МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Физико-технический факультет

Кафедра вычислительной техники и электроники

ИЗУЧЕНИЕ МИКРОКОНТРОЛЛЕРОВ СЕМЕЙСТВА INTEL 8051

Методические указания по выполнению лабораторной работы

Барнаул 2017

Составитель: ст.преподаватель В.В. Белозерских

Рецензент: кандидат физ.-мат. наук, доцент В.В. Пашнев

Представлены методические указания к выполнению лабораторной работы по курсу «Микропроцессорные системы» для студентов направления 230100 «Информатика и вычислительная техника».

Данные методические указания созданы на основании требований программы дисциплины «Микропроцессорные системы».

Тема: Изучение микроконтроллеров семейства Intel 8051.

Цель работы: Изучение архитектуры и организации микроконтроллеров семейства Intel 8051. Получение навыков программирования и тестирования и отладки устройств на базе микроконтроллеров семейства MCS-51 с использованием среды разработки ProView32 и программатора ProAtMic.

Задачи: Изучить архитектуру и организацию микроконтроллеров семейства Intel 8051. Изучить основы программирования микропроцессорных систем на базе микроконтроллера Atmel AT89C51. В соответствии с заданием написать и отладить программы с помощью среды программирования ProView32 и произвести программирование контроллера с помощью программатора ProAtMic. Работоспособность созданного устройства проверить на лабораторной установке.

Теоретические сведения:

Изучение программных средств микроконтроллеров невозможно без предварительного ознакомления с их аппаратными средствами и особенностями. Прежде, чем приступить к программированию, следует четко представлять работу всех периферийных устройств МК, объем и структуру памяти, набор команд и т.п. В связи с этим сначала предлагается ознакомиться с семейством микроконтроллеров Intel 8051 (MCS-51) в целом и МК AT89C51 в частности.

1. Микроконтроллер AT89C51

Среди всех 8-разрядных микроконтроллеров семейство Intel 8051 (MCS-51) является несомненным чемпионом по количеству компаний, выпускающих его модификации. На сегодняшний день существует более 200 модификаций

микроконтроллеров семейства MCS-51, выпускаемых почти 20-ю компаниями. Эти модификации включают в себя кристаллы с широчайшим набором периферии: от простых 20-выводных МК с одним таймером и резидентной памятью программ (РПП) объемом 1 Кбайт до сложнейших 100-выводных кристаллов с 10-разрядными АЦП, массивами таймеров/счетчиков, аппаратными 16-разрядными умножителями и резидентной памятью программ до 64 Кбайт.

Основная область применения 8-разрядных контроллеров - устройства интеллектуального управления промышленной автоматики и бытовой аппаратуры. Большая доля операций управления состоит в преобразовании логической информации. Следовательно, 8-разрядные МК могут с успехом реализовать эти задачи.

Для выполнения лабораторных работ по курсу МПС предлагается использовать микроконтроллер фирмы Atmel AT89C51, который широко применим для подобного рода задач. AT89C51 – высокопроизводительный 8-разрядный микроконтроллер, включающий в себя:

- 8-битное АЛУ, выполняющее арифметические и логические операции, а также операции циклического сдвига, сброса, инвертирования и т.п. Важной особенностью АЛУ является его способность оперировать не только байтами, но и битами. Отдельные программно-доступные биты могут быть установлены, сброшены, инвертированы, переданы, проверены и использованы в логических операциях.

- стираемое программируемое ПЗУ программ (FLASH) емкостью 4 Кбайт, память данных ОЗУ – 128 байт;

- программно доступные регистры, предназначенные для временного хранения операндов;

- универсальный асинхронный приемопередатчик (УАПП), позволяющий осуществлять прием и передачу данных, представленных последовательностью бит;

- четыре программируемых двунаправленных порта ввода/вывода;

- блок двухуровневого векторного прерывания от 6 источников;

- таймеры/счетчики;

Блок-схема внутреннего устройства МК AT89C51 приведена на рис. 1.

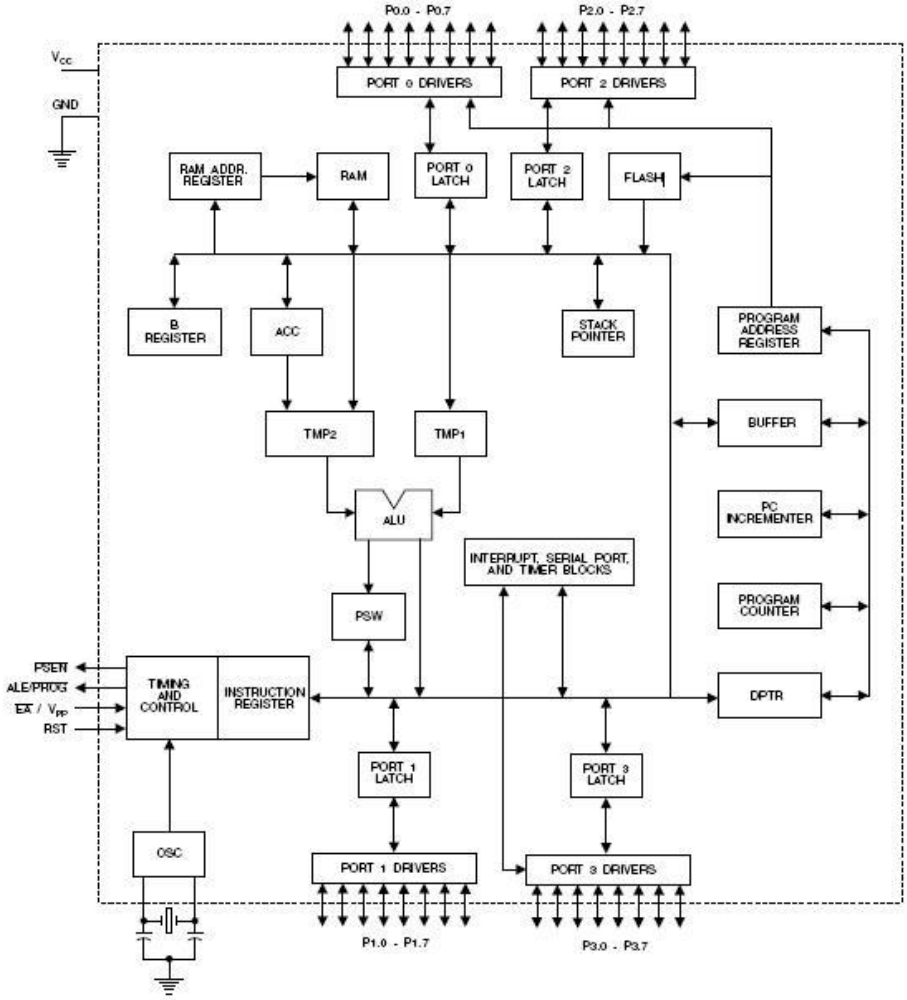

Рис. 1 Блок-схема внутреннего устройства МК AT89C51.

Рассматриваемый МК является типичным микропроцессорным устройством с архитектурой CISC – со стандартным набором команд. Поэтому его система команд довольно обширна и включает в себя 111 основных команд, которые по функциональному признаку можно разделить на пять:

- пересылки данных;
- арифметических операций;
- логических операций;
- операций над битами;
- передачи управления.

