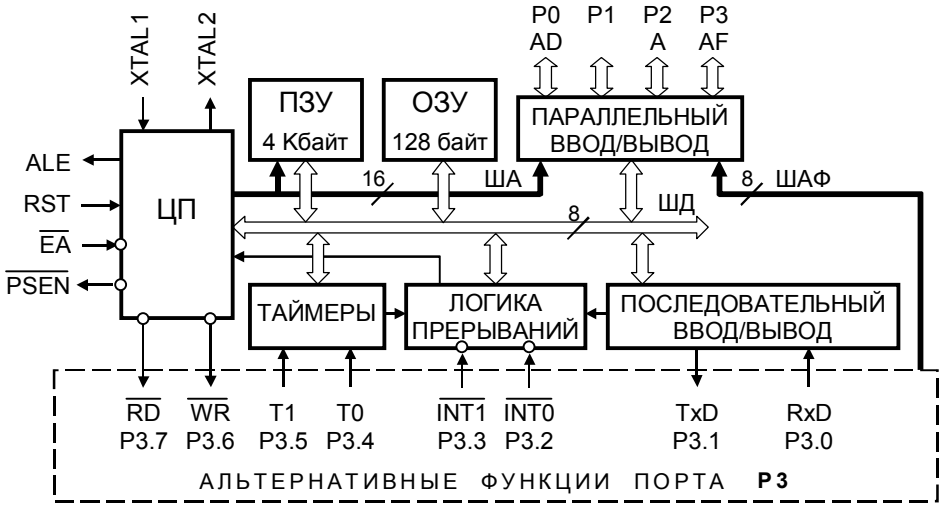
МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
РЯЗАНСКАЯ ГОСУДАРСТВЕННАЯ РАДИОТЕХНИЧЕСКАЯ АКАДЕМИЯ

Ю. П. СОКОЛОВ

МИКРОКОНТРОЛЛЕРЫ СЕМЕЙСТВА MCS-51
АРХИТЕКТУРА, ПРОГРАММИРОВАНИЕ, ОТЛАДКА



Рязань 2002

УДК 681.58:681.32

Микроконтроллеры семейства MCS-51: Архитектура, программирование, отладка: Учеб. пособие / Ю. П. Соколов, Рязан. гос. радиотехн. акад. Рязань, 2002. 72 с. ISBN-5-7722-0200-6.

Рассматриваются архитектура микропроцессоров семейства MCS-51, работа типовых периферийных устройств, вопросы проектирования, программирования и отладки микропроцессорных систем.

Предназначено студентам дневного и вечернего факультетов специальностей 2007, 2010, 2012, 2015, 2016.

Табл. 23. Ил. 37. Библиогр. 10 назв.

Микроконтроллеры, микропроцессоры, микропроцессорные системы, встроенный микропроцессор, семейство MCS-51, программирование микроконтроллеров, отладка микроконтроллеров

Печатается по решению редакционно-издательского совета Рязанской государственной радиотехнической академии.

Рецензент: кафедра радиотехнических систем Рязанской государственной радиотехнической академии (зав. кафедрой В. И. Кошелев)

Соколов Юрий Павлович

Микроконтроллеры семейства MCS-51:
Архитектура, программирование, отладка

Редактор И. П. Перехрест
Корректор Н. А. Орлова
Лицензия № 020446 от 04.03.97.
Подписано в печать . Формат бумаги 60×84 1/16.
Бумага газетная. Печать трафаретная. Усл. печ. л. 4,5.
Уч.-изд. л. 4,5. Тираж 75 экз. Заказ .
Рязанская государственная радиотехническая академия.
390005, Рязань, ул. Гагарина, 59/1.
Редакционно-издательский центр РГРТА.

ISBN-5-7722-0200-6.

Введение

Микроконтроллеры (или однокристальные микроЭВМ) представляют отдельный класс микропроцессорных систем (МПС), составные части которых (центральный процессор, память, подсистемы ввода-вывода, средства поддержки режима реального времени) размещены на одном кристалле. Они ориентированы на применение в качестве встраиваемых в изделие недорогих управляющих МПС реального времени, рабочая программа которых расположена во внутреннем ПЗУ.

Современные микроконтроллеры обладают такими вычислительными ресурсами и возможностями управления в режиме реального времени, для получения которых раньше необходимы были более дорогие многокристальные компоновки.

Периодом становления архитектуры 8-разрядных микроконтроллеров считают 1977-1979 гг., когда появились первые приборы этого класса: 8048 фирмы Intel, 3870 фирмы Mostek и 9940 фирмы Texas Instrument Inc и микроконтроллеры семейства HC05 фирмы Motorola.

В течение четырех лет, начиная с 1976 г., фирмой Intel было разработано семейство однокристальных 8-разрядных микроконтроллеров MCS-48, получивших широкое распространение. В состав семейства вошли 12 микроконтроллеров с единой базовой архитектурой, но функционально различными возможностями, реализованными непосредственно на кристалле.

