Министерство общего и профессионального образования
Российской Федерации
Уральский государственный технический университет

ВАША ПЕРВАЯ ПРОГРАММА ДЛЯ МИКРОКОНТРОЛЛЕРА INTEL 8051

Методические указания к лабораторной работе №1
по курсу “Цифровые устройства и микропроцессоры”
для студентов всех форм обучения специальностей
200700 – Радиотехника;
201500 – Бытовая радиоэлектронная аппаратура

Екатеринбург 1999

УДК 681.322

Составитель В.А.Добряк

Научный редактор доц., канд. техн. наук В.И.Елфимов

ВАША ПЕРВАЯ ПРОГРАММА ДЛЯ МИКРОКОНТРОЛЛЕРА INTEL 8051: Методические указания к лабораторной работе №1 по курсу “Цифровые устройства и микропроцессоры”/ В.А.Добряк. Екатеринбург: Изд-во УГТУ, 1999. 32 с.

Методические указания предназначены для использования при выполнении лабораторного практикума. Содержат описание архитектуры микроконтроллера семейства Intel 8051, введение в интегрированную среду разработки программ ProView для этого семейства, пояснения к разработке учебной программы и задания для самостоятельной работы.

Библиогр.: 4 назв. Рис. 15. Табл. 13.

Подготовлено кафедрой радиоприёмных устройств.

ОГЛАВЛЕНИЕ

1. ЦЕЛЬ И СОДЕРЖАНИЕ РАБОТЫ 4
2. ЗАДАНИЯ ДЛЯ ДОМАШНЕЙ ПОДГОТОВКИ 4
2.1. Изучите особенности однокристальных микроконтроллеров 4
2.2. Изучите архитектуру микроконтроллеров семейства Intel 8051 4
2.3. Изучите систему команд микроконтроллеров 4
2.4. Изучите раздел “Быстрый старт” 4
2.5. Контрольные вопросы 5
3. АРХИТЕКТУРА МИКРОКОНТРОЛЛЕРА КМ1816ВЕ51 6
3.1. Арифметико-логическое устройство 6
3.2. Резидентная память программ и данных 8
3.3. Аккумулятор, регистры общего назначения и флаги 9
3.4. Регистры-указатели 10
3.5. Регистры специальных функций 10
3.6. Устройство управления и синхронизации 10
3.7. Параллельные порты ввода/вывода информации 11
3.8. Таймер/счётчик 12
3.9. Последовательный порт 15
3.9.1. Регистр SBUF 15
3.9.2. Регистр SCON 15
3.9.3. Работа UART в мультиконтроллерных системах 17
3.9.4. Скорость приёма/передачи 17
3.10. Система прерываний 18
4. ВВЕДЕНИЕ В PROVIEW 21
4.1. Оптимизирующий кросс-компилятор C51 21
4.2. Макроассемблер A51 22
4.3. Компоновщик L51 22
4.4. Отладчик/симулятор WinSim51 22
5. БЫСТРЫЙ СТАРТ 23
5.1. Запуск ProView и создание файла проекта 23
5.2. Добавка файла с исходным текстом и его редактирование 25
5.3. Компиляция и компоновка 26
5.4. Тестирование и отладка 26
5.5. Пошаговый режим и выход из отладчика 28
5.6. Следующий шаг 29
6. ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ 31
7. СОДЕРЖАНИЕ ОТЧЁТА 31
БИБЛИОГРАФИЧЕСКИЙ СПИСОК 31

1. ЦЕЛЬ И СОДЕРЖАНИЕ РАБОТЫ

Целью работы является изучение архитектуры микроконтроллеров популярного семейства 8051 Intel, а также интегрированной среды ProView фирмы Franklin Software Inc., предназначенной для разработки программного обеспечения для этого семейства. Работа рассчитана на 4 часа домашней подготовки и 4 часа занятий в лаборатории.

При подготовке к работе изучается структура одного из клонов семейства – микроконтроллера КМ1816ВЕ51, его функциональные узлы и особенности их работы. Перед началом лабораторной работы проводится коллоквиум. Студенты, успешно ответившие на поставленные вопросы, допускаются к лабораторной части работы.

В лаборатории на простом примере по методу “Быстрый старт” изучаются этапы технологии разработки и отладки программ, основные приёмы работы с ProView. После выполнения работы оформляется отчёт с указанным ниже содержанием.

2. ЗАДАНИЯ ДЛЯ ДОМАШНЕЙ ПОДГОТОВКИ

2.1. Изучите особенности однокристальных микроконтроллеров

Семейства однокристальных микроконтроллеров. Обобщённые структуры. Функциональные возможности и области применения [1-4].

2.2. Изучите архитектуру микроконтроллеров семейства Intel 8051

Однокристальные микроконтроллеры серии К1816 и К1830. Микроконтроллеры К1816ВЕ48 и К1816ВЕ51. Арифметико-логическое устройство, память данных и память команд, регистры общего назначения и регистры специальных функций, устройство управления и синхронизации, таймер-счётчик, параллельные порты ввода-вывода, последовательный порт. Система прерываний. Расширение памяти программ и памяти данных. Расширение портов [2, 3].

2.3. Изучите систему команд микроконтроллеров

Общие сведения о системе команд. Типы операндов. Способы адресации. Флаги результата. Группа команд передачи данных. Арифметические команды. Команды логических операций. Команды операций с битами. Группа команд передачи управления [2, 3].

2.4. Изучите раздел “Быстрый старт”

Детально исследуйте исходные тексты прилагаемой программы на языке С и ассемблере.

2.5. Контрольные вопросы

1. Перечислите характерные черты архитектуры однокристальных микроконтроллеров.
2. Укажите программно-доступные узлы К1816ВЕ51 и назначение регистров специальных функций.
3. Дайте определение понятию “булев” процессор.
4. Назовите и охарактеризуйте четыре типа информационных объектов, с которыми может оперировать арифметико-логическое устройство микроконтроллера.
5. Какова ёмкость адресуемой памяти программ и данных в К1816ВЕ51?
6. Какой регистр выполняет функции базового регистра при косвенных переходах в программе?
7. Какие операции могут быть выполнены только с использованием аккумулятора?
8. Какие операции могут быть выполнены без участия аккумулятора?
9. Какой формат имеет слово состояния программы К1816ВЕ51? Укажите назначение флагов.
10. Какие возможности предоставляет наличие нескольких банков регистров общего назначения?
11. Как переключить банк регистров общего назначения?
12. Какой регистр используется для адресации внешней памяти данных?
13. Как совместить адресные пространства памяти программ и данных?
14. Охарактеризуйте способ адресации элементов стека в микроконтроллере.
15. Какова длительность исполнения команд в микроконтроллере?
16. Охарактеризуйте режимы работы таймера/счётчика в К1816ВЕ51.
17. Как с помощью таймера можно измерить длительность импульса?
18. Как выводится адрес внешней памяти?
19. Какова нагрузочная способность портов?
20. Перечислите альтернативные функции портов.
21. Охарактеризуйте режимы работы последовательного порта в К1816ВЕ51.
22. Как изменить скорость передачи данных через последовательный порт?
23. Для чего используется девятый бит при передаче данных через последовательный порт?
24. Нарисуйте схему прерываний в К1816ВЕ51. Перечислите и охарактеризуйте типы прерываний.
25. Для чего нужен регистр масок прерывания? Как изменить приоритеты прерываний?
26. Как переводится микроконтроллер в режим пониженного энергопотребления?
27. Охарактеризуйте режим загрузки и верификации программ.
28. Перечислите этапы технологии разработки программ для микроконтроллеров.
29. Укажите назначение основных модулей ProView.
30. Что такое объектный код, какие функции выполняет компоновщик?
31. Укажите основные тенденции развития микроконтроллеров.

3. АРХИТЕКТУРА МИКРОКОНТРОЛЛЕРА КМ1816ВЕ51

В микропроцессорной технике выделился самостоятельный класс интегральных схем – микроконтроллеры, которые предназначены для встраивания в приборы различного назначения. От класса однокристальных микропроцессоров их отличает наличие встроенной памяти, развитые средства взаимодействия с внешними устройствами.

Микроконтроллер выполнен на основе высокоуровневой n-МОП технологии. Через четыре программируемых параллельных порта ввода/вывода и один последовательный порт микроконтроллер взаимодействует с внешними устройствами. Основу структурной схемы (рис. 1) образует внутренняя двунаправленная 8-битная шина, которая связывает между собой основные узлы и устройства микроконтроллера: резидентную память программ (RPM), резидентную память данных (RDM), арифметико-логическое устройство (ALU), блок регистров специальных функций, устройство управления (CU) и порты ввода/вывода (P0-P3).

3.1. Арифметико-логическое устройство

8-битное арифметико-логическое устройство (ALU) может выполнять арифметические операции сложения, вычитания, умножения и деления; логические операции И, ИЛИ, исключающее ИЛИ, а также операции циклического сдвига, сброса, инвертирования и т.п. К входам подключены программно-недоступные регистры T1 и T2, предназначенные для временного хранения операндов, схема десятичной коррекции (DCU) и схема формирования признаков результата операции (PSW).

Простейшая операция сложения используется в ALU для инкрементирования содержимого регистров, продвижения регистра-указателя данных (RAR) и автоматического вычисления следующего адреса резидентной памяти программ. Простейшая операция вычитания используется в ALU для декрементирования регистров и сравнения переменных.

Простейшие операции автоматически образуют “тандемы” для выполнения таких операций, как, например, инкрементирование 16-битных регистровых пар. В ALU реализуется механизм каскадного выполнения простейших операций для реализации сложных команд. Так, например, при выполнении одной из команд условной передачи управления по результату сравнения в ALU трижды инкрементируется счётчик команд (PC), дважды производится чтение из RDM, выполняется арифметическое сравнение двух переменных, формируется 16-битный адрес перехода и принимается решение о том, делать или не делать переход по программе. Все перечисленные операции выполняются всего лишь за 2 мкс.

Важной особенностью ALU 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. Эта способность достаточно важна, поскольку для управления объектами часто применяются алгоритмы, содержащие операции над входными и выходными булевыми переменными, реализация которых средствами обычных микропроцессоров сопряжена с определенными трудностями.

