Приложение 1

Программа работ лабораторного практикума

П1. Программа первого цикла работ: «Изучение вычислительных возможностей МК»

Цель работы:

- знакомство с программно-аппаратным комплексом поддержки проектирования микроконтроллерных систем на базе МК SAB80C515;
- изучение системы команд МК семейства MCS51 на примере выполнения простейших программ.

Программа работы:

П1.1. Изучите состав и назначение основных устройств и блоков программно-аппаратного комплекса поддержки проектирования микроконтроллерных систем на базе МК семейства MCS51 (среды проектирования Shell51).

П1.2. На примере тестовой программы proba.asm **познакомьтесь** с полным циклом создания прикладного программного обеспечения. Программа proba.asm осуществляет обнуление ячеек заданной области внутренней памяти данных МК.

```
        ; proba.asm

         org 8400h
        mov a,#0h
        mov r0,#50h
m1:     mov @r0,a
        inc r0
        cjne r0,#60h,m1
        ret
```

В соответствии с общими положениями создания прикладного ПО (см. разд.3 пособия, рис 3.1) выполните трансляцию программы proba.asm. Убедитесь в отсутствии в программе синтаксических ошибок. При обнаружении таких ошибок (результат трансляции программы отображается в специальном окне оболочки Shell51, перечень ошибок трансляции в окне «листинг») устраните выявленные ошибки.

Программа proba.asm, как и другие программы этого цикла работ, не предполагает использование устройств ввода/вывода микроконтроллерной системы. Проверка работоспособности такой программы и ее отладка могут быть выполнены как с использованием симулятора (программной модели МК), так и при непосредственном запуске программы на МК и контроле результатов выполнения на инструментальной ВМ. Проверьте работу программы обоими способами, сравните результаты.

При загрузке, трансляции и чтении результатов выполнения программы пользуясь правилами-рекомендациями, приведенными в информационном окне рабочего поля экрана инструментальной ВМ, изучите основные функции среды Shell51 и способы их применения.

Для понимания общего назначения программы и сути каждого оператора (директив ассемблера и ассемблерных команд) используйте справочную информацию, представленную в разделе программирование МК. Дополните текст программы смысловыми комментариями к выполняемым командам. В отчете приведите ответы на следующие вопросы:

- какие способы адресации операндов используются в командах программы?
- какую функцию реализует директива org 8400h?
- с помощью какой команды в программе реализуется цикл? Поясните действие этой команды.

П1.3. Для заполнения ячеек заданной области внутренней памяти данных МК линейно возрастающими (или линейно уменьшающимися) значениями **модифицируйте программу proba.asm и выполните** ее. Диапазон области адресуемой памяти и значения, записываемые в конкретные ячейки памяти, выберите самостоятельно. В отчете по лабораторной работе **прокомментируйте все этапы цикла создания** прикладной модифицированной программы, включая этап обнаружения искусственно введенной синтаксической ошибки.

Знакомство с системой команд МК семейства MCS51 осуществляется в процессе выполнения ряда простейших программ.

П1.4. Разработайте и выполните программу вычисления арифметического выражения заданного вида. Исходные данные – байтовые переменные. Операнды и результат вычисления разместите в ячейках внутренней памяти. При выполнении данного и последующих заданий номер варианта совпадает с номером рабочего места

Варианты задания исходных данных программы «вычисление арифметического выражения заданного вида»:

1. $F = A * [(B + C)/2 – 2*B / C]$, где $B \leq 127$; $(B + C)/2 > 2*B / C$; $C \neq 0$;
2. $F = 1/2 *(A + B) * (C - A/B)$, где $C > A/B$; $B \neq 0$;
3. $F = 1/4 *(A + 2*B) * (B - B/A)$, где $B \leq 127$; $A \neq 0$;
4. $F = (A*B - B)/A * (2*A + B)$, где $A \neq 0$;
5. $F = 2*A * [(B - C) + B/(2*C)]$, где $A \leq 127$; $B > C$; $C \neq 0$;
6. $F = [(A - B)/ (C + 1) + 12h] * C$, где $A > B$; $(C + 1) \leq 127$; $(C + 1) \neq 0$;
7. $F = A + [(B + C) + A * C]$
8. $F = A * [(B / C) + B * C] / 2$, где $C \neq 0$;
9. $F = [(A * C) + B - C] / A$, где $A \neq 0$;
10. $F = [(A - C) * B – 10h] / A$, где $A \neq 0$; $A > C$; $(A - C) * B \leq 255$;

П1.5. Разработайте и выполните программу вычисления логического выражения заданного вида с использованием команд булевого процессора.

Исходные данные – битовые переменные. Операнды и результат вычисления разместите в ячейках внутренней памяти данных битового процессора (в диапазоне адресов 20h –2Fh).

Варианты задания исходных данных программы «вычисление логического выражения заданного вида»:

1. Y = (a AND (NOT b)) XOR (c OR d)
2. Y = (a XOR (b AND c)) OR d
3. Y = (a AND (NOT b)) XOR (c AND d)
4. Y = ((NOT a) AND c) XOR (b AND d)
5. Y = (a AND (NOT b)) XOR (c AND d)
6. Y = (a XOR b) AND (c AND d)
7. Y = (a AND (NOT b)) OR (c XOR d)
8. Y = (a OR (NOT b)) XOR (c OR d)
9. Y = (a AND (NOT c)) XOR (a AND d)
10. Y = ((NOT a) AND (NOT b)) OR (c XOR d)

Здесь a, b, c, d - битовые переменные.

Поскольку в системе команд битового процессора отсутствует команда XOR, при реализации операции XOR используйте выражение:

a XOR b = (a AND (NOT b)) OR ((NOT a) AND b).

П1.6. Разработайте и выполните программу, которая осуществляет заполнение последовательных ячеек **внешней памяти** значениями, линейно изменяющимися в заданных диапазонах.

Варианты задания исходных данных программы «заполнение последовательных ячеек внешней памяти значениями, линейно изменяющимися в заданном диапазоне»:

1. 0 – 10h – 0
2. 10h – 0 – 10h
3. 20h – 30h –20h
4. 30h – 20h – 30h
5. 10h – 18h – 08h –10h
6. F8h – FFh – F0h – F8h
7. 70h – 60h – 70h
8. 30h – 28h – 38h – 30h
9. 30h – 38h – 28h – 30h
10. 20h – 18h – 28h –20h

П1.7. Разработайте и выполните программу функциональной обработки данных..

Варианты задания программа «функциональной обработки данных»:

1. Поиск минимального, максимального и среднего арифметического в массиве.
2. Определение общих для двух массивов элементов, с занесением их в новый массив.
3. Выбор из массива элементов, делящихся нацело на K, с занесением их в новый массив.
4. Подсчет числа вхождений заданной константы в массив.

5. Определение наибольшего общего делителя двух чисел.
6. Определение наименьшего общего кратного двух чисел.
7. Вычисление k-го элемента ряда Фибоначчи.
8. Целочисленное вычисление квадратного корня числа X. При выполнении задания используйте представление квадрата числа суммой k членов ряда $k^2=1+3+...+(2k-1)$.
9. Разработать простейшую программу генерации псевдослучайных чисел (см. приложение 6).

Если по заданию необходимо работать с массивами данных, они должны располагаться во внешней памяти данных.

П2. Программа второго цикла работ: «Работа с портами МК»

Цель работы:

- знакомство с организацией взаимодействия МК с внешними устройствами (цифровыми и аналоговыми), подключаемыми с помощью портов МК;
- овладение приемами программирования периферийных устройств, входящих в состав лабораторного стенда.

При выполнении работ данного цикла необходимо разработать структуру информационных связей МК с подключаемыми внешними устройствами (клавиатурой, модулем ЖКИ, датчиками аналоговых сигналов) и реализовать ее, соединив разъемы портов МК с разъемами подключаемых устройств. Существует сравнительно большое число вариантов подключения внешних периферийных устройств к МК. При выполнении данного и следующих циклов работ структура информационных связей МК с подключаемыми внешними устройствами определяется вариантом задания, представленным в табл.13. Номер варианта совпадает с номером Вашего рабочего места.

Варианты назначения портов ввода-вывода МК,
используемых при подключении внешних устройств Таблица 13

N	Выходной порт МК (вход клавиатуры)	Порт ввода МК (выход клавиатуры)	Шина DB ЖКИ	Шина управления ЖКИ RS R/W E	Входы подключения аналоговых сигналов Потенц-р	Интегр-р	Линия порта, на которой формируется меандр (ШИМ-сигнал) (точка подключения осциллографа)
1.	P1.7 – P1.4	P5.7 – P5.4	P4	P5.0 P5.2 P5.3	ach0	ach7	P1.3
2.	P4.3 – P4.0	P1.7 – P1.4	P5	P4.4 P4.6 P4.7	ach1	ach6	P1.2
3.	P1.7 – P1.4	P5.7 – P5.4	P4	P5.0 P5.2 P5.3	ach2	ach5	P1.1
4.	P4.3 – P4.0	P4.7 – P4.4	P5	P1.**4** P1.6 P1.7	ach3	ach4	P1.1
5.	P5.7 – P5.4	P5.3 – P5.0	P4	P1.4 P1.6 P1.7	ach4	ach7	P1.2
6.	P5.3 – P5.0	P1.7 – P1.4	P4	P5.4 P5.6 P5.7	ach5	ach6	P1.3
7.	P4.7 – P4.4	P4.3 – P4.0	P5	P1.**4** P1.6 P1.7	ach6	ach0	P1.3
8.	P1.7 – P1.4	P5.7 – P5.4	P4	P5.0 P5.2 P5.3	ach7	ach1	P1.2
9.	P5.7 – P5.4	P1.7 – P1.4	P4	P5.0 P5.2 P5.**3**	ach3	ach6	P1.1
10.	P1.7 – P1.4	P4.7 – P4.4	P5	P4.0 P4.**2** P4.**3**	ach5	ach2	P1.2

11.	P4.7 – P4.4	P4.3 – P4.0	P5	P1.**4** P1.6 P1.7	ach0	ach1	P1.3

В качестве примера на рис. П1 приведена структура информационных связей МК с подключаемыми внешними устройствами, реализованная в соответствии с заданием варианта 11.

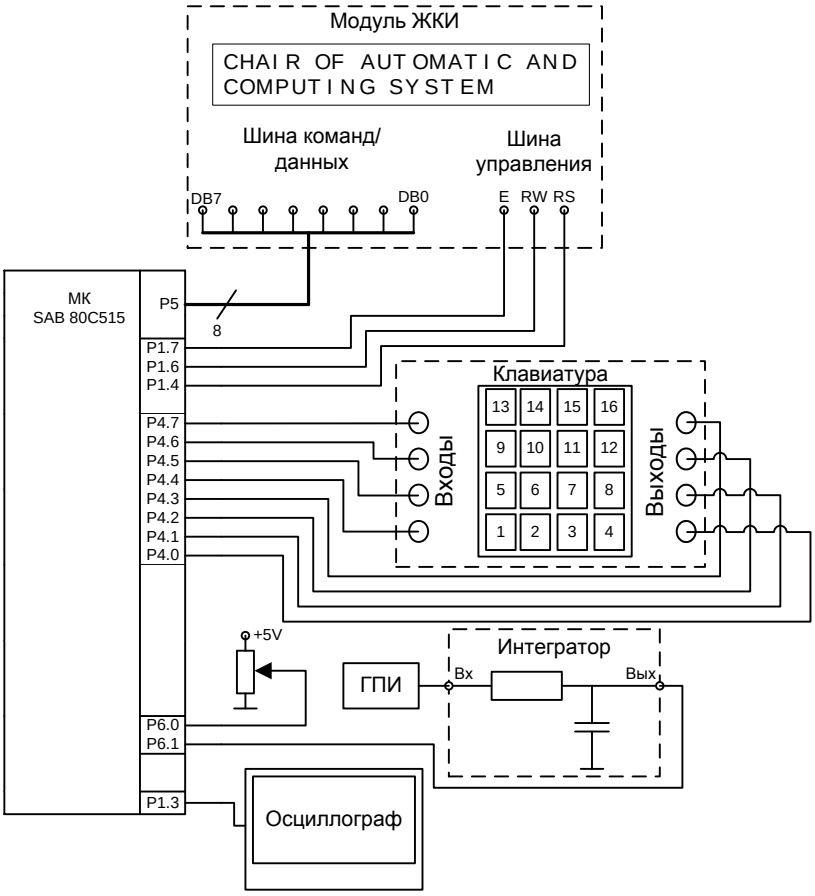


Рис.П1. Структура информационных связей МК с подключенными внешними устройствами

Программа работы:

П2.1. Познакомьтесь с организацией и принципом работы модуля ЖКИ (см. п. 4.2.2 пособия).

Разработайте и выполните программу, реализующую вывод на экран ЖКИ двухстрочного текста (см. рисунок), в котором используются различные алфавиты (латиница и кириллица)

‘Microcontrollers20_ _ Фамилия студента(ов) в русской транскрипции

При разработке программы используйте модуль *indic.asm* (п.4.2.2). Результат выполнения представьте в отчете.

П2.2. Разработайте программу *klav.asm*, реализующую опрос клавиатуры и запись кодов состояния клавиш в ячейки памяти с адресами 30h . . . 33h.

Результат выполнения программы (содержимое карты памяти клавиатуры) представьте в виде таблицы, отображающей состояние кодов клавиш при их нажатии.

Номер клавиши	Входное воздействие Выходы порта P_i $P_{i.j+3}$ $P_{i.j+2}$ $P_{i.j+1}$ $P_{i.j}$	Состояние карты памяти при нажатии клавиши, соответствующей строки (содержимое ячеек) 30h 31h 32h 33h

П2.3. Модифицируйте программу «опрос клавиатуры», дополнив ее модулем, формирующим номер нажатой клавиши. Присвойте ей имя *klav1.asm*.

Если нажата одна клавиша в выбранную ячейку внутренней памяти МК, например 34h, записывайте 16-ричный или десятичный код номера нажатой клавиши. При отсутствии нажатых клавиш фиксируйте в ячейке 34h код 00, а при нажатии нескольких клавиш (более одной) – код FFh. Результат выполнения программы представьте в виде копии экрана вкладки «Окна памяти».

При разработке модифицированной программы *klav1.asm* целесообразно использовать директиву *include asms\klav.asm*, с помощью которой программный модуль *klav.asm* подключается к разрабатываемой программе.

Напомним, что директива *include <имя файла>* указывает ассемблеру на необходимость трансляции текста, расположенного в указываемом файле, начиная с адреса, следующего за адресом последней команды текущего файла. При использовании директивы include <имя файла> часто фиксируются ошибки, связанные с наложением переменных и участков программ (при любых вызовах по общим адресам формируемое обращение будет производиться к участку кода последнего оттранслированного модуля). Поэтому при использовании директивы include <имя файла> необходимо выполнять следующие требования:

- имя файла, указываемого в директиве *include <имя файла>* не должно превышать 8 символов;
- следует избегать абсолютных адресов в подключаемых файлах;
- нельзя использовать одинаковые метки в выполняемой программе и подключаемых файлах.

П2.4. Разработайте программу, отображающую на экране модуля ЖКИ номер нажатой клавиши, используя программы indic.asm (п.4.2.2) и *klav1.asm*, подключив их к разрабатываемой программе директивой *include <имя файла>*.

При выводе текста «Number of button Nx» номер нажатой клавиши *N*x с помощью команды пересылки с прямой адресацией приемника предварительно должен быть помещен в видеобуфер: str1: ‘Number of button Nx’.

Изучение особенностей работы МК при подключении датчиков с аналоговым выходом.

В качестве источников аналоговых сигналов в стенде используются источник регулируемого напряжения, представленный сигналом с выхода потенциометра, и сигнал с выхода интегратора, на вход которого подается сигнал с выхода генератора прямоугольных импульсов (ГПИ) с регулируемой частотой и скважностью. Формирование цифровых кодов входных аналоговых сигналов выполняется с помощью многоканального аналого-цифрового преобразователя в составе МК. Структура АЦП и особенности режимов его работы рассмотрены в подразд.2.5 пособия.

П2.5. Разработайте и выполните программу аналого-цифрового преобразования и вывода на ЖКИ цифровых эквивалентов аналоговых сигналов от двух источников.

В соответствии с заданием для вашего «рабочего места» **разработайте** схему подключения указанных источников аналоговых сигналов к МК и **выполните** необходимые соединения: выход потенциометра соедините с заданным входом многоканального АЦП, а выход интегратора – с другим входом многоканального АЦП. Вход интегратора необходимо соедините с выходом генератора прямоугольных импульсов.

Интегратор выполняет функции ФНЧ. При невысокой скорости изменения параметров колебаний ГПИ он преобразует выходной сигнал генератора в напряжение. Диапазон изменения напряжения на выходе интегратора определяется частотой и скважностью прямоугольных импульсов. **Определите этот диапазон** с помощью осциллографа.

Вывод на ЖКИ цифровых эквивалентов аналоговых сигналов от двух источников целесообразно реализовать в виде циклической процедуры, в каждом цикле которой выполняются следующие действия:

- преобразование напряжения с выхода потенциометра в цифровой код; запоминание результата в ячейке внутренней памяти $N_{\text{пот}}$;
- преобразование напряжения с выхода интегратора в цифровой код; запоминание результата в ячейке памяти $N_{\text{инт}}$;
- конвертирование двоичного кода напряжения ($N_{\text{источника сигнала}}$) в трехразрядное десятичное представление (сотни, десятки, единицы) в ASCII-кодах;
- вывод на экран ЖКИ цифрового кода преобразованного напряжения конкретного источника сигнала;
- временная задержка для отображения преобразованных значений напряжения на экране ЖКИ.

При разработке программы вывода на ЖКИ цифровых эквивалентов аналоговых сигналов от двух источников используйте программы indic.asm (п.4.2.2) и adc_read (п.2.5), подключив их к разрабатываемой программе директивой include <имя файла>.

Для повышения точности преобразования аналоговых сигналов каждый канал АЦП может настраиваться с учетом диапазона изменения

преобразуемого сигнала. Настройка осуществляется путем программирования регистра DAPR.

Диапазон изменения сигнала на выходе потенциометра составляет 0 – 5 В.

Диапазон изменения напряжения на выходе интегратора, как отмечено выше, определяется параметрами генератора прямоугольных импульсов.

Для каждого значения сигнала на выходе интегратора выполните два преобразования: первое - для диапазона опорных напряжений (0-5В) и второе – для более узкого диапазона, выбранного в соответствии с диапазоном изменения напряжения на выходе интегратора. Результаты представьте в виде таблицы (см. пример представления результатов измерения/преобразования в табл.14).

Таблица 14

Диапазон опорных напряжений, δ	Напряжение датчика, измеренное осциллографом	Цифровой эквивалент, измеренного значения	Результат преобразования (цифровой код ЖКИ, зафиксированный на выходе АЦП)	Погрешность измерения Δ
0 – 5 В (DAPR = 0) 20 мВ	4,3 В	215	219	4*20 = 80 мВ
0 – 5 В (DAPR = 0) 20 мВ	1,5 В	74	76	2*20 =40мВ
1,25-3,4375 В (DAPR =В4), 8,5 мВ	1,5 В	176	178	2*8,5=17 мВ

δ – дискрета преобразования: $\delta =$ Диапазон / 256

Δ – погрешность измерения: $\Delta = | N_{осц} - N_{ацп} | \times \delta$

В каждом диапазоне изменения аналогового сигнала необходимо выполнить не менее 5 преобразований входного сигнала.

Преобразуемый аналоговый сигнал контролируйте с помощью осциллографа, подключая его к соответствующему входу порта Р6 (канала АЦП). Цифровые коды сигналов с выхода АЦП считывают с экрана ЖКИ,

Среда проектирования Shell51 не только поддерживает процесс проектирования пользовательских программ (их написание, редактирование, трансляцию, выполнение, отображение результатов), но и содержит средства, упрощающие отладку пользовательского ПО.

Оболочка среды содержит вкладку «Окна управления» (см. приложение 3). При относительно простой модификации пользовательской программы в окне «Выход» вкладки на экране ПК можно отображать результат ее

функционирования. Отображаемая функция представляет собой развернутое во времени значение ячейки внутренней памяти МК, адрес которой указан в поле «Адрес выхода» вкладки.

При изучении особенностей работы аналого-цифрового преобразователя МК SAB 80C535 в окне «Выход» вкладки «Окна управления» можно наблюдать динамику изменения во времени сигнала аналогового датчика. Для этого необходимо модифицировать выполняемую программу аналого-цифрового преобразования, вставив в ее циклический участок команду lcall 128h, и в окне «Адрес выхода» вкладки указать адрес ячейки памяти с кодом преобразованного сигнала выбранного датчика. Модифицированная программа обеспечивает совместную работу пользовательской программы с программой вкладки «Окна управления» оболочки Shell51.

При выполнении задания, вращая ручку потенциометра, изменяйте выходной аналоговый сигнал датчика. Цифровые коды аналогового сигнала с выхода АЦП ($N_{пот}$ или $N_{инт}$) с помощью программы монитора, вызываемой командой lcall 128h, передаются в программный модуль «Окна управления». При нажатии кнопки «Пуск» вкладки динамика изменения сигнала датчика отображается в окне «Выход» вкладки

П2.6. Модифицируйте и **выполните** программу аналого-цифрового преобразования сигналов аналоговых датчиков, которая отображает изменяющийся во времени сигнал датчика.

Осциллограммы изменяющихся во времени аналоговых сигналов с выходов обоих датчиков для различных диапазонов опорных напряжений, задаваемых при программировании регистра DAPR, **приведите** в отчете.

П3. Программа третьего цикла работ: «Изучение таймеров и системы прерываний»

Цель работы:

- приобретение практических навыков программирования таймеров
- изучение принципов программного деления частоты;
- знакомство с организацией и использование многоуровневой системы прерываний.

При выполнении работ данного цикла необходимо разработать структуру информационных связей МК с подключаемыми внешними устройствами (клавиатурой, модулем ЖКИ, осциллографом) и реализовать их, соединив разъемы портов МК с разъемами подключаемых устройств. Вариант задания (табл.13) совпадает с номером рабочего места.

Перед выполнением работ данного цикла познакомьтесь со структурой и основными режимами работы счетчиков/таймеров МК SAB 80C515, а также с принципом организации системы прерываний этого МК (см. п.п.2.6, 2.7).

