

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ПРОВЕДЕНИЮ**

**ЛАБОРАТОРНЫХ ЗАНЯТИЙ**

**по УЧЕБНОЙ ДИСЦИПЛИНЫ**

**ОП.012.Микропроцессорные системы управления**

специальность **090305 Информационная безопасность**

**автоматизированных систем**

( программа базовой подготовки)

Москва

2016

|  |
| --- |
| РЕКОМЕНДОВАНО  ПМЦК специальности 090305 Информационная безопасность автоматизированных систем  Протокол № от « » 2016 г.  Председатель ПМЦК  \_\_\_\_\_\_\_\_\_\_\_ А .А.Юмаева  « » \_\_\_\_\_\_ 2016 г. |

Составитель: Лобанова Н.Г., методист,преподаватель профессионального цикла ГБПОУ г.Москвы «Колледж связи №54»

*Ф.И.О., должность*

Рецензенты:

**Содержание**

**Стр.**

1.Общие положения**……………………………………………………………**.3

2.Методика и средства выполнения лабораторных работ…………………..3

3.Этапы выполнения лабораторных работ…………………………………...4

4.Тематика лабораторных работ и задания к ним……………………………8

5.Учебно - методическое и информационное обеспечение дисциплины…..84

# Общие положения

Целью выполнения лабораторных работ по дисциплине **ОП.12.Микропроцессорные системы управления** является:

* закрепление теоретических сведений, полученных на занятиях;
* получение навыков исследования микропроцессорных систем;
* повышение информационной культуры в решении профессиональных задач будущего специалиста

В результате освоения учебной дисциплины обучающийся должен

**уметь:**

- работать с микропроцессорными системами управления

**знать:**

- виды микропроцессорных систем и способы организации обмена информацией в микропроцессорных системах;

- аппаратно-алгоритмические принципы построения микропроцессорных систем управления (МПСУ).

- принципы функционирования микропроцессорных систем управления

# Методика и средства выполнения лабораторных работ

Методика выполнения лабораторной работы определяется моделью задачи, решаемой студентом на занятии по заданию преподавателя.

Средствами проведения лабораторных работ являются:

* Комплект персональных ЭВМ в лаборатории.
* Лабораторные стенды

**3.Этапы выполнения лабораторных работ**

На первом лабораторном занятии до студентов доводится общий порядок выполнения лабораторных работ.

Преподаватель доводит до студентов под роспись правила техники электро - и пожарной безопасности при выполнении лабораторных работ.

Преподаватель знакомит студентов с лабораторным стендом PU -2000 для выполнения исследований по эксклюзивной методике Degem Systems Ltd.

Это оборудование включает:

* лабораторный стенд PU -2000;
* печатную плату;
* двухканальный осциллограф.

*Лабораторный стенд PU -2000 содержи*т: направляющие и разъем для установки печатной платы, генератор сигналов, работающий в режиме трех сигналов: прямоугольной, синусоидальной и треугольной формы, цифровой мультиметр и два источника питания.

На рабочем месте имеется персональный компьютер.

После ознакомления с лабораторным оборудованием преподаватель проводит постановку задачи конкретного лабораторного занятия, содержание и объем работ, предусмотренных конкретной лабораторной работы.

**Формулируется цели, задачи**, основные этапы работы, последовательность и ход выполнения работы.

Определяются содержание и форма представления результатов работы. Определяются содержание и форма представления результатов работы.

Каждая лабораторная работа студента должна быть оформлена в виде отчет о лабораторной работе.

Поясняется методика составления и оформления отчета о лабораторной работе.

**Ознакомление студента с содержанием и объемом лабораторной работы.**

На этом этапе студент должен тщательно изучить содержание и объем предстоящей лабораторной работы. Если постановка задачи недостаточно ясна, он может обратиться к преподавателю за дополнительными разъяснениями. Затем студент приступает к выполнению задания лабораторной работы

**Порядок выполнения лабораторных работ**

В соответствии с установленной последовательностью этапов работы студент выполняет объем работ, предусмотренных заданием лабораторной работы.

* В ходе выполнения этапов лабораторной работы студент регистрирует полученные результаты в тетради для лабораторных работ.
* После выполнения всех заданий лабораторной работы студент анализирует полученные результаты, делает выводы и предъявляет преподавателю предварительные результаты работы (выведенные на экран монитора или записанные в тетради для лабораторных работ).
* При получении от преподавателя замечаний студент принимает меры к их устранению и затем снова предъявляет результаты преподавателю для контроля.
* Если замечаний нет, то студент приступает к оформлению отчета о лабораторной работе.

В случае замеченных ошибок студент принимает меры к их исправлению и затем снова предъявляет результаты преподавателю для контроля и приема результатов работы. Если в работе ошибок не содержится, то приступает к составлению и оформлению отчета о лабораторной работе

**4.Регистрация результатов и оформление отчета о лабораторной работе**

Оформление отчета выполнить по следующим правилам.

Отчет о лабораторной работе должен содержать следующие обязательные разделы:

* Номер лабораторной работы
* Тема, цель и задачи работы;
* Основные этапы лабораторной работы.

По каждому из этапов приводится содержание выполненных работ и полученные результаты, методика, средства выполнения, схемы, графики, диаграммы,таблицы ( если предусмотрено в описании и требованиями к результатам выполнения лабораторной работы);

* Выводы по работе. Выводы излагаются последовательно по каждому из этапов работы. Формулируются в сжатой и четкой форме.

Текст отчета должен быть изложен лаконично и вместе с тем информативно с соблюдением правил грамматики. В конце отчета может быть указана литература, которую студент применил в лабораторной работе. Библиографические описания литературных источников должны быть оформлены в соответствии с ГОСТ 7.1-84. Правила библиографического описания документации.

**5.Заключительная часть лабораторной работы**

После окончания составления отчета студент проверяет его правильность и устраняет ошибки и предъявляет его преподавателю.

Преподаватель проверяет отчет. При обнаружении ошибок в его оформлении возвращает студенту для доработки. После этого студент исправляет ошибки и повторно предъявляет отчет преподавателю.

При отсутствии ошибок в оформлении отчета преподаватель задает студенту ряд контрольных вопросов. Если студент правильно отвечает на поставленные вопросы, преподаватель принимает его отчет и выставляет оценку за выполнение лабораторной работы.

После успешной сдачи отчета по лабораторной работе студент выключает лабораторное оборудование и наводит порядок на рабочем месте.

**Критерии оценки при выполнении лабораторной работы**

Оценка за работу студенту выставляется с учетом предварительной подготовки к работе, доли самостоятельности при ее выполнении, точности и грамотности оформления отчета по работе

*Критерии оценки при выполнении лабораторной работы*

* Оценка «5» ставится: лабораторная работа выполнена в полном объеме, в соответствии с заданием, с соблюдением последовательности выполнения, , самостоятельно; работа оформлена аккуратно.
* Оценка «4» ставится: лабораторная работа выполнена в полном объеме, в соответствии с заданием, с соблюдением последовательности выполнения, частично с помощью преподавателя, присутствуют незначительные ошибки в действиях; работа оформлена аккуратно.
* Оценка «3» ставится: лабораторная работа выполнена в полном объеме, в соответствии с заданием, частично с помощью преподавателя, присутствуют ошибки при выполнении работы; по оформлению работы имеются замечания.
* Оценка «2» ставится: обучающийся не подготовился к лабораторной работе, при выполнении работы допустил грубые ошибки, по оформлению работы имеются множественные замечания.

**4.Тематика лабораторных работ и задания к ним**

**Лабораторная работа №1.** Исследовние основных компонентов микропроцессора

**Цель занятия**:

**- познакомиться** с основными частями микропроцессора EB-151 и их функциями;

с различными шинами и их функциями;

**- освоить**  процедуру запуска обычной программы.

**Продолжительность занятия** - 2 часа.

**Пояснения к работе**

*Краткие теоретические сведения*

**Микропроцессор —** процессор (устройство, отвечающее за выполнение арифметических, логических операций и операций управления, записанных в машинном коде), реализованный в виде одной микросхемы или комплекта из нескольких специализированных микросхем.

Структура однокристального МП приведена на рис.



Рис. 1. Структура однокристального МП

Доступные блоки МП :

регистр-аккумулятор

* Счетчик команд,
* блок регистров B, C, D, E,
* регистр признаков.

Недоступные блоки МП:

* регистр адреса,
* схема управления,
* арифметико-логическое устройство,
* блок регистров временного хранения данных,
* регистр команд.

Основными блоками МП являются: блок регистров общего назначения

(РОН) со схемой выборки регистров; регистр комад с дешифратором команди формирователем машинных циклов; арифметико - логическое устройство с

регистром-аккумулятором, выполняющим арифметические и логические

операции; регистры временного хранения данных W и Z; флаговый регистр;

устройство управления и синхронизации.

Микросхема процессора обязательно имеет **выводы трех шин: шины адреса, шины данных и шины управления.** Иногда некоторые сигналы и шины мультиплексируются, чтобы уменьшить количество выводов микросхемы процессора

**Разрядность шины данных определяет скорость работы системы**. Разрядность шины адреса определяет допустимую сложность системы. Количество линий управления определяет разнообразие режимов обмена и эффективность обмена процессора с другими устройствами системы.

Все устройства микропроцессорной системы объединяются общей системной шиной (она же называется еще **системной магистралью** или **каналом** ). Системная магистраль включает в себя четыре основные шины нижнего уровня:

* шина адреса (Address Bus);
* шина данных (Data Bus);
* шина управления (Control Bus);
* шина питания (Power Bus).

**Шина адреса служит для определения адреса (номера) устройства, с которым процессор обменивается информацией в данный момент. Каждому устройству (кроме процессора), каждой ячейке памяти в микропроцессорной системе присваивается собственный адрес**. Когда код какого-то адреса выставляется процессором на шине адреса, устройство, которому этот адрес приписан, понимает, что ему предстоит обмен информацией. *Шина адреса может быть однонаправленной или двунаправленной.*

**Шина данных — это основная шина, которая используется для передачи информационных кодов между всеми устройствами микропроцессорной системы**. Обычно в пересылке информации участвует процессор, который передает код данных в какое-то устройство или в ячейку памяти или же принимает код данных из какого-то устройства или из ячейки памяти. Но возможна также и передача информации между устройствами без участия процессора. *Шина данных всегда двунаправленная.*

**Шина управления в отличие от шины адреса и шины данных** состоит из отдельных управляющих сигналов. Каждый из этих сигналов во время обмена информацией имеет свою функцию. Некоторые сигналы служат для стробирования передаваемых или принимаемых данных (то есть определяют моменты времени, когда информационный код выставлен на шину данных). Другие управляющие сигналы могут использоваться для подтверждения приема данных, для сброса всех устройств в исходное состояние, для тактирования всех устройств и т.д. *Линии шины управления могут быть однонаправленными или двунаправленными.*

**Наконец**, **шина питания** предназначена не для пересылки информационных сигналов, а для питания системы. Она состоит из линий питания и общего провода. В микропроцессорной системе может быть один источник питания (чаще +5 В) или несколько источников питания (обычно еще –5 В, +12 В и –12 В). Каждому напряжению питания соответствует своя линия связи. *Все устройства подключены к этим линиям параллельно.*

*Если в микропроцессорную систему надо ввести входной код (или входной сигнал), то процессор по шине адреса обращается к нужному устройству ввода/вывода и принимает по шине данных входную информацию.*

*Если из микропроцессорной системы надо вывести выходной код (или выходной сигнал*), то процессор обращается по шине адреса к нужному устройству ввода/вывода и передает ему по шине данных выходную информацию.

*Если информация должна пройти сложную многоступенчатую* обработку, то процессор может хранить промежуточные результаты в системной оперативной памяти. Для обращения к любой ячейке памяти процессор выставляет ее адрес на шину адреса и передает в нее информационный код по шине данных или же принимает из нее информационный код по шине данных. В памяти (оперативной и постоянной) находятся также и управляющие коды (команды выполняемой процессором программы), которые процессор также читает по шине данных с адресацией по шине адреса. Постоянная память используется в основном для хранения программы начального пуска микропроцессорной системы, которая выполняется каждый раз после включения питания. Информация в нее заносится изготовителем раз и навсегда.

*Таким образом, в микропроцессорной системе все информационные коды и коды команд передаются по шинам последовательно, по очереди. Это определяет сравнительно невысокое быстродействие микропроцессорной системы. Оно ограничено обычно даже не быстродействием процессора (которое тоже очень важно) и не скоростью обмена по системной шине (магистрали), а именно последовательным характером передачи информации по системной шине (магистрали).*

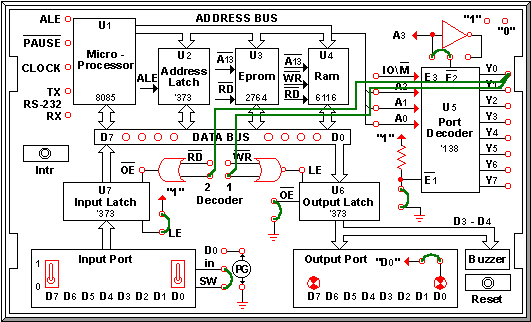
**Оборудование, приборы и элементы:**

* Лабораторный стенд PU-2000
* Плата с печатной схемой EB-151
* Двухканальный осциллограф
* Набор соединительных проводов

**Задание**1.Разработайте обобщенную структурную схему микропроцессора платы EB-151, изученив основные части микропроцессора этой платы

**Порядок выполнения задания**

* 1. Разместите печатную плату EB-151 на стенде PU-2000 и вставьте ее в разъем.
  2. Изучите схему.
  3. Подключите схему, как показано на рисунке:



4.На плате EB-151 найдите микропроцессор 8085.

Это Центральный Процессор, или ЦП платы EB-151, он осуществляет всю передачу данных, арифметические операции, синхронизацию и управление схемами.

5.Найдите на плате кварцевый кристалл (кварц), заключенный в металлический корпус. Этот прибор устанавливает тактовую частоту ЦП.

6.Память находиться в двух микросхемах.

а) 2764 EPROM (Erasable-Programmable Read-Only Memory - стираемое программируемое постоянное запоминающее устройство - СППЗУ), используемое для хранения программы.

б) 6116 RAM (Random Access Memory - оперативное запоминающее устройство - ОЗУ), используемое для хранения данных.

Найдите эти два компонента на плате EB-151.

7.Обратите внимание на две шины, которые соединяют ЦП с другими схемами. Шина передачи данных (data bus) - это двунаправленное 8-контактное соединение для передачи данных между ЦП и памятью, либо схемами ввода-вывода (I/O).

Направление потока данных и отправителя/ получателя данных контролируются ЦП при помощи сигналов управления READ (чтение), WRITE (запись), INPUT-OUTPUT/MEMORY (ввод-вывод/памяти), а также при помощи адресной шины (address bus).

8.Адресная шина - это 16-контактное соединение ЦП с памятью и схемами ввода-вывода. 40 контактный корпус микросхемы процессора 8085 не имеет достаточного количества контактов, требуемых для 16-ти разрядной адресной шины. Он делит контакты шины данных с младшей частью адресной шины.

В первой части каждого цикла, контакты содержат адресные данные, а сигнал управления ALE (Address Latch Enable - включение адресного триггера-защелки) устанавливается в "1", чтобы показать присутствие адреса на шине.

В плате EB-151 адресная информация хранится в триггере-защелке адреса (address latch) 74LS373.

Во время второй части цикла, шина передачи данных используется только для передачи данных.

Найдите адресную шину и ее защелку.

**Задание 2. Осуществите** процедуру запуска обычной программы

**Порядок выполнения задания**

1.Для запуска программы на плате EВ-151 нужно произвести следующие действия:

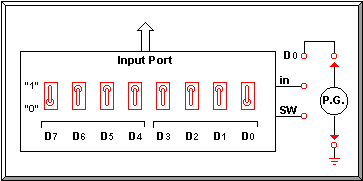
a. Установите номер программы (в двоичном коде) на переключателях порта ввода D6-D0 .

b. Установите переключатель порта ввода D7 в положение "1"

c. Нажмите кнопку RESET (сброс).

d. Для запуска программы переставить переключатель порта ввода D7 в положение "0".

2.Теперь запустите программу №1. Установите переключатели D6-D0 в положение 0000001 и переключатель D7 в положение 1, как показано на рисунке.



11.Выполните сброс микропроцессора.

12.Передвиньте переключатель D7 в положение "0".

13.Программа, которая сейчас запущена, повторяет положение переключателей D7-D0 на выходных светодиодах (LED) D7-D0.

Проверьте это, поменяв положение нескольких переключателей.

Вы изучили основные компоненты микрокомпьютера: ЦП (микропроцессор), СППЗУ, ОЗУ, формирователь тактовых сигналов, шина передачи данных и ее команды, адресная шина и схема ввода-вывода.

●**Оформите отчет.**

**Представьте в отчете** результат разработки обобщенной структурной схемы микропроцессора платы EB-151.

**Сделайте выводы по работе.**

Ответьте на контрольные вопросы

**???Контрольные вопросы**

1. **Сколько контактов имеет микропроцессор 8085?**

* 20 контактов.
* 30 контактов.
* 60 контактов.
* 40 контактов.

1. **Выберите правильное утверждение**:

* 6116 использует только сигналы READ(чтение).
* 6116 и 2764 использует только сигналы READ(чтение).
* 6116 использует сигналы READ(чтение) и WRITE(запись), а 2764 использует только сигналы READ(чтение).
* 6116 использует сигналы READ(чтение) и WRITE(запись), а 2764 использует только сигналы WRITE(запись).

3. **Сколько контактов формируют шину передачи данных?**

* **8**
* 4
* 2
* 16

4.**Адресная шина имеет 16 контактов. Сколько различных адресов может быть выбрано адресной шиной**?

* 16
* 216 = 65 536
* 1616 = 1.8446\*1019
* 162 = 256

**Лабораторная работа** №2.

Проведение анализа архитектуры современных микроконтроллеров на примере микроконтроллера *AVR* фирмы *Atmel*

**Цель занятия.** Получение навыков проведения анализа архитектур современных микроконороллеров

**Продолжительность занятия** - 2 часа.

***Пояснения к работе***

*Краткие теоретические сведения*

**Микроконтроллеры *AVR* фирмы Atmel**

Первые микроконтроллеры под торговой маркой ***AVR*** фирма **Atmel**

выпустила в 1997 г.

Концепция скоростных микроконтроллеров была разработана в научно-исследовательском центре ***ATMEL*** в Норвегии с учетом достигнутого к тому времени мирового опыта и технических решений в построении микропроцессорных систем.

Архитектура, на основе которой построены микроконтроллеры семейства *AVR*, представляет собой мощный гарвардский *RISC* процессор с раздельным доступом к памяти программ и данных.

Особенностью ***гарвардской*** архитектуры является использование

раздельных адресных пространств для хранения команд и данных, как показано на рис.1.

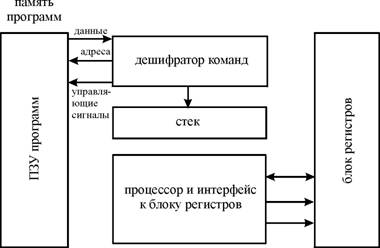


Рис. 1. Структурная схема гарвардской архитектуры

**Фоннеймановская и гарвардская архитектура**

       В 1945 г. американский математик Джон фон Нейман сформулировал основные принципы работы современных компьютеров. Им была предложена архитектура, получившая его имя (von Neumann architecture) и предполагающая хранение программ и данных в общей памяти (1946 г.). Сегодня такая архитектура наиболее характерна для микропроцессоров, ориентированных на использование в компьютерах. Примером могут служить микропроцессоры семейства х86.  
      Архитектура, предполагающая раздельное использование памяти программ и данных, носит название гарвардской (Harvard architecture). Гарвардская архитектура позволяет центральному процессору работать одновременно как с памятью программ, так и с памятью данных, что существенно увеличивает производительность.

Гарвардская архитектура *AVR* реализует полное логическое и физическое разделение не только адресных пространств, но и информационных шин для обращения к ОЗУ и ПЗУ. Такое построение уже ближе к структуре цифровых сигнальных процессоров и обеспечивает существенное повышение производительности.

Отличительной чертой архитектуры *AVR* является регистровый файл быстрого доступа, содержащий 32-х байтовых регистров общего назначения, каждый из которых может работать как регистр-аккумулятор, поскольку все они непосредственно подключены к АЛУ.

Шесть регистров этого файла могут использоваться как три 16-разрядных указателя адреса при косвенной адресации данных, что существенно повышает скорость пересылки данных при работе с прикладными программами.\_

**Продолжительность занятия** - 2 часа.

Принадлежности

1.Раздаточнвй материал

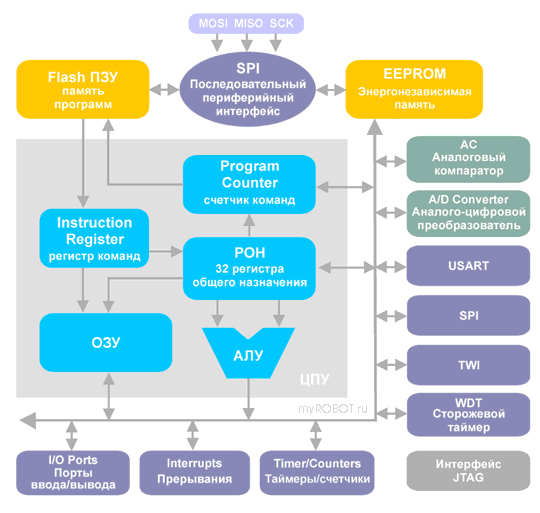
**Задание**

Проведите анализ архитектуры современных микроконтроллеров на примере микроконтроллера *AVR* фирмы *Atmel*

**Порядок работы**

1. Прочитайте информацию о микроконтроллерах *AVR* фирмы *Atmel*
2. Рассмотрите и изучите структурная схема AVR микроконтроллера
3. Ответьте письменно на вопросы (см. в конце работы). Номера вопросов дает преподаватель
4. Приведите примеры использования микроконтроллера *AVR* фирмы *Atmel* вбытовой технике и в системах обработки данных

Микроконтроллер AVR содержит: быстрый RISC-процессор, два типа энергонезависимой памяти (Flash-память программ и память данных EEPROM), оперативную память RAM, порты ввода/вывода и различные периферийные интерфейсные схемы.