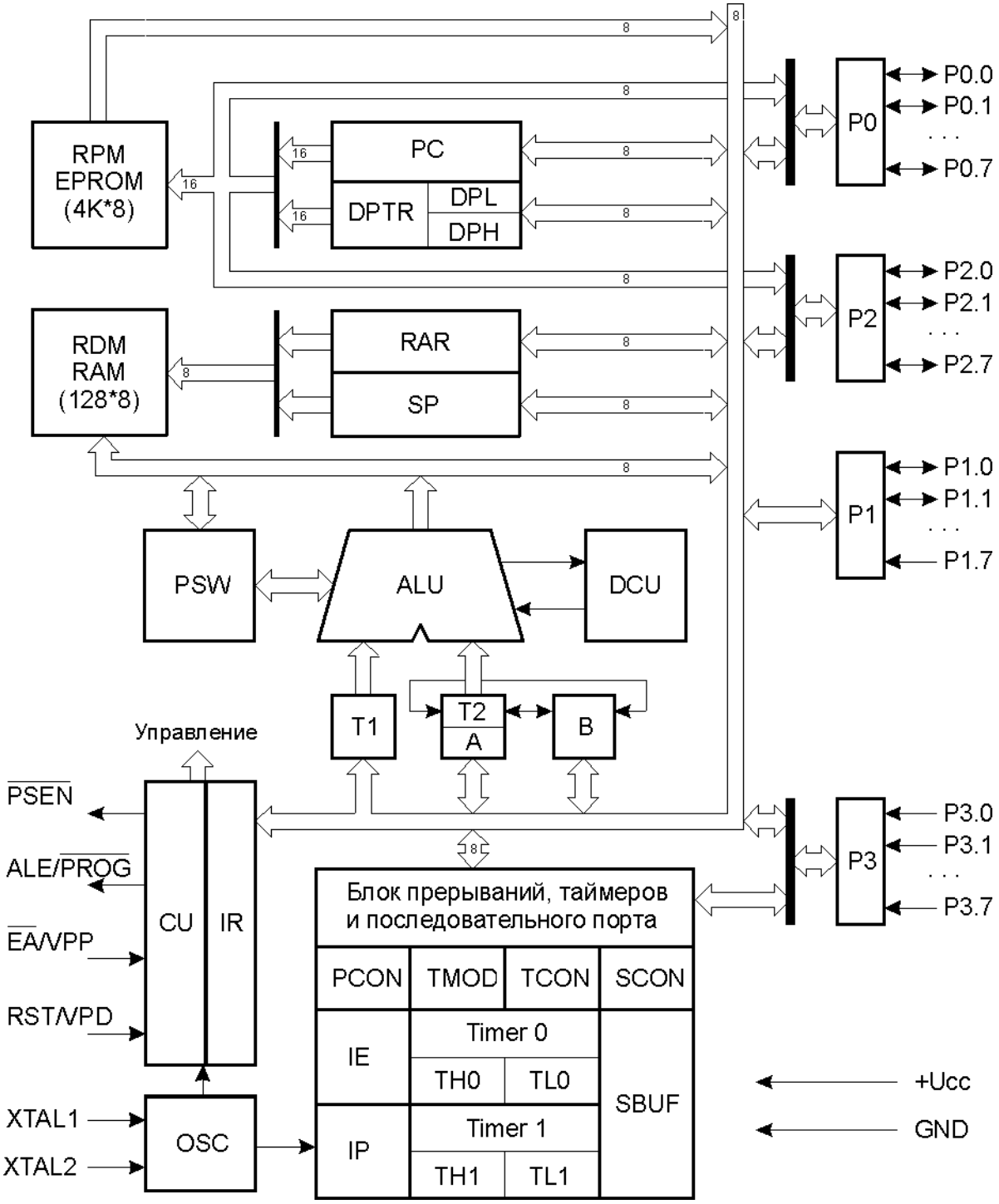

Рис. 1. *Структурная схема микроконтроллера КМ1816ВЕ51*

Таким образом, ALU может оперировать четырьмя типами информационных объектов: булевыми (1 бит), цифровыми (4 бита), байтными (8 бит) и адресными (16 бит). В ALU выполняется 51 различная операция пересылки или преобразования этих данных. Так как используется 11 режимов адресации (7 для данных и 4 для адресов), то путем комбинирования операции и режима адресации базовое число команд 111 расширяется до 255 из 256 возможных при однобайтном коде операции.

3.2. Резидентная память программ и данных

Резидентные (размещённые на кристалле) память программ (RPM) и память данных (RDM) физически и логически разделены, имеют различные механизмы адресации, работают под управлением различных сигналов и выполняют разные функции.

Память программ RPM имеет емкость 4 Кбайта и предназначена для хранения команд, констант, управляющих слов инициализации, таблиц перекодировки входных и выходных переменных и т.п. Память имеет 16-битную шину адреса, через которую обеспечивается доступ из программного счётчика PC или из регистра-указателя данных (DPTR). DPTR выполняет функции базового регистра при косвенных переходах по программе или используется в операциях с таблицами.

Память данных RDM предназначена для хранения переменных в процессе выполнения прикладной программы, адресуется одним байтом и имеет емкость 128 байт. Кроме того, к её адресному пространству примыкают адреса регистров специальных функций, которые перечислены в табл. 1.

Память программ, так же как и память данных, может быть расширена до 64 Кбайт путем подключения внешних микросхем.

Таблица 1

Блок регистров специальных функций

	Символ	Наименование		Адрес
*	A	Аккумулятор		0E0H
*	B	Регистр-расширитель аккумулятора		0F0H
*	PSW	Слово состояния программы		0D0H
	SP	Регистр-указатель стека		81H
	DPTR	Регистр-указатель данных	(DPH)	83H
			(DPL)	82H
*	P0	Порт 0		80H
*	P1	Порт 1		90H
*	P2	Порт 2		0A0H
*	P3	Порт 3		0B0H
*	IP	Регистр приоритетов прерываний		0B8H
*	IE	Регистр маски прерываний		0A8H
	TMOD	Регистр режима таймера/счётчика		89H
*	TCON	Регистр управления/статуса таймера		88H
	TH0	Таймер 0 (старший байт)		8CH
	TL0	Таймер 0 (младший байт)		8AH
	TH1	Таймер 1 (старший байт)		8DH
	TL1	Таймер 1 (младший байт)		8BH
*	SCON	Регистр управления приёмопередатчиком		98H
	SBUF	Буфер приёмопередатчика		99H
	PCON	Регистр управления мощностью		87H

Примечание. *Регистры, имена которых отмечены знаком (*), допускают адресацию отдельных битов.*

3.3. Аккумулятор, регистры общего назначения и флаги

Аккумулятор (A) является источником операнда и местом фиксации результата при выполнении арифметических, логических операций и ряда операций передачи данных. Кроме того, только с использованием аккумулятора могут быть выполнены операции сдвигов, проверка на нуль, формирование флага паритета и т.п. В распоряжении пользователя имеются 8 регистров общего назначения R0–R7 одного из четырёх возможных банков. При выполнении многих команд в ALU формируется ряд признаков операции (флагов), которые фиксируются в регистре PSW. В табл. 2 приводится перечень флагов PSW, даются их символические имена и описываются условия их формирования.

Таблица 2

Формат слова состояния программы PSW

Символ	Разряд	Имя и назначение
C	PSW.7	Флаг переноса. Устанавливается и сбрасывается аппаратно или программно при выполнении арифметических и логических операций
AC	PSW.6	Флаг вспомогательного переноса. Устанавливается и сбрасывается только аппаратно при выполнении команд сложения и вычитания и сигнализирует о переносе или займе в бите 3
F0	PSW.5	Флаг 0. Может быть установлен, сброшен или проверен программой как флаг, специфицируемый пользователем
RS1	PSW.4	Выбор банка регистров. Устанавливается и сбрасывается
RS0	PSW.3	программно для выбора рабочего банка регистров (табл. 3)
OV	PSW.2	Флаг переполнения. Устанавливается и сбрасывается аппаратно при выполнении арифметических операций
-	PSW.1	Не используется
P	PSW.0	Флаг паритета. Устанавливается и сбрасывается аппаратно в каждом цикле и фиксирует нечётное/чётное число единичных битов в аккумуляторе, т.е. выполняет контроль по четности

Таблица 3

Выбор рабочего банка регистров

RS1	RS0	Банк	Границы адресов
0	0	0	00H – 07H
0	1	1	08H – 0FH
1	0	2	10H – 17H
1	1	3	18H – 1FH

Наиболее “активным” флагом PSW является флаг переноса, который принимает участие и модифицируется в процессе выполнения множества операций, включая сложение, вычитание и сдвиги. Кроме того, флаг переноса (C) выполняет функции “булева аккумулятора” в командах, манипулирующих с битами. Флаг переполнения (OV) фиксирует арифметическое переполнение при операциях над це-

лыми числами со знаком и делает возможным использование арифметики в дополнительных кодах. ALU не управляет флагами селекции банка регистров (RS0, RS1), их значение полностью определяется прикладной программой и используется для выбора одного из четырёх регистровых банков.

В микропроцессорах, архитектура которых опирается на аккумулятор, большинство команд работают с ним, используя неявную адресацию. В Intel 8051 дело обстоит иначе. Хотя процессор имеет в своей основе аккумулятор, он может выполнять множество команд и без его участия. Например, данные могут быть переданы из любой ячейки RDM в любой регистр, любой регистр может быть загружен непосредственным операндом и т.д. Многие логические операции могут быть выполнены без участия аккумулятора. Кроме того, переменные могут быть инкрементированы, декрементированы и проверены без использования аккумулятора. Флаги и управляющие биты могут быть проверены и изменены аналогично.

3.4. Регистры-указатели

8-битный указатель стека (SP) может адресовать любую область RDM. Его содержимое инкрементируется прежде, чем данные будут запомнены в стеке в ходе выполнения команд PUSH и CALL. Содержимое SP декрементируется после выполнения команд POP и RET. Подобный способ адресации элементов стека называют прединкрементным/постдекрементным. В процессе инициализации микроконтроллера после сигнала RST в SP автоматически загружается код 07H. Это значит, что если прикладная программа не переопределяет стек, то первый элемент данных в стеке будет располагаться в ячейке RDM с адресом 08H.

Двухбайтный регистр-указатель данных DPTR обычно используется для фиксации 16-битного адреса в операциях с обращением к внешней памяти. Командами микроконтроллера регистр-указатель данных может быть использован или как 16-битный регистр, или как два независимых 8-битных регистра (DPH и DPL).

3.5. Регистры специальных функций

Регистры с символическими именами IP, IE, TMOD, TCON, SCON и PCON используются для фиксации и программного изменения управляющих бит и бит состояния схемы прерывания, таймера/счётчика, приёмопередатчика последовательного порта и для управления энергопотреблением. Их организация будет описана ниже при рассмотрении особенностей работы микроконтроллера в различных режимах.

3.6. Устройство управления и синхронизации

Кварцевый резонатор, подключаемый к внешним выводам микроконтроллера, управляет работой внутреннего генератора, который в свою очередь формирует сигналы синхронизации. Устройство управления (CU) на основе сигналов синхронизации формирует машинный цикл фиксированной длительности, равной 12 периодам резонатора. Большинство команд микроконтроллера выполняется за один машинный цикл. Некоторые команды, оперирующие с 2-байтными словами или связанные с обращением к внешней памяти, выполняются за два машинных цикла. Только команды деления и умножения требуют четырех машинных циклов. На ос-

нове этих особенностей работы устройства управления производится расчёт времени исполнения прикладных программ.