В 1980 г. фирмой Intel было разработано новое семейство однокристальных 8-разрядных микроконтроллеров MCS-51, базовым представителем которого является прибор 8051. Новое семейство обеспечивает совместимость с архитектурой MCS-48, но обладает более обширным адресным пространством памяти программ и данных, усовершенствованными средствами ввода-вывода и поддержкой режима реального времени. Дальнейшее развитие получили система команд и способы доступа к отдельным элементам данных. В состав системы введены команды умножения и деления, реализован однобитовый (булев) процессор. В настоящее время семейство развивается и содержит более 50 микроконтроллеров с различными физическими возможностями. Архитектура семейства MCS-51 была определена столь удачно, что она и в настоящее время является стандартом на мировом рынке 8-разрядных микроконтроллеров.

Ряд известных фирм производят микроконтроллеры, совместимые по архитектуре и системе команд с MCS-51. Процессорное ядро MCS-51 послужило основой для создания многочисленных специализированных микроконтроллеров, в том числе и предназначенных для управления бытовой РЭА.

Появление 16- и 32-разрядных микроконтроллеров и цифровых сигнальных процессоров, значительно превосходящих 8-разрядные по производительности, не вытеснило их. Более того, по количеству модификаций 8-разрядные микроконтроллеры значительно превосходят

все остальные группы. Главная причина кроется в том, что основная область применения 8-разрядных микроконтроллеров – устройства интеллектуального управления промышленной автоматики и бытовой аппаратуры. Специфика алгоритмов управления этих устройств не требует выполнения расчетов высокой точности в жестких условиях реального времени. Основная доля операций управления состоит в преобразовании логической информации, и 8-разрядные микроконтроллеры с успехом реализуют эти задачи.

Другой причиной являются процессы глобальной информатизации. Объединение в информационные сети простых устройств управления на микроконтроллерах (уличное освещение, кассовые аппараты, сигнализация и т. п.) приводит к существенному расширению области применения 8-разрядных микроконтроллеров. Еще одна причина состоит в том, что низкая цена 8-разрядных микроконтроллеров способствует их применению в цифровых устройствах вместо ИС средней интеграции, придавая им новые качества.

Микроконтроллеры семейств MCS-51 фирмы Intel и HC05 фирмы Motorola рекомендованы типовой программой для изучения студентами радиотехнических специальностей.

Цель данного пособия – дать студентам радиотехнических специальностей основные сведения по архитектуре, функционированию и применению микроконтроллеров семейства MCS-51.

Учебное пособие включает в себя три главы и приложения. В первой главе даны архитектура и структура семейства MCS-51, состав и назначение встраиваемых периферийных устройств. Во второй главе рассматривается архитектура базового микроконтроллера, содержащего общие для всего семейства MCS-51 встраиваемые периферийные устройства. Рассмотрены работа периферийных устройств и особенности системы команд. Система команд и основные электрические характеристики двух подсемейств, аналоги которых выпускаются в странах СНГ, приведены в приложении. В третьей главе рассмотрены этапы проектирования МПС и средства для разработки программного обеспечения и отладки микропроцессорной системы. В приложении приведено описание эмулятора EMU-51, разработанного для учебных целей на кафедре радиотехнических систем РГРТА.

1. Архитектура и состав микроконтроллеров семейства MCS-51

В настоящее время семейство микроконтроллеров MCS-51 состоит из десяти подсемейств, имеющих одинаковую базовую структуру, приведенную на рис. 1, и общую систему команд.

В состав микроконтроллера входят процессор (CPU), внутреннее постоянное запоминающее устройство (IROM), внутреннее оперативное запоминающее устройство (IRAM), набор периферийных устройств. К микроконтроллеру можно подключить внешнюю постоянную память (EROM), внешнюю оперативную память (ERAM) и внешние периферийные устройства.

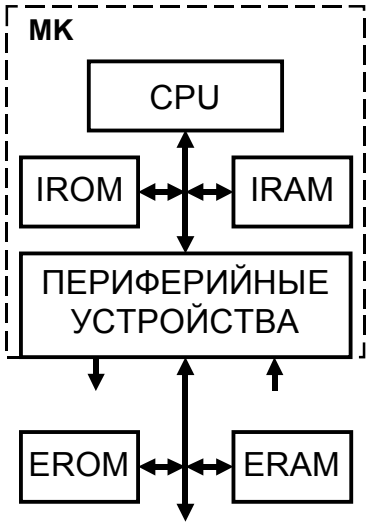


Рис. 1

Периферийные устройства предназначены для приема и выдачи данных в параллельном и последовательном коде, приема и выдачи событий, ввода и вывода аналоговых сигналов, контроля правильности работы микроконтроллера и обслуживания запросов прерывания. Подсемейства различаются емкостью внутренних запоминающих устройств, набором расположенных на кристалле периферийных устройств, быстродействием и другими характеристиками. Это открывает большие возможности при разработке систем, содержащих встроенные микроконтроллеры. В таблице 1 перечислены подсемейства и приведены обозначения типов микроконтроллеров семейства MCS-51 фирмы Intel. Микроконтроллеры одного типа выпускаются в трех исполнениях: без внутреннего ПЗУ (ROM less), с внутренним ПЗУ масочного типа (maskROM) и с внутренним программируемым ПЗУ с ультрафиолетовым стиранием (EPROM). Для стирания ПЗУ EPROM в корпусе микросхемы существует прозрачное окно. При отсутствии окна выполняется лишь однократное программирование (OTPROM) [1,2].

