# **ANOTĀCIJA**

# Šajā darbā tiek aplūkota informācijas apmaiņas organizācija starp personālo datoru un firmai Intel MCS-51 mikrokontrolieri. Darbā tiek uzrādīti mikrokontroliera programmas algoritmi un tā arī strukturālās pieslēgumu shēmas pie personālo datoru portam.

# Darba apjoms 47 lappuses.

# **АННОТАЦИЯ**

В данной работе рассматривается организация обмена информацией между персональным компьютером и микроконтроллером семейства MCS-51 фирмы Intel. В работе представлены алгоритмы программного обеспечения микроконтроллера, а также структурные схемы подключения микроконтроллера к портам персонального компьютера.

Объем работы 47 страниц.

## ANNOTATION

# In given document is considered organization of information exchange between the personal computer and family MCS-51 Company Intel microcontroller. There is presented the algorithms of microcontroller software, as well as structured schemes of connecting a microcontroller to ports of the personal computer.

# Volumes of work 47 pages.

**СОДЕРЖАНИЕ**

стр.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ 7

ВВЕДЕНИЕ 8

1. АНАЛИЗ СОСТОЯНИЯ ВОПРОСА 9

1.1 Постановка глобальных задач 13

1.2 Анализ предыдущей работы 13

1.2.1 Положительные стороны 13

1.2.2 Отрицательные стороны 14

1.3 Постановка задачи 15

2. РАЗРАБОТКА ВОПРОСОВ АППАРАТНОГО И ПРОГРАММНОГО СОПРЯЖЕНИЯ 18

2.1 Аппаратное сопряжение ПК и микроконтроллера 18

2.1.1 Скорость приема/передачи 19

2.1.2 Разработка формата принимаемых и передаваемых данных 19

2.1.3 Разработка схемы подключения микроконтроллера 20

2.1.4 Выбор источника питания 21

2.2 Подключение внешней памяти программ 21

2.3 Программное сопряжение микроконтроллера и ПК 23

2.3.1 Начальная установка MCS-51 23

2.3.2 Программное обеспечение организации обмена информацией между МК и ПК 23

2.3.2.1 Программа «Монитор» 23

2.3.2.2 Подпрограмма запуска программы пользователя в режиме реального времени 26

2.3.2.3 Подпрограмма запуска программы пользователя в пошаговом режиме 28

2.3.2.4 Подпрограмма записи программы пользователя в память программ микроконтроллера. 34

2.3.2.5 Подпрограмма записи информации в программно – доступные узлы микроконтроллера 36

2.3.2.6 Подпрограмма чтения из памяти программ микроконтроллера 37

2.3.2.7 Подпрограмма чтения информации программно – доступных узлов микроконтроллера 38

2.3.2.8 Подпрограмма выдачи ошибки в ПК 39

2.3.2.9 Подпрограмма выдачи одного байта информации 39

2.3.2.10 Подпрограмма приема одного байта информации. 41

СПИСОК ИСПОЛЬЗОВАНОЙ ЛИТЕРАТУРЫ 45

ПРИЛОЖЕНИЯ 46

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

БИС - большая интегральная схема

МК - микроконтроллер

ОЗУ - оперативное запоминающее устройство

ПЗУ - постоянное запоминающее устройство

ПК - персональный компьютер

УАПП - универсальный асинхронный приемопередатчик

ЭВМ - электронная вычислительная машина

ВВЕДЕНИЕ

Развитие микроэлектроники и широкое применение ее изделий в промышленном производстве, в устройствах и системах управления самыми разнообразными объектами и процессами является в настоящее время одним из основных направлений научно-технического прогресса.

Использование микроконтроллеров в изделиях не только приводит к повышению технико-экономических показателей (стоимости, надежности, потребляемой мощности, габаритных размеров), но и позволяет сократить время разработки изделий и делает их модифицируемыми, адаптивными. Использование микроконтроллеров в системах управления обеспечивает достижение высоких показателей эффективности при низкой стоимости.

Микроконтроллеры представляют собой эффективное средство автоматизации разнообразных объектов и процессов.

Все это определяет необходимость изучения микропроцессорных систем. В настоящее время в РАУ имеются учебные методические комплексы УМК ВЭФ, базирующиеся на микропроцессоре I8080, позволяющие получить знания в программировании микропроцессоров. К сожалению, на кафедре нет лабораторной установки, позволяющей получить практические навыки в программировании микроконтроллеров.

Необходимо создание новой лабораторной базы, использующей на наиболее распространенные микроконтроллеры. Такими микроконтроллерами могут послужить микроконтроллеры семейства MCS-51 фирмы Intel.

Такие лабораторные установки могут использоваться не только как учебно-методическое пособие при изучении курса микропроцессоров, но и как устройства управления другими учебно-методическими комплексами, используемыми в других курсах.

Вышесказанное указывает на актуальность рассмотрения вопроса организации обмена информацией между персональным компьютером и микроконтроллером MCS-51 фирмы Intel.

1. АНАЛИЗ СОСТОЯНИЯ ВОПРОСА

В устройствах управления объектами (контроллерах) на основе МК аппаратурные средства и программное обеспечение существует в форме неделимого аппаратурно-программного комплекса. При проектировании контроллеров приходиться решать одну из самых сложных задач разработки, а именно задачу оптимального распределения функций контроллера между аппаратурными средствами и программным обеспечением. Решение этой задачи осложняется тем, что взаимосвязь и взаимовлияние аппаратурных средств и программного обеспечения в микропроцессорной технике претерпевают динамические изменения. Если в начале развития микропроцессорной техники определяющим было правило, в соответствии с которым аппаратурные средства обеспечивают производительность, а программное обеспечение – дешевизну изделия, то в настоящее время это правило нуждается в серьезной корректировке. Так как МК представляет собой стандартный массовый (относительно недорогой) логический блок, конкретное назначение которого определяет пользователь с помощью программного обеспечения, то с ростом степени интеграции и, следовательно, функционально-логических возможностей МК резко понижается стоимость изделия в пересчете на выполняемую функцию, что в конечном итоге и обеспечивает достижение высоких технико-экономических показателей изделий на МК. При этом затраты на разработку программного обеспечения изделия в 2 – 10 раз превышают затраты на приобретение и изготовление аппаратурных средств.