На схеме микроконтроллера к устройству управления примыкает регистр команд (IR). В его функцию входит хранение кода выполняемой команды.

Входные и выходные сигналы устройства управления и синхронизации:

- PSEN – разрешение программной памяти,
- ALE – выходной сигнал разрешения фиксации адреса,
- PROG – сигнал программирования,
- EA – блокировка работы с внутренней памятью,
- VPP – напряжение программирования,
- RST – сигнал общего сброса,
- VPD – вывод резервного питания памяти от внешнего источника,
- XTAL – входы подключения кварцевого резонатора.

3.7. Параллельные порты ввода/вывода информации

Все четыре порта (P0-P3) предназначены для ввода или вывода информации побайтно. Каждый порт содержит управляемые регистр-защёлку, входной буфер и выходной драйвер.

Выходные драйверы портов 0 и 2, а также входной буфер порта 0 используются при обращении к внешней памяти. При этом через порт 0 в режиме временного мультиплексирования сначала выводится младший байт адреса, а затем выдается или принимается байт данных. Через порт 2 выводится старший байт адреса в тех случаях, когда разрядность адреса равна 16 бит.

Все выводы порта 3 могут быть использованы для реализации альтернативных функций, перечисленных в табл. 4. Эти функции могут быть задействованы путем записи 1 в соответствующие биты регистра-защёлки (P3.0-P3.7) порта 3.

Таблица 4

Альтернативные функции порта P3

Символ	Разряд	Имя и назначение
RD	P3.7	Чтение. Активный сигнал низкого уровня формируется аппаратно при обращении к внешней памяти данных
WR	P3.6	Запись. Активный сигнал низкого уровня формируется аппаратно при обращении к внешней памяти данных
T1	P3.5	Вход таймера/счётчика 1 или тест-вход
T0	P3.4	Вход таймера/счётчика 0 или тест-вход
INT1	P3.3	Вход запроса прерывания 1. Воспринимается сигнал низкого уровня или срез
INT0	P3.2	Вход запроса прерывания 0. Воспринимается сигнал низкого уровня или срез
TXD	P3.1	Выход передатчика последовательного порта в режиме UART. Выход синхронизации в режиме регистра сдвига
RXD	P3.0	Вход приёмника последовательного порта в режиме UART. Ввод/вывод данных в режиме регистра сдвига

Порт 0 является двунаправленным, а порты 1-3 - квазидвунаправленными. Каждая линия портов может быть использована независимо для ввода или вывода.

По сигналу RST в регистры-защёлки всех портов автоматически записываются единицы, настраивающие их тем самым на режим ввода.

Все порты могут быть использованы для организации ввода/вывода информации по двунаправленным линиям передачи. Однако порты 0 и 2 не могут быть использованы для этой цели в случае, если система имеет внешнюю память, связь с которой организуется через общую разделяемую шину адреса/данных, работающую в режиме временного мультиплексирования.

Обращение к портам ввода/вывода возможно с использованием команд, оперирующих с байтом, отдельным битом, произвольной комбинацией битов. При этом в тех случаях, когда порт является одновременно операндом и местом назначения результата, устройство управления автоматически реализует специальный режим, который называется “чтение-модификация-запись”. Этот режим обращения предполагает ввод сигналов не с внешних выводов порта, а из его регистра-защёлки, что позволяет исключить неправильное считывание ранее выведенной информации. Этот механизм обращения к портам реализован в командах:

- ANL – логическое И, например, ANL P1,A;
- ORL – логическое ИЛИ, например, ORL P2,A;
- XRL – исключающее ИЛИ, например, XRL P3,A;
- JBC – переход, если в адресуемом бите единица, и последующий сброс бита, например, JBC P1.1, LABEL;
- CPL – инверсия бита, например, CPL P3.3;
- INC – инкремент порта, например, INC P2;
- DEC – декремент порта, например, DEC P2;
- DJNZ – декремент порта и переход, если его содержимое не равно нулю, например, DJNZ r, LABEL;
- MOV PX.Y,C – передача бита переноса в бит Y порта X;
- SET PX.Y – установка бита Y порта X;
- CLR PX.Y – сброс бита Y порта X.

3.8. Таймер/счётчик

В составе микроконтроллера имеются регистровые пары с символическими именами TH0, TL0 и TH1, TL1, на основе которых функционируют два независимых программно-управляемых 16-битных таймера/счётчика событий (T/C0 и T/C1). При работе в качестве таймера содержимое T/C инкрементируется в каждом машинном цикле, то есть через каждые 12 периодов резонатора. При работе в качестве счётчика содержимое T/C инкрементируется под воздействием перехода из 1 в 0 внешнего входного сигнала, подаваемого на соответствующий (T0, T1) вход микроконтроллера. Опрос сигналов выполняется в каждом машинном цикле. Так как на распознавание перехода требуется два машинных цикла, то максимальная частота подсчёта входных сигналов равна 1/24 частоты резонатора. На длительность периода входных сигналов ограничений сверху нет. Для гарантированного прочтения входного считываемого сигнала он должен удерживать значение 1 как минимум в течение одного машинного цикла.

Для управления режимами работы и для организации взаимодействия таймеров с системой прерывания используются два регистра специальных функций TMOD и TCON, описание которых приводится в табл. 5-7. Для обоих T/C режимы работы 0, 1 и 2 одинаковы. Режимы 3 для T/C0 и T/C1 различны.

Таблица 5

Регистр режима работы таймера/счётчика

Символ	Разряд	Имя и назначение
GATE	TMOD.7 для T/C1 TMOD.3 для T/C0	Управление блокировкой. Если бит установлен, то таймер/счётчик “x” разрешен до тех пор, пока на входе “INT x” высокий уровень и бит управления “TRx” установлен. Если бит сброшен, то T/C разрешается, как только бит управления “TRx” устанавливается
C/T	TMOD.6 для T/C1 TMOD.2 для T/C0	Бит выбора режима таймера или счётчика событий. Если бит сброшен, то работает таймер от внутреннего источника сигналов синхронизации. Если бит установлен, то работает счётчик от внешних сигналов на входе “Tx”
M1	TMOD.5 для T/C1 TMOD.1 для T/C0	Режим работы (см. табл. 6)
M0	TMOD.4 для T/C1 TMOD.0 для T/C0	

Таблица 6

Режимы работы таймера/счётчика

M1	M0	Режим работы
0	0	“TLx” работает как 5-битный предделитель
0	1	16-битный таймер/счётчик. “THx” и “TLx” включены последовательно
1	0	8-битный автоперезагружаемый таймер/счётчик. “THx” хранит значение, которое должно быть перезагружено в “TLx” каждый раз по переполнению
1	1	Таймер/счётчик 1 останавливается. Таймер/счётчик 0: TL0 работает как 8-битный таймер/счётчик, и его режим определяется управляющими битами таймера 0. TH0 работает только как 8-битный таймер, и его режим определяется управляющими битами таймера 1

Режим 0. Перевод любого T/C в этот режим делает его 8-разрядным таймером, на вход которого подключен 5-битный предделитель частоты на 32. В этом режиме таймерный регистр имеет разрядность 13 бит. При переходе из состояния “все единицы” в состояние “все нули” устанавливается флаг прерывания от таймера TF1. Входной синхросигнал таймера 1 разрешен (поступает на вход T/C), когда управляющий бит TR1 установлен в 1 и либо управляющий бит GATE (блокировка) равен 0, либо на внешний вход запроса прерывания INT1 поступает уровень 1.

Установка бита GATE в 1 позволяет использовать таймер для измерения длительности импульсного сигнала, подаваемого на вход запроса прерывания.

Таблица 7

Регистр управления/статуса таймера

Символ	Разряд	Имя и назначение
TF1	TCON.7	Флаг переполнения таймера 1. Устанавливается аппаратно при переполнении таймера/счётчика. Сбрасывается при обслуживании прерывания аппаратно
TR1	TCON.6	Бит управления таймера 1. Устанавливается/сбрасывается программой для пуска/останова
TF0	TCON.5	Флаг переполнения таймера 0. Устанавливается аппаратно. Сбрасывается при обслуживании прерывания
TR0	TCON.4	Бит управления таймера 0. Устанавливается/сбрасывается программой для пуска/останова таймера/счётчика
IE1	TCON.3	Флаг фронта прерывания 1. Устанавливается аппаратно, когда детектируется срез внешнего сигнала INT1. Сбрасывается при обслуживании прерывания
IT1	TCON.2	Бит управления типом прерывания 1. Устанавливается/сбрасывается программно для спецификации запроса INT1 (срез/низкий уровень)
IE0	TCON.1	Флаг фронта прерывания 0. Устанавливается по срезу сигнала INT0. Сбрасывается при обслуживании прерывания
IT0	TCON.0	Бит управления типом прерывания 0. Устанавливается/сбрасывается программно для спецификации запроса INT0 (срез/низкий уровень)

Режим 1. Работа любого Т/С в этом режиме такая же, как и в режиме 0, за исключением того, что таймерный регистр имеет разрядность 16 бит.

Режим 2. В этом режиме работа организована таким образом, что переполнение (переход из состояния “все единицы” в состояние “все нули”) 8-битного счётчика TL1 приводит не только к установке флага TF1, но и автоматически перезагружает в TL1 содержимое старшего байта (TH1) таймерного регистра, которое предварительно было задано программным путем. Перезагрузка оставляет содержимое TH1 неизменным. В режиме 2 Т/С0 и Т/С1 работают совершенно одинаково.

Режим 3. В этом режиме Т/С0 и Т/С1 работают по-разному. Т/С1 сохраняет неизменным своё текущее содержимое. Иными словами, эффект такой же, как и при сбросе управляющего бита TR1 в нуль. В этом режиме TL0 и TH0 функционируют как два независимых 8-битных счётчика. Работу TL0 определяют управляющие биты Т/С0 (C/T, GATE, TR0), входной сигнал INT0 и флаг переполнения TF0. Работу TH0, который может выполнять только функции таймера (подсчёт машинных циклов микроконтроллера), определяет управляющий бит TR1. При этом TH0 использует флаг переполнения TF1.

Режим 3 используется в тех случаях, когда требуется наличие дополнительного 8-битного таймера или счётчика событий. Можно считать, что в режиме 3 микроконтроллер имеет в своем составе три таймера/счётчика. В том случае, если

T/C0 используется в режиме 3, T/C1 может быть или включен, или выключен, или переведен в свой собственный режим 3, или может быть использован последовательным портом в качестве генератора частоты передачи, или, наконец, может быть использован в любом применении, не требующем прерывания.

3.9. Последовательный порт

Через универсальный асинхронный приёмопередатчик UART (Universal Asynchronous Receiver-Transmitter) происходит передача информации, представленной последовательным кодом (младшими битами вперед), в полном дуплексном режиме обмена. В состав UART, называемого часто последовательным портом, входят принимающий и передающий сдвигающие регистры, а также специальный буферный регистр (SBUF) приёмопередатчика.