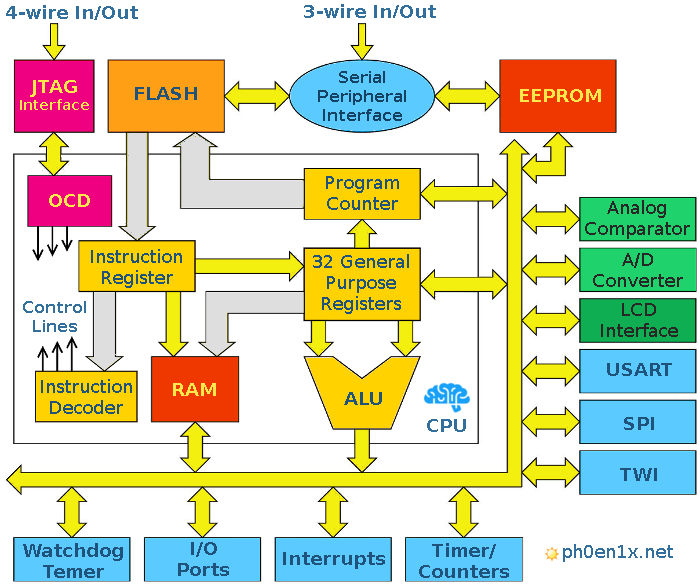


Рис.2. Структурная схема AVR микроконтроллера

Рассмотрим кратко, что изображено на блоках в схеме:

* **JTAG Interface** (Joint Test Action Group Interface) - интерфейс внутрисхемной отладки (4 провода);
* **FLASH** - перепрограммируемая память для сохранения программы;
* **Serial Peripheral Interface, SPI** - последовательный периферийный интерфейс (3 провода);
* **EEPROM** (Electrically Erasable Programmable Read-Only Memory) - перепрограммируемое ПЗУ, энергонезависимая память;
* **CPU** (ЦПУ) - центральный процессор управления, сердце микроконтроллера, 8-битное микропроцессорное ядро;
* **ALU** (АЛУ) - арифметико-логическое устройство, основа блока CPU;
* **RAM** (Random Access Memory) - оперативная память процессора;
* **Program Counter** -  счетчик команд;
* **32 General Purpose Registers** - 32 регистра общего назначения;
* **Instruction Register** - регистр команд, инструкций;
* **Instruction Decoder** - декодер команд;
* **OCD** (On-Chip Debugger) - блок внутренней отладки;
* **Analog Comparator** - аналоговый компаратор, блок сравнения аналоговых сигналов;
* **A/D Converter** (Analog/Digital converter) - аналогово-цифровой преобразователь;
* **LCD Interface** (Liquid-Crystal Display Interface) - интерфейс для подключения жидко-кристаллического дисплея, индикатора;
* **USART** (Universal Asynchronous Receiver-Transmitter), UART - универсальный асинхронный приемопередатчик;
* **TWI** (Two-Wire serial Interface) - последовательный интерфейс с двухпроводным подключением;
* **Watchdog Timer** - сторожевой или контрольный таймер;
* **I/O Ports** - порты вода/вывода;
* **Interrupts** - блок управления и реакции на прерывания;
* **Timers/Counters** - модули таймеров и счетчиков.

Сердцем микроконтроллеров AVR является 8-битное микропроцессорное ядро или центральное процессорное устройство (ЦПУ), построенное на принципах RISC-архитектуры. Основой этого блока служит арифметико-логическое устройство (АЛУ). По системномутактовому сигналу из памяти программ в соответствии с содержимым счетчика команд (Program Counter - PC) выбирается очередная команда и выполняется АЛУ. Во времявыбора команды из памяти программ происходит выполнение предыдущей выбранной команды, что и позволяет достичь быстродействия 1 MIPS на 1 МГц.   
**CISC и RISC**

     По числу команд микропроцессоры подразделяют на CISC (Complex Instruction Set Computer) и RISC (Reduced Instruction Set Computer). Термин CISC обозначает сложную систему команд, RISC - сокращенную.   
     Идея RISC - это тщательный подбор команд, которые можно было бы выполнить за один такт,т. о. упрощается аппаратная реализация процессора, сокращается число транзисторов, снижается потребляемая мощность и цена.  
     Очевидно, что в общем случае одной CISC-команде должны соответствовать несколько RISC-команд. Однако обычно выигрыш в быстродействии у RISC перекрывает потери. Так, самая быстрая команда у 8051 выполняется за 12 тактов. Даже если для каждой CISC-инструкции потребуется выполнить три RISC-инструкции, то в итоге RISC-архитектура будет в 4 раза производительней.   
      В настоящее время грань между RISC и CISC стирается. Например, AVR имеют 133 команды, что соответствует CISC, но большинство из них выполняется за один такт, что является признаком RISC. Поэтому основным признаком RISC стало принято считать выполнение команд за один такт.

АЛУ подключено к регистрам общего назначения РОН (General Purpose Registers - GPR). Регистров общего назначения всего 32, они имеют байтовый формат, то есть каждый из них состоит из восьми бит. РОН находятся в начале адресного пространства оперативной памяти, но физически не являются ее частью. Поэтому к ним можно обращаться двумя способами (как к регистрам и как к памяти). Такое решение является особенностью AVR и повышает эффективность работы и производительность микроконтроллера.  
    Отличие между регистрами и оперативной памятью состоит в том, что с регистрами можно производить любые операции (арифметические, логические, битовые), а в оперативную память можно лишь записывать данные из регистров.

**Память программ (Flash ROM или Flash ПЗУ)**   
    Память программ предназначена для хранения последовательности команд, управляющих функционированием микроконтроллера, и имеет 16-ти битную организацию. Все AVR имеют Flash-память программ, которая может быть различного размера - от 1 до 256 КБайт. Ее главное достоинство в том, что она построена на принципе электрической перепрограммируемости, т. е. допускает многократное стирание и запись информации.

**Память данных**   
    Память данных разделена на три части: регистровая память, оперативная память (ОЗУ - оперативное запоминающее устройство или RAM) и энергонезависимая память (ЭСППЗУ или EEPROM).

**Оперативная память (ОЗУ или RAM)**   
    Внутренняя оперативная статическая память Static RAM (SRAM) имеет байтовый формат и используется для оперативного хранения данных.  
    Размер оперативной памяти может варьироваться у различных чипов от 64 Байт до 4 КБайт. Число циклов чтения и записи в RAM не ограничено, но при отключении питающего напряжения вся информация теряется.   
    Для некоторых микроконтроллеров возможна организация подключения внешнего статического ОЗУ объемом до 64К.  
 **Периферия микроконтроллеров AVR** включает: порты (от 3 до 48 линий ввода и вывода), поддержку внешних прерываний, таймеры-счетчики, сторожевой таймер, аналоговые компараторы, 10-разрядный 8-канальный АЦП, интерфейсы UART, JTAG и SPI, устройство сброса по понижению питания, широтно-импульсные модуляторы.

●**Оформите отчет.**

**Представьте в отчете** письменные ответы на контрольные воросы( номера вопросов указывает преподаватель)

**Сделайте выводы по работе.**

**???Контрольные вопросы**

1. Дайте определение микроконтроллера

1. Организация ядра AVR-контроллеров.

2. Программная модель AVR-микроконтроллеров.

3. Периферийные устройства AVR.

4. Порты ввода/вывода.

5. Таймеры /счетчики.

6. Регистры специального назначения.

7. Универсальный асинхронный приемопередатчик (UART).

8. Аналого-цифровой преобразователь (ADC)

9.Назовите и охарактеризуйте четыре типа информационных объектов, имеющими возможность оперировать с арифметико-логическим устройством микроконтроллера

10. Какие операции могут быть выполнены без участия аккумулятора?

**Лабораторная работа №3.**

Проведение исследования и анализа процесса управление обменом в МПС

параллельным способом

**Цель занятия:**

* изучение процесса управления обменом в МПС параллельным способом;
* получить умения в проведении исследования и анализе процесса управления обменом в МПС параллельным способом

***Продолжительность занятия*** *- 2 часа*

**Пояснения к работе**

*Краткие теоретические сведения*

Передача данных внутри МПС осуществляется в параллельном формате. Обмен данными между МПС и внешними устройствами возможен как в параллельном, так и в последовательном формате. Исходя из этого, *УВВ* подразделяются на:

-УВВ, осущесттвляющие обмен с внешними устройствами в параллельном формате;

- УВВ, осущесттвляющие обмен с внешними устройствами в последовательном формате.

В этом случае имеет место преобразование данных из параллельного формата, используемого внутри МПС, в последовательный формат.

УВВ, обеспечивающие обмен с внешними устройствами в параллельном формате носит название *параллельного программируемого интерфейса*, иногда также называемого параллельным портом. Архитектура его представлена на рис. 1.

**Принадлежности и элементы:**

* Структурная схема архитектуры параллельного программируемого интерфейса
* Краткое описание параллельного программируемого интерфейса МПС

**Задание**

Изучите теоретические сведения об архитектуре и работе параллельного программируемого интерфейса МПС

**Порядок выполнения работы**

1.Зарисуйте архитектуру параллельного программируемого интерфейса. Распишите назначение блоков

2.Зарисуйте временные диаграммы записи данных в параллельный порт.Дайте краткие пояснения к временным диаграммам

3.Зарисуйте временные диаграммы чтения данных из порта. Дайте пояснения к временным диаграммам

|  |
| --- |
| ?n=1 |

Рис. 1.  Параллельный программируемый интерфейс

1. Буфер шины данных — усилитель, предназначенный для подключения к [шине данных](javascript:termInfo(%22Шина%20данных%22)) [МПС](javascript:termInfo(%22МПС%22)).
2. Устройство управления управляет работой параллельного порта. К устройству управления подходят управляющие сигналы RD, WR, CS.
3. Порт представляет собой регистр и усилитель, обеспечивающий подключение к разрядам регистра внешних линий связи. При работе порта в режиме вывода данных, информация, содержащаяся в регистре, выдается на линии порта в виде соответствующих напряжений. При работе порта в режиме ввода данных, внешнее устройство устанавливает на линиях порта напряжения, соответствующие передаваемым данным, при этом в регистре порта устанавливаются значения, соответствующие этим напряжениям, т.е. передаваемые данные.
4. Регистр управления содержит данные, определяющие режим работы параллельного интерфейса (например, направление передачи данных через порт — ввод или вывод). Настройка интерфейса на требуемый режим работы (программирование) осуществляется путем записи соответствующего значения в регистр управления. Эта запись выполняется микропроцессором перед началом работы с УВВ.
5. Адресные линии (A0, A1) необходимы для выбора внутреннего регистра, в который будет происходить запись (чтение) информации при выполнении цикла обмена с [МП](javascript:termInfo(%22МП%22)). В представленном на рисунке примере имеются только два внутренних регистра — сам порт и регистр управления. Часто в одной микросхеме помещают несколько портов.
6. Временная диаграмма записи данных в параллельный порт представлена на рис. 2.

|  |
| --- |
| ?n=2 |

**Рис. 2.**

1. Микропроцессор устанавливает соответствующее значение адреса на [шине адреса](javascript:termInfo(%22Шина%20адреса%22)) и данные на [шине данных](javascript:termInfo(%22Шина%20данных%22)). После активизации сигнала разрешения записи WR данные с шины данных записываются в регистр-порт и выдаются на внешние линии связи (выходы порта). На этих выходах данные будут существовать до тех пор, пока новые данные не будут записаны в порт.
2. Временная диаграмма чтения данных из порта представлена на рис. 3.

|  |
| --- |
| ?n=3 |

**Рис. 3.**

1. Микропроцессор устанавливает соответствующее значение адреса на шине. После активизации сигнала разрешения чтения RD данные из регистра-порта (соответствующие напряжениям на входах порта) поступают на шину данных. Внешнее устройство, разумеется, должно установить данные на линиях порта до начала указанной процедуры чтения.
2. Запись данных в регистр управления (программирование интерфейса) происходит аналогично записи данных в регистр – порта (рис. 2), при этом на шине адреса устанавливается адрес, соответствующие регистру управления, а данные на линиях порта, естественно, не изменяются.

●**Оформите отчет и сделайте выводы по работе**.

Ответьте на контрольные вопросы

**??? Контрольные вопросы**

1.Понятие интерфейса.

2. Параллельная передача данных. Шина данных. Шина адреса. Шина управления.

3. Минимально необходимый набор управляющих сигналов в синхронном параллельном интерфейсе.

4. Отличительные особенности асинхронной передачи в параллельном интерфейсе.

5. Каким образом в параллельном интерфейсе формируются сигналы начала и конца передачи.

**Лабораторная работа №4.**

Исследование структуры и механизма прерываний микропроцессора i960.

(ЕВ 160 урок 8)

**Цель занятия:**

* изучить структуру и механизм работы прерываний микропроцессора i960;
* провести анализ механизма прерываний механизм прерываний микропроцессора i960для использования программ обработки прерываний.

***Продолжительность занятия*** *- 2 часа.*

**Пояснения к работе**

*Краткие теоретические сведения*

*Прерывания* - это временные остановки в работе программы, которые позволяют компьютеру выполнять другие действия. Обычно прерывания вызываются внешними источниками.

В момент, когда происходит прерывание, перед вызовом программы обработки прерываний, осуществляется сохранение текущего состояния программы. Прерывание подобно вызову подпрограммы, однако оно требует дополнительной обработки для того, чтобы сохранить текущее состояние программы.

**Аппаратные прерывания**

Микропроцессор i960 имеет четыре контакта аппаратного прерывания. INT2, т. е. третья линия вызова прерывания, используется для информирования процессора о прерывании от различных устройств ввода/вывода на плате EB-160, таких как: двухпозиционные переключатели, аналого-цифровые преобразователи или таймеры 8254.

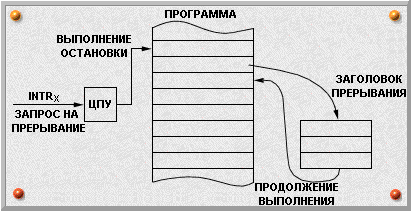


Рис. 1. Обработка прерываний

**Структура данных прерываний**

Процессор использует множество важных структур для обработки прерываний. Здесь Вы можете их рассмотреть:

31 24 23 16 15 8 7 0

|  |  |  |  |
| --- | --- | --- | --- |
| Вектор INT3 | Вектор INT2 | Вектор INT1 | Вектор INT0 |

**Регистр управления прерываниями**

Это – 32-битный регистр процессора. Как Вы видите, он разделен на четыре байта по 8 битов. Каждый байт отвечает за одну линию аппаратного прерывания и может иметь значение от 0 до 255. Эта цифра представляет собой номер элемента в большой таблице, называемой таблицей векторов прерываний. Она имеет уже 256 элементов.

Когда происходит аппаратное прерывание, процессор читает значение из этого регистра и использует его как смещение в таблице векторов прерываний для того, чтобы извлечь из нее адрес программы обработки прерывания и перейти по требуемому адресу.

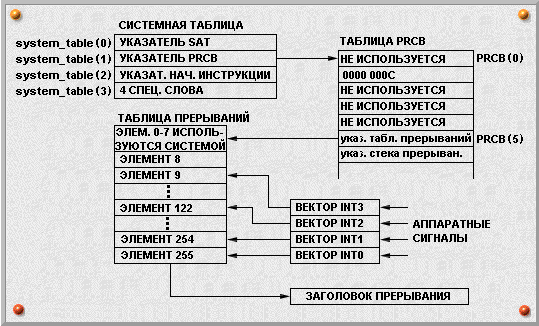


Рис.2. Механизм прерывания

**Таблица векторов прерываний**

Таблица вектора прерываний содержит 256 32-разрядных элементов. Когда происходит аппаратное прерывание, процессор обращается к регистру управления прерываниями и берет из него значение, соответствующее номеру элемента из этой таблицы.

Затем процессор читает 32-разрядное значение требуемого элемента. Это значение является адресом программы обработки прерываний, которая должна быть запущена.

Из 256 возможных элементов - входов прерываний, первые 8 и с 244 по 251 зарезервированы для нужд микропроцессора. В следующем упражнении мы проинициализируем таблицу прерываний так, чтобы она указывала на нашу программу обработки прерывания.

**Плата EB-160 основана на микропроцессоре Intel i960**. В противоположность стандартному CISC микропроцессору, RISC микропроцессор имеет меньшее число команд, что реализуется более простыми аппаратными средствами, но их выполнение происходит значительно быстрее. Аббревиатура RISC расшифровывается как Reduced Instruction Set Computing (Процессор с сокращенным набором команд).

RISC популярен в ресурсоемких приложениях, таких как: передача данных, численный анализ и в сложных замкнутых системах управления.

**Оборудование, приборы и элементы:**

* Плата EB-160 с последовательным кабелем.
* Плата EB-160 основана на микропроцессоре Intel i960
* Стенд EB-2000
* Установленное программное обеспечение для платы EB-160

**Задание**

Изучите и проанализируйте структуру и механизм прерываний микропроцессора i960.

**Порядок выполнения работы**

1. Поместите плату EB-160 на рабочей станции EB-2000
2. Подключите последовательный кабель к входу RS-232 (A) платы с одной стороны и, к COM1 или COM2 (COM порт, который не используется мышкой) персонального компьютера с другой стороны, как показано на следующей схеме:

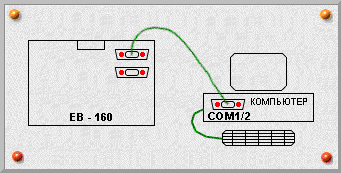
Если Вы используете модуль PU-2000, то Вы должны иметь два свободных последовательных порта на вашем компьютере: один вход для запуска отладчика DB960, а другой – для активизации неисправностей с помощью персонального компьютера для упражнений по нахождению и устранению неисправностей .

Рис. 3. Соединение EB-160 c персональным компьютером

**3.Запустите отладчик**

3.1.Есть два способа запуска отладчика на персональном компьютере:

* 1. Из меню Start/Program панели задач выберите иконку DB960.
  2. Нажмите на иконку DB960 в правом нижнем углу экрана программы обучения.

3.2.Убедитесь, что главное меню отладчика появилось на экране.

Если появится диалоговое окно с ошибкой, нажмите на кнопку RESET на плате, дождитесь загорания светодиода и нажмите на кнопку RETRY в диалоговом окне ошибки.

Если это не поможет, то необходимо выйти из отладчика, сбросить плату и перезапустить отладчик.

*Вы должны нажимать кнопку RESET на плате EB-160 каждый раз, когда Вы выходите из программы отладчика DB960.*

Запустите отладчик, нажав на соответствующую иконку db960.

3.3.Введите команду include exer7.inc. Эта команда запускает специальный файл, который автоматически выполнит заранее заданный набор команд отладчика: загрузит программу, установит контрольные точки и запустит программу до достижения первой контрольной точки.

3.4.Проверьте, что программа загружена и выполнение остановилось в начале основной подпрограммы main( ).

3.5.Нажмите на кнопку REGS, чтобы открыть окошко регистров. Скорректируйте его так, чтобы на нем отображался полный набор регистров.

3.6.Увеличьте окно с исходным текстом программы.

**4.Инициализация прерываний**

4.1.Нажмите на F5, чтобы перейти на первую контрольную точку L01.

4.2.Специальные системные таблицы, которые процессор использует в своей работе, хранятся в специальном секретном месте, и могут быть обнаружены только с помощью специальных сообщений называемых iac сообщениями. На метке L01 программа посылает специальное сообщение внутренней структуре микропроцессора и запрашивает адрес этого места. Этот адрес сохраняется в переменной system\_base.

4.3.Из system\_base мы можем извлечь адрес таблицы прерываний. Таблица прерывания описана выше и содержит 256 32-разрядных векторов, которые используются механизмом прерываний i960, для того чтобы выполнить программу обработки прерывания и руководить запросами прерываний.

4.4.В метках L02 и L03 вы видите, что в элемент таблицы записывается значение DEGEM\_VECTOR+1 , которое указывает на место в памяти, где расположена программа degem\_int. Проверьте это с помощью окна памяти.

4.5.Далее в метке L04 происходит установка регистра управления прерываниями. Как описано выше, он содержит четыре 8-битных байта, соответствующих каждому из четырех входов аппаратного прерывания, в единственном 32-разрядном регистре. Каждый байт указывает на элемент в таблице векторов прерываний. Проверьте переменную "interrupts" в метке L04, и убедитесь, что третий байт равен 0xd2.

4.6.Проверьте переменную "interrupts" в метке L04, и убедитесь, что третий байт равен 0xd2(xxd2xxxх). "d2" - значение вектора INT2 для подпрограммы обработки прерываний degem\_int.

4.7.В метке L04A устанавливается уровень приоритета. Он используется процессором для того, чтобы определить, какие прерывания будут обслуживаться, и в какой последовательности. Для наших задач мы снижаем приоритет до нуля. Это позволит всем прерываниям быть выполненными.

**5.Выполнение прерывания**

5.1.Нажмите на F5, чтобы перейти к L05. Это - начало цикла "for". Вызов подпрограммы InterruptReadStatus() разрешает прерывания на плате EB-160.

5.2.В этом упражнении мы создадим прерывания от переключателей.

5.3.В этом месте программа состоит из двух частей. Цикл "while" в L06 постоянно ожидает установления значения во флаге SW\_flag. Когда происходит изменение состояния переключателя, генерируется импульс прерывания, который поступает на вывод процессора INT2.

5.4. Третий байт регистра управления прерываниями содержит 0xd2 - "смещение" элемента таблицы векторов прерываний, в которой содержится адрес подпрограммы обработки прерывания (degem\_int).

5.5.Эта подпрограмма обслуживает прерывание и выставляет значение флажка SW\_flag. После возврата из прерывания, цикл "while" продолжает проверять значение в этом флажке. После того как флажок оказывается установленным, программа выходит из цикла "while" на метку L07.

5.6.Откройте окно памяти отладчика. Щелкните правой кнопкой мышки и выберите memory display.

5.7.В поле начального адреса введите метку int\_table, которая указывает на начало таблицы векторов прерываний. Заметим, что за исключением первых девяти "слов", все числа равны (0x00000d88).

5.8.В окне управления введите команду "asm 0x00000d88 length 50". Это стандартная программа, используемая для выполнения прерываний. Обратите внимание, что команда "ret" на девятой строчке указывает на конец подпрограммы.

5.9.Вернитесь в окно памяти. Щелкните правой кнопкой мыши внутри окна. Выберите "memory display" и введите следующий начальный адрес памяти: int\_table+((0xd2+1)\*4). Убедитесь, что использовали ORD4 для параметра TYPE. Это начальный адрес нашей подпрограммы обработки прерывания degem\_int ().

5.10.В окне с исходным текстом программы переместитесь в начало подпрограммы degem\_int() и нажмите на F3, чтобы отобразить код ассемблера.

5.11.Проверьте, что начальный адрес degem\_int() соответствует значению в элементе таблицы векторов прерываний в окне памяти.

5.12.Нажмите снова на F3, чтобы отобразить исходный текст программы на языке С. Пролистайте окно исходного текста до контрольной точки L07. Обратите внимание, что в L07 читается состояние переключателей, а состояние светодиодов обновляется в соответствии с текущими позициями всех переключателей.

