Vprd1=0,Vmdd1=0,Vprg1=0,Vmdg1=0;

PO\_Ok=Vmd\_Ok=Vpr\_Ok=0;

Vprg1=Vprg;

Vmdg1=Vmdg;

Vprd1=Vprd;

Vmdd1=Vmdd;

Vmdg=Vprg=speed;

Perevod\_Md();

Perevod\_Pr();

if (Vmdd!=200) { PO\_Ok=1; Vmd\_Ok=1; }

if (Vprd!=200) { PO\_Ok=1; Vpr\_Ok=1; }

/\*----------------------------- 206 --------------------------------------\*/

SlRab=0;

FormPTM(Vprd);

Adress=0x61;

Priznak\_Usvic();

Priznak\_Work(1,0);

CalcChetSl();

if(SlRab!=0) { PO\_Ok=1; Vpr\_Ok=1; }

/\*----------------------------- 207 --------------------------------------\*/

SlRab=0;

FormPTM(Vmdd);

Adress=0xe1;

Priznak\_Usvic();

Priznak\_Work(1,0);

CalcChetSl();

if(SlRab!=0) { PO\_Ok=1; Vmd\_Ok=1; }

SlRab=0;

Vprg=Vprg1;

Vmdg=Vmdg1;

Vprd=Vprd1;

Vmdd=Vmdd1;

}

Назначение: проверка программы. Это осуществляется путем записи тестовых значений в максимальную и приборную скорость кодов грея. После чего производится проверка на правильности перевода из кодов грея в двоичный код, формирования слова и расстановки признаков.

Вход: тестовое значение кодов грея.

Выход: нет.

Процедура проверки скорости, на меньше 50км.

void If\_Speed(unsigned int V){

if(V<200) Priznak\_Work(0,1);

}

Назначение: при полеченной скорости меньше чем 50 км необходимо по ТУ выдавать в выходном слове признак нет вычисленных данных.

Вход: приборная скорость.

Выход: расстановка признаков нет вычисленных данных.

Описание аппаратных функций

Основные функции системы

Программа начинает выполнение при подачи питания на контроллер, независимо от других контроллеров. Программа сначало сбрасывает сторожевой таймер предназначенный от зацикливаниии программы, который необходимо сбрасывать в конце программы. После чего очищает все переменные и подготавливает слова с указанием в них о том, что нет вычисленных данных, т.к. процедура вывода слов работает независимо от основной программы по прерыванию от таймера. Далее программы заполняет ячейки скорости. для последующего использавании и проверки на выход из строя сведодиодов или фотодиодов. После чего выполняет ввод с масок данных, перевод их, проверку на отказ маскок предельной и максимальной скорости, проверка на наличии тест контроля, маштабирования л\для вывода на ЦАП, выполняем тест контроль программного обеспечения с занесением результатов в служебное слово и копирование полученных данных в слова.

Технология инсталляции программы

Данный программный продукт поставляется на предприятие-заказчик для конечного использования в борт компьютерах вертолетах или самолетах с максимальной скоростью не более 250км.

При использовании данной версии программы, программа записывается во внутреннью память процессора, т.к. процессор имеет встроенную постоянную память.

Инструкция пользователю

Данная программа разрабатывалсь на ПЭВМ класса PENTIUM c процессором фирмы INTEL с тактовой частотой 100 МГц и с использование внутрисхемного эмулятора фирмы Phyton PICE-51. На этой же машине, но с эксперементальным образцом контроллера проводились испытания. Операционная система Windows'98.

Оценка качества проекта

Качество настоящего проекта будем оценивать по следующим критериям:- полезность (исходная и общая);- удобство эксплуатации;- эффективность;- модифицируемость;- машинонезависимость;

-завершенность;

- согласованность;

- рациональность;

- доступность;

- коммуникативность;

- информативность;

- расширяемость.

Попробуем проанализировать каждый из вышеперечисленных критериев, и дать по нему оценку контроллера. Полезность:

Продукт проектирования обладает свойством ПОЛЕЗНОСТИ, если он отвечает поставленным целям. Это свойство имеет две стороны:

1) исходная полезность - насколько просто, надежно, эффективно можно использовать основной результат обработки;

2) общая полезность - возможность продолжать использование продукта проектирования при изменении условий его применения.

Исходя из данного определения полезности можно сказать, что предлагаемый проект обладает исходной, но не общей полезностью, так как требования простоты, надежности и эффективности закладывались в его основу как одни из главных, но в случае изменения условий его применения эффективность его использования снизится.

Удобство эксплуатации

Продукт проектирования УДОБЕН в ЭКСПЛУАТАЦИИ, если рассматривается возможность его обновления в соответствии с новыми требованиями.

На основе вышесказанного можно определить контроллер удобен в эксплуатации, покрайней мере, в течение 1 года после внедрения.

Эффективность

Проект проектирования обладает свойством эффективности, если он выполняет требуемые функции без лишних затрат ресурсов.

Так как результат проектирования установлен и удовлетворительно функционирует на процессоре INTEL 8051 широко используемый в самолетостроении в наше время, то можно определить проект эффективным.

Модифицируемость

Продукт проектирования необладает свойством МОДИФИЦИРУЕМОСТИ, т.к. он имеет закрытую структуру, не позволяющую легко вносить требуемые изменения.

Так как под модифицируемостью, видимо подразумевается степень открытости архитектуры, то проект не обладает этим свойством.

Машинонезависимость

Продукт проектирования обладает свойством МАШИНОНЕЗАВИСИМОСТИ, если входящие в него функциональные компоненты могут быть реализованы на иных технических средствах. Машинонезависимость проекта целиком и полностью зависит от машинонезависимости контроллера, версии которой работают на довольно многих типах процессоров. Таким образом, свойство машинонезависимости НЕПРИСУЩЕ.

***Завершенность***

Продукт проектирования обладает свойством ЗАВЕРШЕННОСТИ, если в нем присутствуют все необходимые компоненты, каждый из которых разработан всесторонне.

***Согласованность***

Продукт проектирования обладает свойством ВНУТРЕННЕЙ СОГЛАСОВАННОСТИ, если он всюду содержит единую нотацию, терминологию и символику. Он также обладает свойством ВНЕШНЕЙ СОГЛАСОВАННОСТИ, если можно проследить его соответствие конкретным требованиям.

Рациональность

Продукт проектирования обладает свойством РАЦИОНАЛЬНОСТИ, если он использует все возможности (языка программирования, численных методов, технических и других средств) для минимизации расхода ресурсов.

На основе анализа исходных текстов можно сделать вывод о том что он не обладает указанным свойством.

Доступность

Продукт проектирования обладает свойством ДОСТУПНОСТИ, если он не допускает селективное использование его компонентов. На основании информации из раздела 2 можно сказать, что проект НЕ ДОСТУПЕН.

Коммуникативность

Продукт проектирования обладает свойством КОММУНИКАТИВНОСТИ, если он дает возможность легко описывать входные данные и выдает информацию, форма и содержание которой просты для понимания и несут полезные сведения.

Расширяемость

Продукт проектирования не обладает свойством РАСШИРЯЕМОСТИ, если он позволяет при необходимости увеличить объем обрабатываемых данных или расширить функции отдельных модулей.