Их длина – один, два или три байта, причём большинство из них – одно- или двухбайтные. Все команды выполняются за один или два машинных цикла

(соответственно 1 или 2 мкс при тактовой частоте 12 МГц), исключение составляют команды умножения и деления, которые выполняются за 4 машинных цикла (4 мкс). На основе этих особенностей производится расчет времени исполнения программ.

Набор команд MCS-51 поддерживает следующие режимы адресации: прямая, косвенная, регистровая, непосредственная, индексная, неявная. Список всех команд приведен в таблице.

Список команд МК AT89C51

№	Мнемоника	Циклы	Описание
Операции передачи данных			
1	MOV A, #data	1	(A) <= #data
2	MOV A, addr	1	(A) <= (addr)
3	MOV A, Rn	1	(A) <= (Rn)
4	MOV A, @Ri	1	(A) <= ((Ri))
5	MOV addr, #data	2	(addr) <= #data
6	MOV addr, A	1	(addr) <= (A)
7	MOV addr, Rn	2	(addr) <= (Rn)
8	MOV addr, @Ri	2	(addr) <= ((Ri))
9	MOV addrD, addrS	2	(addrD) <= (addrS)
10	MOV Rn, #data	1	(Rn) <= #data
11	MOV Rn, A	1	(Rn) <= (A)
12	MOV Rn, addr	2	(Rn) <= (addr)
13	MOV @Ri, #data	1	((Ri)) <= #data
14	MOV @Ri, A	1	((Ri)) <= (A)
15	MOV @Ri, addr	2	((Ri)) <= (addr)
16	MOV DPTR, #data16	2	(DPTR) <= #data16
17	MOVC A, @A+DPTR	2	(A) <= ((A) + (DPTR))
18	MOVC A, @A+PC	2	(PC) <= (PC) + 1, (A) <= ((A) + (PC))
19	MOVX A, @Ri	2	(A) <= ((Ri))
20	MOVX A, @DPTR	2	(A) <= ((DPTR))
21	MOVX @Ri, A	2	((Ri)) <= (A)
22	MOVX @DPTR, A	2	((DPTR)) <= (A)
23	PUSH addr	2	(SP) <= (SP) + 1, ((SP)) <= (addr)
24	POP addr	2	(addr) <= ((SP)), (SP) <= (SP) - 1
25	XCH A, addr	1	(A) <=> (addr)
26	XCH A, Rn	1	(A) <=> (Rn)
27	XCH A, @Ri	1	(A) <=> ((Ri))
28	XCHD A, @Ri	1	(A[0-3]) <=> ((Ri)[0-3])
Арифметические операции			
29	ADD A, #data	1	(A) <= (A) + #data
30	ADD A, addr	1	(A) <= (A) + (addr)
31	ADD A, Rn	1	(A) <= (A) + (Rn)
32	ADD A, @Ri	1	(A) <= (A) + ((Ri))
33	ADDC A, addr	1	(A) <= (A) + (addr) + (C)
34	ADDC A, #data	1	(A) < (A) + #data + (C)
35	ADDC A, Rn	1	(A) <= (A) + (Rn) + (C)
36	ADDC A, @Ri	1	(A) <= (A) + ((Ri)) + (C)

37	SUBB A, #data	1	(A) <= (A) - #data - (C)
38	SUBB A, addr	1	(A) <= (A) - (addr) - (C)
39	SUBB A, Rn	1	(A) <= (A) - (Rn) - (C)
40	SUBB A, @Ri	1	(A) <= (A) - ((Ri)) - (C)
41	MUL AB	4	(B).(A) <= (A) * (B)
42	DIV AB	4	(A).(B) <= (A) / (B)
43	INC A	1	(A) <= (A) + 1
44	INC addr	1	(addr) <= (addr) + 1
45	INC Rn	1	(Rn) <= (Rn) + 1
46	INC @Ri	1	((Ri)) <= ((Ri)) + 1
47	INC DPTR	2	(DPTR) <= (DPTR) + 1
48	DEC A	1	(A) <= (A) - 1
49	DEC addr	1	(addr) <= (addr) - 1
50	DEC Rn	1	(Rn) <= (Rn) - 1
51	DEC @Ri	1	((Ri)) <= ((Ri)) - 1
52	DA A	1	десятичная коррекция аккумулятора
Логические операции			
53	ANL A, #data	1	(A) <= (A) & #data
54	ANL A, addr	1	(A) <= (A) & (addr)
55	ANL A , Rn	1	(A) <= (A) & (Rn)
56	ANL A, @Ri	1	(A) <= (A) & ((Ri))
57	ANL addr, #data	2	(addr) <= (addr) & #data
58	ANL addr, A	1	(addr) <= (addr) & (A)
59	ORL A, #data	1	(A) <= (A) OR #data
60	ORL A, addr	1	(A) <= (A) OR (addr)
61	ORL A, Rn	1	(A) <= (A) OR (Rn)
62	ORL A, @Ri	1	(A) <= (A) OR ((Ri))
63	ORL addr, #data	2	(addr) <= (addr) OR #data
64	ORL addr, A	1	(addr) <= (addr) OR (A)
65	XRL A, #data	1	(A) <= (A) XOR #data
66	XRL A, addr	1	(A) <= (A) XOR (addr)
67	XRL A, Rn	1	(A) <= (A) XOR (Rn)
68	XRL A, @Ri	1	(A) <= (A) XOR ((Ri))
69	XRL addr, #data	2	(addr) <= (addr) XOR #data
70	XRL addr, A	1	(addr) <= (addr) XOR (A)
71	CLR A	1	(A) <= 0
72	CPL A	1	(A) <= NOT(A)
73	RL A	1	циклический сдвиг влево
74	RLC A	1	сдвиг влево через перенос
75	RR A	1	циклический сдвиг вправо
76	RRC A	1	сдвиг вправо через перенос
77	SWAP A	1	(A[0-3]) <=> (A[4-7])
Битовые операции			
78	MOV C, bit	1	(C) <= (bit)
79	MOV bit, C	2	(bit) <= (C)
80	ANL C, bit	2	(C) <= (C) & (bit)
81	ANL C, /bit	2	(C) <= (C) & NOT(bit)
82	ORL C, bit	2	(C) <= (C) OR (bit)
83	ORL C, /bit	2	(C) <= (C) OR NOT(bit)
84	CLR C	1	(C) <= 0
85	CLR bit	1	(bit) <= 0
86	SETB C	1	(C) <= 1
87	SETB bit	1	(bit) <= 1
88	CPL C	1	(C) <= NOT(C)
89	CPL bit	1	(bit) <= NOT(bit)
Операции перехода			
90	JMP @A + DPTR	2	безусловный переход по косвенному адресу

91	SJMP rel	2	безусловный переход в пределах страницы 256 байт
92	AJMP addr11	2	безусловный переход в пределах страницы 2 Кб
93	LJMP addr16	2	длинный безусловный переход во всей памяти
94	JZ rel	2	переход, если в аккумуляторе нуль
95	JNZ rel	2	переход, если в аккумуляторе не нуль
96	JC rel	2	переход, если бит переноса установлен
97	JNC rel	2	переход, если бит переноса не установлен
98	JB bit, rel	2	переход, если бит установлен
99	JNB bit, rel	2	переход, если бит не установлен
100	JBC bit, rel	2	переход, если бит установлен со сбросом бита
101	CJNE A, #data, rel	2	сравнение аккум. с константой и переход, если не равно
102	CJNE A, addr, rel	2	сравнение аккум. с байтом и переход, если не равно
103	CJNE Rn, #data, rel	2	сравнение рег. с константой и переход, если не равно
104	CJNE @Ri, #data, rel	2	сравнен. байта памяти с конст. и переход, если не равно
105	DJNZ Rn, rel	2	декримент регистра и переход, если не нуль
106	DJNZ addr, rel	2	декримент значения по адресу и переход, если не нуль
107	ACALL addr11	2	вызов подпрограммы в пределах страницы 2 Кб
108	LCALL addr16	2	длинный вызов подпрограммы во всей памяти
109	RET	2	возврат из подпрограммы
110	RETI	2	возврат из подпрограммы обработки прерывания
111	NOP	1	пустая операция