В настоящее время наибольшее распространение получил методологический прием, при котором весь цикл разработки контроллеров рассматривается как последовательность трех фаз проектирования:

1. анализа задачи и выбора (и/или разработки) аппаратурных средств контроллера;
2. разработка прикладного программного обеспечения;
3. комплексирования аппаратурных средств и программного обеспечения в прототипе контроллера и его отладки.

Фаза разработки программного обеспечения, т.е. фаза получения прикладных программ, в свою очередь, разбивается на два существенно различных этапа:

1. «от постановки задачи к исходной программе»;
2. «от исходной программы к объектному модулю».

Этап разработки «от исходной программы к объектному модулю» имеет целью получение машинных кодов прикладных программ, работающих в МК. Этот этап разработки прикладного программного обеспечения легко поддается формализации и поддержан всей мощью системного программного обеспечения МК, направленного на автоматизацию процесса получения прикладных программ. В состав средств системного программного обеспечения входят трансляторы с различных алгоритмических языков высокого уровня, ассемблеры, редакторы текстов, программы-отладчики, программы-документаторы и т.д. Наличие всех этих системных средств придает инженерной работе на этом этапе проектирования контроллеров характер ремесла, а не инженерного творчества. Так как в конечном изделии имеются только МК и его средства сопряжения с объектом, то выполнять отладку разрабатываемого прикладного программного обеспечения на нем невозможно (из-за отсутствия средств ввода, вывода, ОЗУ большой емкости и операционной системы), и, следовательно, разработчик вынужден обращаться к средствам вычислительной техники для выполнения всех формализуемых стадий разработки: трансляции, редактирования, отладки, загрузки объектных кодов и программируемую постоянную память МК.

Этап разработки «от постановки задачи к исходной программе» не поддается формализации и, следовательно, не может быть автоматизирован. Проектная работа здесь носит творческий характер, изобилует решениями, имеющими «волевую» или «вкусовую» окраску, и решениями, продиктованными конъюнктурными соображениями. На этом этапе разработчик стакивается с наибольшим количеством трудностей.

На обоих этапах разработки необходимо тестировать программное обеспечение не только на эмуляторах, но и на «живом» МК, с целью выявления специфических ошибок (неправильная логика работы устройства, ошибки, связанные с эмуляцией). Это требует многократного перепрограммирования МК, что связанно с большой затратой времени (время стирания информации в ПЗУ с ультрафиолетовым, или электрическим стиранием может достигать нескольких десятков минут). Это время можно сократить используя в качестве памяти программ не ПЗУ, а ОЗУ.

Разрабатываемое устройство значительно упростит оба этапа разработки, позволяя отлаживать программное обеспечение непосредственно на «живом» МК и позволит сэкономить время, связанное с записью и стиранием тестируемых программ.

При решении задач об оптимальном распределении функций между аппаратурными средствами и программным обеспечением необходимо исходить из того, что использование специализированных интерфейсных БИС упрощает разработку и обеспечивает высокое быстродействие системы в целом, но сопряжено с увеличением стоимости, объема и потребляемой мощности. Больший удельный вес программного обеспечения позволяет сократить число компонентов системы и стоимость ее аппаратурных средств, но это приводит к снижению быстродействия и увеличению затрат и сроков разработки и отладки прикладных программ. При этом еще может несколько увеличиваться число БИС внешней памяти МК - системы. Решение о выборе того или иного варианта распределения функций между аппаратурными и программными средствами системы принимается в зависимости от тиражности изделия, ограничений по стоимости, объему, потребляемой мощности и быстродействию изделия. Программная реализация основных элементов алгоритма работы контроллера допускает его модификацию путем перепрограммирования. В то время как возможность изменения уже существующей фиксации элементов алгоритма в аппаратуре контроллера практически отсутствует.

После получения объектного кода программы неизбежно наступает этап отладки, т.е. установления факта ее работоспособности, а также выявления и устранения ошибок. Без этого этапа разработки никакое программное обеспечение вообще не имеет права на существование.

Обычно отладка прикладного программного обеспечения осуществляется в несколько этапов. Простые (синтаксические) ошибки выявляются уже на этапе трансляции. Далее необходимо выполнить:

автономную отладку каждой процедуры в статическом режиме, позволяющую проверить правильность проводимых вычислений, правильность последовательности переходов внутри процедуры (отсутствие «зацикливания») и т.п.;

комплексную отладку программного обеспечения в статическом режиме, позволяющую проверить правильность алгоритма управления (по последовательности формирования управляющих воздействий);

комплексную отладку в динамическом режиме без подключения объекта для определения реального времени выполнения программы и ее отдельных фрагментов.

Эти этапы отладки осуществляются обычно с использованием кросс систем. В состав кросс систем входят программы-отладчики, интерпретирующие выполнение программ написанных для МК. Но как бы ни был хорош интерпретатор, он все равно не может полностью заменить реальный МК.

С использованием разрабатываемого устройства можно будет выполнять рассмотренные этапы отладки уже непосредственно на «живом» МК, подключая к нему реальные физические объекты. Эти этапы отладки можно будет объединить со следующими этапами разработки устройства – отладка отдельных фрагментов программного обеспечения на отладочном модуле в режиме реального времени. Можно будет исключить этап комплексной отладки прикладного программного обеспечения на инструментальной микроЭВМ с внутрисхемным эмулятором.

Разрабатываемое устройство должно обеспечить все необходимые возможности, доступные в кросс системах:

доступ к любому ресурсу МК;

пошаговое исполнение программ.

Разрабатываемое устройство позволит промоделировать практически все возможные варианты работы программы и тем самым убедиться в ее работоспособности. Здесь возможна проверка работоспособности программы при нештатных ситуациях в условиях поступления некорректных входных воздействий.

Можно будет моделировать среду обитания МК, т.е. различного рода объекты и датчики, подключаемые к нему.