3.9.1. Регистр SBUF

Представляет собой два независимых регистра: буфер приёмника и буфер передатчика. Загрузка байта в SBUF немедленно вызывает начало процесса передачи через последовательный порт. Когда байт считывается из SBUF, это значит, что его источником является приёмник последовательного порта. Запись байта в буфер приводит к автоматической переписи байта в сдвигающий регистр передатчика и инициирует начало передачи байта. Наличие буферного регистра приёмника позволяет совмещать операцию чтения ранее принятого байта с приёмом очередного байта. Если к моменту окончания приёма байта предыдущий байт не был считан, то он будет потерян.

Последовательный порт может работать в четырех различных режимах.

Режим 0. Информация передаётся и принимается через вход приёмника RXD. Принимаются и передаются 8 бит данных. Через внешний выход передатчика TXD выдаются импульсы сдвига, которые сопровождают каждый бит. Частота передачи равна 1/12 частоты резонатора.

Режим 1. Через TXD передаются или из RXD принимаются 10 бит: стартбит (0), 8 бит данных и стоп-бит (1). Скорость приёма/передачи – величина переменная и задаётся таймером.

Режим 2. Через TXD передаются или из RXD принимаются 11 бит: стартбит, 8 бит данных, программируемый девятый бит и стоп-бит. При передаче девятый бит может использоваться для повышения достоверности передачи путём контроля по чётности и в него можно поместить значение признака паритета из PSW. Частота приёма/передачи выбирается программно и может быть равна 1/32 или 1/64 частоты резонатора в зависимости от SMOD.

Режим 3. Совпадает с режимом 2, но частота приёма/передачи является величиной переменной и задаётся таймером.

3.9.2. Регистр SCON

Регистр предназначен для управления режимом работы UART. Регистр содержит управляющие биты и девятый бит принимаемых или передаваемых данных RB8 и TB8, а также биты прерывания приёмопередатчика RI и TI. Функциональное назначение битов указано в табл. 8 и 9.

Таблица 8

Регистр у правления/статуса UART

Символ	Разряд	Имя и назначение
SM0	SCON.7	Биты управления режимом работы UART. Устанавливаются/сбрасываются программно (табл. 9)
SM1	SCON.6	
SM2	SCON.5	Бит управления режимом UART. Устанавливается программно для запрета приёма сообщения, в котором девятый бит равен 0
REN	SCON.4	Бит разрешения приёма. Устанавливается/сбрасывается программно для разрешения/запрета приёма последовательных данных
TB8	SCON.3	Передача бита 8. Устанавливается/сбрасывается программно для задания девятого передаваемого бита в режиме UART - 9 бит
RB8	SCON.2	Приём бита 8. Устанавливается/сбрасывается аппаратно для фиксации девятого принимаемого бита в режиме UART - 9 бит
TI	SCON.1	Флаг прерывания передатчика. Устанавливается аппаратно при окончании передачи байта. Сбрасывается программно после обслуживания прерывания
RI	SCON.0	Флаг прерывания приёмника. Устанавливается аппаратно при приёме байта. Сбрасывается программно после обслуживания прерывания

Таблица 9

Режим работы UART

SM0	SM1	Режим работы UART
0	0	Сдвигающий регистр расширения ввода/вывода
0	1	UART - 8 бит. Изменяемая скорость передачи
1	0	UART - 9 бит. Фиксированная скорость передачи
1	1	UART - 9 бит. Изменяемая скорость передачи

Прикладная программа путём загрузки в два старших разряда SCON определяет режим работы UART. Во всех режимах передача инициируется любой командой, где SBUF указан как получатель байта. Приём в UART в режиме 0 происходит при условии RI=0 и REN=1. В режимах 1-3 приём начинается с приходом стартбита, если REN=1.

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

Флаг прерывания передатчика TI устанавливается аппаратно в конце периода передачи восьмого бита данных в режиме 0 и в начале периода передачи стоп-

бита в режимах 1-3. Подпрограмма обслуживания этого прерывания должна сбрасывать бит TI.

Флаг прерывания приёмника RI устанавливается аппаратно в конце периода приёма восьмого бита данных в режиме 0 и в середине периода приёма стол-бита в режимах 1-3. Подпрограмма обслуживания прерывания должна сбрасывать бит RI.

3.9.3. Работа UART в мультиконтроллерных системах

В системах децентрализованного управления, которые используются для управления и регулирования в топологически распределенных объектах, возникает задача обмена информацией между множеством микроконтроллеров, объединенных в локальную вычислительно-управляющую сеть. Как правило, локальные сети на основе Intel 8051 имеют магистральную архитектуру с разделяемым моноканалом (коаксиальный кабель, витая пара, оптическое волокно), по которому осуществляется обмен информацией между контроллерами.

Бит SM2 в SCON позволяет простыми средствами реализовать межконтроллерный обмен. Механизм обмена построен на том, что в режимах 2 и 3 программируемый девятый бит данных при приёме фиксируется в бите RB8. UART может быть запрограммирован таким образом, что при получении стоп-бита прерывание от приёмника будет возможно только при условии RB8=1. Ведущий контроллер всем ведомым передаёт широковещательное сообщение с байтом-идентификатором абонента, которое отличается от байтов данных только тем, что в его девятом бите содержится 1. Ведомые по этому признаку вызывают подпрограммы сравнения байта-идентификатора с кодом собственного сетевого адреса. Адресуемый контроллер сбрасывает свой SM2 и готовится к приёму блока данных. Остальные ведомые микроконтроллеры оставляют неизменными свои SM2=1 и передают управление основной программе. При SM2=1 информационные байты в сети прерывания не вызывают.

В режиме 1 автономного микроконтроллера SM2 используется для контроля истинности стоп-бита. В режиме 0 SM2 не используется и должен быть сброшен.

3.9.4. Скорость приёма/передачи

Скорость зависит от режима работы UART. В режиме 0 частота зависит только от резонатора: $f_0=f_{рез}/12$. За один машинный цикл передаётся один бит.

В режимах 1-3 скорость зависит от значения управляющего бита SMOD в регистре специальных функций PCON (табл. 10).

В режиме 2 частота передачи $f_2=(2^{SMOD}/64)f_{рез}$.

В режимах 1 и 3 в формировании частоты передачи кроме управляющего бита SMOD принимает участие таймер 1. При этом частота передачи зависит от частоты переполнения (OVT1) и определяется следующим образом: $f_{1,3}=(2^{SMOD}/32)f_{OVT1}$. Прерывание от таймера 1 в этом случае должно быть заблокировано. Сам T/C1 может работать и как таймер, и как счётчик событий в любом из трёх режимов. Однако наиболее удобно использовать режим таймера с автоперезагрузкой (старшая тетрада TMOD=0010B). При этом частота передачи определяется выражением $f_{1,3}=(2^{SMOD}/32)(f_{рез}/12)(256-(TH1))$. В табл. 11 приводится описание способов настройки T/C1 для получения типовых частот передачи данных через UART.

Таблица 10

Регистр управления мощностью PCON

Символ	Разряд	Наименование и функция
SMOD	PCON.7	Удвоенная скорость передачи. Если бит установлен в 1, то скорость передачи вдвое больше, чем при SMOD=0
-	PCON.6-4	Не используются
GF1	PCON.3	Флаги, специфицируемые пользователем (флаги общего назначения)
GF0	PCON.2	
PD	PCON.1	Бит пониженной мощности. При установке в 1 микроконтроллер переходит в режим пониженного энергопотребления
IDL	PCON.0	Бит холостого хода. Если бит установлен в 1, то микроконтроллер переходит в режим холостого хода

Примечание. *При одновременной записи 1 в PD и IDL бит PD имеет преимущество. Сброс PCON выполняется путем загрузки в него кода 0XXX0000.*

Таблица 11

Настройка таймера 1 для управления частотой работы UART

Частота приёма/ передачи (BAUD RATE)		Частота резонатора, МГц	SMOD	Таймер/счётчик 1		
				C/T	Режим (MODE)	Перезагружаемое число
Режим 0, макс.:	1 МГц	12	X	X	X	X
Режим 2, макс.:	375 кГц	12	1	X	X	X
Режимы 1,3:	62,5 кГц	12	1	0	2	0FFH
	19,2 кГц	11,059	1	0	2	0FDH
	9,6 кГц	11,059	0	0	2	0FDH
	4,8 кГц	11,059	0	0	2	0FAH
	2,4 кГц	11,059	0	0	2	0F4H
	1,2 кГц	11,059	0	0	2	0E8H
	137,5 Гц	11,059	0	0	2	1DH
	110 Гц	6	0	0	2	72H
	110 Гц	12	0	0	1	0FEEBH

3.10. Система прерываний

Внешние прерывания INT0 и INT1 (рис. 2) могут быть вызваны уровнем или переходом сигнала из 1 в 0 на входах микроконтроллера в зависимости от значений управляющих битов IT0 и IT1 в регистре TCON. От внешних прерываний устанавливаются флаги IE0 и IE1 в регистре TCON, которые инициируют вызов соответствующей подпрограммы обслуживания прерывания. Сброс этих флагов выполняется аппаратно только в том случае, если прерывание было вызвано по переходу (срезу) сигнала. Если же прерывание вызвано уровнем входного сигнала, то сбросом флага IE управляет соответствующая подпрограмма обслуживания прерывания путем воздействия на источник прерывания с целью снятия им запроса.

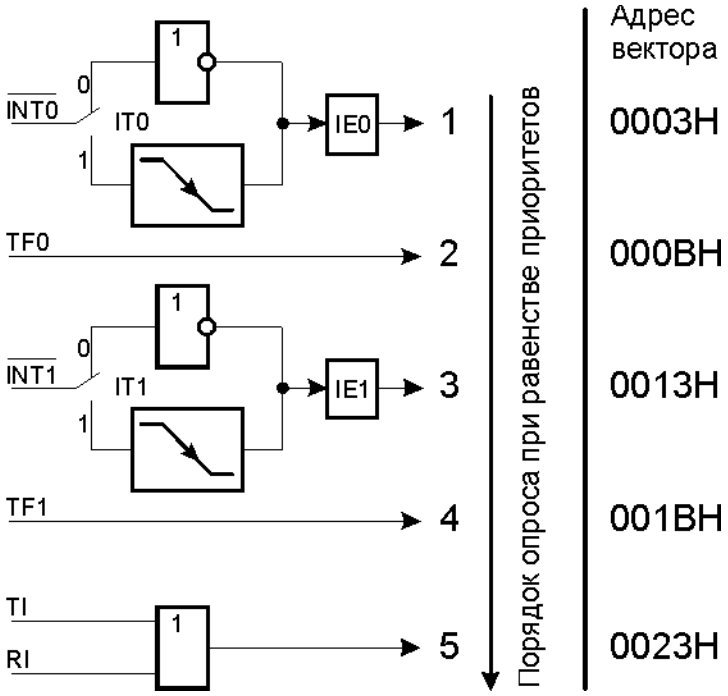

Рис. 2. *Схема прерываний*

Флаги запросов прерывания от таймеров TF0 и TF1 сбрасываются автоматически при передаче управления подпрограмме обслуживания. Флаги запросов прерывания RI и TI устанавливаются UART аппаратно, но сбрасываться должны программой. Прерывания могут быть вызваны или отменены программой, так как все перечисленные флаги программно доступны.