Rn: n = 0, …, 7
@Ri: i = 0, 1

bit: программно доступный бит

#data: 8-ми разрядные данные
#data16: 16-ти разрядные данные
addr1, **addr2**, **addr3**: адрес или имя ячейки памяти
rel: относительный адрес перехода

Контрольные вопросы:

1. Опишите архитектуру микроконтроллера AT89C51.
2. Какие дополнительные функции имеют порты ввода-вывода МК?
3. Как по функциональным признакам можно разделить команды МК AT89C51?
4. Какие функциональные блоки входят в состав микроконтроллера AT89C51?
5. Как осуществляется вызов подпрограмм и выход из них?
6. Как устроен контроллер прерываний? Что такое вектор прерывания?
7. Какие микроконтроллеры семейства Intel 8051 (MCS-51) Вы еще знаете? В чем их особенности?

2. Среда программирования ProView32

Чтобы научиться программировать микропроцессорные системы на базе микроконтроллеров семейства MSC-51, необходимо знать архитектуру и систему команд МК, а также изучить среду программирования и язык программирования низкого или высокого уровня.

Для написания управляющих программ для микроконтроллера используется среда программирования ProView32 (Franklin Software Inc.). ProView32 – интегрированная среда разработки программного обеспечения для однокристальных микроконтроллеров семейства Intel 8051 и его клонов. Она включает в себя всё, что нужно для создания, редактирования, компиляции, трансляции, компоновки, загрузки и отладки программ: стандартный интерфейс Windows; полнофункциональный редактор исходных текстов с выделением синтаксических элементов цветом; организатор проекта; транслятор языка С; ассемблер; отладчик; встроенную справочную систему.

Первый этап разработки программы – запись ее исходного текста на каком-либо языке программирования (в данной лабораторной работе – язык ассемблера). Затем производится компиляция или трансляция его в коды из системы команд микроконтроллера, используя транслятор или ассемблер. Трансляторы и ассемблеры – прикладные программы, которые интерпретируют текстовый файл, содержащий исходный текст программы, и создают объектные файлы, содержащие объектный код.

После компоновки объектных модулей наступает этап отладки программы, устранения ошибок, оптимизации и тестирования программы.

ProView объединяет все этапы разработки программы в единый рекурсивный процесс, когда в любой момент времени возможен быстрый возврат к любому предыдущему этапу.

Рассмотрим основные возможности ProView на примере работы программы Sample, предлагающейся фирмой Franklin Software и находящейся в каталоге Examples в директории установки программы ProView.

1. Запуск ProView и создание файла проекта.

Запуск программы ProView32 осуществляется из меню *Пуск -> Программы -> Franklin Software -> ProView32*.

Любая новая работа в ProView как и во всех современных компиляторах, начинается с создания нового файла проекта. Файл проекта содержит имена всех исходных файлов, связанных с проектом, а также установки компиляции, трансляции и связывания файлов, чтобы генерировать выполняемую программу.

Для того чтобы создать новый проект, необходимо выбрать *New* из меню *Project*. Откроется диалоговое окно *New Project*. Здесь требуется ввести название проекта и путь к нему в поле *Name* (можно использовать для этого кнопку *Browse*). Необходимо выбрать 8051 как тип проекта.

Когда менеджер проекта открывает файл проекта, окно проекта показывает включенные исходные файлы. В данном случае пока нет никаких исходных файлов.

2. Добавление файла с исходным текстом и его редактирование.

Чтобы подключить файлы к проекту, нужно выбрать *Add file* из меню *Project*. Необходимо подключить к проекту файлы sample1.a51, sample2.a51 и sample3.a51 из каталога FSI\EXAMPLES\8051\A51 (рис. 2).

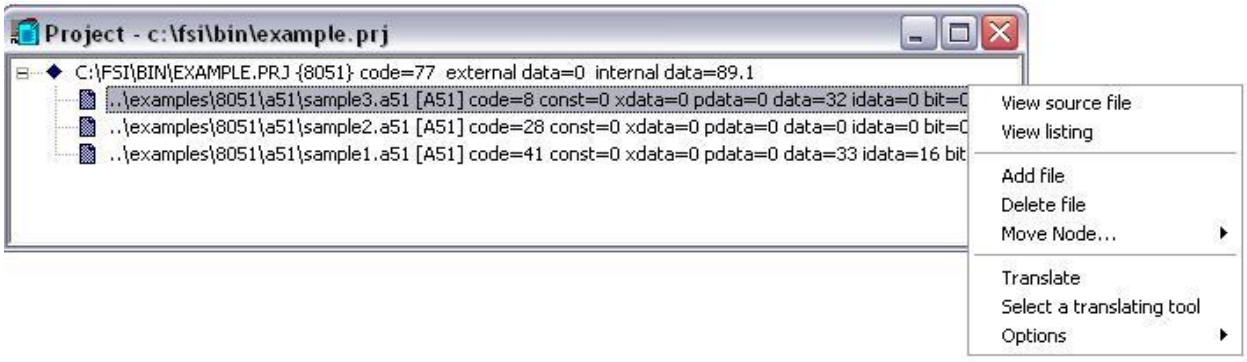

Рис. 2 Окно *Project*.

Теперь можно редактировать текст из файлов sample*.a51. Чтобы просмотреть файл в окне редактирования нужно дважды щелкнуть по его названию в окне *Project* или выбрать пункт *View source file* в контекстном меню, появляющемся при нажатии правой кнопки мыши.