П3.1. Разработайте и выполните программу, генерирующую на заданном выводе порта МК импульсный сигнал прямоугольной формы

(меандр) с частотой, определяемой номером $N_{клав}$ нажатой клавиши блока клавиатуры (F = $N_{клав}$ * 100 Гц).

При разработке программы используйте программные модули, рассмотренные в п.п. 4.2.1 и 2.6.1. Двухбайтные константы перезагрузки таймера, определяющие частоту генерируемого меандра, целесообразно разместить в строке внешней памяти. При нажатии клавиши ее номер используйте в качестве индекса соответствующей константы в строке. Упрощенный алгоритм программы генерирования меандра управляемой частоты представлен на рис.5.2.

Формируемые колебания наблюдайте с помощью осциллографа. **Измерьте** параметры формируемых меандров при расчетных значениях $N_{загр}$.

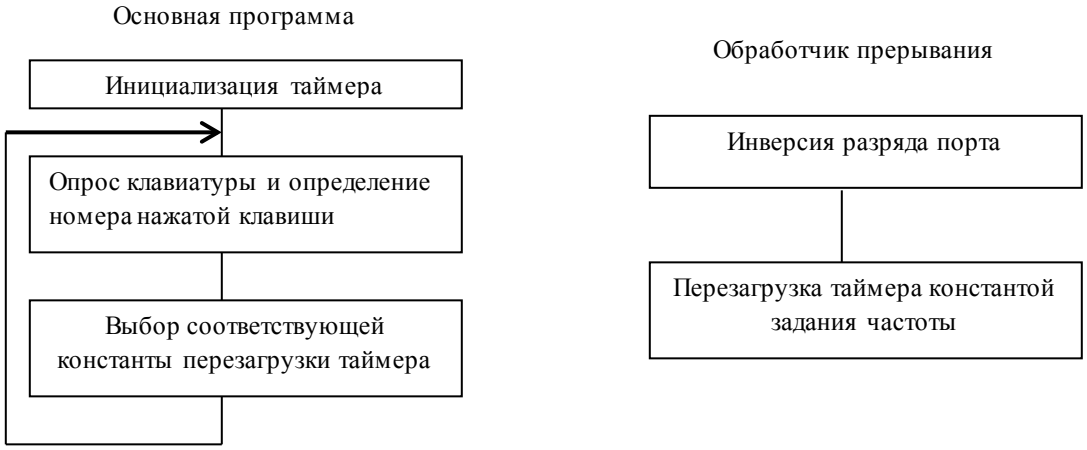


Рис.5.2 Алгоритм программы генерирования меандра требуемой частоты

В п.2.6.1 было отмечено, что временные затраты на вызов и выполнение команд обработчика при переполнении счетчика таймера τ_{ov} увеличивают длительность импульсов, формируемых таймером. Это приводит к ошибкам формирования временных интервалов, определяющих период формируемого меандра. Для компенсации временные затраты на программную перезагрузку в обработчике значение перезагружаемой константы $N_{загр}$ должно быть скорректировано.

Определите скорректированные значения констант перезагрузки $N_{загр}$ и **измерьте** параметры формируемых меандров при этих значениях $N_{загр}$.

Результаты выполнения программы представьте в виде таблицы:

$N_{клав}$ Номер нажатой клавиши	Расчетное значение частоты	Измеренное значение частоты при расчетном значении $N_{загр}$	Значение частоты при скорректированном значении $N_{загр}$

П3.2. Разработайте и выполните программу формирования ШИМ-сигнала заданной частоты (F= N*100 Гц, где N - номер Вашего

рабочего места). При разработке программ целесообразно использовать программные модули, рассмотренные в п.п. 2.5 и 2.6.3.

Формирование ШИМ-сигнала с фиксированной скважностью рассмотрено в п.2.6.3. Там же приведен алгоритм формирования ШИМ-сигнала с управляемой скважностью. Для управления скважностью используйте цифровой код АЦП, преобразующего аналоговый сигнал с выхода потенциометра. Для запоминания кода управления скважностью используйте ячейку памяти pwm_var. Поскольку на выходе АЦП формируется 8-разрядный код, а для управления скважностью используется 16-разрядный код, содержимое ячейки pwm_var необходимо масштабировать (см. п. 2.6.3). Работу ШИМ-генератора контролируйте с помощью осциллографа.

Результат выполнения программы представьте в виде зависимости измеренных значений τ_{pwm} как функция напряжения на выходе потенциометра $\tau_{pwm} = f(U_{потенц})$. Для измерения указанных параметров используйте осциллограф.

П3.3. Модифицируйте программу предыдущего задания, используя для управления скважностью ШИМ-сигнала цифровые коды управляющего воздействия, формируемого инструментальной ЭВМ при работе с вкладкой «Окна управления» (Окно «Вход»).

Для связи с инструментальной ЭВМ в циклический участок модифицированной программы вставьте команду lcall 128h – вызова однократного сеанса связи, а в окне «адрес входа» вкладки «Окна управления» установите адрес ячейки памяти pwm_var, в которую передается управляющий код.

Коды управляющего воздействия, формируемые оболочкой при нажатии кнопки «Пуск» вкладки, запоминаются в приемнике pwm_var и используются для управления скважностью ШИМ-сигнала.

График управляющего воздействия, отображаемый в окне «Вход» вкладки, строится пользователем или выбирается из ранее построенных с помощью кнопки «Открыть» вкладки (см. описание вкладки). При выполнении программы используйте оба варианта формирования управляющего воздействия.

Работу ШИМ-генератора контролируйте с помощью осциллографа.

П3.4. Разработайте программу «Электронные часы» с отображением на экране ЖКИ текущего времени с точностью 0,1 с. В качестве счетных импульсов временных «тиков» используйте прерывания таймера, следующие с частотой 2 кГц. Программный счетчик «тиков» должен быть реализован в фоновой циклической программе. При разработке программы используйте описание электронных часов, рассмотренное в п. 2.6.

П3.5. Разработайте и выполните программу «Электронный секундомер», которая определяет интервал времени между внешними

прерываниями int0 и int1, генерируемыми при нажатии двух клавиш разных столбцов клавиатуры стенда. При разработке программы используйте программный модуль «Электронные часы».

Источником прерываний int0 и int1 является перепад на соответствующих входах микроконтроллера. На стенде входы внешних прерываний int0 и int1 соединены с одноименными выводами. Для подключения прерывания int0 и int1 необходимо соединить два выхода клавиатуры со входами указанных прерываний. На вход клавиатуры следует подать постоянный 0. Этот сигнал должен быть сформирован аппаратно или программно.

Программа иллюстрирует взаимодействие трех обработчиков прерываний (одного внутреннего и двух внешних). При поступлении запроса int0 его обработчик осуществляет запуск таймера, обнуляет счетчик времени, запрещает собственные прерывания int0 и разрешает прерывания int1. Обработчик прерывания int1 должен остановить таймер, блокировать собственные прерывания int1 и разрешить прерывания int0. Значение интервала времени между внешними прерываниями int0 и int1 необходимо отображать на экране ЖКИ.

П3.6. Исследуйте работу системы (электронных часов) при наличии нескольких источников прерываний. Дополнительным источником прерывания в программе является флаг RI порта SP.

В электронных часах точность работы критична к стабильности средней частоты аппаратных «тиков», формируемых обработчиком прерывания от таймера, и в общем случае зависит от числа источников прерываний в системе, установленных приоритетов их обслуживания и длины самих обработчиков. В системе с одним источником прерываний от таймера ничто не влияет на среднюю частоту аппаратных «тиков», и часы работают правильно. При наличии нескольких источников прерываний запросы могут поступать асинхронно, в том числе, когда уже обрабатывается один из запросов. Очевидно, что при неправильном выборе приоритетов обслуживания обработчиков работа программы подсчета времени может нарушаться.

Цель исследования – анализ влияния приоритетов различных источников прерываний на точность работы часов.

Модифицируйте программу «Электронные часы», дополнив ее обработчиком прерывания от приемопередатчика последовательного порта. Сам обработчик должен содержать команду однократного сеанса связи с инструментальной ЭВМ (команду lcall 128h). Вариант программы обработчика прерывания последовательного порта приведен на рис.5.3.

```
        org 8023h       ;Обработчик прерывания последовательного порта
        ljmp usart
usart:  jnb ri,skip     ;проверка наличия запроса RI необходима, поскольку обработчик запускается
                        ; как от RI, так и от TI. При RI=0 осуществляется выход из обработчика
        push a          ;поскольку программа монитора использует некоторые ресурсы МК,
        push 0          ;содержимое регистров a, r0 и psw сохраняется в стеке
        push psw
        lcall 128h      ;вызов однократного сеанса связи с инструментальной ЭВМ
        clr ri
        pop psw
        pop 0
        pop a
```

Модифицированная программа обеспечивает совместную работу программы
«Электронные часы» с программой вкладки «Окна управления» оболочки Shell51 (опция «Выход»). Контролируемым параметром в программе является содержимое счетчика долей секунд (адрес счетчика необходимо установить в окне «Адрес выхода» вкладки). Изменение содержимого счетчика долей секунд при работе «часов» отображается в окне «Выход»

Программа работает следующим образом. При нажатии кнопки-панели ПУСК оболочка Shell51 формирует последовательность периодически повторяющихся информационных посылок, которые по последовательному каналу поступают на вход исследуемого МК (стенда). При приеме старт-импульса посылки порт SP микроконтроллера формирует запрос прерывания RI, обработчик которого в сеансе связи с ПК отображает содержимое счетчика долей секунд в окне «Выход» оболочки.

Проконтролируйте работу счетчика долей секунд, используя опцию «Выход» вкладки «Окна управления

Подпрограмму инициализации электронных часов в модифицированной программе необходимо дополнить командой разрешения прерывания от последовательного порта (setb es) и командами задания приоритетов прерываний (*mov A9h,#const* и *mov B9h,#const*, где #const – код устанавливаемых уровней приоритета задействованных прерываний).

С помощью средств среды Shell51 зафиксируйте осциллограммы изменения во времени содержимого счетчика долей секунд для различных комбинаций задаваемых приоритетов источников используемых прерываний.

Определите влияние задаваемых уровней приоритетов прерывания на точность работы электронных часов. Осциллограммы **изменения во времени содержимого счетчика долей секунд** приведите в отчете**.**

П3.7.Разработайте программу простейшей многозадачной операционной системы с разделением времени. Система должна выполнять диспетчеризацию трех циклически выполняемых программ: определение номера нажатой клавиши и преобразование его в ASCII-код (задача 1), отображение на экране ЖКИ номера нажатой клавиши (задача 2) и индикация времени, формируемого программой «Электронные часы» (задача 3).

Пояснения и рекомендации по выполнению задания приведены в приложении 6.

Формат дескрипторов задач, включающий адрес возврата (т.е. адрес команды, с которой возобновляется выполнение задачи при очередном вызове), квант времени выполнения, буфер обмена и т.п., выберите самостоятельно.

Программы задач дополните командой-инвертором cpl P1.n, где n – номер задачи. С помощью этой команды удобно контролировать выполнение задачи многозадачной системой: в процессе работы системы в течение кванта времени, выделяемого для выполнения задачи, на выходе P1.n формируются прямоугольные импульсы с периодом, равным удвоенному времени цикла выполнения задачи. Убедитесь в этом, контролируя сигналы на выходах P1.n с помощью осциллографа.

Определите время выполнения каждой задачи. Осциллограммы работы задач и результаты измерения времени выполнения задач **приведите в отчете.**

П4. Программа четвертого цикла работ: «Организация и исследование межпроцессорного обмена»

Цель работы:

- ознакомление с принципами организации обменов по последовательному каналу;
- приобретение навыков создания коммуникационных протоколов последовательной связи;
- знакомство с организацией межпроцессорных обменов.

Организация и исследование обменов по последовательному каналу связи изучаются на примерах взаимодействия лабораторного стенда с инструментальной ЭВМ и другим стендом. Последовательные порты SP должны иметь одинаковые параметры коммуникационного протокола. Настройка порта SP компьютера выполняется с помощью вкладки «Терминал» оболочки Shell51. Настройка порта SP МК реализуется программно.

«Поле настроек» вкладки «Терминал» позволяет задавать следующие параметры коммуникационного протокола:

- скорость передачи данных, бит/с. Большая скорость обычно обеспечивает меньшую надежность передачи, поскольку возрастают требования к форме импульсов синхронизации приемника и передатчика;

- длину слова информационной посылки, бит. Этот параметр (обычно 8 бит) используется при передачах символов, представленных различными кодами, например, при 4-битной кодировке BCD, 5-битной BAUDOT, 6-битной EBCD или 7-битной ASCII;

- количество стоповых бит (1 или 2). Два стоповых бита облегчают синхронизацию, но замедляют скорость обмена;

- контроль передачи на четность/нечетность. При использовании контроля информационная посылка дополняется битом четности.

В качестве примера приведем задаваемый протокол обмена со следующими параметрами приемопередатчика: скорость передачи данных -

1200 бит/с; длина слова передаваемых данных - 8 бит; количество стоповых бит – 2, контроль передачи на четность.

Программа работы.

Прием-передачу данных по последовательному каналу между ПК и МК можно реализовать, выполнив программу **send_rec.asm**.

Программа send_rec объединяет три подпрограммы: init, receive и send. Алгоритм и текст программы представлены на рис.5.4 и 5.5.

Подпрограмма init обеспечивает настройку последовательного порта SP и таймера T/C1 МК на заданный режим работы. Подпрограмма receive реализует прием последовательных данных в МК, а подпрограмма send - передачу последовательных данных из МК.

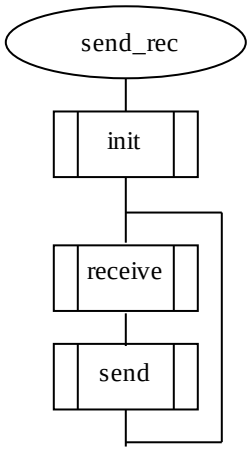


Рис.5.4. Схема алгоритма программы send_rec

```
        ;программа send_rec.asm
            org 8100h

            lcall init
work:       lcall receive
            lcall send
            sjmp work

init:       clr tr1                     ;останов таймера T/C1
   ;задание режима работы SP  ;9-битный асинхронный режим
            mov scon,#11010010b         с переменной скоростью и
                                        ;установленным флагом TI
   ; задание скорости работы SP (программирование таймера T/C1 для
   ; работы в режиме автогенератора на основе 8-разрядного регистра TLx
            anl tmod,#0Fh
            orl tmod,#00100000b
   ; Значения констант перезагрузки T/C1 для произвольных скоростей
   ; обмена вычисляется по формуле (см.п.2.4.2). Для стандартных
   ; скоростей обмена эти значения выбираются из табл.2 (п.2.4.2),
            mov th1,#e6h                ;скорость обмена 1200 бит/с
   ; программирование бита BD в регистре ADCON и бита SMOD
   ; в регистре PCON, влияющих на скорость обмена
            anl D8h,#7Fh                ; сброс бита BD в регистре ADCON
            anl 87h,# 7Fh               ; сброс бита SMOD в регистре PCON
            setb tr1                    ; разрешение работы таймера T/C1
            ret
   ; подпрограмма receive приема последовательных данных в МК
receive:    jnb ri,receive              ;ожидание завершения приема
            mov a,sbuf
            clr ri
            ret

   ; подпрограмма receive передачи последовательных данных из МК
   ; для ;правильной работы передатчика флаг TI при инициализации
   ; должен быть установлен.
send:       jnb ti,send                 ;ожидание завершения передачи
            mov sbuf,a
            clr ti
            ret
```

Рис.5.5. Текст программы send_rec

П4.1. Выполните программу send_rec.asm, реализующую обмен данными между инструментальной ЭВМ (вкладкой «Терминал» оболочки Shell51) и лабораторным стендом.

Перед выполнением данной программы и других программ обмена данными по последовательному каналу с помощью «Поля настроек» вкладки «Терминал» запрограммируйте последовательный порт компьютера для работы по заданному или выбранному протоколу обмена.

Для проверки работы программы в окне «Передача кода» вкладки «Терминал» наберите 16-ричный код однобайтной посылки и нажмите кнопку «Послать код». Формируемый при этом старт-бит инициирует работу приемника порта SP микроконтроллера. По завершению приема информационной посылки принятый код переписывается в регистр SBUF приемника и устанавливается флаг прерывания приемника RI. Подпрограмма receive обрабатывает принятую информацию (в данном случае переписывает данные из SBUF в аккумулятор). Поскольку флаг TI установлен, запускается подпрограмма send, реализующая передачу данных из аккумулятора в SP МК. Последний формирует последовательную посылку и пересылает ее в порт SP ПК. При нажатии кнопки вкладки «Принять однократно» принятый по последовательному каналу код отображается на экране ЭВМ в поле «Принятый код».

Проверьте работу программы при других параметрах коммуникационного протокола обмена.

П4.2. Модифицируйте программу send_rec, дополнив ее командой cpl a, выполняемой после команды lcall receive (перед командой lcall send). Приведите пример выполнения модифицированной программы.

П4.3. Разработайте и выполните программу rec_lcd, реализующую вывод на экран ЖКИ лабораторного стенда текста, передаваемого с инструментальной ЭВМ (вкладки «Терминал» среды Shell51). Вариант схемы соединений МК с инструментальной ЭВМ и блоком ЖКИ показан на рис.5.6.

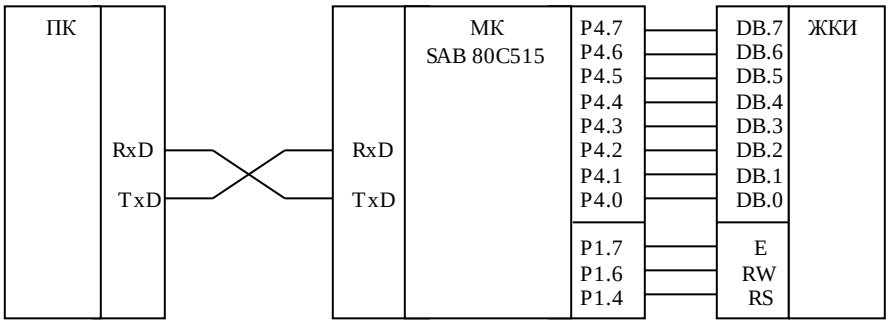


Рис.5.6. Схема информационных связей МК с инструментальной ЭВМ блоком ЖКИ

Передаваемый текст (строка символов) вводится в окно «Передача текстовых сообщений». Ограничений на число передаваемых символов нет. Строка символов должна заканчиваться символом конца посылки, например ‘.’.

При нажатии кнопки «Послать текст» подготовленное сообщение по последовательному каналу должно передаваться в МК для последующего отображения на экране ЖКИ стенда. Данные, поступающие по последовательному каналу в МК целесообразно размещать в видеобуфере блока ЖКИ (внешней памяти МК), При приеме необходимо контролировать символ конца посылки, число отображенных символов и продолжительность приема посылки. Прием информационной посылки завершается при обнаружении символа конца посылки.

По окончанию приема посылки содержимое принятой посылки должно отображаться на экране ЖКИ, начиная с первого знакоместа экрана. Если число символов в посылке ≤ 40, все символы выводятся на экран ЖКИ, при этом незаполненная часть экрана (дополнение до 40) должна отображаться пробелами. При длине принятой посылки более 40 символов из видеобуфера на экран выводятся только первые 40 символов. При завершении вывода на экран ЖКИ управление должно передаваться подпрограмме приема последовательных данных, которая ожидает поступления следующей посылки.

Нарушение протокола обмена, в частности из-за отсутствия символа конца посылки в течение определенного времени, например 1с, необходимо контролировать с помощью таймера T/C0. Таймер должен запускаться при поступлении первого импульса посылки. При нормальной работе сброс таймера (его обнуление и запрет счета) осуществляется при обнаружении символа конца посылки. В случае нарушения протокола обмена на экран ЖКИ необходимо выводить сообщение «Ошибка протокола» и управление должно передаваться подпрограмме приема последовательных данных. С помощью таймера T/C0 в программе реализуются функции сторожевого таймера WDT.

Результаты тестирования программы отразите в отчете.

П4.4. Разработайте и выполните сервисную программу статистической обработки данных rec_static.asm.

На основе программы rec_txt разработайте программу статистической обработки последовательности чисел, поступающих на вход МК с вкладки «Терминал» оболочки Shell51. Программа определяет минимальное, максимальное и среднеарифметическое значение массива чисел. Вариант схемы соединений МК с инструментальной ЭВМ и блоком ЖКИ лабораторного стенда показан на рис.5.6.

Передаваемый текст (код инструкции обработки и числовые данные) вводится в окно «Передача текстовых сообщений» вкладки «Терминал». При нажатии кнопки «Послать текст» подготовленная строка символов по

последовательному каналу передается в МК. Протокол передачи сформируйте самостоятельно.

Формат информационной посылки включает код операции обработки (первый символ посылки), числовые данные и признак конца посылки. Числа представлены ASCII-кодами цифр разрядов числа. При приеме числа его разряды поступают последовательно, начиная со старшего разряда. Код инструкции обработки и числа друг от друга отделяются символами-разделителями, например, «пробелом». Поскольку обработка числовых данных выполняется 8-разрядным МП, значения чисел и количество чисел ограничим величиной 255. «Разделитель» и «конец посылки» являются служебными символами протокола.

МК принимает информационную посылку и в соответствии с кодом инструкции обрабатывает ее. Результат обработки (максимальное, минимальное или среднее арифметическое из чисел) и код инструкции необходимо отобразить на экране ЖКИ стенда.

П4.5. Разработайте и выполните программу обмена информацией между соседними микроконтроллерными стендами.

Соединение двух стендов по последовательному каналу связи осуществляется при нажатии специальной кнопки на задней панели стенда. В нажатом состоянии этой кнопки разрывается связь стенда с инструментальной ЭВМ лабораторного комплекса и реализуется подключение к каналу связи с соседним стендом.