В блоке регистров специальных функций есть два регистра, предназначенных для управления режимом прерываний и уровнями приоритета. Форматы этих регистров, имеющих символические имена IE и IP описаны в табл. 12 и 13 соответственно.

Таблица 12

Регистр масок прерывания IE

Символ	Разряд	Имя и назначение
EA	IE.7	Снятие блокировки прерываний. Сбрасывается программно для запрета всех прерываний независимо от состояний IE4-IE0
-	IE.6, 5	Не используются
ES	IE.4	Бит разрешения прерывания от UART. Установка/сброс программой для разрешения/запрета прерываний от флагов TI, RI
ET1	IE.3	Бит разрешения прерывания от таймера 1. Установка/сброс программой для разрешения/запрета прерываний от таймера 1
EX1	IE.2	Бит разрешения внешнего прерывания 1. Установка/сброс программой для разрешения/запрета прерываний
ET0	IE.1	Разрешение прерывания от таймера 0. Работает аналогично IE.3
EX0	IE.0	Разрешения внешнего прерывания 0. Работает аналогично IE.2

Таблица 13

Регистр приоритетов прерывания IP

Символ	Разряд	Имя и назначение
-	IP.7-5	Не используются
PS	IP.4	Бит приоритета UART. Установка/сброс программой для назначения прерыванию от UART высшего/низшего приоритета
PT1	IP.3	Бит приоритета таймера 1. Установка/сброс программой для назначения прерыванию от таймера 1 высшего/низшего приоритета
PX1	IP.2	Бит приоритета внешнего прерывания 1. Установка/сброс программой для назначения прерыванию INT1 высшего/низшего приоритета
PT0	IP.1	Бит приоритета таймера 0. Работает аналогично IP.3
PX0	IP.0	Приоритет внешнего прерывания 0. Работает аналогично IP.2

Возможность программной установки/сброса любого управляющего бита в этих двух регистрах делает систему прерываний исключительно гибкой.

Флаги прерываний опрашиваются в каждом машинном цикле. Ранжирование прерываний по приоритету выполняется в течение следующего машинного цикла. Система прерываний сформирует аппаратно вызов LCALL соответствующей подпрограммы обслуживания, если она не заблокирована одним из условий:

- в данный момент обслуживается запрос прерывания равного или более высокого уровня приоритета;
- текущий машинный цикл – не последний в цикле выполняемой команды;
- выполняется команда RETI или любая команда, связанная с обращением к регистрам IE или IP.

Примечание. *Если флаг прерывания был установлен, но по одному из перечисленных условий не получил обслуживания и к моменту окончания блокировки уже был сброшен, то запрос прерывания теряется.*

По аппаратно сформированному коду команды LCALL система прерывания помещает в стек содержимое программного счётчика PC и загружает в PC адрес вектора прерывания соответствующей подпрограммы обслуживания. По этому адресу должна быть расположена команда безусловного перехода JMP к начальному адресу подпрограммы обслуживания прерывания. Эта подпрограмма в случае необходимости должна начинаться командами записи в стек PUSH слова состояния программы PSW, аккумулятора A, расширителя аккумулятора B, указателя данных DPTR и т.д. и заканчиваться командами восстановления из стека POP. Подпрограммы обслуживания прерывания обязательно завершаются командой RETI, по которой в программный счётчик перезагружается из стека сохранённый адрес возврата в основную программу. Команда RET также возвращает управление, но при этом не снимает блокировку прерывания.

4. ВВЕДЕНИЕ В PROVIEW

ProView фирмы Franklin Software Inc. – 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 с языка С,
- ассемблер,
- отладчик,
- встроенную справочную систему.

Среда разработки подобна Visual C++ Microsoft и Borland C++ для Windows. Пользователи, знакомые с любым из этих изделий, будут чувствовать себя в ProView, как дома.

Первый этап разработки программы – запись её исходного текста на каком-либо языке программирования.

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 прикладной программы в единый рекурсивный процесс, когда в любой момент времени возможен быстрый возврат к любому предыдущему этапу.

ProView имеет следующие компоненты.

4.1. Оптимизирующий кросс-компилятор C51

Язык С - универсальный язык программирования, который обеспечивает эффективность кода, элементы структурного программирования и имеет богатый набор операторов. Универсальность, отсутствие ограничений реализации делают язык С удобным и эффективным средством программирования для широкого разнообразия задач. Множество прикладных программ может быть написано легче и эффективнее на языке С, чем на других более специализированных языках.

C51 - полная реализация стандарта ANSI (Американского национального института стандартов), насколько это возможно для архитектуры Intel 8051. C51 генерирует код для всего семейства микроконтроллеров Intel 8051. Транслятор сочетает гибкость программирования на языке С с эффективностью кода и быстродействием ассемблера.

Использование языка высокого уровня С имеет следующие преимущества над программированием на ассемблере:

- глубокого знания системы команд процессора не требуется, элементарное знание архитектуры Intel 8051 желательно, но не необходимо;
- распределение регистров и способы адресации управляются полностью транслятором;
- лучшая читаемость программы, используются ключевые слова и функции, которые более свойственны человеческой мысли;
- время разработки программ и их отладки значительно короче в сравнении с программированием на ассемблере;
- библиотечные файлы содержат много стандартных подпрограмм, которые могут быть включены в прикладную программу;
- существующие программы могут многократно использоваться в новых программах, используя модульные методы программирования.

4.2. Макроассемблер A51

Ассемблер A51 совместим с ASM51 Intel для всего семейства микроконтроллеров Intel 8051. Ассемблер транслирует символическую мнемонику в перемещаемый объектный код, имеющий высокое быстродействие и малый размер. Макросредства ускоряют разработку и экономят время, поскольку общие последовательности могут быть разработаны только один раз. Ассемблер поддерживает символический доступ ко всем элементам микроконтроллера и перестраивает конфигурацию для каждой разновидности Intel 8051.

A51 транслирует исходный файл ассемблера в перемещаемый объектный модуль. При отладке или при включенной опции “Include debugging information” этот объектный файл будет содержать полную символическую информацию для отладчика/имитатора или внутрисхемного эмулятора.

4.3. Компоновщик L51

Компоновщик объединяет один или несколько объектных модулей в одну исполняемую программу. Компоновщик размещает внешние и общие ссылки, назначает абсолютные адреса перемещаемым сегментам программ. Он может обрабатывать объектные модули, созданные транслятором C51, ассемблером A51, транслятором PL/M-51 Intel и ассемблером ASM51 Intel.

Компоновщик автоматически выбирает соответствующие библиотеки поддержки и связывает только требуемые модули из библиотек. Установки по умолчанию для L51 выбраны так, чтобы они подходили для большинства прикладных программ, но можно определить и заказные установки.

4.4. Отладчик/симулятор WinSim51

Отладчик исходных текстов используется с транслятором C51, ассемблером A51, транслятором PL/M-51 Intel и ассемблером ASM51 Intel. Отладчик/симулятор позволяет моделировать большинство особенностей Intel 8051 без наличия аппаратных средств. Можно использовать его для проверки и отладки прикладной программы прежде, чем будут изготовлены аппаратные средства. При этом моделируется широкое разнообразие периферийных устройств, включая последовательный порт, внешний ввод - вывод и таймеры.

5. БЫСТРЫЙ СТАРТ

“Быстрый старт” – это обычный приём разработчиков современных программных средств. Цель состоит в том, чтобы, не углубляясь пока в подробности, дать новичку или достаточно опытному пользователю первое представление о программном средстве, дать возможность быстро получить конкретный результат. Полное представление, знания и умения появятся позже в процессе работы и изучения справочных материалов.

В качестве примера возьмём простейшую программу, с которой начинают изучение языков программирования многие поколения студентов. “Hello World” - программа из папки \Fsi\Examples\Hello\, которая выдаёт в последовательный порт (UART) микроконтроллера строку символов “Hello World” (“Привет Мир”). Весь исходный текст программы содержится в файле hello.c:

```
/***************************/
/* YOUR FIRST 8051 PROGRAM */
/***************************/

#include <reg51.h> /* special function register declarations  */
                   /* for the intended 8051 derivative        */
#include <stdio.h> /* prototype declarations for I/O functions*/

/*****************/
/* main program */
/*****************/
void main (void)  { /* execution starts here after stack init */
   SCON  = 0x50;    /* SCON: mode 1, 8-bit UART, enable rcvr  */
   TMOD |= 0x20;    /* TMOD: timer 1, mode 2, 8-bit reload    */
   TH1 = 0xf3;      /* TH1:  reload value for 2400 baud       */
   TR1 = 1;         /* TR1:  timer 1 run                      */
   TI  = 1;         /* TI:   set TI to send first char of UART*/

   printf ("Hello World\n");  /* the 'printf' function call */

   while (1) {      /* An embedded program does not stop and  */
   ;  /* ... */    /* never returns.  We've used an endless  */
   }               /* loop.  You may wish to put in your own */
}                  /* code were we've printed the dots (...) */
```

Прежде чем начать разработку проекта, скопируйте папку \Fsi\Examples\Hello\ в свою личную папку. В этой папке находится всего лишь один файл hello.c.

5.1. Запуск ProView и создание файла проекта

ProView запускается из стартового меню Windows подобно остальным приложениям (рис. 3). Если необходимо запустить программу из командной строки, её синтаксис имеет вид: PV32 [projectfile], где projectfile - имя файла проекта с расширением [.PRJ].

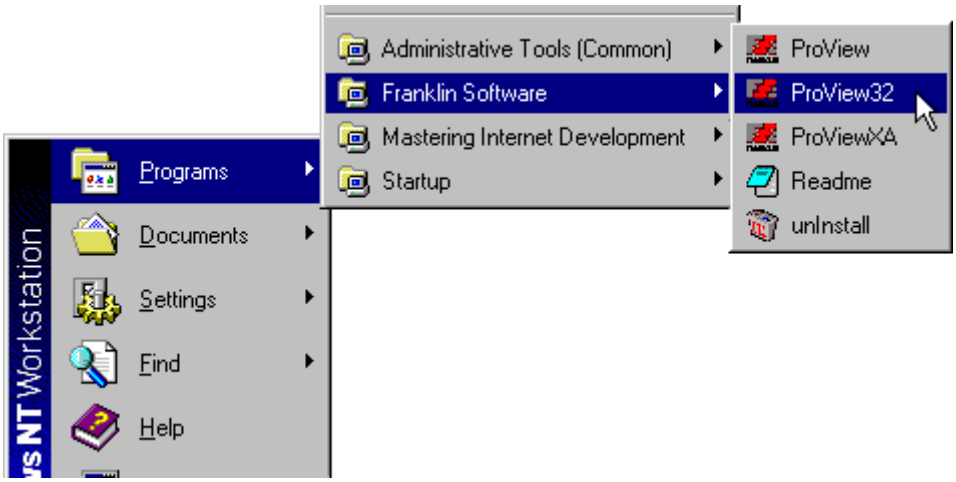

Рис. 3. *Запуск программы*

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 файл проекта, выберите New из меню Project. Откроется диалоговое окно New Project (рис. 4). Используйте кнопку Browse, чтобы войти в свою папку. Найдите папку \Hello и нажмите кнопку [OK]. Затем выберите “8051” как тип проекта.

