ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

**Федеральное государственное
образовательное бюджетное учреждение
высшего профессионального образования
«САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ
им. проф. М. А. БОНЧ-БРУЕВИЧА»**

А. А. Прасолов, С. А. Шпак

МИКРОКОНТРОЛЛЕРЫ В РАДИОСИСТЕМАХ

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ
К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ**

СПбГУТ)))

**САНКТ-ПЕТЕРБУРГ
2013**

УДК 004.383.3(077)
ББК 32.811я7
П70

Рецензент
кандидат технических наук, профессор кафедры ЦОС СПбГУТ
А. И. Солонина

Рекомендованы к печати
редакционно-издательским советом СПбГУТ

Прасолов, А. А.

П70 Микропроцессоры в радиосистемах : методические указания к выполнению лабораторных работ / А. А. Прасолов, С. А. Шпак ; СПбГУТ. – СПб., 2013 – 52 с.

Посвящены применению микроконтроллеров (МК) в аппаратуре радиосвязи, где они используются как встраиваемые элементы. Набор функций, решаемых МК весьма широк – от реализации терминала данных до цифрового формирования радиосигналов.

В первой части лабораторного практикума изучаются особенности МК и периферийных узлов на примере управления индикаторами, а затем исследуются вопросы передачи сообщений с заданной скоростью и синтез сигналов цифровыми методами с использованием цифроаналогового преобразователя.

Все работы выполняются на лабораторных макетах EASY8081B и программируются с использованием интегрированной среды разработки MikroC.

Предназначены для специальностей 210405.65 «Радиосвязь, радиовещание и телевидение» и 210700.62 «Инфокоммуникационные технологии и системы связи»

УДК 004.383.3(077)
ББК 32.811я7

СОДЕРЖАНИЕ

Лабораторная работа 1

ИЗУЧЕНИЕ МИКРОКОНТРОЛЛЕРА И СИСТЕМЫ РАЗВИТИЯ 8051

Цель работы

Изучение архитектуры микроконтроллера (МК) 8051 Intel (МК51) и выполнение простых программ управления. Изучение структурной схемы и конструкции микрокомпьютерной системы на основе МК51 и периферийных устройств.

Основные вопросы, изучаемые перед выполнением работы:

1. Структура микроконтроллера 8051.
2. Элементы языка программирования ассемблер.

Содержание работы

1. Изучение программной модели микроконтроллера 8051.
2. Изучение системы развития на микроконтроллере 8051.

Порядок выполнения

1. Произведите запуск программы отладчика из командной строки, например: **AVSIM51.EXE -C1.** Ключ -C1 (color) определяет цвет экрана, если его не указывать, экран моделирующего отладчика (Simulator/Debugger) будет работать в черно-белом режиме.

После запуска отладчика на экране появляется перечень типов МК. Для моделирования работы выбранного МК необходимо нажать клавишу с соответствующей буквой, например, B для 8052.

После этого на экране появляется рабочее окно (рис. 1). В левой части экрана находится поле мнемокодов программы. Каждому адресу ячейки памяти программ ставится в соответствие мнемокод команды, записанной по этому адресу.

На текущую команду указывает подсветка строки, например, здесь DEC A по адресу 0002H, который соответствует содержимому программного счетчика (PC-program counter).

В правой части экрана размещены основные узлы МК (ОЭВМ). Нижняя строка представляет собой меню основных режимов отладки. Выбрать один из пунктов меню можно с помощью клавиш управления курсором. Выход в основное меню из подменю осуществляется нажатием комбинации клавиш: Ctrl+C. Для изменения содержимого любого из регистров или флагов необходимо перейти из области меню в область экрана клавишей ESC. Затем подвести курсор клавишами управления курсором в необходимое место и ввести новое значение. Изменять значение регистров и флагов можно также в режиме screln.

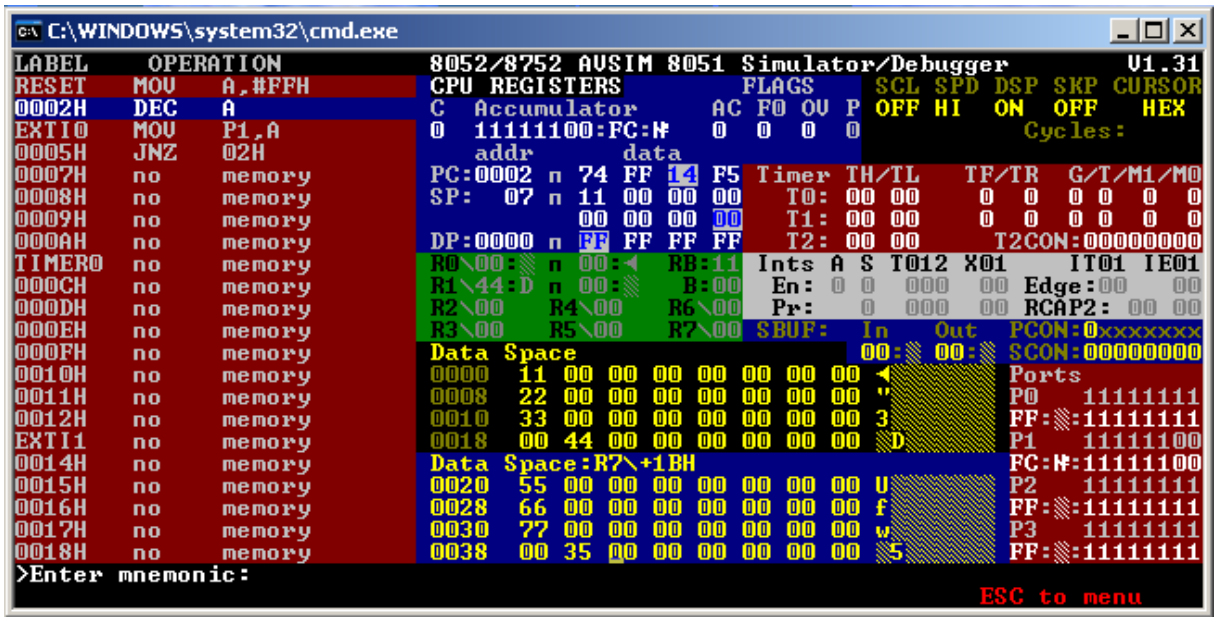


Рис. 1. Экран моделирующего отладчика

2. Выполните упражнения по следующим заданиям.

Задание 1

1) Установите PC в 0.

2) Нажмите Esc и введите по Patch команды, показанные на рис. 1.

3) Нажмите Esc и установите PC в 0.

4) Нажимайте F10 и пошагово проследите за содержимым аккумулятора и порта P1.

5) Напишите комментарий к каждой строке программы.

6) Выберите rb0, запишите 11 (например), см. Data Space (ОЗУ).

7) Выберите rb1, запишите 22 (например), см. Data Space (ОЗУ).

Задание 2

Пользуясь описанием МК48 [1], найдите регистры МК и дайте им названия по назначению.

Опишите все, что вы нашли.

Задание 3

1) Измените программу, загрузив вначале 00h и INC вместо DEC.

2) Введите DA A.

3) Исследуйте формирование кодов и признаков до значения 99D.

Сохраните полученные данные в отчете.

3. Изучение устройства EASY8051 в следующем порядке: назначение, структура устройства, лабораторная установка.

Плата EASY8051B предназначена для исследования работы основных узлов цифровой системы передачи информации и управления. Структурная схема устройства приведена на рис. 2.

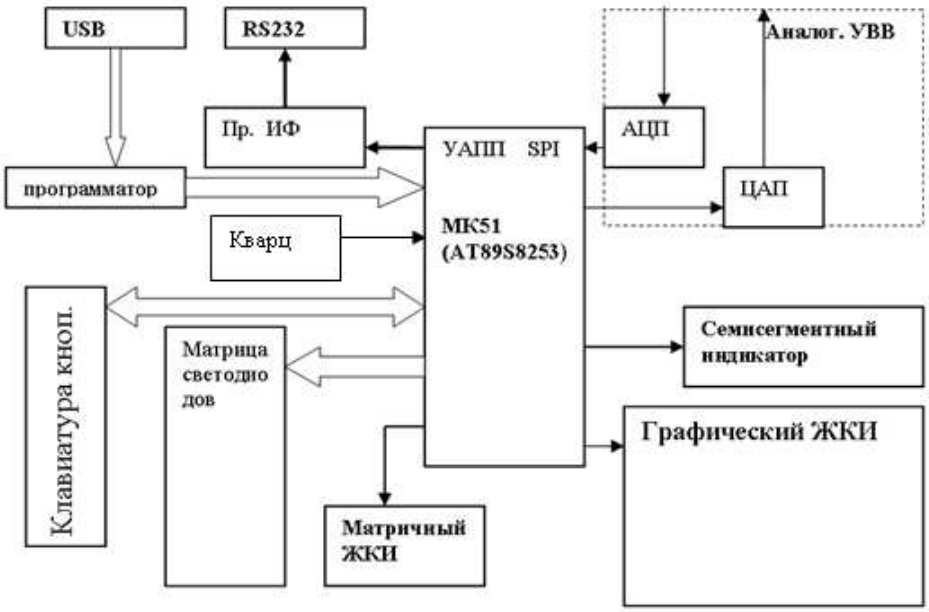


Рис. 2. Структурная схема основных узлов платы развития

3.1. Рассмотрите основные узлы модуля развития, показанные на рис. 2.

Центральный узел – микроконтроллер МК51, известный в вариантах К1816ВЕ51, ATMEL8051, AT89S8253 и других. Как ясно из предыдущей части работы, МК51 содержит АЛУ гарвардского типа, 4 параллельных 8-ми разрядных порта ввода/вывода, регистры специальных функций.

Пр.ИФ – преобразователь интерфейса RS232-UART. UART-УАПП (универсальный асинхронный приемопередатчик) – основа телекоммуникации, модемной связи.

Аналог. УВВ – аналоговое устройство ввода-вывода с АЦП и ЦАП

3.2. Изобразите структурную схему системы.

3.3. Пользуясь прил. 1, кратко опишите функции узлов USB, RS232 – связь с ПК.

3.4. Запишите значение частоты кварца в генераторе тактовых импульсов – это опорная частота микропроцессорной системы.

3.5. Найдите остальные узлы структурной схемы, используя прил. 1, где дана топология конструкции печатной платы.

4. Определение характеристик микроконтроллера и периферийных устройств платы развития.

4.1. Определите необходимый объем ОЗУ для хранения массива из 10 элементов типа int, char и double (см. прил. 2).

4.2. Объем ОЗУ микроконтроллера 4 кбайт. Определите максимальное число элементов типа int, char и double, которое может быть записано в ОЗУ.

4.3. Разрядность ЦАП установленного на плате развития 12 бит. Определите объем памяти, который потребуется для хранения 36 отсчетов сигнала.

Содержание отчета

1. Структурная схема микроконтроллера 8051.
2. Результаты выполнения заданий 1–3, 3.1–4.2.

Лабораторная работа 2

ИЗУЧЕНИЕ РАБОТЫ ПОРТОВ ВВОДА/ВЫВОДА

Цель работы

Изучение интегрированной среды разработки mikroC, создание программ управления портами ввода/вывода микроконтроллеров типа 8051.

Основные вопросы, изучаемые перед выполнением работы

1. Архитектура микроконтроллера AT89S8253 [2–3, 6].
2. Структура лабораторного макета EASY8051 [4].

Содержание работы

1. Изучение интегрированной среды разработки mikroC.
2. Изучение структуры программного проекта.
3. Освоение принципов написания и отладки программ с помощью mikroC.
4. Изучение работы портов ввода вывода и программирование их работы.

Порядок выполнения

1. Получить индивидуальное задание у преподавателя.
2. Изучить принципиальную схему соединений клавиатуры (рис. 1) и светодиодной матрицы (рис. 2).

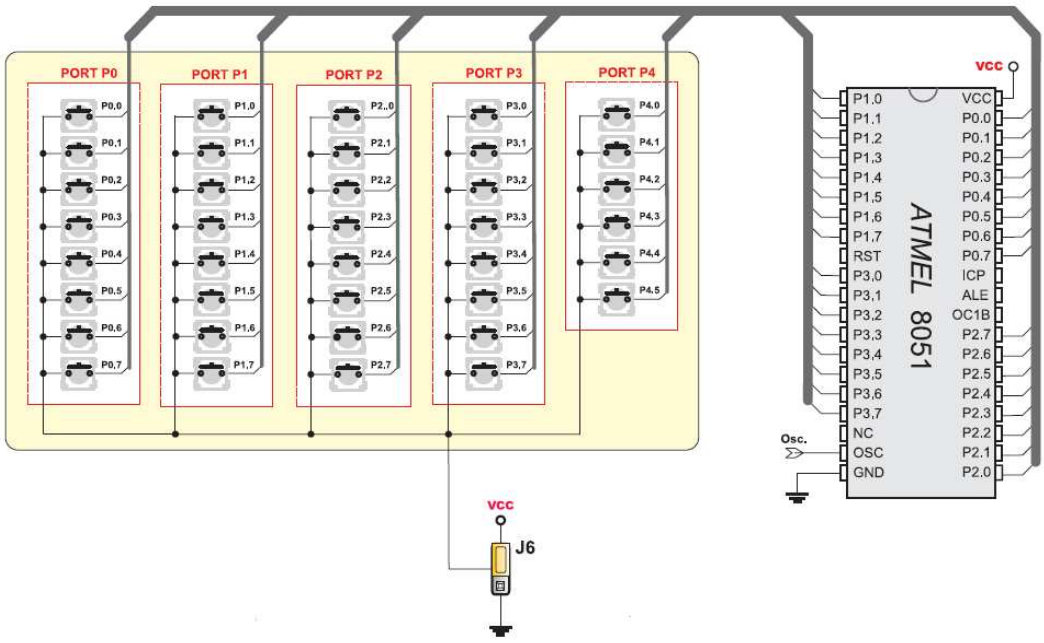


Рис. 1. Принципиальная схема подключения клавиатуры
к портам ввода/вывода микроконтроллера

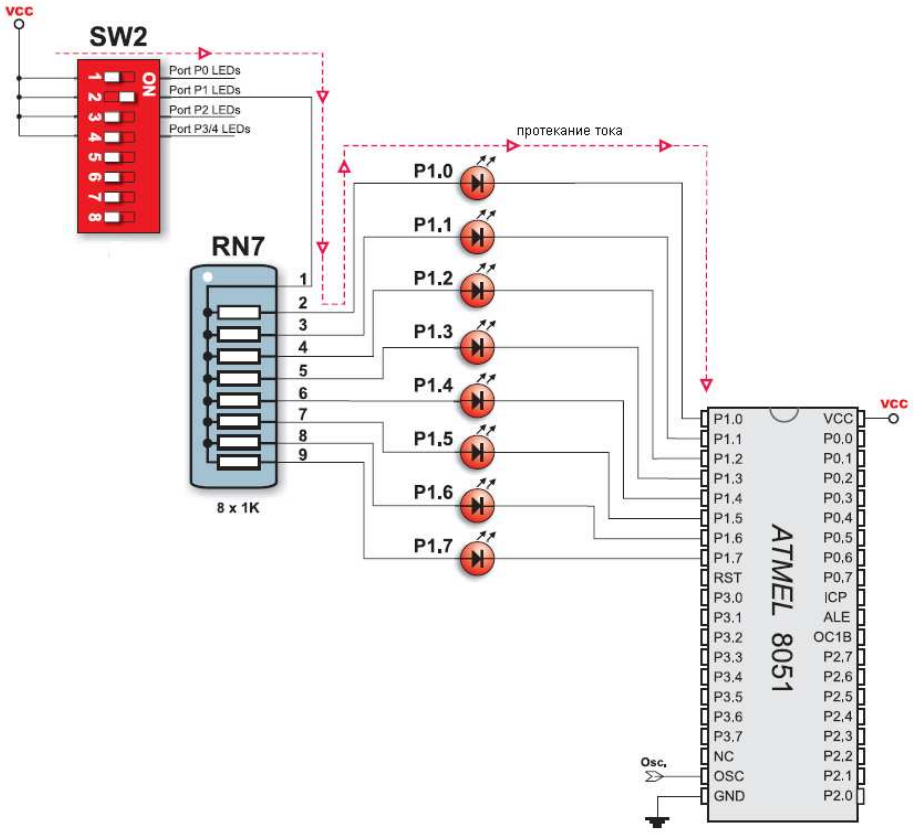


Рис. 2. Принципиальная схема подключения светодиодов
к порту 1 микроконтроллера

3. Запустите программу mikroC. Компилятор mikroC (фирмы MIKROELERTRONIKA) представляет собой удобную интегрированную среду разработки (Integrated Development Enviroment) (рис. 3).