Для иллюстрации межконтроллерного взаимодействия результат выполнения программы на одном из стендов отображается на табло ЖКИ собственного стенда и передается по последовательному каналу для отображения на табло ЖКИ соседнего стенда. На каждом из стендов выполняется циклическая (фоновая) программа определения номера нажатой клавиши. При нажатии клавиши ее номер отображается в верхней строке табло ЖКИ этого стенда и поступает в буфер передатчика последовательного порта. Передатчик пересылает информацию о номере нажатой клавиши на соседний стенд. Приемник этого стенда принимает посылку, при необходимости преобразует принятый код в ASCII-код и отображает номер нажатой клавиши другого стенда в нижней строке табло ЖКИ. Для сокращения объема пересылок между стендами передатчики стендов должны передавать только обновленную информацию.

Перед выполнением задания необходимо разработать структуру информационных связей обоих стендов (МК с подключаемым модулем ЖКИ). Далее выполняется трансляция, загрузка и запуск программы определения номера нажатой клавиши на обоих стендах. После этого при фиксации специальных кнопок-переключателей на задней панели реализуется соединение стендов по последовательному каналу связи.

П4.6. Модифицируйте программу предыдущего задания для реализации межконтроллерного обмена информацией по последовательному каналу связи в режиме «Master-Slave».

Режим «Master-Slave» используется для организации взаимодействия ведущего МК с ведомыми, которые соединены с ведущим радиальными каналами связи. Для реализации данного режима необходимо задействовать специально предназначенные для этого ресурсы МК, а именно биты SM2, RB8 и TB8 регистра SCON.

Структура информационных связей обоих стендов, выполняемые программы, трансляция, загрузка и запуск программ определения номера нажатой клавиши на обоих стендах соответствуют заданию предыдущей программы.

Программа, иллюстрирующая межконтроллерный обмен, предполагает использование нескольких кнопок, имеющих специальное назначение.

Будем считать, что кнопка 15 предназначена для перевода МК в режим «Master». При ее нажатии МК присваивается статус «Master» с отображением символа «M» на табло ЖКИ, при этом код кнопки передается в канал связи. Ведомые МК (в данном случае МК соседнего стенда) принимают посылку и при декодировании кода 15 переводятся в режим «Slave»: в них передача от «Slave» к «Master» запрещается и в регистре SCON устанавливается бит SM2 (SM2=1).

В режиме «Master-Slave» изменяется протокол обмена: ведущий МК сначала передает адресную информацию, после чего разрешается передача данных. Код адресной информации имеет установленный бит TB8 (TB8=1), а код данных – сброшенный бит TB8. При SM2=1 ведомый МК может принять только адресную информацию, т.е. посылку с установленным битом RB8.

Кнопки 13 и 14 кодируют адреса ведомых МК, при этом код кнопки 14 соответствует адресу соседнего стенда. При нажатии кнопки 14 (или 13) на стенде ведущего МК последний формирует и отправляет адресную посылку с установленным битом TB8, на экране ЖКИ стенда высвечивается информация «addrN» (N=14 или 13). Завершающим этапом посылки адреса является запрет установки бита TB8 в последующих посылках, которые являются посылками данных.

Все ведомые МК, имеющие в исходном состоянии установленный бит SM2, получают адресную посылку, поскольку в ней бит RB8=1, и анализируют ее. Если адрес 14, ведомый МК идентифицирует себя как приемник. Он перепрограммирует свой последовательный порт (сбрасывает бит SM2, подготавливаясь к приему данных) и отображает на экране ЖКИ номер «addr14». Если адрес не равен 14, ведомый МК указанные действия не выполняет.

После отправки адреса последующие посылки, принимаемые ведомым МК, являются посылками данных.

Кнопка 12 кодирует команду «конец посылки данных». При декодировании кода 12 оба МК переводятся в режим межконтроллерного взаимодействия (см.предыдущий пункт программы): в обоих МК

сбрасывается бит SM2 и в любом МК разрешены прием/передача данных. В этом режиме при нажатии кнопки 15 любой из МК может быть может быть переведен в режим «Master».

Кнопки 0-11 – это обычные информационные кнопки. При их нажатии на передающей стороне осуществляется процедура определения номера нажатой клавиши и отображения этого номера на экране ЖКИ стенда. Код номера нажатой клавиши передается ведомому. Значение бита TB8 безразлично. На приемной стороне (при SM2=0) данные принимаются и отображаются, поскольку при SM2=0 прием не зависит от значения бита RB8 в посылке.

Выполните программу обмена в режиме «Master-Slave», задавая адрес ведомого МК кнопками 13 и 14. Результаты выполнения отразите в отчете.

П5. Разработка программ индивидуальных заданий

По заданию преподавателя **разработайте и выполните программу индивидуального задания.** Варианты индивидуальных заданий представлены ниже.

1. Статистическая обработка сигналов.

По заданию преподавателя реализуется один из способов ввода числовых данных для обработки:

- ввод числовых данных с использованием вкладки «Терминал». Числовые данные вводятся в окно «Передача текстовых сообщений». При нажатии кнопки «Послать текст» подготовленная строка символов в ASCII-кодах по последовательному каналу передается в МК. Числа отделяются друг от друга «пробелами». Последним символом строки является «точка» Количество и значения вводимых чисел не превышает 255. При приеме ASCII-коды чисел преобразуются в двоичный код. Двоичные коды чисел запоминаются во внешней памяти МК.
- использование АЦП для формирования цифровых эквивалентов входного аналогового сигнала. На вход АЦП поступает аналоговый сигнал с выхода потенциометра, изменяемый случайным образом. Двоичные коды на выходе АЦП запоминаются во внешней памяти МК. Количество преобразованных значений 50-70.
- числовые данные формируются генератором псевдослучайных чисел (см. приложение 6). Способ реализации генератора задает преподаватель.

В результате обработки массива числовых данных определяется минимальное, максимальное и среднеарифметическое значение массива, а также выполняются вычисления, необходимые для построения гистограммы распределения чисел. Результаты статистической обработки отображаются на экране ЖКИ. Гистограмма распределения должна отображаться в окне «Выход» вкладки «Окна управления» оболочки Shell51.

2. Калькулятор.

Программа должна реализовать арифметико-логическую обработку цифровых данных, задаваемых с клавиатуры стенда, отображение на ЖКИ значений входных данных и результатов вычислений.

Формат представления чисел - 16-ричные со знаком с фиксированной точкой. Дополнительные функции: выполняемые операции – сложение, вычитание, умножение, извлечение квадратного корня, взаимное преобразование кодов в 2-, 8-, 10-, 16-ричной системах исчисления, округление результатов, вычисление логических функций.

3. Электронная записная книжка.

Программа должна поддерживать следующие режимы:

- часы-календарь с установкой даты и времени;
- ввод строки телефонной базы данных в формате «телефон – текстовая строка». Строки вводятся с использованием вкладки «Терминал» оболочки Shell51;
- ввод новых данных, поиск, просмотр и удаление информационных строк базы данных. Содержимого строк отображается блоком ЖКИ стенда;
- формирование специального сигнала, например, мигания строки ЖКИ при наступлении программируемого момента времени.

4. Электронный экзаменатор.

Программа предлагает пользователю ответить на ряд вопросов с вариантами ответа, отображаемыми на ЖКИ. Проводится подсчет времени каждого ответа, количество правильных и неправильных ответов. Результаты теста отображаются на экране ЖКИ.

Дополнительные рекомендации по программе «Электронный экзаменатор» приведены в приложении 6.

5. Игровой тренажер внимания.

Программа реализует следующий алгоритм работы:

По экрану ЖКИ слева направо перемещаются символы, соответствующие определенным клавишам. Задача обучаемого – отреагировать на очередной символ нажатием соответствующей ему клавиши. Правильные и неправильные действия фиксируются в специальных счетчиках. Запуск и останов программы осуществляются при нажатии выбранных пользователем одноименных клавиш. В процессе выполнения программы темп перемещения предъявляемых символов ускоряется. Статистка выполнения программы должна отображаться на экране ЖКИ после завершения программы. Один из алгоритмов работы тренажера рассмотрен в приложении П6.

6. Частотомер

Реализовать на базе лабораторного стенда частотомер, который обеспечивает измерение частоты входного периодического сигнала на входе

АЦП. Выбор типа входного сигнала: гармонический или прямоугольный, обеспечивается с помощью клавиатурного блока платы МК

Предусмотреть возможность измерения высокочастотного и низкочастотного сигналов наиболее точными методами.

Режимы работы и результат измерения частоты необходимо отображать на экране модуля ЖКИ.

7. Многофункциональный генератор (см. приложение 6).

Формирует в зависимости от комбинации нажатых клавиш 5 видов периодических сигналов: синусоидальный, прямоугольный, треугольный, пилообразный, колоколообразный, трапециидальный; для любого сигнала возможно формирование нескольких (не менее 5) значений периода. Выход генератора – ШИМ-сигнал, моделирующий заданный периодический сигнал.

8. Модель рекламно-информационного табло.

Программа моделирует работу рекламного или информационного табло с несколькими режимами смены информации, а также режим временного выключения («спящий режим»).

Необходимо предусмотреть:

- смену кадра, т.е. планируемой циклической замены выводимого на табло сообщения по истечении выбранного интервала времени;
- перемещение или перестановку строк сообщения (смена информации «по вертикали»);
- чередующийся со сменой информации «по вертикали» режим «бегущих строк» влево и/или вправо с выбираемой с клавиатуры скоростью (смена информации «по горизонтали»);
- перевод табло в «спящий режим», моделирующий ночное выключение.

Приложение 2

Система команд микроконтроллеров семейства MCS-51

Условные обозначения:

Rn – регистры R0 – R7 текущего (рабочего) банка регистров;

bit – прямо адресуемый бит, находящийся в RRAM или регистре специальных функций;

#data или #d – 8-битная константа, входящая в состав команды;

#data16 или #d16 – 16-битная константа, входящая в состав команды;

direct – 8-битный адрес младших 128 байт RRAM или адрес регистра блока SFR;

@Ri – обозначение косвенной адресации через регистр Ri (i = 0, 1);

@DPTR – обозначение косвенной адресации через регистр DPTR;

addr16 – 16-битный адрес безусловной передачи управления;

addr11 – 11-битный адрес безусловной передачи (только в командах передачи управления);

rel – относительный адрес (8-битное значение со знаком, суммируемое с содержимым PC для формирования адреса перехода), в командах передачи управления (условного и короткого безусловного переходов);

/bit – слэш перед операндом указывает, что используется инвертированное значение бита-источника.

Группа команд пересылок

Команды пересылок с мнемоникой MOV

Мнемокод	КОП	Байты	Циклы	Функция
MOV A, Rn	1110 1nnn	1	1	(A) ← (Rn)
MOV A, direct	1110 0101	2	1	(A) ← (direct)
MOV A, @Ri	1110 011i	1	1	(A) ← ((Ri))
MOV A, #data	1110 0100	2	1	(A) ← #data
MOV Rn, A	1111 1nnn	1	1	(Rn) ← (A)
MOV Rn, direct	1010 1nnn	2	2	(Rn) ← (direct)
MOV Rn, #data	0111 1nnn	2	1	(Rn) ← #data
MOV direct, A	1111 0101	2	1	(direct) ← (A)
MOV direct, Rn	1000 1nnn	2	2	(direct) ← (Rn)
MOV direct, direct	1000 0101	3	2	(direct) ← (direct)
MOV direct, @Ri	1000 011i	2	2	(direct) ← ((Ri))
MOV direct, #data	0111 0101	3	2	(direct) ← #data
MOV @Ri, A	1111 011i	1	1	((Ri)) ← (A)
MOV @Ri, direct	1010 011i	2	1	((Ri)) ← (direct)
MOV @Ri, #data	0111 011i	2	1	((Ri)) ← #data

Команда загрузки 16-битных данных

MOV DPTR, #data16	1001 0000	3	2	((DPTR)) ← #data16

Команды пересылок в/из внешней памяти данных

MOVX A, @Ri	1110 001i	1	2	(A) ← ((Ri))
MOVX @Ri, A	1111 001i	1	2	((Ri)) ← (A)
MOVX A, @DPTR	1110 0000	1	2	(A) ← ((DPTR))
MOVX @DPTR, A	1111 0000	1	2	((DPTR)) ← (A)

Команды пересылок из внешней программной памяти

MOVC A, @A+DPTR	1001	0011	1	2	(A) ← ((A)+(DPTR))
MOVC A, @A+PC	1000	0011	1	2	(A) ← ((A)+(PC))

Команды загрузки/извлечения из стека

PUSH direct	1100	0000	2	2	(SP) ← (SP)+1 ((SP)) ← (direct)
POP direct	1101	0000	2	2	(direct) ← ((SP)) (SP) ← (SP)-1

Команды обмена

XCH A, Rn	1100	1nnn	1	1	(A) ↔ (Rn)
XCH A, direct	1100	0101	2	1	(A) ↔ (direct)
XCH A, @Ri	1100	011i	1	1	(A) ↔ ((Ri))

Команды обмена тетрадами

XCHD A, @Ri	1101	011i	1	1	$(A_{3-0}) \leftrightarrow ((Ri)_{3-0})$
SWAP A	1100	0100	1	1	Обмен тетрад аккум-ра

Команды арифметических операций

Мнемокод	КОП		Байты	Циклы	Функция
ADD A, Rn	0010	1nnn	1	1	(A) ← (A) + (Rn)
ADD A, direct	0010	0101	2	1	(A) ← (A) + (direct)
ADD A, @Ri	0010	011i	1	1	(A) ← (A) + ((Ri))
ADD A, #data	0010	0100	2	1	(A) ← (A) + #data
ADDC A, Rn	0011	1nnn	1	1	(A) ← (A) + (C) + (Rn)
ADDC A, direct	0011	0101	2	1	(A) ← (A) + (C) + (direct)
ADDC A, @Ri	0011	011i	1	1	(A) ← (A) + (C) + ((Ri))
ADDC A, #data	0011	0100	2	1	(A) ← (A) + (C) + #data
SUBB A, Rn	1001	1nnn	1	1	(A) ← (A) – (C) – (Rn)
SUBB A, direct	1001	0101	2	1	(A) ← (A) – (C) – (direct)
SUBB A, @Ri	1001	011i	1	1	(A) ← (A) – (C) – ((Ri))
SUBB A, #data	1001	0100	2	1	(A) ← (A) – (C) – #data
DA A	1101	0100	1	1	Десятичная коррекция содержимого аккумулятора
INC A	0000	0100	1	1	(A) ← (A) + 1
INC Rn	0000	1nnn	1	1	(Rn) ← (Rn) + 1
INC direct	0000	0101	2	1	(direct) ← (direct) + 1
INC @Ri	0000	011i	1	1	((Ri)) ← ((Ri)) + 1
INC DPTR	1010	0011	1	2	(DPTR) ← (DPTR) + 1
DEC A	0001	0100	1	1	(A) ← (A) – 1
DEC Rn	0001	1nnn	1	1	(Rn) ← (Rn) – 1
DEC direct	0001	0101	2	1	(direct) ← (direct) – 1
DEC @Ri	0001	011i	1	1	((Ri)) ← ((Ri)) – 1
MUL AB	1010	0100	1	4	$(B)_{15-8}, (A)_{7-0} \leftarrow (A)*(B)$
DIV AB	1000	0100	1	4	$(A)_{цел}, (B)_{ост} \leftarrow (A)/(B)$

Команды логических операций

Мнемокод	КОП	Байты	Циклы	Функция
ANL A, Rn	0101 1nnn	1	1	(A) ← (A) ∧ (Rn)
ANL A, direct	0101 0101	2	1	(A) ← (A) ∧ (direct)
ANL A, #data	0101 0100	2	1	(A) ← (A) ∧ #data
ANL A, @Ri	0101 011i	1	1	(A) ← (A) ∧ ((Ri))
ANL direct, A	0101 0010	2	1	(direct) ← (direct) ∧ (A)
ANL direct, #data	0101 0011	3	2	(direct) ← (direct) ∧ #data
ORL A, Rn	0100 1nnn	1	1	(A) ← (A) ∨ (Rn)
ORL A, direct	0100 0101	2	1	(A) ← (A) ∨ (direct)
ORL A, @Ri	0100 011i	1	1	(A) ← (A) ∨ ((Ri))
ORL A, #data	0100 0100	2	1	(A) ← (A) ∨ #data
ORL direct, A	0100 0010	2	1	(direct) ← (direct) ∨ (A)
ORL direct, #data	0100 0011	3	2	(direct) ← (direct) ∨ #data
XRL A, Rn	0110 1nnn	1	1	(A) ← (A) ⊕ (Rn)
XRL A, direct	0110 0101	2	1	(A) ← (A) ⊕ (direct)
XRL A, @Ri	0110 011i	1	1	(A) ← (A) ⊕ ((Ri))
XRL A, #data	0110 0100	2	1	(A) ← (A) ⊕ #data
XRL direct, A	0110 0010	2	1	(direct) ← (direct) ⊕ (A)
XRL direct, #data	0110 0011	3	2	(direct) ← (direct) ⊕ #data
CLR A	1110 0100	1	1	Обнуление аккум-ра (A) ← 0
CPL A	1111 0100	1	1	Инверсия аккум-ра (A) ← $(\overline{A})$

Команды сдвиговых операций

RL A	0010	0011	1	1	Сдвиг ACC циклический влево
RLC A	0011	0011	1	1	Сдвиг ACC влево через перенос
RR A	0000	0011	1	1	Сдвиг ACC циклический вправо
RRC A	0001	0011	1	1	Сдвиг ACC вправо через перенос
SWAP A	1100	0100	1	1	Циклический сдвиг ACC на 4 разряда $(A_{3-0}) \leftrightarrow (A_{7-4})$

Команды, влияющие на установку флагов

Команда	Флаг			Команда	Флаг		
	C	OV	AC		C	OV	AC
ADD	+	+	+	ANL C, bit	+	–	–
ADDC	+	+	+	ANL C, /bit	+	–	–
SUBB	+	+	+	ORL C, bit	+	–	–
MUL	0	+	–	ORL C, bit	+	–	–
DIV	0	+	–	MOV C, bit	+	–	–
DA	+	–	–	CLR C	0	–	–
RRC	+	–	–	CPL C	+	–	–
RLC	+	–	–	CJNE	+	–	–
SETB C	1	–	–	POP PSW	+	+	+

Команды битового процессора

Команды пересылки бита

Мнемокод	КОП	Байты	Циклы	Функции
MOV C, bit	1010 0010	2	1	(C) ← bit
MOV bit, C	1001 0010	2	2	(bit) ← (C)

Команды установки и сброса бит

SETB C	1101 0011	1	1	(C) ← 1
SETB bit	1101 0010	2	1	(bit) ← 1
CLR C	1100 0011	1	1	(C) ← 0
CLR bit	1100 0010	2	1	(bit) ← 0

Команды инверсии бит

CPL C	1011	0011	1	1	$(C) \leftarrow (\overline{C})$
CPL bit	1011	0010	2	1	$(bit) \leftarrow (\overline{bit})$

Команды логических операций для битовых переменных

ANL C, bit	1000	0010	2	2	$(C) \leftarrow (C) \wedge (bit)$
ANL C, /bit	1011	0000	2	2	$(C) \leftarrow (C) \wedge (\overline{bit})$
ORL C, bit	0111	0000	2	2	$(C) \leftarrow (C) \vee (bit)$
ORL C, /bit	1010	0000	2	2	$(C) \leftarrow (C) \vee (\overline{bit})$

Команды условных переходов битового процессора

JB bit, rel	0010	0000	3	2	(PC) ← (PC) + 3 if (bit) = 1 then (PC) ← (PC) + rel
JBC bit, rel	0001	0000	3	2	(PC) ← (PC) + 3 if (bit) = 1 then (bit) ← 0 и (PC) ← (PC) + rel
JNB bit, rel	0011	0000	3	2	(PC) ← (PC) + 3 if (bit) = 0 then (PC) ← (PC) + rel
JC rel	0100	0000	2	2	(PC) ← (PC) + 2 if (C) = 1 then (PC) ← (PC) + rel
JNC rel	0101	0000	2	2	(PC) ← (PC) + 2 if (C) = 0 then (PC) ← (PC) + rel

Команды передачи управления

Команды безусловной передачи управления

Мнемокод	КОП	Байты	Цикл	Функции
LJMP addr16	0000 0010	3	2	(PC) ← $addr_{15-0}$
AJMP addr11	a_{10}-$a_8$0 0001	2	2	(PC) ← (PC) + 2, $(PC)_{10-0}$ ← $addr_{10-0}$
SJMP rel	1000 0000	2	2	(PC) ← (PC) + 2, (PC) ← (PC) + rel
JMP @A + DPTR	0111 0011	1	2	Косвенный переход (PC) ← (A) + (DPTR)
LCALL addr16	0001 0010	3	2	(PC) ← (PC) + 3 (SP) ← (SP) + 1 ((SP)) ← (PC_{7-0}) (SP) ← (SP) + 1 ((SP)) ← (PC_{15-8}) (PC) ← $addr_{15-0}$
ACALL addr11	a_{10}-$a_8$1 0001	2	2	(PC) ← (PC) + 2 (SP) ← (SP) + 1 ((SP)) ← (PC_{7-0}) (SP) ← (SP) + 1 ((SP)) ← (PC_{15-8}) (PC_{10-0}) ← $addr_{10-0}$
RET	0010 0010	1	2	(PC_{15-8}) ← ((SP)) (SP) ← (SP) − 1 (PC_{7-0}) ← ((SP)) (SP) ← (SP) − 1
RETI	0011 0010	1	2	(PC_{15-8}) ← ((SP)) (SP) ← (SP) − 1 (PC_{7-0}) ← ((SP)) (SP) ← (SP) − 1