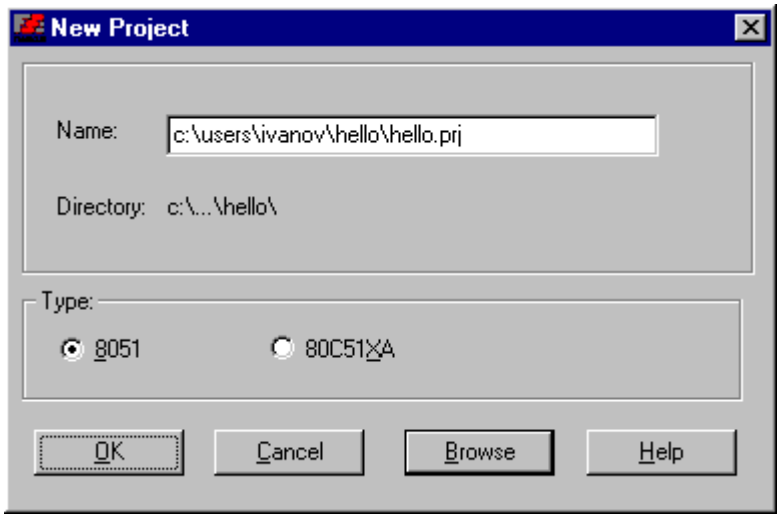

Рис. 4. *Диалоговое окно New Project*

Когда менеджер проекта открывает файл проекта, окно проекта показывает включенные исходные файлы. В данном случае пока нет никаких исходных файлов. Имеется только один исходный файл, который необходимо подключить - hello.c.

5.2. Добавка файла с исходным текстом и его редактирование

Теперь можно добавить hello.c к проекту. Выберите Add file из меню Project. Откроется диалоговое окно Add File (рис. 5). Выберите hello.c из списка.

Наш проект имеет только один исходный файл. В дальнейшем Ваши проекты, возможно, будут состоять из множества исходных файлов. Диалог Add File позволит Вам выбрать и добавить несколько файлов сразу. Для этого используют комбинацию клавиши [CTRL] и указателя мыши. Когда Вы нажмёте [Open], исходные файлы будут добавлены к проекту в выбранном порядке.

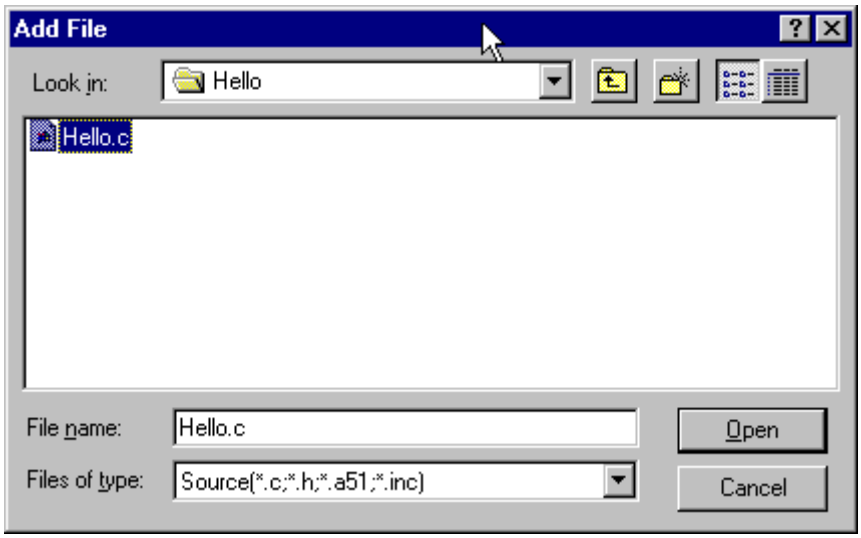

Рис. 5. *Диалоговое окно Add File*

Теперь можно редактировать текст из файла hello.c. Выберите hello.c из окна Project (рис. 6). Нажмите его правой кнопкой мыши и выберите View source file, или просто дважды щёлкните мышью для того, чтобы просматривать файл в окне редактирования.

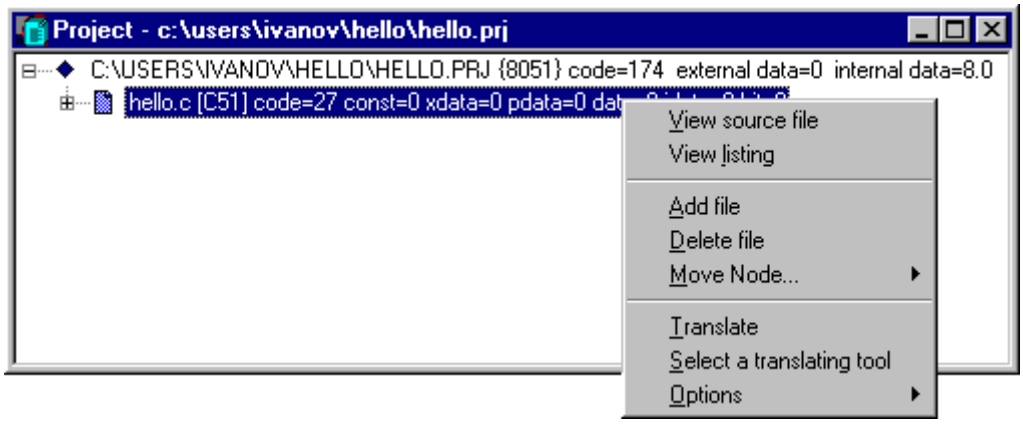

Рис. 6. *Диалоговое окно Project*

ProView загружает и показывает содержание hello.c в окне, где можно редактировать файл. Окно редактирования (рис. 7) - полнофункциональный редактор ис-

ходного текста, предлагающий такие возможности, как высвечивание синтаксических элементов и контекстный поиск. Если выбрать “printf” и нажать клавишу [F1], ProView откроет систему справки и перейдёт к разделу справки о “printf”.

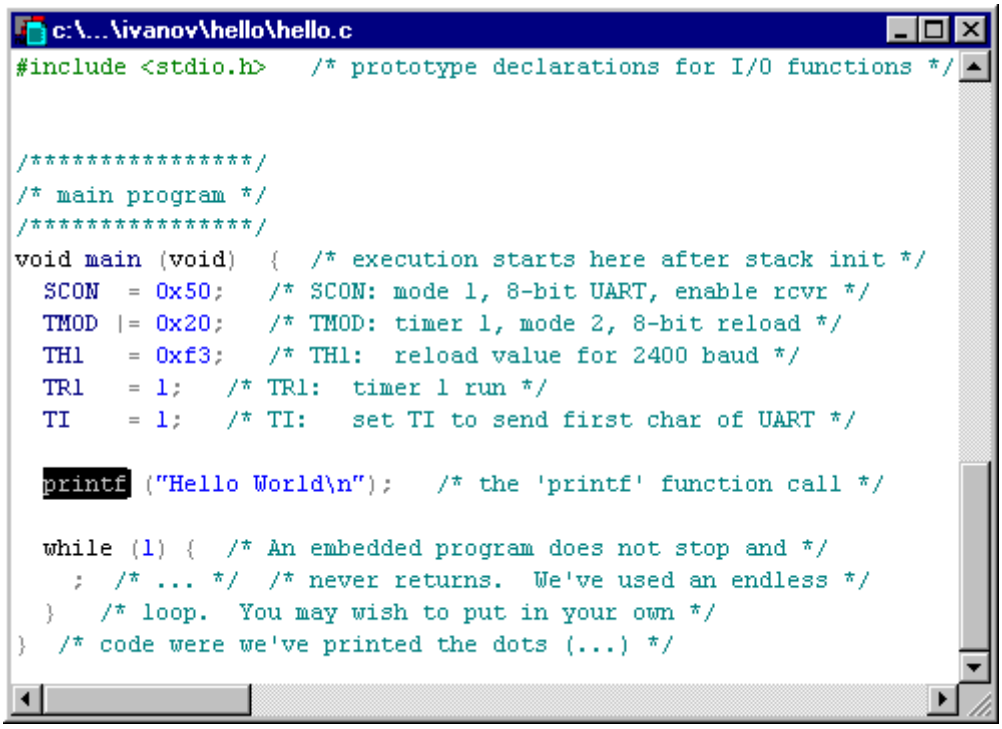

Рис. 7. *Окно редактирования*

5.3. Компиляция и компоновка

Этот процесс компилирует, связывает hello.c с библиотеками и создает абсолютный объектный модуль, который мы сможем проверить в отладчике WinSim.

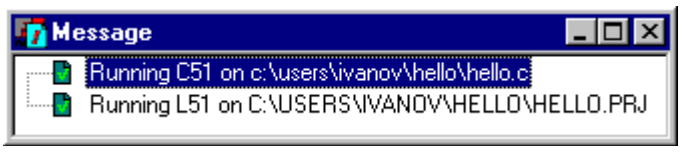

Рис. 8. *Окно сообщений*

Выберите Make из меню Project. ProView отображает окно, показывая текущее состояние процесса. Когда процесс компиляции закончится, в окне Message (рис. 8) отображается сообщение завершения. Если были обнаружены какие-нибудь ошибки, о них сообщается здесь же.

5.4. Тестирование и отладка

Выполним отладку программы. Если проект новый, откроется диалоговое окно Debug Options (рис. 9), где Вы можете изменять установки отладчика. В дальнейшем можно установить опции отладчика, выбрав Debug из меню Options. Наш проект использует значения по умолчанию.