Выводы

В данном разделе выпускной работы были проведены следующие работы:

* описание фнкций основных модулей программы;
* описана технология инсталяции программы;
* дана инструкция пользователю;
* проведена оценка качества проекта;

Организационно-техническая часть

Расчет затрат на разработку программы

Затраты на разработку программного продукта рассчитываются по формуле:

Zрпп = Zзп + Zзпд + Остр + Оам + Zээ + Rкосв , (1)

где

Zзп - затраты на заработную плату программисту, руб.;

Zзпд - затраты на дополнительную заработную плату, руб.;

Остр - отчисление на социальное страхование, руб.;

Оам - амортизационные отчисления, руб.;

Zээ - затраты на электроэнергию во время разработки программы, руб.;

Rкосв - косвенные расходы, руб.

Разработка программы производилась инженером-программистом, с заработанной платой 600 рублей в месяц. На разработку программы ушло 5 месяца, поэтому затраты на зарплату составляют:

Zзп = 600 \* 5 = 3000 (руб.)

Дополнительная заработная плата составляет 30% от основной заработной платы, тогда затраты на дополнительную заработную плату:

Zзпд = Zзп \* 0.3 = 3000 \* 0.3 = 900 (руб.)

Отчисления на социальное страхование составляют 37% от общей заработной платы, то есть

Остр = ( Zзп + Zзпд ) \* 0.37 = ( 3000 + 900 ) \* 0.37 = 1443 (руб.)

Стоимость ПЭВМ, используемой для разработки программы составляет 15 500 руб. Срок службы персонального компьютера составляет 8 лет. Тогда на данный момент амортизационные отчисления за месяц составляют:

Оам1 = 15500 / ( 8 \* 12 ) = 161.45 (руб.)

За 5 месяца:

Оам = 161.45 \* 5 = 807.25 (руб.)

На отладку и тестирование программы с использованием ПЭВМ затрачено 120 рабочих дней, из них, в среднем каждый день по 5 часа. Стоимость электроэнергии 0.14 руб. за киловатт. Мощность, потребляемая ПЭВМ 300 Вт. Посчитаем затраты на электроэнергию во время разработки программы, руб.;

Zээ = ( 120 \* 5 ) \* 0.14 \* 0.300 = 25.2 (руб.)

Косвенные расходы составляют 110% от основной заработанной платы программиста:

Rкосв = 3000 \* 1.1 = 3300 (руб.)

Таким образом, подставляя вычисленные значения затрат в формулу (1), получим стоимость затрат на разработку программы:

Zрпп = 3000 + 900 + 1443 + 807.25 + 25.2 +3300= 9475,45 (руб.)

Расчет экономической эффективности от внедрения программного продукта

Рассчитаем экономическую эффективность от внедрения разработанного программного продукта на примере сравнения годовых затрат на содержание отдела рефератированния (5 человек) и содержание ПЭВМ, обслуживающего персонала (1 человек) и консультанта по работе с технической информацией, заменяющих вышеназванный отдел.

Экономическую эффективность рассчитаем по формуле:

Eэ = Zдв - Zпв, (2)

где

Zдв - затраты до внедрения;

Zпв - затраты после внедрения.

Затраты до внедрения

Затраты до внедрения программного продукта рассчитаем по формуле:Zдв = Zзп + Zзпд + Oстр, (3)гдеZзп - затраты на заработную плату работнику, руб.;Zзпд - затраты на дополнительную заработную плату, руб.;

Остр - отчисление на социальное страхование, руб.;

Средняя заработанная плата работников отдела технической информации составляет 800 руб., тогда:

Zзп = 800 \* 5\*12 = 48 000 (руб.)

Дополнительная заработная плата составляет 30% от основной заработной платы, тогда затраты на дополнительную заработную плату:

Zзпд = Zзп \* 0.3 = 48 000 \* 0.3 = 14400 (руб.)

Отчисления на социальное страхование составляют 37% от общей заработной платы, тогда:

Oстр =(Zзп + Zзпд)\* 0.37 , то есть

Остр = (48000 + 14400) \* 0.37 = 23088 (руб.),

Таким образом, подставляя вычисленные значения затрат в формулу (3), получим стоимость затрат до внедрения программы:

Zдв = 48 000 + 14400 + 23088 = 85488 (руб.).

Затраты после внедрения

Затраты после внедрения программного продукта рассчитаем по формуле:

Zпв = Zзп + Zзпд + Oстр + Zээ + Oам + Sпо + Zвпп, (4)

где

Zзп - затраты на заработную плату программисту, руб.;

Zзпд - затраты на дополнительную заработную плату, руб.;

Остр - отчисление на социальное страхование, руб.;

Zээ - затраты на электроэнергию для ПЭВМ, руб.;Оам - амортизационные отчисления, руб.;

Zвпп - затраты на внедрение программного продукта, руб.

Заработанная плата программисту равна 600 тыс. руб., тогда:

Zзп = 600 \* 12 + 400 \* 12 = 12 000 (руб.).

Дополнительная заработная плата составляет 30% от основной заработной платы, тогда затраты на дополнительную заработную плату:

Zзпд = Zзп \* 0.3 = 12 000 \* 0.3 = 3600 (руб.).

Отчисления на социальное страхование составляют 37% от общей заработной платы, тогда:

Oстр = ( Zзп + Zзпд ) \* 0.37 , то есть

Oстр = (12000+3600)\*0.37= 5772 (руб.).

ПЭВМ на рабочем месте работает 8 часов в день. Количество рабочих дней в году 264. Стоимость электроэнергии 0.14 руб. за киловатт. Мощность, потребляемая ПЭВМ 300 Вт. Посчитаем годовые затраты на электроэнергию, потребляемую ПЭВМ, руб.;

Zээ = (8 \* 264) \* 0.14 \* 0.300 = 88 (руб.).

Стоимость ПЭВМ, используемой при эксплуатации разработанной программы составляет 15 500 руб. Срок службы персонального компьютера составляет 8 лет. Тогда амортизационные отчисления за год составляют:

Oам = 15 500 / 8 = 1937.5 (руб.).

Стоимость программного обеспечения равняется себестоимости программного продукта, посчитанной в предыдущем пункте данной главы. Следовательно:

Sпо = 9475.45(руб.).

В затраты на внедрение программного продукта входят затраты на изготовление сопроводительной документации, на заработную плату службы сопровождения программного продукта, затраты на обучение пользователя работе с программой, на установку программного продукта, на адаптацию программного продукта к условиям пользователя и т.д. Так как размер данных затрат зависит от конкретных условий, то стоимость затрат на этапе внедрения можно оценить только качественно, и можно сказать что она будет соизмерима со стоимостью затрат на этапе разработки. Затраты на внедрение программного продукта составляют 110% от затрат на этапе разработки программного обеспечения:

Zвпп = 9475.45 \* 1.1 = 10423 (руб.)

Таким образом, подставляя вычисленные значения затрат в формулу (4), получим стоимость затрат после внедрения программы:

Zпв = 12000 + 3600 + 5772 + 88 + 1937.5 + 9475.45 + 10423 = 43295,95 (руб.)

Подставляя рассчитанные затраты до и после внедрения программы в формулу (2), получим значение экономической эффективности от внедрения разработанного программного продукта:

Eэ = Zдв - Zпв = 85488 - 43295,95 = 42192.05 (руб.)

Выводы

Полученные результаты говорят об эффективности внедрения разработанного программного продукта в процессы, требующие обработки большого объема принятия и выдача с всевозможных датчиков. Что существенно уменьшает затраты на разработку схем на вентильной логики.

Приложение 1. Основная программа.