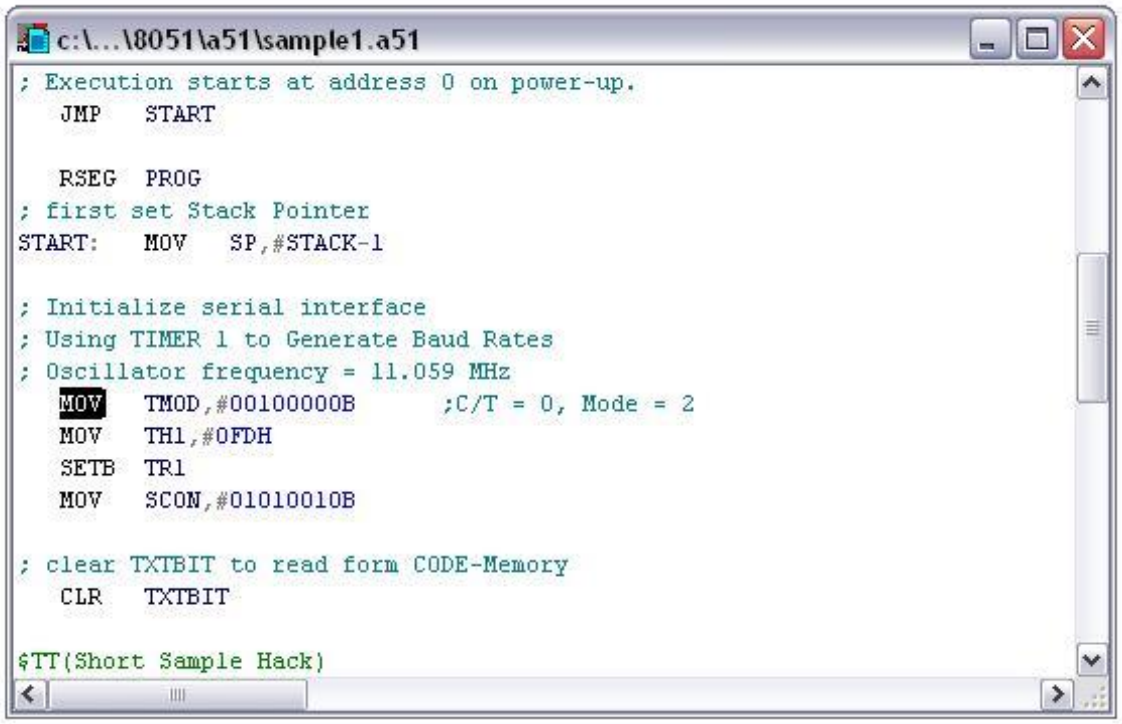

Рис. 3 Окно редактирования.

ProView загружает и показывает содержимое файла в окне редактирования. Окно редактирования (рис. 3) – полнофункциональный редактор исходного текста, предлагающий такие возможности, как высвечивание синтаксических элементов и контекстный поиск. Если выбрать «MOV» и нажать клавиши Ctrl+F1, ProView откроет систему справки и перейдет к разделу справки о «MOV».

3. Компиляция и компоновка.

Этот процесс компилирует, связывает исходные файлы с библиотеками и создает абсолютный объектный модуль, который можно проверить в отладчике.

Далее необходимо выбрать *Make* из меню *Project*. В окне Message (рис. 4) отображается сообщение об окончании компиляции или об обнаруженных в ходе компиляции ошибках.

Рис. 4 Окно сообщений.

4. Тестирование и отладка.

Чтобы запустить отладчик программы требуется выбрать пункт *Star*t из меню *Debug*. Если проект новый появится диалоговое окно *Debug Options* (рис. 5).

В дальнейшем можно установить опции отладчика выбрав *Debug* из меню *Options*. Здесь нужно выбрать тип микроконтроллера 8051, а все остальные настройки оставите включенными по умолчанию.

Теперь можно запустить отладчик программы (меню *Debug -> Start* или нажать кнопки Ctrl+D на клавиатуре). Если в режиме отладки выбрать *Hardware* (аппаратные средства) из меню *View*, а затем – *UART*, то откроется диалоговое окно последовательного порта. В ходе выполнения программы здесь можно будет увидеть все, что выводит микроконтроллер в последовательный порт.

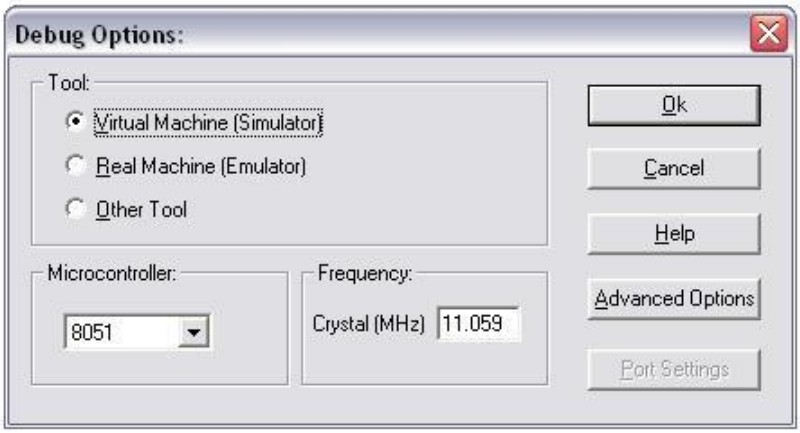

Рис. 5 Опции отладчика.

Запустить отладку программы можно, выбрав *Debug* -> *Run* или нажав кнопку *GO* на главной панели инструментов. Следует обратить внимание на то, что в окно *UART* выведен текст «TEST PROGRAM» (рис. 6).

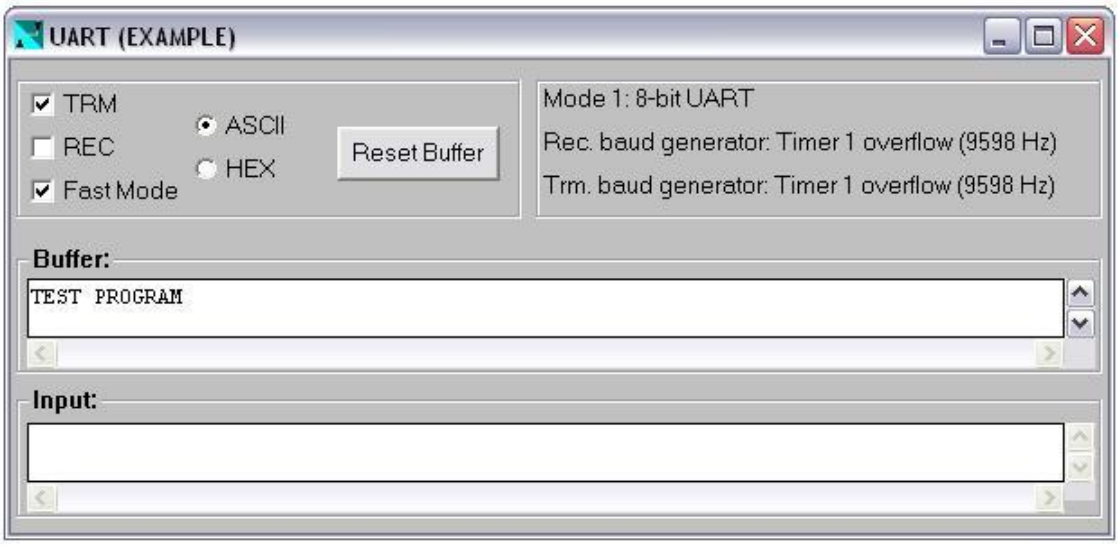

Рис. 6 Окно последовательного порта.

С помощью кнопки *Reset* можно сбросить в начало процесс отладки программы, а с помощью линейного регулятора при нажатой кнопке *Animate* – изменять скорость работы отладчика.

5. Пошаговый режим и выход из отладчика.

Можно использовать отладчик, чтобы перемещаться по программе в пошаговом режиме. Для этого необходимо выбрать *Reset* из меню *Debug* (эта команда сбросит моделируемый процесс), а затем *Step Into* или *Step Over* из того же меню *Debug*. Команды *Step* позволяют «шагать» по каждой строке исходного текста. Текущая команда высвечивается на каждом шаге. *Step Into* позволяет войти в вызываемую функцию (подпрограмму), *Step Over* – перешагнуть через нее не входя во внутрь.

Для завершения работы в отладчике в любой момент времени можно выбрать *Terminate* из меню *Debug* и вернуться в режим редактирования.

6. Вспомогательные окна.

Стоит обратить внимание, что в режиме отладки на экране видны еще два окна. Первое – окно кода (рис. 7), где в пошаговом режиме параллельно с исходным текстом идет трассировка текста на ассемблере.