5.13.Теперь мы готовы к запуску прерывания. По нажатию на F5 запускается цикл "while", который продолжается пока программа обработки прерывания degem\_int() не установит флажок SW\_flag. После этого управление остановиться в метке L07. Теперь попробуйте это выполнить.

5.14.Нажмите на F5, чтобы запустить цикл "for" и ждущий цикл "while".

*ЗАМЕЧАНИЕ: Чтобы гарантировать надежную связь между EB-160 и компьютером, не выполняйте больше, чем одного переключения между контрольными точками. Ждите не менее пяти секунд, перед тем как изменять положения переключателей*.

5.15. Измените положение одного из переключателей и проверьте, что выполнение программы остановилось на метке L07, указывая тем самым, что подпрограмма обработки прерывания была выполнена и флажок SW\_flag установлен.

5.16.Снова нажмите на F5, чтобы убедиться, что состояние светодиодов правильно обновлено.

●Оформите отчет.

**Представьте** в отчете приведите рис3.Соединение EB-160 c персональным компьютером и дайте письменно ответы на контрольные воросы( номера вопросов указывает преподаватель)

**●Сделайте выводы по работе**

**??? Контрольные вопросы**

1. Термин RISC расшифровывается как

* Класс микропроцессоров, менее надежные, чем другие
* Микропроцессор, который имеет упрощенную систему команд
* Микропроцессор с низкой скоростью вычисления
* Ни один из вышеперечисленных

2. Программа в EB-160 была загружена:

* При использовании специального параллельного порта компьютера
* При использовании принтерного порта компьютера
* При использовании последовательных COM1 или COM2 портов

Ни один из вышеперечисленных ответов не правилен

3. Регистр управления прерыванием процессора i960 обеспечивает

* + 32-разрядный вектор, который является адресом подпрограммы обработки прерываний.
  + Четыре 8-битных вектора, которые указывают на 1 из 256 элементов в таблице прерываний 960SA.
  + 32-разрядный вектор, который указывает на 1 из 1024 элементов в таблице прерываний i960.
  + Прерывания не поддерживаются в i960.

4. Сигнал на аппаратное прерывание посылается процессору i960 при одном из состояний двухпозиционного переключателя:

* Любой переключатель в верхнем положении.
* Любой переключатель в нижнем положении.
* Переключение с нижнего в верхнее положение.
* Переключение с верхнего в нижнее положение.
* Переключение с нижнего в верхнее или с верхнего в нижнее положение.

**Лабораторная работа№5.**

Проведение анализа работы порта последовательной передачи МП

**цель работы:**

* Понять принципы работы порта последовательной передачи данных 8051.
* Устанавливать связь между EB-153 и персональным компьютером

**Пояснения к работе**

*Краткие теоретические сведения*

МП 8051 имеет последовательный порт ввода-вывода, который нужен для связи с периферийными устройствами через стандартный асинхронный интерфейс в полнодуплексном режиме.

Последовательный порт также может работать в синхронном режиме.

Полнодуплексный последовательный порт ввода-вывода может работать в асинхронных режимах, для поддержки связи со стандартными UART устройствами, такими как персональные компьютер, принтеры и т.д.

Во время связи со стандартными UART устройствами последовательный канал может быть запрограммирован для приема/передачи 10-битового блока данных (фрейма) (Режим 1) или для приема/передачи 11-битового блока данных (фрейма) (Режим 2 или 3).

Фрейм содержит стартовый бит, восемь или девять битов данных и стоп бит.

В Режиме 1 и 3 модуль синхронизации передачи получает импульс от счетчика 1 каждый раз, когда счетчик переполняется.

Вход счетчика 1 может быть подключен к внешнему источнику или получать внутренний сигнал, частота которого равна одной двенадцатой частоты встроенного генератора.

Режим автоперезагрузки счетчика T1 обеспечивает связь на скорости 122 - 31250 бит в секунду (включая стартовый и стоп биты), при частоте встроенного кварцевого генератора 12 МГц.

Управление последовательным портом и получение информации о его состоянии обеспечивается регистром управления последовательного порта (SCON).

Содержимое 8-битового SCON - регистра показано ниже

Биты управления режимом SM0 и SM1 программируют последовательный порт для работы в одном из четырех режимов.

Режим 1 устанавливает UART интерфейс в режим приема/передачи 10-битового фрейма (8 бит данных, 1 стоп бит и 1 старт бит) при переменной скорости передачи.

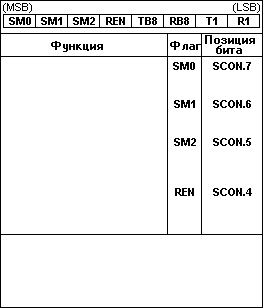
Данные для передачи и приема находятся в регистре буфера последовательного порта (SBUF).

Запись в SBUF обновляет регистр передатчика, тогда как при чтении из SBUF считывается информация из буфера, который обновляется регистром приемника если/когда флаг RI сброшен.

Приемник имеет два буфера, чтобы избежать переполнения, которое может произойти, если ЦП не ответит на прерывание приемника перед началом следующего фрейма.

Бит разрешения приема (REN) служит для перезагрузки алгоритма начала/конца приема.

Когда программа записывает в REN единицу (1), генератор скорости приема данных устанавливается в исходное состояние и обеспечивается возможность приема.



**Примечание:**

**Состояние SM0, SM1 определяет:**

**(0,0) расширение I/O (входа/выхода) регистра сдвига**

**(0,1) фрейм 8 бит, переменная скорость передачи**

**(1,0) фрейм 9 бит, фиксированная скорость передачи**

**(1,1) фрейм 9 бит, переменная скорость передачи**

**Режим последовательного порта. Бит 0. Установка/сброс выполняется програмно см. примечание.**

**Режим последовательного порта. Бит 1. Установка/сброс выполняется програмно см. примечание.**

**Режим последовательного порта. Бит 2. Установка выполняется программой для отмены приема фреймов, в которых бит 8=0.**

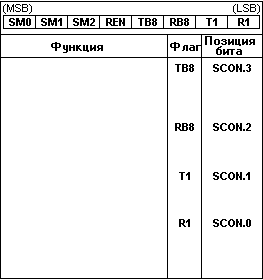
**Бит разрешения приема. Установка/сброс выполняется программой для разрешения/запрета приема последовательных данных.**

Значение REN должно быть установлено при инициализации последовательного канала. Если значение REN равно нулю, прием запрещен.

ЦП получает информацию, что область передатчика в SBUF пустая или область приемника заполнена, считывая значения T1 и R1 соответственно.

Значения T1 и R1 должны быть очищены программой обработки прерывания, чтобы не создавать непрерывного цикла прерывания ЦП.

Так как запрос прерывания последовательного порта является результатьм логического сложения (ИЛИ) значений T1 и R1, эти биты должны быть последовательно опрошены, чтобы определить источник прерывания.



**Передача бита 8. Установка/сброс выполняется аппартным оборудованием для определения состояния 9-ого бита передаваемого в режиме 9-ти битного UART.**

* **Прием бита 8. Установка/сброс выполняется аппратным оборудованием для индикации состояния 9-ого бита принятых данных.**

**Флаг прерывания передачи. Устаналивается аппратным оборудованием после передачи байта. Сбрасывается программой после обработки данных.**

**Флаг прерывания приема. Устанавливается аппаратным оборудованием, когда байт принят. Сбрасывается программой после обработки данных.**

**Задание1.**Изучите принципы работы порта последовательной передачи данных 8051

**Задание 2**.Устанавливать связь между EB-153 и персональным компьютером

*Краткая теория к заданию 2.*

Команды связи позволяет пользователю послать и получить данные через последовательный интерфейс RS-232 EB-153.

Эти команды управляют работой UART в 8031.

Последовательные данные ввода-вывода передаются или принимаются от микрокомпьютера EB-153 по кабелю связи.

Другой конец этого кабеля должен быть подключен к последовательному порту COM1/COM2 персонального компьютера.

Используются только три сигнала RS-232:

* TxD (передаваемые данные)
* RxD (принимаемые данные)
* GROUND (общий провод)

Сигнал TxD ПК (контакт 2) должен быть подключен к сигналу RxD микрокомпьютера EB-153.

Сигнал RxD ПК (контакт 3) должен быть подключен к сигналу TxD микрокомпьютера EB-153.

Общий провод (контакт 5) должн быть связан с общим проводом микрокомпьютера EB-153.

8031 обеспечивает подачу тактовых импульсов для своего внутреннего UART.

Системная программа задает скорость передачи, равную 2400 бод.

**Оборудование, приборы и элементы:**

* Лабораторный стенд PU-2000
* Печатная плата EB-153
* Комплект соединительных кабелей DL-20

**Порядок работы**

1.Присоедините коммуникационный кабель.

2.Наберите команду UPLOAD на клавиатуре микрокомпьютера EB-153-10.

3.Вставьте коммуникационную дискету в дисковод, если программа передачи данных не была скопирована на жесткий диск компьютера.

4.Нажмите на иконку Приложение в правом нижнем углу.

5.На экране компьютера появиться программа, которая предложит пользователю выполнить все необходимые последующие действия (задать имя файла, адрес и т.д.). Адреса указываются с помощью программы передачи данных, установленной на ПК.

6.Скопируйте блок управляющей программы EB-153 в файл на жестком диске ПК.

7.Наберите имя файла и нажмите OK.

8.Выберите стартовый адрес 0000H и конечный адрес 0200H. Нажмите Apply (применить).

Данные, находящиеся по адресу 0000H - 0200H будут переданы в файл на жестком диске ПК.

9.Файл CN показывает файлы в шестнадцатеричном формате на экране ПК. Для вызова программы просто нажмите на иконку Application (Приложение) в правом нижнем углу экрана.

10.Команда загрузки копирует программу, находящуюся в файле на жестком диске ПК в память микрокомпьютера EB-153. Формат команды: DOWN.

Адреса определяются с помощью программы передачи данных ПК.

11.Чтобы загрузить программу, придерживайтесь следующего порядка действий:

**а)** Подключите кабель для последовательной связи.

**б)** Наберите команду DOWN на клкавиатуре микрокомпьютера EB-153.

**в)** Вставьте коммуникационную дискету в дисковод, если программа передачи данных не была скопирована на жесткий диск ПК.

Команда "DO" выполняет загрузку вашего двоичного файла в память EB-153, используя последовательную связь с ПК.

**г)** Нажмите иконку Application (Приложение) в правом нижнем углу.

**д)** На экране компьютера появиться программы, которая сообщит пользователю о всех необходимых последующих действиях (задание имени файла, адреса и т.д.).

12.Адреса определяются программой передачи данных, установленной на ПК.

13.Скопируйте файл расположенный на жестком диске в ОЗУ EB-153. Укажите имя файла, который Вы хотите загрузить в ОЗУ и нажмите ОК.

14.Выберите стартовый адрес E000 и конечный адрес E200. Нажмите Apply (Применить).

Данные из указанного файла будут переданы в ОЗУ. Просмотрите переданные данные с помощью команду CBYTE.

15.Выключите PU-2000. Отключите ЕВ-153 от PU-2000.

●Оформите отчет и ответьте на контрольные вопросы.

**??? Контрольные вопросы**

**1.** Какой регистр осуществляет управление последовательным портом и обеспечивает информацию о его состоянии?

**2.** Укажите тип интерфейса и режим его работы, через который последовательный порт ввода-вывода МП 8051 осущесталяет связь МП с периферийными устройствами?

3. Приведите примеры стандартных UART устройств

4. На какой скорости режим автоперезагрузки счетчика T1 обеспечивает связь при частоте встроенного кварцевого генератора 12 МГц?

5. В каком регистре последовательного порта находятся данные для передачи и приема?

**Лабораторная работа№6.**

Проведение анализа выполнения асинхронной передачи данных в микропроцессоре

10

**цель работы:**

* Передавать ASCII символы в асинхронном формате.

**Оборудование, приборы и элементы:**

* Лабораторный стенд PU-2000
* Печатная плата EB-153
* Комплект соединительных кабелей DL-20
* Осциллограф
* Персональный компьютер

**Порядок выполнения работы**

1.Подключите EB-153 к PU-2000. Включите PU-2000.

2.Следующая программа устанавливает Таймер №1 в режим генератора скорости передачи данных со значением 2400 бод и определяет конфигурацию порта последовательной передачи данных.

Данная программа передает значение символа ASCII "A" (41H) по TxD линии.

Введите программу в память микрокомпьютера.

АДРЕС КОМАНДА КОММЕНТАРИЙ

**главная программа**

E000 MOV DPTR,#FF20H устанавливает указатель на адрес

перехода T1

E003 MOV A,#02H LJMP код

E005 MOVX @DPTR,A записывает JMP E100H в FF20H

E006 INC DPTR указывает на первый операнд

E007 MOV A,#E1H старший разряд адреса

E009 MOVX @DPTR,A записывает первый операнд

E00A INC DPTR указывает на второй операнд

E00B MOV A,#00H младший разряд адреса

E00D MOVX @DPTR,A записывает второй операнд

E00E MOV B8H,#10H определяет регистр IP, адрес B8H, T1

высший приоритет

E011 MOV 89H,#20H определяет регистрTMOD, адрес 89H,

режим 2- 8-битная перезагрузка

E014 MOV 8DH,#F3H определяет регистр TH1, адрес 8DH,

значение перезагрузки F3H для

скорости двоичной передачи=2400

E017 MOV 98H,#40H определяет регистр SCON, адрес 98H,

режим 1 - запрещен приемник TI=1

E01A MOV 88H,#40H определяет регистр ТCON, адрес 88H,

запускает Таймер 1

E01D MOV A8H,#90H определяет регистр IE, адрес A8H,

прерывание последовательного порта

разрешено

E020 MOV 99H,#41H устанавливает SBUF (99H) в "A"

E023 LJMP E023H ждет прерывания

**программа обработки прерываний**

E100 MOV 99H,#41H устанавливает SBUF (99H) в "A"

E103 CLR 99H

E102 RETI возвращается из режима прерывания

Подключите осциллограф к клемме TxD.

Запустить программу с помощью команды GO.

На экране осциллографа должен появиться сигнал асинхронного формата (8 битов данных, 1 стартовый импульс и 1 стоп импульс).

Обратите внимание, что сигналы имеют логические уровни присущие RS-232, а не TTL уровни, и они инвертированы.

Нажмите кнопку RESET, чтобы остановить программу.

Запустите данную программу еще раз для другово ASCII символа.

В этой части эксеримента система изменит передаваемые данные.

Что произошло с линией связи?

●Оформите отчет и ответьте на контрольные вопросы.

**??? Контрольные вопросы**

1. Разница между синхронной и асинхронной передачей
   * Асинхронная передача данных добавляет старт бит и стоп бит к байту, и в ней нет синхронизующих сигналов.
   * Синхронная передача данных добавляет стартовый бит и стоп бит к байту, и ей необходимы синхронизующие сигналы.
   * Отличаются по скорости передачи.
2. Логические уровни сигналов RS-232
   * TTL уровни.
   * "1" = +24V, "0" = 0V или наоборот.
   * "1" = -12V, "0" = +12V.

1. Скорость передачи информации- это
   * Величина, которая показывает скорость последовательной передачи (биты в секунду).
   * Количество передаваемых байт (бит/секунда)
   * Число стоп битов, добавленных к байтам данных.

**Лабораторная работа№7.**

Исследование применение шины передачи данных МП

**цель работы:**

* Научиться проводить анализ потоков данных в микропроцессорных системах.
* Понимать назначение состояния ожидания.
* Научиться обнаруживать неисправности в микропроцессорных схемах.

**Продолжительность занятия** - 2 часа.

**Пояснения к работе**

*Краткие теоретические сведения*

**Архитектура** микропроцессорных систем — это принципы организации обмена информацией по шинам таких систем. Без этого невозможно разработать аппаратную часть системы, а без аппаратной части не будет работать никакое программное обеспечение. **Принципы организации обмена** **по шинам гораздо важнее, чем особенности конкретных микропроцессоров.** Стандартные системные магистрали живут гораздо дольше, чем тот или иной процессор.

Разработчики новых процессоров ориентируются на уже существующие стандарты магистрали. Более того, некоторые системы на основе совершенно разных процессоров используют одну и ту же системную магистраль. **То есть** **магистраль оказывается самым главным системообразующим фактором в микропроцессорных системах**.

Микросхема процессора обязательно имеет **выводы трех шин: шины адреса, шины данных и шины управления.** Иногда некоторые сигналы и шины мультиплексируются, чтобы уменьшить количество выводов микросхемы процессора

**Разрядность шины данных определяет скорость работы системы**. Разрядность шины адреса определяет допустимую сложность системы. Количество линий управления определяет разнообразие режимов обмена и эффективность обмена процессора с другими устройствами системы

**Большое достоинство шинной структуры связей состоит в том, что все устройства, подключенные к шине, должны принимать и передавать информацию по одним и тем же правилам (протоколам обмена информацией по шине ).** Соответственно, все узлы, отвечающие за обмен с шиной в этих устройствах, должны быть единообразны, унифицированы.

**Существенный недостаток шинной структуры связан с тем,** что все устройства подключаются к каждой линии **связи параллельно.** Поэтому любая неисправность любого устройства может вывести из строя всю систему, если она портит линию связи. По этой же причине отладка системы с шинной структурой связей довольно сложна и обычно требует специального оборудования.

**Шина данных — это основная шина, которая используется для передачи информационных кодов между всеми устройствами микропроцессорной системы**. Обычно в пересылке информации участвует процессор, который передает код данных в какое-то устройство или в ячейку памяти или же принимает код данных из какого-то устройства или из ячейки памяти. Но возможна также и передача информации между устройствами без участия процессора.

*Шина данных всегда двунаправленная.* Двунаправленная линия (шина) — линия (шина), по которой сигналы могут передаваться в обоих направлениях (по очереди).

**Оборудование, приборы и элементы:**

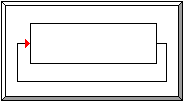
* Лабораторный стенд PU-2000
* Плата с печатной схемой EB-151-11(7)
* Двухканальный осциллограф
* Набор соединительных проводов

**Порядок работы**

* 1. Разместите печатную плату EB-151 на стенде и вставьте ее в разъем PU-2000.
  2. Первая программа в этом эксперименте записывает байт данных в порт вывода 0.

В двоичном исчислении байт записывается как « 01010101», в шестнадцатеричном – «55».

Эти действия показаны на рисунке, представленном ниже.

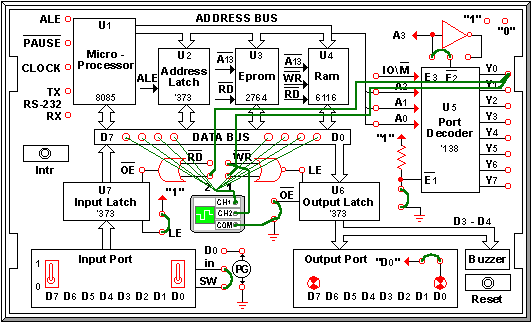


**ЗАПИСЬ 55H**

**В ПОРТ ВЫВОДА 0**

Запись байта данных в порт 0

3.Соедините схему, как показано на рис.:



4.Установите переключатели в положение 10010000.

5.Для запуска программы 16, нажмите клавишу RESET и установите переключатель D7 в положение "0" .

6.Подсоедините первый канал осциллографа к (WR)', а с помощью второго контакта осциллографа, поочередно проверьте сигналы шины передачи данных.

7.Зарисуйте формы сигналов шины передачи данных, наблюдаемых на осциллографе во время цикла записи. Синхронизируйте осциллограф по каналу, который подключен к (WR)'.

8.Соедините (WR)' с (PAUSE)'. Этим вы создадите "состояние ожидания". Микропроцессор перестанет работать, и сигналы шины останутся постоянными.

**9.**Проверьте сигналы шины передачи данных и (WR)' с помощью осциллографа.

10.Запишите наблюдаемые логические состояния:

(Обратите внимание, что светодиоды показывают обратный код данных)

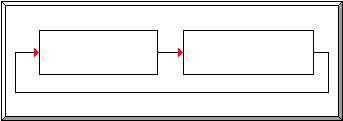
D7 D6 D5 D4 = \_\_\_\_\_\_\_

D3 D2 D1 D0 = \_\_\_\_\_\_\_

11.Отсоедините (WR)' от (PAUSE)'. Это вернет микропроцессор в нормальный режим работы.

12.Вторая программа этого эксперимента считывает байт данных с Порта ввода 0 и записывает его в Порт вывода 2.

Эта операция показана на рисунке ниже.



**ЗАПИСАТЬ В**

**ПОРТ ВЫВОДА 2**

**СЧИТАТЬ С**

**ПОРТА ВВОДА 0**

***Программа практического задания***

13.Удалите соединительный провод, соединяющий выход Декодера 1 с выходом Декодера порта Y0, и подсоедините ее к Y2 (см. схему внизу). Установите переключатели в положение 10010001.

14.Для запуска программы 17, нажмите клавишу RESET и установите переключатель D7 в положение "0".

15.Используйте осциллограф для проверки логических уровней сигналов RD', WR', ДЕКОДЕРА 1, ДЕКОДЕРА 2.

16.Запишите амплитуду логических уровней "0" и "1" для различных сигналов. Схема по-прежнему работает правильно.

RD' - "0" =

"1" =

WR' - "0" =

"1" =

ДЕКОДЕР 1 - "0" =

"1" =

ДЕКОДЕР 2 - "0" =

"1" =

17.В плату EB-151 внесена неисправность.

18.Установите переключатели в указанные ниже положения.

19.Обратите внимание на состояние светодиодов. Изучите состояния светодиодов Порта вывода.

|  |  |
| --- | --- |
| ВХОДНЫЕ ДАННЫЕ | ВЫХОДНЫЕ ДАННЫЕ |
|  |  |
| a. 0000 |  |
| b. 0000 |  |
| c. 0000 |  |
| d. 0000 |  |
| e. 0001 |  |
| f. 0010 |  |
| g. 0100 |  |
| h. 1000 |  |

20.Если вы обнаружили неправильный выходной сигнал, используйте осциллограф, чтобы определить место неисправности. Исследуйте сигналы шины передачи данных вместе с сигналами RD' и WR'.

Какое утверждение верно?

* Стробирующий импульс IO/M' нестационарен.
* Уровень "1" стробирующего импульса WR' слишком низок.
* Уровень "0" стробирующего импульса WR' слишком высок.
* Уровень "1" стробирующего импульса RD' слишком низок.

Уровень "0" стробирующего импульса RD' слишком высок

Оформите отчет и сделайте выводы по работе. Ответьте на контрольные вопросы

**???Контрольные вопросы**

1. **При записи, данные должны появиться в шине передаче данных при:**

* (RD)' = 1
* (RD)' = 0
* (WR)' = 1
* (WR)' = 0

**2**. **Чтобы остановить работу шины передачи данных, вы должны:**

* **Соединить (WR)' с (RD)'.**
* Соединить (WR)' с (PAUSE)'.
* Соединить (WR)' с (PAUSE)' через элемент "НЕ".