Микроконтроллеры подсемейств 51 и 52 изготавливаются по высококачественной n-МОП (HMOS) технологии, а остальных подсемейств – по комплементарной КМОП (CHMOS) технологии.

В странах СНГ выпускаются аналоги микроконтроллеров подсемейств 51 и 51С, выполненные по n-МОП технологии [2]:

КР1816ВЕ31 (8031АН), КР1816ВЕ51 (8051АН);

по n-МОП технологии с УФ стиранием – КМ1816ВЕ751 (8751Н);

по комплементарной КМОП технологии:

КР1830ВЕ31 (80С31ВН), КР1830ВЕ51 (80С51ВН).

В скобках указаны аналоги фирмы Intel. Основные электрические характеристики этих подсемейств приведены в приложении 1.

В таблице 2 приведен перечень всех периферийных устройств, используемых в микроконтроллерах семейства MCS-51. В таблице 3 показан состав периферийных устройств у микроконтроллеров различ-

Таблица 1

Подсемейство МК	Тип МК			Объем памяти	
	ROM less	maskROM	EPROM OTPROM	ROM, Кбайт	RAM, байт
51	8031AH*	8051AH*	8751H*	4	128
		8051AHP	8751BH	4	128
C51	80C31BH*	80C51BH*	87C51*	4	128
		80C51BHP		4	128
52	8032AH	8052AH	8752BH	8	256
C52	80C32	80C52	87C52	8	256
		80C54	87C54	16	256
		80C58	87C58	32	256
L52		80L52	87L52	8	256
		80L54	87L54	16	256
		80L58	87L58	32	256
CRX	80C51RA	83C51RA	87C51RA	8	512
		83C51RB	87C51RB	16	512
		83C51RC	87C51RC	32	512
CFX	80C51FA	83C51FA	87C51RA	8	256
		83C51FB	87C51RB	16	256
		83C51FC	87C51RC	32	256
LFX	80L51FA	83L51FA	87L51RA	8	256
		83L51FB	87L51RB	16	256
		83L51FC	87L51RC	32	256
GB	80C51GB	83C51GB	87C51GB	8	256
152	80C152JA	83C152JA		8**	256
	80C152JB			8**	256
	80C152JC	83C152JC			256

* - аналог выпускается в СНГ
** - только для 83C152JX

ных подсемейств и типов, указаны число однотипных периферийных устройств и число режимов у многорежимных устройств. Таймеры/счетчики T/C0, T/C1 и система прерываний IC имеются у микроконтроллеров всех подсемейств и они в таблице 3 не представлены.

Параллельный порт (**P**) предназначен для приема и выдачи байта данных в параллельном коде. Микроконтроллеры разных подсемейств и типов имеют от 4 до 7 восьмиразрядных портов (P0, P1,…). Некоторые порты или их отдельные разряды выполняют альтернативные функции (AF) – осуществляют прием запросов прерываний и других сигналов извне и выдают специальные сигналы управления из микроконтроллера. В графе AF табл.3 указано суммарное число разрядов параллельных портов, выполняющих альтернативные функции.

Таблица 2

1. Устройства приема и выдачи данных
1.1. Параллельный порт. **P**ort (P0, P1,…)
1.2. Последовательный порт. **S**erial **P**ort (SP)
1.3. Усовершенствованный последовательный порт. **E**nhanced **S**erial **P**ort (ESP)
1.4. Последовательный порт расширения. **S**erial **E**xpansion **P**ort (SEP)
1.5. Общий последовательный канал. **G**lobal **S**erial **C**hannel (GSC)
1.6. Блок прямого доступа к памяти. **D**irect **M**emory **A**ccessing Unit (DMA)
2. Устройства приема и выдачи событий
2.1. Таймер/счетчик. **T**imer/**C**ounter (T/C0, T/C1)
2.2. Усовершенствованный таймер/счетчик (T/C2)
2.3. Программируемая счетная матрица. **P**rogrammable **C**ounter **A**rray (PCA)
3. Устройства ввода аналоговых сигналов
3.1. Аналого-цифровой преобразователь. **A**nalog-to **D**igital **C**onverter (ADC)
4. Устройства контроля
4.1. Сторожевой таймер. **W**atch**d**og **T**imer (WDT)
4.2. Детектор падения частоты. **O**scillator **F**all **D**etect (OFD)
5. Контроллер прерываний. **I**nterrupt **C**ontroller (IC)

Последовательный порт (**SP**) используется для приема и выдачи данных в последовательном коде.

Усовершенствованный последовательный порт (**ESP**), кроме функций последовательного порта SP, позволяет выполнить автоматическое опознание адреса при работе в простейшей локальной сети и осуществить автоматический контроль формата принимаемого кадра.

Последовательный порт расширения (**SEP**) имеет аппаратные средства поддержки протокола I^2C для работы в локальной сети. Он имеется только у микроконтроллеров подсемейства GB.