Это устройство устраняет главный недостаток кросс систем – невозможность прогона программы в реальном масштабе времени, т.е. со скоростью близкой к скорости выполнения программы в самом МК, а также невозможность комплексирования аппаратурных и программных средств разрабатываемой системы. Именно эти причины влияют на достоверность прикладных программ, отлаженных в кросс системах. Эта достоверность, как правило, не достаточно высока.

Задачей данной работы является разработка необходимого программного обеспечения и аппаратных средств сопряжения МК и ПК.

1.1 Постановка глобальных задач

Организация обмена информацией предполагает:

* рассмотрение вопросов аппаратных средств;
* создание необходимого программного обеспечения.

Аппаратные средства должны обеспечить:

* физическое сопряжение портов ПК и микроконтроллера;
* сопряжение МК с внешней памятью программ.

Программное обеспечение должно обеспечить решение следующего ряда задач:

* запись программы, отлаженной на ПК, в память программ и данных МК;
* выполнение программы в режиме реального времени;
* выполнение программы в пошаговом режиме;
* запись информации из ПК в программно-доступные узлы МК;
* чтение содержимого программно-доступных узлов и индикация их на мониторе ПК.

1.2 Анализ предыдущей работы

Вопрос об организации обмена информацией между персональным компьютером и микроконтроллером семейства Intel MCS-51 был уже рассмотрен в бакалаврской работе [3]. В этой работе были рассмотрены проблемы аппаратного и программного сопряжения МК с ПК в составе планируемой лабораторной установки.

1.2.1 Положительные стороны

В работе [3] было предложено использовать в качестве интерфейса сопряжения с ПК интерфейс RS-232C. Выбор данного интерфейса основывался на простоте подключения, а также минимальных затратах связанных с аппаратным подключением. Положительным фактом для выбора этого интерфейса служит также возможность удаленного подключения и возможность дуплексной передачи данных, что не могут обеспечить внутренняя шина ПК и интерфейс Centronics.

Кроме того каждый компьютер оборудован хотя бы одним асинхронным последовательным адаптером. Асинхронный адаптер обычно содержит несколько СОМ – портов, через которые к компьютеру можно подключить внешние устройства.

Используя СОМ – порт данные не надо синхронизировать. Для синхронизации передающего и принимающего устройства группе битов предшествует специальный стартовый бит, а после группы битов следуют один или два стоповых бита. Также в конце группы передаваемых битов может следовать бит четности. Наличие или отсутствие бита четности, количество стартовых и стоповых битов определяют формат передачи данных.

В работе [3] был предложен формат состоящий из одного стартового бита, семи информационных битов, бита четности и двух стоповых битов. Данный формат не является удобным, т.к. передавать информацию, разбивая ее на семь информационных битов не удобно. Гораздо удобнее использовать побайтовую пересылку.

Одним из основных плюсов данной работы является предложение использовать в качестве памяти программ ОЗУ вместо ПЗУ, что позволяет существенно увеличить скорость перезаписи программ пользователя.

ОЗУ используется для размещения программ пользователя. При записи программ в ОЗУ, обращение к памяти программ осуществляется как обращение к памяти данных, чтение как к памяти программ.

Использование памяти программ во время записи как память данных обуславливается тем, что во время работы микроконтроллера память программ доступна пользователю только для чтения.

1.2.2 Отрицательные стороны

Одним из минусов работы [3] является предложение о согласовании уровней между интерфейсом RS-232C и последовательным портом МК. Были разработаны схемы приемника и передатчика. Эти схемы имеют ряд недостатков. Одним из недостатков является необходимость в дополнительных источниках питания на +12 и –12 вольт. Данные схемы не являются надежными. Они используют устаревшую электронную базу и не удовлетворяют реальным требованиям, поставленным перед ними. В настоящее время имеется ряд микросхем, позволяющий осуществить согласование уровней сигналов между интерфейсом RS-232C и последовательным портом МК. Такие схемы более надежны и не требуют дополнительного питания на +12 и –12 вольт.

В работе [3] предлагалось использовать МК 8031. Данный МК не имеет внутренней памяти программ. В связи с этим возникла необходимость в установке внешней памяти программ. Эта память программ разбивалась на ПЗУ, для хранения программы «Монитор» и ОЗУ, как память программ пользователя. Это ограничивает общую внешнюю память программ, используемую как память программ пользователя.

В работе 3 не были решены вопросы с доступом к программно-доступным узлам микроконтроллера. При переходе к подпрограмме микроконтроллер сохраняет в стеке только содержимое счетчика команд. Поэтому перед началом приема или передачи данных необходимо позаботиться о сохранение содержимого аккумулятора и регистров, которые будут использоваться при приеме или передачи данных.

Программа «Монитор» учитывала только ряд задач, которые могут возникнуть у пользователя на этапах разработки программного обеспечения, тестирования его или использования системы на практике.

1.3 Постановка задачи

Из проведенного выше анализа следует, что при реализации процесса обмена информацией между микроконтроллером MCS-51 и персональным компьютером необходимо решить следующий ряд вопросов:

* Вопросы аппаратных средств.
* Разработка схемы сопряжения портов ПК и микроконтроллера с использованием современной электронной базы. Данная схема должна обеспечить согласование уровней сигналов между СОМ – портом ПК и микроконтроллером, не должна требовать дополнительных источников питания, должна иметь небольшие масса - габаритные характеристики, должна быть дешевой и надежной.
* Разработка схемы подключения памяти к микроконтроллеру. Данная схема должна обеспечить возможность использования ОЗУ в качестве памяти программ. Во время записи программы пользователя память программ должна использоваться, как память данных. Адресное пространство внешней памяти данных и памяти программ не должно перекрываться.
* Вопросы программного обеспечения.
* Разработать формат передачи данных между ПК и микроконтроллером. Этот формат должен использовать восемь информационных битов.
* Разработать алгоритм записи программы, отлаженной на ПК, в память программ МК.
* Разработать алгоритм запуска программы в режиме реального времени.
* Разработать алгоритм выполнения программы в пошаговом режиме. В этом режиме микроконтроллер на каждом шагу программы должен передавать в персональный компьютер содержимое всех программных узлов, а также следующую команду из памяти программ. При необходимости, по запросу из персонального компьютера микроконтроллер должен передать содержимое любой ячейки памяти данных, или любого блока ячеек памяти данных. Кроме того микроконтроллер должен передавать в персональный компьютер содержимое ячеек памяти программ, если это необходимо. При необходимости должна обеспечиваться запись в программно доступные узлы микроконтроллера, в память программ и память данных. Алгоритм должен позволять запускать программу с любого адреса указанного пользователем, продолжать выполнение прерванной программы в пошаговом режиме, или в режиме реального времени.
* Разработать алгоритм записи информации в программно – доступные узлы микроконтроллера.
* Разработать алгоритм записи информации в память данных.
* Разработать алгоритм чтения информации из программно – доступных узлов микроконтроллера.
* Разработать алгоритм чтения информации из памяти данных.
* Разработать алгоритм чтения информации из памяти программ.
* Разработать алгоритм программы «Монитор». Программа «Монитор» должна обеспечивать запуск всех рассмотренных выше подпрограмм в зависимости от кода, переданного из персонального компьютера. Поэтому необходимо разработать управляющие слова, которые и будут задавать режим работы системы. Управляющие коды не должны превышать 8 бит. Для реализации гибкости системы, необходимо при получении неиспользуемых кодовых комбинаций обеспечить переход выполнения программы в область памяти программ на заранее зарезервированный адрес. Этот адрес можно хранить в памяти программ по строго определенному адресу. Чтобы избежать выполнение случайного кода, во время начальной установки необходимо установить по этому адресу, адрес в памяти программ, в котором находится команда возврата из подпрограммы.
* При переходе на разрабатываемые подпрограммы необходимо сохранить в стеке содержимое аккумулятора. Перед выходом из этих подпрограммы содержимое аккумулятора необходимо восстановить.

2. РАЗРАБОТКА ВОПРОСОВ АППАРАТНОГО И ПРОГРАММНОГО СОПРЯЖЕНИЯ

2.1 Аппаратное сопряжение ПК и микроконтроллера

Для решения задачи сопряжения ПК и микроконтроллера было решено использовать интерфейс RS-232C.

Последовательный порт используется в качестве универсального асинхронного приемопередатчика (УАПП) с фиксированной или переменной скоростью последовательного обмена информацией и возможностью дуплексного включения.

Последовательный интерфейс микроконтроллера МК-51 может работать в следующих четырех режимах:

* **Режим 0.** Информация передается и принимается через вход RxD приемника (вывод P3.0). Через выход передатчика TxD (вывод P3.1) выдаются импульсы синхронизации, стробирующие каждый передаваемый или принимаемый бит информации. Формат посылки – 8 бит. Частота приема и передачи – тактовая частота микроконтроллера.
* **Режим 1.** Информация передается через выход передатчика TxD, а принимается через вход приемника RxD. Формат посылки – 10 бит: старт-бит (ноль), восемь бит данных, программируемый девятый бит и стоп-бит (единица). Частота приема и передачи задается таймером/счетчиком 1.
* **Режим 2.** Информация передается через выход передатчика TxD, а принимается через вход приемника RxD. Формат посылки – 11 бит: старт-бит (ноль), восемь бит данных, программируемый девятый бит и 2 стоп-бита (единицы). Передаваемый девятый бит данных принимает значение бита ТВ8 из регистра специальных функций SCON. Бит ТВ8 в регистре SCON может быть программно установлен в «0» или в «1», или в него, к примеру, можно поместить значение бита Р из регистра PSW для повышения достоверности принимаемой информации (контроль по паритету). При приеме девятый бит данных принятой посылки поступает в бит RB8 регистра SCON. Частота приема и передачи в режиме 2 задается программно и может быть равна тактовой частоте микроконтроллера деленной на 32 или на 64.
* **Режим 3.** Режим 3 полностью идентичен режиму 2 за исключением частоты приема и передачи, которая в режиме 3 задается таймером/счетчиком 1.

Для реализации обмена информацией между ПК и микроконтроллером наиболее удобным является режим 2, т.к. для работы в этом режиме не требуется таймер/счетчик. Этот режим полностью удовлетворяет предъявленным требованиям.

2.1.1 Скорость приема/передачи

Скорость приема/передачи, т.е. частота работы универсального асинхронного приемопередатчика (УАПП) в режиме 2 зависит от значения управляющего бита SMOD в регистре специальных функций.

Частота передачи определяется выражением:

*f*=(2SMOD/64)*f*рез.

Иными словами, при SMOD=0 частота передачи равна (1/64)*f*рез, а при SMOD=1 равна (1/32)*f*рез.

Скорость приема/передачи должна быть кратна 115,2 Кбод [6].

Исходя из вышеизложенного, выберем частоту приема данных при SMOD=1. Если *f*рез=1,8432 МГц, тогда частота приема данных будет 57,6 КБод.

Другие значения частот кварца могут быть выбраны из таблиц в приложении 1 и приложении 2.

2.1.2 Разработка формата принимаемых и передаваемых данных

Формат принимаемых и передаваемых данных почти полностью описан режимом 2 работы последовательного интерфейса.

Формат должен состоять из 11 бит:

* стартовый бит – ноль;
* восемь бит данных;
* девятый бит – контроль по паритету, для повышения достоверности принимаемой информации;
* два стоповых бита – единицы.

2.1.3 Разработка схемы подключения микроконтроллера

При сопряжении интерфейса RS-232 со входами последовательного интерфейса микроконтроллера MCS-51, необходимо обеспечить согласование уровней сигналов. Стандартный уровень сигналов RS-232C – +12 и –12 В. Стандартный уровень сигналов асинхронного интерфейса микроконтроллера MCS-51 – +5 В.

В зависимости от требований, предъявляемых к проектируемой схеме, преобразователи уровней сигналов могут быть выполнены различными способами.