**3. Шина передачи данных необходима:**

* Чтобы передать сигналы, которые управляют потоком данных.
* Для передачи самих данных.
* Для обеспечения шинами компании Data Corp.

**Лабораторная работа №8**.

Исследование организации работы интерфейса клавиатуры( ЕВ153 11)

**Цель занятия:**

* Исследоватьорганизации работы интерфейса клавиатуры
* Изучить программу для опроса клавиатуры

**Продолжительность занятия** - 2 часа

**Пояснения к работе**

***Краткие теоретические сведения***

***ИСПОЛЬЗОВАНИЕ КЛАВИАТУРЫ EB-153***

EB-153-11 имеет встроенную клавиатуру, которая позволяет пользователю вводить команды и данные.

Клавиши клавиатуры составляют матрицу. Ее ряды и колонки подключены к программируему переферийному интерфейсу ППИ, как показано на следующем рис.1.

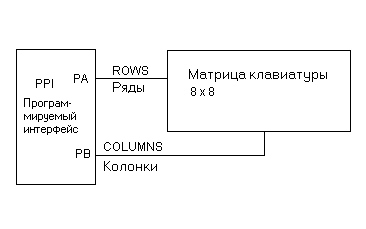
******

Рис. 1. Подключение клавиатуры

Системная программа выполняет следующие задачи:

* + 1. Генерирует импульсы сканирования рядов, которые подаются на ряды клавиатуры.
    2. Считывает состояние колонок клавиатуры и определяет нажатые клавиши.
    3. Устраняет эффект дребезга контактов нажатой кнопки.
    4. Преобразует скан-коды в АSCII символы.

Нажатая клавиша определяется следующим образом:

1. На одну из сигнальных линий колонок РВ подается "0", все остальные установлены в "1".
2. Сигнальные линии рядов PA - это линии возврата сигнала. Если нажата клавиша, на соответствующей сигнальной линии ряда, считывается "0".
3. Сканируя все колонки программа может определить нажатую клавишу.

**Оборудование, приборы и элементы:**

* Лабораторный стенд EB-2000
* Печатная плата EB-153
* Комплект соединительных кабелей DL-20

**Порядок работы**

1.Подключите EB-153 к PU-2000. Включите PU-2000

2. Следующая программа определяет нажатую клавишу и выдает на экран номера ряда и колонки нажатой клавиши.

Пользователь может присвоить любое ASCII значение полученным скан-кодам.

**Программа клавиатуры**

АДРЕС КОМАНДА КОММЕНТАРИЙ

E000H LCALL FF80H очищает дисплей

E003H MOV DPTR, #0003H устанавливает адрес Режима ППИ

E006H MOV A, #99H определяет выходы PA и PB

E008H MOVX @DPTR, A устанавливает Режим ППИ

E009H MOV R0, #00H устанавливает в исходное положение

указатели ряда и колонки

E00BH MOV R1, #FEH устанавливает в исходное состояние

активную колонку

E00DH MOV DPTR, #0001H устанавливает адрес PB

E010H MOV A, R1 считывает данные колонки PB

E011H MOVX @DPTR, A устанавливает активную колонку

E012H MOV DPTR, #0000H устанавливает адрес PA

E015H MOVX A, @DPTR читает данные ряда

E016H CPL A инвертирует данные ряда

E017H JNZ 0CH определяет нажатую клавишу

E019H INC R0 увеличивает значение указателя

колонки

E01AH MOV A, R1 получает новый указатель

E01BH RL A циклически сдвигает указатель

колонки

E01CH MOV R1, A устанавливает новую активную

колонку

E01DH MOV A, R0 читает значение указателя

E01EH ADD A, #F8H проверяет, последняя ли колонка

E020H JZ E7H Если да, возвращается в начальное

состояние

E022H LJMP E00DH если нет, проверяет следующую

колонку

E025H MOV R2, #00H программа нажатой клавишы

E027H RRC A ряд определен

E028H JC 04H проверяет бит переноса

E02AH INC R2 следующий ряд

E02BH LJMP E027H продолжает

E02EH MOV A, R0 получает номер ряда

E02FH ADD A, #30H преобразует в ASCII

E031H MOV DPTR, #FF93H указатель дисплея

E034H MOVX @DPTR, A отображает на дисплее

E035H MOV A, R0 получает номер колонки

E036H ADD A, #30H преобразует в ASCII

E038H MOV DPTR, #FFA3H указатель диспля

E03BH MOVX @DPTR, A отображает на дисплее

E03CH MOV DPTR, #FF90H отображает ряд

E03FH MOV A, #52H 'R'

E041H MOVX @DPTR, A определяет изображение

E042H INC DPTR следующее изображение

E043H MOV A, #3DH '='

E045H MOVX @DPTR, A опеределяет изображение

E046H MOV DPTR, #FFA0H отображает колонку

E049H MOV A, #43H 'C'

E04BH MOVX @DPTR, A опеределяет изображение

E04CH INC DPTR следующее изображение

E04DH MOV A, #3DH '='

E04FH MOVX @DPTR, A опеределяет изображение

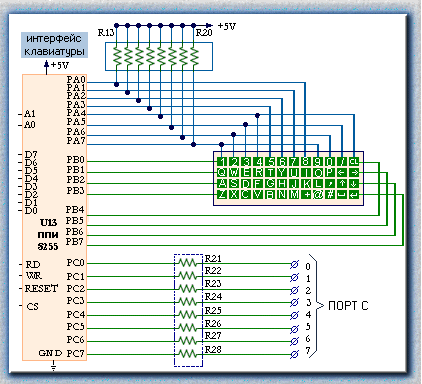
E050H LCALL FF83H отображает ряд и колонку

E053H LJMP E009H возвращается к началу цикла

1. Выполните программу.

5.Нажимайте на разные клавиши и наблюдайте за номерами ряда и колонки отображемыми на экране.

Сравните их с фактическим подключением клавиатуры, показанным на следующем рис..



●Оформите отчет и ответьте на контрольные вопросы.

**??? Контрольные вопросы**

1. Для сканировании матрицы из 64 клавиш, необходимо , чтобы число контактов порта было равно
   * 8
   * 12
   * 16
   * 64

**2**. На схеме подключения клавиатуры, выходы Порта А соединены с шиной питания через резисторы, потому что

* + Они устанавливают все невыбранные входы в "0".
  + Они устанавливают все невыбранные входы в "1".
  + Они устанавливают все выходные линии в "1".

**Лабораторная работа №9.**

Исследование организации работы интерфейса ЖК дисплея

**Цель работы:**

* Научиться программировать встроенный модуль ЖК-дисплея.

**Продолжительность занятия** – 2 часа

**Пояснения к работе**

*Краткие теоретические сведения*

В настоящее время в микропроцессорных системах для отображения широко

используют жидкокристаллические индикаторы (ЖКИ).

Условно все ЖКИ можно разделить на две категории: символьные, или знакосинтезирующие и графические.

Графические индикаторы представляют собой матрицу из m строк и n столбцов, на

пересечении которых находятся пиксели.

Пиксель представляет собой неделимый объект прямоугольной или круглой формы, обладающий определённым цветом; пиксель – наименьшая единица растрового изображения. Если на определенный столбец и строку подать электрический сигнал, то пиксель на их пересечении изменит свой цвет. Подавая группу сигналов на столбцы и строки можно формировать по точкам произвольное графическое изображение.

Так работает графический ЖКИ. В символьном же ЖКИ матрица пикселей разбита на подматрицы , каждая подматрица предназначена для формирования одного символа: цифры, буквы или знака препинания. Как правило, для формирования одного символа используют матрицу из восьми строк и пяти столбцов.

Символьные индикаторы бывают одно-, двух- и четырехстрочными.

Для упрощения взаимодействия микропроцессорной системы и ЖКИ используют

специализированную микросхему – контроллер (драйвер) ЖКИ. Он управляет пикселями

жидкокристаллического дисплея и интерфейсной частью индикатора. Обычно такой

контроллер входит в состав индикатора. В целом жидкокристаллический индикатор

представляет собой печатную плату, на которой смонтирован сам дисплей, контроллер и необходимые дополнительные электронные компоненты.

ЖК матричный модуль, входящий в состав EB-153 позволяет отобразить 32 символа. Этот модуль имеет свой собственный контроллер и принимает от внешнего микропроцессора информацию о том, что необходимо отообразить и на какой позиции. Пользователь может прочитать или записать Управляющую Функцию (адрес 1001H и 1002H) или прочитать или записать Данные (адреса 1001Н - 1003Н).

**Оборудование, приборы и элементы:**

* Лабораторный стенд EB-2000
* Печатная плата EB-153
* Комплект соединительных кабелей DL-20

**Порядок работы**

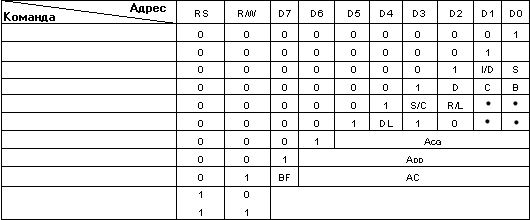
1.Подключите EB-153 к PU-2000. Включите PU-2000

2. Ознакомьтесь с набором команд, используемых для работы ЖК дисплея ( см. таблицу ниже).

Следующая программа показывает, как использовать некоторые функции ЖК дисплея.

Эта программа очищает дисплей и посылает ЖК модулю команды, чтобы отобразить весь набор символов.

***ЖК КОМАНДЫ***



**Очистка дисплея**

**Исходная позиция курсора**

**Установка режима ввода**

**Управление вкл/выкл дисплея**

**Сдвиг курсора**

**Установка функции**

**Установка адреса CG RAM**

**Установка адреса DD RAM**

**Чтение адреса/флага готовности**

**Запись в CG RAM/DD RAM**

**Чтение из CG RAM/DD RAM**

записать данные

считать данные

I/D=1 увеличение на 1 I/D=0 уменьшение на 1

S=1 сдвиг индикации S=0 фиксирование индикации

D=1 включить дисплей D=0 выключить дисплей

C=1 индикация курсора C=0 выключить индикацию курсора

B=1 символ на позиции курсора мигает B=0 символ на позиции курсора не мигает

S/C=1 сдвиг индикации S/C=0 движение курсора

R/L=1 сдвиг вправо R/L=0 сдвиг влево

DL=1 8 бит DL=0 4 бита

BF=1 в течении внутренней операции BF=0 окончание внутренней операции

3..Введите следующую программу.

АДРЕС КОМАНДА КОММЕНТАРИЙ

E000H MOV DPTR, #1000H определяет адрес команды

E003H MOV A, #38H устанавливает длину данных 8 бит

E005H MOVX, @DPTR, A записывает команду

E006H LCALL E200H задержка

E009H MOV DPTR, #1000H определяет адрес команды

E00CH MOV A, #01H очищает дисплей

E00EH MOVX @DPTR, A записывает команду

E00FH LCALL E200H задержка

E012H MOV DPTR, #1000H определяет адрес команды

E015H MOV A, #06H устанавливает режим – увеличивает

адрес на 1

E017H MOVX @DPTR, A записывает команду

E018H LCALL E200H задержка

E01BH MOV DPTR, #1000H определяет адрес команды

АДРЕС КОМАНДА КОММЕНТАРИЙ

E01EH MOV A, #0CH дисплей включен - курсор выключен

E020H MOVX @DPTR, A записывает команду

E021H LCALL E200H задержка

E024H MOV DPTR, #E100H устанавливает данные дисплея

E027H MOVX A, @DPTR считывает значение счета

E028H MOV R7,A записывает A в R7

E029H ANL A,#E0H проверяет данные в A

E02BH JNZ 02H переходит, если А больше чем 20H

E02DH MOV R7,#20H записывает 20H в R7

E02FH MOV A,R7 перемещает значение R7 в A

E030H INC A увеличивает счет на 1

E031H ANL A,#7FH ограничивает счет

E033H MOVX @DPTR, A Сохраняет счет

E034H DPTR, #1001H Записывает указатель данных

E037H MOVX @DPTR, A Отображает счет

E038H LCALL E200H Задержка

E03BH LJMP E024H Возвращается в начало цикла

E200H MOV DPTR, #FFC0H Устанавливает указатель на

задержку программы

E203H MOV A, #01H Определяет постоянную задержки

E205H MOVX @DPTR, A Записывает постоянную

E206H LCALL FF8CH Подпрограмма задержки

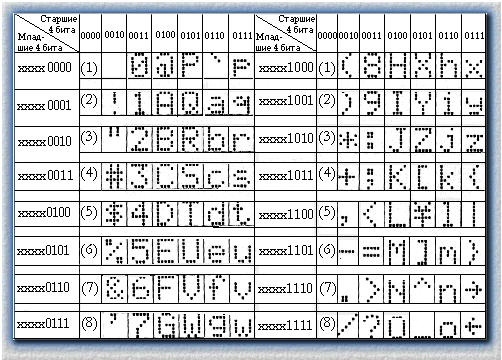
E209H RETI Возвращается в программу

4..Программа задержки (адрес E200H) упрощает программу и позволяет обойтись без проверки флага готовности дисплея.

5.Выполните программу и проследите за символами, появляющимися на ЖК дисплее. Вы можете сравнить их с таблицей символов ЖК дисплея.

Заметьте, что задержка между отображением строк обусловлена загрузкой информации во внутреннее ОЗУ ЖК.

Таблица знакогенератора ЖК дисплея



Оформите отчет и ответьте на контрольные вопросы

**???Контрольные вопросы**

1.Какой физический принцип заложен в сущности работы ЖКИ?

2. Какие используются режимы отображения ЖКИ

3.На какие категории по принципу отображения информации делятся ЖКИ?

4.Какие функции выполняет драйвер ЖКИ?

5.Каким образом происходит управление ЖКИ?

**Лабораторная работа № 10**.

Исследование интерфейса связи RS232

**Цель работы:**

исследовать и изучить:

* Сигналы RS-232
* Использование интерфейса "приемник-передатчик"
* Подключение микрокомпьютера к персональному компьютеру

**Продолжительность занятия – 2 часа**

**Пояснения к работе**

*Краткие теоретические сведения*

***ФОРМАТЫ ПЕРЕДАЧИ***

Последовательная передача при помощи канала передачи данных или локального периферийного устройства, происходит в одном из двух основных форматов: *асинхронном* или *синхронном*.

Эти форматы похожи тем, что для правильного определения символа получателем требуется служебная информация.

Главное отличие между двумя форматами состоит в том, что в асинхронном формате к каждому символу добавляется служебная информация, а в синхронном формате служебная информация добавляется к блокам данных или сообщений.

В асинхронном формате передача основных битов данных начинается со СТАРТ бита и заканчивается СТОП битом.

СТАРТ бит - это логический ноль, и определяется он, как положительный уровень напряжения RS-232.

СТОП бит - это логическая единица, и определяется он, как отрицательный уровень напряжения RS-232.

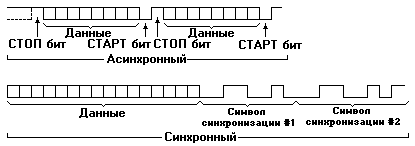
СТАРТ бит это команда для принимающего устройства начинать прием. Она позволяет синхронизовать принимающее устройство с передатчиком.

Один или более СТОП битов добавляются к символу в конце, чтобы гарантировать, что СТАРТ бит следующего символа вызовет переходное состояние на канале связи.

Синхронный формат, вместо добавления битов к каждому символу, группирует символы в блоки и добавляет служебные символы к блокам.

Служебные символы, известные как символы синхронизации, используются получателем для определения, где находятся границы блока в последовательности битов.

Пример синхронного и асинхронного форматов приведен ниже:



***ОПИСАНИЕ ОСНОВНЫХ ФУНКЦИЙ 8251A***

8251A - это универсальный синхронный/асинхронный приемо-передатчик (USART), специально спроектированный для систем микрокомпьютера.

Как и другие устройства ввода-вывода в системе микрокомпьютера, его функциональная конфигурация программируется системным программным обеспечением.

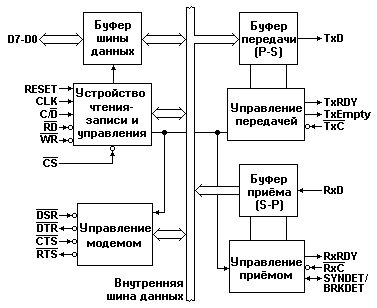
8251A может поддерживать практически любую форму синхронной передачи данных, используемую в настоящее время.

В коммуникационном оборудовании, интерфейсное устройство, должно преобразовывать данные из параллельного формата в последовательный при передаче данных, а при приеме, производить обратное преобразование.

Устройство, осуществляющее интерфейс, должно также удалять и вставлять биты или символы, которые индивидуальны для каждого вида коммуникационной технологии.

Интерфейс должен быть согласован с ЦПУ, обеспечивать ввод или вывод системных данных в байтах.

На рисунке изображена блок-схема 8251A:



**БУФЕР ШИНЫ ДАННЫХ**

Двунаправленный 8-разрядный буфер, имеющий 3 состояния, используется для подключения 8251A к системной шине данных.

Данные, управляющие слова, командные слова и информация состояния передается через буфер шины данных.

Состояние, запись данных, чтение данных - это отдельные 8-разрядные регистры для обеспечения двойной буферизации.

***RESET (СБРОС)***

Сигнал высокого уровня на этом входе переводит 8251A в режим "бездействия".

Устройство будет находиться в этом режиме, пока в него не поступит новый набор управляющих слов.

Минимальная ширина импульса RESET составляет 6 тактовых импульсов (генератор тактовых импульсов должен быть включен).

***CLK (тактовая частота)***

Вход CLK используется для внутренней синхронизации устройства.

Никакие внешние входы или выходы, напрямую, не зависят от сигналов CLK, однако, частота CLK должна превышать скорость получения или передачи битов данных в 30 раз.

*** (ЗАПИСЬ)***

Сигнал низкого уровня на этом входе информирует 8251A, что ЦПУ записывает в него данные или управляющие слова.

*** (ЧТЕНИЕ)***

Сигнал низкого уровня на этом входе информирует, что ЦПУ читает из него данные или информацию о состоянии.

***C/ (УПРАВЛЕНИЕ/ДАННЫЕ)***

Входной сигнал на этом выводе, вместе с входами  и  , сообщает 8251A, что слово на шине данных является либо символьными данными, либо управляющим словом или информацией о состоянии.

1 = УПРАВЛЕНИЕ/СТАТУС

2 = ДАННЫЕ

*** (ВЫБОР ЧИПА)***

Низкий уровень сигнала на этом входе означает выбор 8251A.

До тех пор пока устройство не выбрано, не происходит ни чтения, ни записи данных.

Когда сигнал  - высокого уровня, шина данных "отключена" от устройства, и сигналы  и  не будут влиять на работу 8251A.

**УПРАВЛЕНИЕ МОДЕМОМ**

8251A имеет набор входных и выходных управляющих сигналов, которые упрощают взаимодействие с передающей аппаратурой.

Управляющие сигналы модема являются сигналами общего назначения и, если необходимо, могут быть использованы для функций, не связанных с управлением модемом.

*** (ИСТОЧНИК ДАННЫХ ГОТОВ)***

Входной сигнал [overline|DSR ] является инвертирующим однобитным портом общего назначения.

ЦПУ может проверить его состояние при помощи операции чтения.

 вход обычно используется, для проверки состояния модема.

** ***(ТЕРМИНАЛ ДАННЫХ ГОТОВ)***

 является выходным инвертирующим однобитным портом общего назначения.

Низкий уровень сигнала  может быть установлен путем программирования соответствующего бита в командном слове.

 выходной сигнал обычно используется для команд

управления модемом, таких как "терминал данных готов" или "выбор скорости передачи".

*** (ЗАПРОС НА ПЕРЕДАЧУ)***

Выходной сигнал  является инвертирующим однобитным портом общего назначения.

Низкий уровень сигнала  может быть установлен путем программирования соответствующего бита в командном слове.

Выходной сигнал  обычно используется для управления модемом, например, как "запрос на передачу".

** ***(ГОТОВНОСТЬ ПРИЕМА)***

Сигнал низкого уровня на входе  "разрешает" 8251A осуществлять передачу данных, если значение бита Tx Enable (передача разрешена) в командном байте равно "единице".

Если значение бита Tx Enable становиться равно "нулю" или сигнал  переходит в состояние высокого уровня, в то время как TxD (передача данных) активен, то TxD передает все данные, полученные до прихода команды отключения передачи - Tx Disable.

***БУФЕР ПЕРЕДАТЧИКА***

Буфер передатчика получает параллельные данные от буфера шины данных и преобразует их в последовательный поток битов, вставляя при этом соответствующие синхронизирующие символы или биты (в зависимости от вида передачи). Затем, при ниспадающем фронте TxC, выдает последовательный поток данных на выходной контакт TxD.

Передатчик начнет передачу, если =0.

Выход (TxD) будет удерживаться в состоянии логической "единицы" после общего сброса системы, когда TxEnable/  низкого уровня или при высоком уровне сигнала TxEmpty (передатчик пуст).

***УПРАВЛЕНИЕ ПЕРЕДАЧЕЙ***

Блок управления передачей управляет всеми процессами передачи последовательных данных.

***TxRDY (ПЕРЕДАТЧИК ГОТОВ)***

Этот выход сигнализирует центральному процессору, что передатчик готов получить символ данных.

Выходной контакт TxRDY может использоваться как запрос на прерывание процессора. В связи с тем, что он маскируется путем выключения разрешения передачи (TxDisabled), его состояние может быть опрошено процессором с помощью операции чтения состояния.

TxRDY автоматически сбрасывается по восходящему фронту входного сигнала , когда символ данных загружается из процессора.

Заметьте, что при операции опроса состояния, бит статуса TxRDY не маскируется сигналом TxEnabled, а только отражает состояние Пуст/Полон регистра ввода данных передатчика.

***TxE (ПЕРЕДАТЧИК ПУСТ)***

Уровень сигнала TxЕ становиться высоким, когда у 8251A нет символов, готовых для передачи.

При получении символа от ЦПУ, он автоматически сбрасывается.

TxЕ может быть использован для обозначения конца режима передачи. Таким образом, ЦПУ "узнаёт", когда необходимо "сменить направление передачи" в режиме полудуплексной передачи.

TxЕ не зависит от состояния бита TxEnabled в команде управления.

 ***(ТАКТОВЫЙ СИГНАЛ ПЕРЕДАТЧИКА)***

Тактовый сигнал передатчика управляет скоростью передачи информации.

В синхронном режиме, скорость передачи информации равна частоте  .