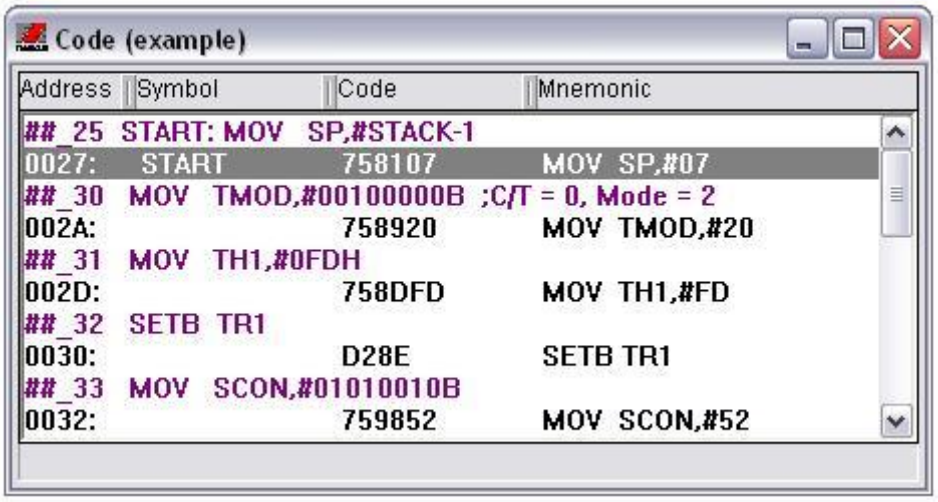

Рис. 7 Окно кода.

Второе окно, которое присутствует на экране во время отладки, – *Main Registers* (рис 8).

В этом окне постоянно отображается текущее состояние всех программно-доступных регистров микроконтроллера. Более того, содержимое регистров можно менять во время отладки.

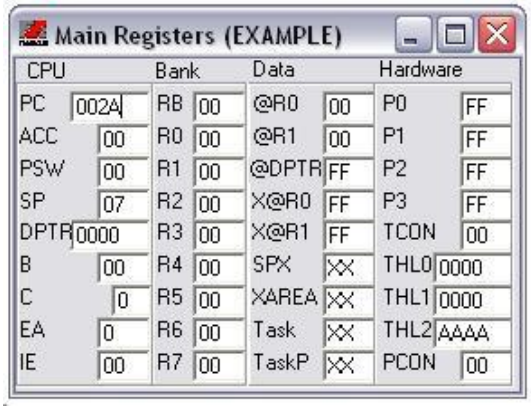

Рис. 8 Окно регистров.

Необходимо очень внимательно изучить интерфейс отладчика. Меню *View* позволяет наглядно представить все аппаратные средства микроконтрол-

лера (состояние таймеров, УАПП, портов ввода-вывода), состояние ОЗУ, внешней памяти и т.д. Отладчик позволяет быстро и эффективно производить тестирование программы и предоставляет для этого все необходимые средства.

Контрольные вопросы:

1. Для чего нужно создавать файл проекта в среде разработки ProView?
2. В чем заключается различие между директивами компилятора CSEG, DSEG, ISEG, XSEG, RSEG?
3. Что такое директива ORG, и какие ограничения возникают при ее использовании?
4. К чему приводит выход из подпрограммы командами LJMP, SJMP и т.д.?
5. По умолчанию указатель стека указывает на адрес 07H. Как его переместить на адрес 18H?
6. Как в отладчике ProView просмотреть состояние внутренней и внешней памяти микроконтроллера, портов, таймеров, узнать время выполнения программы?
7. Как с помощью отладчика ProView запустить на исполнение только часть программы?
8. Какие среды разработки и отладки программного обеспечения для микроконтроллеров семейства Intel 8051 (MCS-51) Вы еще знаете?

3. Пример выполнения задания с помощью АРМС

Пройдем все этапы разработки программного обеспечения на автоматизированном рабочем месте студента (АРМС).

Задача: Реализовать устройство «Электронный переключатель». В состав устройства входят кнопка и два индикатора (светодиоды), находящиеся на линиях портов микроконтроллера. В любой момент времени горит только лишь один светодиод. По нажатию кнопки происходит переключение индикаторов.

Для решения задачи необходимо использовать все средства АРМС: редактор ProView для написания и правки текста программы, отладчик для проверки работоспособности программы, программатор для программирования микроконтроллера, т.е. занесения программы в его память, отладочную плату для окончательной проверки программы.

Отладочная плата:

Отладочная плата (рис. 9) используется для тестирования написанной программы и реализации конечного устройства.

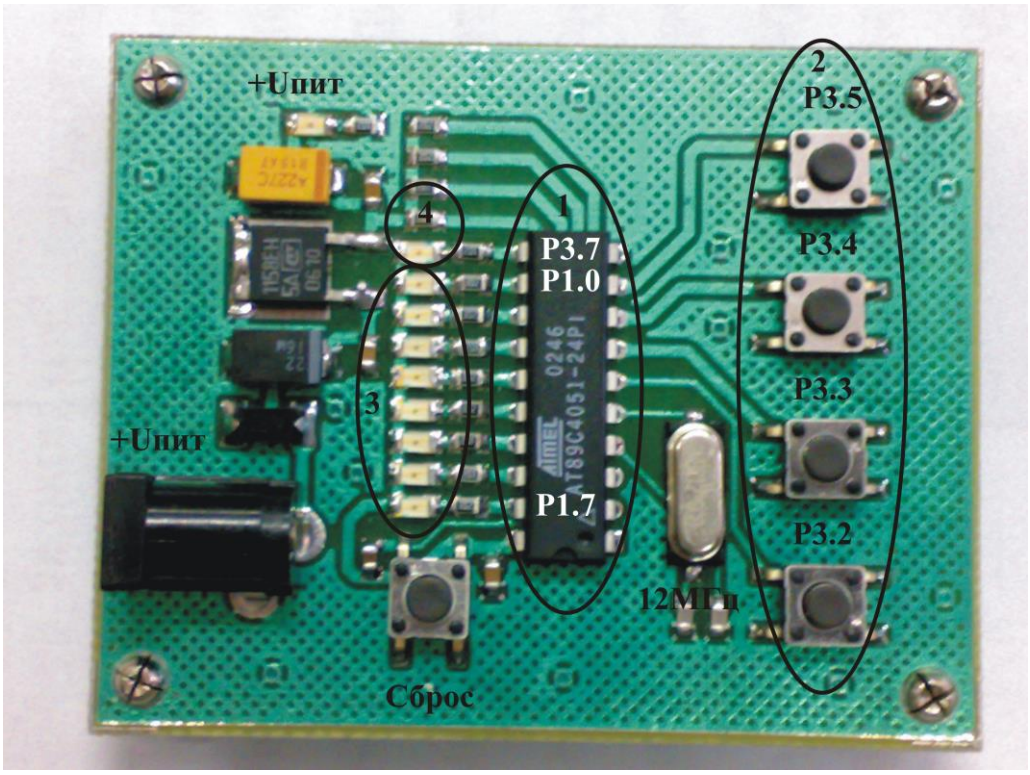

Рис.9 Отладочная плата.

Отладочная плата включает в себя:

– микроконтроллер AT89C4051 (1), который имеет 2 порта ввода-вывода P1 и P3. Ниже приведены блок-схема внутреннего устройства микроконтроллера AT89C4051 и конфигурация выводов.