Таблица 3

ПОДСЕМЕЙСТВО	P	AF	SP	ESP	SEP	GSC	DMA	T/C2	PCA	ADC	WDT	OFD
51, C51	4	24	+									
52	4	26	+					+				
C52, L52	4	26		+				+				
CRx	4	26		+				+			+	
CFx*, LFx	4	32		+				+	1			
8xC51FA, x=0;3	4	32		+				+	1			
GB	6	45		+	+			+	2	8	+	+
152**	5	24	+			+	+					
80C152JB,JD	7	40	+			+	+					

* - кроме 80C51FA, 83C51FA
** - кроме 80C152JB

Общий последовательный канал (**GSC**) предназначен для обмена данными при работе контроллера в локальной сети. Канал можно запрограммировать для работы с различными протоколами обмена. GSC имеется только у микроконтроллеров подсемейства 152.

Блок прямого доступа к памяти (**DMA**) служит для управления обменом данными между IRAM и ERAM, а также между IRAM или ERAM и буферной памятью передатчика или приемника последовательных портов SP или GSC без участия процессора. На пересылку одного байта в режиме DMA между IRAM и ERAM затрачивается два машинных цикла, а на обмен с буферной памятью - один машинный цикл.

Таймеры/счетчики T/C0 и **T/C1** имеются у микроконтроллеров всех подсемейств и служат для счета времени (таймер) или счета внешних событий (счетчик). При работе последовательного порта (SP, ESP) таймер/счетчик T/C1 используется в качестве генератора синхросигнала, частота которого определяет скорость обмена данными.

Усовершенствованный таймер/счетчик T/C2, кроме функций обычного таймера/счетчика (T/C0 или T/C1), может выполнять ряд дополнительных функций: запоминание текущего состояния; авто перезагрузку с изменением направления счета; генерацию второго синхросигнала для последовательного порта SP или ESP, позволяющую вести прием и передачу данных с разной скоростью; формирование внешнего сигнала программируемой частоты.

Программируемая счетная матрица (**PCA**) состоит из 16-разрядного таймера/счетчика, состояние которого передается в пять 16-разрядных модулей фиксации - сравнения, управляемых внешними событиями. Появление события на входе любого модуля может фиксировать состояние счетчика, сравнивать его состояние с заданным, осуществлять быстрый вывод кода состояния таймера/счетчика, формировать сигнал с широтно-импульсной модуляцией (ШИМ). Модуль может выполнять функции сторожевого таймера.

Сторожевой таймер (**WDT**) служит для предотвращения зависания микроконтроллера при зацикливании программы и представляет собой 14-разрядный счетчик машинных циклов процессора. При его переполнении микроконтроллер сбрасывается в исходное состояние. Программа должна периодически сбрасывать сторожевой таймер в нулевое состояние, не допуская его переполнения. При сбое в работе микроконтроллера очередной сброс таймера не выполняется, происходит сброс микроконтроллера, и программа начинает выполняться с нулевого адреса.

Детектор падения частоты (**OFD**) предназначен для сброса микроконтроллера и удержания его в этом состоянии при снижении тактовой частоты ниже допустимого значения.

Система прерываний (**IC**) в базовой конфигурации включает 5 источников прерываний – два внешних и три внутренних. Прерывание от каждого источника может иметь высокий или низкий приоритет и может быть маскировано. Число источников запросов прерываний опре-

деляется подсемейством. Подсемейство GB, например, имеет 15 источников прерываний, из которых 8 внешних.

Аналого-цифровой преобразователь (**ADC**) имеется только в подсемействе GB. Встроенный 8-разрядный АЦП обслуживает с помощью мультиплексора 8 аналоговых каналов. Результат преобразования аналогового сигнала каждого канала фиксируется в отдельном регистре. Время преобразования для одного канала составляет около 26 мкс при тактовой частоте микроконтроллера 12 МГц.

2. Структура базового микроконтроллера семейства MCS-51

Базовая конфигурация микроконтроллера представлена на рис. 2. Она содержит общие для всего семейства MCS-51 периферийные устройства. В состав микроконтроллера входят: 8-разрядный центральный процессор ЦП; два 16-разрядных таймера/счетчика; система двухуровневого прерывания; последовательный порт ввода/вывода; четыре 8-разрядных параллельных порта, у которых каждую из 32 линий можно настроить на ввод или вывод, а 24 линии могут выполнять альтернативные функции. Внутренние ПЗУ программ IROM и ОЗУ данных IRAM имеют минимальный объем 4 Кбайта и 128 байт соответственно. Базовая конфигурация содержит встроенные средства расширения своих ресурсов, позволяющие реализовать вне кристалла память программ EROM и память данных ERAM до 64 Кбайт каждая. Все расположенные на кристалле устройства подключены к внутренней мультиплексированной шине данных ШД. В любой момент к шине может быть подключен только один источник данных. Для этого выходы всех источников должны иметь третье состояние. Число подключаемых приемников ограничено нагрузочной способностью шины.