Команды условных переходов

Мнемокод	КОП	Байты	Циклы	Функции
JZ rel	0110 0000	2	2	(PC) ← (PC) + 2 if (A) = 0 then (PC) ← (PC) + rel
JNZ rel	0111 0000	2	2	(PC) ← (PC) + 2 if (A) ≠ 0 then (PC) ← (PC) + rel
JC rel	0100 0000	2	2	(PC) ← (PC) + 2 if (C) = 1 then (PC) ← (PC) + rel
JNC rel	0101 0000	2	2	(PC) ← (PC) + 2 if (C) = 0 then (PC) ← (PC) + rel
DJNZ Rn, rel	1101 1nnn	2	2	(PC) ← (PC) + 2, (Rn) ← (Rn) − 1 if (Rn) ≠ 0 then (PC) ← (PC) + rel
DJNZ direct, rel	1101 0101	2	2	(PC) ← (PC) + 2, (direct) ← (direct) −1 if (direct) ≠ 0 then (PC) ← (PC) + rel
CJNE A, direct, rel	1011 0101	3	2	(PC) ← (PC) + 3 if (A) ≠ (direct) then (PC) ←(PC)+ rel при этом if (A) < (direct) then (C) ← 1 else (C) ← 0
CJNE A, #data, rel	1011 0100	3	2	(PC) ← (PC) + 3 if (A) ≠ #data then (PC) ← (PC) + rel при этом if (A) < #data then (C) ← 1 else (C) ← 0
CJNE Rn, #data, rel	1011 1nnn	3	2	(PC) ← (PC) + 3 If (Rn) ≠ #data then (PC) ← (PC) + rel при этом if (Rn) < #data then (C) ← 1 else (C) ← 0
CJNE @Ri, #data, rel	1011 011i	3	2	(PC) ← (PC) + 3 If ((Ri)) ≠ #data then (PC)←(PC) + rel при этом if ((Ri)) < #data then (C) ← 1 else (C) ← 0

Приложение 3

Регистры специальных функций

Таблица

№ п/п	Обозначение Имя регистра SFR	Имена управляющих бит								Адрес Регистра
		7	6	5	4	3	2	1	0	
1*	P0 Порт 0	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	80h
2	SP (Указатель стека)									81h
3	DPL (Младший байт указателя данных)									82h
4	DPH (Старший байт указателя данных)									83h
5	PCON (Регистр управления питанием)	SMOD	PDS	IDLS	-	GF1	GF0	PDE	IDLE	87h
6*	TCON (Регистр управления Таймерами T/C0 и T/C1	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	88h
7	TMOD (Регистр режимов таймеров)	GATE1	C/T1	M1	M0	GATE0	C/T0	M1	M0	89h
8	TL0 (Младший байт Регистра таймера 0)									8Ah
9	TL1 (Младший байт Регистра таймера 1)									8Bh
10	TH0 (Старший байт регистра таймера 0)									8Ch
11	TH1 (Старший байт регистр таймера 1)									8Dh
12*	P1 Порт 1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	90h
		T2	CLKout	T2EX	INT2	INT6	INT5	INT4	INT3	
						CC3	CC2	CC1	CC0	
13*	SCON (Регистр управления) последовательным портом	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	98h
14	SBUF (Буфер последовательного порта)									99h
15*	P2 Порт 2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	A0h
16*	IEN0 Регистр разрешения Прерываний 0	EAL	WDT	ET2	ES	ET1	EX1	ET0	EX0	A8h
17	IP0 Регистр приоритетов 0	-	WDTS	IP0.5	IP0.4	IP0.3	IP0.2	IP0.1	IP0.0	A9h
18*	P3	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	B0h
	Порт 3	RD	WR	T1	T0	INT1	INT0	TxD	RxD	
19*	IEN1 Регистр разрешения прерываний 1	EXEN2	SWDT	EX6	EX5	EX4	EX3	EX2	EADC	B8h
20	IP1 Регистр приоритетов 1	-	-	IP1.5	IP1.4	IP1.3	IP1.2	IP1.1	IP1.0	B9h
21*	IRCON Регистр управления флагами прерываний	EXF2	TF2	IEX6	IEX5	IEX4	IEX3	IEX2	IADC	C0h
22	CCEN Регистр разрешения захвата/привязки)	CC3h	CC3l	CC2h	CC2l	CC1h	CC1l	CC0h	CC0l	C1h
23	CCL1 (мл. байт) Регистр захвата/сравнения 1									C2h
24	CCH1 (ст. байт) Регистр захвата/сравнения 1									C3h

Регистры специальных функций Продолжение таблицы

№ п/п	Обозначение Имя регистра SFR	Имена управляющих бит								Адрес Регистра
		7	6	5	4	3	2	1	0	
25	CCL2 Регистр захвата/сравнения 2 младший байт									C4h
26	CCH2 Регистр захвата/сравнения 2 старший байт									C5h
27	CCL3 Регистр захвата/сравнения 3 младший байт									C6h
28	CCH3 Регистр захвата/сравнения 3 старший байт									C7h
29*	T2CON Регистр управления таймером 2	T2PS	I3FR	I2FR	T2R1	T2R0	T2CM	T2I1	T2I0	C8h
30	CRCL Регистр загрузки, захвата/сравнения Младший байт									CAh
31	CRCH Регистр загрузки, захвата/сравнения старший байт									CBh
32	TL2 Регистр таймера 2 младший байт									CCh
33	TH2 Регистр таймера 2 старший байт									CDh
34*	PSW Слово состояния программы	C	C'	F0	RS1	RS0	OV	F1	P	D0h
35*	ADCON Регистр управления АЦП	BD	CLK	-	BSY	ADM	MX2	MX1	MX0	D8h
36	ADDAT Регистр данных АЦП									D9h
37	DAPR Регистр программирования источника внутренних опорных напряжений									DAh
38*	ACC Аккумулятор	ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	ACC.0	E0h
39*	P4 Порт 4	P4.7	P4.6	P4.5	P4.4	P4.3	P4.2	P4.1	P4.0	E8h
40*	B Регистр B	B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0	F0h
41*	P5 Порт 5	P5.7	P5.6	P5.5	P5.4	P5.3	P5.2	P5.1	P5.0	F8h
42	P6 Порт 6 только в версии ACMOS									DBh

* Регистры, к которым возможно побитовое обращение.

Приложение 4

ОПИСАНИЕ СРЕДЫ ПРОЕКТИРОВАНИЯ SHELL51

Система Shell51. Общие сведения

Одним из направлений применения среды проектирования Shell51 является ее использование для практического ознакомления со структурной организацией МК семейства MCS-51, обучения проектированию прикладного ПО и приобретения опыта построения микропроцессорных систем на основе МК этого семейства.

Система Shell51 является программной составляющей программно-аппаратного комплекса поддержки проектирования прикладного ПО микропроцессорных систем. Комплекс включает инструментальную ЭВМ, лабораторный стенд, реализованный в виде микропроцессорной системы с МК SAB80C535, и собственно систему проектирования ПО Shell51.

Начало работы с системой. Основные требования

Вызов оболочки (вход в среду) Shell51 осуществляется при запуске пакета программ Shell51 (двойным щелчком левой кнопки манипулятора «мышь» на пиктограмме с названием «Система Shell51»), при этом на экране компьютера отображается основное рабочее поле среды (рис. П4.1).

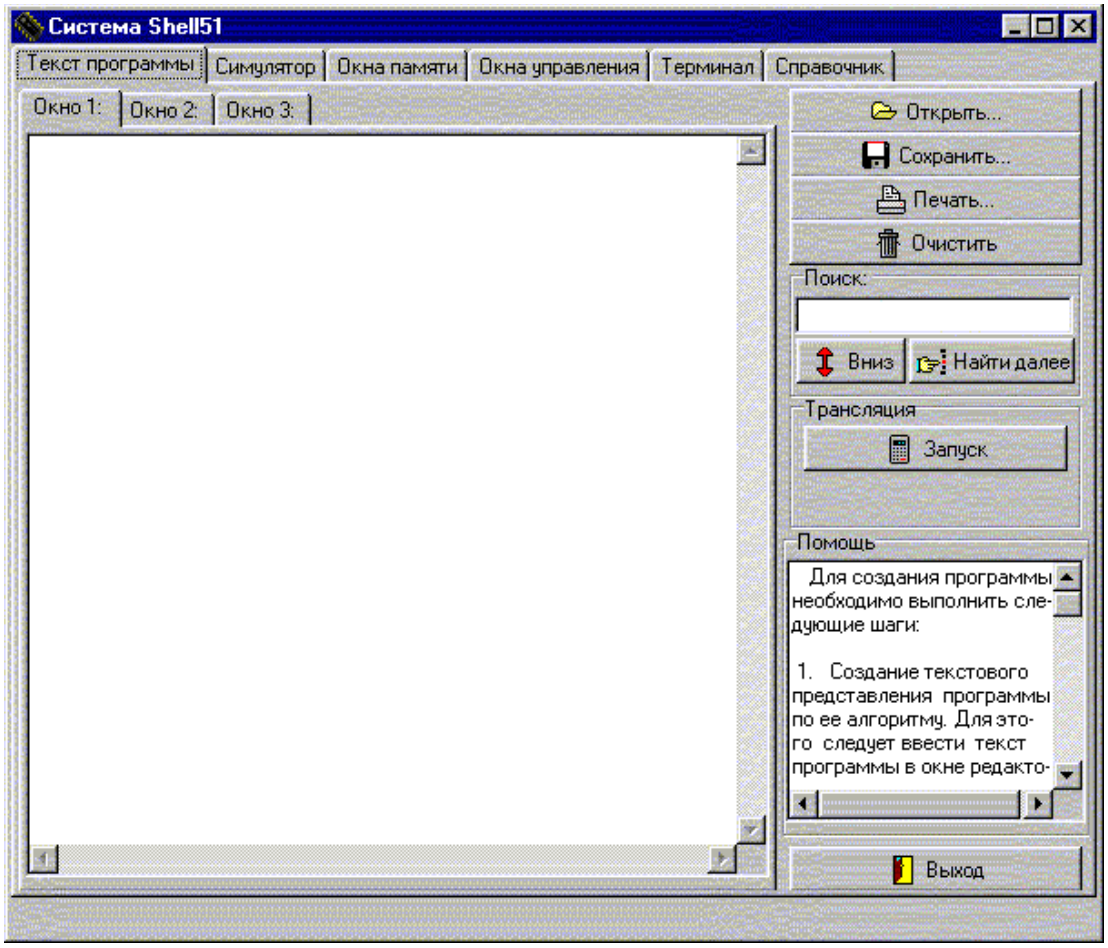


Рис. П4.1. Окно среды Shell51

Поле среды представляет собой набор вкладок-опций, каждая из которых ориентирована на реализацию одного из типовых действий программиста при проектировании ПО микроконтроллерных систем. Программы оболочки Shell51 исполняются на инструментальной ЭВМ в среде Windows.Перед началом работы в среде Shell51 пользователь, используя вкладку «Текст программы», должен создать свой подкаталог в каталоге с номером его группы C:\Shell51\43501/x\Name. Всю работу в среде следует производить только с программными файлами собственного каталога. В этот каталог помещаются файлы с тестовыми программами и разрабатываемыми в процессе работы пользовательскими программами. Модифицировать, копировать, удалять любые файлы в каталоге C:\Shell51 и его подкаталогах, за исключением файлов в собственном каталоге, запрещено.

Лабораторный стенд подключают к сети 220 В с помощью сетевого адаптера при нажатии левой кнопки на задней панели стенда.

По последовательному каналу связи стенд соединяется либо с ПК, либо с соседним стендом. Управление последовательным каналом осуществляется с помощью кнопки (переключателя) - второй слева на задней панели стенда. При ненажатой кнопке МК по последовательному каналу соединяется с ПК. При нажатии кнопки реализуется соединение двух соседних стендов i-го и i+1-го (i – номер рабочего места).

Инициализация подпрограммы связи ПК со стендом осуществляется при нажатии кнопки «Сброс», размещенной на лицевой панели стенда под кнопками клавиатуры.

Для поддержания корректной связи ПК со стендом (МК) по завершению пользовательских программ последние должны завершаться оператором ret, который возвращает управление программе «Монитор» оболочки Shell51.

Вкладка «Справочник»

Вкладка «Справочник» дает общее представление о возможностях системы Shell51 и содержит сведения, необходимые для изучения аппаратуры и программирования микроконтроллерных систем. Вкладка организована в виде гипертекстового документа (рис. П3.2). Выделенные цветом и подчеркиванием элементы документа являются ссылками на соответствующие разделы пособия, переход на которые осуществляется щелчком левой кнопки манипулятора «мышь». Разделы в свою очередь состоят из подразделов с аналогичным принципом переходов. Для возврата на предыдущий уровень используются ссылки «Назад» или «Возврат», присутствующие во всех документах.

Вкладка «Текст программы»

Вкладка «Текст программы» позволяет работать с исходным текстом программы на языке ассемблера. Окно вкладки совпадает с окном среды Shell51 (см. рис. П4.1). Вкладка содержит три независимых окна текста (слева), поле справки по использованию вкладки (справа внизу), набор кнопок-панелей, реализующих функции вкладки, а также кнопку-панель «Выход».

Активизация функции, назначенной кнопке-панели, производится путем нажатия левой кнопки манипулятора «мышь» над изображением этой панели.

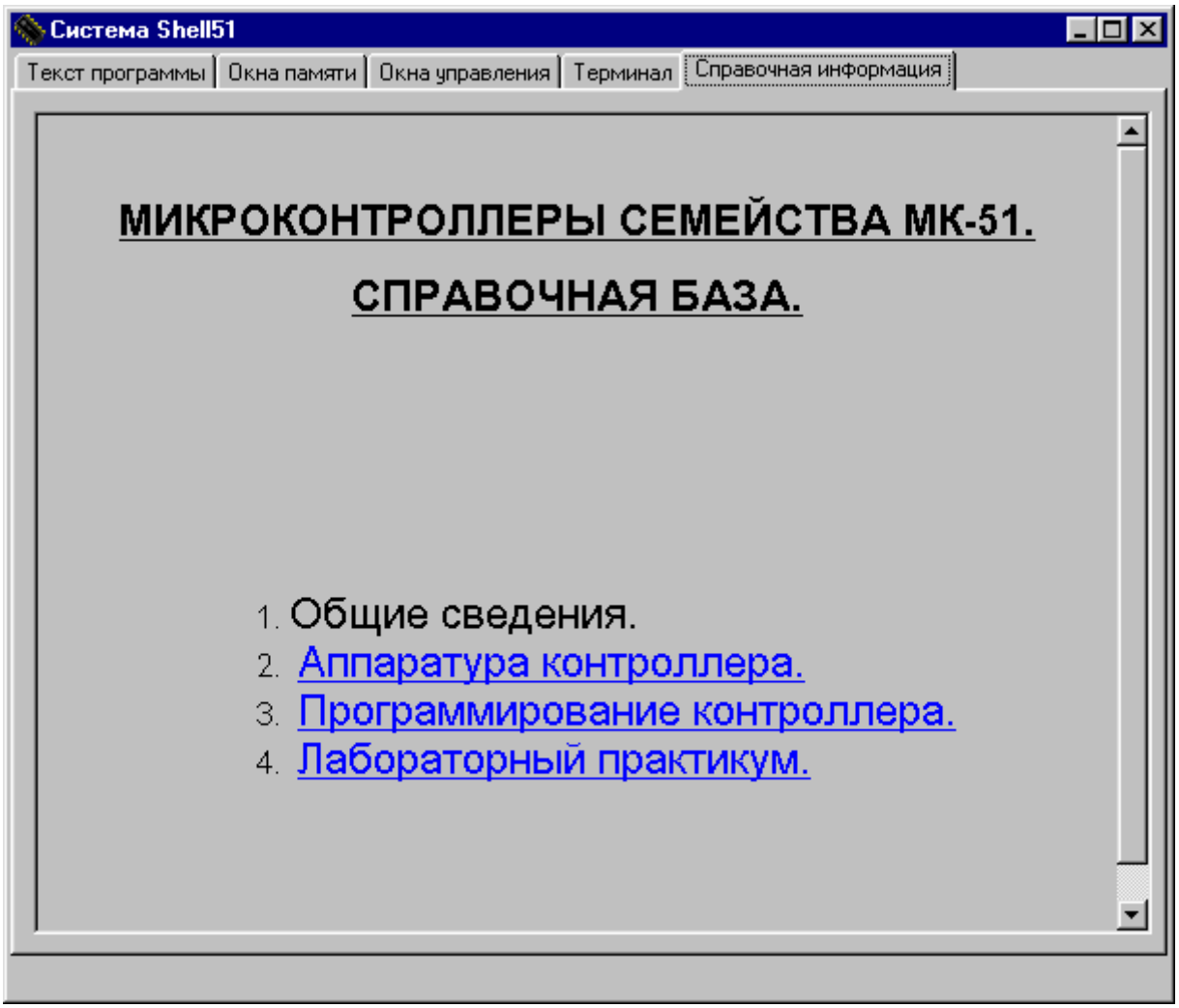


Рис. П4.2. Вкладка «Справочная информация»

Кнопка-панель «Открыть» активизирует диалог выбора необходимого пользователю файла с текстом программы, сохраненного на дисковых накопителях инструментальной ЭВМ (рис. П4.3).

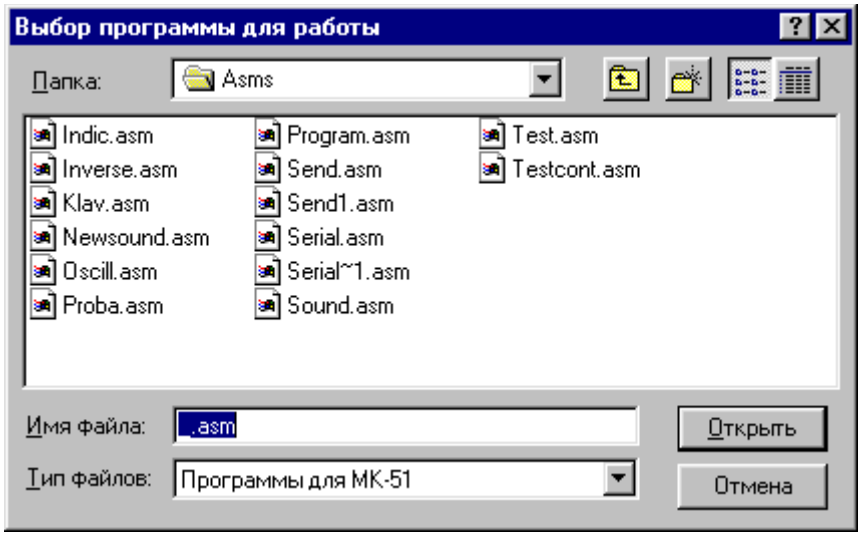


Рис. П4.3. Диалог открытия файла

Для открытия требуемого файла следует выделить его (нажатием левой кнопки манипулятора) и активизировать **кнопку-панель «Открыть»** в правой части диалогового окна (рис. П4.3). В результате текст программы выбранного файла отображается в активном окне и обеспечивается возможность его просмотра и коррекции.

Для создания новой программы достаточно ввести ее текст в поле активного окна. Удаление текста из активного окна осуществляется с помощью **кнопки-панели «Очистить»** вкладки «Текст программы».

Кнопка-панель «Сохранить» вкладки «Текст программы» активизирует диалог записи текста активного окна на диск инструментальной ЭВМ с возможностью модификации имени файла сохраняемой программы (рис. П4.4).

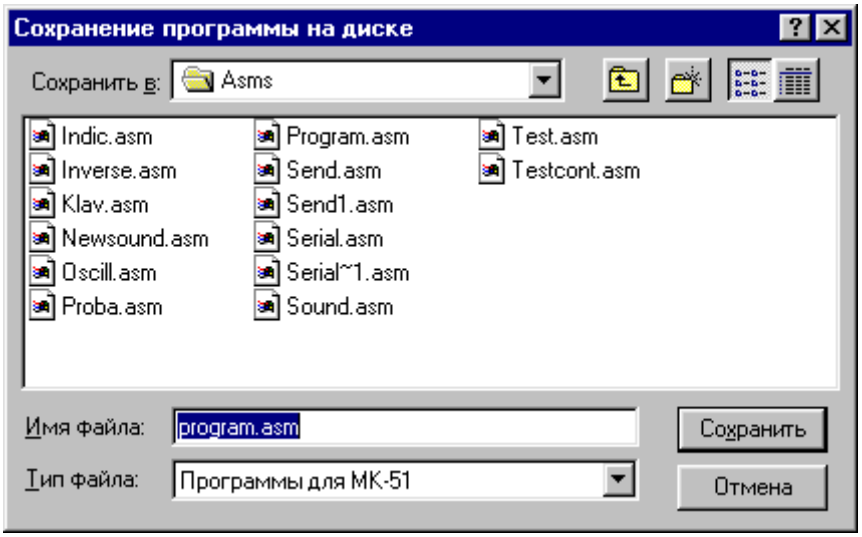


Рис. П4.4. Диалог сохранения файла

При наличии на диске файла с именем, идентичным имени сохраняемого файла, высвечивается окно «Сохранение программы на диске», уточняющее

выполняемое действие: с заменой существующего файла или отмены операции записи.

Опция «Поиск» вкладки «Текст программы» используется для облегчения работы с текстами большого объема, в том числе для поиска строк, указанных транслятором как содержащие ошибки.

Текстовую информацию, подлежащую поиску, следует поместить в поле ввода, с помощью кнопки-панели «Вверх» («Вниз») задать направление поиска (от начала документа к концу или наоборот). По нажатию кнопки-панели «Найти далее» в активном окне текста будет выделена строка, содержащая указанный текст. Каждое следующее нажатие этой кнопки-панели продолжает поиск в выбранном направлении (рис. П4.5). Отсутствие выделенной строки означает отсутствие введенного текста в данной программе.

Кнопка-панель «Печать» вкладки «Текст программы» активизирует диалог назначения параметров печати информации, находящейся в поле активного окна текста программы. При выполнении лабораторных работ вывод текста программы на печатающее устройство не предусмотрен.