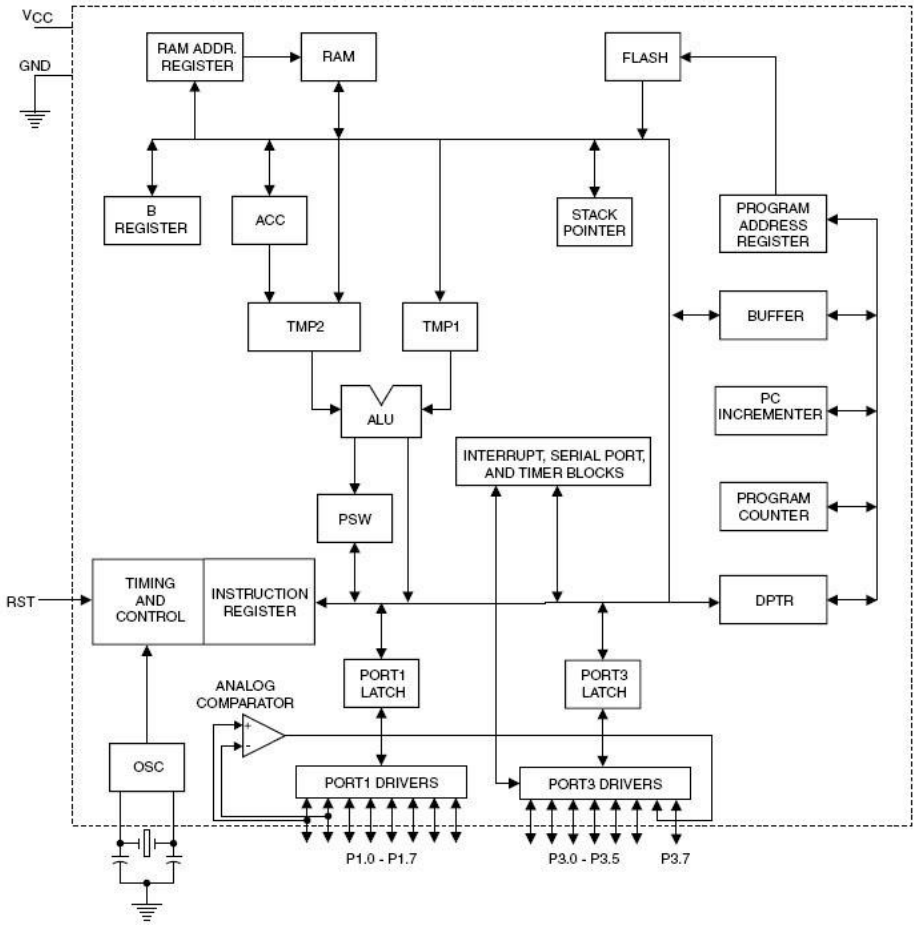

Рис. 10 Блок-схема внутреннего устройства МК AT89C4051.

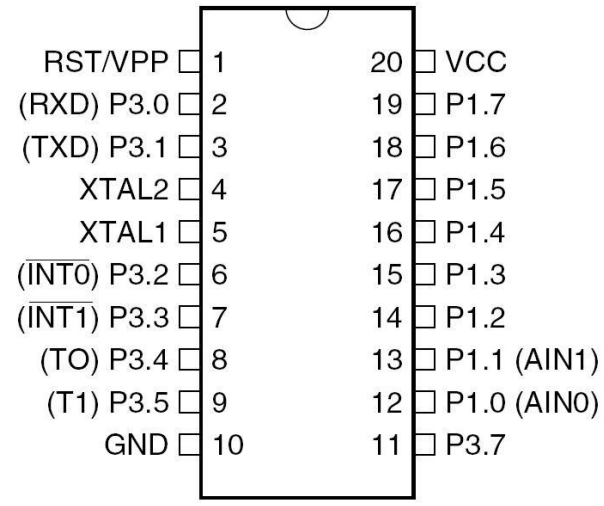

Рис. 11 Конфигурация выводов МК AT89C4051.

– тактовые кнопки (2), находящиеся на линиях P3.2 – P3.5 порта P3. Кнопки с «дребезгом» (длительность «дребезга» достигает порядка 5 мс), активный уровень – логический нуль. Отдельно расположена кнопка «Сброс».

– 8 красных светодиодов (3), подключенных к линиям порта P1 и зажигающихся при достижении уровня логического нуля. К линии P3.7 подключен зеленый светодиод (4), который также зажигается логическим нулем.

– кварцевый резонатор, частота которого составляет 12 МГц.

Пусть кнопка – переключатель индикаторов находится на линии порта микроконтроллера P3.2, светодиоды – линиях P1.7 и P1.6. В любой момент времени горит только один светодиод.

Особенностью здесь является то, что при работе МК с датчиками, имеющими механические или электромеханические контакты (кнопки, клавиши, реле и клавиатуры), возникает явление, называемое дребезгом. Это явление заключается в том, что при замыкании контактов возможно появление отскока (BOUNCE) контактов, которое приводит к переходному процессу. При этом сигнал с контакта может быть прочитан МК как случайная последовательность нулей и единиц. Подавить это нежелательное явление можно схемотехническими средствами с использованием буферного триггера, но чаще это делается программным путем.

Наибольшее распространение получили два программных способа ожидания установившегося значения:

1) подсчет заданного числа совпадающих значений сигнала;

2) временная задержка.

Будем использовать в программе первый способ.

На данном этапе разработки поставлена задача, определены проблемы, которые могут возникнуть в ходе работы устройства, предложены и выбраны способы их устранения. Теперь требуется реализовать блок-схему алгоритма (БСА) работы программы.

На языке схем алгоритмов разработчик обычно описывает метод, выбранный для решения поставленной задачи. Довольно часто бывает, что одна и та же задача может быть решена различными методами. Способ решения задачи, выбранный на этапе ее инженерной интерпретации, на основе которого формируется БСА, определяет не только качество разрабатываемой прикладной программы, но и качественные показатели конечного изделия.

Блок-схема работы программы «Электронный переключатель» приведена на рис. 12.