Для сокращения ширины физического интерфейса (числа контак-

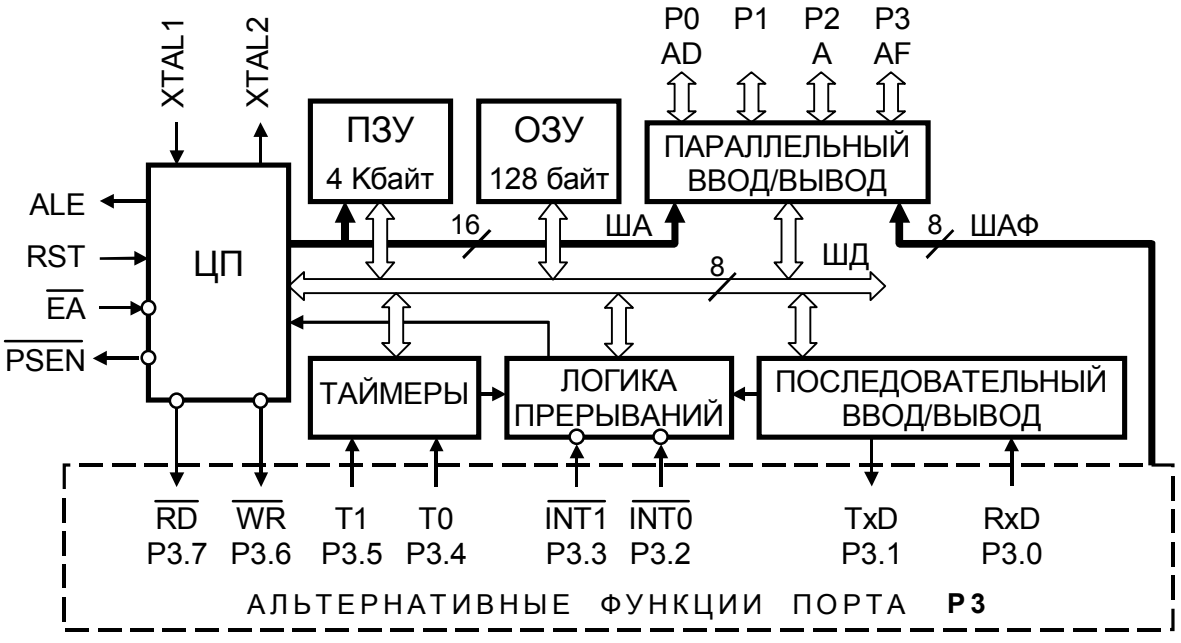


Рис. 2

тов ИС) линии параллельного порта выполняют альтернативные функции. При обращении к внешней памяти порт P0 выполняет функции совмещенной шины адреса/данных (AD), а P2 – шины старшего байта адреса (A). Все линии порта P3 выполняют альтернативные функции управления и специального ввода/вывода (AF).

2.1. Центральный процессор

Ядром микроконтроллера является центральный процессор. Он выполняет две основные функции: управление процессом преобразования (обработки) данных каждой командой и выполнение этого преобразования (обработки). Первую функцию решает блок управления, в состав которого входят: устройство управления и синхронизации, генератор тактовых импульсов, регистр команд и устройство формирования адреса; вторую - операционное устройство (рис. 3).

Центральный процессор соединен встроенной системной магистралью (16-разрядная шина адреса ША, 8-разрядная шина данных ШД и шина управления ШУ), физически совмещенной с портами P0, P2 и P3 с памятью и всеми периферийными устройствами (рис. 2).

Устройство формирования адреса выдает по шине адреса ША в память программ (CSEG) адрес очередной команды. Считанный из памяти программ код команды по шине данных ШД записывается в регистр команд. Устройство управления и синхронизации дешифрирует команду и выдает по шине управления ШУ сигналы управления всеми внутренними устройствами микроконтроллера, а также управление внешними устройствами: сигналы разрешения фиксации младшего байта адреса ALE (**A**ddress **L**ath **E**nable), чтения внешней памяти программ $\overline{\text{PSEN}}$ (**P**rogramm **S**tore **En**able). При использовании внешней памяти данных (XSEG) по линиям порта P3 выдаются сигналы записи $\overline{\text{WR}}$ (линия P3.6) и чтения $\overline{\text{RD}}$ (линия P3.7).

По выделенным линиям ШУ устройство управления получает оповещающие сигналы о ходе выполнения команды, что позволяет осуществлять ветвление в программе.

Под действием внутренних управляющих сигналов устройство формирования адреса выдает на шину адреса ША адрес следующего байта (команды или данных). Операционное устройство выбирает операнды, выполняет заданную командой операцию над операндами и выдает результат операции на шину дан-

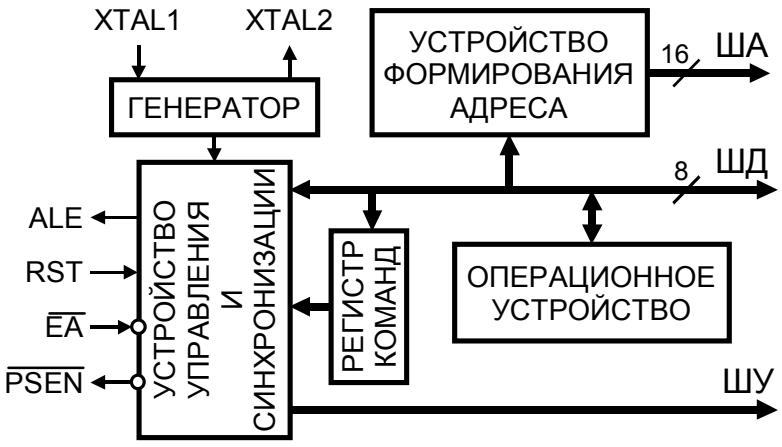


Рис. 3

ных ШД и признаки (флаги) результата на выделенные линии ШУ.