В асинхронном режиме, скорость передачи информации - это доля от фактической частоты .

Команда установки режима устанавливает этот параметр; он может быть равен 1, 1/16 или 1/64 от .

Например:

Если скорость передачи равна 100 бит:

-  равен 110Гц (1X).

-  равен 1.76кГц (16X).

-  равен 7.04кГц (64X).

Ниспадающий фронт сигнала  вызывает сдвиг последовательности данных в 8251A.

**БУФЕР ПРИЕМА**

Приемник получает последовательные данные, преобразует их в параллельный формат, проверяет служебные биты или символы, которые зависят от метода связи, и посылает обработанный символ в ЦПУ.

Последовательные данные приходят на вход RxD, и синхронизируется по восходящему фронту сигнала [overline|RxC ].

***УПРАВЛЕНИЕ ПРИЕМОМ***

Этот функциональный блок управляет всеми относящимися к получению данных функциями, такими как:

Схема инициализации RxD предотвращает ошибочное "восприятие" неиспользуемой входной линии, в качестве сигнала низкого уровня, в "состоянии паузы".

После сброса 8251A, перед началом приема последовательных символов на входе RxD, должна присутствовать логическая единица.

После прихода сигнала высокого уровня, происходит ожидание требуемого сигнала низкого уровня (СТАРТ бит).

Эта особенность есть только при работе в асинхронном режиме. Такая процедура выполняется только один раз после каждого сброса.

Схема обнаружения фальшивого СТАРТ бита предотвращает ошибочные старты, вызванные шумом переходного процесса. Осуществляет она это, выявляя сначала ниспадающий фронт поступившего сигнала, а затем синхронизирует условный центр СТАРТ бита (сигнал RxD низкого уровня).

Схемы переключения четности и ошибки четности используются для обнаружения ошибки четности и установки соответствующего бита состояния.

Флаг Ошибки Кадрирования устанавливается, если в конце байта данных отсутствует СТОП бит (асинхронный режим). Он также, устанавливает соответствующий бит состояния.

***RxRDY (ПРИЕМНИК ГОТОВ)***

Этот выход показывает, что 8251A содержит символ, готовый для передачи в ЦПУ. RxRDY может включаться в структуру прерываний ЦПУ. Кроме этого его состояние можно проверить, опросив соответствующий регистр.

*** (ТАКТОВЫЙ СИГНАЛ ПРИЕМНИКА)***

Тактовый сигнал приемника управляет скоростью приема символа.

В синхронном режиме, скорость приема (1X) равняется фактической частоте  .

В асинхронном режиме скорость приема - это доля от фактической частоты  .

Команда установки режима устанавливает этот параметр: 1, 1/16 или 1/64 от частоты  .

Скорость равняется 2400 битам в секунду, если:

-  равняется 2400 Гц (1X)

-  равняется 38.4 кГц (16X)

-  равняется 153.6 кГц (64X)

***SYNDET (ОПРЕДЕЛЕНИЕ СИНХРОНИЗАЦИИ)/BRKDET (ОПРЕДЕЛЕНИЕ ПАУЗЫ)***

В синхронном режиме этот вывод используется для обнаружения синхронизации (SYNDET). С помощью управляющего слова он может быть запрограммирован для работы в качестве как входного, так и выходного сигнала.

По сбросу он устанавливается в низкое логическое состояние на вывод.

При использовании в качестве выходного (режим внутренней синхронизации), вывод SYNDET будет переходить в "1" для указания, что 8251А обнаружил символ синхронизации в режиме приема.

Если 8251А запрограммирован для использования двойных символов синхронизации, вывод SYNDET будет переходить в "1" в середине последнего бита второго символа синхронизации.

SYNDET автоматически сбрасывается при операции чтения состояния.

При использовании в качестве выходного (режим внешней синхронизации), "высокий" сигнал заставит 8251A начать сбор данных на восходящем фронте следующего [overline|RxC] сигнала.

После синхронизации, высокий уровень входного сигнала может быть удален.

Когда запрограммирован режим внешней синхронизации, режим внутренней синхронизации не работает.

***ОПРЕДЕЛЕНИЕ ПАУЗЫ (Break Detect) (ТОЛЬКО В АСИНХРОННОМ РЕЖИМЕ)***

На этом выходе будет сигнал высокого уровня всегда, когда входной сигнал приемника будет оставаться на низком уровне в течение двух последовательных СТОП битов (включая СТАРТ биты, биты данных и биты четности).

Этот сигнал может быть прочитан так же, как бит состояния.

Он сбрасывается только в случае, если происходит сброс чипа или RxD возвращается в состояние "1".

**ДЕТАЛЬНОЕ ОПИСАНИЕ РАБОТЫ 8251A**

Полный набор функциональных возможностей 8251A программируется системным программным обеспечением.

Для инициализации 8251A и программирования формата связи, с которым он должен будет работать, ЦПУ должен послать в него управляющие слова.

Управляющие слова программируют СКОРОСТЬ ПЕРЕДАЧИ, ДЛИНУ СИМВОЛА, ЧИСЛО СТОП БИТОВ, СИНХРОННУЮ или АСИНХРОННУЮ работу, ЧЕТНОСТЬ и т.д.

В синхронном режиме работы, необходимо также выбрать режим внешней или внутренней синхронизации.

Будучи запрограммированным, 8251A готов к выполнению своих функций связи.

Выходной сигнал TxRDY становится высокого уровня, сообщая ЦПУ, что 8251A готов начать прием символов.

При записи символа в 8251A, этот сигнал (TxRDY) автоматически сбрасывается.

В то же время, 8251A принимает данные от модема или другого устройства ввода-вывода.

После принятия символа, сигнал RxRDY становится высокого уровня, сообщая ЦПУ о наличии в 8251A символа, готового для передачи в ЦПУ.

По окончании операции чтения данных из 8251A, RxRDY автоматически сбрасывается.

8251A не может начать передачу до тех пор, пока не будет установлен бит TxEnabled в управляющей команде и не будет получен сигнал CTS (готовность к передаче).

***ПРОГРАММИРОВАНИЕ 8251A***

Перед тем как начать передачу данных или их прием, 8251A должен быть загружен набором управляющих слов, генерированных ЦПУ.

Эти управляющие сигналы полностью определяют работу 8251A и должны следовать сразу же за операцией сброса (внутреннего или внешнего).

Управляющие слова разбиваются на два формата:

1. Инструкции по установке режима

2. Управляющие команды

***ИНСТРУКЦИЯ ПО УСТАНОВКЕ РЕЖИМА***

Этот формат определяет основные характеристики работы 8251A.

Инструкция по установке режима должна следовать непосредственно после операции сброса.

Только после этого, в 8251А могут загружаться требуемые символы синхронизации и управляющие команды.

***УПРАВЛЯЮЩИЕ КОМАНДЫ***

Их формат определяет слово состояния, которое используется для управления работой 8251A.

Для правильной работы устройства, инструкция по установке режима и управляющие команды должны следовать друг за другом в определенном порядке.

Инструкция по установке режима должна следовать непосредственно после операции сброса.

Все управляющие слова, записанные в 8251А после инструкции по установке режима, будут загружать управляющие команды.

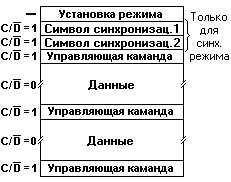
Управляющие команды могут быть записаны в любое время работы 8251А.

Для возврата в формат инструкции по установке режима, необходимо произвести сброс 8251А, активизировав соответствующий бит в слове управляющей команды. После выполнения операции сброса, 8251А автоматически перейдет в формат установки режима.

Управляющая команда должна следовать за установкой режима или символами синхронизации.

Следующая диаграмма показывает стандартный блок данных.

***Примечание:*** Если установлен режим внутренней синхронизации с одиночным символом, то второй символ синхронизации пропускается. Если 8251A запрограммирован для работы в асинхронном режиме, пропускаются оба символа синхронизации.



***УСТАНОВКА РЕЖИМА***

8251A может использоваться как для асинхронной, так и для синхронной передачи данных.

Для того, чтобы понять, как инструкция по установке режима определяет работу 8251A, надо представить, что устройство состоит из двух отдельных блоков: синхронного и асинхронного, объединенных в одном корпусе.

Изменить формата можно только после выполнения сброса.

Для лучшего понимания работы устройства, два формата будут рассматриваться отдельно друг от друга.

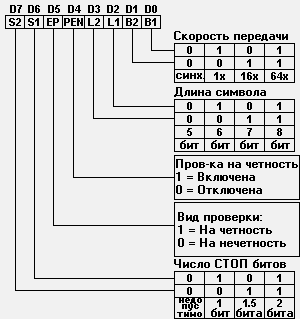
***Примечание:*** Бит четности не учитывается при определении длины слова.

Бит четности, принимаемый по линии RxDATA, не может быть считан с шины данных.

В случае если длина символа меньше 8 бит, информация будет содержаться в младших битах шины данных. При записи данных в 8251A, состояние неиспользуемых битов "безразлично", а при чтении из 8251A значение неиспользуемых битов равно "нулю".

На рисунке представлен формат инструкции по установке формата для асинхронного режима.

Порядок работы



***АСИНХРОННЫЙ РЕЖИМ (ПЕРЕДАЧА)***

Всякий раз, когда символ данных передается из ЦПУ, 8251A автоматически добавляет СТАРТ бит (низкий уровень сигнала), за которым следуют биты данных (сначала младший разряд), и определенное число СТОП битов в конце каждого символа.

Кроме этого, перед СТОП битами также помещаются биты проверки на четность/нечетность, согласно установкам режима.

Затем, символ передается как поток последовательных данных на выход TxD.

Последовательные данные сдвигаются при ниспадающем фронте сигнала TxC со скоростью равной 1, 1/16 или 1/64 от TxC, согласно установленному режиму.

Символы "ПАУЗЫ" могут непрерывно посылаться TxD, если есть соответствующая на то команда.

Если в 8251A нет никаких данных готовых к передаче, на TxD будет оставаться сигнал высокого уровня, если не было запрограммировано состояние "ПАУЗЫ" (непрерывный сигнал низкого уровня).

***АСИНХРОННЫЙ РЕЖИМ (ПРИЕМ)***

Сигнал RxD обычно высокого уровня.

Ниспадающий фронт на RxD означает получение СТАРТ бита.

Достоверность СТАРТ битов проверяется повторным их стробированием (только режим 16X или 64X).

Если повторно принят сигнал низкого уровня, это означает, что был принят достоверный СТАРТ бит. В этот момент начинается отсчет битов.

Счетчик битов определяет местоположение битов данных, битов четности и СТОП битов.

Если происходит ошибка четности, выставляется флаг ошибки четности.

Биты данных и четности принимаются на входе RxD и синхронизируются восходящим фронтом RxC.

Если во время принятия СТОП бита, обнаружен сигнал низкого уровня, то выставляется флаг ошибки кадрирования.

СТОП бит сигнализирует о конце символа.

Заметьте, что приемник требует только один СТОП бит, не взирая на их запрограммированное число.

После этого, символ загружается в параллельный буфер ввода-вывода 8251A.

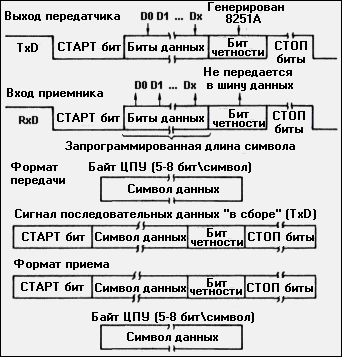
Сигнал RxRDY сообщает ЦПУ, о наличии в буфере 8251А принятого символа.

Если предыдущий символ не был считан ЦПУ, то текущий символ замещает его в буфере ввода-вывода (предыдущий символ теряется). Кроме этого выставляется флаг ошибки переполнения.

Все флаги ошибок могут быть сброшены командой "СБРОС ОШИБКИ".

Появление всех этих ошибок не влияет на работу 8251A.

На рисунке показан формат данных для асинхронного режима.



**УПРАВЛЯЮЩИЕ КОМАНДЫ 8251A**

После того как в 8251А записана инструкция по установке режима и загружены символы синхронизации, устройство готово к приему/передаче.

Управляющие команды управляют фактической работой выбранного формата.

Они обеспечивает такие функции, как: РАЗРЕШЕНИЕ ПРИЕМА/ПЕРЕДАЧИ, ОШИБКА, СБРОС и УПРАВЛЕНИЕ МОДЕМОМ.

После установки режима 8251A и, по необходимости, определении символов синхронизации, все последующие "управляющие записи" (C/D=1), загружают управляющие команды.

Операция СБРОСА (внешнего или внутреннего) вернет 8251A к установке режима.

Формат управляющей команды:

E154f56

EH: ВОЙТИ В РЕЖИМ ПОИСКА

(Не работает в асинхронном режиме)

1 = Разрешен поиск символов синхронизации

IR: ВНУТРЕННИЙ СБРОС

Сигнал высокого уровня возвращает 8251A к установке режима

RTS: ЗАПРОС НА ПЕРЕДАЧУ

Сигнал высокого уровня выставляет нулевой сигнал на .

ER: СБРОС ОШИБКИ

1 = Сбрасывает флаги ошибки; PE, OE, FE

Он должен быть произведен всякий раз, когда

запрограммированы RxEnable и EH.

Формат управляющей команды:

E154f56

SBRK: ПОСЛАТЬ СИМВОЛ ПАУЗЫ

1 = Устанавливает на выходе TxD сигнал низкого уровня; 0 = Обычная работа

RxE: ПРИЕМ РАЗРЕШЕН

1 = Разрешение; 0 = Запрещение

DTR: ГОТОВ ВЫВОД ДАННЫХ

Сигнал высокого уровня выставляет нулевой сигнал на 

TxEN: ПЕРЕДАЧА РАЗРЕШЕНА

1 = Разрешение; 0 = Запрещение

**ОПРЕДЕЛЕНИЕ СОСТОЯНИЯ 8251A**

В телекоммуникационных системах часто требуется проверять состояние активного устройства для определения, не возникла ли ошибка или другое условие, требующее вмешательства процессора.

8251А имеет средство чтения состояния, что позволяет системному программному обеспечению прочитать состояние устройства в любой момент во время работы.

(Во время чтения состояния, обновление состояния запрещено).

Для осуществления операции чтения состояния, процессором вырабатывается обычная команда чтения, при C/D=1.

Некоторые из битов слова состояния имеют то же значение, что и внешние выходные сигналы. Это позволяет работать с 8251А как на уровне прерываний, так и на программном уровне.

TxRDY - исключение.

Обратите внимание, что между фактическим изменением состояния интерфейса и значением регистра, присутствует задержка, максимальное значение которой равно 28 периодам тактового сигнала.

Формат регистра состояния:

E154f56

DSR: НАБОР ДАННЫХ ГОТОВ

Указывает, что DSR - "ноль".

FE: ОШИБКА КАДРИРОВАНИЯ (FE) (не используется при

синхронной связи)

Флаг FE выставляется, если в конце символа не зафиксирован СТОП бит. Этот флаг сбрасывается битом ER управляющей команды. Установка FE не прекращает работу 8251A.

PE: ОШИБКА ЧЕТНОСТИ (РЕ)

Флаг PE выставляется, если возникла ошибка четности. Он сбрасывается битом ER управляющей команды. Установка PE не прекращает работу 8251A.

Формат регистра состояния:

E154f56

OE: ОШИБКА ПЕРЕПОЛНЕНИЯ (ОЕ)

Флаг OE выставляется, если ЦПУ не успел считать принятый символ, за то время, пока не пришел следующий. Установка OE не прекращает работу 8251A, однако предыдущий символ становиться утерянным.

TxRDY: Значение бита состояния TxRDY несколько отличается от значения одноименного контакта микросхемы 8251А.

Первый не зависит от CST и Enable, второй зависит и от того, и от другого.

Т.е. бит состояния TxRDY = DB буфер пуст контакт TxRDY = DB буфер пуст\*([overline|CTS ]=0)\*(TxEN=1)

**Оборудование, приборы и элементы:**

* Учебный стенд PU-2000
* Печатная плата EB-153
* Набор проводов DL-20
* Двухканальный осциллограф
* Персональный компьютер
* Кабель последовательной передачи

**Порядок работы**

1.Поместите плату ЕВ-153 на стенд PU-2000 и вставьте ее в разъем. Соедините EB-153 с EB-154

2.С помощью ленточного кабеля, подключите разъем расширения платы EB-153 к одному из разъемов расширения платы EB-154

**Задание 1.**

3.Проведение исследования использования программируемых интерфейсов связи. Программа тестирует асинхронный режим, посылая байт в USART (последовательный интерфейс).

3.1.Подключите CK0 (тактовый вход Таймера 0) к генератору тактовых импульсов с частотой импульсов 1МГц.

3.2.Подключите TxC к D0 (выход Таймера 0). Подайте на G0 (вход Таймера 0) сигнал высокого уровня (+5V), а на CTS сигнал низкого уровня ("Общий провод").

3.3. Включите стенд. Нажмите кнопку RESET на EB-153 и на EB-154.

Введите программу.

АДРЕС КОМАНДА КОММЕНТАРИЙ

E000H MOV DPTR,#7007H Устанавливает указатель адреса на

регистр режима

E003H MOV A,#3EH Определяет режим

E005H MOVX @DPTR,A Записывает режим

E006H MOV DPTR,#7008H Устанавливает указатель адреса на

T0

E009H MOV A,#08H Определяет младший разряд

E00BH MOVX @DPTR,A Записывает в T0 - 1МГц/9600 =68H

=104D

E00CH MOV A,#00H Определяет младший разряд

E00EH MOVX @DPTR,A Записывает в T0

E00FH MOV DPTR,#7008H Устанавливает указатель адреса

регистра команд

E012H MOV A,#4DH Определяет режим - 8 бит – один

СТОП бит

E014H MOVX @DPTR,A Записывает режим

E015H OV A,#05H Определяет команду - TxD и RxD

разрешены

E017H MOVX @DPTR,A Записывает команду

E018H MOV A,#55H Устанавливает 55H в

последовательный порт

E01AH MOV DPTR,#7004H Устанавливает указатель адреса на

регистр данных

E01DH MOVX @DPTR,A Посылает данные в USART

E01EH MOV DPTR,#7005H Устанавливает указатель адреса на

регистр команд

E021H MOVX A,@DPTR Читает состояние из USART

E022H MOV C,E0H Переносит бит 0 ACC биту переноса

E024H JNC FBH Не готов - переходит к E021H

E026H MOV A,#01H Определяет константу задержки

E028H MOV DPTR,#FFC0H Устанавливает адрес задержки

E02BH MOVX @DPTR,A Записывает задержку

E02CH LCALL FF8CH Вызывает задержку

E02FH LJMP E018H Набор данных готов

Эта программа использует Таймер 0, для генерирования скорости передачи данных в 9600 бод.

Для того, что бы получить желаемую частоту 9600 бод из входной тактовой частоты, равной 1МГц, необходимо использовать таймер в качестве делителя частоты на 1МГЦ/9600Гц = 104.

Именно на этой скорости USART будет передавать данные. Выбранные данные - 55H.

3.4. Подключите осциллограф к выводу TxD.

3.5.Выполните программу и пронаблюдайте за сигналами на осциллографе.

3.6.Подключите контакт TxD микросхемы USART к контакту TxIN интерфейса RS-232C. С помощью осциллографа, проверьте уровни сигнала, передаваемого через RS-232C.

3.7.Подключите контакт TxOUT к контакту RxIN интерфейса RS-232C.

3.8.С помощью осциллографа, проверьте уровни сигнала, принимаемого через RS-232C и преобразованного TTL сигнала.

3.9.Нажмите кнопку RESET на EB-153 и на EB-154.

**Задание 2.**

В следующем эксперименте мы соединим EB-153 с персональным компьютером через USART и интерфейс RS-232C платы EB-154.

1.Соедините USART с последовательным интерфейсом (TxD с TxIN и RxD с RxOUT).

2.Соедините RS-232C с вашим персональным компьютером: RxIN с Transmit Data (передаваемые данные) и TxOUT с Receive Data (принимаемые данные).

3.Соедините выход Таймера 0 с синхронизирующим входом TxC устройства USART. На вход таймера CK0 подайте сигнал тактового генератора с частотой 1МГц, а на вход G0 подайте сигнал высокого уровня.

4.С помощью интерпретатора языка БЕЙСИК, наберите и запустите программу, представленную на следующем экране.

Эта программа выводит на экран символы, полученные из последовательного порта.

Использование задержек при передаче данных, позволяет избежать переполнения буфера передачи на медленных компьютерах.

5.Наберите программу (написанную на языке БЕЙСИК) на вашем персональном компьютере.

10 OPEN "COM1:9600,N,8,1,RS.CS.DS.CD" AS #1

20 IF EOF (1) = -1 THEN 20 ELSE 30

30 A$ = INPUT$(1,#1): PRINT A$,: GOTO 20

6.Запустите программы для проверки асинхронной передачи данных, одновременно на компьютере и на плате EB-153.

7.Посмотрите на символы, появившиеся на экране персонального компьютера.

8.Нажмите кнопку RESET на EB-153 и на EB-154.

9.Выключите стенд.

10.Отсоедините провода и кабели от EB-153 и EB-154.

Оформите отчет и сделайте выоды по работе.

 Ответьте на контрольные вопросы

**???Контрольные вопросы**

1. USART - это
   * Универсальный синхронный/асинхронный приемопередатчик.
   * Универсальный высоко скоростной асинхронный приемопередатчик.
   * Универсальный синхронный усовершенствованный приемопередатчик.
2. Разница между синхронной и асинхронной передачей данных состоит в том, что
   * В асинхронном формате служебная информация не добавляется к блокам данных, а в синхронном формате добавляется.
   * В асинхронном формате служебная информация не добавляется к каждому символу, а в синхронном формате добавляется.
   * В асинхронном режиме служебная информация добавляется к каждому символу, а в синхронном режиме к блокам данных.
3. Скорость передачи данных - это
   * Скорость передачи данных - это количество блоков данных переданных за секунду.
   * Скорость передачи данных - количество байтов переданных за секунду.
   * Скорость передачи данных - это количество битов данных переданных за секунду.

**Лабораторная работа № 11.**

Исследование выполнения управляющих подпрограмм в микропроцессоре ЕВ153 урок3

**Цель работы:**

* научиться запускать простые программы;
* использовать управляющие программы;
* выполнять программы в пошаговом режиме и в режиме реальноого времени;

**Пояснения к работе**

*Краткие теоретические сведения*

В работе используются команды Ассемблера.

Эти команды позволяют пользователю читать и писать программы на языке Ассемблера. Команда ассемблера загружает мнемоническую команду в программную память.

Управляющая программа содержит указатель адреса команды, которая должна быть выполнена в ходе следующего рабочего цикла.