#include <8051.h>

#include <intrpt.h>

#include <float.h>

/\* 206 => 0x61 207 => 0xE1 271=>9D \*/

const int speed=172; /\* это должно быть 200 \*/

const int test\_speed=2; /\* 0.5==2 \*/

static int a @ 0x18;

/\* static int flag @ 0x18;\*/

/\* static int flag1 @ 0x1a;\*/

static bit unsigned char Usvic\_Ok @ 0x50;

static bit unsigned char PK\_Vpr @ 0x51;

static bit unsigned char Vpr\_Ok @ 0x52;

static bit unsigned char Vmd\_Ok @ 0x53;

static bit unsigned char PO\_Ok @ 0x54;

/\*--- номер выводимого параметра 0=Sl206,1=Sl207,2=Sl271 -\*/

static unsigned char NomParam @ 0x1c;

static unsigned char BaitV @ 0x2a;

/\*------------Выходное слово SlVmd 206 ------------------\*/

static unsigned long SlVmd @ 0x10;

static unsigned char SlVmd\_1 @ 0x10;

static unsigned char SlVmd\_2 @ 0x11;

static unsigned char SlVmd\_3 @ 0x12;

static unsigned char SlVmd\_4 @ 0x13;

/\*------------Выходное слово SlVpr 207 -------------------\*/

static unsigned long SlVpr @ 0x3c;

static unsigned char SlVpr\_1 @ 0x3c;

static unsigned char SlVpr\_2 @ 0x3d;

static unsigned char SlVpr\_3 @ 0x3e;

static unsigned char SlVpr\_4 @ 0x3f;

/\*------------Выходное слово SlDs 271 --------------------\*/

static unsigned long SlDs @ 0x38;

static unsigned char SlDs\_1 @ 0x38;

static unsigned char SlDs\_2 @ 0x39;

static unsigned char SlDs\_3 @ 0x3a;

static unsigned char SlDs\_4 @ 0x3b;

/\*------------Выходное слово для ЦАП CapSlowo -------------\*/

static unsigned int CapSlowo @ 0x28;

static unsigned char CapSlowo0 @ 0x28;

static unsigned char CapSlowo1 @ 0x29;

/\*------- Рабочая область стеков ------------------\*/

/\* static unsigned long Stec @ 0x28;\*/

static bit unsigned char Test\_Pr @ 0x40;

/\*--------- Сохранение переведенных скоростей ----------------\*/

static unsigned int StecVprd @ 0x30;

static unsigned int StecVmdd @ 0x32;

/\*--------- Int вводимых скорости ----------------------------\*/

static unsigned int Vprg @ 0x20;

static unsigned int Vmdg @ 0x24;

/\*--------- Int переведенных скорости ------------------------\*/

static unsigned int Vprd @ 0x22;

static unsigned int Vmdd @ 0x26;

/\*------- Рабочее слово SlRab -------------------------\*/

static unsigned long SlRab @ 0x2c;

static unsigned char SlRab\_4 @ 0x2c;

static unsigned char SlRab\_3 @ 0x2d;

static unsigned char SlRab\_2 @ 0x2e;

static unsigned char SlRab\_1 @ 0x2f;

static unsigned char Adress @ 0x2f;/\* адреса (206/207/271)\*/

static bit unsigned char SlRab\_bit31 @ 0x67;

static bit unsigned char SlRab\_bit30 @ 0x66;

static bit unsigned char SlRab\_bit29 @ 0x65;

static bit unsigned char SlRab\_bit28 @ 0x64;

static bit unsigned char SlRab\_bit27 @ 0x63;

static bit unsigned char SlRab\_bit26 @ 0x62;

static bit unsigned char SlRab\_bit25 @ 0x61;

static bit unsigned char SlRab\_bit24 @ 0x60;

static bit unsigned char SlRab\_bit23 @ 0x6f;

static bit unsigned char SlRab\_bit22 @ 0x6e;

static bit unsigned char SlRab\_bit21 @ 0x6d;

static bit unsigned char SlRab\_bit20 @ 0x6c;

static bit unsigned char SlRab\_bit19 @ 0x6b;

static bit unsigned char SlRab\_bit18 @ 0x6a;

static bit unsigned char SlRab\_bit17 @ 0x69;

static bit unsigned char SlRab\_bit16 @ 0x68;

static bit unsigned char SlRab\_bit15 @ 0x77;

static bit unsigned char SlRab\_bit14 @ 0x76;

static bit unsigned char SlRab\_bit13 @ 0x75;

static bit unsigned char SlRab\_bit12 @ 0x74;

static bit unsigned char SlRab\_bit11 @ 0x73;

static bit unsigned char SlRab\_bit10 @ 0x72;

static bit unsigned char SlRab\_bit9 @ 0x71;

static bit unsigned char SlRab\_bit8 @ 0x70;

static bit unsigned char SlRab\_bit7 @ 0x7f;

static bit unsigned char SlRab\_bit6 @ 0x7e;

static bit unsigned char SlRab\_bit5 @ 0x7d;

static bit unsigned char SlRab\_bit4 @ 0x7c;

static bit unsigned char SlRab\_bit3 @ 0x7b;

static bit unsigned char SlRab\_bit2 @ 0x7a;

static bit unsigned char SlRab\_bit1 @ 0x79;

static bit unsigned char SlRab\_bit0 @ 0x78;

/\*-- Входные битовые Скорость Vpr Grey -----------------------\*/

static bit unsigned char Vprg\_bit9 @ 0x01;

static bit unsigned char Vprg\_bit8 @ 0x00;

static bit unsigned char Vprg\_bit7 @ 0x0f;

static bit unsigned char Vprg\_bit6 @ 0x0e;

static bit unsigned char Vprg\_bit5 @ 0x0d;

static bit unsigned char Vprg\_bit4 @ 0x0c;

static bit unsigned char Vprg\_bit3 @ 0x0b;

static bit unsigned char Vprg\_bit2 @ 0x0a;

static bit unsigned char Vprg\_bit1 @ 0x09;

static bit unsigned char Vprg\_bit0 @ 0x08;

/\*-- Выходные битовые Скорость Vpr Din -----------------------\*/

static bit unsigned char Vprd\_bit9 @ 0x11;

static bit unsigned char Vprd\_bit8 @ 0x10;

static bit unsigned char Vprd\_bit7 @ 0x1f;

static bit unsigned char Vprd\_bit6 @ 0x1e;

static bit unsigned char Vprd\_bit5 @ 0x1d;

static bit unsigned char Vprd\_bit4 @ 0x1c;

static bit unsigned char Vprd\_bit3 @ 0x1b;

static bit unsigned char Vprd\_bit2 @ 0x1a;

static bit unsigned char Vprd\_bit1 @ 0x19;

static bit unsigned char Vprd\_bit0 @ 0x18;

/\*-- Входные битовые Скорость Vmd Grey -----------------------\*/

static bit unsigned char Vmdg\_bit9 @ 0x21;

static bit unsigned char Vmdg\_bit8 @ 0x20;

static bit unsigned char Vmdg\_bit7 @ 0x2f;

static bit unsigned char Vmdg\_bit6 @ 0x2e;

static bit unsigned char Vmdg\_bit5 @ 0x2d;

static bit unsigned char Vmdg\_bit4 @ 0x2c;

static bit unsigned char Vmdg\_bit3 @ 0x2b;

static bit unsigned char Vmdg\_bit2 @ 0x2a;

static bit unsigned char Vmdg\_bit1 @ 0x29;

static bit unsigned char Vmdg\_bit0 @ 0x28;