Рассмотрим создание простого проекта, содержащего один файл исходного текста, который компилируется и собирается с использованием mikroC. Изучите назначение элементов управления проектом среды разработки mikroC (рис. 3).

Проводник по коду (**Code Explorer**) – позволяет осуществлять быстрый поиск переменных и функций в проекте.

Параметры проекта (**Project Settings**) – содержат установки выбора микроконтроллера и его параметров (частота кварцевого генератора и модель памяти).

Редактор кода (**Code Editor**) – расширенный редактор кода с подсветкой синтаксиса.

Менеджер проекта (**Project Manager**) – позволяет выполнить быстрое перемещение по файлам проекта.

Сообщения (**Messages**) – окно вывода всех сообщений, генерируемых при компиляции и сборке проекта.

Менеджер библиотек функций (**Library Manager**) – позволяет подключать к проекту встроенные функции различного назначения.

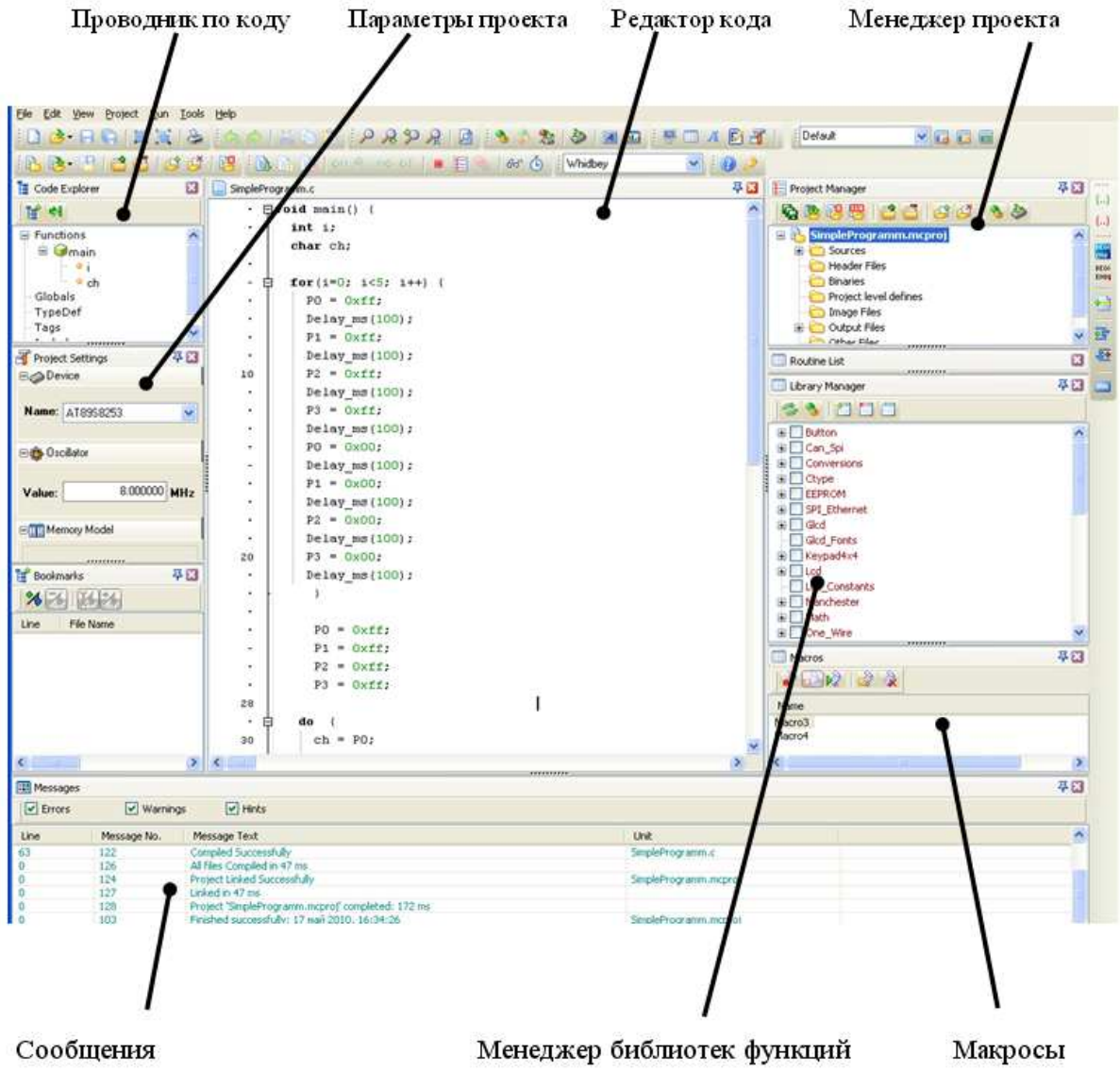


Рис. 3. Окно интегрированной среды разработки mikroC

Разработка программ на mikroC начинается с создания так называемого ПРОЕКТА. Он представляет собой набор файлов, необходимых для создания исполняемого файла для микроконтроллера, таких как файл проекта (.mcproj), один или несколько исходных файлов на языке С (.c), бинарные файлы (.mcl), заголовочные файлы (.h).

4. Создайте новый проект, для этого, выберите пункт меню **Project → New Project** или нажимаем иконку  , расположенную на панели инструментов. На экране появится окно Мастера создания нового проекта (**New Project Wizard**), показанное на рис. 4, и устанавливаем следующие параметры:

Device Name: AT89S8253 (тип контроллера).

Device Clock: Частоту установленного кварцевого резонатора посмотрите на плате (см. прил. 1, расположение 20).

Memory Model: Small

New Project Location: <Каталог группы>/Lab1

Мастер создания проекта автоматически добавит в проект файл Lab1.c и откроет его в редакторе кода.

Рис. 4. Окно мастера создания проекта

5. Изучите программу приведенную ниже, которая выполняет следующие функции: при нажатии кнопки клавиатуры P0.7 на светодиодной матрице отображается «1», а при нажатии кнопки P0.6 – «2», при этом джампер (перемычка) J6 подключает клавиатуру к земле (GND), а J1-J5 подключают порты микроконтроллера к питанию (Pull Up):

```
void main() {   // Главная функция
 char ch;       // Объявление переменной ch типа char

  do  {          // Начало бесконечного цикла
   ch = P0;      // Запись содержимого порта 0 в переменную ch

    switch(ch) {
     case 127:    // Если переменная ch равна 01111111b, т.е. была
                  // нажата кнопка P0.7, тогда
      P0 = 0xfb;  // Вывод в порт 0 числа 11111011b
      P1 = 0xfd;  // Вывод в порт 1 числа 11111101b
      P2 = 0x00;  // Вывод в порт 2 числа 00000000b
      P3 = 0xff;  // Вывод в порт 3 числа 11111111b
      Delay_ms(1000); // Вызов функции задержки на 1000 мс
      break;
     case 191:    // Если переменная ch равна 10111111b, т.е. была
                  // нажата кнопка P0.6, тогда
      P0 = 0x7c;  // Вывод в порт 0 числа 01111100b
      P1 = 0x3e;  // Вывод в порт 1 числа 00111110b
```

```
    P2 = 0x5e;  // Вывод в порт 2 числа 01011110b
    P3 = 0x60;  // Вывод в порт 3 числа 01100000b
    Delay_ms(1000);
  break;
  }
    P0 = 0xff;   // Вывод в порт 0 числа 11111111b
    P1 = 0xff;   // Вывод в порт 1 числа 11111111b
    P2 = 0xff;   // Вывод в порт 2 числа 11111111b
    P3 = 0xff;   // Вывод в порт 3 числа 11111111b

 } while (1);   // Конец тела цикла

 }  // Конец тела функции главной программы
```

6. На основе программы, приведенной в п. 5, напишите программу согласно индивидуальному заданию.

7. После того, как программа написана, необходимо осуществить компиляцию проекта с помощью команды меню **Project→Build**. Если в программе отсутствуют ошибки, то в окне сообщений будет выведена информация об успешном выполнении компиляции (рис. 5).

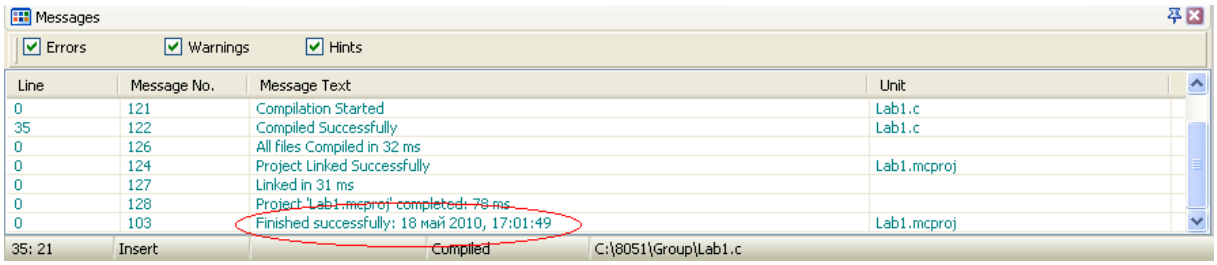


Рис. 5. Сообщение об успешной компиляции проекта

В случае наличия ошибок в программе, информационные сообщения будут подсвечены красным цветом и указано место в программе, где присутствует ошибка и характер ошибки (рис. 6). В этом случае необходимо устранить ошибку и повторить компиляцию проекта.

Как видно из рис. 6, ошибка заключается в том, что отсутствует оператор «;» в 16-й строке программы.

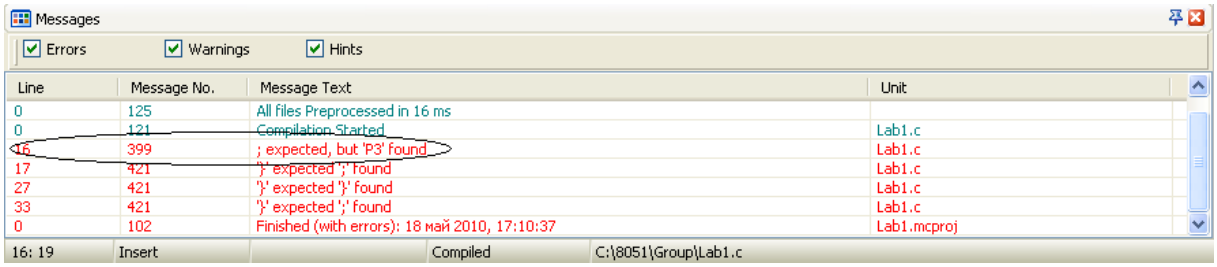


Рис. 6. Сообщение об ошибках в программе

8. После того, как программа скомпилирована, ее необходимо записать в ПЗУ микроконтроллера и запустить на выполнение для проверки правильности работы. Для этого выберите пункт меню **Project→Build + Program**. Среда разработки выполнит повторную компиляцию проекта, и запустит программатор (**8051 Flash Programmer**), который осуществит запись программы в микроконтроллер. Внешний вид окна программатора с необходимыми параметрами показан на рис. 7.

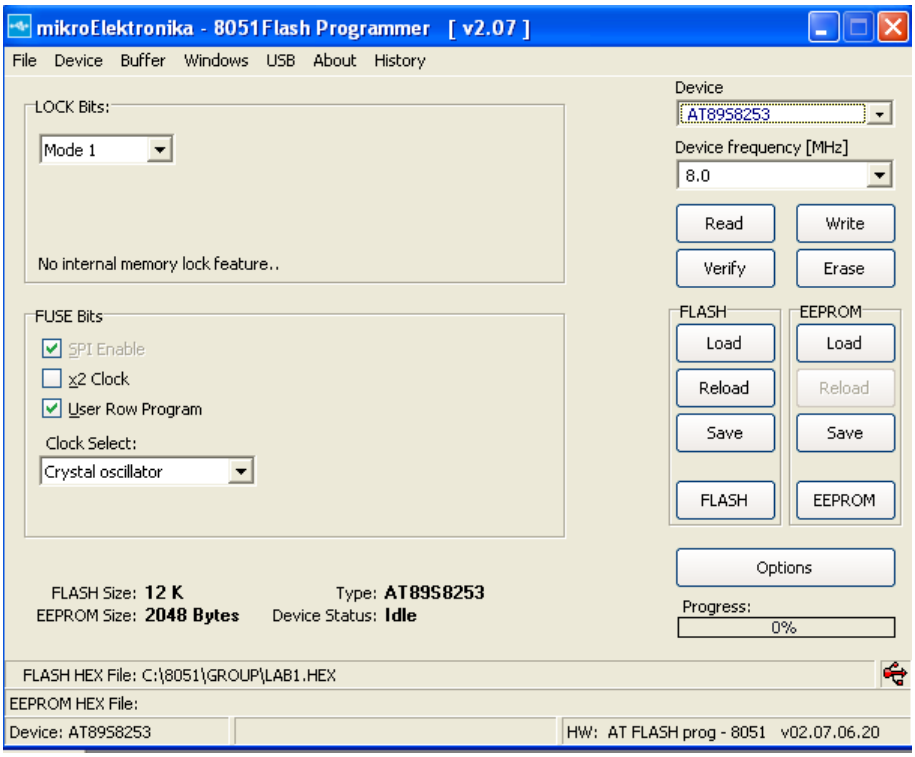


Рис. 7. Окно программатора

9. Убедитесь, что джамперы (перемычки) **J1**, **J2**, **J3**, **J4**, **J5** в положении ***Pull Up***, а **J6** в положении ***GND***, и все переключатели (**1–8**) в группе **SW1** находятся в положении **OFF**, переключатели **1–4** группы **SW2** в положении **ON**, а переключатели **5–8** в **OFF**.

10. Нажмите кнопку **RESET**, это приведет к перезапуску микроконтроллера, и убедитесь в правильности его работы. В случае неправильной работы возвратитесь к п. 5 и проанализируйте алгоритм программы, и добейтесь правильной работы.