Команда ASM начинает свою работу с адреса, указанного после ключевого слова АSM. Если адрес не указан, используется значение указателя по умолчанию. Это значение равно первому доступному коду ОЗУ (Е000Н).

Управляющая программа отображает адрес указателя.

Для редактирования мнемонической команды 8051, пользователь должен нажать клавишу ENTER.

Стрелки вправо и влево могут быть использованы для правки текста.

Если пользователь введет неправильную команду, на дисплее высветится сообщение о синтаксической ошибке.

Синтаксическая ошибка не изменяет значение указателя.

**Приборы и элементы:**

* Лабораторный стенд PU-2000;
* Печатная плата EB-153;

- Комплект соединительных кабелей DL-20

### Порядок выполнения работы

1. Поместите печатную плату ЕВ-153 на лабораторный стенд.

Включите питание

**Эксперимент 1.** Изучение программы увеличения содержимого аккумулятора

1..Выполните команду ASM:

1.1. наберите следующее: ASM E000H и нажмите клавишу ENTER.

1.2.Наберите на клавиатуре команды указанные ниже и нажмите клавишу "CLEAR" после последней команды.

CLR A

INC A

LJMP E001H

1.3. Наберите DASM E000H и нажмите клавишу ЕNTER.

1.4.С помощью клавиши "стрелка вниз", выведите на экран программу сохраненную в памяти.

Сравните ее с командами Ассемблера, которые Вы ввели.

*Кнопка "стрелка вниз" может быть использована для увеличения указателя адреса на 1 и вывода других дезассемблированных команд на дисплей.*

1.5.Нажмите клавишу "CLEAR", чтобы выйти из данного режима.

1.6.С помощью команды CBYTE, убедитесь, что команды были записаны в память правильно, в шестнадцатеричном формате.

**Эксперимент 2.** Изучение команд режима ИСПОЛНЕНИЯ ПРОГРАММЫ

**2.**Команда GO запускает выполение программы в режиме реального времени.

2.1. Запустите программу GO, наберите следующюю команду и нажмите клавишу ENTER.

GO E000H

Во время выполнения программы на дисплее высвечивается сообщение: RUNNING USER'S PROGRAM (идет выполнение программы пользователя).

Программа будет выполняться до тех пор, пока вы не нажмете клавишу INTR.

Тогда выполнение программы остановится, и значение счетчика команд и соответствующая ему команда отобразиться на дисплее.

2.2.Нажимая клавишу "стрелка вниз", Вы можете вывести на дисплей следующие регистры и их значения:

SP (указатель стека) A (аккумулятор)

DPTR (указатель данных) РSW (слово состояния программы)

2.3.Нажмите клавишу "стрелка вниз" снова, чтобы вернуться в управляющую программу

**Эксперимент 3.** Изучениепрограммы прерывания программы.

При прерывании программы, используя специальные команды отображения данных (команды дампинга), можно вывести на дисплей содержимое регистров и внутренней памяти.

3.1.Наберите RDUMP EOH и нажмите клавишу ENTER, чтобы вывести на экран содержимое аккумулятора.

3.2.Нажмите на клавишу "CLEAR", чтобы отменить эту команду.

**Эксперимент 4.** Изучение оаботы программы в пошаговом режиме

В пошаговом режиме выполнение программы останавливается после выполнения каждой команды.

После остановки, управляющая программа отображает следующую команду и ее адрес (значение счетчика команд), содержимое аккумулятора, указатель данных, указатель стека и слово состояния программы.

Вы можете приостановить автоматическое пошаговое выполнение программы, нажав любую клавишу. В таком режиме отображемые на экране данные не меняются до тех пор пока Вы не отпустите нажатую клавишу. После того как вы отпустите клавишу, выполнение программы продолжиться.

4.1.Наберите на клавиатуре STEP E000H и нажмите клавишу ENTER, чтобы запустить пошаговое выполнение программы.

Наблюдайте за работой программы и приращением аккумулятора.

4.2.Попробуйте нажимать клавиши для того, чтобы приостанавливать выполнение программы на различных участках.

4.3.Нажмите клавишу INTR, чтобы прекратить выполнение программы.

4.4.Нажмите клавишу "стрелка вниз", чтобы вывести на дисплей содержание регистров.

4.5.Снова нажмите клавишу "стрелка вниз", чтобы выйти из режима пошагового выполнения программы.

**Оформите отчет и ответьте на контрольные вопросы**

??? **Контрольные вопросы**

1. Каким образом можно установить аккумулятор в 0?

* Нажатием клавиши "CLEAR" или клавиши "стрелка вниз".
* Нажатием клавиши INTR.
* С помощью команды RDUMP E0H.

1. Сколько ASCII-символов может быть отображено?

* 32
* 64
* 128

1. Какова разница между пошаговым выполнением программы и выполнением в реальном времени?

* В пошаговом режиме программа прерывается после выполнения каждой команды. В режиме реального времени программа прерывается только по адресу контрольной точки или при нажатии клавиши INTR.
* Никакой разницы нет, разница только в применении программы.
* Выполнение в реальном времени не может быть прервано.

**Лабораторная работа № 12**

Исследовние процесса передачи и извлечения данных в микропроцессоре

(ЕВ153 урок4)

**Цель работы:**

* научиться понимать разницу между режимами адресации микрокомпьютера;

- научиться передавать и извлекать данные и команды из различных областей памяти

**Пояснения к работе**

*Краткие теоретические сведения*

Микроконтроллер 8051 имеет четыре блока, каждый из которых содержит 8 регистров. Эти блоки находятся по адресам 0 - 31 (00H - 1FH) во внутреннем ОЗУ данных.

Адресация регистра обеспечивает доступ к восьми регистрам (R0-R7) выбранного блока регистров (RB).

Один из четырех блоков выбирается с помощью двухбитного поля в регистре PSW (слово состояния программы), показанного на схеме ниже.

Это два бита RS0 и RS1.

E153f31

**Приборы и элементы:**

* Лабораторный стенд PU-2000
* Печатная плата EB-153
* Комплект соединительных кабелей DL-20

### Порядок выполнения работы

Подключите EB-153 к PU-2000. Включите питание PU-2000.

**Эксперимент 1**. Управлять блоками регистров микроконтроллера 8051

1.1. Введите следующую программу в память микрокомпьютера.

АДРЕС КОМАНДА КОММЕНТАРИЙ

E000H MOV D0H,#18H выбор блока регистров 3

RS1=1, RS0=1

E003H MOV R0,#55H устанавливает R0=55H

E005H MOV R1,#66H устанавливает R1=66H

E007H MOV R2,#77H устанавливает R2=77H

E009H MOV D0H,#00H восстановление блока регистров 0

RS1=0, RS0=0

E00CH LJMP E00CH ПАУЗА

1.2. Запустите программу с помощью команды GO.

1.3. Нажмите кнопку INTR, чтобы остановить программу и покинуть режим выполнения программы. Не используйте команду "STEP", если вы работаете с регистром PSW.

*Используйте команду DDUMP, чтобы вывести на экран содержимое регистров R0 и R2, блока 2. Обратите внимание, что числовые значения из нашей программы были сохранены по соответствующим адресам (адрес с 18Н по 1АН).*

**Эксперимент 2.** Изучение примеров программы прямой адресации к внутреннему ОЗУ данных.

2.1.Введите в память микрокомпьютера программу, показанную ниже.

АДРЕС КОМАНДА КОММЕНТАРИЙ

E000H MOV 20H,#11H запись 11H в адрес 20H

E003H MOV 21H,20H запись содержимого адреса 20Н

в адрес 21H

E006H MOV 22H,21H запись содержимого адреса 21H

в адрес 22H

E009H LJMP E009H Пауза

2.2.Выполните программу и нажмите кнопку INTR для остановки программы.

*Используйте команду DDUMP, чтобы вывести на экран содержимое внутреннего ОЗУ данных. Обратите внимание, что числовое значение 11H сохранилось в соответствующих ячейках памяти (адрес 20Н и 21H).*

2.3.Определите различие между командами MOV.

Знак решетки (#) указывает на численное значение, которое передается по определенному адресу.

В нашем случае, число 11H передается по адресу 20H (команда MOV 21H,#11H).

Если знак "решетка" отсутствует, переданные данные - это содержимое ячейки по данному адресу.

В нашей программе, данные, находящиеся по адресу 20H передаются по адресу 21H (команда MOV 21H,20H) и адресу 22H (команда MOV 22H,21H).

**Эксперимент 3.** Изучение примера программы косвенной адресации к внутреннему ОЗУ данных.

3.1.Введите следующую программу в память микрокомпьютера:

АДРЕС КОМАНДА КОММЕНТАРИЙ

E000H MOV R0,#20H установка указателя адреса на 20H

E002H MOV @R0,#55H перемещение 55H в адрес укзанный в R0

E004H INC R0 увеличение указателя адреса

E005H MOV @R0,#66H перемещение 66H в адрес укзанный в R0

E007H INC R0 увеличение указателя адреса

E008H MOV @R0,#77H перемещение 77H в адрес укзанный в R0

E00AH MOV A,@R0 перемещение указателя данных

в регистр А

E00BH LJMP E00BH остановка

3.2.Запустите данную программу. Нажмите кнопку INTR, чтобы прекратить выполнение программы.

3.3.Используйте команду DDUMP, чтобы вывести на экран содержимое внутреннего ОЗУ данных. Наберите на клавиатуре DDUMP 20H и нажмите клавишу ENTER. Обратите внимание, что программа записывает значения 55Н, 66H и 77Н по адресам 20Н, 21Н и 22Н соответственно.

3.4.Используйте команду RDUMP, чтобы вывести на экран содержимое аккумулятора. Наберите на клавиатуре RDUMP E0H и нажмите кнопку ENTER.

Заметьте, что эта программа записывает последнее значение указателя адреса в аккумулятор.

**Эксперимент 4. Изучение**  примера программы доступа к внешней памяти данных и к памяти для хранения программ.

Программа, показанная на следующем экране, копирует первые байты управляющей программы в ячейки внешнего ОЗУ.

4.1.Введите программу, в память микрокомпьютера.

|  |  |  |
| --- | --- | --- |
| АДРЕС | КОМАНДА | КОММЕНТАРИЙ |
| E000H | MOV DPTR,#0000H | установка указателя адреса на 0000H |
| E003H | CLR A | очистка аккумулятора |
| E004H | MOVC A,@A+DPTR | читает первый байт кода |
| E005H | MOV DPTR,#E100H | установка указателя адреса на E100H |
| E008H | MOVX @DPTR,A | запись байта во внешнее ОЗУ |
| E009H | MOV DPTR,#0001H | установка указателя адреса на 0001H |
| E00CH | CLR A | очистка аккумулятора |
| E00DH | MOVC A,@A+DPTR | читает второй байт кода |
| E00EH | MOV DPTR,#E101H | установка указателя адреса на E101H |
| E011H | MOVX @DPTR,A | запись байта во внешнее ОЗУ |
| E012H | MOV DPTR,#0002H | установка указателя адреса на 0002H |
| E015H | CLR A | очистка аккумулятора |
| E016H | MOVC A,@A+DPTR | читает третий байт кода |
| E017H | MOV DPTR,#E102H | установка указателя адреса на E102H |
| E01AH | MOVX @DPTR,A | запись байта во внешнее ОЗУ |
| E01BH | LJMP E01BH | остановка |

4.2.Запустите программу. Нажмите кнопку INTR, чтобы остановить выполнение программы.

4.3.Используйте команду DASM, чтобы сравнить адреса ПЗУ с 0000H по 0002H и адреса ОЗУ с E100H по E102H соответственно.

4.4.Выключите стенд. Отключите ЕВ-153 от лабораторного стенда

**Оформите отчет и ответьте на контрольные вопросы**

### ???Контрольные вопросы

1. Как можно прочитать код программы из памяти для хранения программ?

* С помощью команды MOV.
* С помощью команды MOVХ.
* С помощью команды MOVC.

1. С помощью какой команды программа осуществляет доступ к данным во внутренней памяти?

* Используя команды MOV.
* Используя команды RDUMP.
* Используя команды MOVX.

1. С помощью какой команды программа осуществляет доступ к данным из внешней памяти?

* Используя команды MOV.
* Используя команды MOVХ
* Используя команды DASM.

1. Какие сигналы активируются во время передачи кода из памяти для хранения программ?
   * ANSWER

* PSEN
* CLEAR ENTRY
* INTR

1. Какие сигналы активизируются во время передачи внешних данных?

* ANSWER
* RD и WR
* PSEN
* INTR

**Лабораторная работа № 13.**

Исследование работы программируемого таймера ЕВ154 (урок 3) 13

**Цель работы научиться:**

* Согласовывать микросхему программируемого интервального таймера 8253 с микрокомпьютером на основе 8051.
* Использовать таймер для генерации различных сигналов;

- Использовать цифровые сигналы, для управления счетчиком

**Пояснения к работе**

*Краткие теоретические сведения*

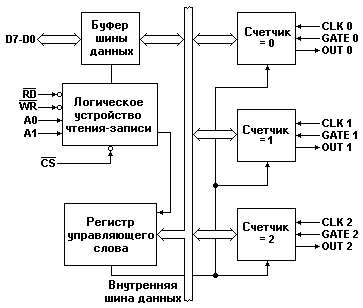
Intel 8253 это программируемый счетчик/генератор, разработанный для использования в качестве периферии микропроцессора.

Блок-диаграмма 8253 представлена на следующей странице.

8253 организован как три независимых 16-разрядных счетчика, каждый со скоростью пересчета до 2МГц.

Все режимы работы являются программируемыми.

В общем, он функционирует как многофункциональный счетчик, который рассматривается системным программным обеспечением как массив портов ввода/вывода.



8253 организован как три независимых 16-разрядных счетчика, каждый со скоростью пересчета до 2МГц.

Все режимы работы являются программируемыми.

В общем, он функционирует как многофункциональный счетчик, который рассматривается системным программным обеспечением как массив портов ввода/вывода.

8253 решает одну из основных проблем любой микрокомпьютерной системы - генерирование точных временных задержек под программным управлением.

Вместо установки циклических повторений в программном обеспечении системы, программист конфигурирует 8253 в соответствии со своими требованиями, инициализирует один из его счетчиков на желаемое число, и затем командует 8253 начать отсчет задержки и прервать работу ЦПУ, когда эта задача выполнится.

Нетрудно заметить, что таким образом уменьшаются расходы на программное обеспечение. Кроме этого существует возможность использовать несколько задержек путем назначения уровней приоритета.

Другие функции 8253, которые не связаны с задержками, но также часто применяются в большинстве микрокомпьютеров:

* Программируемый генератор скорости передачи
* Счетчик событий
* Двоичный умножитель частоты
* Часы реального времени
* Цифровой одновибратор
* Сложный контроль электродвигателя

**Приборы и оборудование:**

**-** Учебный стенд PU-2000

* Печатная плата EB-153
* Набор проводов DL-20
* Двухканальный осциллограф

### Порядок выполнения работы

1.Поместите плату ЕВ-153 на стенд PU-2000 и вставьте ее в разъем. Соедините ЕB-153 с EB-154.

2.С помощью ленточного кабеля, подключите разъем расширения платы EB-153 к одному из разъемов расширения платы EB-154.

3.Включите стенд.

**Эксперимент 1.** Производить таймером 0 деление частоты на четыре

1.1.Соедините СK0 (тактовый вход Таймера 0) со встроенным генератором тактовых импульсов с частотой 1MГц.

1.2.Подключите первый канал осциллографа для измерения сигнала Out0 (0 выход).

1.3.Подключите второй канал осциллографа для измерения сигнала CK0.

1.4.Подайте на G0 (синхр. вход таймера 0) сигнал высокого логического уровня.

1.5.Введите программу.

АДРЕС КОМАНДА КОММЕНТАРИЙ

E000H MOV DPTR,#7007H Устанавливает указатель адреса в

регистр режима

E003H MOV A,#3EH Определить режим 3 - Прямоугольный

импульс

E005H MOVX @DPTR,A Записывает режим

E006H MOV DPTR,#7008H Устанавливает указатель адреса на

Таймер 0

E009H MOV A,#04H Определяет младший значащий байт

E00BH MOVX @DPTR,A Записывает в T0

E00CH MOV A,#00H Определяет старший значащий байт

E00EH MOVX @DPTR,A Записывает в T0

E00FH LJMP E00FH Цикл

1.5.Выполните программу и проследите за прямоугольными импульсами на первом канале осциллографа.

Обратите внимание на то, что выходной сигнал будет оставаться в состоянии высокого уровня, до тех пор, пока не пройдет половина счета. Затем выходной сигнал перейдет в состояние низкого уровня.

1.6.Повторите эксперимент при другой частоте, изменив значение младшего байта, находящегося по адресу E00AH.

1.7.Нажмите кнопку RESET, чтобы остановить программу.

**Эксперимент 2.** В этом эксперименте Таймер 1 используется в качестве генератора

2.1.Соедините СK1 (тактовый вход Таймера 1) со встроенным генератором тактовых импульсов с частотой 1MГц.

2.2.Подключите первый канал осциллографа для измерения сигнала OUT1 (выход Таймера 1)

2.3.Подайте на G1 (синхр. вход таймера 1) сигнал высокого логического уровня.

2.4. Введите программу.

АДРЕС КОМАНДА КОММЕНТАРИЙ

E000H MOV DPTR,#7007H Устанавливает указатель адреса на

регистр режима

E003H MOV A,#74H Определяет режим 2

E005H MOVX @DPTR,A Записывает режим

E006H MOV DPTR,#7009H Устанавливает указатель адреса на

Таймер 1

E009H MOV A,#06H Определяет младший бит

E00BH MOVX @DPTR,A Записывает в T0

E00CH MOV A,#00H Определяет старший бит

E00EH MOVX @DPTR,A Записывает в T0

E00FH LJMP E00FH Цикл

2.5.Выполните программу и посмотрите на форму сигнала на экране осциллографа.

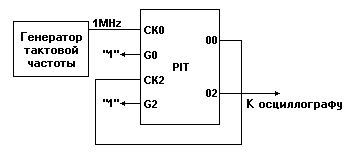
Заметьте, что на выходе будет сигнал низкого уровня в течение одного тактового периода.

2.6.Проверьте работу таймера на других частотах, изменяя программу.

2.7.Нажмите кнопку RESET, чтобы остановить программу.

**Эксперимент 3**. Использование Таймер 0 для уменьшения частоты, а Таймер 2 работает в режиме прерывания по окончании счета.

Схема подключения таймера:



3.1.Соедините СK0 (тактовый вход Таймера 0) со встроенным генератором тактовой частоты (1MГц).

3.2.Подключите первый канал осциллографа для измерения выходного сигнала 02.

3.3.Соедините выход O0 со входом CK2.

3.4.Подайте на G0 и G2 (синхр. входы таймеров 0 и 2) сигналы высокого логического уровня.

3.5.Введите программу, представленную на следующих страницах.

АДРЕС КОМАНДА КОММЕНТАРИЙ

E000H MOV DPTR,#7007H Устанавливает указатель адреса

на регистр режима

E003H MOV A,#3EH Определяет режим 3 –

Прямоугольный сигнал

E005H MOVX @DPTR,A Записывает режим

E006H MOV DPTR,#7008H Устанавливает указатель адреса

На счетчик 0

E009H MOV A,#04H Определяет младший байт

E00BH MOVX @DPTR,A Записывает в T0

E00CH MOV A,#00H Определяет старший байт

E00EH MOVX @DPTR,A Записывает в T0

E00FH LJMP E00FH Устанавливает указатель адреса на

регистр режима

(Продолжение)

АДРЕС КОМАНДА КОММЕНТАРИЙ

E012H MOV A,#B0H Определяет PIT Режим 0

E014H MOVX @DPTR,A Записывает режим

E015H MOV DPTR,#7006H Устанавливает указатель адреса на

Таймер 2

E018H MOV A,#80H Определяет младший бит

E01AH MOVX @DPTR,A Записывает в T2

E01BH MOV A,#00H Определяет старший бит

E01DH MOVX @DPTR,A Записывает в T2

E01EH LJMP E01EH Цикл

3.6.Выполните программу и посмотрите, что после нескольких секунд, выходной сигнал O2 принимает значение высокого уровня.

Заметьте, что выход Таймера 2 будет первоначально низкого уровня после установления режима.

После загрузки значения счета в выбранный регистр счетчика (значение счета находится по адресу E019H), на выходе остается сигнал низкого уровня, а счетчик продолжает счет.

Когда счет достигнет конца, выходной сигнал примет значение высокого уровня.

3.7.Нажмите кнопку RESET, чтобы остановить программу.

3.8.Выключите стенд. Отсоедините соединительные провода и кабели.

Извлеките платы EB-154 и EB-153.

●Оформите отчет и сделайте выводы по работе

Ответьте на контрольные вопросы

**??? Контрольные вопросы**

1. Каково назначение тактового входа в Режиме 2 - генератор частоты?
   * Он устанавливает режим регистра.
   * Он синхронизирует счетчик.
   * Он записывает управляющие слова в регистр управления.

2.Каким образом мы можем генерировать частоту 500 Гц, при входном сигнале с частотой 1 МГц?

* + Установив два счетчика в Режим 2, и разделив частоту на 2000 (один делит на 200, другой на 10).
  + Установив счетчик в Режим 3 для генерации прямоугольных колебаний частотой 500 МГц на выходе.
  + Установив счетчик в Режим 4, в котором тактовый импульс переключает счетчик.

3.Как образом должен быть подключен программируемый таймер, чтобы делить частоту на 100 000?

* + Нужно соединить три счетчика в каскад.
  + Нужно соединить два счетчика в каскад.
  + Требуется большее количество счетчиков.

4.В каком режиме программируемый таймер работает в качестве строба программной синхронизации?

* + Режим 0
  + Режим 1
  + Режим 2
  + Режим 4

**Лабораторная работа № 14**

Объединение периферийных устройств ЕВ 154 урок 7 14

**Цель работы научиться:**

* Использовать систему, состоящую из различных периферийных устройств.
* Анализировать работу такой системы.
* Соединять контроллер и его периферийные устройства с компьютером.

**Пояснения к работе**

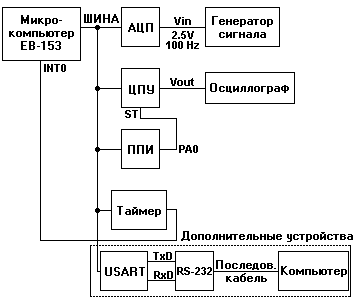
*Краткие теоретические сведения*

В этой лабораторной работе объединены большинство периферийных устройств, с которыми вы познакомились на предыдущих уроках.