/\*-- Выходные битовые Скорость Vmd Din -----------------------\*/

static bit unsigned char Vmdd\_bit9 @ 0x31;

static bit unsigned char Vmdd\_bit8 @ 0x30;

static bit unsigned char Vmdd\_bit7 @ 0x3f;

static bit unsigned char Vmdd\_bit6 @ 0x3e;

static bit unsigned char Vmdd\_bit5 @ 0x3d;

static bit unsigned char Vmdd\_bit4 @ 0x3c;

static bit unsigned char Vmdd\_bit3 @ 0x3b;

static bit unsigned char Vmdd\_bit2 @ 0x3a;

static bit unsigned char Vmdd\_bit1 @ 0x39;

static bit unsigned char Vmdd\_bit0 @ 0x38;

/\*------------------------------------------------------------------------\*/

/\* Процедура записи бита четности для SlRab \*/

/\*------------------------------------------------------------------------\*/

void CalcChetSl(void){

SlRab\_bit31=1;

SlRab\_bit31=SlRab\_bit31^SlRab\_bit0^SlRab\_bit1^SlRab\_bit2;

SlRab\_bit31=SlRab\_bit31^SlRab\_bit3^SlRab\_bit4^SlRab\_bit5;

SlRab\_bit31=SlRab\_bit31^SlRab\_bit6^SlRab\_bit7^SlRab\_bit8;

SlRab\_bit31=SlRab\_bit31^SlRab\_bit9^SlRab\_bit10^SlRab\_bit11;

SlRab\_bit31=SlRab\_bit31^SlRab\_bit12^SlRab\_bit13^SlRab\_bit14;

SlRab\_bit31=SlRab\_bit31^SlRab\_bit15^SlRab\_bit16^SlRab\_bit17;

SlRab\_bit31=SlRab\_bit31^SlRab\_bit18^SlRab\_bit19^SlRab\_bit20;

SlRab\_bit31=SlRab\_bit31^SlRab\_bit21^SlRab\_bit22^SlRab\_bit23;

SlRab\_bit31=SlRab\_bit31^SlRab\_bit24^SlRab\_bit25^SlRab\_bit26;

SlRab\_bit31=SlRab\_bit31^SlRab\_bit27^SlRab\_bit28^SlRab\_bit29;

SlRab\_bit31=SlRab\_bit31^SlRab\_bit30;

}

/\*------------------------------------------------------------------------\*/

/\* Процедура перевода из грея в двоичную допустимую скорость \*/

/\*------------------------------------------------------------------------\*/

void Perevod\_Md(void){

Vmdd=0;

Vmdd\_bit9=Vmdg\_bit9;

Vmdd\_bit8=Vmdg\_bit8^Vmdd\_bit9;

Vmdd\_bit7=Vmdg\_bit7^Vmdd\_bit8;

Vmdd\_bit6=Vmdg\_bit6^Vmdd\_bit7;

Vmdd\_bit5=Vmdg\_bit5^Vmdd\_bit6;

Vmdd\_bit4=Vmdg\_bit4^Vmdd\_bit5;

Vmdd\_bit3=Vmdg\_bit3^Vmdd\_bit4;

Vmdd\_bit2=Vmdg\_bit2^Vmdd\_bit3;

Vmdd\_bit1=Vmdg\_bit1^Vmdd\_bit2;

Vmdd\_bit0=Vmdg\_bit0^Vmdd\_bit1;

}

/\*------------------------------------------------------------------------\*/

/\* Процедура перевода из грея в двоичную предельную скорость \*/

/\*------------------------------------------------------------------------\*/

void Perevod\_Pr(void){

Vprd=0;

Vprd\_bit9=Vprg\_bit9;

Vprd\_bit8=Vprg\_bit8^Vprd\_bit9;

Vprd\_bit7=Vprg\_bit7^Vprd\_bit8;

Vprd\_bit6=Vprg\_bit6^Vprd\_bit7;

Vprd\_bit5=Vprg\_bit5^Vprd\_bit6;

Vprd\_bit4=Vprg\_bit4^Vprd\_bit5;

Vprd\_bit3=Vprg\_bit3^Vprd\_bit4;

Vprd\_bit2=Vprg\_bit2^Vprd\_bit3;

Vprd\_bit1=Vprg\_bit1^Vprd\_bit2;

Vprd\_bit0=Vprg\_bit0^Vprd\_bit1;

}

/\*------------------------------------------------------------------------\*/

/\* Процедура расстановки состояний \*/

/\*------------------------------------------------------------------------\*/

void Priznak\_Work(bit unsigned char a,bit unsigned char b){

SlRab\_bit30=a;

SlRab\_bit29=b;

}

/\*------------------------------------------------------------------------\*/

/\* Процедура расстановки идентифткатора \*/

/\*------------------------------------------------------------------------\*/

void Priznak\_Usvic(viod){

SlRab\_bit8=0;/\* Признак \*/

SlRab\_bit9=1;/\* от УСВИЦ-250 \*/

}

/\*------------------------------------------------------------------------\*/

/\* Процедура имтация входных данных \*/

/\*------------------------------------------------------------------------\*/

void ImVxodMd(void){

Vmdg=344;

}

/\*------------------------------------------------------------------------\*/

/\* Процедура имтация входных данных \*/

/\*------------------------------------------------------------------------\*/

void ImVxodPr(void){

Vprg=172;

}

/\*------------------------------------------------------------------------\*/

/\* Процедура проверки на отказ Vmd \*/

/\*------------------------------------------------------------------------\*/

void Cikl\_Otkaz\_Vmd(void){

int c=3,St;

Vmd\_Ok=0;

T1=0;

do

{

/\* vvodvmd();\*/

ImVxodMd(); /\* имитация входа \*/

Perevod\_Md();/\* перевод из грея в двоичную \*/

St=StecVmdd;

if ((St+test\_speed>Vmdd)&&(St-test\_speed<Vmdd))

{

StecVmdd=Vmdd;

break;

}

if (c==0)

{

Vmd\_Ok=1;

T1=1;

break;

}

c--;

}

while(c>(-1));

}

/\*------------------------------------------------------------------------\*/

/\* Процедура проверки на отказ Vpr \*/

/\*------------------------------------------------------------------------\*/

void Cikl\_Otkaz\_Vpr(void){

int b=3,St;

Vpr\_Ok=0;

RXD=0;

do

{

/\* vvodvpr(); \*/

ImVxodPr(); /\* имитация входа \*/

Perevod\_Pr();/\* перевод из грея в двоичную \*/

St=StecVprd;

if ((St+test\_speed>Vprd)&&(St-test\_speed<Vprd))

{

StecVprd=Vprd;

break;

}

if (b==0)

{

Vpr\_Ok=1;

RXD=1;

break;

}

b--;

}

while(b>(-1));

}

/\*------------------------------------------------------------------------\*/

/\* Процедура маштабирование для ЦАП с соответствием 1==0.63 \*/

/\*------------------------------------------------------------------------\*/

void Machtab(unsigned int V1){

double rab1;

rab1=(double)V1;

rab1\*=0.63;

CapSlowo=0;

CapSlowo=(unsigned int)rab1;

CapSlowo0<<=6;

}

/\*------------------------------------------------------------------------\*/

/\* Процедура формирование слова с соответствием с РТМ \*/

/\*------------------------------------------------------------------------\*/

void FormPTM(unsigned int V1){

double rab2;