11. В процессе компиляции перед сборкой проекта происходит преобразование (трансляция) текса программы с языка Си на язык ассемблера. Результат трансляции можно посмотреть, выбрав команду меню **Project→View Assembly**. Сравните текст программы на этих двух языках и обратите внимание на реализацию подпрограммы (функции) задержки на языке ассемблера.

Содержание отчета

1. Принципиальные схемы подключения светодиодной матрицы и клавиатуры к портам контроллера.
2. Тексты программы на языках ассемблера и Си с комментариями.

Задания к лабораторной работе

Установки перемычек для всех заданий: джампер J6 подключает клавиатуру к земле (GND), а J1–J5 подключают порты микроконтроллера к питанию (Pull Up).

1. При нажатии кнопки клавиатуры P3.5 на светодиодной матрице отображается «3», при нажатии кнопки P3.4 – «4», а кнопки P2.5 – «A».
2. При нажатии кнопки клавиатуры P2.2 на светодиодной матрице отображается «6», при нажатии кнопки P0.1 – «7», а кнопки P0.2 – «B».
3. При нажатии кнопки клавиатуры P3.7 на светодиодной матрице отображается «9», при нажатии кнопки P3.6 – «C», а кнопки P0.3 – «5».
4. При нажатии кнопки клавиатуры P2.4 на светодиодной матрице отображается «1», при нажатии кнопки P2.3 – «H», а кнопки P0.0 – «8».
5. При нажатии кнопки клавиатуры P2.1 на светодиодной матрице отображается «2», при нажатии кнопки P2.0 – «G», а кнопки P3.7 – «A».
6. При нажатии кнопки клавиатуры P2.7 на светодиодной матрице отображается «U», при нажатии кнопки P3.6 – «B», а кнопки P3.7 – «0».
7. При нажатии кнопки клавиатуры P3.5 на светодиодной матрице отображается «S», при нажатии кнопки P3.4 – «J», а кнопки P0.3 – «L».
8. При нажатии кнопки клавиатуры P3.2 на светодиодной матрице отображается «-», при нажатии кнопки P3.1 – «O», а кнопки P0.0 – «Z».

Лабораторная работа 3

УПРАВЛЕНИЕ СЕМИСЕГМЕНТНЫМИ ИНДИКАТОРАМИ

Цель работы

Изучить принцип вывода информации на семисегментные индикаторы в статическом и динамическом режимах.

Основные вопросы, изучаемые перед выполнением работы

1. Архитектура микроконтроллера AT89S8253.
2. Структура лабораторного макета EASY8051.

Содержание работы

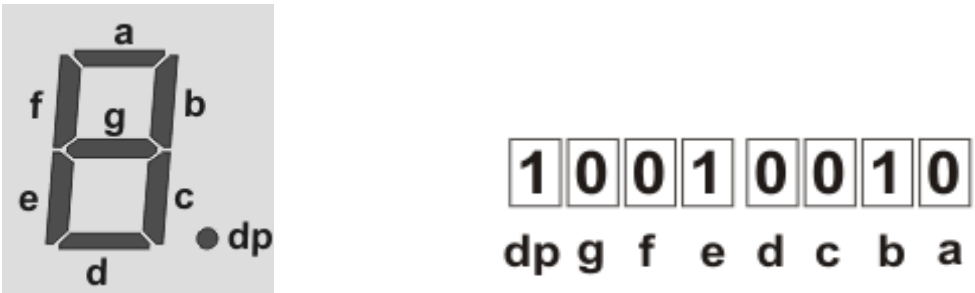
1. Изучение принципа формирования сигналов управления семисегментными индикаторами.
2. Изучение работы семисегментных индикаторов в статическом режиме.
3. Изучение работы семисегментных индикаторов в динамическом режиме.

Порядок выполнения

1. Получить индивидуальное задание у преподавателя.
2. Изучить принцип работы семисегментных индикаторов.

Семисегментный индикатор, содержит 7 прямоугольных СИД, коммутируемых по отдельности, их комбинации могут составлять упрощённые изображения цифр и некоторых букв A, C, E, J, F, U, H, L, b, c, d, o, r, t , которые иногда используются для обозначения регистров МП.

Сегменты индикатора обозначаются буквами от A до G; восьмой элемент – десятичная точка DP (decimal point) для отображения дробных чисел (рис. 1). Индикаторы различаются по типу соединения светодиодов – общий анод, общий катод, по количеству отображаемых разрядов и по цвету красный, зеленый, желтый и другие.



а) б)

Рис. 1. Обозначения сегментов индикатора – а
и их положение в разрядах кода – б

2. Напишите управляющее слово, которое соответствует работе индикатора, как показано на рис. 2.

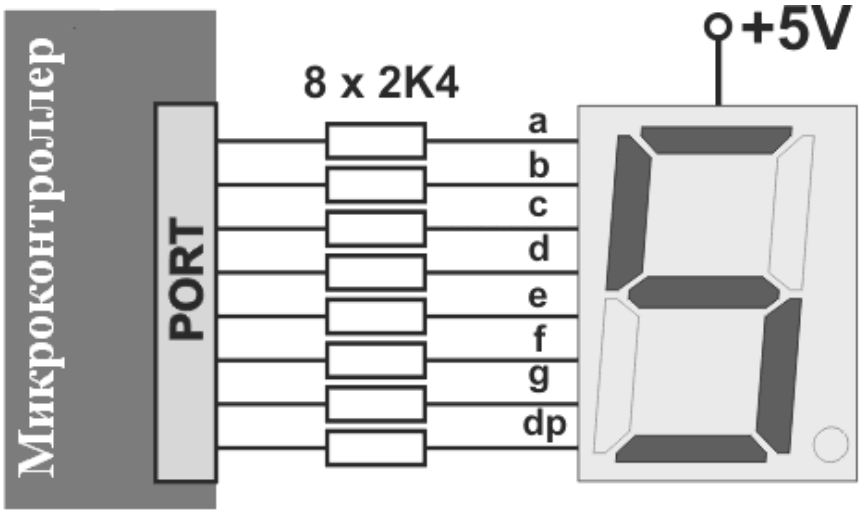


Рис. 2. Схема подключения семисегментного индикатора
к порту микроконтроллера

3. Изучите принципиальную схему подключения семисигментных индикаторов в динамическом режиме.

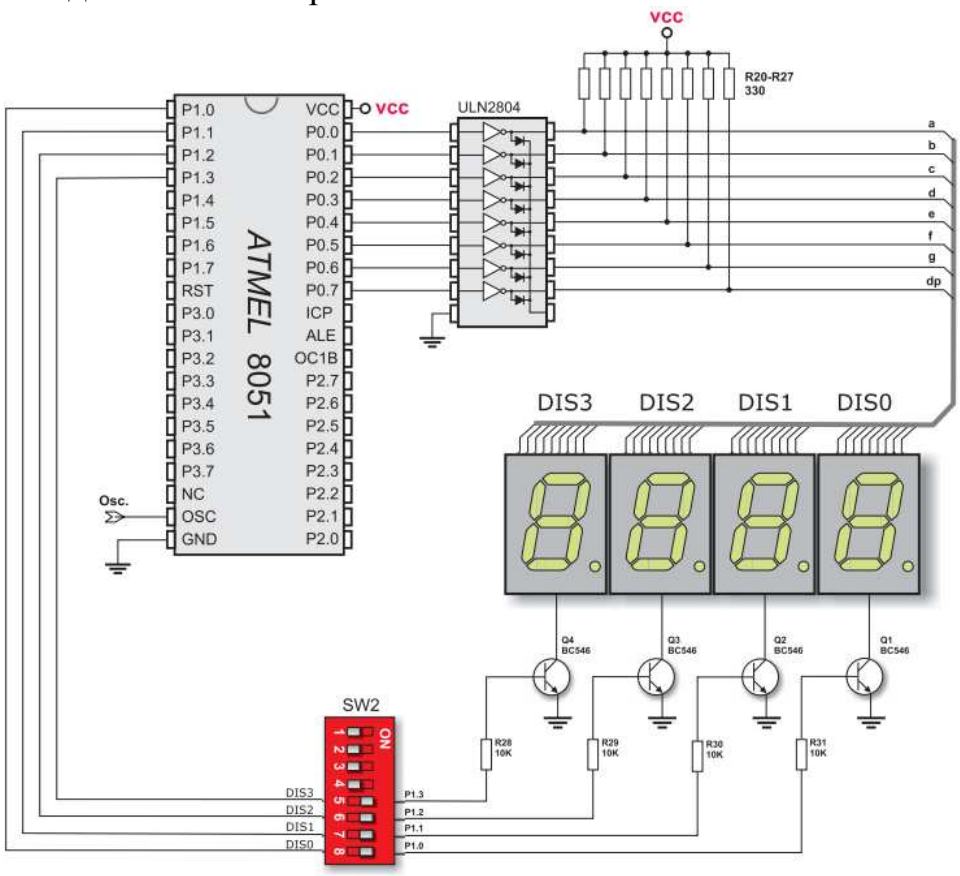


Рис. 3. Схема подключение семисегментных индикаторов
на плате развития EASY 8051B

Обратите внимание, что для работы семисегментных индикаторов согласно схеме подключения, изображенной на рис. 3, необходимо подавать управляющее слово в порт 0 в инверсном коде.

4. Изучите программу вывода числа «6» на семисегментный индикатор ***DIS3***. Слово управления для вывода числа «6» в прямом коде будет соответствовать значению ***0x7D*** (0111 1101), а в обратном – ***0x82*** (1000 0010);

```
void main() {   // Главная функция

  P1 = 0;  // Очистка порта 1

  P0 = 0x82;   // Вывод в порт 0 числа 1000 0010b
  P1.B3 = 1;   // Вывод в 3-й бит порта 1 -  «1»

}
```

5. Наберите и скомпилируйте программу из п. 4.

6. Убедитесь, что все переключатели (**1–8**) в группе **SW1** находятся в положении **OFF**, переключатели **1–4** группы **SW2** в положении **OFF**, а переключатели **5–8** в **ON**.

7. Запишите программу в ПЗУ микроконтроллера и проверьте правильность работы.

8. Напишите программу согласно индивидуальному заданию 1 и проверьте ее работу на плате развития.

9. Напишите программу работы семисегментных индикаторов в динамическом режиме, согласно индивидуальному заданию 2 и проверьте ее работу на плате развития.

Содержание отчета

1. Принципиальные схемы подключения семисегментных индикаторов к портам контроллера.

2. Тексты программ на языке Си с комментариями.

Задания к лабораторной работе

Задание 1

1. На семисегментном индикаторе **DIS3** при нажатии кнопки клавиатуры P3.5 отображается число «3», при нажатии кнопки P3.4 – «2».

2. На семисегментном индикаторе **DIS2** при нажатии кнопки клавиатуры P2.2 отображается число «1», при нажатии кнопки P2.1 – «4».

3. На семисегментном индикаторе **DIS1** при нажатии кнопки клавиатуры P3.7 отображается число «5», при нажатии кнопки P3.6 – «6».

4. На семисегментном индикаторе **DIS0** при нажатии кнопки клавиатуры P2.4 на светодиодной матрице отображается «7», при нажатии кнопки P2.3 – «8».

5. На семисегментном индикаторе **DIS1** при нажатии кнопки клавиатуры P2.1 отображается «9», при нажатии кнопки P2.0 – «0».

6. На семисегментном индикаторе **DIS2** при нажатии кнопки клавиатуры P2.7 на светодиодной матрице отображается «A», при нажатии кнопки P3.6 – «b».

7. На семисегментном индикаторе **DIS3** при нажатии кнопки клавиатуры P3.5 на светодиодной матрице отображается «C», при нажатии кнопки P3.4 – «F».

8. На семисегментном индикаторе **DIS0** при нажатии кнопки клавиатуры P3.2 на светодиодной матрице отображается «E», при нажатии кнопки P3.1 – «H».

Задание 2

Выведете на семисегментные индикаторы (**DIS0 – DIS3**) четырехзначное число:

1. 1300
2. 2400
3. 4800
4. 9600
5. 1030
6. 7200
7. 5040
8. 0920

Лабораторная работа 4

ИССЛЕДОВАНИЕ УНИВЕРСАЛЬНОГО ПРИЕМО-ПЕРЕДАТЧИКА МИКРОКОНТРОЛЛЕРА И ИНТЕРФЕЙСА RS232

Цель работы

Изучение режимов передачи информации в программируемом универсальном асинхронном приемопередатчике (УАПП – UART). Получение навыков его программирования.

Основные вопросы, изучаемые перед выполнением работы

1. Структура и принцип работы УАПП.
2. Основные режимы работы УАПП.
3. Инициализация УАПП для заданного режима работы (скорости передачи, формата информационного символа, проверки четности, количества стоп-бит).

Содержание работы

1. Изучение структуры и принципа работы УАПП.
2. Изучение режимов работы УАПП в асинхронном режиме.
3. Выполнение инициализации УАПП и программирование его на передачу байта информации с заданной скоростью.
4. Расчет скорости передачи.

Порядок выполнения

1. Получить индивидуальное задание у преподавателя.
2. Изучите принцип и режимы работы УАПП по прил. 3.
3. Ознакомьтесь со структурной схемой лабораторной установки (рис. 1). Компьютер подключаем к МК AT89S8253 через программатор USB2.0. МК подключаем к порту RS-232 с помощью группы выключателей SW1, в которой выключатели требуется установить в положение, показанное на рис. 2.
4. Рассчитать значения регистра счетчика TH для группы скоростей передачи, указанных в индивидуальном задании по формуле:

$$TH1 = \frac{98304 - \dfrac{2^{SMOD} \cdot f_{рез}}{f_{1,3}}}{384},$$

где *SMOD* – значение 7-го бита регистра PCON, $f_{1,3}$ – требуемая скорость передачи, $f_{рез}$ – значение частоты кварцевого резонатора установленного на плате.

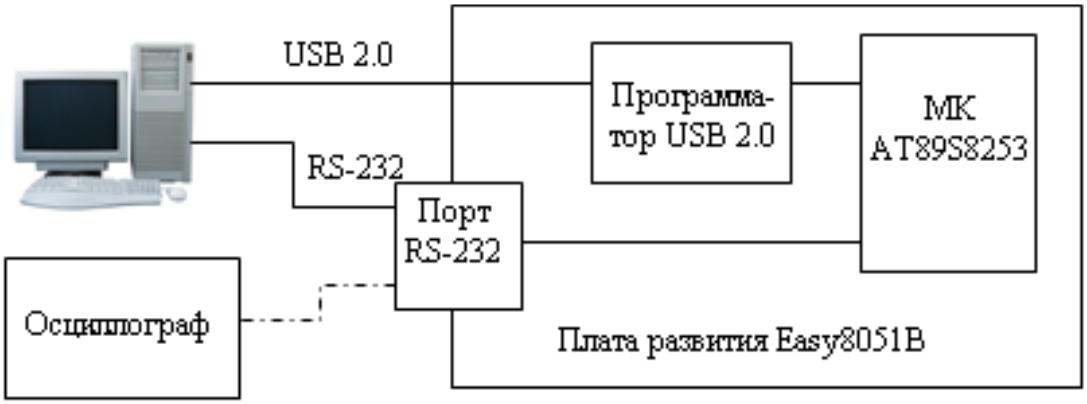


Рис. 1. Структурная схема лабораторной установки.