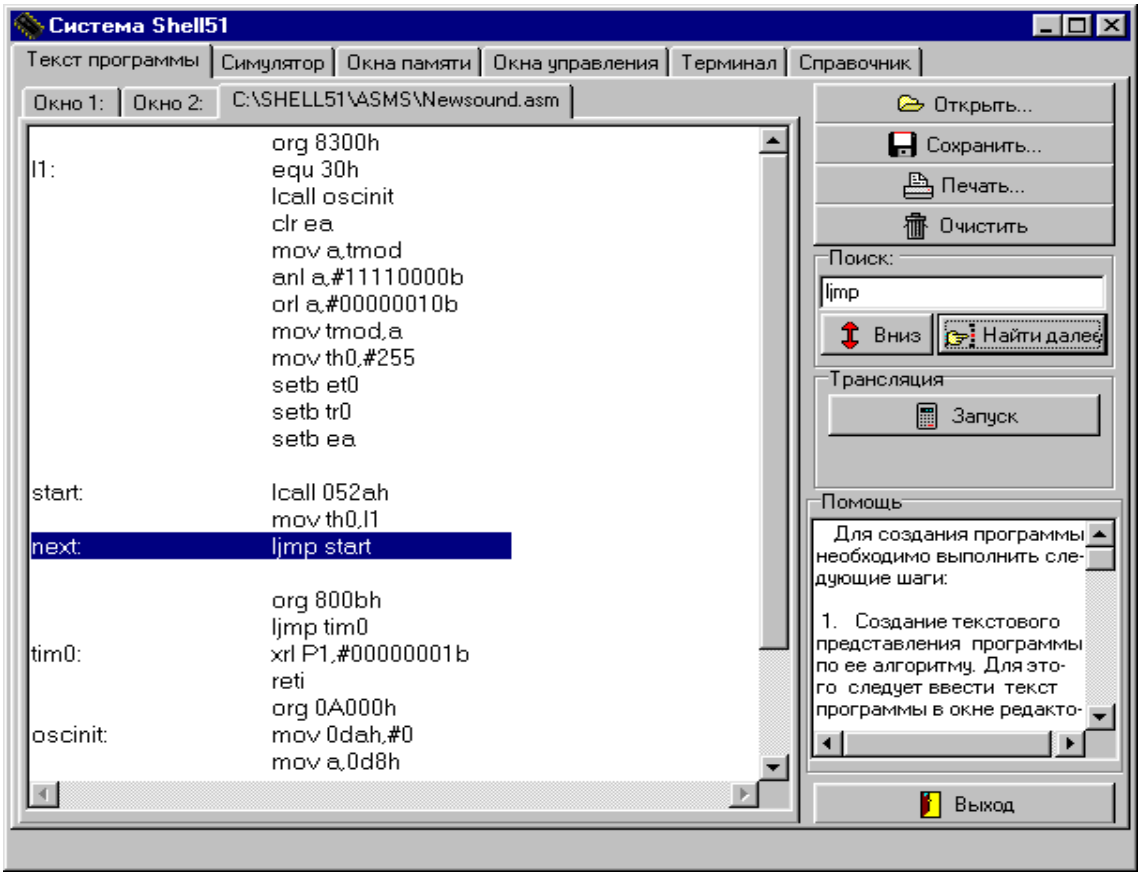


Рис. П4.5. Поиск в тексте программы

Опция «Трансляция» вкладки «Текст программы» обеспечивает трансляцию исходного текста программы в активном окне в представление, пригодное для загрузки в микроконтроллер: ассемблерный текст программы

преобразуется в объектный (машинный) код, после чего происходит преобразование объектного кода в шестнадцатеричный исполняемый код.

Процесс трансляции запускается при нажатии **кнопки-панели «Запуск»** в поле «Трансляция». Перед запуском трансляции новая или модифицированная программа должна быть сохранена. Результат трансляции отображается в виде двух диагностических сообщений: «ОК» - при отсутствии ошибок в программе и предупреждения «Внимание!» со стороны транслятора и компоновщика - в противном случае.

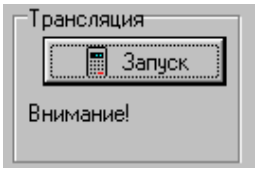


Рис. П4.6. Текст с сообщением о результате трансляции

Детальная информация о результатах трансляции и компоновки расположена на вкладке «Листинги».

Вкладка «Листинги» предназначена для анализа результатов выполнения этапа трансляции и компоновки текста исходной программы. Для вызова опции «Листинги» необходимо щелкнуть манипулятором «мышь» на строке-сообщении о результате трансляции «Внимание» (рис. П4.6). Название опции «Листинги» появляется во вкладке «Текст программы» на месте опции «Симулятор».

При активизации опции «Листинги» щелчком манипулятора «мышь» на имени опции открывается окно-вкладка «Листинг трансляции» (рис. П4.7). Вкладка содержит поле вывода листинга результатов трансляции (слева вверху), поле вывода листинга результатов компоновки (слева внизу), поле перечня ошибок, обнаруживаемых ассемблером (справа вверху), поле перечня ошибок, обнаруживаемых компоновщиком (по центру внизу), поле помощи по использованию данной вкладки (справа внизу), поле ввода строки контекстного поиска, а также три управляющих кнопки-панели.

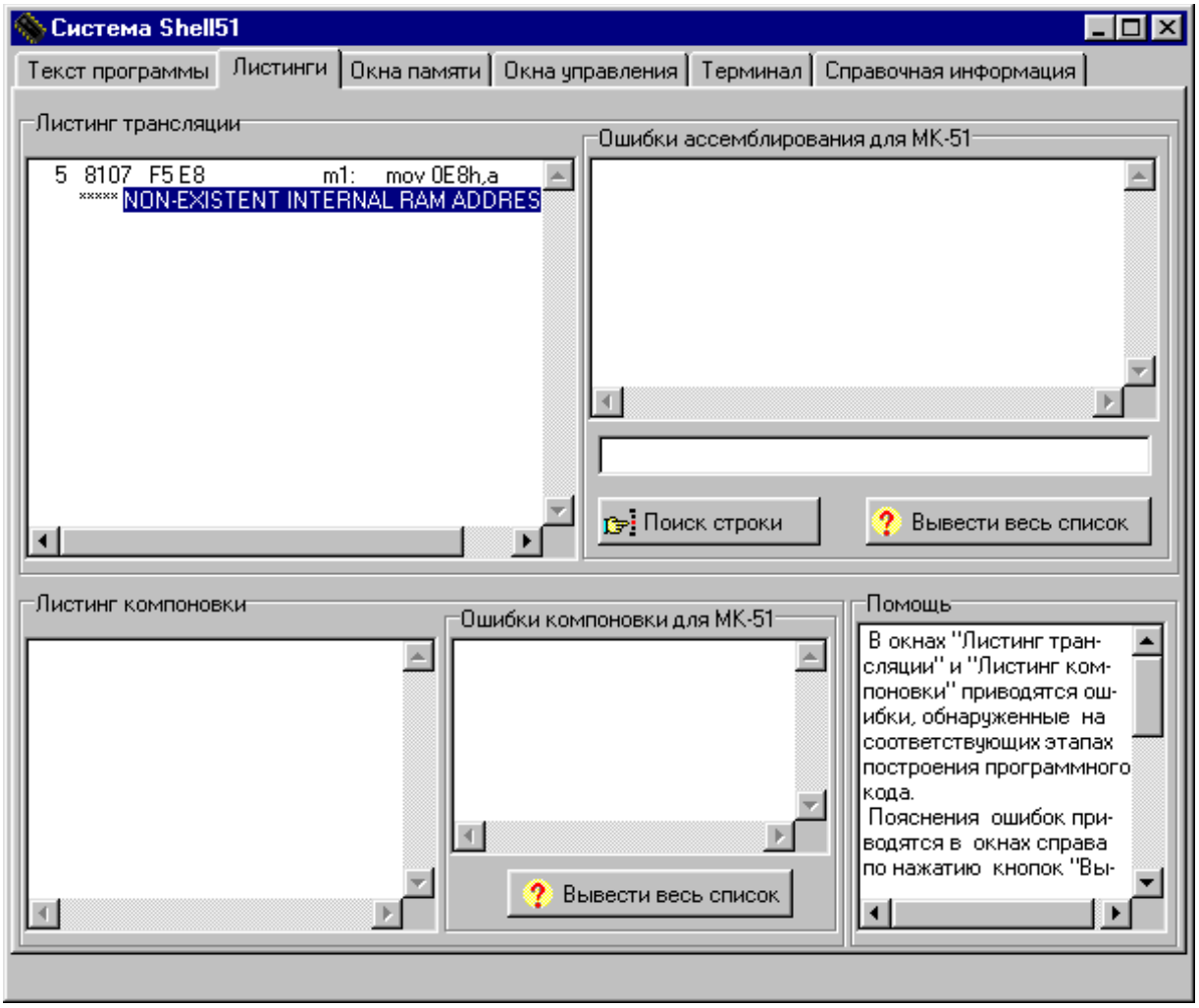


Рис. П4.7. Окно-вкладка листинга трансляции компоновки

В полях листинга отображаются результаты прохождения пользовательской программой этапов ассемблирования (преобразования в объектный код) и компоновки (преобразования в шестнадцатеричный исполняемый код). При наличии сообщений об ошибках необходим анализ текста программы, выявление недопустимых элементов пользовательской программы и их исключение.

С помощью кнопок-панелей «Вывести весь список» система Shell51 позволяет просмотреть полный перечень ошибок этапов ассемблирования и компоновки. Дополнительно имеется возможность выполнить контекстный поиск информации по конкретной ошибке трансляции с языка ассемблера. Для проведения контекстного поиска в поле «Поиск строки» необходимо вставить строку с сообщением об ошибке и активизировать одноименную кнопку-панель «Поиск строки». Сообщение об ошибке может быть введено в поле «Поиск строки» путем набора сообщения с помощью клавиатуры. Альтернативным способом ввода сообщения является его загрузка из буфера обмена (при нажатии комбинации клавиш Ctrl+V), в который сообщение предварительно необходимо скопировать из списка ошибок, указанных в листинге трансляции.

При выполнении указанных операций в поле «Ошибки ассемблирования» появится детальное описание данной ошибки и рекомендации по ее устранению (рис. П4.8). Пустое поле свидетельствует о неточном вводе строки либо отсутствии описания данной ошибки.

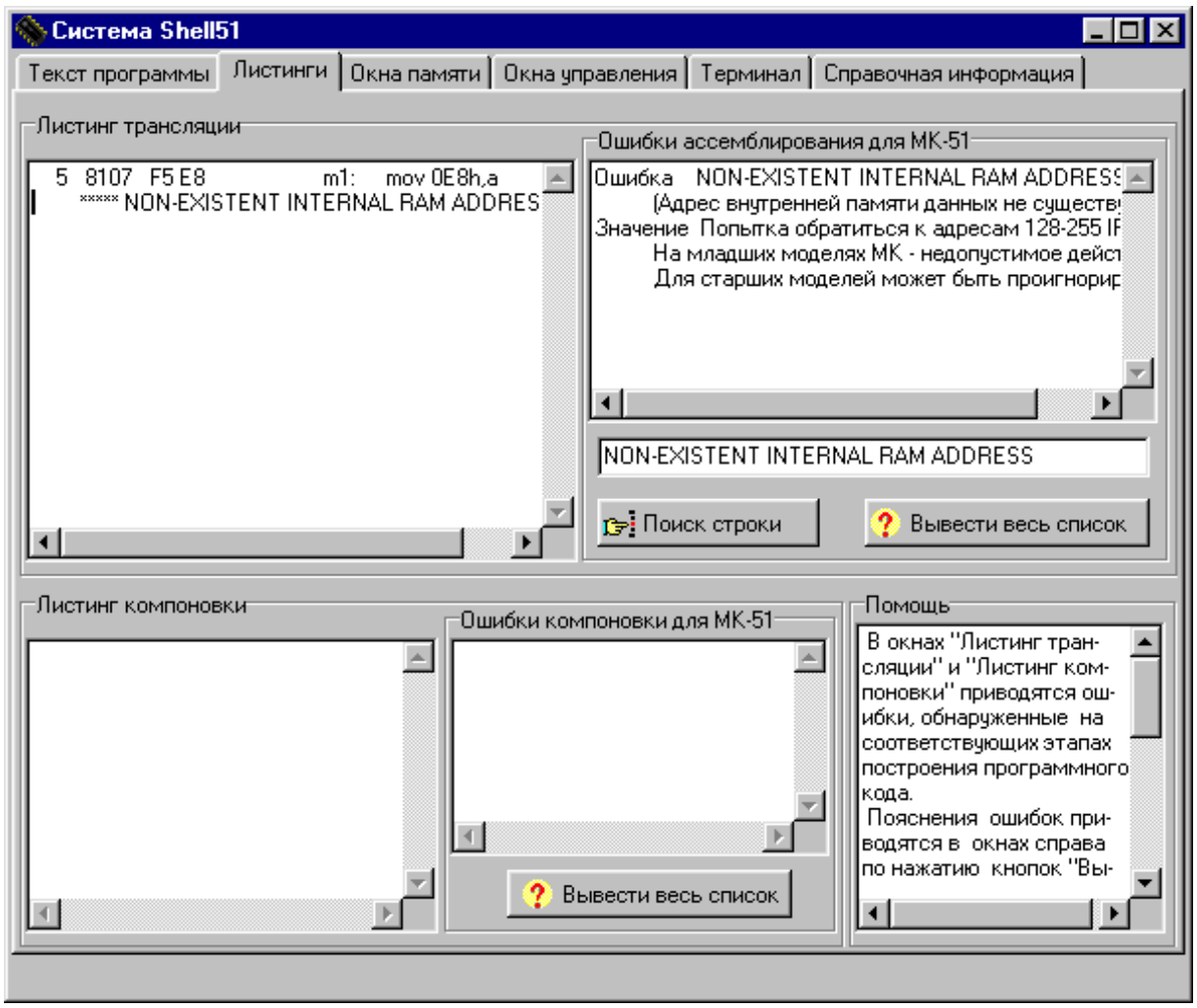


Рис. П4.8. Проведение контекстного поиска

При наличии ошибок результирующий исполняемый файл для микроконтроллера создан не будет.

Неправомочное игнорирование пользователем предупреждений может повлечь за собой работу с исполняемым файлом, не соответствующим исходной программе.

Вкладка «Симулятор»

Симулятор является программным средством отладки программ. Симулятор содержит в своем составе программную модель МК и отладчик. Содержимое внутренней памяти данных МК и состояние периферийных устройств МК с помощью программы-отладчика отображается на экране инструментальной ЭВМ.

Окно-вкладка «Симулятор» (рис. П4.9) открывается при активизации одноименной опции «Симулятор» вкладки «Текст программы».

Вкладка «Симулятор» содержит окно памяти данных программной модели МК (симулятора), окно памяти программ симулятора (справа), окно помощи (слева внизу) и органы управления симулятором (справа внизу).

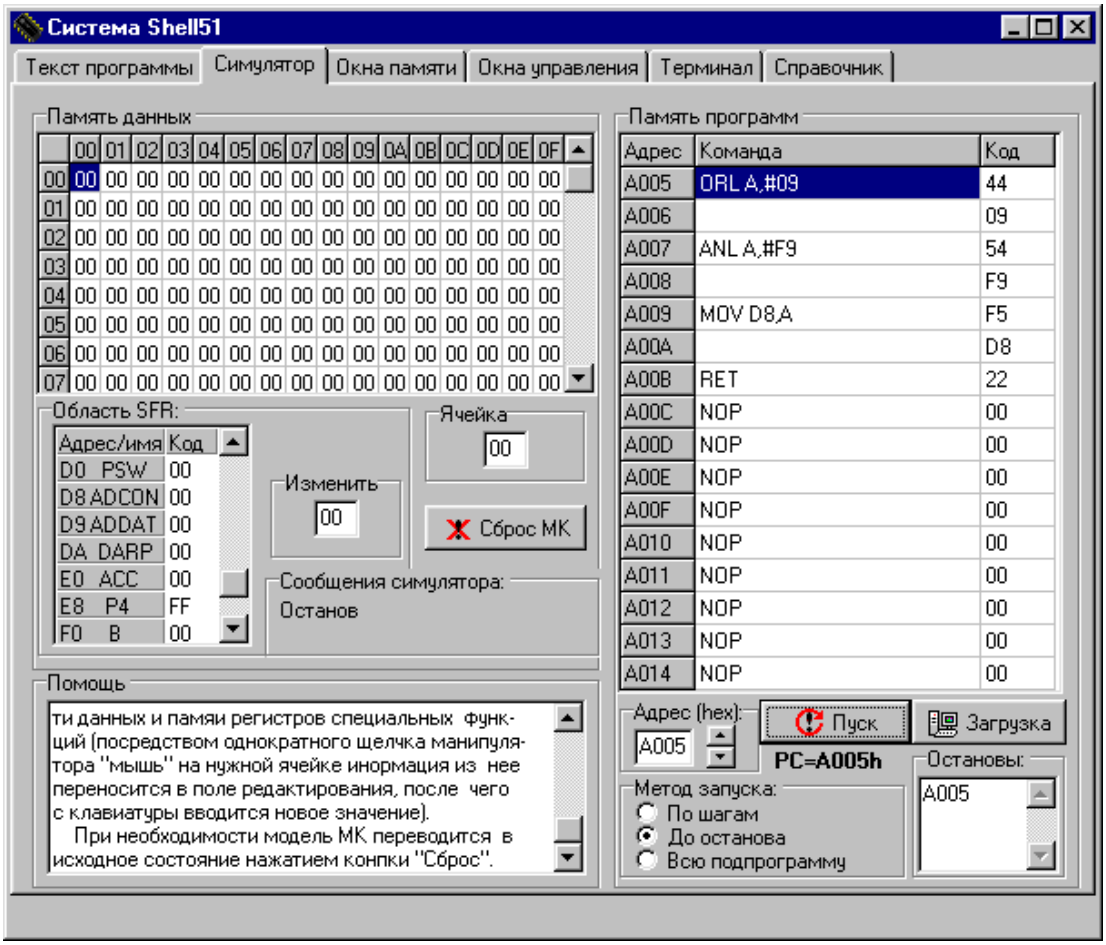


Рис. П4.9. Вкладка "Симулятор".

Симулятор позволяет загрузить оттранслированную программу в память симулятора и отслеживать ее выполнение либо по шагам, либо до установленного пользователем адреса, либо целиком.

Загрузка оттранслированной программы производится при нажатии кнопки-панели «Загрузка». Способ выполнения программы задается с помощью во вкладке «Метод запуска» (рис. П4.10). Выполнение команд программы реализуется при нажатии кнопки «Пуск».

При выборе способа исполнения «По шагам» останов симулятора производится после выполнения каждой команды.

Способ исполнения «До останова» обеспечивает выполнение команд программы до точки останова, соответствующей команде, адрес которой указан в окне «Остановы». Точки останова задаются/удаляются в окне «Остановы» однократным щелчком манипулятора «мышь» на строке команды в памяти программ (рис. П4.11).

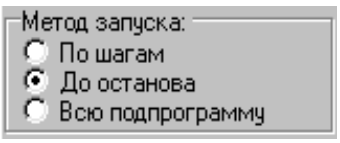


Рис. П4.10. Метод запуска на выполнение

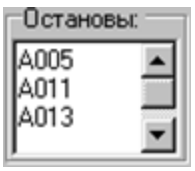


Рис. П4.11. Список Остановов

Способ исполнения «Всю подпрограмму» обеспечивает выполнение всей программы.

Установка счетчика команд (PC) на задаваемый адрес исполняемой команды реализуется двойным щелчком манипулятора «мышь» на строке памяти программ, содержащей необходимую команду. По ходу выполнения программы указатель PC отображает адрес очередной команды из памяти программ симулятора.

Просмотр содержимого памяти программ производится с помощью окна «Адрес». Допускается как непосредственный ввод шестнадцатеричного адреса участка памяти, подлежащего просмотру, так и использование бегунка (рис. П4.12).

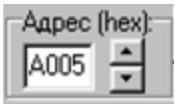


Рис. П4.12. Ввод адреса просмотра программы

Память данных симулятора представлена в виде двумерного массива ячеек, адрес которых определяется как $У_{16}*10_{16}+Х_{16}$, где Х и У – шестнадцатеричные значения номеров столбца и строки, которые содержат искомую ячейку. Память регистров специальных функций представлена линейным списком с указанием символических имен регистров SFR.

Пользователь имеет возможность корректировать информацию в памяти данных и памяти регистров SFR. Для модификации данных в указанных областях памяти необходимо выделить нужную ячейку. Эта операция выполняется при подведении курсора к требуемой ячейке и однократном щелчке манипулятором «мышь». При выделении ячейки информация из нее переносится в поле редактирования содержимого ячейки памяти данных симулятора (рис. П4.13) или регистра SFR (рис. П4.14). После этого с помощью клавиатуры в выбранную ячейку можно вводить новое значение.

При нажатии кнопки-панели «Сброс» модель МК переводится в исходное состояние.

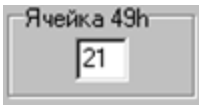


Рис. П4.13. Изменение значения ячейки памяти данных симулятора

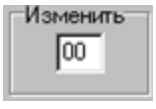


Рис. П4.14. Изменение значения регистра SFR

Симулятор позволяет быстро проверить логику выполняемой программы, но, к сожалению, он не учитывают в полном объеме реальных характеристик МК и подключаемых к нему внешних устройств.

Комплексная проверка работы микроконтроллерной системы с подключенными внешними устройствами выполняется на лабораторном стенде с использованием вкладки «Окна памяти» системы Shell51.

Вкладка «Окна памяти»

Вкладка «Окна памяти» (рис. П4.15) предназначена для работы с содержимым памяти программ и данных МК, загрузки в МК пользовательских программ и их запуске на исполнение.