Приемники и передатчики выполненные на дискретных элементах имеют ряд недостатков:

* большие массогабаритные характеристики;
* высокая потребляемая мощность;
* недостаточная надежность;
* необходимость в дополнительных источниках питания;
* стоимость.



*Рис. 1. Схема подключения микроконтроллера MCS-51 фирмы Intel к портам персонального компьютера.*

Исходя из этих соображений, для организации сопряжения уровней портов ПК и микроконтроллера можно воспользоваться микросхемой ADM202 фирмы Analog Devices. Данная микросхема позволяет согласовать уровни сигналов, не требуя дополнительных источников питания [4].

Микроконтроллер принимает данные через вход приемника (вывод P3.0), а передает данные через выход передатчика (вывод P3.1). Микроконтроллер может запретить прием данных из ПК с помощью сигналов на выводе Р2.6. Это необходимо для сигнализации ПК о приеме ошибочных данных. В свою очередь ПК может запретить передачу данных из микроконтроллера с помощью сигналов на выводе Р2.7.

2.1.4 Выбор источника питания

В зависимости от места установки, источником питания может служить либо блок питания ПК, либо отдельный блок питания, специально предназначенный для данной установки. В связи с тем, что данную установку планируется использовать в компьютерном классе, наиболее оптимальным вариантом является использование блока питания ПК. Напряжение источника питания *+5V*, необходимое для лабораторной установки, можно взять с внешнего разъема клавиатуры. Подключение контактов этого разъема представлено на рисунке 2.



*Рис. 2. Разъем клавиатуры персонального компьютера.*

Реализация подключения потребует изготовления специального переходного кабеля, входящего в состав лабораторной установки.

2.2 Подключение внешней памяти программ

Как уже отмечалось ранее, для организации обмена информацией между ПК и микроконтроллером целесообразно использовать в качестве внешней памяти программ ОЗУ. ОЗУ позволит быстро и многократно перепрограммировать микроконтроллер, что позволит использовать его как отладочный модуль, или как часто перепрограммируемое устройство управления (например, для управления различными лабораторными установками).

В качестве микросхемы ОЗУ можно использовать микросхему КР537РУ8. Объем памяти данной микросхемы составляет 2К. В связи с тем, что данная установка планируется как УМК в курсе «Микропроцессоры», то этого объема памяти вполне достаточно для размещения программы пользователя, т.к. программы, реализуемые в процессе лабораторных работ, как правило не превышают 100 команд. Объем таких программ не превышает 300 байт.

При записи программы пользователя в память программ, обращение к ОЗУ осуществляется как к памяти данных. Во время работы программы, как к памяти программ.



Рис. 3. Схема подключения внешней памяти программ к микроконтроллеру.

Подобная организация памяти программ позволит пользователю сделать программу более гибкой. Во время работы программы можно будет без особого труда изменять значение переменных, жестко – зашитых в программе.

2.3 Программное сопряжение микроконтроллера и ПК

2.3.1 Начальная установка MCS-51

Инициализация (сброс) микросхемы осуществляется подачей сигнала RST (высокий уровень напряжения) при подключенном кварце. Вход RST – внутренний триггер Шмидта. Состояние RST проверяется в фазе S5P2 каждого машинного цикла. Пока уровень RST высокий, ALE и PME тоже имеют высокий уровень. Их установка происходит после установки «0» на RST. Внутренний алгоритм при подаче сигнала «СБРОС» производит следующие действия:

* устанавливает счетчик команд и все регистры специальных функций, кроме портов – защелок Р0 – Р3, указателя стека и регистра SBUF в ноль;
* указатель стека принимает значение равное 07Н;
* запрещает все источники прерываний, работу таймеров – счетчиков и последовательного порта;
* выбирает БАНК 0 памяти данных, подготавливает порты Р0 – Р3 для приема данных и определяет выводы ALE и PME, как входы для внешней синхронизации;
* в регистрах специальных функций PCON, IP и IE резервные биты принимают случайные значения;
* устанавливает фиксаторы – защелки портов Р0 – Р3 в «1».

Сигнал RST не влияет на внутреннюю память данных микроконтроллера. После включения питания содержимое ячеек внутренней памяти данных принимает случайные значения.

2.3.2 Программное обеспечение организации обмена информацией между МК и ПК

2.3.2.1 Программа «Монитор»

Как уже отмечалось, программное обеспечение организации обмена информацией между микроконтроллером и ПК должно обеспечивать:

* запись программы пользователя в память программ МК;
* запись информации в программно – доступные узлы МК;
* чтение информации из памяти программ МК;
* чтение информации из программно – доступных узлов МК;
* запуск программы пользователя в режиме реального времени;
* запуск программы пользователя в пошаговом режиме.

Все эти функции выполняет программ «Монитор».

В начале работы программы «Монитор» происходит начальная установка. Эту установка должна произвести следующие действия:

* выбрать БАНК 2 памяти данных микроконтроллера, для хранения временных переменных, необходимых для работы программы «Монитор»;
* установить значение управляющего бита SMOD в регистре специальных функций в 1;
* выдать сообщение ПК о начале работы программы «Монитор».



### Рис. 4. Алгоритм программы «Монитор»

Программа «Монитор» принимает от ПК код выполняемой операции, декодирует его и запускает соответствующую подпрограмму.

Подпрограмма приема одного байта данных ожидает прихода информации, проверяет правильность данных и заносит принятый байт в регистр R0. Если данные были приняты не верно, то подпрограмма приема одного байта данных заносит в регистр R4 нулевое значение.

В случае не верно принятого кода операции микроконтроллер предлагает ПК повторно передать команду.

**

*Рис. 5. Алгоритм декодирования кода операции в программе «Монитор»*

После выполнения выбранной подпрограммы, программа «Монитор» снова переходит к режиму ожидания очередного кода операции.

Алгоритм выбора подпрограммы показан на рисунке 5.