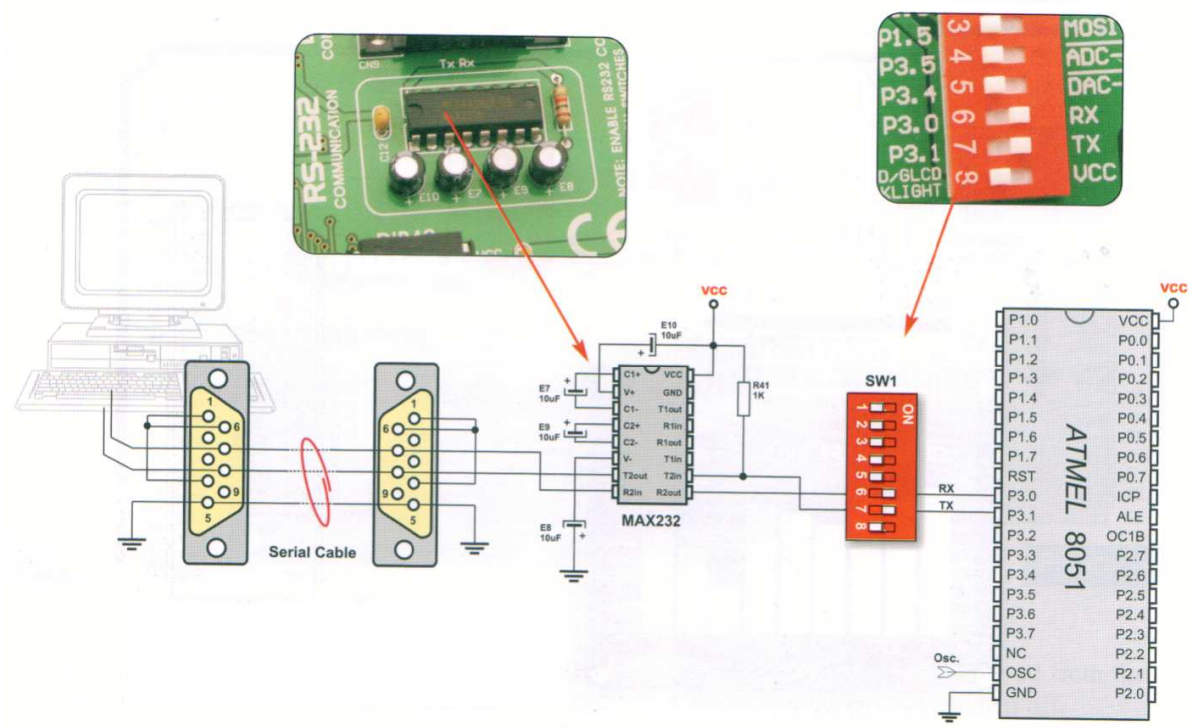


Рис. 2. Схема подключения порта RS-232 к МК AT89S8253

5. Изучите программу инициализации УАПП, таймера и передачу символа 0x1B через последовательный интерфейс в режиме 1 на скорости 2400 бод, число бит – 8, без проверки на четность:

```
void main() {

  SCON = 0x50;  // SCON: режим 1, число бит 8
  TMOD = 0x20;  // TMOD: таймер 1 - режим

  PCON.B7 = 0;
  TH1 = 0xf7;
```

```
 TR1 = 1;              // TR1: таймер 1 запуск
 TI = 0;

do{
  SBUF = 0x1B; //запись в Serial BUFfer байта данных
  Delay_ms(100);
 } while(1);
}
```

6. Изучите алгоритм программы передачи символа через УАПП и вывода скорости передачи на 7-сегментные индикаторы (рис. 3).

Рис. 3. Алгоритм программы передачи байта

7. После того, как программа написана, согласно индивидуальному заданию, необходимо осуществить компиляцию проекта и записать программу в ПЗУ микроконтроллера.

Обратите внимание, для того чтобы скорости передачи соответствовали константам, записанным в регистр таймера, необходимо в окне программатора (рис. 4) выключить значение ***x2 Clock***.

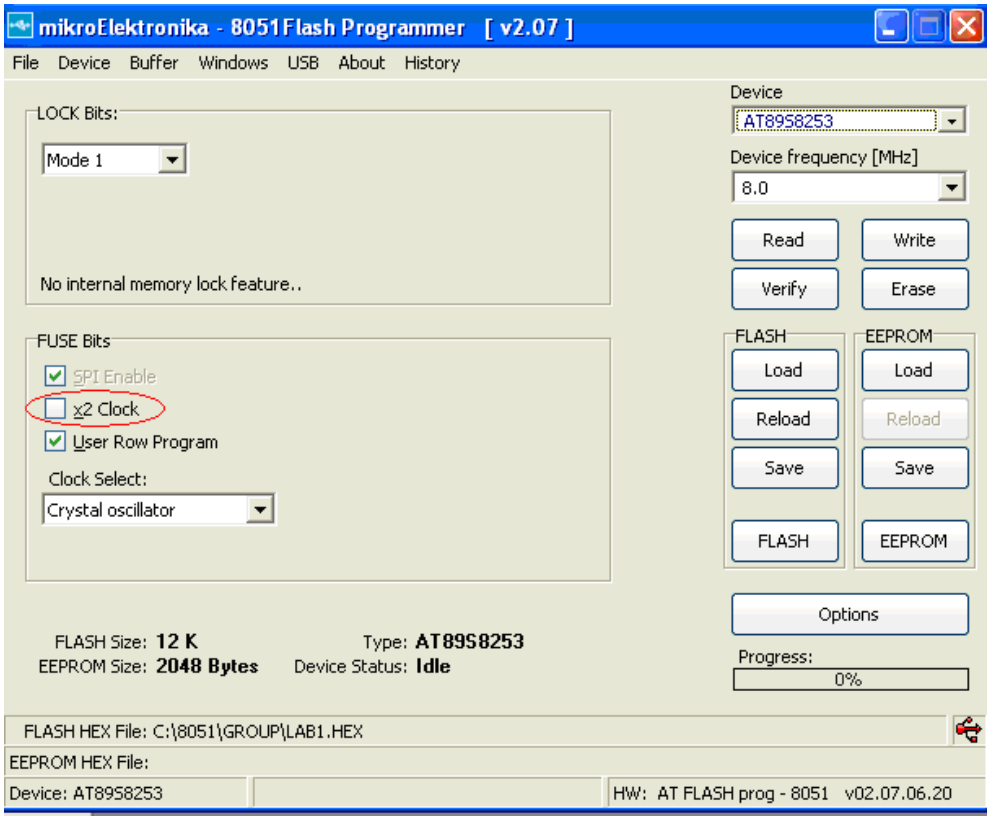


Рис. 4. Установки программатора

8. Проконтролировать передачи байта по интерфейсу RS-232, воспользовавшись утилитой **USART Terminal**, находящейся в меню **Tools** (рис. 5). Для этого в окне **USART Terminal** необходимо установить параметры передачи в блоке **Settings**:

Com Port: COM1, COM2, COM3…

Baud: 110, 300, 600, 1200, 2400 и т.д. (скорость, на которой принимаем сигнал)

Stop bits: One Stop Bit, One and a Half Stop Bits, Two Stop Bits

Parity: None (бит паритета не используется)

Data bits: Five, Six, Seven, Eight (количество бит данных)

Выберите шестнадцатеричный формат отображаемых принятых данных в поле **Receive data as** (ASCII, HEX, DEC).

После установки всех необходимых параметров нажмите кнопку **Connect.**

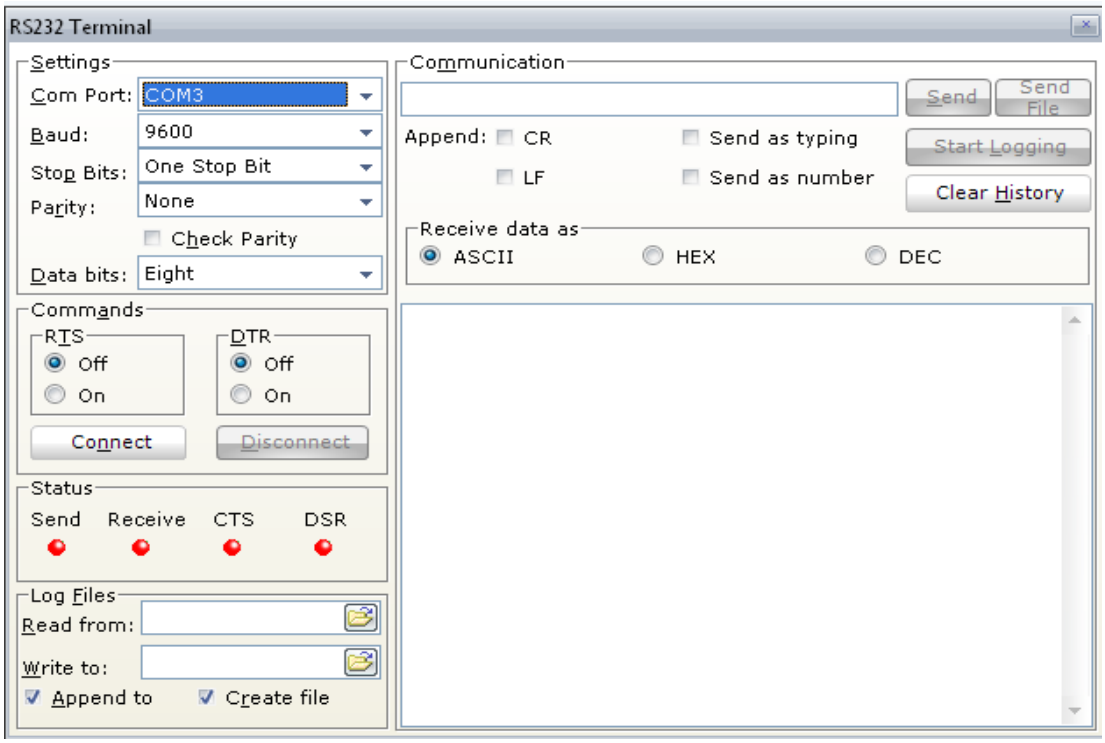


Рис. 5. Окно RS-232 Terminal

Содержание отчета

1. Структурная схема лабораторной установки.
2. Структурная схема алгоритма программы.
3. Исходные тексты программы на языке Си с комментариями.

Задания к лабораторной работе

Скорость передачи отображается на 7-сегментных индикаторах.

1. Режим работы UART – 1, число бит 8. При нажатии кнопки клавиатуры P2.0 скорость передачи устанавливается 600, при нажатии кнопки P2.1 – 1200.

2. Режим работы UART – 3, число бит 8. При нажатии кнопки клавиатуры P2.2 скорость передачи устанавливается 1200, при нажатии кнопки P2.3 – 2400.

3. Режим работы UART – 1, число бит 8. При нажатии кнопки клавиатуры P2.7 скорость передачи устанавливается 9600, при нажатии кнопки P2.6 – 4800.

4. Режим работы UART – 1, число бит 8. При нажатии кнопки клавиатуры P2.5 скорость передачи устанавливается 4800, при нажатии кнопки P2.4 – 2400.

5. Режим работы UART – 3, число бит 8. При нажатии кнопки клавиатуры P2.4 скорость передачи устанавливается 600, при нажатии кнопки P2.7 – 2400.

6. Режим работы UART – 1, число бит 8. При нажатии кнопки клавиатуры P2.2 скорость передачи устанавливается 1200, при нажатии кнопки P2.3 – 4800.

Лабораторная работа 5

ПЕРЕДАЧА СООБЩЕНИЙ С ПОМОЩЬЮ УАПП И ИНТЕРФЕЙСА RS-232

Цель работы

Изучение принципов передачи информации в программируемом универсальном асинхронном приемопередатчике с помощью прерываний микроконтроллера.

Основные вопросы, изучаемые перед выполнением работы

1. Структура и принцип работы УАПП.
2. Инициализация УАПП для заданного режима работы (скорости передачи, формата информационного символа, проверки четности, количества стоп-бит).
3. Принцип работы системы прерываний микроконтроллера.

Содержание работы

Программирование УАПП на передачу информационного слова (несколько байт) с заданной скоростью и использованием прерываний.

Порядок выполнения

1. Получить индивидуальное задание у преподавателя.
2. Изучить систему прерываний микроконтроллера и принцип работы с ними в прил. 4.
3. Изучить структурную схему алгоритма подпрограммы передачи массива символов, показанную на рис. 1.

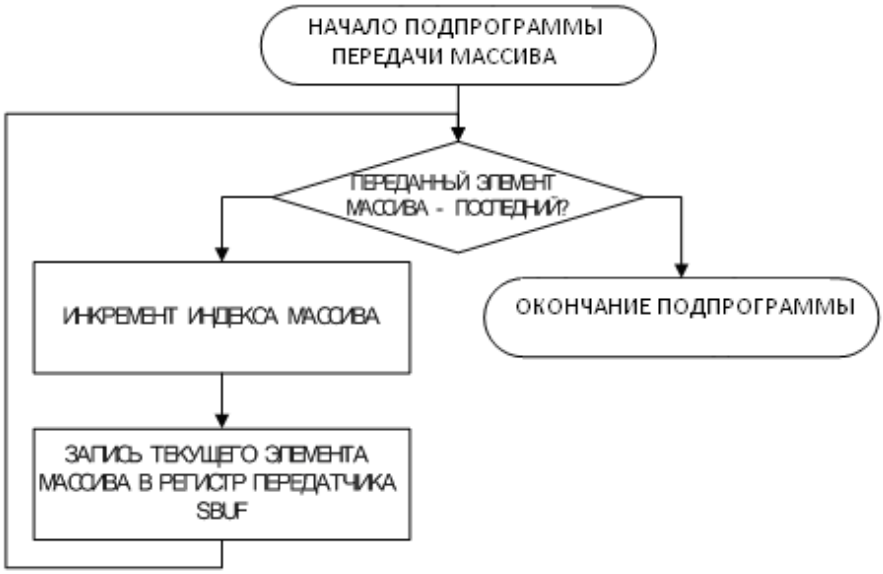


Рис. 1. Алгоритм подпрограммы передачи массива символов

В начале подпрограммы передачи идет проверка, является ли текущий элемент массива последним, и пока не будет передан последний элемент, индекс будет инкрементироваться, а текущий элемент массива – записываться в SBUF, т. е. выход из подпрограммы передачи будет возможен только после окончания передачи всех символов массива.

4. Изучите программу передачи массива символов с использованием прерываний:

```
char uart_data[]="This programm for GUT";
char i;
void main() {

SCON = 0x50;
TMOD = 0x20;

PCON.B7 = 1;
TH1 = 0xf7; // TH1: передача со скоростью 4800 бод
TR1 = 1;
ES = 1;   /* Разрешение прерываний  */
EA = 1;

i=1;
SBUF = uart_data[0];

void serial_IT(void) org 0x23 {
  if(i<21) {
    SBUF = uart_data[i];
    i++;
  }
}
```

5. Используя программу передачи массива в п. 4, нарисовать структурную схему алгоритма и написать программу согласно индивидуальному заданию.