SlRab=0;

rab2=(double)V1;

rab2/=0.11575;

SlRab=(unsigned long)rab2;

rab2-=(double)SlRab;

if (rab2>0.5) SlRab++;

SlRab<<=12;

}

/\*------------------------------------------------------------------------\*/

/\* Процедура тест контроля ПО \*/

/\*------------------------------------------------------------------------\*/

void TestControlPO(void){

int Vprd1=0,Vmdd1=0,Vprg1=0,Vmdg1=0;

PO\_Ok=Vmd\_Ok=Vpr\_Ok=0;

Vprg1=Vprg; /\* сохраняем Vprg \*/

Vmdg1=Vmdg; /\* сохраняем Vmdg \*/

Vprd1=Vprd; /\* сохраняем Vprd \*/

Vmdd1=Vmdd; /\* сохраняем Vmdd \*/

Vmdg=Vprg=speed;

Perevod\_Md();

Perevod\_Pr();

if (Vmdd!=200) { PO\_Ok=1; Vmd\_Ok=1; }

if (Vprd!=200) { PO\_Ok=1; Vpr\_Ok=1; }

/\*----------------------------- 206 --------------------------------------\*/

SlRab=0;

FormPTM(Vprd);

Adress=0x61;

Priznak\_Usvic();/\* признак от УСВИЦ-250 \*/

Priznak\_Work(1,0);/\* признак тест \*/

CalcChetSl();/\* Подсчет четности \*/

if(SlRab!=1080820321) { PO\_Ok=1; Vpr\_Ok=1; }

/\*----------------------------- 207 --------------------------------------\*/

SlRab=0;

FormPTM(Vmdd);

Adress=0xe1;

Priznak\_Usvic();/\* признак от УСВИЦ-250 \*/

Priznak\_Work(1,0);/\* признак тест \*/

CalcChetSl();/\* Подсчет четности \*/

if(SlRab!=3228304097) { PO\_Ok=1; Vmd\_Ok=1; }

SlRab=0;

/\*------------------- востанавливаем состояния --------------------------\*/

Vprg=Vprg1; /\* востанавливаем Vprg \*/

Vmdg=Vmdg1; /\* востанавливаем Vmdg \*/

Vprd=Vprd1; /\* востанавливаем Vprd \*/

Vmdd=Vmdd1; /\* востанавливаем Vmdd \*/

}

/\*------------------------------------------------------------------------\*/

/\* Процедура проверки скорости < 50км(200), => - нет вычисленных данных \*/

/\*------------------------------------------------------------------------\*/

void If\_Speed(unsigned int V){

if(V<200) Priznak\_Work(0,1); /\* признак нет вычисленных данных \*/

}

/\*------------------------------------------------------------------------\*/

/\* Запуск таймера \*/

/\*------------------------------------------------------------------------\*/

void Zapusk(void){

/\*---------- Чистка рабочих областей параметров--------------\*/

SlRab=0;NomParam=0;

/\*------------- Запуск таймера --------------------------\*/

asm(" mov TCON,#1 ");

asm(" anl IP,#0 ");

asm(" orl IP,#2 ");

asm(" mov TMOD,#01 ");

asm(" mov a,#0eah "); /\* число для загрузки таймера \*/

asm(" mov TH0,a "); /\* 65536-20\*36-685+67, где старшая часть \*/

asm(" mov a,#0d3h "); /\* равна eah, а младшая - d3h \*/

asm(" mov TL0,a ");

asm(" setb TCON.4 ");

asm(" anl IE,#0 ");

asm(" orl IE,#82h ");

}

/\*------------------------------------------------------------------------\*/

/\* Процедура растановки признаков проверок \*/

/\*------------------------------------------------------------------------\*/

void Priznak\_Test(void){

if((PK\_Vpr==0) && (Vpr\_Ok==0) && (Vmd\_Ok==0) && (PO\_Ok==0))

SlRab\_bit28=0;

else SlRab\_bit28=1;

SlRab\_bit27=PK\_Vpr;

SlRab\_bit26=Vpr\_Ok;

SlRab\_bit25=Vmd\_Ok;

SlRab\_bit24=PO\_Ok;

}

/\*------------------------------------------------------------------------\*/

/\* Процедура разовой команды ТЕСТ КОНТРОЛЬ \*/

/\*------------------------------------------------------------------------\*/

void Test\_Kontrol(void){

int V=0;

/\* vvodvmd(); \*/

ImVxodMd(); /\* имитация входа Vmd \*/

Perevod\_Md(); /\* перевод из грея в двоичную \*/

/\* vvodvpr();\*/

ImVxodPr(); /\* имитация входа Vmd \*/

Perevod\_Pr(); /\* перевод из грея в двоичную \*/

/\*----------------- Сохраняем тестовое значение --------------------------\*/

V=Vprd;

/\*----------------- Прабавляем тестовое значение -------------------------\*/

Vprd+=166\*4;

/\*----------------- Тест на наличие Vmd >= Vpr ---------------------------\*/

if(Vprd>=Vmdd) { RXD=0; PK\_Vpr=1; }

else { RXD=1; PK\_Vpr=0; }

/\*----------------- Отнимаем тестовое значение ---------------------------\*/

Vprd-=166\*4;

/\*----------------- Востанавливаем тестовое значение ---------------------\*/

Vprd=V;

/\*----------------- Маштабирование Vprd ----------------------------------\*/

Machtab(StecVprd);

VivodCapSl();

/\*------------- копируем в поле 271 функциональный тест -----------------\*/

SlRab=0;

Priznak\_Work(1,0);/\* признак функциональный тест \*/

CalcChetSl(); /\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlDs\_1=SlRab\_1;

SlDs\_2=SlRab\_2;

SlDs\_3=SlRab\_3;

SlDs\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*------------- копируем в поле 206 функциональный тест -----------------\*/

SlRab=0;

Priznak\_Work(1,0);/\* признак функциональный тест \*/

CalcChetSl(); /\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlVpr\_1=SlRab\_1;

SlVpr\_2=SlRab\_2;

SlVpr\_3=SlRab\_3;

SlVpr\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*------------- копируем в поле 207 функциональный тест -----------------\*/

SlRab=0;

Priznak\_Work(1,0);/\* признак функциональный тест \*/

CalcChetSl(); /\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlVmd\_1=SlRab\_1;

SlVmd\_2=SlRab\_2;

SlVmd\_3=SlRab\_3;

SlVmd\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*----------------- Тест Контроль программы ------------------------------\*/

TestControlPO();

/\*-------- Формирование дискретного 271 слова ----------------------------\*/

SlRab=0;

Adress=0x9d;/\* адрес слова 271 \*/

Priznak\_Test();/\* растановка признаков тестирования в слове \*/

Priznak\_Usvic();/\* признак от УСВИЦ-250 \*/

CalcChetSl();/\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlDs\_1=SlRab\_1;

SlDs\_2=SlRab\_2;

SlDs\_3=SlRab\_3;

SlDs\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*------ Формирование 206 слова в соответствии с РТМ, изм,3 для Vpr ------\*/

SlRab=0;

FormPTM(StecVprd);

Adress=0x61;/\* адрес слова 206 \*/

Priznak\_Usvic();/\* признак от УСВИЦ-250 \*/

Priznak\_Work(1,1);/\* признак нормальной работы \*/

If\_Speed(Vprd);/\* проверка на < 50км \*/

if ((Vpr\_Ok==1)||(PO\_Ok==1)) Priznak\_Work(0,0);/\* отказе \*/

CalcChetSl();/\* подсчет четности \*/

/\*------------- Копирование параметров в область вывода ------------------\*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlVpr\_1=SlRab\_1;