Мы соединим между собой различные периферийные устройства, для записи аналогового сигнала:

1. Программируемый таймер будет генерировать прерывания каждые 500 микросекунд.
2. Микрокомпьютер будет распознавать прерывания и дискретизировать аналоговый сигнал с помощью АЦП.
3. Цифровое значение аналогового сигнала будет сохраняться в памяти микрокомпьютера.
4. Этот процесс повторится 1024 раз, и дискретизированный аналоговый сигнал будет сохраняться в буфере памяти.
5. На втором этапе мы будем восстанавливать сигнал с помощью ЦАП и наблюдать его на экране осциллографа.
6. Затем записанный нами сигнал может быть передан удаленному компьютеру через USART и интерфейс RS-232C.

На рисунке изображена соответствующая блок-схема.



**Приборы и элементы:**

* Учебный стенд PU-2000
* Печатная плата EB-153
* Набор соединительных проводов DL-20
* Двухканальный осциллограф
* Генератор сигналов

### Порядок выполнения работы

1. Поместите плату ЕВ-153 на стенд PU-2000 и вставьте ее в разъем.

2.Соедините EB-153 с EB-154.

3.С помощью ленточного кабеля, подключите разъем расширения платы EB-153 к одному из разъемов расширения платы EB-154.

4.Включите стенд

**Эксперимент 1.** Выполнение программы настройки системы, Программирование таймера на генерирование прерываний каждые 500 микросекунд

1.1.Подключите выход D0 Таймера 0 к входу INT0 на EB-153.

1.2.Подайте на вход CK0 Таймера 0 сигнал тактового генератора с частотой к 1MГц.

1.3.Подайте на вход G0 Таймера 0 сигнал высокого уровня.

1.4.Подайте на вход RANGE сигнал высокого уровня.

1.5.Подключите VIN к генератору сигналов, а VOUT к осциллографу.

1.6.Подключите вывод Порта A0 ППИ (Программируемого Периферийного Интерфейса) ко входу [overline|ST ] АЦП.

1.7.Введите программу.

АДРЕС КОМАНДА КОММЕНТАРИЙ

E000H MOV A,#80H Определяет управляющее слово

ППИ - режим 0

E002H MOV DPTR,#7003H Устанавливает указатель адреса на

регистр режима

E005H MOVX @DPTR,A Записывает управляющее слово ППИ

E006H MOV A,#00H Очищает Порт A - ST=0

E008H MOV DPTR,#7000H Устанавливает указатель адреса

Порта А

E00BH MOVX @DPTR,A Записывает в Порт А

E00CH MOV A,#3EH Определяет Режим Таймера 3 –

Прямоугольная волна

E00EH MOV DPTR,#7007H Устанавливает указатель адреса

Таймера

E011H MOVX @DPTR,A Записывает управляющее слово

Таймера

E012H MOV DPTR,#7008 Устанавливает указатель Таймера 0

E015H MOV A,#F4H Определяет младший разряд

E017H MOVX @DPTR,A Записывает в T0

E018H MOV A,#01H Определяет старший разряд

E01AH MOVX @DPTR,A Записывает в T0 - 1MГц/2KГц =

01F4H

E01BH MOV R0,#00H Очищает указатель буфера –

младший разряд

E01DH MOV DPTR,#FF00H Определяет адрес перехода INT0

E020H MOV A,#02H Определяет команду LJMP E100H

E022H MOVX @DPTR,A Записывает код - LJMP = 02H

E023H MOV A,#E1H Определяет старший байт адреса –

E1H

E025H INC DPTR Инкрементирует указатель

E026H MOVX @DPTR,A Записывает стар. байт адреса

E027H MOV A,#00H Младший байт адреса - 00H

E029H INC DPTR Инкрементирует указатель

E02AH MOVX @DPTR,A Записывает млад. байт адреса

E02BH MOV R1,#E2H Очищает указатель буфера - старший

байт

E02DH MOV 88H,#01H Устанавливает INT0 прерыванием

E030H MOV A8H,#81H Открывает INT0 = IE=1000001B

E033H LJMP E033H Ждет прерывания

E100H MOV DPTR,#7000H Устанавливает указатель ППИ

E103H MOV A,#01H ST = 1

E105H MOVX @DPTR,A Начинает преобразование

E106H MOV A,#00H ST = 0

E108H MOVX @DPTR,A Заканчивает подачу импульса ST

E109H MOV DPTR,#700AH Устанавливает указатель АЦП

E10CH MOVX A,@DPTR Получает данные из АЦП

E10DH MOV 82H,R0 DPL = R0

E10FH MOV 83H,R1 DPH = R1

E111H MOVX @DPTR,A Сохраняет данные из АЦП

E112H INC DPTR Инкрементирует буфер указателя

E113H MOV R0,82H R0 = DPL

E115H MOV R1,83H R1 = DPH

E117H MOV A,R1 Получает старший байт адреса

E118H CJNE A,#E6H,03H Сравнивает длину буфера с 400H

E11BH LJMP E11FH Переход на конец программы

E11EH RETI Возврат из прерывания

E11FH MOV 90H,#00H Выключает светодиоды - P1=0

E122H LJMP E122H Ждет сброса

1.8.Установите на выходе генератора синусоидальный сигнал с частотой 100Гц и размахом от 0V до 2V.

1.9.Выполните программы.

Заметьте, что программа будет использовать частоту дискретизации 2кГц для сигнала частотой 100Гц.

Прерывание микропроцессора происходит, каждые 500 микросекунд. При заполнении буфера, гаснет светодиод Порта 1.

1.10.Подождите, пока не погаснет светодиод Порта 1, а затем нажмите кнопку RESET.

1.11.Проверьте буфер памяти, содержащий сохраненный сигнал (адрес E200H - E5FFH).

**Эксперимент 2.**Выполнение программы обработки прерываний

2.1.Имените программу обработки прерываний.

Введите представленную ниже программу.

АДРЕС КОМАНДА КОММЕНТАРИЙ

E100H MOV 82H,R0 DPL = R0

E102H MOV 83H,R1 DPH = R1

E104H MOVX A,@DPTR Получает данные ЦАП

E105H OV R2,A Сохраняет данные ЦАП в R2

E106H INC DPTR Инкрементирует указатель буфера

E107H MOV R0,82H R0 = DPL

E109H MOV R1,83H R1 = DPH

E10BH MOV A,R1 Получает старший байт адреса

E10CH CJNE A,#E6H,02H Сравнивает длину буфера с 400H

E10FH MOV R1,#E2H Устанавливает старший байт адреса

E111H MOV DPTR,#700AH Устанавливает указатель ЦАП

E114H MOV A,R2 Получает данные ЦАП

E115H MOVX @DPTR,A Выводит данные АЦП

E116H RETI Возврат из прерывания

Это программа воспроизводит, с помощью ЦАП, сигнал, записанный в памяти.

2.2.Запустите программу, которая регистрирует аналоговый сигнал, и программу обработки прерываний для отображения сигнала.

2.3.Проследите за аналоговым сигналом, производимым микрокомпьютером, на экране осциллографа.

2.4.Усовершенствуйте программу так, чтобы она передавала данные удаленному компьютеру через устройство USART.

2.5.С помощью интерпретатора языка БЕЙСИК, наберите и запустите программу, представленную ниже.

10 OPEN "COM1:9600,N,8,1,RS.CS.DS.CD" AS #1

20 IF EOF (1) = -1 THEN 20 ELSE 30

30 A$ = INPUT$(1,#1): PRINT A$,: GOTO 20

Эта программа выводит на экран символы, полученные из последовательного порта.

Задержка между передающимися байтами позволяет избежать переполнения буфера на медленных компьютерах.

2.6.Запустите программу и пронаблюдайте за символами, появляющимися на экране.

2.7.Нажмите кнопку RESET на EB-153 и EB-154. Выключите стенд.

2.8.Извлеките соединительные провода и кабели из EB-153 и EB-154 и сами платы из стенда.

●Оформите отчет и сделайте выводы по работе.

Ответьте на контрольные вопросы

**??? Контрольные вопросы**

1. Могут ли регистрируемые сигналы выводится на экран со скоростью отличной от скорости записи?
   * Да, при изменении скорости передачи данных.
   * Да, при изменении частоты прерываний.
   * Да, при изменении периода дискретизации.

1. Зависит ли разрешение отображаемого сигнала от частоты входного сигнала?
   * Нет, оно не зависит от частоты дискретизации и максимальной частоты входного сигнала.
   * Да, оно зависит от частоты дискретизации и максимальной частоты входного сигнала.
   * Да, оно зависит от скорости передачи сигнала и частоты дискретизации.

1. Зависит ли разрешение отображаемого сигнала от частоты дискретизации?
   * Нет, оно зависит только от частоты сигнала.
   * Нет, оно не зависит от частоты дискретизации.
   * Да, оно зависит от частоты дискретизации и максимальной частоты входного сигнала.

**Лабораторная работа № 15**

Исследование работы параллельного контроллера ввода-вывода

**Цель работы:**

* Понимать работу параллельного контроллера ввода-вывода 8255.

**Пояснения к работе**

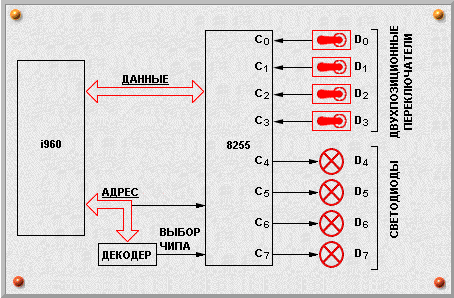
*Краткие теоретические сведения*

Плата EB-160 (4,15)используют микросхему параллельного ввода-вывода Intel 8255, который снабжен тремя независимыми 8-битовыми портами, которые, могут быть запрограммированы для работы входами, или выходами.

В EB-160 эти три порта используются следующим образом:

|  |  |  |  |
| --- | --- | --- | --- |
| Порт I/O 8255 | Размещение на EB-160 | Тип ввода-вывода | Функции на плате EB-160 |
| Порт A | Внешний разъем | Вход или Выход | Разъем расширения |
| Порт B | Внешний разъем | Вход или Выход | Разъем расширения |
| Порт C | Биты 0-3 | Только Вход | Набор переключателей |
| Порт C | Биты 4-7 | Только Выход | Светодиоды 1-3 |

***Рис. 4.1: Параллельный порт ввода вывода***



**Основы порта 8255**

8255 – универсальное устройство ввода-вывода, которое может быть сконфигурировано для работы в различных режимах. На плате EB-160 количество и типы режимов ограничены в соответствии с конфигурацией аппаратного обеспечения по вышеупомянутой таблице.

На плате EB-160 адресные выводы A0 и A1 микросхемы 8255 связаны с адресными шинами A1 и A2. Это необходимо для того, чтобы позволить микропроцессору i960 иметь два отдельных сигнала BE0 и BE1, используемые для выбора верхнего и нижнего кристалла памяти.

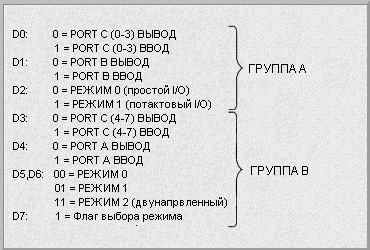
|  |  |
| --- | --- |
| Порт | Адрес |
| A | 4000 0000 |
| B | 4000 0002 |
| C | 4000 0004 |
| Регистр управления | 4000 0006 |

В EB-160 используется только РЕЖИМ 0 (MODE 0), который обеспечивает вызов базовой операции ввода-вывода устройства. Конфигурирование РЕЖИМА 0 выполняются автоматически платой. В наших упражнениях мы будем пользоваться только портами.

Порты A и B могут конфигурироваться программистом. Через регистр управления задается направление работы порта. Чтение и запись данных из/в порты, выполняется через регистры данных, выделенные каждому порту.

**Регистр управления**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |



Имеется возможность чтения или записи только одного бита в порту C с использованием управляющего слова установки/сброса бита.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

|  |  |
| --- | --- |
| D0: | 0 = СБРОС ВЫБРАННОГО БИТА НА 0 |
|  | 1 = УСТАНОВКА ВЫБРАННОГО БИТА НА 1 |
| D1, D2, D3: | 000 = ВЫБОР C0 |
|  | 001 = ВЫБОР C1 |
|  | 010 = ВЫБОР C2 |
|  | 011 = ВЫБОР C3 |
|  | 100 = ВЫБОР C4 |
|  | 101 = ВЫБОР C5 |
|  | 110 = ВЫБОР C6 |
|  | 111 = ВЫБОР C7 |
| D2: | 0 = Флажок установки/сброса бита |

**Приборы и элементы:**

- Плата EB-160 с последовательным кабелем

- Рабочая станция EB-2000

- Установленное программное обеспечение для платы EB-160

- Вольтметр или осциллограф

**Порядок выполнения работы**

**Начальная установка**

Проверьте, что EB-160 подключен к EB-2000 и последовательный кабель подключен к последовательному порту персонального компьютера (COM1 или COM2).

**Включите источник питания.**

1. Чтобы запустить программу отладчика, нажмите на иконку db960, расположенную в правом нижнем углу экрана.
2. Введите команду "include exer4.inc". Эта команда запускает специальный файл, который автоматически выполнит заранее заданный набор команд отладчика: загрузит программу, установит контрольные точки и запустит программу до достижения первой контрольной точки.
3. Если Вы наберете "type exer4.inc", то получите в ответ распечатку

load exer4 ;загружает программу для отладки

cache = off ;отключает кэш

br[0] = :exer4.main.L01 ;устанавливает первую контрольную точку на строке помеченной L01

br[1] = :exer4.main.L02 ;устанавливает вторую контрольную точку

br[2] = :exer4.main.L03

br[3] = :exer4.main.L04

wa[0] = Switches ;добавляет переменную названную Switches на окно просмотра

wa[1] = Leds ;добавляет Leds на окно просмотра

go ;запускает программу (F5)

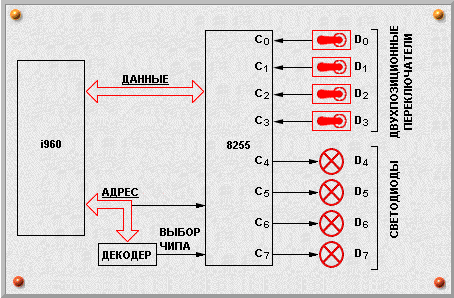
1. Проверьте, что программа загрузилась, и что ее выполнение остановилось на контрольной точке L01 внутри основной подпрограммы main( ).
2. Откройте и измените размеры окна просмотра переменных, используя пункт меню view/watch.

**Тестирование светодиодов**

1. Программа остановилась на первой контрольной точке L01. Просмотрите код. Заметьте, что значение 0xEF напрямую пишется в порту C устройства ввода/вывода 8255.
2. Если Вы предпочитаете просматривать программу на ассемблере, нажмите на F3. Вы можете открыть окно для просмотра переменных, чтобы видеть состояние "Switches" и "Leds". Также, Вы можете проверить значение ячейки памяти 10022960, для того чтобы увидеть адрес порта C - 0x40000004, который сохранен в g4.
3. Значение 0xEF, которое хранится в g5, означает, что все биты, за исключением бита 4, активны (1). Чтобы включить светодиод, этот бит должен быть нулевым. Также Вы может открыть окно регистров, чтобы просмотреть содержимое g4 и g5 в процессе работы программы.
4. Теперь выполните один шаг с помощью F8 и пронаблюдайте, что LED#1 загорелся вследствие того, что бит 4 стал нулевым.
5. Выполните пошагово оставшуюся часть программы и пронаблюдайте, как сброс значения бита вызывает зажигание соответствующего светодиода.
6. Пошагово дойдите до L02 или нажмите F5.
7. Следующая программа демонстрирует бегущие огни светодиодов. По 25 итераций в цикле, каждый из светодиодов загорается по очереди с некоторой временной задержкой между загораниями. Просмотрите код программы и нажмите F5 для выполнения.

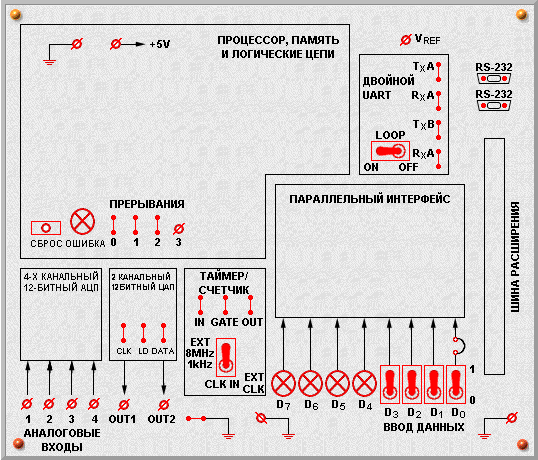
**Тестирование переключателей**

1. В этом разделе мы будем изучать двухпозиционные переключатели, и обрабатывать их состояния. Для начала, удостоверьтесь, что ваша программа остановлена в точке L03. Если нет, дойдите до нее пошагово или нажмите на F5.
2. Переключатели D0-D3 физически связаны с битами 0-3 на порту C. Чтение состояния тумблеров включает в себя чтение байта из порта C и маскировка незначащих старших битов. Разомкнутое состояние переключателя, т. е. нижнее положение, соответствует нулевому уровню сигнала на порту C.

****

***Рис. 1. Параллельный порт ввода-вывода***

1. Просмотрите часть программы после метки L03 и убедитесь, что она действительно читает байт из порта C. Если необходимо, то с помощью клавиши F3 посмотрите программу на ассемблере для этой команды.
2. Выключите переключатель D0, переводя его рычаг в нижнее положение. Включите переключатели D1, D2 и D3, установив рычаги переключателей в верхние позиции.
3. Подключите соединительный кабель в разрыв рядом с переключателем D0, как показано на схеме соединения.



***Рис.2.Схема соединений***

1. В отладчике откройте окно для просмотра переменных. В нем Вы будете наблюдать за изменением переменной Switches в процессе работы программы.
2. Нажмите на F10, чтобы выполнить текущую строку.
3. Обратите внимание на значение переменной Switches и убедитесь, что она корректно отображает состояние переключателей - D0 включен, а другие – выключены.
4. По своему выбору установите позиции переключателей. Вычислите ожидаемый результирующий набор битов 0-3. Затем переходите к следующим командам и проследите за изменением переменной.
5. Также Вы может измерить логический уровень каждого переключателя, используя вольтметр или осциллограф.
6. Сверьте ваши результаты. Повторяйте, пока не достигнете метки L04.

**Тестирование переключателей и светодиодов**

1. В этом разделе мы будем читать информацию о состоянии переключателей и в соответствии с ней будем включать или выключать светодиоды.
2. В данный момент Ваша программа должна находится на метке L04.
3. Проследите за программой. Первая строка читает состояние переключателей в порт С и маскирует незначащие биты. Затем каждый бит, соответствующий определенному переключателю сравнивается с определенным значением, и если значения равны, то изменяется соответствующий бит переменной LED. Затем осуществляется единственная запись в порт C обеспечивающая включение светодиодов, соответствующих замкнутым переключателям.
4. Установите переключатели в произвольные позиции по своему усмотрению. При этом помните, что свечение светодиода обеспечивает нижнее положение переключателя.

ЗАМЕЧАНИЕ:

Так как в цикле имеется контрольная точка, сначала измените положение переключателя, а затем нажмите на F5. Вы должны увидеть изменение в состоянии светодиодов.

Сравните соответствие позиций переключателей и светодиодов. Повторите эту процедуру для различных комбинаций переключателей.

●Оформите отчет и сделайте выводы по работе.

Ответьте на контрольные вопросы

**??? Контрольные вопросы**

1. В нашем эксперименте, светодиоды были зажжены в следствии:

* Светодиоды напрямую соединены с переключателями. Выключенное состояние переключателя включает светодиод.
* Программа читает состояния переключателей и выборочно включает и отключает светодиоды.
* Светодиоды не могут быть зажжены переключателями, так как они не соединены.
* Под управлением программы светодиоды могут быть включены и выключены по желанию.

2.Какое из следующих утверждений является истинным для порта C контроллера 8255 на плате EB-160:

* + Для управления светодиодами были использованы все биты.
  + Все биты были использованы, чтобы читать состояния тумблерных переключателей.
  + Чтобы управлять светодиодами требуется чтение из порта.
  + По одному биту используется для каждого светодиода и тумблера.

3.Каждый вывод порта 8255 может программироваться, чтобы работать как:

* + Только вход.
  + Только выход.
  + Вход или выход.
  + Ни один из вышеперечисленных.

4.Какое из следующих утверждений является правильным для порта C устройства ввода/вывода 8255 на плате EB-160?

* Для управления светодиодами используются все биты.
* Для чтения состояний переключателей используются все биты.
* Чтобы управлять светодиодами, требуется чтения от порта.
* Для каждого светодиода и переключателя используется по одному биту.

**Лабораторная работа № 16.**

Создание интерфейс аналоговых сигналов с микропроцессором(ЕВ154 урок 5)

**Цель работы научиться:**

* Анализировать сигналы аналоговых устройств.
* Создавать интерфейс аналоговых сигналов с микропроцессором.
* Преобразовывать аналоговые входные сигналы в цифровые.
* Преобразовывать цифровые сигналы в аналоговые.

**Пояснения к работе**

*Краткие теоретические сведения*

Цель данной лабораторной работе познакомить обучающихся с понятиями и техникой интерфейса цифровых компьютеров с аналоговыми электронными устройствами.

Для того, чтобы микрокомпьютеры приносили пользу в реальной жизни, их необходимо подключить, что называется, к "внешнему миру".

Проектирование программного обеспечения для практической реализации интерфейсов между микрокомпьютером и внешним оборудованием, является, возможно, самой большой трудностью, с которой вам придется столкнуться.

Микрокомпьютер EB-154 обеспечивает все необходимое для связи микропроцессора с внешними аналоговыми устройствами.

Он содержит полную 8-разрядную аналоговую систему ввода-вывода на одной микросхеме:

* АЦП (аналого-цифровой преобразователь) работающий по принципу высокоскоростного последовательного приближения, со временем преобразования 2 мс, частотой преобразования 200 кГц;
* ЦАП (цифро-аналоговый преобразователь) и буферный усилитель, со временем установки 1 мс и стабильным опорным напряжением.

Цифровой интерфейс осуществляется посредством 8-разрядной шины данных и стандартной шины управления.

Раздельное включение всех процессов преобразования, позволяет управлять частотой преобразования и АЦП, для точно контролируемого периода оцифровки

***ЦИФРО-АНАЛОГОВОЕ ПРЕОБРАЗОВАНИЕ***

Цифро-аналоговые преобразователи (ЦАП) могут различаться по виду выходного сигнала: с токовым выходом и выходом в виде напряжения.

Выходной сигнал прямо пропорционален значению входного цифрового сигнала, обычно двоичного.

Большинство ЦАП резисторные и используют "бинарновзвешенную" комбинацию резисторов для деления опорного напряжения R, 2R, 4R, 8R и т.д. или многозвенную схему R-2R.