6. Осуществите компиляцию проекта и запишите программу в ПЗУ микроконтроллера.

7. Проконтролировать передачу по интерфейсу RS-232, воспользовавшись утилитой **USART Terminal**.

Содержание отчета

1. Структурная схема лабораторной установки.
2. Структурная схема алгоритма программы.
3. Исходные тексты программы на языке Си с комментариями.

Задания к лабораторной работе

Режим работы и одна из скоростей передачи берутся из задания к лабораторной работе 4.

1. Массив для передачи «123456789» при нажатии кнопки клавиатуры P2.0 и «AT+CPIN?» при нажатии кнопки P2.1.

2. Массив для передачи «AT+CGMI» при нажатии кнопки клавиатуры P2.2 и «AT+CSCS=?» при нажатии кнопки P2.2.

3. Массив для передачи «ATDT 3451798» при нажатии кнопки клавиатуры P2.7 и «AT+WPCS=?» при нажатии кнопки P2.6.

4. Массив для передачи «AT+CGMM» при нажатии кнопки клавиатуры P2.5 и «AT+WPCS=CUSTOM» при нажатии кнопки P2.4.

5. Массив для передачи «AT+CSCS=GSM» при нажатии кнопки клавиатуры P2.4 и «AT+CIMI» при нажатии кнопки P2.7.

6. Массив для передачи «AT+WPCS=TRANSPARENT» при нажатии кнопки клавиатуры P2.2 и «AT+CCID=?» при нажатии кнопки P2.3.

Лабораторная работа 6

ФОРМИРОВАНИЕ СИГНАЛОВ УПРАВЛЕНИЯ

Цель работы

Изучение схемы подключения цифроаналогового преобразователя (ЦАП) к однокристальному микроконтроллеру на EASY8051, особенностей их взаимодействия и освоение алгоритма программного генерирования гармонических сигналов и сигналов произвольной формы.

Основные вопросы, изучаемые перед выполнением работы

1. Принцип формирования сигналов управления.
2. Архитектура и принцип работы ЦАП MSP4921.
3. Особенности передачи информации по интерфейсу SPI (Serial Peripheral Interface).

Содержание работы

1. Изучение принципов программно-аппаратного генерирования сигналов заданной формы.
2. Исследование алгоритмов и программ прямого цифрового синтеза частот.

Порядок выполнения

1. Получить индивидуальное задание у преподавателя.
2. Методы прямого цифрового синтеза (ПЦС).

Задача ПЦС – получить на выходе сигнал синусоидальной формы заданной частоты. Поскольку в ПЦС формирование выходного сигнала происходит в цифровой форме, совершенно очевидна необходимость цифроаналогового преобразования. Это означает, что в структуре ПЦС должен быть ЦАП. Для получения синусоидального сигнала на вход ЦАП необходимо подать последовательность отсчетов функции $\sin(x)$.

Синус можно рассчитать следующими методами:

– табличным, использующим записанную в ПЗУ таблицу синусов;

– вычислительным, при котором синус рассчитывается представлением его рядом Тейлора, цепными дробями или каким-либо другим способом;

– гибридным, тем или иным образом сочетающим использование таблицы синусов с вычислениями.

В данном случае будем рассматривать табличный метод.

В этом случае необходимо рассчитать значение отсчетов функции синуса по формуле:

$$Y = 2^{N-1} \cdot \sin(X) + 2^{N-1},$$

где N – разрядность ЦАП.

Для наглядного представления синусоиды рассчитайте 12 значений отсчетов и составьте таблицу значений X и Y.

SIN	X	Y
0	0	2048
…	…	…
…	…	…

3. Для преобразования цифрового сигнала в аналоговый Easy 8051B имеет на плате 12-битный ЦАП MCP4921, который подключен к микроконтроллеру с использованием SPI интерфейса (рис. 1). Изучите работу ЦАП и управление им в прил. 5.

Обратите внимание на то, что сигнал MISO не используется, так как обмен данными между микроконтроллером и микросхемой ЦАП осуществляется только в одностороннем режиме (от микроконтроллера к ЦАПу).

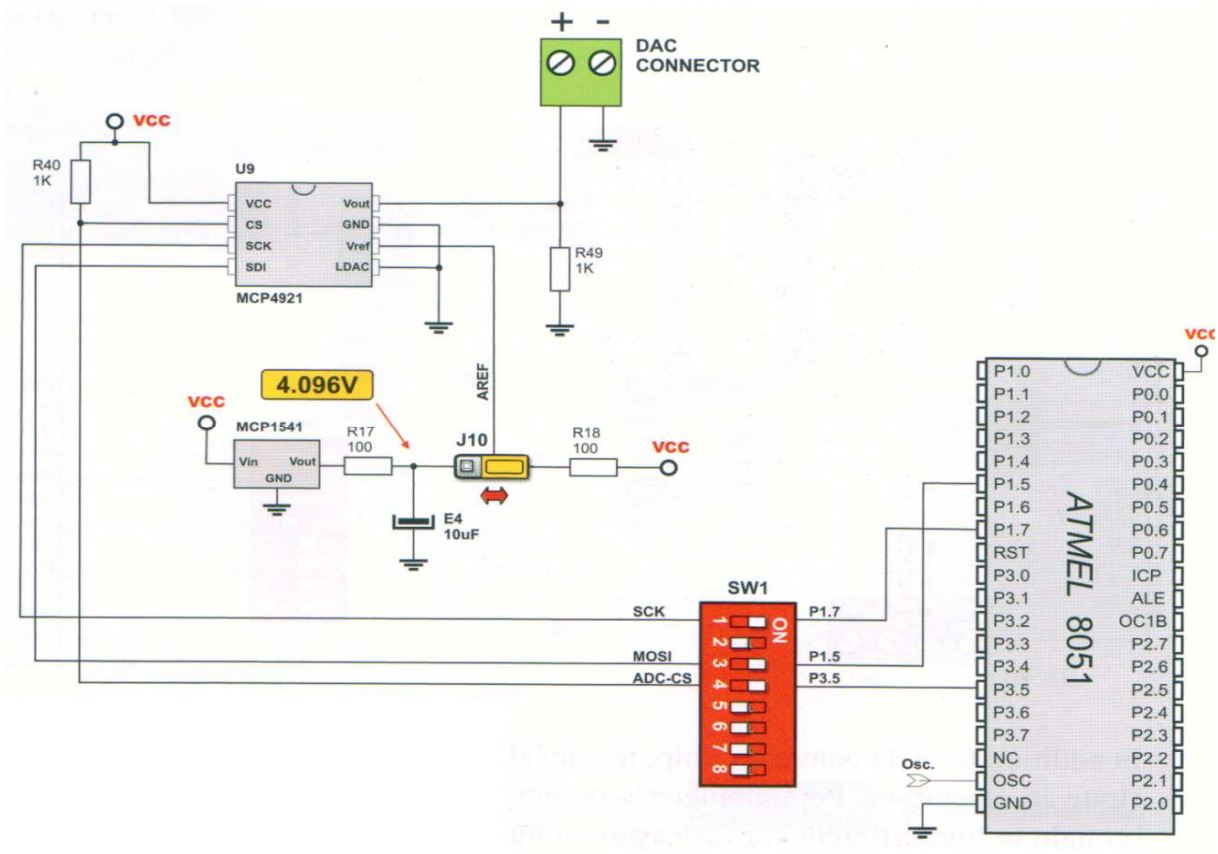


Рис. 1. Схема подключения ЦАП MCP4921 к микроконтроллеру на плате EASY8051B

4. Структурная схема лабораторной установки показана на рис. 2.

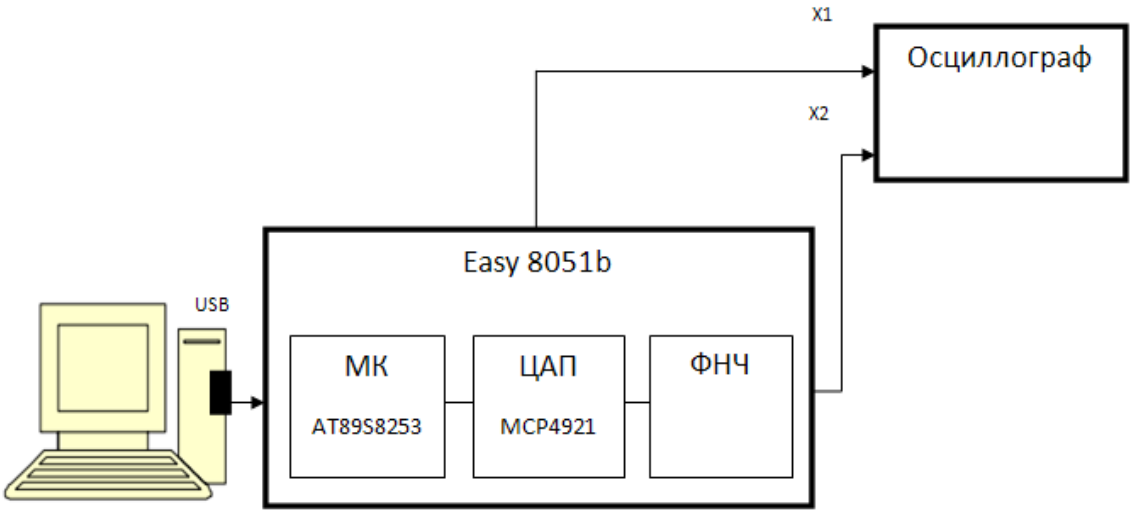


Рис. 2. Структурная схема лабораторной установки

5. Изучите алгоритм генерирования гармонического сигнала, представленный на рис. 3.

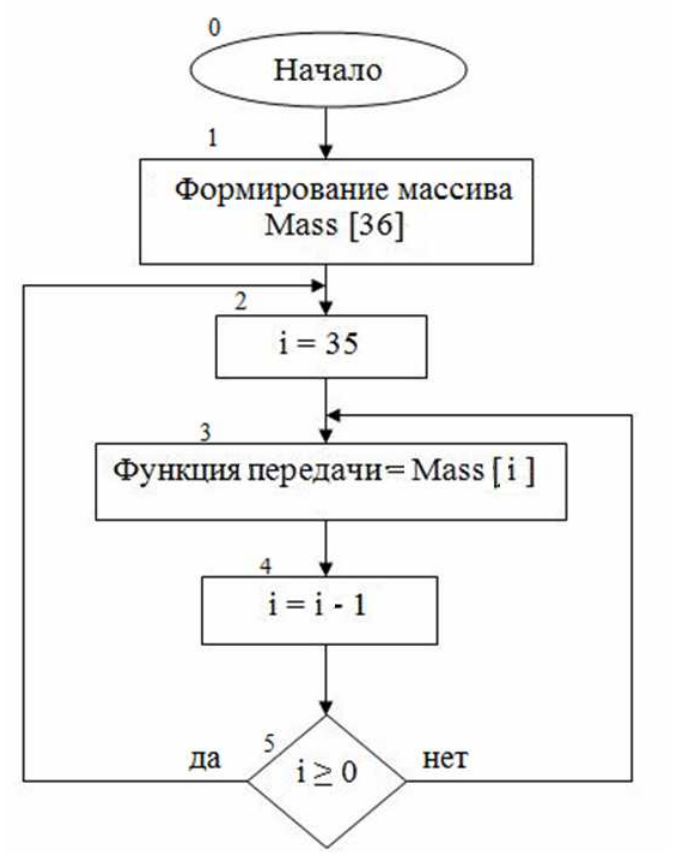


Рис. 3. Алгоритм генерирования гармонического сигнала

6. Изучите, приведенную ниже, подпрограмму управления микросхемой ЦАП:

```
// Определение портов микроконтроллера для
// работы с SPI интерфейсом
sbit Chip_Select_DAC at P3.B4;
sbit SCK at P1.B7;
sbit MISO at P1.B6;
sbit MOSI at P1.B5;

void InitMain() {  // Функция инициализации SPI

Spi_Init_Advanced(MASTER_OSC_DIV4 | DATA_ORDER_MSB |
CLK_IDLE_LOW | IDLE_2_ACTIVE);
}
//Функция передачи команды управления

void DAC_Output(unsigned int valueDAC) {
  char temp;

  Chip_Select_DAC = 0;                  // Выбор микросхемы ЦАП

  temp = (valueDAC >> 8) & 0x0F;      // Запись управляющего слова и
  temp |= <Управляющее слово>;     //формирование старшего байта
                                    //команды управления
 SPI_Write(temp);  //передача старшего байта в ЦАП

  temp = valueDAC;              //формирование младшего байта команды
управления

  SPI_Write(temp);            // передача младшего байта в ЦАП
  Chip_Select_DAC = 1;     // отключение ЦАП от шины SPI
}
```

7. Запустите среду разработки MikroC 8051. Создайте новый проект (см. лабораторную работу 2) и назовите его Lab6.

В окне менеджера библиотек функций (**Library Manager**) подключите библиотеку SPI (рис. 4).

На основе изложенного в прил. 4 принципа взаимодействия микросхемы ЦАП с микроконтроллером, приведенному ниже примеру, а также индивидуальному заданию, необходимо написать функцию передачи команды управления по интерфейсу SPI. Согласно индивидуальному заданию определите <Управляющее слово>, которое соответствует битам [15 … 12] в команде управления.

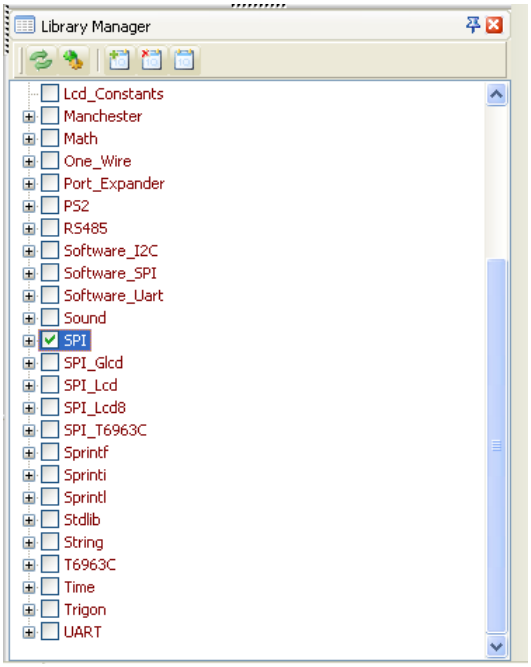


Рис. 4. Подключение внешней библиотеки

8. По индивидуальному заданию напишите программу генерирования сигнала на основе приведенного ниже шаблона:

```
void main(){
  int m[12] = {<Значение1>,<Значение2>,…}; //массив из 12 значений
  int i;

  InitMain(); // Инициализация интерфейса

  while(1) {   // Бесконечный цикл

    for(i=35; i>=0; i--) {          //Цикл повторения массива
      DAC_Output(m[i]);       //Вывод i-го значения массива в ЦАП
    }
  }
}
```

9. Убедитесь, что переключатели 1–3 и 5 в группе **SW1** находятся в положении **ON**, а переключатели 1–8 в группе **SW2** в положении **OFF**.

10. Осуществите компиляцию проекта, запись программы в ПЗУ микроконтроллера и запустите на выполнение. Убедитесь в правильности работы программы, и зарисуйте полученный сигнал.