Внешний сигнал рестарта RST (**Rest**art) производит сброс микроконтроллера в исходное состояние, а сигнал $\overline{EA}$ (**E**xternal **A**ccess) управляет конфигурацией внутренней и внешней памяти программ.

2.1.1. Операционное устройство

Операционное устройство выполнено по классической схеме и служит для обработки 8-разрядных данных (рис. 4). Оно содержит арифметико-логическое устройство ALU, аккумулятор A, два программно-недоступных регистра временного хранения TMP1 и TMP2, регистр слова состояния программы PSW (**P**rogram **S**tatus **W**ord) и регистр B.

В ALU выполняется операция над двумя операндами, находящимися в регистрах временного хранения TMP1 и TMP2. При выполнении операций данные интерпретируются как целые числа без знака. Результат операции выдается на внутреннюю шину данных ШД, а во многих командах он также записывается в аккумулятор A. В командах умножения и деления роль источника и приемника информации выполняют регистры A и B.

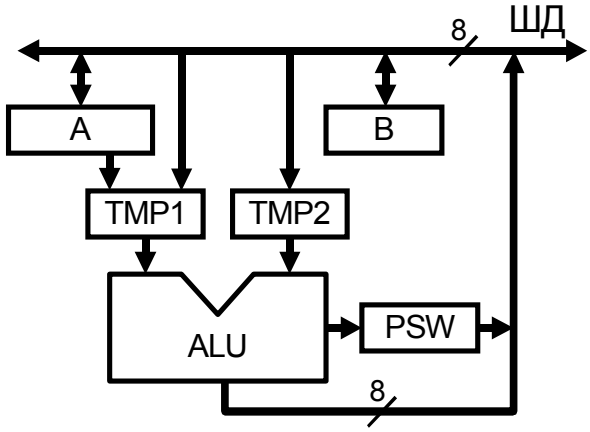


Рис. 4

При выполнении арифметических и логических операций в ALU вырабатываются признаки результата, которые записываются в регистр PSW. Все биты регистра PSW, расположенного в области BSEG регистров специальных функций SFR, программно-доступны. Их можно устанавливать и сбрасывать командами программы.

Назначение разрядов регистра PSW дано в таблице 4.

Флаг паритета P устанавливается любыми командами, в том числе и командами передачи данных, регистром-приемником которых яв-

Таблица 4. Регистр PSW (D0H)

7	6	5	4	3	2	1	0
C	**AC**	**F0**	**RS1**	**RS0**	**OV**	**P**	**-**

Назначение разрядов регистра:

PSW.0	-	Зарезервирован.
PSW.1	P	Флаг паритета.
PSW.2	OV	Флаг арифметического переполнения ALU.
PSW.3	RS0	Младший бит номера банка регистров.
PSW.4	RS1	Старший бит номера банка регистров.
PSW.5	F0	Флаг пользователя общего назначения.
PSW.6	AC	Флаг переноса из младшей тетрады ALU в старшую.
PSW.7	C	Флаг переноса из старшего разряда ALU.

ляется аккумулятор. Если в разрядах аккумулятора содержится нечетное число единиц, то P=1. Девятиразрядное слово, составленное из содержимого аккумулятора и бита P, всегда содержит четное число единиц (четный паритет).

Таблица 5

RS1	RS0	Адресуемый банк
0	0	RB0 - банк 0
0	1	RB1 - банк 1
1	0	RB2 - банк 2
1	1	RB3 - банк 3

Битами RS1 и RS0 выбирается в качестве рабочего один из четырех регистровых банков RBx (x=0...3) (таблица 5).

Флаг арифметического переполнения OV=1 устанавливается, если при сложении двух чисел с одинаковыми знаками результат имеет противоположный знак.

В командах битовой адресации аккумулятором является бит C.

2.1.2. Генератор

Синхронизация работы микроконтроллеров семейства MCS-51 может осуществляться как от внутреннего, так и от внешнего тактового генератора. На рис. 5 приведена схема внутреннего тактового генератора для подсемейств МК, выполненных по КМОП технологии. В подсемействах МК, выполненных по n-МОП технологии, отсутствует управление режимами пониженного энергопотребления. Внутренний генератор тактовых сигналов построен по классической схеме кварцевого генератора на ЛЭ DD1. Для работы генератора необходимо подключить к выводам XTAL1 и XTAL2 внешние элементы: кварцевый резонатор ZQ1 и конденсаторы C1=C2=30 пФ. При использовании внешней синхронизации выход внешнего генератора подключается к выводу XTAL1, а вывод XTAL2 остается свободным.

Режимы пониженного энергопотребления

Микроконтроллеры, выполненные по КМОП технологии, могут быть переведены в энергосберегающие режимы работы – режим холостого хода и режим микропотребления. Переход в энергосберегающие режимы работы осуществляется установкой бит IDL (Idle) или PD (Power Down) регистра PCON, расположенного в области регистров специальных функций SFR. При установке PD=1 и IDL=1 преимущество имеет бит PD. Действие этих бит показано на рис. 5.