В программе «Монитор» зарезервированы следующие коды операций:

* **01h - подпрограмма 1** - подпрограмма запуска программы пользователя в режиме реального времени;
* **02h - подпрограмма 2 -** подпрограмма запуска программы пользователя в пошаговом режиме;
* **03h - подпрограмма 3 -** подпрограмма записи программы пользователя в память программ МК;
* **04h - подпрограмма 4 -** подпрограмма записи информации в программно – доступные узлы МК;
* **05h - подпрограмма 5 -** подпрограмма чтения из памяти программ МК;
* **06h - подпрограмма 6 -** подпрограмма чтения программно – доступных узлов МК;
* **другие - подпрограмма 7 -** зарезервированные коды операций для запуска подпрограмм пользователя.

2.3.2.2 Подпрограмма запуска программы пользователя в режиме реального времени

Перед запуском программы пользователя, микроконтроллер принимает из ПК адрес начала программы и сохраняет его в регистре DPTR.

Для приема адреса используется подпрограмма приема двух байтов информации.

Алгоритм запуска программы пользователя в режиме реального времени представлен на рисунке 6.



Рис. 6. Алгоритм запуска программы пользователя в режиме реального времени.



Рис. 7. Алгоритм программы приема двух байтов информации.

Для приема байта информации, подпрограмма использует алгоритм приема одного байта данных. Ошибка приема проверяется по содержимому регистра R4.

Значение первого принятого байта сохраняется в регистре DPL.

Прием второго байта адреса программы пользователя из ПК аналогичен приему первого байта, с той лишь разницей, что его значение сохраняется в регистре DPH.

В случае возникновения ошибки подпрограмма завершает свою работу, оставляя без изменения содержимое регистра R4.

Подпрограмма запуска программы пользователя в режиме реального времени проверяет правильность принятия адреса по регистру R4. В случае возникновения ошибки передает в ПК код 0Fh. После чего возвращается в программу «Монитор»

Непосредственно перед запуском программы пользователя необходимо изменить рабочий БАНК памяти данных МК на БАНК 0.

Алгоритм программы приема двух байтов информации представлен на рисунке 7.

2.3.2.3 Подпрограмма запуска программы пользователя в пошаговом режиме

На этапе отладки прикладной программы очень удобным для разработчика оказывается режим пошагового (покомандного) исполнения программ. Этот режим можно организовать, используя систему прерываний микроконтроллера.

Внешний запрос прерывания не будет обслужен до тех пор, пока обслуживается прерывание с равным приоритетом. Этот запрос будет воспринят лишь после того, как будет выполнена одна команда после команды возврата из подпрограммы. Иными словами, однажды вызвав подпрограмму обслуживания прерывания, вызвать ее вновь невозможно до тех пор, пока хотя бы одна команда основной программы не будет исполнена.

Для использования этого свойства системы прерываний микроконтроллера можно запрограммировать одно из внешних прерываний (например, *INT0*) на обслуживание пошагового режима работы.



Рис. 8. Аппаратная реализация пошагового режима выполнения программы пользователя.

При реализации этого режима можно воспользоваться двумя способами:

* На вывод *INT0* подавать сигнал от кнопки «ШАГ», и микроконтроллер по сигналу *INT0=0* вызовет подпрограмму обслуживания внешнего прерывания.
* Предусмотреть возможность программного изменения состояния уровня сигнала на выводе *INT0*, используя один из свободных выводов порта Р2.

Второй способ является наиболее удобным, т.к. позволяет организовать пошаговый режим непосредственно на ПК. Кроме того первый способ требует больших материальных вложений для устранения дребезга контактов от кнопки «ШАГ».

После того, как адрес программы пользователя будет принят и проверен, подпрограмма выдаст активный сигнал на вход P3.2. По этому сигналу микроконтроллер перейдет в режим обработки прерывания *INT0*. До того, как будет выполнена первая команда в программе пользователя, основная программа должна выполнить две команды (выставить сигнал *INT0=0* и запустить программу пользователя). Поэтому в подпрограмме обработки прерывания необходим счетчик пропущенных шагов. Этот счетчик можно будет использовать для выполнения блока команд в режиме пошагового исполнения программы пользователя.

Алгоритм запуска программы пользователя в пошаговом режиме представлен на рисунке 9.



Рис. 9. Алгоритм выполнения программы пользователя в пошаговом режиме.

Подпрограмма обработки прерывания *INT0* аналогична программе «Монитор». В начале программы выполняется первоначальная установка:

* сохраняется содержимое аккумулятора, слова состояния программы и регистра DPTR в стеке;
* устанавливается БАНК 2, как активный банк памяти данных микроконтроллера;
* в регистр R4 записывается количество допустимых ошибок.

После выполнения первоначальной установки, микроконтроллер проверяет содержимое регистра R3. Этот регистр выполняет роль счетчика пропущенных шагов программы. Если счетчик не равен нулю, то подпрограмма восстанавливает содержимое аккумулятора, слова состояния программы, DPTR и выходит из режима обработки прерывания, для выполнения следующего шага программы пользователя.

Алгоритм программы обработки прерывания *INT0* показан на рисунке 10.



*Рис. 10. Алгоритм программы обработки прерывания INT0.*

Если счетчик был равен нулю, тогда микроконтроллер переходит в режим приема кода операции от ПК.

Алгоритм дешифрации кода операции в программе обработки прерывания *INT0* аналогичен алгоритму дешифрации кода операции в программе «Монитор». Здесь предусмотрены следующие коды операций:

* **01h -** чтение памяти программ МК;
* **02h -** запись информации в программно – доступные узлы МК;
* **03h -** запись в память программ МК;
* **04h -** выдача кода следующей команды;
* **05h -** выполнение блока команд программы пользователя;
* **06h -** выполнение программы в режиме реального времени;
* **07h -** останов программы;
* **08h -** выполнение следующего шага программы;
* **другие –** выполнение программ пользователя.

Большая часть подпрограмм, выбираемых пользователем в пошаговом режиме, аналогична подпрограммам, выбираемым в программе «Монитор».

Если значение регистра R5, при выходе из подпрограммы будет нулевым, то будет выполнен следующий шаг основной программы.

Для выполнения останова программы, достаточно перевести состояние сигнала на выходе P2.5 в нулевое состояние и в стеке подменить адрес следующей команды на адрес начала программы «Монитор».