Содержание отчета

1. Структурная схема лабораторной установки.
2. Программы инициализации и генерирования заданного сигнала.
3. Результаты исследования при помощи осциллографа.

Задания к лабораторной работе

1. Генерирование гармонического сигнала на основе 12 отсчетов за период, с амплитудой 2 В и частотой 1 кГц.
2. Генерирование гармонического сигнала на основе 12 отсчетов за период, с амплитудой 1 В и частотой 1 кГц.
3. Генерирование гармонического сигнала на основе 12 отсчетов за период, с амплитудой 1,5 В и частотой 0,5 кГц.
4. Генерирование гармонического сигнала на основе 12 отсчетов за период, с амплитудой 1 В и частотой 2 кГц.
5. Генерирование гармонического сигнала на основе 12 отсчетов за период, с амплитудой 2 В и частотой 0,1 кГц.
6. Генерирование гармонического сигнала на основе 12 отсчетов за период, с амплитудой 1,2 В и частотой 300 Гц.
7. Генерирование пилообразного сигнала с частотой 1 кГц и размахом 4 В.
8. Генерирование пилообразного сигнала с частотой 0,5 кГц и размахом 2 В.
9. Генерирование пилообразного сигнала с частотой 2 кГц и размахом 1 В.

СПИСОК ЛИТЕРАТУРЫ

1. *Шпак, С. А.* Микропроцессоры в радиосистемах: методические указания / С. А. Шпак ; СПбГУТ. – СПб., 2007. – 51 с.

2. *Микушин, А. В.* Цифровые устройства и микропроцессоры / А. В. Микушин, А. М. Сажнев, В. И. Сединин. – СПб. : БХВ-Петербург, 2010. – 832 с.

3. *Сташин, В. В.* Проектирование цифровых устройств на однокристальных микроконтроллерах / В. В. Сташин, А. В. Урусов, О. Ф. Мологонцева. – М. : Энергоатомиздат, 1990. – 224 с.

4. *Магда, Ю. С.* Микроконтроллеры серии 8051: практический подход / Ю. С. Магда. – М. : ДМК Пресс, 2008. – 228 с.

5. MikroElektronika. EASY8051B User's Manual / http://www.mikroe.com/downloads/get/133/easy8051b_manual.pdf

6. *Verle, M.* Architecture and programming of 8051 MCU'S. Microelectronika, 2009 // www.mikroe.com/products/view/267/architecture-and-programming-of-8051-mcu-s/

7. Almel Corp. Almel 8051 Microcontrollers Hardware Manual, 2007.

ПРИЛОЖЕНИЕ 1

Структура платы развития EASY8051

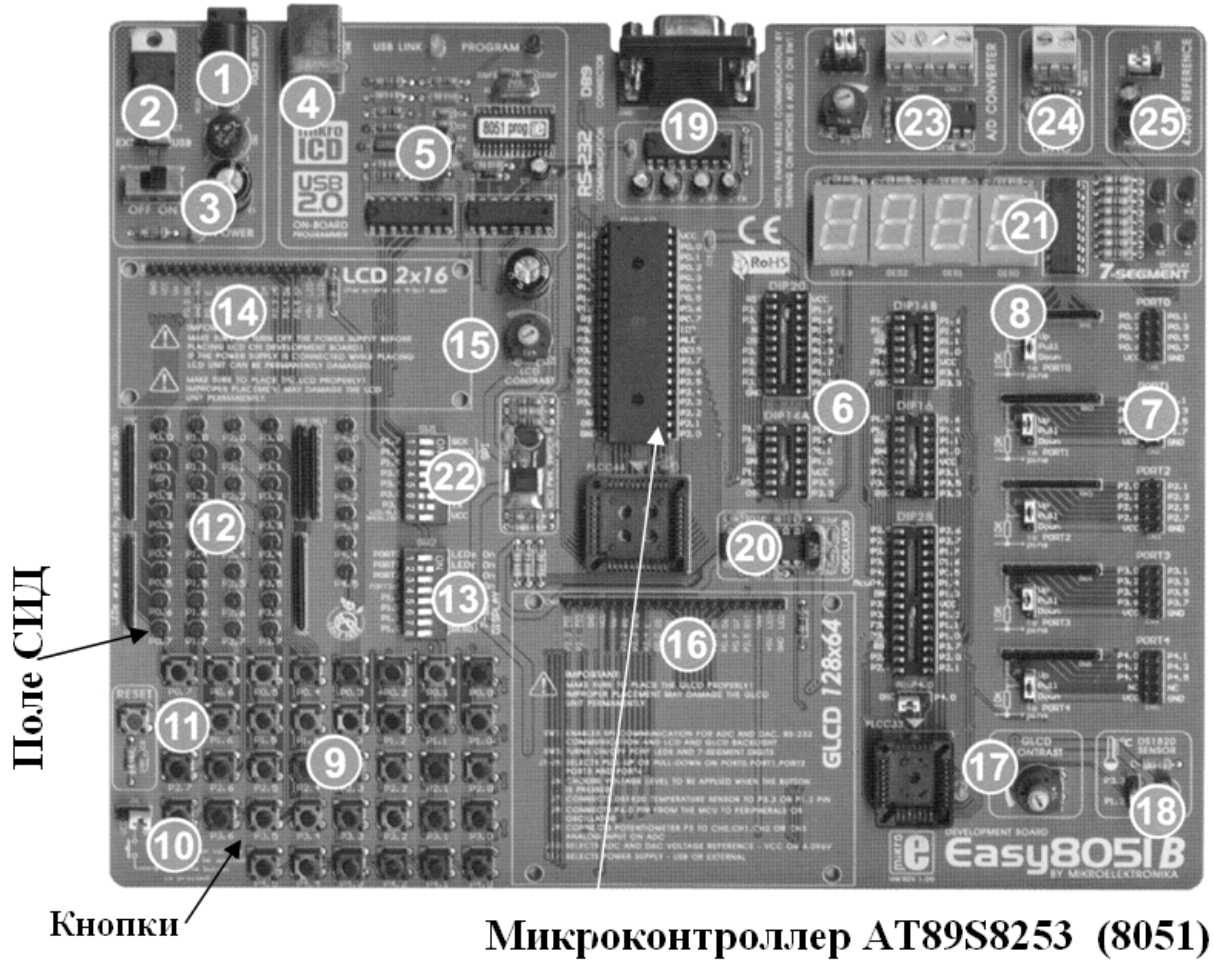


Рис. П1.1. Размещение основных узлов МК системы 8051B

1. Разъем питания 8–16 В постоянного тока.
2. Разъем питания: внешний или через USB.
3. Выключатель питания.
4. Разъем USB2.0, соединение с ПК для программирования МК.
5. Программатор
6. Гнезда для микроконтроллеров в корпусах DIP14, DIP20, DIP28, DIP40 и PLCC44, это означает, что почти весь ряд микроконтроллеров ATMEL8051 может быть использован с устройством развития EASY8051B.
7. Разъемы портов для прямого доступа.
8. Перемычки для определения действия входного вывода.
9. Клавиатура – 38 кнопок модификации состояния любого вывода МК, т. е. ввода информации.
10. Переключатель SW6 выбора уровня («1» или «0») на выводах при нажатии кнопки.

11. Кнопка RESET.
12. Светодиоды отображения состояния каждого бита (линии) портов.
13. Группа выключателей SW2.
14. Разъем 2x16 для жидкокристаллического индикатора (ЖКИ).
15. Потенциометр для настройки и регулировки ЖКИ.
16. Разъем для графического ЖКИ.
17. Потенциометр для настройки контрастности графического ЖКИ.
18. Гнездо для подключения датчика температуры DS1820.
19. Порт RS-232.
20. Цепь тактового генератора.
21. 7-сегментный светодиодный индикатор
22. Группы выключателя SW1.
23. 12-разрядный АЦП.
24. 12-разрядный ЦАП.
25. Источник опорного напряжения 4.096 В.

Easy 8051B имеет 2 группы переключателей. Группа переключателей SW1 используется для расширения интерфейса.

Группа переключателей SW2 используется для коммутации светодиодов и 7-сегментных индикаторов к портам МК.

При использовании кабеля USB устройство развития Easy 8051B присоединяется к ПК – активируется внутренний программатор платы USB 2.0, а также подается питание напряжением 5 В.

ПРИЛОЖЕНИЕ 2

Лексические единицы языка С-51

Комментарии. Комментарии куски текста, используемого для аннотирования программы. Комментарии предназначены исключительно для программиста. Они удаляются из исходного текста перед синтаксическим анализом. Есть два способа написания комментариев: Метод С и С++. Оба поддерживают MikroC.

С комментарии: любая последовательность символов, помещенная после символа /* и заканчивающаяся символом */. Например:

```
/* Введите здесь свой комментарий. Он может занимать несколько
строк. */
```

С++ комментарии: MikroC позволяет использовать однострочные комментарии с помощью символа две косые черты (//). Комментарий может начинаться в любой позиции и продолжается до следующей новой строки:

```
// Введите здесь свой комментарий.
// Он может занимать только одну строку.
```

Константы. Константы предназначены для введения чисел в состав выражений операторов языка программирования С. В отличие от идентификаторов, всегда начинающихся с буквы, константы всегда начинаются с цифры. В языке программирования С-51 разделяют четыре типа констант: целые знаковые и беззнаковые константы, константы с плавающей запятой, символьные константы и литеральные строки.

Целочисленные константы могут быть записаны как двоичные, восьмеричные, десятичные или шестнадцатеричные числа в зависимости от того, какая система счисления удобнее для представления константы. При работе с выводами микроконтроллера или передаче двоичных данных удобнее пользоваться двоичными числами или их более короткой формой записи – восьмеричными или шестнадцатеричными числами.

Десятичная константа состоит из одной или нескольких десятичных цифр, причем первая цифра не может быть нулем (иначе число будет воспринято как восьмеричное).

Восьмеричная константа состоит из обязательного нуля и одной или нескольких восьмеричных цифр (среди цифр должны отсутствовать цифры восемь и девять, так как эти цифры не входят в восьмеричную систему счисления). Если константа содержит цифру, недопустимую в восьмеричной системе счисления, то константа считается ошибочной.

Шестнадцатеричная константа начинается с обязательной последовательности символов 0x или 0X и содержит одну или несколько шестнадцатеричных цифр (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F).

Двоичная константа начинается с обязательной последовательности символов 0b или 0B, например 0b1101.

Десятичная константа	Восьмеричная константа	Шестнадцатеричная константа	Двоичная константа
16	020	0x10	0b10000
127	0117	0x2B	0b1111111
240	0360	0XF0	0b11110000

Константа с плавающей запятой – это десятичное число, представленное в виде действительного числа с десятичной запятой и порядком числа. Формат записи константы с плавающей запятой:

[цифры].[цифры] [E|e [+|-] цифры].

Число с плавающей запятой состоит из целой и дробные части и (или) порядка числа. Для определения отрицательного числа необходимо сформировать константное выражение, состоящее из знака минуса и положительной константы. Примеры записи констант с плавающей запятой:

115.75, 1.5E-2, -0.025, .075, -0.85E2

Символьная константа – представляется ASCII или ANSI символом, заключенном в апострофы. Управляющая последовательность тоже может быть использована в символьных константах. При этом она рассматривается как одиночный символ. Значением символьной константы является числовой код символа. Примеры символьных констант: ' ' – пробел, 'Q' – буква Q, '\n' – символ новой строки, '\\' – обратная дробная черта, '\v' – вертикальная табуляция .

Символьные константы имеют тип char и при преобразовании типов дополняются знаком. Символьные константы используются обычно при управлении микроконтроллерным устройством с клавиатуры.

Строковые константы. Если символьные константы используются обычно при вводе информации с клавиатуры, то при отображении информационных сообщений обычно используются целые строки символов. В строке допускается использование пробелов.

Строковая константа (литерал или литеральная строка) – последовательность символов (включая строковые и прописные буквы русского и латинского алфавитов, а также цифры), заключенные в кавычки ("). Например: "Школа N 35", "город Тамбов", "YZPT КОД".

Литеральная строка рассматривается как массив символов (char[]). Отметим важную особенность: число элементов массива равно числу символов в строке плюс 1, так как нулевой символ (символ конца строки) также является элементом массива.

Выражением называется комбинация знаков операций и операндов, результатом которой является определенное значение. Каждый операнд в выражении в свою очередь может быть выражением. Значение выражения зависит от расположения знаков операций и круглых скобок в выражении, а также от приоритета выполнения операций.

Примеры выражений:

```
F=A+B;
A*(B+C)-(D-E)/F;
```

Операндом в выражении может быть переменная, числовая константа, подпрограмма-функция или указатель. Любой операнд, который имеет константное значение, называется константным выражением. Каждый операнд имеет тип.

В языке программирования С-51 используются **арифметические операции**, результат которых зависит от типа операндов: **'+'** суммирование; **'-'** вычитание; **'*'** умножение; **'/'** деление; **'%'** вычисление остатка от целочисленного деления.

В языке программирования С-51 также определено несколько одноместных арифметических операций: '**-**' изменение знака операнда на противоположное значение; '**+**' знак плюс не влияет на значение операнда; **++** увеличение значения операнда на единицу; **--** уменьшение значения операнда на единицу.

Для одноместной операции требуется один операнд, которому она предшествует, например:

```
P3 = -5;   //Присвоить порту P3 значение числа -5
a = -b;    //Присвоить переменной a отрицательное значение переменной b
c=++a + 2; //Увеличить значение переменной a на 1 и прибавить 2
c=a++ + 2; //Прибавить к переменной a 2, присвоить это значение
```

переменной c и только после этого увеличить значение переменной a на 1.

Над операндами можно осуществлять **логические операции**: '&&' логическое «и»; '&' побитовое логическое «и»; 'll' логическое «или»; 'l' побитовое логическое «или»; '^' «исключающее или» (суммирование по модулю два).

Здесь следует объяснить различие между логическими и побитовыми логическими операциями. Дело в том, что в стандартном языке ANSI C не существует битовых переменных. Для хранения битовых значений истина '1' и ложь '0' используются стандартные целые типы переменных. В простейшем случае это байт. При этом все значения переменной, отличающиеся от 0, считаются 1. Например, пусть в переменной a хранится число 5, а в переменной b – число 6, тогда:

a|b=7 //00000101 or 00000110 = 00000111
a||b=1 //(00000101)->1 (00000110)->1; (1 or 1= 1), Результат равен 00000001
a&b=4 //00000101 and 00000110 = 00000100
a&&b=1 //(00000101)->1 (00000110)->1; (1 and 1= 1), Результат равен 00000001
a^b=3 //00000101 xor 00000110 = 00000011

Исполняемые операторы: оператор присваивания; условный оператор; структурный оператор; оператор цикла for; оператор цикла с проверкой условия до тела цикла; оператор цикла с проверкой условия после тела цикла; оператор continue; оператор выбора; оператор безусловного перехода; оператор выражение; оператор возвращения из подпрограммы; пустой оператор.

Оператор присваивания записывается в виде:
Переменная=выражение;

Выражение вычисляется, и полученное значение присваивается переменной, например:

P0=2; //Установить начальные потенциалы на выводах второго порта микроконтроллера
a=cos(b*5); //Этот оператор присваивания осуществляет вызов подпрограммы-функции.