Многозвенная схема со значениями R и 2R предпочтительнее из-за простоты производства и лучших электрических характеристик.

Очень распространены ЦАП с токовым выходом. Их преимущество - это малое время установки, обычно несколько сотен наносекунд.

***АНАЛОГО-ЦИФРОВОЕ ПРЕОБРАЗОВАНИЕ***

Аналого-цифровые преобразователи (АЦП или А/Ц преобразователи) играют важную роль во многих компьютерных системах.

Они позволяют цифровым системам преобразовывать аналоговый сигнал в цифровую величину.

Принцип работы большинства типов АЦП основан на сравнении входного напряжения с напряжением ЦАП.

При создании компьютерного интерфейса обычно применяются преобразователи с линейной трансформацией сигнала или использующие последовательное приближение.

Пилообразный АЦП, работает по принципу сравнения неизвестного напряжения, с напряжением монотонно возрастающего пилообразного сигнала.

**Приборы и элементы:**

Для проведения этого эксперимента, вам понадобится следующее оборудование:

* Учебный стенд PU-2000
* Печатная плата EB-153
* Набор проводов DL-20
* Двухканальный осциллограф

**Порядок выполнения работы**

**Установка**

1.Поместите плату ЕВ-153 на стенд PU-2000 и вставьте ее в разъем.

2.Соедините ЕB-153 с EB-154.

3.С помощью ленточного кабеля, подключите разъем расширения платы EB-153 к одному из разъемов расширения платы EB-154.

4.Включите стенд.

**Порядок работы**

В роли аналогового сигнала будет выступать гармонический сигнал.

1.Соедините осциллограф для измерения сигнала Vout и подсоедините вход RANGE к "общему проводу".

2.Введите программу, представленную на следующей странице.

АДРЕС КОМАНДА КОММЕНТАРИЙ

E000H MOV DPTR,#700AH Устанавливает адрес ЦАП

E003H MOV A,#00H Очистка

E005H MOVX @DPTR,A Записывает в ЦАП

E006H INC A Увеличивает счет

E007H LJMP E005H Цикл

Выполните программу и проследите за тем, как компьютер управляет ЦАП для генерации гармонического сигнала на выходе Vout.

Низкий уровень выходного сигнала равен 0 V , а высокий уровень +1.25 V.

Подсоедините вход RANGE к логической "1" и посмотрите на новое значение сигнала высокого уровня (+2.5V).

**Эксперимент 1**

Следующая программа считывает аналоговое напряжение, при помощи АЦП, и выводит его значение на ЖК-дисплей.

1.Соедините PA0 (находится на плате EB-154) с [overline|ST ] входом АЦП.

Сигнал PA0 этого порта разрешит начать преобразование АЦП.

2.Соедините вход RANGE преобразователя с логической "1".

АДРЕС КОМАНДА КОММЕНТАРИЙ

E000H MOV A,#80H Определяет управляющее слово

Режим 0

E002H MOV DPTR,#7003H Устанавливает указатель адреса на

регистр режима.

E005H MOVX @DPTR,A Записывает управляющее слово

E006H MOV A,#00H Очищает аккумулятор

E008H LCALL FF80H Очищает дисплей

E00BH MOV DPTR,#7000H Устанавливает адресный указатель

Порта A

E00EH MOV A,#01H PA0 = 1

E010H MOVX @DPTR,A Записывает в Порт A

E011H MOV A,#00H PA0 = 0

E013H MOVX @DPTR.A Начинает преобразование

E014H MOV DPTR,#700AH Устанавливает адрес АЦП

E017H MOVX A,@DPTR Читает из АЦП

E018H MOV DPTR,#E060H Устанавливает временный адрес для

хранения

E01BH MOVX @DPTR,A Записывает данные в E060H

E01CH MOV F0H,#0AH Устанавливает B = 10D (десятич.)

E01FH DIV AB Делит на 10D

E020H MOV A,F0H Получает единицы

E022H ADD A,#30H Преобразует в ASCII

E024H MOV DPTR,#FF92H Устанавливает адрес дисплея

E027H MOVX @DPTR,A Записывает единицы в буфер

дисплея

E028H MOV DPTR,#E060H Устанавливает временный АЦП

адрес

E02BH MOVX A,@DPTR Получает АЦП данные из E060H

E02CH MOV F0H,#A0H Устанавливает B = 10D

E02FH DIV AB Делит на 10D

E030H MOV F0H,#0AH Устанавливает B = 10D

E033H DIV AB Делит на 10D

E034H MOV A,#F0H Получает десятки

E036H ADD A,#30H Преобразует в ASCII

E038H MOV DPTR,#FF91H Устанавливает адрес дисплея

E03BH MOVX @DPTR,A Устанавливает десятки в буфер

дисплея

E03CH MOV DPTR,#E060H Устанавливает временный АЦП

адрес для хранения

E03FH MOVX A,@DPTR Получает данные АПЦ из E060H

E040H MOV F0H,#64H Устанавливает B = 10D

E043H DIV AB Делит на 10D

E044H MOV F0H,#0AH Устанавливает B = 10D

E047H DIV AB Делит на 10D

E048H MOV A,#F0H Получает сотни

E04AH ADD A,#30H Преобразует в ASCII

E04CH MOV DPTR,#FF90H Устанавливает адрес дисплея

E04FH MOVX @DPTR,A Сохраняет сотни в буфере дисплея

E050H LCALL FF83H Отображает буфер

E053H MOV A,#04H Определяет константу задержки

E055H MOV DPTR,#FFC0H Устанавливает указатель задержки

E058H MOVX @DPTR,A Устанавливает задержку

E059H LCALL FF8CH Вызывает программу задержки

E05CH LJMP E00BH Цикл

3.Выполните программу и подключите вход Vin АЦП по очереди к GND и VCC.

4.Проследите за цифровым значением на ЖК-дисплее.

5.Подключите выход Vin и источнику регулируемого напряжения и проверьте цифровые значения, показываемые ЖК-дисплеем, соответствующие входному аналоговому сигналу в диапазоне от 0 V до 2.5 V.

**Эксперимент 2**

Прием информации может осуществляться при помощи АЦП, управляемого микрокомпьютером: для вывода цифрового значения сигнала на дисплей и, ЦАП, для вывода аналогового сигнала на экран осциллографа.

2.1.Подключите осциллограф к Vout и Vin.

2.2.Установите чувствительность вертикальной развертки осциллографа на 0.5 V/дел.

2.3.Соедините PA0 (находится на плате EB-154) с входом , а вход RANGE преобразователя с логической "1".

2.4.Введите программу:

АДРЕС КОМАНДА КОММЕНТАРИЙ

E000H MOV A,#80H Определяет управляющее слово –

Режим 0

E002H MOV DPTR,#7003H Устанавливает адресный указатель

на регистр режима

E005H MOVX @DPTR,A Записывает управляющее слово

E006H MOV A,#00H Очищает аккумулятор

E008H LCALL FF80H Очищает дисплей

E00BH MOV DPTR,#E070H Устанавливает адрес счетчика

E00EH MOVX A,@DPTR Читает из временного адреса счета

E00FH INC A Увеличивает счет

E010H MOVX @DPTR,A Сохраняет счет

E011H MOV DPTR,#700AH Устанавливает адрес ЦАП

E014H MOVX @DPTR.A Записывает в ЦАП

E015H MOV DPTR,#7000H Устанавливает адресный указатель

Порта A

АДРЕС КОМАНДА КОММЕНТАРИЙ

E018H MOV A,#01H PA0 = 1

E01AH MOVX @DPTR,A Записывает в Порт A

E01BH MOV A,#00H PA0 = 0

E01DH MOVX @DPTR,A Начинает преобразование

E01EH MOV DPTR,#700AH Устанавливает АЦП адрес

E021H MOVX A,DPTR Читает из АЦП

E022H MOV F0H,#0AH Устанавливает B = 10D

E025H DIV AB Делит на 10D

E026H MOV A,F0H Получает единицы

E028H ADD A,#30H Преобразует в АSCII

E02AH MOV DPTR,#FF92H Устанавливает адрес дисплея

E02DH MOVX @DPTR,A Сохраняет единицы в буфере

дисплея

E02EH MOV DPTR,#700AH Устанавливает адрес АЦП

E031H MOVX A,@DPTR Читает АЦП

E032H MOV F0H,#0AH Устанавливает B = 10D

E035H DIV AB Делит на 10D

E036H MOV F0H,#0AH Устанавливает B = 10D

E039H DIV AB Делит на 10D

E03AH MOV A,#F0H Получает десятки

E03CH ADD A,#30H Преобразует в АSCII

E03EH MOV DPTR,#FF91H Устанавливает адрес дисплея

E041H MOVX @DPTR,A Сохраняет десятки в буфере дисплея

E042H MOV DPTR,#700AH Устанавливает адрес АЦП

E045H MOVX A,@DPTR Читает АЦП

E046H MOV F0H,#64H Устанавливает B = 100D

E049H DIV AB Делит на 100D

E04AH MOV F0H,#0AH Устанавливает B = 10D

E04DH DIV AB Делит на 10D

E04EH MOV A,#F0H Получает сотни

E050H ADD A,#30H Преобразует в ASCII

E052H MOV DPTR,#FF90H Устанавливает адрес дисплея

E055H MOVX @DPTR,A Сохраняет сотни в буфере дисплея

E056H LCALL FF83H Отображает буфер

E059H MOV A,#01H Определяет константу задержки

E05BH MOV DPTR,#FFC0H Устанавливает указатель задержки

E05EH MOVX @DPTR,A Устанавливает задержку

E05FH LCALL FF8CH Вызывает функцию задержки

E062H LJMP E00BH Цикл

2.5.Выполните программу и проследите за изменением аналогового сигнала на экране осциллографа, и соответствующим ему цифровым сигналом на ЖК-дисплее.

2.6.Нажмите кнопку RESET на EB-153 и EB-154, чтобы остановить программу.

2.7.Выключите стенд.

* Оформите отчет и сделайте выводы по работе

**??? Контрольные вопросы**

1. Каким образом ЦАП генерирует треугольную волну?
   * Значение цифрового сигнала будет установлено на максимум, а затем по мере уменьшения, каждый раз на единицу, достигнет своего минимального значения.
   * Значение цифрового сигнала будет устанавливаться на максимум, а затем на минимум.
   * Значение цифрового сигнала будет увеличиваться на единицу, пока не достигнет максимума, а затем будет уменьшаться на единицу, пока не достигнет минимального значения.

1. Какое значение цифрового сигнала, полученного на выходе 8-разрядного АЦП, соответствующее максимально возможному напряжению?
   * FFH
   * 0100H
   * 00H

1. Какое значение цифрового сигнала, полученного на выходе 8-разрядного АЦП, соответствующее нулевому входному напряжению?
   * 0100H
   * FFH
   * 00H

**Лабораторная работа № 17.**

**Знакомство со Средой Отладчика ЕВ 160**

**Цель работы:**

- Включать плату и проверять, что соединение к отладчику осуществлено.

- Проверять регистры микропроцессора.

- Проверять ячейки памяти.

- Изменять ячейки в памяти.

**Пояснения к работе**

*Краткие теоретические сведения*

Плата EB-160 основана на микропроцессоре Intel i960. В противоположность стандартному CISC микропроцессору, RISC микропроцессор имеет меньшее число команд, что реализуется более простыми аппаратными средствами, но их выполнение происходит значительно быстрее. Аббревиатура RISC расшифровывается как Reduced Instruction Set Computing (Процессор с сокращенным набором команд).

RISC популярен в ресурсоемких приложениях, таких как: передача данных, численный анализ и в сложных замкнутых системах управления

В этом курсе мы рассмотрим следующие темы:

- Базовая архитектура микропроцессора i960.

- Некоторые команды микропроцессора i960, предназначенные для перемещения данных и выполнения арифметических операций.

- Программирование ресурсов платы EB-160, таких как цифро-аналоговый преобразователь, аналого-цифровой преобразователь, таймер, последовательный порт ввода-вывода и т.д.

**Приборы и элементы:**

- Плата EB-160 с последовательным кабелем

- Рабочая станция EB-2000

- Установленное программное обеспечение для платы EB-160

**Порядок выполнения работы**

**1.Установка аппаратных средств**

ВНИМАНИЕ: Не подключайте плату EB-160 с включенным питанием.

Перед продолжением работы убедитесь, что питание отключено.

1.1.Поместите плату EB-160 на рабочей станции EB-2000.

1.2.Подключите последовательный кабель к входу RS-232 (A) платы с одной стороны и, к COM1 или COM2 (COM порт, который не используется мышкой) персонального компьютера с другой стороны, как показано на следующей схеме:

Если Вы используете модуль PU-2000, то Вы должны иметь два свободных последовательных порта на вашем компьютере: один вход для запуска отладчика DB960, а другой – для активизации неисправностей с помощью персонального компьютера для упражнений по нахождению и устранению неисправностей .

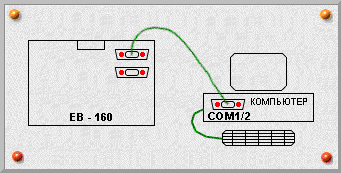
******

Рис. 1.1: Соедините EB-160 к персональному компьютеру

**2.Запуск отладчика**

Всякий раз, когда Вы запускаете отладчик, Вы можете свободно продолжать пользоваться руководством по плате EB-160 и использовать свободное место на экране для размещения различных окон отладчика.

В конце каждого сеанса отладки закройте приложение, вернитесь в программу обучения. Не забывайте переходить к обзорным вопросам, чтобы завершить урок.

1. 2.Есть два способа запуска отладчика на персональном компьютере:
   1. Из меню Start/Program панели задач выберите иконку DB960.
   2. Нажмите на иконку DB960 в правом нижнем углу экрана программы обучения.

2.2.Убедитесь, что главное меню отладчика появилось на экране.

Если появится диалоговое окно с ошибкой, нажмите на кнопку RESET на плате, дождитесь загорания светодиода и нажмите на кнопку RETRY в диалоговом окне ошибки.

Если это не поможет, то необходимо выйти из отладчика, сбросить плату и перезапустить отладчик.

Вы должны нажимать кнопку RESET на плате EB-160 каждый раз, когда Вы выходите из программы отладчика DB960

.

**3.Отображение регистров и ячеек памяти**

3.1.В окне команд введите команду:

"byte 0x10020000 length 30".

Проверьте, что отображены 48 байтов (30 hex).

3.2.В режиме просмотра (View) выберите Memory Window внизу окна, которое отобразит диалоговое окно памяти. Введите "0x10020000" для начального адреса и длину "30"hex (шестнадцатеричный) и выберите формат отображения "ORD1". Для подтверждения команды нажмите OK.

3.3.Проверьте, что отображены те же самые значения памяти.

ЗАМЕЧАНИЕ:Окно на экране может быть достаточно маленьким. Однако Вы можете растянуть его легким движением мышки. Это позволит отобразить больше информации на экране.

3.4.В управляющем окне наберите команду "regs" и нажмите "enter". Проверьте, что появилось полное отображение регистров i960.

**4.Модификация ячеек памяти**

4.1.Наберите команду "byte 0x10020000 length 30 = 0" и нажмите "enter". Убедитесь, что 48 (30 hex) байтов отображены с нулевыми значениями.

4.2.Наберите команду " byte 0x10020000 length 30 = 'A' " и нажмите "enter". Теперь наберите "byte 0x10020000 length 20".

Убедитесь, что 32 байта отображены со значениями 'A'.

●Оформите отчет и сделайте выводы по работе

Ответьте на контрольные вопросы

**??? Контрольные вопросы**

1. Как расшифровывается термин RISC?

* Класс микропроцессоров, менее надежные, чем другие.
* Микропроцессор, который имеет упрощенную систему команд.
* Микропроцессор с низкой скоростью вычисления.
* Ни один из вышеперечисленных.

1. В вышеприведенных упражнениях Вы отображали и модифицировали память. Сколько битов составляет адресная шина микропроцессора i960?

* 1 бит.
* 8 битов.
* 16 битов.
* 32 бита.

**Лабораторная работа № 18.**

**Проведение работ по загрузке и отладке отладчика** (ЕВ 160 урок 1 и 15)

**Цель работы:**

– получить представление об ошибках в программах, которые Вам понадобится находить и исправлять.

**Приборы и элементы:**

- Плата EB-160 с последовательным кабелем

- Стенд EB-2000

- Установленное программное обеспечение для платы EB-160

- Цифровой мультиметр

- Осциллограф (рекомендуется)

**Порядок выполнения работы**

**1.Загрузка простой программы**

Цель этой процедуры – загрузить и выполнить простейшую программу.

1.1.В окне команд наберите команду "load exer1". Наблюдая динамическое обновление на строке состояния внизу экрана, проверьте, что программа загружается.

1.2.Удостоверьтесь, что окно I/O видимо. Если это не так, то сделайте его видимым, выбрав окно I/O из меню View. Если оно уже выбрано, то выберите окно I/O из меню Window.

1.3.После того, как программа загрузится, нажмите на клавишу GO или нажмите на F5.

1.4.Проверьте, что окно I/O отображает сообщение: "Hello to the World of Degem".

1.5.Перед продолжением, выйдите из отладчика.

**2.Запуск отладчика**

2.1.Перейдите в окна отладчика или запустите его, щелкнув по иконке db960.

2.2.Откройте и откорректируйте размер окон для просмотра, с помощью пунктов меню view/watch.

2.3.Подключите цифровой мультиметр к выводу OUT1 и к заземлению.

**Запуск**

1. В системном приглашении отладчика введите команду "include exer14.inc". Она загрузит программу и установит контрольные точки.
2. Проверьте, что программа остановилась в первой контрольной точке L01. Обратите внимание на два вызова DACWrite( ) и DACWrite2( ).
3. Вы сейчас находитесь внутри цикла, который проверяет состояние переключателей, обновляет значение на светодиодах. Это же 4-х битное значение используется в качестве старших битов 12-битного значения подготовленного для загрузки в ЦАП. Это означает, что в этом упражнении шаг изменения напряжения составляет 256 mV.
4. Занесите переменную dacValue в окно Inspect для наблюдения за ней. Установите переключатели на плате EB-160 в "0".
5. Из значения dacValue в окне просмотра, вычислите ожидаемое выходное напряжение.
6. Нажмите на F10, чтобы выполнить процедуру DACWrite( ).
7. Проверьте, что значение в OUT1, измеренное цифровым мультиметром соответствует вашим вычислениям (3.84 V). Также проверьте, что напряжение в OUT2 - близко к напряжению OUT1.
8. Нажмите на F10, чтобы выполнить процедуру DACWrite2( ).
9. Заметьте, что значение на выходах OUT1 и OUT2, замеренные с помощью цифрового мультиметра, изменились и стали не соответствовать ожидаемым значениям.
10. Нажмите на F5, и повторите действия, начиная с пункта 6, но на сей раз, используйте пошаговое выполнение программы с помощью F8 для того, что бы определить и исправить ошибку.

*ПОДСКАЗКА:* Используйте цифровой мультиметр или осциллограф, чтобы подсчитать количество переходов сигнала CLOCK(ASCLK) от "0" к "1", и заметьте, что оно не равно числу повторов в цикле "For". Для того чтобы определить адрес DATA(ADI) и CLOCK(ASCLK), нажмите на F3 для перехода в смешанный режим просмотра C/Ассемблер.

Чтобы понять причину ошибки, проверьте, как объявляются (задаются) адреса ASCLK и AD1 в файле EB-160A.H. Сравните его со стандартным объявлением в EB-160.H.

●Оформите отчет и сделайте выводы по работе

Ответьте на контрольные вопросы

**??? Контрольные вопросы**

1. Можно ли отладить рассмотренную программу без физической проверки?
   * Да.
   * Нет
2. Почему необходима физическая проверка?
   * Для обнаружения ошибки необходимо измерить вывод ЦАП.
   * Синхросигнал должен быть проверен визуально, чтобы определить, что он функционирует не должным образом.
   * Все вышеперечисленное.
   * Ни один из вышеперечисленных.

# 

# 5.Учебно-методическое и информационное обеспечение дисциплины.

* 1. **Основная литература.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Наименование | Авторы | Место издания | Год издания | Наличие | |
| в библиотеке, экз | в ЭБС, адрес в сети Интернет |
| 1 | Микропроцессорные системы бытовой техники | Баев Б.П | М.: Горячая линия –Телеком  2010 | 2010 | 30 | <http://lib.ks54.ru/resources/library-fund> |
| 2 | Современные микроконтроллеры. Архитектура, программирование, разработка устройств | Магда Ю.С | М.:  ДМК Пресс | 2010г. |  |  |
| 3 | Микропроцессорные устройства и системы | Русанов В.В., Шевелев М.Ю. | М.: ДМК Пресс | 2012г. |  |  |

**5.2 Дополнительная литература.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Наименование | Авторы | Место издания | Год издания | Наличие | |
| в научно-технической библиотеке, экз | в ЭБС, адрес в сети Интернет |
| 1 | Цифровые устройства и микропроцессорные системы | Калабеков Б.А. | М.: Горячая линия | 2007 | 30 | <http://lib.ks54.ru/resources/library-fund> |
| **2** | Цифровая схемотехника | Мышляева И.М. | М.: ИЦ Академия | 2005 | 30 | <http://lib.ks54.ru/resources/library-fund> |
| **3** | Микропроцессорные системы | Хартов В.Я. | М.: ИЦ Академия | 2010г. |  |  |

**5.3. Базы данных, информационно-справочные и поисковые системы.**

1.<http://studopedia.net/10_90892_sovremennie-mikroprotsessori.html>. Лекции по микропроцессорам

2. <http://bigor.bmstu.ru/?cnt/?doc=MPSU/MPSU_T.the> .Микропроцессорные системы управления

3. <http://www.myshared.ru/slide/976259/> Основы МП техники

4. [Презентация на тему "Лекция 1. Предмет и основные понятия микропроцессорной техники. Классификация ЭВМ." в формате .ppt (PowerPoint)](javascript:void(0);)

5. <http://www.myshared.ru/slide/287724/Презентация> Универсальные МП ИМПК

6. <http://www.myshared.ru/slide/996481/>

7. <http://xreferat.com/33/4925-1-organizaciya-i-primenenie-mikroprocessornyh-sistem-obrabotki-dannyh-i-upravleniya.html/> Реферат по микроконтроллерам. Организация и применения микропроцессорных систем обработки данных

8.<https://docviewer.yandex.ru/?url=http%3A%2F%2Fwww.tverhtk.ru%2Flibrary%2Fpredmets%2Fpc_systems%2FMikroprocessornye_sistemy_2009.pdf&name=Mikroprocessornye_sistemy_2009.pdf&lang=ru&c=56697ba0a5e4&page=147>.Учебник: микропроцессорные системы