SlVpr\_2=SlRab\_2;

SlVpr\_3=SlRab\_3;

SlVpr\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*----- Формирование 207 слова в соответствии с РТМ, изм,3 для Vmd ------\*/

SlRab=0;

FormPTM(StecVmdd);

Adress=0xe1;/\* адрес слова 207 \*/

Priznak\_Usvic();/\* признак от УСВИЦ-250 \*/

Priznak\_Work(1,1);/\* признак нормальной работы \*/

If\_Speed(Vmdd); /\* проверка на < 50км \*/

if ((Vmd\_Ok==1)||(PO\_Ok==1)) Priznak\_Work(0,0);/\* отказе \*/

CalcChetSl();/\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlVmd\_1=SlRab\_1;

SlVmd\_2=SlRab\_2;

SlVmd\_3=SlRab\_3;

SlVmd\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

SlRab=0;

/\*-------- Сброс сторожевого таймера -------------------------------------\*/

TXD=0;

TXD=1;

}

/\*------------------------------------------------------------------------\*/

/\* Процедура очистки переменных \*/

/\*------------------------------------------------------------------------\*/

void Clear(void){

StecVmdd=StecVprd=0;

Vprd=Vmdd=Vprg=Vmdg=0;

SlRab=SlVmd=SlVpr=SlDs=CapSlowo=0;

SlRab=0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Н А Ч А Л О П Р О Г Р А М М Ы /

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main(void)

{

/\*---------- Запуск таймера на вывод -------------------------------------\*/

set\_vector(TIMER0,Timer);

Zapusk();

/\*-------- Сброс сторожевого таймера -------------------------------------\*/

TXD=0;

TXD=1;

/\*----------------- Обнуление данных -------------------------------------\*/

Clear();

/\*------------------------ Вывод на ЦАП ----------------------------------\*/

Machtab(StecVprd);

VivodCapSl();

/\*--------- Подготовка вывода 206 слова ----------------------------------\*/

SlRab=0;

Adress=0x61;

Priznak\_Usvic();/\* признак от УСВИЦ-250 \*/

Priznak\_Work(0,1);/\* признак нет вычисленных данных \*/

CalcChetSl();/\* Подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlVmd\_1=SlRab\_1;

SlVmd\_2=SlRab\_2;

SlVmd\_3=SlRab\_3;

SlVmd\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*--------- Подготовка вывода 207 слова ----------------------------------\*/

SlRab=0;

Adress=0xe1;

Priznak\_Usvic();/\* признак от УСВИЦ-250 \*/

Priznak\_Work(0,1);/\* признак нет вычисленных данных \*/

CalcChetSl();/\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlVpr\_1=SlRab\_1;

SlVpr\_2=SlRab\_2;

SlVpr\_3=SlRab\_3;

SlVpr\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*--------- Заполнение начальными данными Vpr и Vmd ----------------------\*/

a=3;

do {

/\* vvodvmd(); \*/

ImVxodMd(); /\* имитация входа Vmd \*/

Perevod\_Md(); /\* перевод из грея в двоичную \*/

StecVmdd=Vmdd; /\* Сохранение Vmd.\*/

/\* vvodvpr();\*/

ImVxodPr(); /\* имитация входа Vpr \*/

Perevod\_Pr(); /\* перевод из грея в двоичную \*/

StecVprd=Vprd; /\*Сохранение Vpr \*/

a--; }

while (a>0);

Nach: {

/\*-------- Сброс сторожевого таймера -------------------------------------\*/

TXD=0;

TXD=1;

/\*-------------------- Обновление данных Vpr и Vmd -----------------------\*/

/\* vvodvmd();\*/

ImVxodMd(); /\* имитация входа Vmd \*/

Perevod\_Md(); /\* перевод из грея в двоичную \*/

StecVmdd=Vmdd; /\*Сохранение Vм.д.\*/

/\* vvodvpr();\*/

ImVxodPr(); /\* имитация входа Vpr \*/

Perevod\_Pr(); /\* перевод из грея в двоичную \*/

StecVprd=Vprd; /\*Сохранение Vпр.д.\*/

/\*----------------- Цикл проверки на отказ Vmd ---------------------------\*/

Cikl\_Otkaz\_Vmd();

/\*----------------- Цикл проверки на отказ Vpr ---------------------------\*/

Cikl\_Otkaz\_Vpr();

/\*----------------- Тест на наличие РК "ТК" ------------------------------\*/

/\* {\*/

/\* if(INT0==1) Test\_Kontrol();\*/

/\* }\*/

/\* while(INT0==1)\*/

/\*----------------- Тест на наличие Vmd >= Vpr --------------------------\*/

if(Vprd>=Vmdd) { RXD=0; PK\_Vpr=1; }

else { RXD=1; PK\_Vpr=0; }

/\*----------------- Маштабирование Vprd ---------------------------------\*/

Machtab(StecVprd);

VivodCapSl();

/\*------------- копируем в поле 271 функциональный тест ----------------\*/

SlRab=0;

Priznak\_Work(1,0);/\* признак функциональный тест \*/

CalcChetSl(); /\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlDs\_1=SlRab\_1;

SlDs\_2=SlRab\_2;

SlDs\_3=SlRab\_3;

SlDs\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*------------- копируем в поле 206 функциональный тест ----------------\*/

SlRab=0;

Priznak\_Work(1,0);/\* признак функциональный тест \*/

CalcChetSl(); /\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlVpr\_1=SlRab\_1;

SlVpr\_2=SlRab\_2;

SlVpr\_3=SlRab\_3;

SlVpr\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*------------- копируем в поле 207 функциональный тест ----------------\*/

SlRab=0;

Priznak\_Work(1,0);/\* признак функциональный тест \*/

CalcChetSl(); /\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlVmd\_1=SlRab\_1;

SlVmd\_2=SlRab\_2;

SlVmd\_3=SlRab\_3;

SlVmd\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*----------------- Тест Контроль программы -----------------------------\*/

TestControlPO();

/\*-------- Формирование дискретного 271 слова ----------------------------\*/

SlRab=0;

Adress=0x9d;/\* адрес слова 271 \*/

Priznak\_Test();/\* растановка признаков тестирования в слове \*/

Priznak\_Usvic();/\* признак от УСВИЦ-250 \*/

CalcChetSl();/\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlDs\_1=SlRab\_1;

SlDs\_2=SlRab\_2;

SlDs\_3=SlRab\_3;

SlDs\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*------ Формирование 206 слова в соответствии с РТМ, изм,3 для Vpr ------\*/

SlRab=0;

FormPTM(StecVprd);

Adress=0x61;/\* адрес слова 206 \*/

Priznak\_Usvic();/\* признак от УСВИЦ-250 \*/

Priznak\_Work(1,1);/\* признак нормальной работы \*/

If\_Speed(Vprd);/\* проверка на < 50км \*/

if ((Vpr\_Ok==1)||(PO\_Ok==1)) Priznak\_Work(0,0);/\* отказ \*/

CalcChetSl();/\* подсчет четности \*/

/\*------------- Копирование параметров в область вывода -----------------\*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlVpr\_1=SlRab\_1;

SlVpr\_2=SlRab\_2;