Достаточно часто требуется изменять значение какой-либо переменной, т. е. и источником и приёмником данных служит одна и та же переменная. В этом случае можно воспользоваться составным оператором присваивания. Использование составного оператора сокращает исходный текст программы, например:

sum+=3; //Оператор эквивалентен оператору sum=sum+3;
Umensh-=5; //Оператор эквивалентен оператору Umensh=Umensh-5;
a*=10; //Оператор эквивалентен оператору a=a*0;

mask&=0x10;//Оператор эквивалентен оператору mask=mask&0×10;
Обычно используется для записи нулей в определённые биты переменной.

Условный оператор if обеспечивает условное выполнение операторов и записывается в следующей форме:

```
if(<выражение>)
   <operator-1>;
 [else
   <operator-2>;]
```

При этом ключевое слово else со следующим за ним исполняемым оператором представляют собой необязательную часть условного оператора. Если результат вычисления выражения равен 1 (истина), то выполняется operator-1. Если результат вычисления выражения равен 0 (ложь), то выполняется operator-2. Если выражение ложно и отсутствует оператор-2, то выполняется оператор, следующий за условным.

Пример записи условного оператора:

```
if(Wes<Min)        /*Условная операция*/
  Schetch=Schetch+1;   /*Плечо 1*/
 else
  Schetch=0;         /*Плечо 2*/
```

Оператор цикла for – это наиболее общий способ организации цикла. Оператор цикла for записывается в следующей форме:

for (выражение 1 ; выражение 2 ; выражение 3) тело цикла;

Выражение 1 обычно используется для установки начального значения переменных, управляющих циклом. Выражение 2 – это выражение, определяющее условие, при котором тело цикла будет выполняться. Выражение 3 определяет изменение переменных, управляющих циклом после каждого выполнения тела цикла. В качестве тела цикла может служить любой исполняемый оператор языка С-51, в том числе и составной оператор. Внутри составного оператора может быть заключено любое количество исполняемых операторов.

Обратите внимание, что проверка условия всегда выполняется в начале цикла. Это значит, что тело цикла может ни разу не выполниться, если условие выполнения сразу будет ложным.

Пример использования оператора for:

```
for(i=1;i<10;i++)  //от i равного 1 до 10 с шагом 1
  b=i*i;
```

Оператор цикла с проверкой условия до тела цикла while имеет следующий формат:

while (выражение) тело цикла;

Оператор while содержит условную операцию (такую же, как в операторе if), и вызывает исполнение операторов в этом блоке до тех пор, пока условие верно. Проверка условия производится до выполнения тела цикла, поэтому оператор тела цикла может быть не выполнен ни разу. В качестве выражения допускается использовать любое выражение языка Си, а в качестве тела любой оператор, в том числе пустой, одиночный или составной.

Оператор while может быть полезен для ожидания срабатывания какого-либо устройства микроконтроллера, например таймера:

```
...
while(!TF0);  //Программа ожидает переполнения таймера T0
TF0=0;
TL0=time;     //Настроить таймер T0
TH0=time>>8;  //на очередной интервал времени
...
```

Оператор цикла с проверкой условия после тела цикла do while имеет следующий формат:

do тело цикла while (выражение);

Оператор do while содержит условную операцию (такую же, как в операторе if), и вызывает исполнение операторов в этом блоке до тех пор, пока условие верно. Проверка условия производится после выполнения тела цикла, поэтому оператор тела цикла будет выполнен хотя бы один раз. В качестве выражения допускается использовать любое выражение языка Си, а в качестве тела любой оператор, в том числе пустой или составной.

Чтобы прервать выполнение цикла до того, как условие станет ложным, можно использовать оператор break.

Оператор do while в большинстве случаев соответствует одной машинной команде, поэтому может быть использован для написания эффективных по коду и быстродействию программ, например:

```
i=10
do тело цикла; while(--i<0);
```

Оператор break обеспечивает прекращение выполнения самого внутреннего из охватывающих его операторов switch, do, for, while. После выполнения оператора break управление передается оператору, следующему за прерванным оператором цикла или выбора.

Оператор continue, как и оператор break, используется только внутри операторов цикла, но в отличие от него выполнение цикла не прерывается, а начинается следующая итерация цикла.

Формат записи оператора:

```
continue;
Пример:
int main()
{int a,b;
 for(a=1,b=0;a<100;b+=a,a++)
  {if(b%2)continue;
   ... /* обработка четных сумм */
  }
 return 0;
}
```

Оператор выбора switch предназначен для организации выбора из множества различных вариантов. Формат оператора следующий:

```
switch ( выражение )
 {[объявление]
     ...
  [ case  константное-выражение1]:
[ список-операторов1]
  [ case  константное-выражение2]:
[ список-операторов2]
     ...
     ...
  [ default: [ список операторов ]]
 }
```

Выражение, следующее за ключевым словом switch в круглых скобках, может быть любым выражением, допустимым в языке СИ, значение которого должно быть целым. Отметим, что можно использовать явное приведение к целому типу.

Значение этого выражения является ключевым для выбора из нескольких вариантов. Тело оператора switch состоит из нескольких операторов, помеченных ключевым словом case с последующим константным выражением.

Так как константное выражение вычисляется во время трансляции, оно не может содержать переменные или вызовы функций. Обычно в качестве константного выражения используются целые или символьные константы.

Все константные выражения в операторе switch должны быть уникальны. Кроме операторов, помеченных ключевым словом case, может быть, но обязательно один фрагмент, помеченный ключевым словом default.

Список операторов может быть пустым, либо содержать один или более операторов. Причем в операторе switch не требуется заключать последовательность операторов в фигурные скобки.

Следует отметить, что с точки зрения программиста этот оператор выглядит очень эффектно. Однако при рассмотрении машинного кода получается довольно «страшная» конструкция с использованием таблиц переходов. Использование нескольких условных операторов if несмотря на то, что они не очень красиво выглядят в исходном тексте программы, но приводит к короткому и быстродействующему машинному коду программы.

Объявление переменных в языке программирования имеет огромнейшее значение, так как именно оно в большинстве случаев определяет объем программы. Обычно большой объем загрузочного модуля программы вызван неправильным объявлением переменных в исходном тексте программы. Обращение к внутренним регистрам микроконтроллеров и внешним ресурсам разрабатываемого устройства тоже производится с помощью заранее объявленных переменных.

В языке программирования С51 любая переменная должна быть объявлена до первого использования этой переменной. Как уже говорилось ранее, этот язык программирования предназначен для написания программ для микроконтроллеров семейства MCS-51, поэтому в составе языка должна отображаться внутренняя структура этого семейства микроконтроллеров. Эти особенности отражены во введении новых типов данных. В остальном язык программирования С-51 не отличается от стандарта ANSI.

Объявление переменной в языке программирования С51 представляется в следующем виде:

[спецификатор класса памяти] спецификатор типа
[спецификатор типа памяти] описатель
[=инициатор] [,описатель [= инициатор]]...

Описатель – идентификатор простой переменной либо более сложная конструкция с квадратными скобками, круглыми скобками или звездочкой (набором звездочек).

Спецификатор типа – одно или несколько ключевых слов, определяющие тип объявляемой переменной. В языке СИ имеется стандартный набор типов данных, используя который можно сконструировать новые (уникальные) типы данных.

Инициатор – задает начальное значение или список начальных значений, которые (которое) присваивается переменной при объявлении.

Спецификатор класса памяти – определяется одним из четырех ключевых слов языка С: auto, bit, extern, register, sbit, sfr, sfr16 static auto, extern, register, static и указывает, каким образом будет распределяться память под объявляемую переменную, с одной стороны, а с другой, область видимости этой переменной, т. е. из каких частей программы можно к ней обратиться.

Спецификатор типа памяти – определяется одним из шести ключевых слов языка С-51: code, data, idata, bdata, xdata, pdata и указывает, в какой области памяти микроконтроллера будет размещена переменная.

Отметим, что переменные с спецификатором класса памяти размещаются во внутреннем ОЗУ. Неизменяемость контролируется только на этапе трансляции. Для размещения переменной в ПЗУ лучше воспользоваться спецификатором типа памяти code

Для определения данных целого типа используются различные ключевые слова, которые определяют диапазон значений и размер области памяти, выделяемой под переменные.

Тип	**Размер памяти**		**Диапазон значений**
	бит	**байт**	
bit	1	–	от 0 до 1
char	8	1	от –128 до 127
unsigned shar	8	1	от 0 до 255
int, short	16	2	от –32 768 до 32 767
long	32	4	от –2 147 483 648 до 2 147 483 647
unsigned int, unsigned short	16	2	от 0 до 65 535
unsigned long	32	4	от 0 до 4 294 967 295
sbit	1		0 или 1
sfr	8	1	от 0 до 255
sfr16	16	2	от 0 до 65 535
float	32	4	от ±1.175 494E-38 до ±3.402 823E+38

Отметим, что ключевые слова signed и unsigned необязательны. Они указывают, как интерпретируется нулевой бит объявляемой переменной, т. е. если указано ключевое слово unsigned, то нулевой бит интерпретируется как часть числа, в противном случае нулевой бит интерпретируется как знаковый. В случае отсутствия ключевого слова unsigned целая переменная считается знаковой. В том случае, если спецификатор типа состоит из ключевого типа signed или unsigned и далее следует идентификатор переменной, то она будет рассматриваться как переменная типа int.

ПРИЛОЖЕНИЕ 3

Основные принципы работы УАПП

Через универсальный асинхронный приемопередатчик (УАПП) осуществляется прием и передача информации, представленной последовательным кодом (младшими битами вперед). УАПП разработан как последовательный порт для реализации стандарта RS-232 и является полным дуплексным портом (рис. П3.1) [7].

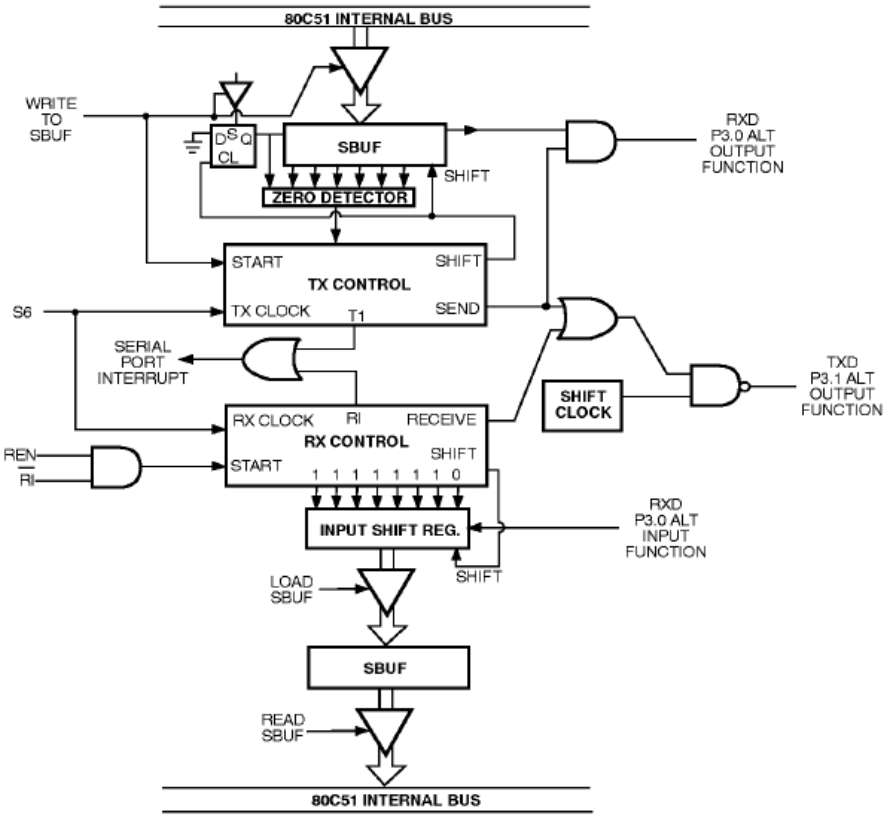


Рис. П3.1. Структурная схема
основных узлов последовательного порта

В состав УАПП, называемого часто последовательным портом, входят принимающий и передающий сдвигающие регистры, а также специальный буферный регистр (SBUF) приемопередатчика. Запись байта в буфер приводит к автоматической переписи байта в сдвигающий регистр передатчика и инициирует начало передачи байта. Наличие буферного регистра приемника позволяет совмещать операцию чтения ранее принятого байта с приемом очередного байта. Если к моменту окончания приема байта предыдущий байт не был считан из буфера, то он будет потерян.

Линии последовательного порта TXD и RXD используются для работы универсального асинхронного приемопередатчика (УАПП) на передачу и прием.

На вывод TXD поступает старт-бит, биты данных. Каждый интервал передачи бита равен 16 тактам внутреннего счетчика.

Из рис. П.3.1 видна организация основных блоков: последовательный буфер, регистр сдвига, регистр управления – на передачу и аналогично – на прием. Показаны также сигналы управления и связь с внутренней шиной данных.

Прием начинается по обнаружению перехода сигнала на входе RXD из «1» в «0».

Регистры приемника и передатчика последовательного порта доступны через регистр SBUF. На самом деле регистр SBUF это два раздельных регистра, буферный регистр передатчика и буферный регистр приемника. Запись в регистр SBUF загружает данные в регистр передатчика, а чтение SBUF обеспечивает доступ к физически раздельному регистру приемника.

Управление режимом работы УАПП осуществляется через специальный регистр с символическим именем SCON (рис. П3.2 и П3.3). Этот регистр содержит не только управляющие биты, определяющие режим работы последовательного порта, но и девятый бит принимаемых или передаваемых данных (RB8 и TB8) и биты прерывания приемопередатчика (R1 и T1).

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SM0/FE	SM1	SM2	REN	TB8	RB8	T1	R1

Рис. П3.2. Регистр управления SCON

Прикладная программа путем загрузки в старшие биты регистра SCON 2-битного кода определяет режим работы УАПП (рис. П3.4). Во всех четырех режимах работы передача из УАПП инициируется любой командой, в которой буферный регистр SBUF указан как получатель байта. Прием в УАПП в режиме 0 осуществляется при условии, что R1 = 0 и REN = 1. В режимах 1, 2, 3 прием начинается с приходом старт-бита, если REN = 1.

В бите TB8 программно устанавливается значение девятого бита данных, который будет передан в режиме 2 или 3. В бите RB8 фиксируется в режимах 2 и 3 девятый принимаемый бит данных. В режиме 1, если SM2 = 0, в бит RB8 заносится стоп-бит. В режиме 0 бит RB8 не используется.

Флаг прерывания передатчика T1 устанавливается аппаратно в конце периода передачи восьмого бита данных в режиме 0 и в начале периода передачи стоп-бита в режимах 1, 2 и 3. Соответствующая подпрограмма обслуживания прерывания должна сбрасывать бит T1.

Флаг прерывания R1 устанавливается аппаратно в конце периода приема восьмого бита данных в режиме 0 и в середине периода приема стоп-бита в режимах 1, 2 и 3. Подпрограмма обслуживания прерывания должна сбрасывать бит R1.