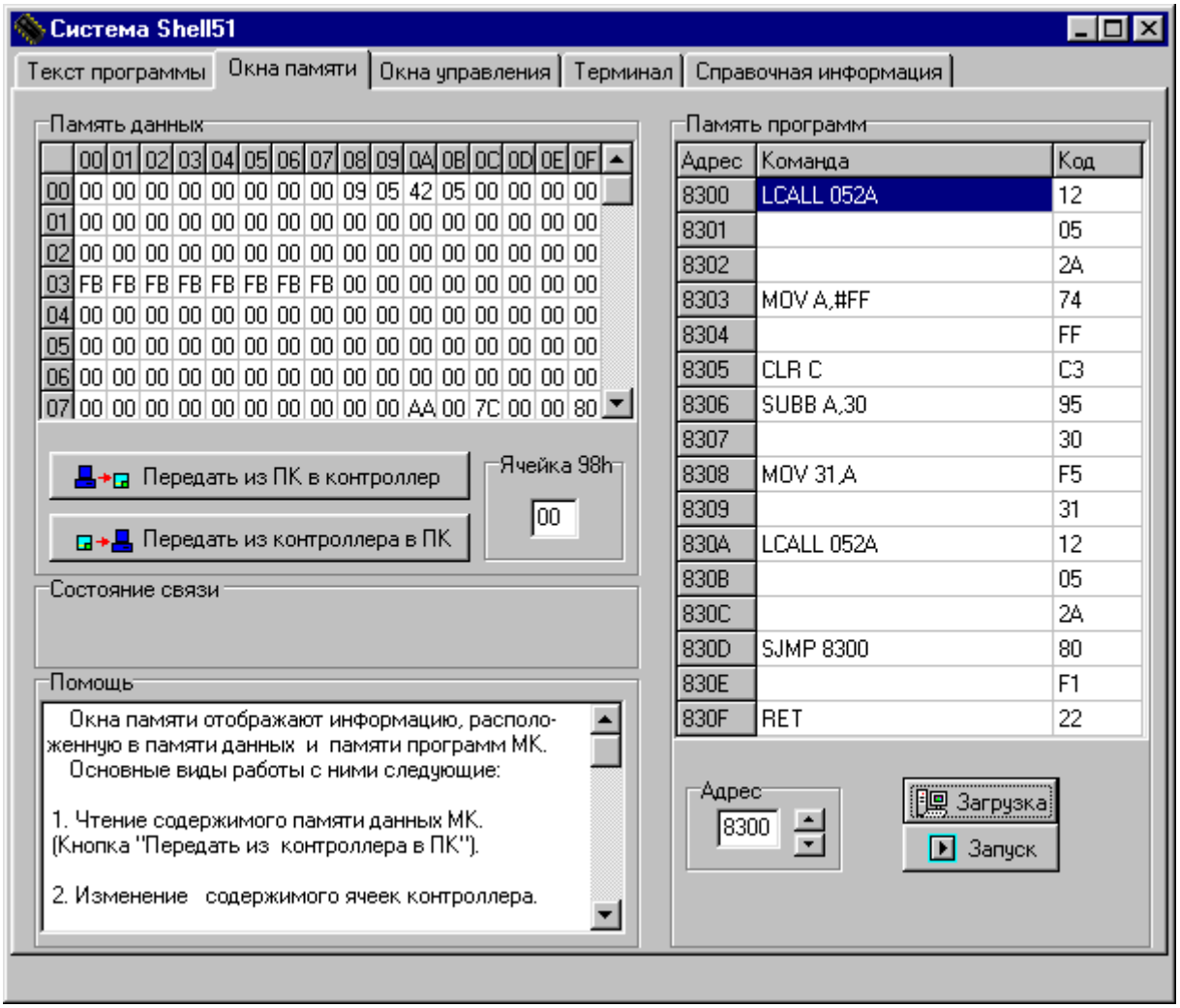


Рис. П4.15. Вкладка "Окна памяти" системы

Вкладка содержит окно внутренней памяти данных микроконтроллера (слева вверху), окно текущей ячейки внутренней памяти данных (в центре), окно внешней совмещенной памяти программ-данных (справа), окно справки по использованию данной опции (слева внизу), поле состояния связи и ряд кнопок-панелей.

Внутренняя память данных МК представлена в виде двумерного массива ячеек, адрес которых определяется как $У_{16}*10_{16}+Х_{16}$, где Х и У – шестнадцатеричные значения номеров столбца и строки ячейки.

Любая ячейка внутренней памяти данных допускает модификацию. Для изменения содержимого ячейки памяти ее необходимо выделить (выполняется однократным щелчком левой кнопкой манипулятора «мышь» на ячейке), при этом содержимое выделенной ячейки переносится в поле редактирования ячейки памяти данных микроконтроллера (рис. П4.16). Затем

с помощью клавиатуры требуемое шестнадцатеричное значение данных вводится в окно редактирования содержимого выбранной ячейки и из него переписывается в окно памяти данных отладчика. Измененное значение необходимо передать во внутреннюю память данных МК.

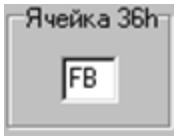


Рис. П4.16.
Ввод значения
выбранной ячейки

Обмены информацией между окном внутренней памяти данных среды Shell51 и внутренней памятью данных МК, выполняемые по последовательному каналу, синхронизированы. Обмены выполняются в сеансах связи, инициируемых при нажатии кнопок-панелей «Передать из ПК в контроллер» и «Передать из контроллера в ПК». Для записи новых значений в память данных контроллера используется кнопка-панель «Передать из ПК в контроллер». Чтение (индикация) содержимого внутренней памяти данных контроллера выполняется при нажатии кнопки-панели «Передать из контроллера в ПК».

В ходе обмена информацией поле «Состояние связи» отображает статус линий связи. Информационное сообщение «Обмен с МК» свидетельствует о корректности выполнения очередного сеанса обмена. Сообщение «Нет связи» информирует о невозможности выполнения обмена и блокировке дальнейших сеансов обмена. Это сообщение формируется при отсутствии соединения стенда с ПК (необходимо проверить наличие «питания» стенда и состояние кнопки управления последовательным каналом – она должна быть не нажата). Более распространенной причиной нарушения синхронизации обмена по последовательному каналу являются ошибки в пользовательской программе. Вызов сеанса связи разрешен, если пользовательская программа завершена, т.е. выполнена команда *ret*, передающая управление оболочке Shell51. Если команда *ret* не выполняется (она отсутствует в тексте программы или пользовательская программа содержит незавершающиеся циклы), вызов сеанса связи невозможен. Пользователь должен выявить причину нарушения синхронизации обмена и устранить ее, а затем снять блокировку двойным щелчком манипулятора «мышь» по сообщению.

Перед работой с памятью МК (чтение, загрузка) рекомендуется проверить наличие связи «ПК-МК» путем передачи информации из памяти данных МК в ПК. Эта проверка выполняется при нажатии кнопки-панели «Передать из контроллера в ПК».

Пользовательская программа в исполняемом формате, полученном на этапе трансляции, загружается в память программ контроллера при нажатии кнопки-панели «Загрузить». При загрузке программы в окне памяти программ, начиная с адреса, заданного директивой ассемблера «org <адрес>», отображаются коды первых 16 байт пользовательской программы и их дизассемблированный текст.

Собственно выполнение пользовательской программы активизируется при нажатии кнопки-панели «Запуск». Исполнение программы, осуществляется с адреса, указанного в окне «Адрес». При необходимости значение адреса может быть скорректировано пользователем (рис.П4.17), при этом в окне памяти программ отобразится иной 16-байтный участок памяти программ, начинающийся с заданного адреса.

Рис. П4.17.
Поле ввода адреса
памяти программ

Вкладка «Окна управления»

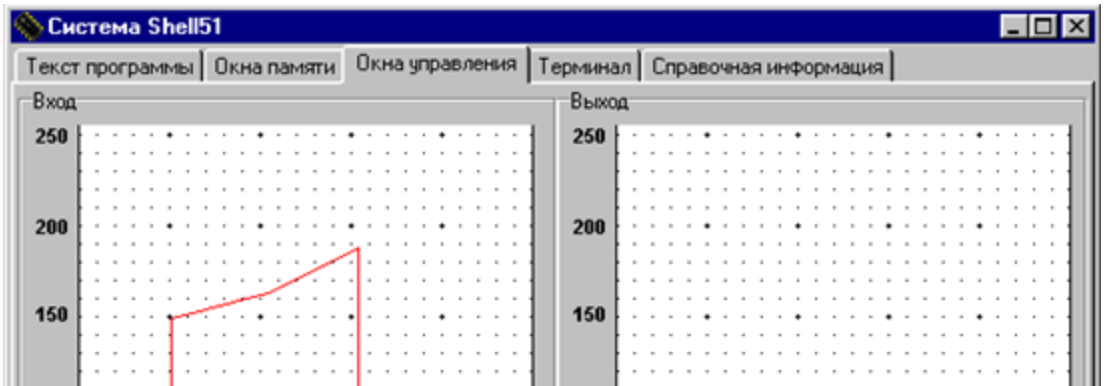
Вкладка «Окна управления» предназначена для интерактивного исследования систем управления и отладки пользовательских программ. Вкладка содержит: поле формируемого управляющего воздействия на контроллер (слева), поле реакции контроллера на управления (справа), поле справки (слева внизу), и поле настроек подсистемы (справа внизу) (рис.П4.18) .

В окне поля управляющего воздействия, называемом «Вход», пользователь имеет возможность сформировать или выбрать тип управляющего воздействия в виде дискретной функции $\Phi(n)$, где n - номер выборки функции Φ.

Выбор ранее созданной и сохраненной на диске ПК функции управляющего воздействия Φ осуществляется при нажатии кнопки-панели «Открыть».

График управляющего воздействия $\Phi(n)$ строится с помощью манипулятора «мышь» в виде кусочно-линейной функции. Двойной щелчок означает начало/окончание режима ввода, однократный - фиксацию текущего отрезка графика. Для удобства построения графика $\Phi(n)$ в поле «Позиция» отображаются текущие координаты манипулятора. График $\Phi(n)$ можно запомнить на диске ПК, нажав кнопку-панель «Сохранить».

В окне «Выход» отображается динамика изменения контролируемого параметра пользовательской программы в процессе ее выполнения. По выполняемой функции окно «Выход» фактически является «цифровым осциллографом». При запуске «осциллографа» на экране ПК отображается график функции $M(n)$, являющейся результатом функционирования пользовательской программы на контроллере. Формируемый в окне «Выход» график $M(n)$ можно запомнить на диске ПК, нажав кнопку-панель «Сохранить».



Функции $\Phi(n)$ и $M(n)$ представляют собой развернутые во времени значения задаваемых пользователем ячеек внутренней памяти данных, в связи с чем обе функции нужно адресовать, указав необходимые значения в полях «Адрес входа и «Адрес выхода».

Длительность развертывания наблюдаемых процессов в условных временных единицах задается изменением положения движка «Период» посредством манипулятора «мышь».

Начало сеанса интерактивного управления/регистрации инициируется при нажатии кнопки-панели «Пуск».

Возможны три варианта использования вкладки «Окна управления»:

- формирование управляющего воздействия $\Phi(n)$ на МК в окне «Вход»;
- отображение (регистрация) состояния контроллера - функция $M(n)$ в окне «Выход»;
- совместное применение функций $\Phi(n)$ и $M(n)$ при формировании управляющего воздействия $\Phi(n)$ на МК в окне «Вход» с получением резу результирующего отклика в окне «Выход».

Выбор необходимого варианта осуществляется установкой переключателя «Цикл» в требуемое положение.

Пользовательская программа, работающая совместно с программами вкладки «Окна управления», должна содержать команду вызова подпрограммы однократного сеанса связи МК с ПК *lcall 128h*. В сеансе связи компьютер выдает очередное значение управляющей функции $\Phi(n)$ и/или принимает значение отображаемого параметра с выхода МК $M(n)$. Поскольку

программное управление обычно организуется в виде цикла, команду *lcall 128h* следует поместить внутрь цикла.

Другим вариантом организации взаимодействия оболочки Shell51 (компьютера) с МК (стендом) является взаимодействие по прерываниям от приемопередатчика последовательного порта. При данном способе организации взаимодействия ПК с МК команда *lcall 128h* помещается в обработчик прерывания последовательного порта МК.

Вкладка «Терминал»

Вкладка «Терминал» предназначена для исследования последовательного интерфейса связи МК с ЭВМ без участия связных компонент программы-монитора, и построения пользовательских связных компонент.

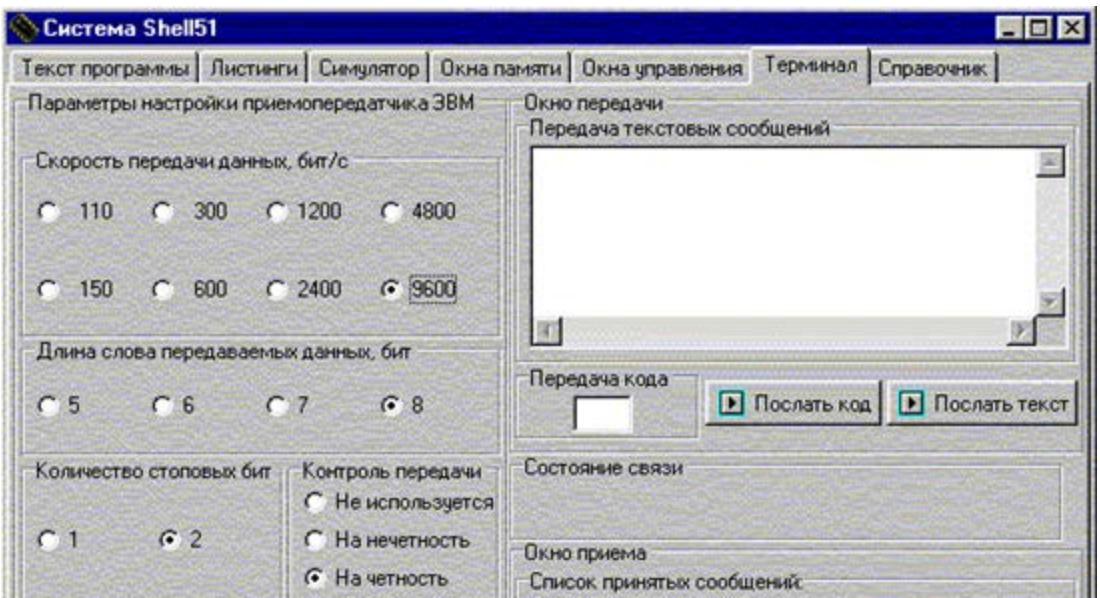
Окно вкладки (рис. П3.19) содержит: поле настроек приемопередатчика последовательного порта инструментальной ЭВМ (слева вверху), окно справки (слева внизу), поле окна передачи (справа вверху), поле окна приема (справа внизу) и поле состояния связи (справа в центре).

Основой правильной работы последовательного интерфейса является синхронность работы приемника и передатчика, достигаемая заданием одинаковых значений скорости передачи и длины слова передаваемых бит для приемопередатчиков ПК и МК. Требуемый режим работы приемопередатчика ПК задается с помощью переключателей поля настроек приемопередатчика (рис. П4.19). Аналогичный режим приемопередатчика МК программируется пользователем.

При исследовании системы связи компьютера с МК пользователю доступны следующие варианты организации работы подсистемы:

1. Посылка информации в порт:

- если посылка однобайтная, используется окно передачи шестнадцатеричных кодов и панель-кнопка «Послать код», активизирующая выдачу значения в последовательный порт на МК;
- если посылка многобайтная, целесообразно использовать окно передачи текста, предварительно введенного с клавиатуры ЭВМ, и панель-кнопку «Послать текст», при нажатии которой производится выдача кодов символов, составляющих текст.



2. Прием информации из порта:

- при активизации кнопки-панели «Принять однократно» код, содержащийся в буфере приемника ЭВМ, отображается в окне «Принятый код», а также очередным эквивалентным символом в окне «Принятый текст». Кнопка-панель «Очистить» позволяет освободить окна приема от содержащейся в них информации;
- при активизации кнопки-панели «Выдать старт-байт и принимать циклически» система посылает в последовательный порт код $A5_{16}$, а затем переходит к циклу ожидания поступления входной информации из последовательного порта. Старт-байт может быть использован пользователем как уведомление о необходимости передавать результаты в ЭВМ.

3. Комбинация вышеуказанных вариантов.

Приложение 5

ЧАСТО ИСПОЛЬЗУЕМЫЕ ДИРЕКТИВЫ АССЕМБЛЕРА X8051

Введение в программирование на ассемблере x8051. В настоящее время при проектировании микропроцессорных систем широко применяются программные среды, в которых программирование выполняется на языках высокого уровня. Такой подход позволяет значительно упростить и ускорить

процедуру создания проекта, поскольку он не требует знаний тонкостей архитектуры и структуры используемых технических средств. Некоторым недостатком способа программирования на языках высокого уровня является то, что он не всегда обеспечивает оптимальное использование аппаратных ресурсов процессора и не позволяет минимизировать объем программного кода и некоторые другие параметры системы. При повышенных требованиях к быстродействию, массогабаритным характеристикам, пониженному энергопотреблению применяют машинно-зависимые языки- ассемблеры.

Принципиальное отличие языка ассемблера от других языков программирования состоит в том, что язык ассемблера оперирует командами, которые непосредственно распознает и выполняет процессор. По исходному тексту программы, написанной на ассемблере, всегда можно определить время исполнения программы и требуемый объем программной памяти. Ассемблер позволяет программисту наилучшим образом использовать аппаратные ресурсы процессора, обеспечивая доступ к любому из встроенных периферийных устройств наиболее быстрым и рациональным способом. Именно поэтому, несмотря на некоторую специфику использования ассемблера, связанную с необходимостью хорошего знания аппаратной части процессора, ассемблер является основным языком программирования микропроцессорных систем.

Ассемблеры разрабатываются для конкретного процессора или серии процессоров, имеющих единую базовую систему команд. Главное достоинство языка ассемблера заключается в том, что он допускает представление всех элементов программы в символической (буквенно-цифровой) форме, отражающих их содержательный смысл. Преобразование символических наименований в двоичные коды машинного языка возлагается на специальную программу, называемую ассемблирующей программой или транслятором с ассемблера. Процесс трансляции символических имен в машинный объектный код называется ассемблированием или компиляцией.

Любая программа на ассемблере, кроме машинных команд, исполняемых процессором, содержит неисполняемые команды - директивы ассемблера.

Машинные команды - это команды из системы команд микроконтроллера. При трансляции исходного текста программы мнемоники команд процессора в конечном итоге преобразуются в исполняемый объектный код.

Директивы ассемблера - это специальные команды транслятору о необходимости выполнения определенных действий в процессе ассемблирования. Фактически они являются командами управления процессом ассемблирования. Директивы ассемблера не преобразуются в двоичные коды, а потому не могут быть выполнены процессором. Они являются псевдокомандами и применяются для задания структуры программы, определения переменных, резервирования памяти и ряда других целей.

Директивы можно разделить на несколько групп:

- директивы определения программного модуля и организации межмодульных связей: *MODULE, PUBLIC, EXTERN, GLOBAL, END*;
- директивы управления размещением данных в отдельных сегментах: *USING, SEG, DEFSEG*:
- директивы определения символических имен переменных и задания атрибутов переменных (значения данных, их типа): *EQU, SET*;
- директивы резервирования места переменных в памяти: *DB, DW, DS*;
- директивы условного ассемблирования: *IF, ENDIF*;
- специальные операторы для использования в макросах : *IFB, IFNB, IFEQ*

Для удобства представления и чтения программы, написанные на ассемблере, представляют в виде последовательности строк (операторов) стандартного формата: каждый оператор, т.е. любая машинная команда или директива, записываются в одной строке, имеющей четыре поля:

Метка Код Операнд Комментарий

Поля ассемблерной строки имеют следующее назначение:

Поле «Метка» может содержать необязательное символическое имя. Метки ассоциируются с адресами памяти. Метки, как и имена операндов, нельзя дублировать. Метка имеет длину от одного до шести символов, первым из которых должна быть буква. За последним символом метки должен указываться специальный разделитель - двоеточие. Метка может предшествовать машинным командам, макровызовам, директивам резервирования памяти, пустым операторам. Часто метки используются в качестве адресов команд передачи управления. Они освобождают программиста от необходимости оперировать абсолютными адресами. Для некоторых директив ассемблера, например, директив MACRO, EQU, SET имя, указываемое в поле «Метка» является обязательным атрибутом. В директивах, в которых используется символическое имя, двоеточие после имени не ставится.

Поле мнемоники (поле «Код») является единственным обязательным полем команды. Оно содержит символическое описание машинной команды, директивы ассемблера или имя пользовательской макрокоманды.

Большинство мнемоник представляют собой аббревиатуры предложений, характеризующих основные функции машинных команд и директив. Если содержимое этого поля не входит в множество допустимых мнемоник, ассемблер выдает сообщение о недействительной команде.

Поле «Операнд» в большинстве случаев содержит имена операндов, участвующих в операции. Содержимое и синтаксис этого поля зависят от мнемокода команды, указываемой в поле «Код». В качестве операндов могут

фигурировать адреса памяти, внутренние регистры МК, адреса портов ввода-вывода, числовые и символьные константы. В большинстве команд символическое имя операнда-приемника в поле «Операнд» указывается первым и отделяется от символического имени операнда-источника запятой. В некоторых машинных командах и директивах поле «Операнд» не используется.

Константы бывают двух типов: численные и адресные. Численные константы в командах обозначаются как непосредственные операнды *#data*, а адресные не имеют специальных обозначений. Символы могут быть заключены в апострофы (‘’): *mov a,#‘C’*. Константы могут быть представлены явно (конкретными значениями) и неявно (выражениями, вычисляемыми компилятором). При явном задании констант с помощью спецификаторов указывается система счисления. Спецификаторы могут указываться как перед константой, так и после нее. Обозначение спецификатора зависит от его типа и системы счисления:

Основание	Начальный спецификатор	Конечный спецификатор
2	%	B
8	@	O or Q
16	$	H

Примеры: mov a,#254; mov a,#%11111110; mov a,# 11111110b;
mov a,#@200; mov a,#376Q; mov a,$FE; mov a,#FEh; mov dptr,# A5F2h

Поле комментария используется для обозначения действий, которые выполняет команда в контексте конкретной программы. Поле комментария должно начинаться с некоторого разделителя (часто с точки запятой). Комментарий не ассемблируется, поэтому в нем можно помещать любой текст. В отношении содержания комментария желательно руководствоваться следующими рекомендациями. Во-первых, в комментарии акцентируется не общая функция команды, которая определяется мнемоникой, а действие данной команды в контексте данной программы. Во-вторых, не следует сокращать объем комментария в ущерб смыслу. Четкие и содержательные комментарии облегчают понимание и использование программы.

Ассемблер допускает обращение к переменным по символическим адресам, а к данным (константам) по предварительно определенным символическим именам. Поддержка символических адресов исключает необходимость работы с абсолютными адресами, при этом возможность задания значимых имен адресам переменных и константам делает программу более понятной.