SlVpr\_3=SlRab\_3;

SlVpr\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

/\*----- Формирование 207 слова в соответствии с РТМ, изм,3 для Vmd -----\*/

SlRab=0;

FormPTM(StecVmdd);

Adress=0xe1;/\* адрес слова 207 \*/

Priznak\_Usvic();/\* признак от УСВИЦ-250 \*/

Priznak\_Work(1,1);/\* признак нормальной работы \*/

If\_Speed(Vmdd); /\* проверка на < 50км \*/

if ((Vmd\_Ok==1)||(PO\_Ok==1)) Priznak\_Work(0,0);/\* отказ \*/

CalcChetSl();/\* подсчет четности \*/

asm(" clr TCON.4 ");/\* остановка таймера \*/

SlVmd\_1=SlRab\_1;

SlVmd\_2=SlRab\_2;

SlVmd\_3=SlRab\_3;

SlVmd\_4=SlRab\_4;

asm(" setb TCON.4 ");/\* запуск таймера \*/

SlRab=0;

/\*-------------------- G O T O -> N A C H ---------------------------\*/

}

goto Nach;

}

Приложение 2. Программы вывода слов в магистраль.

defseg c\_text,class=code

seg c\_text

int38T equ 65535-80\*36-80\*32-67+33+54

BaitV equ 2ah ; адрес выводного байта 28

Bit6V equ 51h ; Bit6V прямой вывод бита 80

Bit7V equ 52h ; Bit7V инверсной вывод бита 81

Sl206 equ 10h ; адрес слова 206

Sl207 equ 3ch ; адрес слова 207

Sl271 equ 38h ; адрес слова 271

NomParam equ 1bh ; Номер выводимого параметра 0=Sl206

global \_Timer,\_Vivod206,\_Vivod207;,\_Vivod271

\_Timer:

PUSH PSW ; время от начала прерывания \_\_\_11

CLR TCON.4 ; Останов таймеpа 1

CLR TCON.5 ; Сбpос флага пеpеполнения 1

PUSH 0E0h ; 2

MOV A,#low(int38T) ; Выбоp младшего байта INT\_38T 1

MOV TL0,A ; запись в TL0 1

MOV A,#high(int38T) ; Выбоp стаpшего байта INT\_38T 1

MOV TH0,A ; запись в TH0 1

SETB TCON.4 ; Запуск таймеpа Т0 1 \_\_\_20

MOV BaitV,#0ffh ; 2

MOV A,NomParam ; 1

;--------------------------------------------------------------------- 23

CJNE A,#0,GoPp ; 2

NOP ; 1

NOP ; 1

CALL \_Vivod206 ; 2\_\_\_33

INC NomParam ; 1

JMP GoEnd ; 2

GoPp: CJNE A,#1,GoSl ; 2

CALL \_Vivod207 ; 2

INC NomParam ; 1

JMP GoEnd ; 2

GoSl: CALL \_Vivod271 ; 2

MOV NomParam,#0 ; 2

NOP ; 1

GoEnd: MOV BaitV,#0h ; 2

POP 0E0h ; 2

POP PSW ; 2

RETI ; 2

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\_Vivod206:

PUSH 0D0H ; 2 mkc

PUSH 0E0H ; 2 mkc

PUSH 0 ; 2 mkc

PUSH 1 ; 2 mkc

PUSH 2 ; 2 mkc

PUSH 3 ; 2 mkc

CLR PSW.3 ; 1 mkc

CLR PSW.4 ; 1 mkc

ORL P0,#0C0H ; 2 mkc

MOV R3,#4 ; 1 mkс Количество пеpесылаемых байт #4

MOV R0,#Sl206 ; 1 mkс

;-------------------------------------------------------------------- 18

Bait206:

MOV A,@R0 ; 1 mkc

MOV R2,#7 ; 1 mkс

Bit206:

RRC A ; 1 mkс

ORL P0,#0ffh ; 2 mkc

;-------------------------------------------------------------------- 0

MOV Bit6V,C ; 2 mkc

MOV Bit7V,C ; 2 mkс

CPL Bit7V ; 1 mkс

MOV R1,#15 ; 1 mkс

Pauza206\_0: DJNZ R1,Pauza206\_0 ; 2\*15=30 mkc 36

NOP ; 1 mkc

NOP ; 1 mkc

MOV P0,BaitV ; 2 mkс 40 63

;-------------------------------------------------------------------- 40

;-------------------------------------------------------------------- 0

MOV R1,#18 ; 1 mkс

Pauza206\_1: DJNZ R1,Pauza206\_1 ; 2\*17=34 mkc

DJNZ R2,Bit206 ; 2 mkс 37

RRC A ; 1 mkс

ORL P0,#0ffh ; 2 mkc

;-------------------------------------------------------------------- 40

;-------------------------------------------------------------------- 0

MOV Bit6V,C ; 2 mkс

MOV Bit7V,C ; 2 mkс

CPL Bit7V ; 1 mkс

MOV R1,#16 ; 1 mkс 6

Pauza206\_2: DJNZ R1,Pauza206\_2 ; 2\*16=32 mkc 38

MOV P0,BaitV ; 2 mkс

;-------------------------------------------------------------------- 40

;-------------------------------------------------------------------- 0

MOV R1,#15 ; 1 mkс 00

Pauza206\_3: DJNZ R1,Pauza206\_3 ; 2\*15=30 mkc 31

INC R0 ; 2 mkc 33

DJNZ R3,Bait206 ; 2 mkс 35

;-------------------------------------------------------------------- 35

NOP ; 1 mkс

NOP ; 1 mkс

NOP ; 1 mkс

ORL P0,#0C0H ; 2 mkc 40

;-------------------------------------------------------------------- 40

;-------------------------------------------------------------------- 0

POP 3 ; 2 mkc

POP 2 ; 2 mkc

POP 1 ; 2 mkc

POP 0 ; 2 mkc

POP 0E0H ; 2 mkc

POP 0D0H ; 2 mkc

RET ; 2 mkc

;-------------------------------------------------------------------- 14

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\_Vivod207:

PUSH 0D0H ; 2 mkc

PUSH 0E0H ; 2 mkc

PUSH 0 ; 2 mkc

PUSH 1 ; 2 mkc

PUSH 2 ; 2 mkc

PUSH 3 ; 2 mkc

CLR PSW.3 ; 1 mkc

CLR PSW.4 ; 1 mkc

ORL P0,#0C0H ; 2 mkc

;-----------------------------------------------------------------------

MOV R3,#4 ; 1 mkс Количество пеpесылаемых байт #4

MOV R0,#Sl207 ; 1 mkс 18

;-----------------------------------------------------------------------

Bait207:

MOV A,@R0 ; 1 mkc

MOV R2,#7 ; 1 mkс

Bit207:

RRC A ; 1 mkс

ORL P0,#0ffh ; 2 mkc //0

;-----------------------------------------------------------------------

MOV Bit6V,C ; 2 mkc

MOV Bit7V,C ; 2 mkс

CPL Bit7V ; 1 mkс

MOV R1,#15 ; 1 mkс

Pauza207\_0: DJNZ R1,Pauza207\_0 ; 2\*15=30 mkc 36

NOP ; 1 mkc

NOP ; 1 mkc

MOV P0,BaitV ; 2 mkс 40 63

MOV R1,#18 ; 1 mkс 00

Pauza207\_1: DJNZ R1,Pauza207\_1 ; 2\*17=34 mkc

DJNZ R2,Bit207 ; 2 mkс 37

RRC A ; 1 mkс

ORL P0,#0ffh ; 2 mkc 40

;------------------------------------------------------------------------

MOV Bit6V,C ; 2 mkс 00

MOV Bit7V,C ; 2 mkс

CPL Bit7V ; 1 mkс

MOV R1,#16 ; 1 mkс 6

Pauza207\_2: DJNZ R1,Pauza207\_2 ; 2\*16=32 mkc 38

MOV P0,BaitV ; 2 mkс 40

MOV R1,#15 ; 1 mkс 00

Pauza207\_3: DJNZ R1,Pauza207\_3 ; 2\*15=30 mkc 31

;---------------------------------------------------------------------

INC R0 ; 2 mkc 33

DJNZ R3,Bait207 ; 2 mkс 35

NOP

NOP

NOP

ORL P0,#0C0H ; 2 mkc 40

;-------------------------------------------------------------------------

POP 3 ; 2 mkc

POP 2 ; 2 mkc

POP 1 ; 2 mkc

POP 0 ; 2 mkc

POP 0E0H ; 2 mkc

POP 0D0H ; 2 mkc

RET ; 2 mkc

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\_Vivod271:

PUSH 0D0H ; 2 mkc

PUSH 0E0H ; 2 mkc

PUSH 0 ; 2 mkc

PUSH 1 ; 2 mkc

PUSH 2 ; 2 mkc

PUSH 3 ; 2 mkc

CLR PSW.3 ; 1 mkc

CLR PSW.4 ; 1 mkc

ORL P0,#0C0H ; 2 mkc

MOV R3,#4 ; 1 mkс Количество пеpесылаемых байт #4

MOV R0,#Sl271 ; 1 mkс 18

;-----------------------------------------------------------------------

Bait271:

MOV A,@R0 ; 1 mkc

MOV R2,#7 ; 1 mkс

Bit271:

RRC A ; 1 mkс

ORL P0,#0ffh ; 2 mkc //0

;-----------------------------------------------------------------------

MOV Bit6V,C ; 2 mkc

MOV Bit7V,C ; 2 mkс

CPL Bit7V ; 1 mkс

MOV R1,#15 ; 1 mkс

Pauza271\_0: DJNZ R1,Pauza271\_0 ; 2\*15=30 mkc 36

NOP ; 1 mkc

NOP ; 1 mkc

MOV P0,BaitV ; 2 mkс 40 63

MOV R1,#18 ; 1 mkс 00

Pauza271\_1: DJNZ R1,Pauza271\_1 ; 2\*17=34 mkc

DJNZ R2,Bit271 ; 2 mkс 37

RRC A ; 1 mkс

ORL P0,#0ffh ; 2 mkc 40

;------------------------------------------------------------------------

MOV Bit6V,C ; 2 mkс 00

MOV Bit7V,C ; 2 mkс

CPL Bit7V ; 1 mkс

MOV R1,#16 ; 1 mkс 6

Pauza271\_2: DJNZ R1,Pauza271\_2 ; 2\*16=32 mkc 38

MOV P0,BaitV ; 2 mkс 40

MOV R1,#15 ; 1 mkс 00

Pauza271\_3: DJNZ R1,Pauza271\_3 ; 2\*15=30 mkc 31

;---------------------------------------------------------------------

INC R0 ; 2 mkc 33

DJNZ R3,Bait271 ; 2 mkс 35

NOP

NOP

NOP

ORL P0,#0C0H ; 2 mkc 40

;-------------------------------------------------------------------------

POP 3 ; 2 mkc

POP 2 ; 2 mkc

POP 1 ; 2 mkc

POP 0 ; 2 mkc

POP 0E0H ; 2 mkc

POP 0D0H ; 2 mkc

RET ; 2 mkc

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

END

Приложени 3. Программа вывода приборной скорости на ЦАП.

defseg c\_text,class=code

seg c\_text

CapSlowo0 equ 28h;

CapSlowo1 equ 29h;

global \_VivodCapSl

\_VivodCapSl:

push 0e0h

push 1

push 2

push 3

clr tcon.4

mov a,CapSlowo0

mov p0,a

mov a,CapSlowo1

mov p2,a

setb p0.3

nop

nop

nop

nop

nop

clr p0.3

setb tcon.4

pop 3

pop 2

pop 1

pop 0e0h

ret

End

Приложение 4. Программа считывания приборной скорости с маски.

defseg c\_text,class=code

seg c\_text

;адреса битов на Vmd

bitvmd0 equ 28h

bitvmd1 equ 29h

bitvmd2 equ 2ah

bitvmd3 equ 2bh

bitvmd4 equ 2ch

bitvmd5 equ 2dh

bitvmd6 equ 2eh

bitvmd7 equ 2fh

bitvmd8 equ 20h

bitvmd9 equ 21h

global \_vvodvmd

\_vvodvmd:

push 0e0h

push 1

push 2

push 3

clr tcon.4

orl p0,#1 ;запрещение работы D1

anl p0,#1 ;запрещение работы D1

orl p1,#14h ;запрещение D2 и разрешение D4

anl p1,#14h ;запрещение D2 и разрешение D4

;считываем для Vmd

call Read\_P\_1\_5

mov bitvmd0,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvmd1,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvmd2,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvmd3,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvmd4,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvmd5,c

orl p1,#11h ;запрет d2 и разрешение d4

anl p1,#11h ;запрет d2 и разрешение d4

anl p0,#0 ;запрет d5

call Read\_P\_1\_5

mov bitvmd6,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvmd7,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvmd8,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvmd9,c

setb tcon.4

pop 3

pop 2

pop 1

pop 0e0h

ret

Read\_P\_1\_5:

setb p1.5

mov r3,#30 ;250

Pause\_P\_1\_5:

nop

nop

djnz r3,Pause\_P\_1\_5

mov c,p1.5

ret

end

Приложение 5. Программа считывания максимальной допустимой скорости с маски.

defseg c\_text,class=code

seg c\_text

;адреса битов на Vpr

bitvpr0 equ 08h

bitvpr1 equ 09h

bitvpr2 equ 0ah

bitvpr3 equ 0bh

bitvpr4 equ 0ch

bitvpr5 equ 0dh

bitvpr6 equ 0eh

bitvpr7 equ 0fh

bitvpr8 equ 00h

bitvpr9 equ 01h

global \_vvodvpr

\_vvodvpr:

push 0e0h

push 1

push 2

push 3

clr tcon.4

orl p0,#1 ;запрещение работы D1

anl p0,#1 ;запрещение работы D1

orl p1,#1 ;запрещение D4 и разрешение D2

anl p1,#1 ;запрещение D4 и разрешение D2

;считываем для Vpr

call Read\_P\_1\_5

mov bitvpr0,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvpr1,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvpr2,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvpr3,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvpr4,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvpr5,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvpr6,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvpr7,c

orl p1,#10h ;запрещение D2 и разрешение D4

anl p1,#10h ;запрещение D2 и разрешение D4

call Read\_P\_1\_5

mov bitvpr8,c

inc p1

inc p1

call Read\_P\_1\_5

mov bitvpr9,c

setb tcon.4

pop 3

pop 2

pop 1

pop 0e0h

ret

Read\_P\_1\_5:

setb p1.5

mov r3,#30 ;250

Pause\_P\_1\_5:

nop

nop

djnz r3,Pause\_P\_1\_5

mov c,p1.5

ret

end