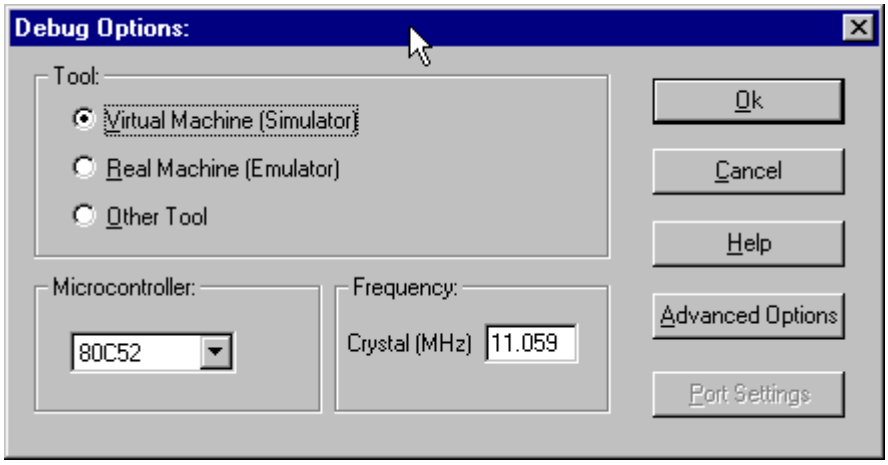

Рис. 9. *Окно диалога опций отладчика*

Выберите Start из меню Debug.

Выберите Hardware (аппаратные средства) из меню View. Выберите UART, откроется окно последовательного порта (рис. 10). В дальнейшем при работе программы здесь можно будет увидеть всё, что выводит микроконтроллер в последовательный порт.

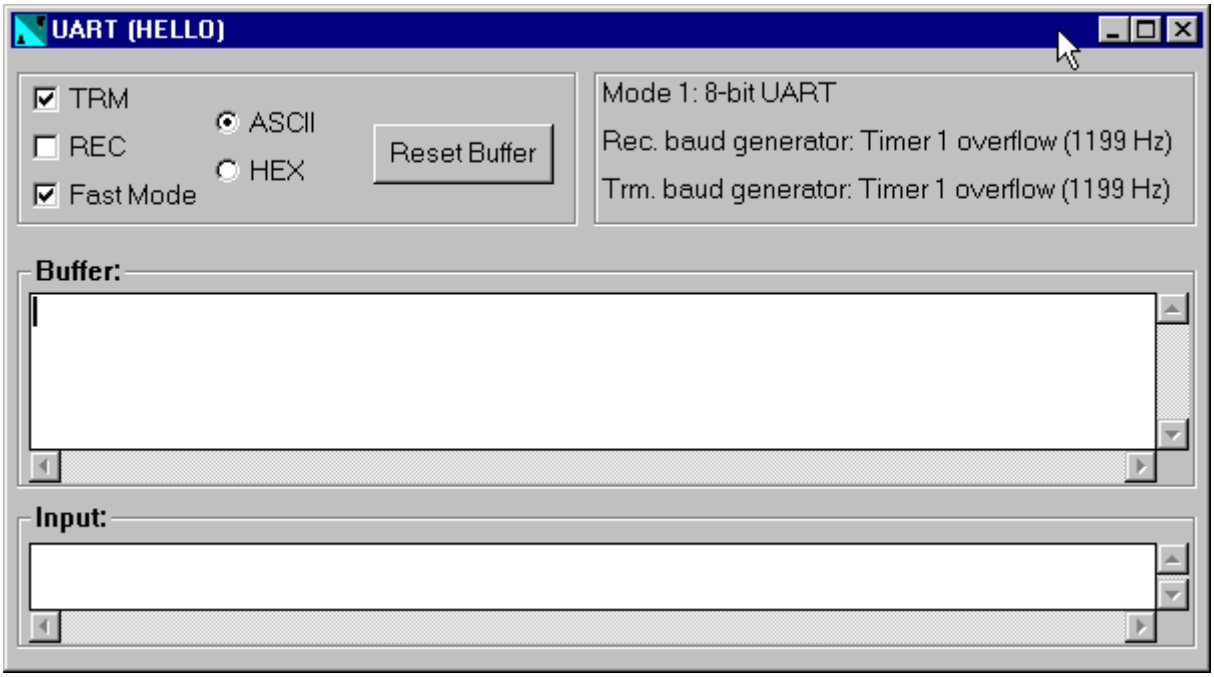

Рис. 10. *Окно последовательного порта*

Выберите Run из меню Debug или нажмите кнопку 

.

Рис. 11 показывает, как выглядит экран отладчика WinSim при выполнении программы. Обратите внимание, что в окно UART выведен текст “Hello World”.

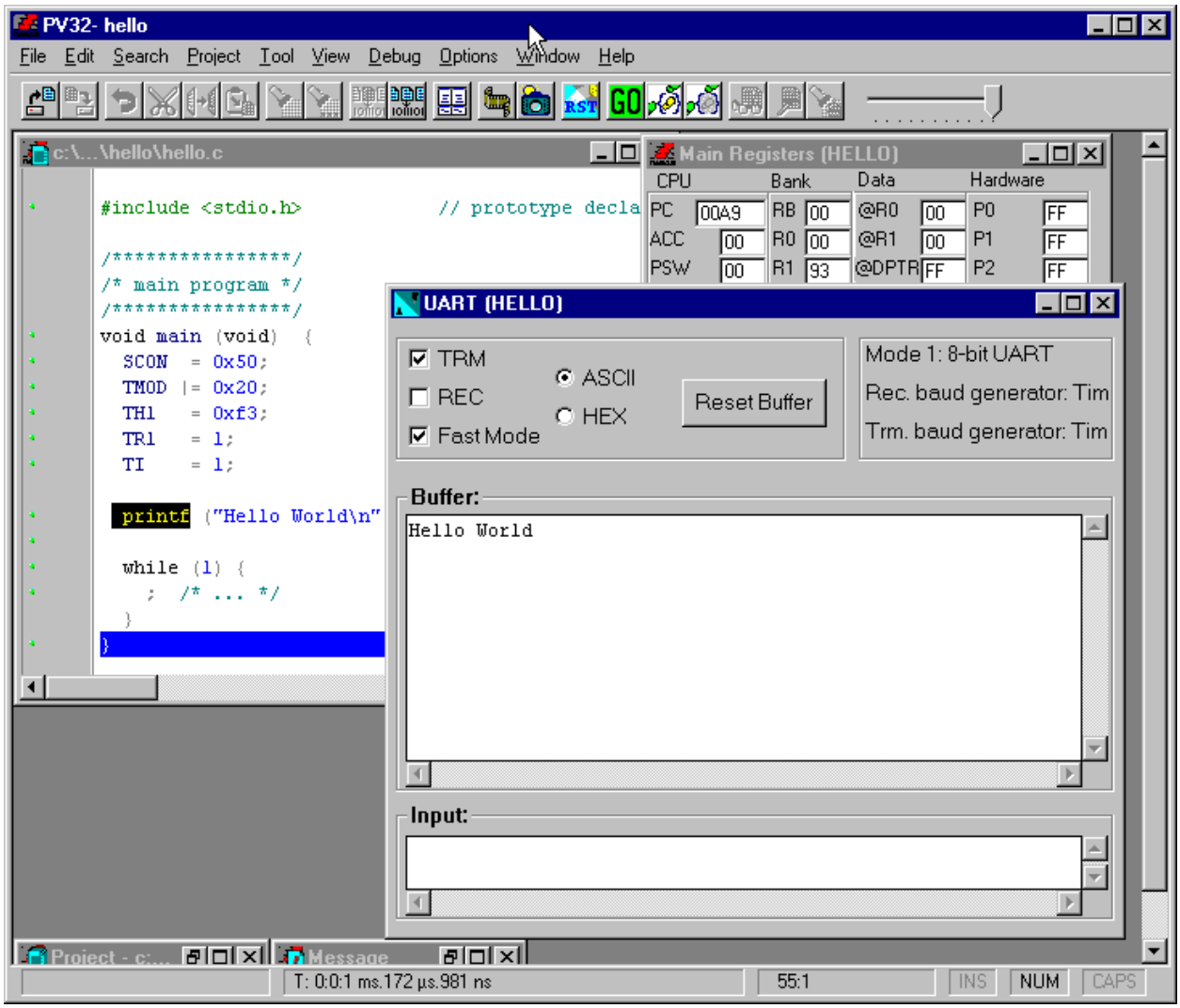

Рис. 11. Экран отладчика при выполнении программы

При выводе символов в порт начинается выполнение бесконечного цикла. Вы можете остановить выполнение программы, выбрав Stop из меню Debug. С помощью регулятора при нажатой кнопке Animate на панели инструментов можно менять скорость работы отладчика. Строка состояния показывает текущее реальное время.

5.5. Пошаговый режим и выход из отладчика

Вы можете использовать отладчик, чтобы перемещаться по программе. Выберите Reset из меню Debug (эта команда сбросит моделируемый процессор) и выберите Step Into и Step Over из меню Debug.

Команды Step позволяют “шагать” по каждой строке исходного текста. Текущая команда высвечивается на каждом шаге. Step Into позволяет войти в вызываемую функцию, Step Over – перешагнуть через неё, не входя во внутрь.

Проделайте эти операции.

Для завершения работы с отладчиком в любой момент времени Вы можете выбрать Terminate из меню Debug и возвратиться в режим редактирования.

5.6. Следующий шаг

Обратите внимание, что в режиме отладки на экране видны ещё два окна. Первое – окно кода (рис. 12), где в пошаговом режиме параллельно с исходным текстом на языке С идёт трассировка текста на ассемблере.