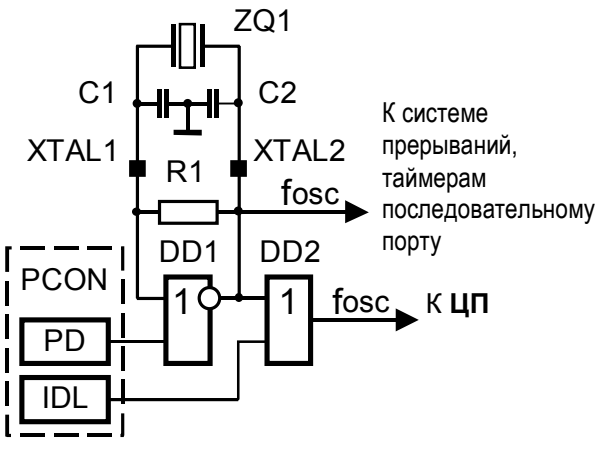


Рис. 5

Обозначение разрядов регистра PCON для микропроцессоров, выполненных по КМОП технологии, приведено в таблице 6. При n-МОП технологии режимы пониженного энергопотребления не поддерживаются.

Таблица 6. Регистр PCON (87H)

7	6	5	4	3	2	1	0
SMOD	**-**	**-**	**-**	**GF1**	**GF0**	**PD**	**IDL**

Назначение разрядов регистра:

PCON.0	IDL	Бит установки режима холостого хода (IDL=1).
PCON.1	PD	Бит установки режима микропотребления (PD=1).
PCON.2	GF0	Флаг общего назначения.
PCON.3	GF1	Флаг общего назначения.
PCON.4	-	Зарезервирован.
PCON.5	-	Зарезервирован.
PCON.6	-	Зарезервирован.
PCON.7	SMOD	Бит удвоения скорости приема/передачи последовательного канала в режимах 1, 2 и 3.

Регистр PCON содержит только один бит SMOD.

Биты общего назначения GFx (General purpose flag, x=0,1) используются по усмотрению программиста.

Режим холостого хода выполняется по команде в программе, устанавливающей бит IDL=1. В этом режиме блокируется центральный процессор, а периферийные устройства продолжают работать (рис.5). Содержимое программного счетчика PC и данных в областях RSEG и DSEG сохраняются. Ток потребления уменьшается в четыре раза.

Выйти из режима холостого хода можно активизацией разрешенного прерывания или аппаратным сбросом по входу RST. После исполнения команды RETI в программе обслуживания прерывания или подачи сигнала RST устанавливается бит IDL=0 и следующей будет выполнена команда, адрес которой сохранен в PC.

Режим микропотребления осуществляется программно установкой бита PD=1. Работа внутреннего тактового генератора блокируется, что приводит к прекращению работы всех узлов МК. Содержимое внутреннего ОЗУ данных сохраняется.

В этом режиме напряжение питания МК можно снизить до 2 В. Ток потребления падает до 10-15 мкА. Перед выходом из режима напряжение питания должно быть восстановлено до номинального значения.

Единственной возможностью выхода из режима микропотребления является аппаратный сброс по входу RST. В этом случае программный счетчик PC обнуляется и программа выполняться сначала.

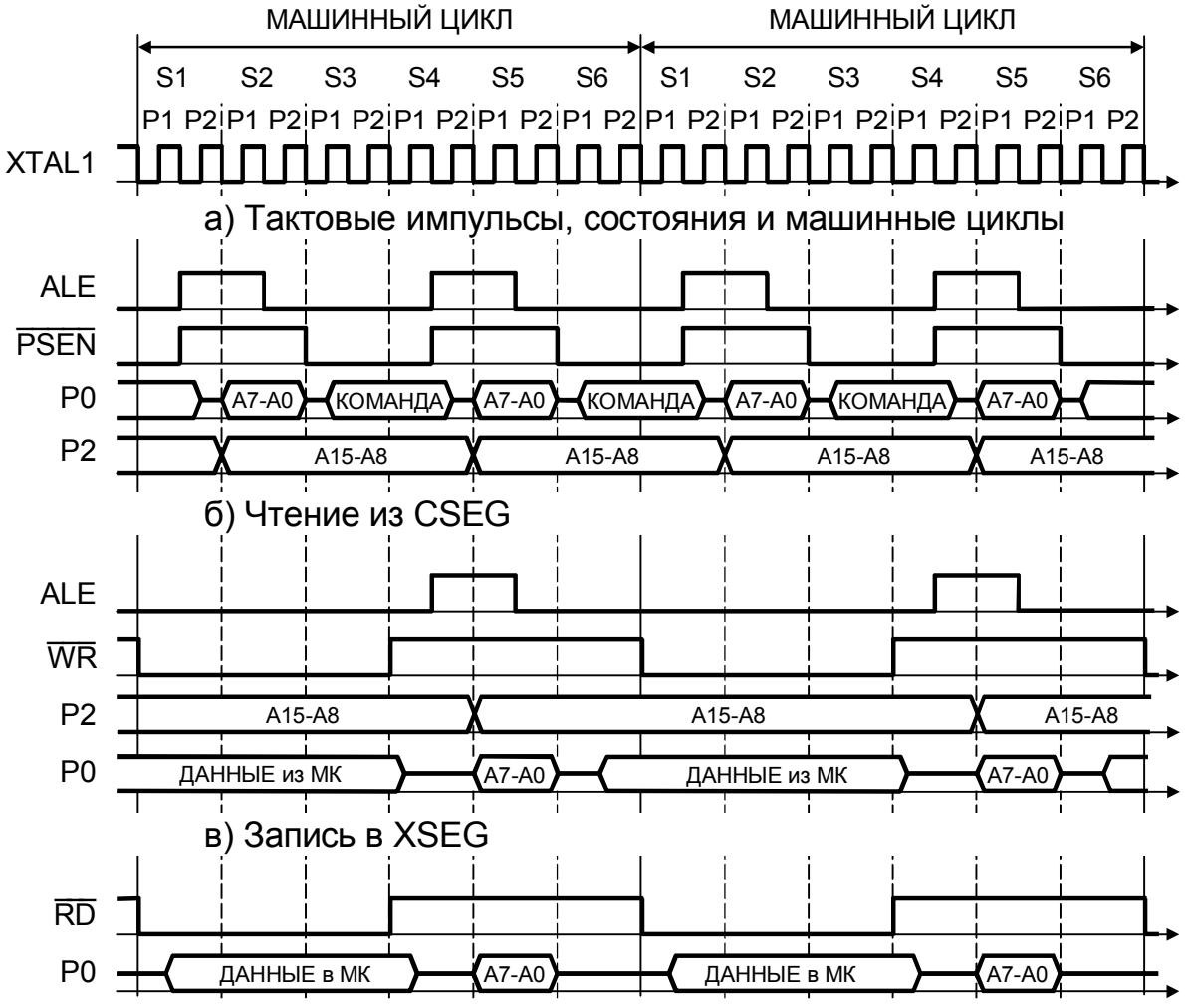
2.1.3. Устройство управления и синхронизации