Имена, определяемые пользователем, являются идентификаторами и могут содержать до 31 символов. Символические имена должны начинаться с алфавитного символа (или специального символа ? или _) и могут включать любую комбинацию букв, цифр, знака вопроса и подчеркиваний. Регистр

символов внутри идентификатора ассемблером игнорируется (допускается произвольное смешивание строчных и прописных букв; все они воспринимаются как прописные). В качестве символических имен не допускается использование зарезервированных слов. Зарезервированные слова - это имена, которые имеют специфическое значение а ассемблере и недоступны для определения пользователем. Примером зарезервированных слов являются мнемоники машинных команд, директивы ассемблера, макродирективы, идентификатор STACK. Особое значение в ассемблере имеет символ $, называемый счетчиком размещения. С помощью символа $ отмечается текущая ячейка внутри активного сегмента программы. Счетчик размещения обычно используется для поддержки режимов относительной адресации операндов.

Для справки приведем описание часто употребляемых директив ассемблера x8051.

Директивы управления адресами и файлами программ:

Команды размещаются в программном сегменте по последовательным адресам. Адрес команды определяется содержимым счетчика команд PC. При компиляции значение счетчика PC в соответствии с программным кодом автоматически модифицируется. Для задания начального значения счетчика PC (начального адреса программы) применяется директива *ORG VALUE*, имеющая следующий формат:

Метка	Код	Операнд
[Label:]	ORG	<адресное выражение>

До новой директивы ORG команды и данные размещаются в смежных ячейках памяти. Если в самом начале программы директива ORG отсутствует, то по умолчанию подразумевается наличие директивы ORG с нулевым операндом.

Использование меток для обозначения адресов команд позволяет реализовать ветвления по командам условного (JC, JNC, JZ, JB, JNB, JBC, DJNZ) и безусловного (JMP) переходов или команды CALL.

Команды *JMP* и *CALL* конкретизируются компилятором в соответствующие команды *LJMP, AJMP, SJMP, LCALL, SCALL, JMP@+dptr* с учетом адресного интервала при переходе.

В качестве метки команды можно использовать символ $, который, как отмечено выше, отображает текущее значение счетчика команд PC. Например, команда

```
      djnz   R0,$    эквивалентна команде перехода с меткой
m1:   djnz   R0, m1
```

После сегмента CODE в ассемблерном тексте рекомендуется размещать сегмент DATE, который предназначен для хранения переменных и результатов вычислений. В сегменте DATE можно использовать только директивы резервирования переменных DB, DW, DS.

Директива END VALUE отмечает конец программы или подключенного файла. Выражение, следующее за END, дополнительное и, если существует, указывает стартовый адрес программы.

Директива INCLUDE <имя файла> позволяет подключать в тело программы фрагменты кода файла, задаваемого в поле «имя файла». Директива INCLUDE указывает ассемблеру на необходимость трансляции текста, расположенного в подключаемом файле, начиная с адреса, следующего за адресом последней команды, расположенной в текущем файле.

При использовании директивы *include <имя файла>* часто фиксируются ошибки, связанные с наложением переменных и участков программ (при любых вызовах по общим адресам формируемое обращение будет производиться к участку кода последнего оттранслированного модуля). Поэтому при использовании директивы *include <имя файла>* необходимо выполнять следующие требования:

- имя файла, указываемого в директиве *include <имя файла>* не должно превышать 8 символов;
- следует избегать абсолютных адресов в подключаемых файлах;
- нельзя использовать одинаковые метки в выполняемой программе и подключаемых файлах;
- при указании имени подключаемого файла необходимо указывать полный адрес этого файла: asm\4081_3\…\имя_файла.asm.

Директивы определения символических имен

Для определения символических имен в ассемблере предназначены *директивы EQU и SET*, которые имеют следующий формат:

Метка	Код	Операнд	Комментарий
<имя>	EQU	<выражение>	
<имя>	SET	<выражение>	

Директива EQU (EQUAL - приравнять, присвоить) присваивает абсолютное значение, псевдоимя или текстовое символьное имя имени, указанному в поле «Метка». Константу, определенную директивой EQU, нельзя переопределять. При ассемблировании вместо имени, определенного этой директивой, подставляется присвоенное ему значение.

Директива SET (установить) имеет такой же формат и выполняет те же функции, что и директива EQU. Однако в отличие от директивы EQU, значение символического имени в директиве SET можно изменять с помощью новой директивы SET. Имя директивы SET можно рассматривать как переменную, значение которой зависит от этапа компиляции и может быть использовано в условной компиляции.

Директивы резервирования памяти

Директивы DB, DW, DS резервируют память под элементы памяти соответствующего типа и имеют следующий формат

Метка	Код	Операнд	Комментарий
[Label :]	DB	<список>	;формат директивы DB
[Label :]	DW	<список>	;формат директивы DW
[Label :]	DS	<выражение>	;формат директивы DS

Операнд директивы DB может быть последовательностью 8-битных значений, разделяемых запятыми, либо цепочкой символов, заключенных в апострофы. Примеры использования директивы DB:

Метка	Код	Операнд	Комментарий
ARRAY:	DB	3, 7, 15, 31	;запоминаются четыре значения
	DB	'HELLO'	;запоминаются пять символов
COMPL:	DB	- 63	;запоминается дополнительный код числа - 63.

Директивы DB и *DW* не только резервируют память, но и осуществляют инициализацию данных. Операндом директивы DW является последовательность 16-битных значений.

Примеры использования директивы DW:

Метка	Код	Операнд	Комментарий
ADDR:	DW	0FF00h	;(ADDR) = 00h, (ADDR + 1) = FFh
DATA:	DW	100h, 200h	;инициализируются четыре ячейки памяти

Директива DS (STRING – строка) используется только для резервирования памяти. Операндом директивы DS является выражение, определяющее число ячеек (байт) памяти, резервируемых для размещения данных. Инициализация ячеек памяти не выполняется. Примеры использования директивы DS

Метка	Код	Операнд	Комментарий
ARRAY:	DS	32	;резервируются 32 байта в памяти
TABLE:	DS	64	;резервируются 64 байта в памяти

Директива BIT связывает адрес бита с именем булевой переменной. Она имеет следующий формат:

Метка	Код	Операнд
[Name]	BIT	<address bit>
Например:		
X1	BIT	P3.2
S	BIT	ACC.0

Приложение 6

Пояснения и рекомендации по выполнению некоторых заданий

1. Многозадачная операционная система

Многозадачность - это способ одновременного выполнения нескольких задач однопроцессорной системой. Суть многозадачности заключается в организации работы МП при одновременном выполнении нескольких задач

Один из способов реализации многозадачности, называемый *разделением времени*, заключается в предоставлении каждой задаче некоторого интервала времени (кванта обслуживания), в течение которого МП выполняет команды соответствующей программы. Задачи обслуживаются последовательно.

Квант времени выполнения задач обычно задается таймером. Если в течение выделенного программе кванта времени ее обработка не заканчивается, программа прерывается, и она становится в очередь программ, ожидающих обработку. Чтобы имитировать одновременное выполнение процессов, квант времени выполнения задач не должен быть слишком большим.

С целью исключения взаимного влияния задач друг на друга информация о состоянии некоторых ресурсов процессора на момент переключения сохраняется. Эта информация называется контекстом или дескриптором задачи. В состав контекста обязательно входит адрес возврата (адрес команды, с которой будет возобновлено выполнение программы задачи при следующем вызове), а также может входить содержимое регистров общего назначения и регистров специальных функций (для МК SAB 80C515 - это регистр *B*, аккумулятор *A*, слово состояния *psw*, указатель *dptr*, содержимое стека задачи). Поскольку контекст задачи в общем случае имеет достаточно большой объем, контексты задач целесообразно сохранять во внешней памяти МК. В последующем изложении определение «контекст задачи» будем использовать для обозначения области внутренней памяти МК, выделяемой для размещения информации о состоянии некоторых ресурсов процессора при выполнении текущей задачи. Определением «дескриптор задачи» будем обозначать структуру данных с контекстом задачи во внешней памяти.

В многозадачной системе переключение задач осуществляет диспетчер системы – программный модуль, реализующий необходимые действия при переключении задач. Поскольку разрешенное прерывание может прервать любую программу, диспетчер системы целесообразно реализовать в виде обработчика прерывания по переполнению таймера. На обработчик возлагаются следующие функции:

- задание кванта времени выполнения задачи. Квант времени определяется константой перезагрузки счетчика таймера. Каждой задаче в зависимости от

ее паритета может быть задан одинаковый или собственный квант времени выполнения. В системе с разными паритетами задач константы перезагрузки выбираются из контекста вызываемой задачи;

- определение адреса дескриптора выполняемой задачи (определяется по номеру задачи);

- сохранение контекста прерываемой задачи. Запоминание контекста задачи упрощается, если его предварительно разместить в непрерывной области внутренней памяти. В этом случае заполнение дескриптора (запоминание контекста) реализуется командами пересылки во внешнюю память в циклической процедуре;

- определение номера вызываемой задачи и адреса ее дескриптора (выбор очередной задачи);

- восстановление контекста вызываемой задачи;

- передача управления новой задаче по завершении программы обработчика. Передача управления осуществляется автоматически посредством замены содержимого счетчика PC адресом возврата, выбираемым из контекста (восстановленного стека) вызываемой задачи при выполнении последней команды обработчика *reti*.

Формат дескриптора задачи определяется объемом контекста, который, в свою очередь, зависит от числа локальных переменных задачи и их размещении в памяти МК. Учитывая минимально необходимый объем контекста (он рассмотрен выше) будем считать, что объем контекста переключаемых задач не превышает 32 байт и для размещения контекста задачи используется начальная область внутренней памяти МК с адресами 00h – 1Fh. Область внутренней памяти с адресами большими 20h является общим ресурсом для всех задач. Она не перезаписывается при смене задач и может содержать данные, являющиеся для одних задач входными параметрами и выходными для других.

В МК семейства MCS51 указатель стека SP при включении питания инициализируется значением 07h и стек растет вверх (в область старших адресов). Нетрудно заметить, что содержимое регистров R0 – R7 и стека, если указатель SP не инициализировался в программе выполняемой задачи, на момент переключения задач уже являются частью контекста, при этом в вершине находится адрес команды, которая должна выполняться при следующем вызове прерываемой задачи. Этот адрес является адресом возврата $PC_H : PC_L$ в эту задачу. Для формирования полного контекста задачи расширим ее стек, включив в него содержимое регистров SFR, входящих в контекст (*dptr, psw, a, b*). Кроме того, поместим в стек содержимое регистров R0 и R1, которые будут использоваться самим диспетчером при выполнении процедур сохранения/восстановления контекста в дескрипторе задачи. Расширение стека выполним командами *push direct* (*push dph, push dpl, push psw, push b, push a, push 0, push 1*). В результате контекст задачи принимает вид (рис. П.6.1).

Имя	R0	R1	R2	R3	R4	R5	R6	R7	…	…	PC_L	PC_H	dph	dpl	psw	b	a	R0	R1	…	…
Адрес	00	01	02	03	04	05	06	07	Стек задачи				Продолжение области контекста								1Fh

Рис. П6.1 Контекст задачи

Нетрудно заметить, что при выбранной структуре контекста его объем определяется значением указателя SP, который адресует вершину стека. При такой структуре контекста собственно стек задачи не должен иметь глубину более (32-17)=13 байт. Здесь 17 – минимально необходимое число параметров контекста задачи.

Сохранение контекста во внешней памяти и его возврат во внутреннюю память реализуются с помощью циклических процедур, которые могут быть реализованы в виде следующих фрагментов программы (рис.П6.2.2).

```
Циклическая процедура
сохранения контекста
        . . . . . .
        mov r0,sp
        inc r0
        mov dptr, # d_tskn
        mov r1,#0
m1:     mov a,@r1
        movx @dptr,a
        inc r1
        inc dptr
        djnz r0,m1
```

```
Циклическая процедура
восстановления контекста
        . . . . . .
        . . . . . .
        . . . . . .
        mov dptr, # d_tskn
        mov r1,#0
m2:     movx a,@dptr,
        mov @r1,a
        inc r1
        inc dptr
        djnz r0,m2
```

Рис. П6.2. Фрагменты программ сохранения/восстановления

Для задания параметров циклов сохранения/восстановления контекста используются регистры R0 и R1. Регистр R0 содержит число сохраняемых (извлекаемых) параметров контекста. Это число равняется значению указателя SP плюс 1. В процедуре сохранения контекста регистр R0 инициализируется вычисленным значением числа параметров. В процедуре восстановления контекста в регистр R0 заносится число извлекаемых параметров контекста из дескриптора вызываемой программы. Регистр R1 используется в качестве указателя адреса внутренней памяти МК с контекстом задачи.

В процедуре сохранения контекста содержимое контекста переписывается в дескриптор задачи *d_tskn*, размещаемый во внешней памяти по адресу *#d_tskn*. По завершению процедуры сохранения значение R0, равное числу сохраненных параметров sp+1, запоминается в первом байте дескриптора.

В циклической процедуре восстановления контекста вызываемой задачи содержимое дескриптора прерывающей (вызываемой) задачи переписывается в область контекста (внутреннюю память МК), при этом регистр R0 модифицируется автоматически значением числа извлекаемых параметров из

дескриптора вызываемой задачи. Поскольку объем контекста задачи определяется значением указателя SP, значение числа извлекаемых параметров R0 можно использовать для модификации указателя стека SP вызываемой задачи. Эту операцию можно выполнить командами *dec R0* и *mov SP,R0*.

Содержимое регистров SFR вызываемой задачи восстанавливается из стека задачи командами *pop 1, pop 0, pop a, pop b, pop psw, pop dpl, pop dph.* После выполнения указанных команд адрес возврата в вызываемую задачу оказывается в вершине стека. Передача управления этой задаче осуществляется по завершению программы обработчика командой *reti*.

Адрес дескриптора задачи определяется номером прерываемой задачи. Этот номер хранится в переменной *num_tsk*. Преобразование номера задачи в ее адрес и адрес дескриптора реализуется в зависимости от способа задания этих адресов. Адреса дескрипторов задач *d_tskn* (*d_tsk0*, *d_tsk1*, *d_tsk2*) и адреса самих задач (*tsk0*, *tsk* и *tsk0*) могут быть заданы оператором *org addr* или определены в процессе компиляции программы (см. ниже). Для хранения дескрипторов во внешней памяти должны быть зарезервированы области данных, размером равным длине дескриптора. По завершению процедуры сохранения значение R0, равное числу сохраненных параметров sp+1, запоминается в первом байте дескриптора.

Выбор очередной задачи на исполнение.

Порядок выполнения задач в системе определяется последовательностью 0, 1, 2, 0, 1, 2 и т.д. В соответствии с этой последовательностью должен определяться и номер *num_tsk* следующей задачи. Алгоритм определения номера следующей задачи достаточно очевиден. Если номер выполняемой задачи (значение num_tsk) меньше 2, то значение num_tsk увеличивается на 1, в противном случае номеру присваивается значение 0 (*num_tsk*:=0). Номер задачи используется для определения адреса ее дескриптора *d_tskn*.

Дескрипторы имеют размер кратный 2^n (как правило, не меньше 16 байт и не больше 64 байт) и обычно занимают смежные области памяти. Определение адреса дескриптора по номеру задачи рассмотрим на примере. Пусть для дескрипторов всех задач зарезервирована область памяти, задаваемая оператором *org E000h*, и размер дескриптора равен 32 байт. Адреса дескрипторов: для первой задачи *d_tsk0 = E000h*, для второй задачи *d_tsk1 = E020h*, для третьей задачи *d_tsk2 = E040h*. Нетрудно заметить, что адреса дескрипторов задач отличаются только значением младшего байта. Для определения этого значения достаточно сдвинуть содержимое ячейки *num_tsk* с номером задачи на 3 разряда вправо (или 5 разрядов влево). После определения адреса дескриптора задачи содержимое дескриптора можно модифицировать значениями сохраняемого контекста для прерываемой задачи или использовать для восстановления контекста вызываемой задачи.

При инициализации программы, имитирующей работу многозадачной операционной системы, дескрипторы задач *tskn_d* (рис.П6.1) должны содержать контекст, обеспечивающий правильную работу программы при ее запуске. Обязательной информацией в исходном контексте задачи является

длина дескриптора и адрес возврата. В рассматриваемом примере длина дескриптора определяется числом сохраненных параметров и равна 11h (рис.П6.1). Это число необходимо указать в нулевой ячейке дескриптора. Поскольку при первом вызове задачи адресом возврата является адрес самой задачи, то этот адрес должен быть указан в девятой (PC_L) и десятой (PC_H) ячейках дескриптора. В первой ячейке дескриптора должно быть задано значение 1, обеспечивающее корректное выполнение процедуры восстановления контекста. Остальные параметры контекста задачи безразличны, поскольку они инициализируются соответствующей подпрограммой *init* самой задачи.

Пример определения дескрипторов операторами org:

```
        org 9000h
task1:   sjmp task1

        org 9200h
task2:  sjmp task2

        org 9400h
task3:  sjmp task3

        org E000h
tsk1_d: db 11h,1,0,0,0,0,0,0,00,90h,  0,0,0,0,0,0,0,0

        org E020h
tsk2_d: db 11h,1,0,0,0,0,0,0,00h,92h,  0,0,0,0,0,0,0,0

        org E040h
tsk3_d: db 11h,1,0,0,0,0,0,0,  00h,94h,  0,0,0,0,0,0,0,0
```

Пример инициализации дескрипторов, адреса которых определяются в процессе компиляции:

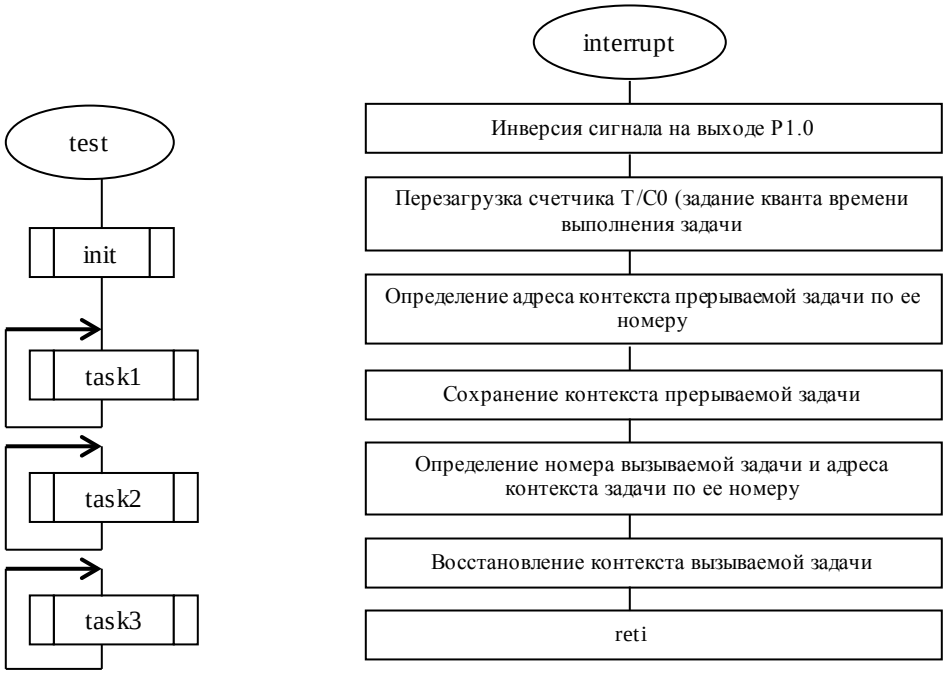
```
;инициализации дескриптора d_tsk2 второй задачи

        mov dptr,#task2         ; загрузка адреса второй задачи
        mov r0,dpl              ; запоминание адреса второй задачи в регистрах r0 (младший байт адреса) и
        mov r1,dph              ; r1 (старший байт адреса)
        mov dptr,#tsk2_d                ; загрузка адреса дескриптора второй задачи

; формирование адреса ячейки дескриптора, в которую заносится адрес второй задачи (адрес возврата)
        mov a,dpl
        add a,#8
        mov dpl,a
; загрузка адреса второй задачи в дескриптор
        mov a,r0
        movx @dptr,a
        inc dptr
        mov a,r1
        movx @dptr,a

; инициализации дескриптора d_tsk3 третьей  задачи выполняется аналогично

task1:  sjmp task1

task2:  sjmp task2

task3:  sjmp task3
```

```
tsk1_d:  db 11h,1,0,0,0,0,0,0,  00,00,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
tsk2_d:  db 11h,1,0,0,0,0,0,0,  00,00,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
tsk3_d:  db 11h,1,0,0,0,0,0,0,  00,00,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Из представленных пояснений видно, что сохранение и восстановление контекста является сравнительно сложной процедурой. Для контроля правильности ее выполнения целесообразно предельно упростить переключаемые задачи, сосредоточив основное внимание на обработчике прерывания таймера, реализующем функции диспетчера при переключении задач. Разработайте и выполните тестовую программу, управляющую циклическим выполнением 3-х простейших задач, каждая из которых представлена только одной циклически выполняемой командой *cpl P1.n,* где *n* – номер разряда порта P1, совпадающий с номером задачи.

Тестовая программа позволяет проконтролировать последовательность переключения выполняемых задач и правильность сохранения / восстановления контекстов задач при их переключении. Алгоритм тестовой программы, иллюстрирующий работу многозадачной ОС и алгоритм обработчика прерывания по переполнению T/C0 (программа диспетчера) показаны на рис.П6.3.

В подпрограмме *init* таймер T/C0 настраивается для циклического формирования кванта времени выполнения отдельной задачи, например, 2-5 мс (см. подразд. 2.6.1). Кроме того, в подпрограмме *init* выполняется инициализация дескрипторов второй и третьей задач. Примеры инициализации дескрипторов рассмотрены выше.



Алгоритм программы обработчика прерывания по переполнению T/C0 (алгоритм диспетчера)

Рис.П6.3. Алгоритм тестовой программы, иллюстрирующей работу многозадачной ОС

Для контроля выполнения переключаемых программ в исходные коды программ необходимо добавить команды инвертирования соответствующих разрядов порта P1.

После верификации тестовой программы разработка основной программы задания П3.7 не должна представлять труда. Задачи (программы реальных процессов) подключаются к ядру ОС операторами include.

2. Генерирование последовательностей псевдослучайных чисел

Последовательности чисел, выбранных случайным образом из некоторого множества, называются последовательностями случайных чисел. Методы генерирования случайных чисел разделяют на аппаратные и программные. Аппаратные генераторы случайных чисел используют внешние источники энтропии. Такие генераторы существенно сложнее программных. Они обладают особыми свойствами, например свойством необратимости.

К программным методам относят различные алгоритмы генерирования последовательностей чисел.

Вычислительные устройства, генерирующие последовательности чисел, всегда работают по заданной программе. Они могут генерировать только детерминированные последовательности, в которых числа повторяются с определенным шагом. В таких последовательностях числа могут быть похожи на случайные, иметь близкое к случайному распределение, но все же являются не случайными, а *псевдослучайными* (ПСЧ).

В большинстве случаев методы формирования псевдослучайных целых чисел базируются на выборе некоторой функции *f*, отображающей множество целых чисел в себя. Выбирается какое-нибудь начальное число x_0, а каждое следующее число порождается с помощью рекуррентного соотношения::

$$x_{k+1} = f(x_k) \ .$$

Основные характеристики формируемой последовательности (длина цикла или количество итераций, после которого ГПСЧ зацикливается, а также, насколько распределение последовательности отклоняется от равномерного) зависят от выбора начальных данных (значений двух первых членов последовательности). При нулевых начальных значениях последовательность вырождается в нулевую.

Наиболее распространенными алгоритмами получения псевдослучайных чисел являются: **метод серединных квадратов, метод серединных произведений, метод перемешивания и линейный конгруэнтный метод.**

Метод серединных квадратов

В *методе «серединных квадратов»* (предложен в 1946 г. Дж. фон Нейманом) очередной элемент последовательности формируется на основе предыдущего путем возведения его в квадрат и выделения средних цифр полученного числа. Для формирования последовательности псевдослучайных чисел данным методом берут произвольное число, состоящее из двоичных цифр с четным числом разрядов 2n, и возводят его в

квадрат. В результате получается число, состоящее из 4n двоичных цифр. Серединные разряды этого числа (от n+1 до 3n) образуют новое псевдослучайное число. Для получения следующих чисел описанная процедура повторяется многократно.

Рассмотрим пример. Пусть число $01011101_2 = 93_{10}$ является начальным числом последовательности псевдослучайных чисел. Возведя его в квадрат, получим значение $0010000111001001_2 = 8649_{10}$, из которого выделим следующее псевдослучайное число: $00011100_2 = 1Ch = 28_{10}$.

Программная реализация метода *«серединных квадратов»* на МК достаточно проста, но если на некотором шаге псевдослучайное значение становится равным 0, то последовательность вырождается.

Метод серединных произведений является модификацией метода Дж. фон Неймана. В соответствии с этим методом выбирается произвольная пара чисел x_0 и x_1, числа перемножаются. Серединные разряды произведения образуют следующее псевдослучайное число x_2. Процесс повторяется для чисел x_1, x_2 с формированием числа x_3. Все последующие случайные числа последовательности вычисляются по данной схеме.

Метод серединных произведений дает меньшее отклонение псевдослучайной последовательности от равномерного распределения. Недостатком, как и в *методе «серединных квадратов»*, является вырождаемость последовательности при формировании «нулевого» числа.

Метод перемешивания

Идея метода состоит в следующем. Пусть в ячейке хранится начальное число R_0. Циклически сдвигая содержимое ячейки влево на 1/4 длины ячейки, получаем новое число R_0*. Точно так же, циклически сдвигая содержимое ячейки R_0 вправо на 1/4 длины ячейки, получаем второе число R_0**. Сумма чисел R_0* и R_0** дает новое случайное число R_1. Далее R_1 заносится в R_0, и вся последовательность операций повторяется.

Линейный конгруэнтный метод

В данном методе итеративные вычисления значений формируемых псевдослучайных чисел x_k выполняются в соответствии с рекуррентным соотношением:

$$x_k = (A * x_{k-1} + C) \bmod m, \text{ где}$$

m –диапазон генерируемых случайных чисел ($m > 0$, например, $m = 2^8 = 256$),
x_k и x_{k-1} – текущий и предыдущий члены последовательности,
x_0 –первый член последовательности ($0 \le x_0 < m$),
A – множитель ($0 \le A < m$),
C – аддитивный параметр ($0 \le C < m$) - некоторая константа
mod m – указатель на использование остатка от деления на m.

Статические свойства, в том числе период последовательности формируемых случайных чисел, определяется выбором параметров рекуррентного соотношения. Рядом авторов показано, что линейная конгруэнтная последовательность, определенная числами m, A, C и x_0, периодична с периодом, не превышающим m, при этом период последовательности равен m, если выполняются следующие условия:

1) наибольший общий делитель параметров C и m равен 1 (числа C и m являются взаимно простыми);

2) значение числа A-1 кратно любому простому числу, являющемуся делителем числа m. Например, если m кратно 4, то и A-1 кратно 4.

Имеется большое число модификаций линейного конгруэнтного метода формирования псевдослучайных чисел. Одной из модификаций является *мультипликативный конгруэнтный метод*, в котором коэффициент C в рекуррентном соотношении равен нулю:

$$x_k = (A * x_{k-1}) \bmod m,$$

При реализации этого метода вычисление значений псевдослучайных чисел упрощаются, однако качество формируемой последовательности случайных чисел ухудшается (генерируемые числа имеют меньший период, чем при $C \neq 0$). Доказано, что при $C = 0$ для получения максимального цикла повторения последовательности, необходимо в качестве значения параметра m выбирать простое число.

Примером реализации *мультипликативного конгруэнтного метода* является ГПСЧ, реализующий следующий алгоритм.

Имеются две 8-битных константы A и B. Первое число формируемой последовательности случайных чисел x_1 является младшим байтом произведения A и B. Все последующие числа определяются в соответствии с формулой:

$$x_{n+1} = A * x_n ,$$

где x_n – предыдущее, а x_{n+1} – формируемое псевдослучайное число. (x_{n+1} принимается равным младшему байту произведения, поскольку последний обладает большей «случайностью» по сравнению со старшим байтом),

Важным преимуществом рассмотренного алгоритма генерации псевдослучайных чисел является его простота: для поиска следующего числа последовательности необходимо знать только предыдущее число (фактически отсутствуют затраты памяти). Недостатком выступает низкая защищенность генерации, поскольку считается, что легко подобрать формулу, с помощью которой можно предсказать следующее число последовательности.

Существуют генераторы ППСЧ, при построении которых используются более сложные математические зависимости, например, «вихрь Мерсенна» и др. По сравнению с рассмотренными они обладают лучшими статистическими характеристиками и длиной псевдослучайной последовательности.

3. Игровой тренажер внимания

Тренажеры внимания широко применяют для тестирования скорости реакции испытуемых: пилотов, водителей, военных, представителей других профессий, для которых важна реакция на факт наступления определенного

события. Не менее существенным является использование тренажеров внимания при обучении «слепой печати».

Один из вариантов алгоритма работы тренажера следующий:

Символы перемещаются слева направо в верхней (нижней) строке модуля ЖКИ. Скорость перемещения символов изменяется в процессе выполнения тестирования: при высоком проценте «правильной» реакции (≥70 %) скорость предъявления символов возрастает, при низком проценте «правильной» реакции (≤ 40 %) скорость снижается. Скорость предъявления символов должна отображаться в нижней (верхней) строке и может изменяться дискретно с шагом символ в 0,1 (или 0,2) сек. Например, нормальное значение скорости – 1 символ в 1 сек, максимальная скорость – 1 символ в 0,1 сек. Минимальная скорость не регламентируется. Для простоты реализации ее можно ограничить величиной 1 символ в 2 сек.

Вариантом усложненной реализации является тестирование в условиях возникновения помех. Помехи представляют собой символы, отличные от контролируемых. Они появляются в строке с информационным символом в случайные моменты времени и в случайных позициях экрана со скоростью, более высокой по сравнению со скоростью перемещения символа. Контролируемые символы, как и помехи, генерируются генератором случайных чисел (ГСЧ). Количество контролируемых символов ограничено количеством кнопок клавиатуры, выделенных для идентификации символов.

Принцип работы тренажера.

При запуске программы ГСЧ формирует контролируемый символ, который высвечивается в начальной области информационной строки (например, в позиции 0 … 8) и с заданной (начальной) скоростью начинает перемещаться по последовательным позициям строки экрана.

Скорость перемещения предъявляемых символов задается с помощью таймера. При переполнении таймера положение символа изменяется на одно знакоместо вправо (при этом на «старом» знакоместе символ стирается и помещается на «новое» знакоместо). Поскольку процедура обновления содержимого экрана является достаточно продолжительной, ее целесообразно выполнять в основной (фоновой) программе. В относительно «коротком» обработчике прерывания по переполнению таймера выполняются только перезагрузка таймера константой, определяющей скорость перемещения символов, и изменяется координата очередного отображаемого символа.

При работе программы тестирования пользователь с помощью соответствующей идентификационной клавиши реагирует на предъявляемую информационную строку. Сканирование клавиатуры (определение нажатой клавиши) выполняется в основном цикле программы. При «правильной» реакции счетчик правильных ответов инкрементируется, «старый» символ исчезает с экрана и предъявляется «новый». При «неправильной» реакции возможны варианты продолжения программы. Например, инкрементируется счетчик неправильных ответов, «старый» символ исчезает с экрана и предъявляется «новый». Другим вариантом является только фиксация

неправильного ответа с возможностью исправления: символ продолжается перемещаться по экрану. При исправлении (нажатии «правильной» клавиши) тренажер продолжает работу.

Если за время перемещения символа по экрану «правильная» клавиша не нажата, по достижению границы строки контролируемый символ исчезает с экрана и предъявляется новый символ, при этом в счетчик неправильных ответов добавляется штрафная сумма, например, 5.

Интерфейс программы.

При вызове программы тренажера на экране ЖКИ может высвечиваться какое-либо приветственное сообщение с приглашением начать работы, например, VNIMANIE. PRESS ANY KEY

Далее при нажатии определенной (или любой) клавиши на экране высвечивается меню, с помощью которого обеспечивается настройка тренажера на заданный режим работы (выбор уровня сложности, скорости перемещения символов, включение помех и пр.).

При нажатии кнопки «Пуск» начинается тестирование.

При нажатии кнопки «Стоп» тестирование заканчивается, при этом на экран ЖКИ выводится сообщение с результатами тестирования:

- время затраченное на тестирование $X_{мин}$ $Y_{сек}$;
- количество правильных и неправильных ответов.

Далее при нажатии любой клавиши высвечивается стартовое сообщение и может быть начато новое тестирование.

Некоторые рекомендации:

Важной функцией программы «Тренажер внимания» является перемещение символа по экрану. Эта функция реализуется в обработчике прерывания по переполнению таймера T_0. Обработчик не только осуществляет перезагрузку таймера константой, определяющей требуемую задержку (скорость перемещения отображаемого символа), но и изменяет координату символа в строке – переменную *pos*.

Само перемещение реализуется в основной (фоновой) программе: если при сравнении текущей координаты символа $pos_т$ с переменной *pos* выявляется несовпадение, необходимо выполнить стирание отображаемого символа в текущей позиции и перезаписать его в новую позицию.

4. Электронный экзаменатор

Электронный экзаменатор – устройство, представляющее на экране модуля ЖКИ контрольные вопросы теста и варианты ответа на них. Контрольные вопросы отображаются в верхней (1-й) строке дисплея ЖКИ, варианты ответа – в нижней строке.

При вызове программы «Электронный экзаменатор» на экране модуля ЖКИ может высвечиваться приветствие - приглашения начать новое тестирование, например, «Press any key to start». После нажатия любой

клавиши запускается тест: в верхнюю строку экрана ЖКИ выводится 1-й вопрос, а в нижнюю – 1-й вариант ответа. Одновременно запускается таймер, подсчитывающий время тестирования.

Для просмотра вариантов ответа обучаемый (тестируемый) может использовать специально выделенные кнопки клавиатуры, например, кнопки одной из строк клавиатуры (1-я кнопка – 1-й вариант ответа, 2-я кнопка – 2-й вариант ответа и т.д.) или использовать специальные кнопки прокрутки вариантов ответа (вверх и вниз по списку вариантов ответа).

Ответ на вопрос вводится при нажатии кнопки с номером выбранного варианта ответа в клавишной строке ответов, при этом выбор варианта ответа может выполняться, независимо от того все ли варианты ответа просмотрены. После ответа на очередной вопрос (автоматически) появляется следующий вопрос и 1-й вариант ответа на него. Количество правильных и неправильных ответов фиксируется.

После завершения тестирования (ответа на последний вопрос) результат тестирования отображается в текстовом сообщении на экране ЖКИ: в первой строке отображается число правильных и неправильных ответов, во второй – время выполнения теста. При нажатии на кнопку «Сброс» осуществляется возврат в исходное состояние программы с отображением приветствия.

Число контрольных вопросов теста и число вариантов ответа на каждый вопрос определяется заданием.

Программа может быть дополнена блоком оценки качества тестирования. Оценка тестирования зависит от числа правильных ответов. Например, «отлично» - при числе правильных ответов больше 80 %, «хорошо» - 60 %, «удовлетворительно» - 50 %, «неудовлетворительно» - меньше 50 % правильных ответов или тест не выполняется в контрольное время.

По окончанию теста возможен просмотр результатов тестирования на экране ЖКИ. Программа может быть расширена более подробным анализом результатов тестирования с выводом номеров вопросов, на которые был выбран неправильный ответ.

Другой реализацией «Электронного экзаменатора» является программа, использующая вкладку «Терминал» оболочки Shell 51 для ввода исходных данных и результата вычисления в предлагаемом для решения выражении.

Некоторые пояснения к программе.

Для контроля знаний по разделу «Вычисление арифметических выражений» предварительно разрабатывается набор программ, иллюстрирующих выполнение арифметических операций.

При запуске программы на экран ЖКИ выводится строка: «Введите вариант задания и исходные данные для вычислений». Ввод осуществляется при нажатии клавиши «послать сообщение» вкладки «Терминал», при этом на экране ЖКИ высвечивается строка: «Исходные данные: A = , B = , C = ». После выполнения вычислений студент вводит ответ в строку «Текстовое сообщение» вкладки и пересылает его в МК. При приеме «ответа» запускается программа вычисления заданного выражения. Результат

выполнения сравнивается с «ответом». Результат тестирования «ОК» или «ERROR» выводится на экран ЖКИ.

5. Многофункциональный генератор инфранизких частот

Многофункциональный генератор инфранизких частот должен формировать аналоговые сигналы заданной формы (синусоидальный, колоколообразный, треугольный, пилообразный, трапециидальный прямоугольный) в заданном диапазоне частот.

Для воспроизведения сигналов сложной формы (требуемой функции времени) широко используются функциональные генераторы, среди которых в настоящее время наиболее высокими метрологическими и эксплуатационными характеристиками обладают функциональные генераторы, построенные на принципе преобразования кода в аналоговый сигнал, известные в литературе как цифро-аналоговые генераторы.

Основным методом приближенного представления сигналов заданной формы является метод аппроксимации. Сигналы заданной формы сравнительно просто аппроксимируются ступенчатой кривой. Генераторы аппроксимационного типа имеют структуру, легко перестраиваемую с одного вида функциональной зависимости на другой, что позволяет реализовывать на их основе многофункциональные генераторы.

Функциональные генераторы аппроксимационного типа воспроизводят на своем выходе аппроксимирующую функцию $f(t,A)$, зависящую в общем случае от фиксированного числа параметров $A = \{\alpha_1, \alpha_2,\ldots, \alpha_N\}$, которую получают в результате решения задачи приближения $f(t,A)$ к $f(t)$.

В основе решения задачи приближения лежит возможность представления заданной функции $f(t)$ со сколь угодно большой точностью в смысле равномерной или среднеквадратичной нормы ошибки в виде обобщенного полинома $f(t,A)$ по системе базисных функций $\{\xi_i(t)\}_{i=0}$

$$f(t,A = \sum \alpha_i \cdot \xi_i(t),$$

где α_i – весовые коэффициенты при базисных функциях.

Основной структурой функционального генератора аппроксимационного типа является цифро-аналоговый генератор, в котором для реализации математических алгоритмов представления функциональных зависимостей используется метод весового суммирования. Наиболее сложным блоком такого генератора является цифро-аналоговый преобразователь, преобразующий выходной цифровой код генератора в выходной аналоговый сигнал.

При формировании инфранизких частот для преобразования кода в напряжения можно использовать ШИМ-генератор с выходным фильтром низких частот. ЦАП, реализованный на базе ШИМ-генератора с выходным ФНЧ, обладает высокими точностными характеристиками, но имеет низкое быстродействие. На выходе ФНЧ (в простейшем случае интегратора) с

некоторой задержкой, связанной с инерционностью RC-цепочки, формируется напряжение требуемого уровня. Максимальная частота ШИМ при 8-разрядном управлении равна $f_0/256$. Для f_0 = 1 МГц частота ШИМ равна ≈ 4 КГц (Δt = 256 мкс). Частота ШИМ является важным параметром генератора аппроксимационного типа, поскольку она также определяет период дискретизации формируемого сигнала заданной формы и частоту формируемых сигналов ($T_{сигн\ мин} = n \cdot \Delta t$), где n – число точек аппроксимации, Δt – период импульсов дискретизации. (Число точек аппроксимации в n раз уменьшает частоту формируемого сигнала). При увеличении частоты формируемого сигнала (уменьшении периода дискретизации и значения частоты ШИМ) на выходе ФНЧ формируется сигнал с нежелательными искажениями формы. Степень искажения определяется видом формируемого сигнала (ФНЧ не пропускает фронты сигнала и наиболее эффективен при формировании «гладких» сигналов).

При числе точек аппроксимации n = 20 и опорной частоте ШИМ-генератора 1 МГц максимальная частота воспроизводимых сигналов заданной формы не превышает 200 Гц. Нижняя частота воспроизводимых сигналов ограничена максимально возможным значением периода импульсов дискретизации Δt. При больших значениях параметра Δt пульсации формируемых сигналов на выходе фильтра нижних частот могут оказаться выше допустимого значения, например, 50 мВ.

8-разрядный код управления скважностью $N_{упр}$, определяющий точность задания выборки, рассчитывается для максимальной частоты дискретизации (максимальной частоты ШИМ $f_{макс}$ = 4 КГц). При уменьшении частоты дискретизации (увеличении Δt) значения кодов $N_{упр}$ должны быть пересчитаны (промасштабированы)

$N_{упр}$ - 256 мкс

$$N_x = N_{упр} \cdot \Delta t / 256$$

N_x - Δt

Для качественного восстановления сигнала на основе его выборки, каждая из выборок должна удерживаться в течение нескольких периодов Δt (время удержания должно быть достаточным для точного интегрирования уровня выборки).

6. Наибольший общий делитель и наименьшее общее кратное

Простым и достаточно эффективным алгоритмом для нахождения наибольшего общего делителя (НОД) пары целых положительных чисел является алгоритм Евклида.

Для заданной пары целых положительных чисел m и n последовательно формируются пары чисел, состоящие из меньшего числа и разности между большим и меньшим числом. Процедура повторяется, пока числа в паре не станут равными. Найденное число и есть НОД исходной пары чисел.

Наименьшее общее кратное (НОК) двух целых чисел m и n есть наименьшее натуральное число, которое делится на m и n без остатка.

НОК двух целых положительных чисел просто вычислить, если известен их НОД. НОК равно произведению этих чисел, деленное на их НОД.

Приложение 7

Порядок подготовки и проведения лабораторных НИР

Подготовка к проведению НИР. Выполнению НИР должна предшествовать **обязательная** домашняя подготовка. В процессе подготовки к НИР по материалам разделов 1–3 данного пособия изучаются принципы орга+низации и особенности функционирования исследуемого МК. Итогом домашней подготовки является макет отчета по НИР, содержащий краткое описание работы исследуемых подсистем МК, структуру информационных связей МК с подключаемыми внешними устройствами (клавиатурой, модулем ЖКИ, датчиками аналоговых сигналов), алгоритмы и программы тестовых и индивидуальных заданий, протокол исследований с таблицами для записи результатов. Наличие макета отчета является основанием для допуска к выполнению исследований в лаборатории.

Проведение НИР. При выполнении работ необходимо соединить разъемы портов МК с разъемами подключаемых устройств в соответствии с разработанной структурой информационных связей. Результаты исследований и данные по отладке программ заносятся в протокол отчета. Записи в отчете должны быть четкими и ясными. Выполняемые программы должны быть снабжены поясняющими комментариями.

Исследования отдельных циклов работ предполагают их выполнение в течение 2-х - 3-х лабораторных занятий. Работа считается выполненной после подписи преподавателем протокола отчета. Единый (окончательно оформленный) отчет по законченному исследованию представляется к проверке и защите не позднее начала работ следующего цикла.

Рекомендуемое содержание отчета.

Отчет должен содержать титульный лист и собственно отчет.

На титульном листе необходимо указать тему НИР, дату ее проведения, фамилию, имя, отчество студента и номер его академической группы. (Образец представлен ниже).

Собственно отчет должен включать следующие разделы:

– цель исследования;
– программу исследований;
– краткое описание работы исследуемых подсистем микроконтроллера;
– схемы информационных связей МК с подключаемыми внешними устройствами (клавиатурой, модулем ЖКИ, датчиками аналоговых сигналов);
– схемы алгоритмов и программы работы по каждому пункту заданий;
– протокол исследований с таблицами результатов и комментариями по каждому пункту программы исследований (графики, полученные средствами оболочки Shell51, осциллограммы зафиксированных процессов);
– анализ полученных результатов на каждом этапе исследования и общие выводы по результатам работы в целом.

Образец титульного листа отчета

Санкт-Петербургский государственный политехнический университет
Институт информационных технологий и управления
Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе
по дисциплине «Микропроцессорные системы»
(Название отчета соответствует теме выполненной работы), например,
«Изучение вычислительных возможностей МК SAB 80C515»

Работу выполнил студент группы № ФИО *подпись*

Работу принял преподаватель ФИО *подпись*

Санкт-Петербург

201_

ПАВЛОВСКИЙ Евгений Григорьевич
ЖВАРИКОВ Владимир Анатольевич
КУЗЬМИН Александр Александрович

ОРГАНИЗАЦИЯ МИКРОКОНТРОЛЛЕРОВ И ОСНОВЫ ПРОЕКТИРОВАНИЯ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

Учебное пособие и методические указания к лабораторному практикуму