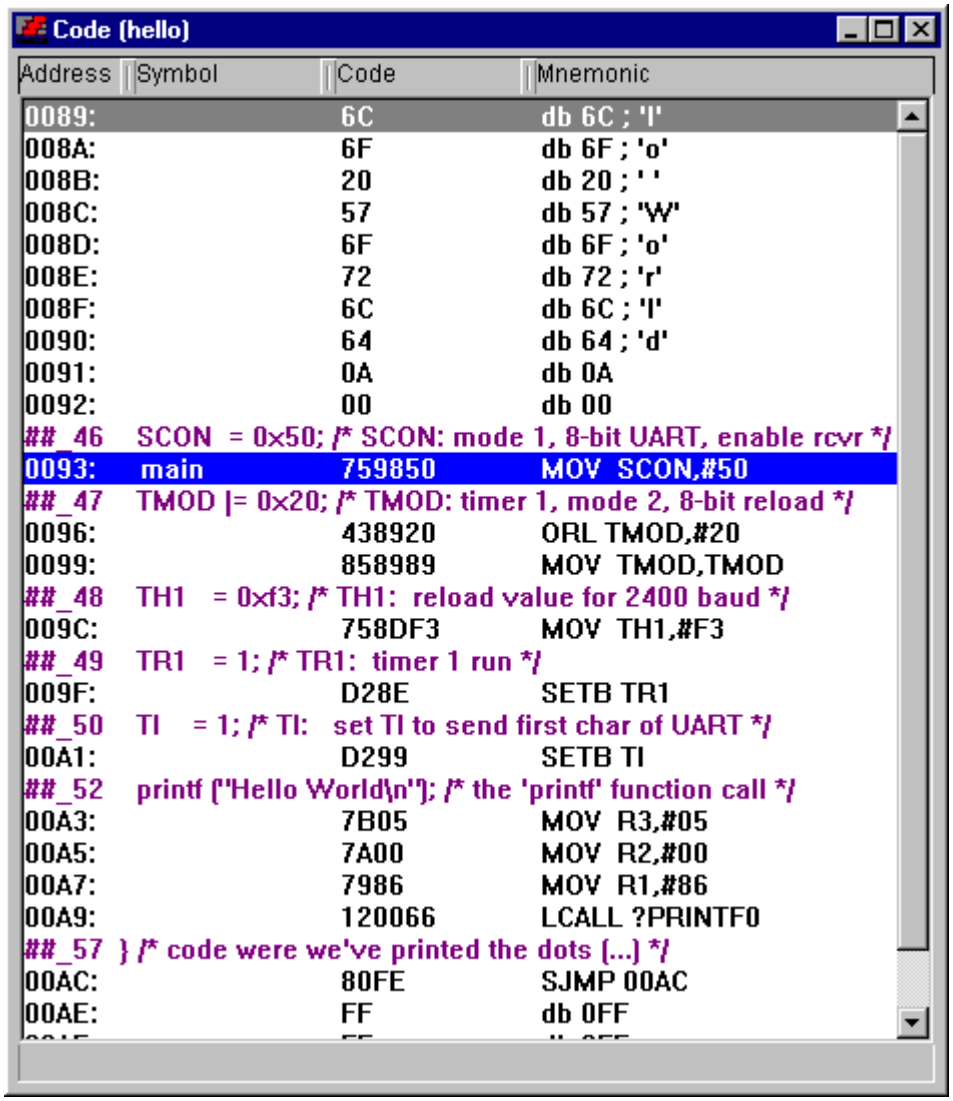

Рис. 12. *Окно кода*

Прокрутите окно кода и изучите ассемблерный аналог исходного текста. С символов “##” начинаются строки, с помощью которых легко сопоставить ассемблерный текст и текст на языке С. Обратите внимание на то, сколько кода пришлось бы написать, если проектировать программу на ассемблере.

Ассемблерный аналог текста сохраняется в файле hello.lst, если в опциях проекта (Project из меню Options) отмечено Generate Listing (рис. 13). Здесь же можно указать, какую информацию включать в листинг.

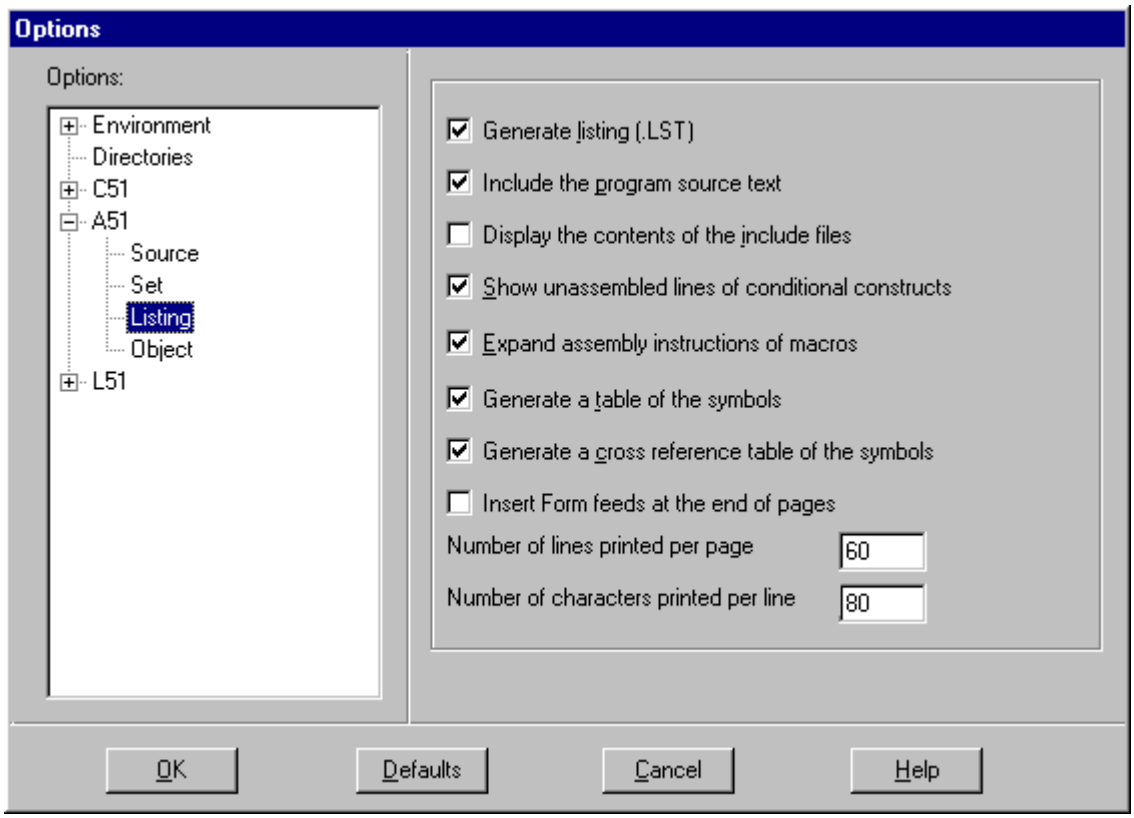

Рис. 13. *Диалог опций проекта*

Изучите смысл других опций проекта в разделах Environment, C51, A51, L51. Откройте файл листинга (рис. 14) с помощью View listing из меню View.

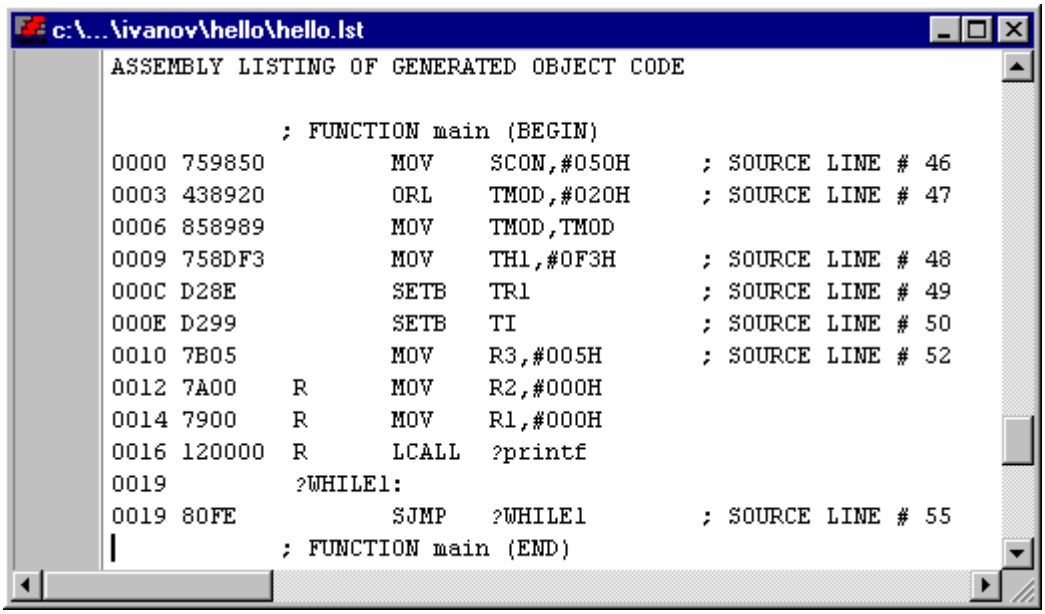

Рис. 14. *Окно файла листинга*

Изучите и постарайтесь понять содержание разделов файла листинга.

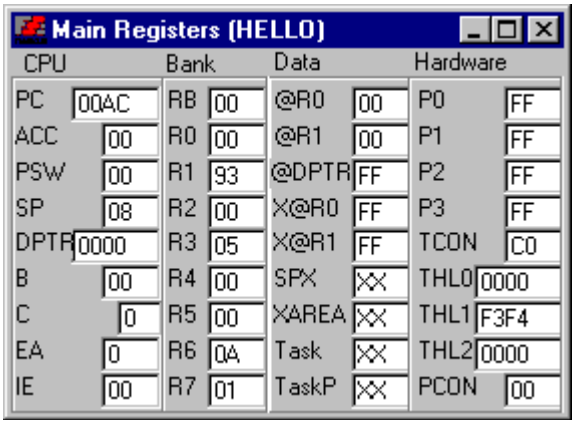

Рис. 15. *Окно регистров*

Второе окно, которое присутствует на экране во время отладки, – Main Registers (рис. 15).

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

С помощью пункта Data dump из меню View можно посмотреть содержимое памяти различного типа в режиме отладки. Попробуйте это сделать.

6. ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Для углублённого изучения возможностей ProView и её компонентов самостоятельно изучите содержание и смысл всех пунктов меню, кнопок инструментальной панели, окон и настроек. Для этого воспользуйтесь встроенной справочной системой, которая вызывается через меню Help. Эти знания потребуются при выполнении следующих лабораторных работ.

7. СОДЕРЖАНИЕ ОТЧЁТА

Отчёт о лабораторной работе должен содержать:

- титульный лист;
- цель и задачи работы;
- структурную схему микроконтроллера;
- текст программы на языке С;
- текст программы на ассемблере с комментариями у каждой команды;
- выводы по работе.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Хвощ С.Т., Варлинский Н.Н., Попов Е.А. Микропроцессоры и микро-ЭВМ в системах автоматического управления: Справочник/ Под ред. С.Т.Хвоща. Л.: Машиностроение. Ленингр. отд-ние, 1987. 640 с.
2. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах. М.: Энергоатомиздат, 1990. 224 с.
3. Однокристальные микроЭВМ/ А.В.Боборыкин, Г.П.Липовецкий, Г.В.Литвинский и др. М.: МИКАП, 1994. 400 с.
4. Микропроцессоры. В 3-х кн. Кн. 1. Архитектура и проектирование микро-ЭВМ. Организация вычислительных процессов: Учебник для втузов/ П.В.Нестеров, В.Ф.Шаньгин, В.Л.Горбунов и др.; Под ред. Л.Н.Преснухина. М.: Высшая школа, 1986. 495 с.

ВАША ПЕРВАЯ ПРОГРАММА
ДЛЯ МИКРОКОНТРОЛЛЕРА INTEL 8051

Составитель Добряк Вадим Алексеевич

Редактор Н.П.Кубыщенко

Подписано в печать 08.02.99 Формат 60x84 1/16
Бумага типографская Офсетная печать Усл. п. л. 1,86
Уч.-изд. л. 1,78 Тираж 100 Заказ 38 Цена “С”

Издательство УГТУ
620002, Екатеринбург, Мира, 19
ЗАО УМЦ УПИ. 620002, Екатеринбург, Мира, 17