Символ	Функция
FE	Бит ошибки кадра. Устанавливается приемником при обнаружении ошибочного стоп-бита. Бит FE сбрасывается программно. Бит SMOD0* должен быть установлен, для фиксации значения FE.
SM0 SM1	Бит 0. Режим последовательного порта. Бит 1. Режим последовательного порта.
SM2	Разрешает функцию автоматического распознавания адреса в режимах 2 и 3. Когда SM2 = 1.
REN	Разрешение приема. Устанавливается/сбрасывается программно для разрешения/запрета приема.
TB8	9-й бит данных, который будет передаваться в режимах 2 и 3. Устанавливается/сбрасывается по усмотрению пользователя.
RB8	В режимах 2 и 3 является 9-м принятым битом данных. В режиме 1, при SM2 = 0, RB8 является принятым стоп-битом. В режиме 0 бит RB8 не используется.
TI	Флаг прерывания передатчика. Устанавливается аппаратно в конце передачи 8-го бита в режиме 0 или в начале передачи стоп-бита в других режимах. Должен быть сброшен программно.
RI	Флаг прерывания приемника. Устанавливается аппаратно в конце приема 8-го бита в режиме 0 или в середине приема стоп-бита в других режимах (если SM2 = 0, см. SM2). Должен быть сброшен программно.
Примеч.: *SMOD0 находится в 7 бите регистра PCON.	

Рис. П3.3. Регистр управления последовательным портом

SM0	SM1	Режим	Описание	Скорость передачи
0	0	0	8 бит	$F_т/12$
0	1	1	УАПП, 10 бит	переменная
1	0	2	УАПП, 11 бит	$F_т/64$ или $F_т/32$
1	1	3	УАПП, 11 бит	переменная

Рис. П3.4. Установка режима последовательного порта: SM[2:0]

Последовательный порт имеет 4 режима работы.

Скорость передачи в некоторых режимах фиксирована, в других определяется таймером T1 или T2.

Во всех четырех режимах передача начинается любой командой, которая использует SBUF как регистр назначения. Прием в режиме 0 начинается при условии: RI = 0, REN = 1. Прием в других режимах начинается с приходом старт-бита, если бит REN установлен.

Режим работы 0 задаётся записью комбинации 00 в биты SM0 и SM1 регистра SCON. В синхронном режиме работы информация передается и принимается через вывод входа приемника RxD, т. е. в этом режиме работы последовательный порт работает в симплексном режиме. Через вывод TxD выдаются импульсы синхронизации, которые сопровождают каждый информационный бит. Скорость передачи зависит только от частоты кварцевого резонатора $f_1 = \left(1/12\right) \cdot f_{\text{рез}}$. За один машинный цикл последовательный порт передает один бит информации.

Режим работы 1 задаётся записью комбинации 01 в биты SM0 и SM1 регистра SCON. В асинхронном режиме работы информация передается через вывод передатчика последовательного порта микроконтроллера TxD, а принимается через вывод входа приемника RxD, в этом режиме последовательный порт может работать в дуплексном режиме; передача и приём информации могут вестись независимо друг от друга. Скорость передачи в режиме 1 переменная, если использовать таймер T1 и/или T2 – один на передачу, другой на прием для управления скоростью передачи.

В режимах 1, 2 и 3 скорость приема/передачи зависит от значения управляющего бита SMOD регистре PCON.

Режим работы 2, основной особенностью которого является передача девятого информационного бита, который может быть использован для контроля достоверности передаваемой информации. Для вычисления чётности передаваемого байта можно воспользоваться аппаратным вычислителем, подключенным к аккумулятору ЦП. Результат вычисления чётности байта сохраняется в бите чётности P регистра PSW, откуда его можно скопировать в девятый информационный бит последовательного порта TB8, расположенный в регистре управления последовательным портом SCON. Скорость передачи фиксирована, как и в режиме 0, но может иметь два значения определяемых выражениями $f_2 = \left(2^{\text{SMOD}}/64\right) \cdot f_{\text{рез}}$: $f_2 = \left(1/64\right) \cdot f_{\text{рез}}$ (SMOD = 0) или, $f_2 = \left(1/32\right) \cdot f_{\text{рез}}$ (SMOD = 1). Она измеряется в «бит/с» или «бод», поэтому генератор синхроимпульсов называют BRG – Baud Rate Generator – генератор бодовой скорости (БС).

Режим работы 3 аналогичен режиму 2, но скорость передачи может изменяться, как в режиме 1, и можно использовать таймер Т1 и/или Т2 для задания скорости передачи.

В режимах 1 и 3 в формировании частоты передачи кроме управляющего бита SMOD принимает участие таймер 1. Сам Т/С1 может работать и как таймер, и как счетчик событий в любом из трех режимов.

Однако наиболее удобно использовать режим таймера с автоперезагрузкой (старшая тетрада TMOD = 0010B), при этом частота передачи определяется следующим выражением:

$$f_{1,3} = \frac{2^{\text{SMOD}} \cdot f_{\text{рез}}}{32 \cdot 12 \cdot (256 - \text{TH1})},$$

регистр таймера 1 (TH1) принимает значения от 0 до 255.

ПРИЛОЖЕНИЕ 4

Схема прерываний микроконтроллера 8051

На рис. П4.1 показана упрощенная схема прерываний микроконтроллера 8051.

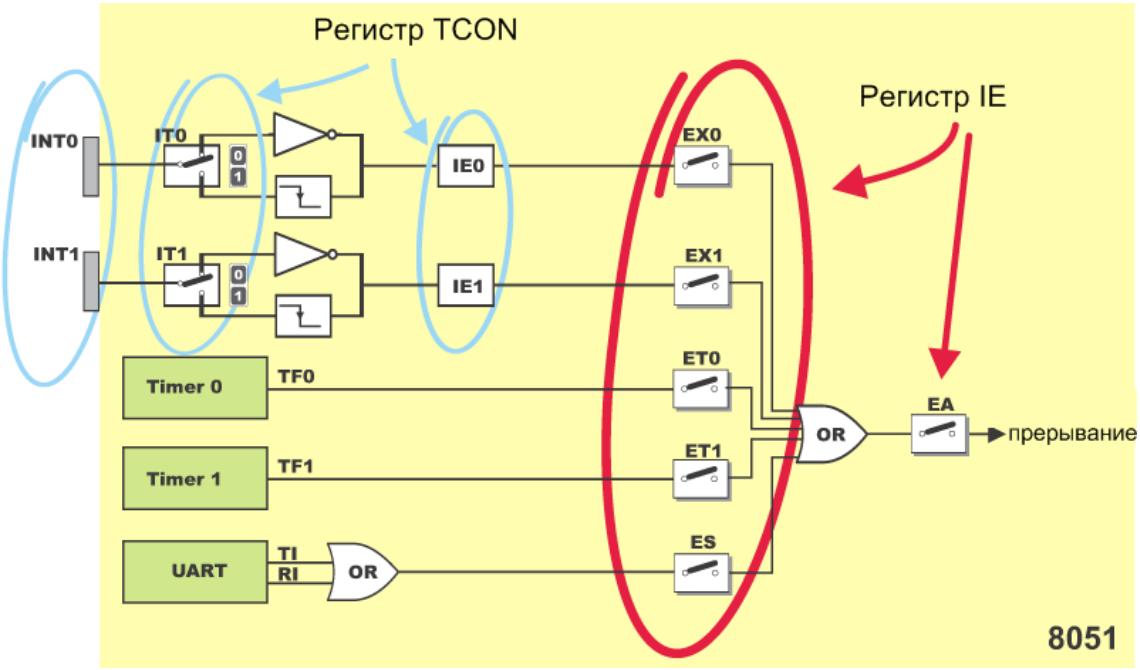


Рис. П4.1. Схема прерываний МК51

Внешние прерывания $\overline{\text{INT0}}$ и $\overline{\text{INT1}}$ могут быть вызваны либо уровнем входного сигнала, либо переходом сигнала из 1 в 0 на входах 8051 в зависимости от значений управляющих бит IT0 и IT1в регистре TCON. При наличии внешнего прерывания в регистре TCON устанавливаются флаги IE0 и IE1, они инициируют переход к соответствующей подпрограмме прерывания. Если прерывание вызвано переходом сигнала из 1 в 0, то сброс флага прерывания выполняется аппаратно. Если прерывание вызвано уровнем входного сигнала, то сброс флага необходимо осуществлять программно при обработке прерывания.

Флаги запросов прерывания от таймеров TF0 и TF1 сбрасываются автоматически при передаче управления подпрограмме обслуживания прерывания.

Флаги запросов прерывания TI и RI устанавливаются аппаратно при передаче и приеме данных через UART. Сброс бит необходимо осуществлять программным путем. ***Адрес вектора прерываний USART 0x23***.

В блоке регистров специальных функций находятся два регистра, предназначенных для управления режимом прерываний (IE – рис. П4.2) и уровнями приоритета (IP). Описание регистра IE приведено на рис. П4.3. Все названные флаги прерывания программно доступны.

D7	D6	D5	D4	D3	D2	D1	D0
EA	-	-	ES	ET1	ET1	ET0	EX0

Рис. П4.2. Структура регистра управления режимом прерываний

	Символ	Назначение бита
D7	EA	Бит блокировки прерывания. Сбрасывается программно для запрета всех прерываний независимо от состояний D0-D4
D4	ES	Бит разрешения прерывания от приемопередатчика. Устанавливается/сбрасывается программно для разрешения/запрета прерываний от флагов TI и RI
D3	ET1	Бит разрешения прерывания от таймера 1. Устанавливается/сбрасывается программно для разрешения/запрета прерываний от таймера 1 (флаг TF1)
D2	EX1	Бит разрешения внешнего прерывания 1. Устанавливается/сбрасывается программно для разрешения/запрета внешнего прерывания 1 (флаг INT1)
D1	ET0	Бит разрешения прерывания от таймера 0. Устанавливается/сбрасывается программно для разрешения/запрета прерываний от таймера 0 (флаг TF0)
D0	EX0	Бит разрешения внешнего прерывания 0. Устанавливается/сбрасывается программно для разрешения/запрета внешнего прерывания 0 (флаг INT0)

Рис. П4.3. Назначение битов регистра управления режимом прерываний

Система прерываний формирует аппаратный вызов, соответствующей подпрограмме обслуживания, помещает в стек содержимое счетчика команд (PC) и загружает в него адрес вектора соответствующей подпрограммы обслуживания прерывания. По адресу вектора должна быть записана команда безусловного перехода к начальному адресу подпрограммы прерывания. При выходе из подпрограммы прерывания в счетчик команд из стека загружается сохраненный адрес возврата основную программу.

ПРИЛОЖЕНИЕ 5

Управление ЦАП MCP4921 по интерфейсу SPI

SPI (последовательный периферийный интерфейс, шина SPI) – последовательный синхронный стандарт передачи данных в режиме полного дуплекса, разработанный компанией Motorola для обеспечения сопряжения микроконтроллеров и периферии. SPI также иногда называют четырехпроводным интерфейсом.

В отличие от стандартного последовательного порта SPI является синхронным интерфейсом, в котором любая передача синхронизирована с общим тактовым сигналом, генерируемым ведущим устройством (процессором). Принимающая периферия (ведомая) синхронизирует получение битовой последовательности с тактовым сигналом. К одному последовательному периферийному интерфейсу ведущего устройства – микросхемы может присоединяться несколько микросхем. Ведущее устройство выбирает ведомое для передачи, активируя сигнал «выбор кристалла» (CS, - Chip Select) на ведомой микросхеме. Данный сигнал может иметь активный уровень или низкий, или высокий в зависимости от спецификации конкретного производителя SPI устройств.

Периферия, не выбранная процессором, не принимает участие в передаче по SPI, поэтому в SPI используется четыре различных сигнала:

1. SCK – тактирование последовательной связи (синхронизация);
2. MOSI – (Master Out Slave In) последовательный ввод (данные от ведущего к ведомому);
3. MISO – (Master In Slave Out) последовательный вывод (данные от ведомого к ведущему).
4. $\overline{\text{CS}}$ или $\overline{\text{SS}}$ – выбор микросхемы, выбор ведомого.

В связи с отсутствием общих технических требований к SPI-интерфейсу временная диаграмма тактового сигнала зависима от устройства (микросхемы), поскольку каждый производитель использует собственную временную диаграмму. Рассмотрим более подробно формат данных и особенности программирования микросхемы MCP4921.

Данная микросхема ЦАП позволяет взаимодействовать с микроконтроллером по последовательному периферийному интерфейсу в двух режимах: режим 0,0 и режим 1,1 (рис. П5.1). Команда управления состоит из 16 бит и используется для управления режимами работы ЦАП и передачи данных. Формат команды управления показан на рис. П5.2.

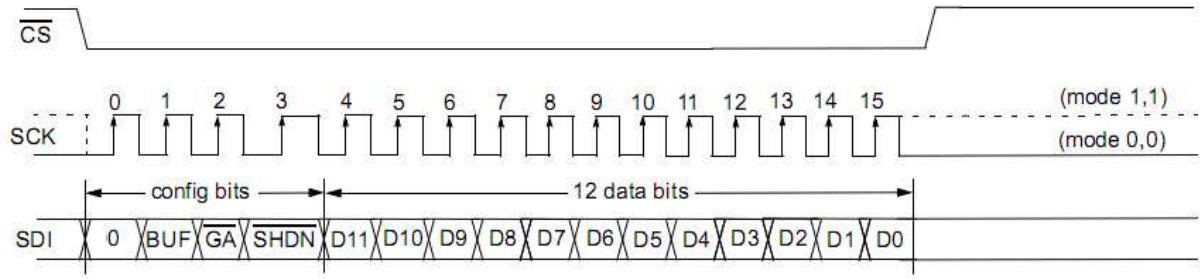


Рис. П5.1. Эпюры напряжений сигналов SPI при передаче команды управления

Старший байт:

W-x	W-x	W-x	W-0	W-x	W-x	W-x	W-x
0	BUF	$\overline{GA}$	$\overline{SHDN}$	D11	D10	D9	D8
бит 15							бит 8

Младший байт:

W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
D7	D6	D5	D4	D3	D2	D1	D0
бит 7							бит 0

Рис. П5.2. Формат команды управления ЦАПом

Формат команды управления:

14 бит BUF: управление входным буфером: 1 = буферизирован; 0 = не буферизирован.

13 бит $\overline{GA}$: управление выходным усилителем:

1 = 1x ($V_{OUT} = V_{REF} \cdot D/4096$); 0 = 2x ($V_{OUT} = 2 \cdot V_{REF} \cdot D/4096$),

где V_{OUT} – выходное напряжение, V_{REF} – опорное напряжение, D – 12-битное число данных.

12 бит $\overline{SHDN}$: Управление выходом ЦАПа: 1=Выходной буфер включен; 0= Выходной буфер отключен.

Биты 11-0– D11…D0: биты данных, имеющие значения от 0 до 4095.

Прасолов Александр Александрович
Шпак Станислав Антонович

МИКРОКОНТРОЛЛЕРЫ В РАДИОСИСТЕМАХ

Методические указания
к выполнению лабораторных работ

Редактор *Л. А. Медведева*

План 2013 г., п. 84

Подписано к печати 15.04.2013
Объем 3,25 усл.-печ. л. Тираж 110 экз. Заказ 311
Редакционно-издательский центр СПбГУТ.
191186 СПб., наб. р. Мойки, 61
Отпечатано в СПбГУТ