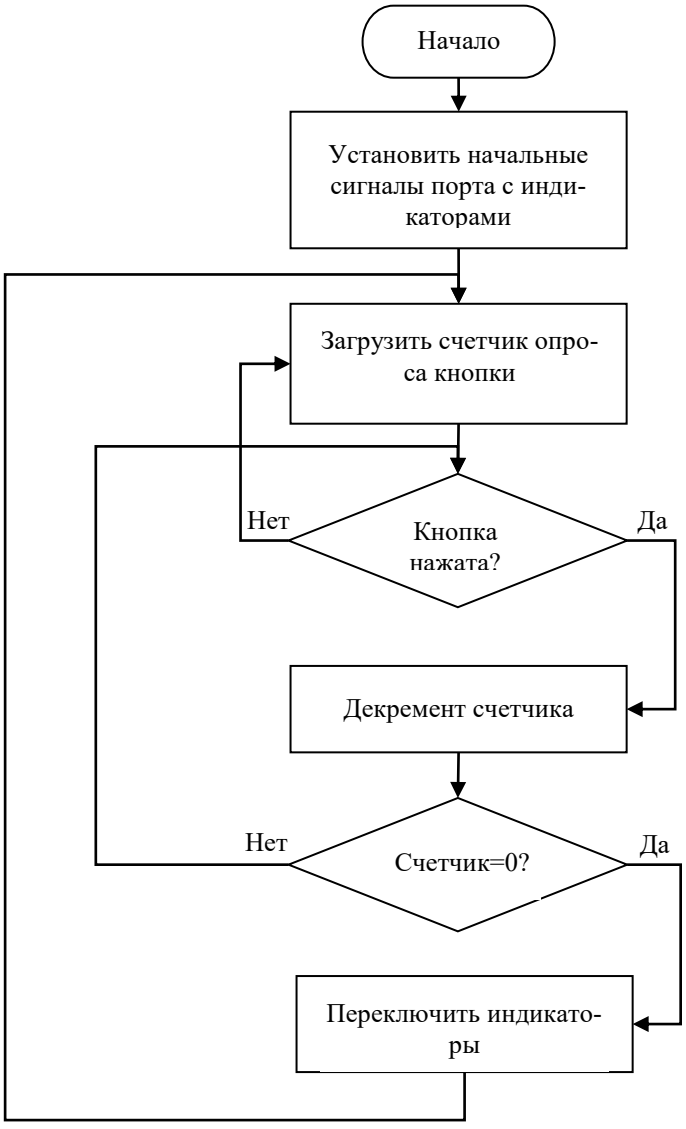

Рис. 12 Блок-схема программы «Электронный переключатель».

После того, как разработана блок-схема программы, можно приступить к написанию и отладке кода программы. Получаем следующий код:

```
        NAME SWITCH                         ; название программы

        MAIN SEGMENT CODE                   ; объявление сегмента памяти программ
        CSEG AT 0                           ; определение абсолютного сегмента памяти с адре-
сом 0h
        LJMP start                          ; переход на начало программы
        USING 0                             ; использование 0 банк регистров
        RSEG MAIN                           ; выбор сегмента памяти MAIN
        start:  MOV P1,#07FH                ; зажечь только светодиод P1.7
        dbnc:
                MOV R4,#5                   ; запись старш. байта счетчика проверки кнопки
        dbnc1:  MOV R3,#0FFH                ; запись младш. байта счетчика проверки кнопки
        dbnc2:
                JB P3.2, dbnc               ; если кнопка не нажата – в начало
                DJNZ R3, dbnc2              ; декремент мл.байта счетчика, если 0 – дальше
                DJNZ R4, dbnc1              ; декремент ст.байта счетчика
                JNB P3.2,$                  ; проверка кнопки на прекращение нажатия

                JB P1.7, switch             ; проверка светодиода (если не горит – на switch)
                SETB P1.7                   ; погасить P1.7
                CLR P1.6                    ; зажечь P1.6
                JMP dbnc                    ; перейти к опросу кнопки

        switch: CLR P1.7                    ; зажечь P1.7
                SETB P1.6                   ; погасить P1.6
                JMP dbnc                    ; перейти к опросу кнопки
        END                                 ; конец
```

Перед этапом компиляции и отладки программы необходимо в опциях отладчика установить частоту *12MHz* в поле *Frequency*, а в диалоговом окне *Options -> Project -> пункт L51 -> Linker ->* установить флажок *Intel Hex* для создания hex-файла (рис. 13). Этот файл будет использоваться для программирования микроконтроллера с помощью программатора.

На рис. 14 изображен процесс отладки программы «Электронный переключатель». В процессе отладки можно сымитировать нажатие кнопки, наблюдать за состоянием индикаторов, оценить время опроса кнопки и т.п.

Теперь необходимо записать полученную программу (hex-файл) в память микроконтроллера с помощью программатора. Правила работы с программатором приведены в приложении.

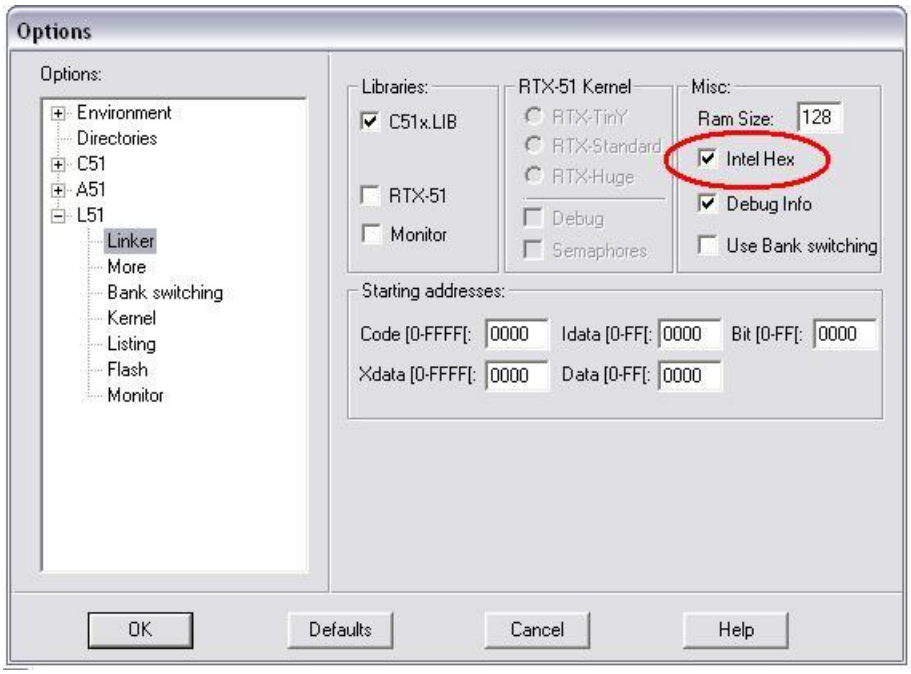

Рис. 13 Настройка проекта на создание hex-файла.