Для того чтобы перевести программу пользователя из пошагового режима, в выполнение программы в режиме реального времени необходимо установить состояние сигнала P2.5 в нулевое состояние.

Алгоритм декодирования кода операции представлен на рисунке 11.



*Рис. 11. Алгоритм декодирования кода операции в пошаговом режиме работы программы пользователя.*

Для выдачи кода следующей команды необходимо считать 3 байта информации из памяти программ. Начальный адрес следующей команды можно взять из указателя стека.

2.3.2.4 Подпрограмма записи программы пользователя в память программ микроконтроллера.

В начале работы подпрограмма принимает количество передаваемых байтов из ПК. Для этого используется подпрограмма приема двух байтов информации. После ее выполнения данные необходимо перенести в счетчик количества принятых байтов. Функцию счетчика выполняют регистры R1 и R2.

Алгоритм записи программы пользователя в память программ микроконтроллера представлен на рисунке 12.



Рис. 12. Алгоритм подпрограммы записи программы пользователя в памяти программ микроконтроллера.

После того, как количество принимаемых байтов будет принято, необходимо принять адрес вершины принимаемого массива информации.

После этого становится возможным прием самой программы пользователя и запись ее в память программ микроконтроллера.

Алгоритм приема данных представлен на рисунке 13.



*Рис. 13. Алгоритм приема данных в подпрограмме записи программы пользователя в память МК.*

Адрес следующего байта программы содержится в регистре DPTR. После каждого принятого байта данных процедура проверяет правильность завершения подпрограммы приема одного байта данных, записывает его по адресу, указанному в регистре DPTR, увеличивает значение этого регистра и уменьшает счетчик принятых байтов.

После этого процедура проверяет счетчик принятых байтов программы пользователя и, в случае его обнуления, завершает процесс принятия данных.

2.3.2.5 Подпрограмма записи информации в программно – доступные узлы микроконтроллера

Подпрограмма записи информации в программно – доступные узлы микроконтроллера аналогична процедуре записи программы пользователя. Разница заключается в том, что для записи информации в программно – доступные узлы используется однобайтовая адресация, а при записи в память программ используется двухбайтовая адресация.

Алгоритм записи информации в программно – доступные узлы микроконтроллера представлен на рисунке 14.



*Рис. 14. Алгоритм записи информации в программно – доступные узлы микроконтроллера.*

Адрес принимаемых данных находится в регистре R1.

2.3.2.6 Подпрограмма чтения из памяти программ микроконтроллера

Память программ микроконтроллера имеет емкость в 2К байт. При чтении памяти программ микроконтроллера в ПК необходимо передать все 2К данных. Для обращения к памяти программ микроконтроллера, необходимо иметь два байта адреса.

В качестве указателя адреса ячейки памяти, подлежащей передачи в ПК, и счетчика передаваемых байтов данных используется регистр DPTR. Для пересылки байта информации в МК используется подпрограмма выдачи одного байта данных.

Для реализации этой процедуры необходимо, чтобы байт данных, подлежащий выдачи в ПК, находился в аккумуляторе.

Алгоритм подпрограммы чтения из памяти программ микроконтроллера представлен на рисунке 15.



*Рис. 15. Алгоритм подпрограммы чтения памяти программ микроконтроллера.*

После выдачи каждого байта информации, процедура проверяет правильность переданной информации.

Максимальный адрес памяти программ микроконтроллера – 07FFh.

2.3.2.7 Подпрограмма чтения информации программно – доступных узлов микроконтроллера

Данная подпрограмма аналогична процедуре чтения памяти программ микроконтроллера. Разница заключается в том, что для адресации к программно – доступным узлам МК необходим однобайтовый адрес, а для обращения к памяти программ – двухбайтовый.

Алгоритм подпрограммы чтения из программно – доступных узлов микроконтроллера представлен на рисунке 16.



Рис. 16. Алгоритм подпрограммы чтения из программно – доступных узлов микроконтроллера.

В качестве счетчика и указателя адреса передаваемых данных используется регистр R1. Максимально допустимый адрес - FFh.

2.3.2.8 Подпрограмма выдачи ошибки в ПК

В качестве кода ошибки выбран код 0Fh.

Процедура выдачи ошибки использует подпрограмму выдачи одного байта. Поэтому, перед запуском этой процедуры, код ошибки необходимо поместить в аккумулятор.

В связи с тем, что для сигнализации ошибки все процедуры используют регистр R4, перед запуском подпрограммы выдачи одного байта данных содержимое регистра R4 необходимо сохранить. Для этой цели используется регистр R7.

Перед завершением подпрограммы значение регистра R4 необходимо восстановить.

Алгоритм подпрограммы выдачи ошибки представлен на рисунке 17.

*Рис. 17. Алгоритм подпрограммы выдачи ошибки в ПК.*

2.3.2.9 Подпрограмма выдачи одного байта информации

Алгоритм подпрограммы выдачи одного байта информации в ПК представлен на рисунке 18.



*Рис. 18. Алгоритм подпрограммы выдачи одного байта информации в ПК.*

Подпрограмма передает данные находящиеся в аккумуляторе в последовательный порт вывода. В начале работы процедура переносит бит четности в буфер последовательного порта, задает значения счетчиков ошибок и ожиданий, и ожидает, пока буфер порта будет свободен. Как только последовательный порт будет готов к пересылке информации, программа начнет проверять готовность ПК к приему данных.

Если ПК будет не готов к приему информации долгое время, то процедура завершит свою работу с занесением в регистр R4 кода 00h, символизирующего ошибку передачи данных.

Как только ПК будет готов к приему данных подпрограмма передаст содержимое аккумулятора в последовательный порт. Подождет, пока порт не будет готов к передаче следующего байта данных и запустит подпрограмму задержки, для того, чтобы ПК успел проверить переданную информацию и, в случае ошибки, выдал бы запрос на повторную передачу данных.

Если ПК выставит сигнал ошибки переданной информации, подпрограмма будет пытаться передать эту информацию еще раз. Этот процесс будет повторяться, до тех пор, пока не обнулится счетчик ошибок.