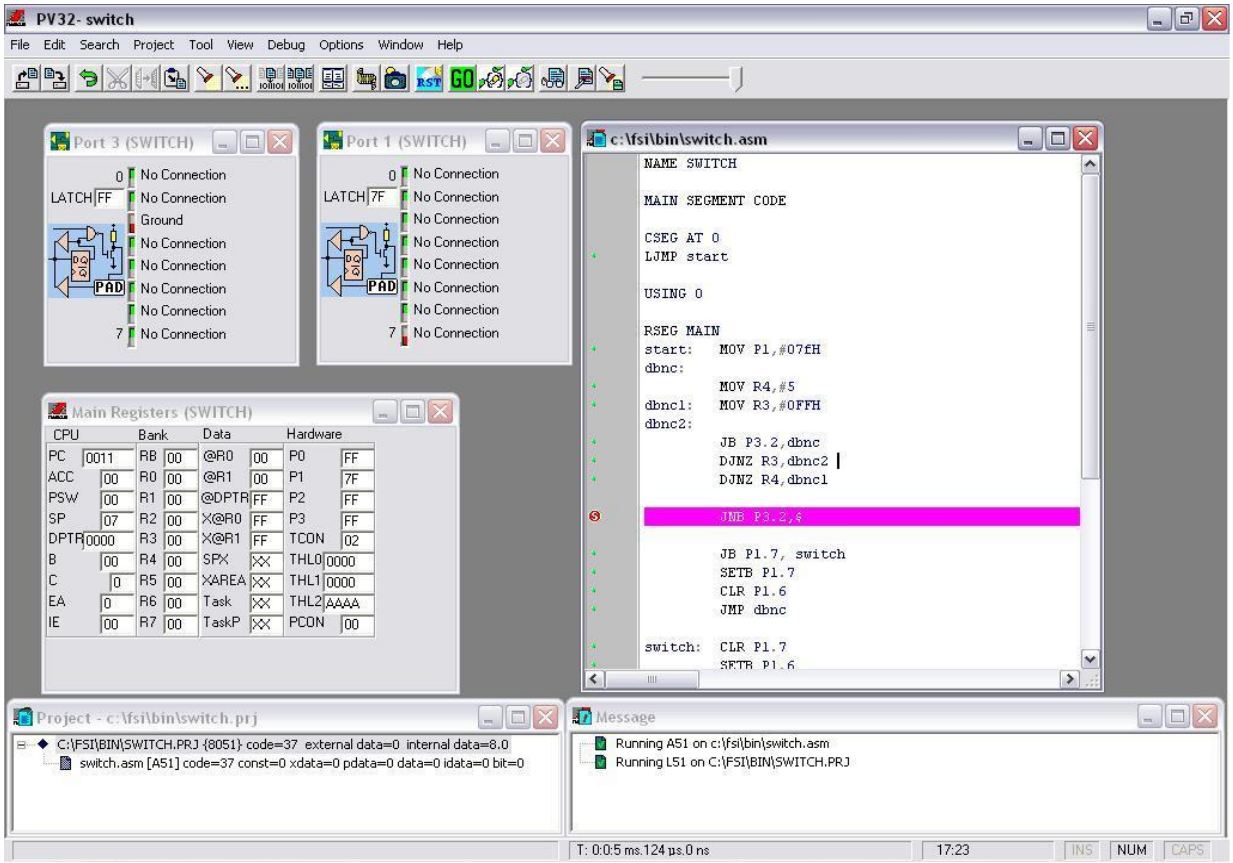

Рис. 14 Отладка программы «Электронный переключатель».

Когда микроконтроллер запрограммирован, можно вставить его в специальный разъем на отладочной плате, подключить отладочную плату к сети питания и убедиться в корректности работы программы, используя кнопки (элементы управления) и светодиоды (элементы индикации).

Контрольные вопросы:

1. Опишите все средства, входящие в состав АРМС.
2. Расскажите об устройстве отладочной платы (микроконтроллер, на каких линиях находятся светодиоды и кнопки).
3. Какова частота кварцевого резонатора, входящего в состав макетной платы? Какова длительность машинного цикла МК при такой частоте?
4. Как настроить проект ProView на создание Hex-файла? Зачем это нужно?
5. Расскажите правила работы с программатором.

Литература:

1. В. Я. Хартов Микропроцессорные системы: учеб. пособие для вузов. – М.: Академия, 2010. – 352 с.
2. Андреев Д.В. Программирование микроконтроллеров MCS-51: Учебное пособие. – Ульяновск: УлГТУ, 2000. – 88 с.
3. Сташин В. В., Урусов А. В., Мологонцева О. Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах. – М.: Энергоатомиздат, 1990. – 218 с.

Задания:

1. Написать программу сложения числа, находящегося во второй половине внутренней памяти данных МК, с числом 5. Результат поместить во внешнюю память.
2. Написать программу вычитания из числа, находящегося во внешней памяти данных МК, числа 5. Результат поместить во внутреннюю память данных, используя косвенную адресацию.
3. Написать программу умножения двух трехбайтных чисел. Первое число находится во внешней памяти, второе число принимается побайтно из порта 2 микроконтроллера. Результата умножения вывести в последовательный порт (UART). Скорость и режим работы UART задается преподавателем. Обязательное использование подпрограмм и символических переменных в отдельном файле проекта.
4. Реализовать задержку программным способом, используя вызов подпрограммы.
5. Реализовать задержку аппаратным способом, используя таймер T1 или T0 по выбору преподавателя. Обязательно использование прерываний.

В заданиях 4 и 5 частота работы МК равна 12 МГц. Точность длительности задержки должна быть выше, чем 0.01%.

6. Написать программу «Бегущие огни». Программа «Бегущие огни» должна выводить световые эффекты (не менее 16) с помощью 8 красных светодиодов (см. рис. 9). Эффекты переключаются двумя кнопками. При достижении максимального или минимального номера эффекта дважды по 0.5с. должен засветиться зеленый светодиод. Другая пара кнопок должна регулировать скорость (не менее 16 скоростей) смены кадров в эффектах. При достижении минимальной или максимальной скорости должен засветиться зеленый светодиод на 1с. Если в течении 20 секунд не нажата ни одна кнопка, программа автоматически должна переключать эффекты в сторону увеличения по циклу с выдержкой каждого эффекта не менее 30 секунд.

Проект ProView необходимо настроить на создание файла *.hex во время компиляции (*Options -> Project -> пункт L51 -> Linker ->* установить флажок *Intel Hex*). Файл *.hex, содержащий код программы необходимо записать в память программ МК с помощью программатора. Далее программа тестируется на макетной плате.

Требования к защите работы:

1. Демонстрация написанных и полностью отлаженных программ на компьютере (пример и задание 6 – на макетной плате).
2. Отчет по лабораторной работе, оформленный в соответствии с правилами.
3. Готовность отвечать на вопросы преподавателя по теме работы.

Правила оформления отчета к работе:

Отчет, является документом, отражающим результаты и ход выполнения лабораторной работы. Отчет должен содержать следующие пункты:

1. Титульный лист, содержащий тему и номер лабораторной работы, фамилии выполнявшего студента и проверявшего преподавателя.
2. Цель работы – указываются цели, которые необходимо достигнуть в результате выполнения лабораторной работы.
3. Задачи – указываются задачи, решаемые в ходе выполнения лабораторной работы.
4. Выполнение работы – указывается последовательность действий или алгоритм работы программы, приводятся выполненные расчеты.
5. Результаты работы – приводятся алгоритм, блок-схема алгоритма работы программы и текст работы программы с комментариями на русском языке.
6. Вывод по каждому пункту, в соответствии с заданием.

6. Общий вывод по работе в соответствии с заявленными целями. – пункт содержит перечень решенных в ходе работы задач и выполненных целей, а также выводы по достигнутым результатам.

Обратить внимание:

1. Отчет принимается только в печатном виде на бумажном носителе. Т.е. отчет должен быть полностью набран на компьютере и распечатан. Отчеты в электронном виде рассматриваться не будут.
2. Листы отчета должны быть пронумерованы, скреплены между собой и помещены в папку.

ПРИЛОЖЕНИЕ

Использование программатора ProAtMic

1. Подключить программатор к сети питания. Подключить программатор к COM-порту компьютера.

2. Запустить на компьютере программу ProAtMic.

3. Выбрать последовательный порт, к которому подключен программатор.

4. Вставить программируемый 20-выводный микроконтроллер в специальный разъем.

5. Выбрать тип программируемого микроконтроллера.

6. Произвести процесс стирания (кнопка «Стереть») микроконтроллера, или ввести имя файла и произвести процесс чтения (кнопка «Чтение») памяти программ микроконтроллера в указанный файл (расширение *.bin).

7. С помощью кнопки «Программирование» можно запрограммировать микроконтроллер (необходимо обязательно произвести операцию «Стирание» перед программированием!), предварительно выбрав записываемый шестнадцатеричный (*.hex) или двоичный (*.bin) файл. Процесс программирования и проверки можно проследить по индикатору состояния и текстовому окну поля «Статус».

8. Вынуть микроконтроллер из разъема. При необходимости повторить шаги 4-7 (вставить следующий микроконтроллер для записи / чтения / стирания памяти программ).

9. Отключите программатор от сети питания.