Если счетчик ошибок будет равен нулю, процедура завершит свою работу с ошибкой.

В качестве сигнала готовности ПК, и сигнала ошибки переданной информации используется сигнал на входе P2.7.

2.3.2.10 Подпрограмма приема одного байта информации.

В начале работы, подпрограмма задает счетчик ошибок, счетчик тактов ожидания и разрешает передачу информации из ПК в микроконтроллер.

После этого подпрограмма ожидает приема информации, проверяет ее и, если данные приняты верно, сохраняет результат в регистре R0.

В случае возникновения ошибок, микроконтроллер передает в ПК сообщение об ошибке и предлагает повторить передачу.

Алгоритм подпрограммы приема одного байта информации представлен на рисунке 19.



*Рис. 19. Алгоритм подпрограммы приема одного байта информации.*

В случае многократного повторения ошибок, подпрограмма завершает свою работу и выставляет в регистре R4 код 00h, символизирующий ошибочность принятых данных.

Если от ПК долгое время не будет передана информация, подпрограмма также завершит свою работу и выставит в регистре R4 код 00h, символизирующий ошибочность принятых данных.

В качестве счетчика времени ожидания используется регистр R6.

ЗАКЛЮЧЕНИЕ

В работе была рассмотрена организация обмена информацией между персональным компьютером и микроконтроллером семейства MCS-51 фирмы Intel. Была разработана, на уровне структурной схемы, аппаратная часть сопряжения микроконтроллера с портами персонального компьютера, а также схема подключения внешней памяти программ к микроконтроллеру. Разработано, на уровне алгоритмов, программное обеспечение микроконтроллера.

На основе проделанной работы можно создать установку, позволяющую автоматизировать процесс программирования микроконтроллерной системы. Данную установку можно применять в качестве лабораторной установки для изучения курса «Микропроцессоры» и в качестве программируемого управляющего элемента более сложной системы.

СПИСОК ИСПОЛЬЗОВАНОЙ ЛИТЕРАТУРЫ

1. «Однокристальные микроЭВМ» справочник / Бобрыкин А.В., Липовецкий Г.П. и др. / Москва / 1994 г.
2. «Проектирование цифровых устройств на однокристальных микроконтроллерах» / Сташин В.В., Урусов А.В., Мологонцева О.Ф. / Москва, «Энергоатомиздат» / 1990 г.
3. «Лабораторная установка на базе микроконтроллера MCS-51 по курсу микропроцессоры и микропроцессорные системы» работа на соискание академической степени бакалавра / Баринов Р. / Рига, РАУ / 1998 г.
4. «1996 IC Master for Windows» / компакт диск / © Copyright Heart business puplishinc.
5. «STANDART IBM PC. Устройство, установка, техническое обслуживание и ремонт персонального компьютера» справочник / Кишинев / 1991 г.
6. «Модемы и факс-модемы. Программирование для MS-DOS и Windows» / А.В. Фролов, Г.В. Фролов / Москва / «Диалог-Мифи» / 1995 г.

ПРИЛОЖЕНИЯ

#### Приложение 1

##### Точные значения кварцев

|  |  |  |  |
| --- | --- | --- | --- |
| Кратность | Скорость передачи  (Кбод) | Частота кварца (МГц) | |
| SMOD=0 (1/64) | SMOD=1 (1/32) |
| 1 | 115,2 | 7,3728 | 3,6864 |
| 2 | 57,6 | 3,6864 | 1,8432 |
| 3 | 38,4 | 2,4576 | 1,2288 |
| 4 | 28,8 | 1,8432 | 0,9216 |
| 5 | 23,04 | 1,4746 | 0,73728 |
| 6 | 19,2 | 1,2288 | 0,6144 |
| 7 | 16,457142 | 1,053257 | 0,526628 |
| 8 | 14,4 | 0,9216 | 0,4608 |
| 9 | 12,8 | 0,8192 | 0,4096 |
| 10 | 11,52 | 0,73728 | 0,36864 |
| 12 | 9,2 | 0,6144 | 0,3072 |

**Приложение 2**

###### Возможные значения кварцев

SMOD=0

|  |  |  |  |
| --- | --- | --- | --- |
| Кратность | Скорость передачи  (Кбод) | Частота кварца (МГц) | |
| SMOD=0 (1/64) | SMOD=1 (1/32) |
| 1 | 115,2 | 7,366503 | 7,378725 |
| 2 | 57,6 | 3,673807 | 3,698251 |
| 3 | 38,4 | 2,438711 | 2,475377 |
| 4 | 28,8 | 1,818014 | 1,866903 |
| 5 | 23,04 | 1,443078 | 1,504189 |
| 6 | 19,2 | 1,191022 | 1,264355 |
| 7 | 16,457142 | 1,009183 | 1,094738 |
| 8 | 14,4 | 0,871229 | 0,969007 |
| 9 | 12,8 | 0,762533 | 0,872533 |
| 10 | 11,52 | 0,674317 | 0,796539 |
| 12 | 9,2 | 0,538844 | 0,685511 |

SMOD=1

|  |  |  |  |
| --- | --- | --- | --- |
| Кратность | Скорость передачи  (Кбод) | Частота кварца (МГц) | |
| SMOD=0 (1/64) | SMOD=1 (1/32) |
| 1 | 115,2 | 3,683252 | 3,689363 |
| 2 | 57,6 | 1,836904 | 1,849126 |
| 3 | 38,4 | 1,219356 | 1,237689 |
| 4 | 28,8 | 0,909007 | 0,933452 |
| 5 | 23,04 | 0,721539 | 0,752095 |
| 6 | 19,2 | 0,595511 | 0,632178 |
| 7 | 16,457142 | 0,504592 | 0,547369 |
| 8 | 14,4 | 0,435615 | 0,484504 |
| 9 | 12,8 | 0,381267 | 0,436267 |
| 10 | 11,52 | 0,337159 | 0,398270 |
| 12 | 9,2 | 0,269422 | 0,342756 |