Устройство управления и синхронизации представляет собой цифровой автомат, формирующий сигналы для управления всеми внутренними и внешними узлами микроконтроллера. Внешние сигналы управления имеют следующее назначение:

ALE	Разрешение фиксации младшего байта адреса во внешнем регистре (**A**ddress **L**ath **E**nable).
$\overline{\text{PSEN}}$	Чтение из внешней памяти программ CSEG (**P**rogramm **S**tore **En**able).
$\overline{\text{RD}}$	Чтение из внешней памяти данных XSEG.
$\overline{\text{WR}}$	Запись во внешнюю память данных XSEG.
$\overline{\text{EA}}$	Конфигурирование памяти программ (**E**xternal **A**ccess).
RST	Внешний сигнал сброса (RST=1) микроконтроллера в исходное состояние (**Res**tart).

Управляющий автомат синхронизируется тактовыми импульсами генератора. Он имеет 6 состояний (S1...S6), образующих машинный цикл (рис. 6,а). Каждое из состояний автомата содержит две фазы (P1 и P2). Обычно в фазе P1 выполняется операция в АЛУ, а в фазе P2 межрегистровый обмен данными.

Машинный цикл имеет фиксированную длительность, равную 12 периодам частоты (f_{OSC}) внутреннего или внешнего генератора и служит, в основном, для целей внутреннего микропрограммного управления. При описании последовательности сигналов или событий фазам в



а) Тактовые импульсы, состояния и машинные циклы

б) Чтение из CSEG

в) Запись в XSEG

г) Чтение из XSEG

Рис. 6

машинном цикле присваиваются номера с S1P1 по S6P2 (рис. 6,а). Цикл выполнения каждой команды состоит из одного, двух и четырех машинных циклов.

По срезу ALE передаваемый через порт P0 младший байт адреса фиксируется во внешнем регистре. Старший байт адреса передается через порт P2.

За один машинный цикл осуществляется два обращения к CSEG. Считываемая из CSEG информация поступает в устройство управления. Чтение производится по фронту импульса $\overline{PSEN}$ (фаза S1P1 и S4P1) (рис. 6,б). Первый байт команды записывается в регистр команд и дешифрируется устройством управления. Устройство управления формирует на шине управления последовательность управляющих сигналов, необходимую для выполнения команды.

Если команда однобайтовая, то второй считанный байт игнорируется. Он будет повторно считан в следующем машинном цикле. Второй байт двухбайтовых команд, а также второй и последующие байты трех- и четырехбайтовых команд записываются либо в устройство формирования адреса, либо в программно-недоступные регистры устройства управления или операционного устройства.

Цикл обращения к внешнему CSEG автоматически инициируется всякий раз при выходе адреса за пределы адресного пространства внутреннего ПЗУ, а также при отключении внутреннего ПЗУ ($\overline{EA}$=0).

Цикл обращения к внешней памяти данных (XSEG) инициируется командой MOVX. При записи в XSEG данные сохраняются истинными во время действия низкого уровня сигнала $\overline{WR}$ (рис. 6,в). Данные из XSEG считываются в аккумулятор операционного устройства по фронту импульса $\overline{RD}$ (фаза S3P2) (рис. 6,г).

2.1.4. Устройство формирования адреса

Устройство формирования адреса предназначено для формирования текущего 16-разрядного адреса памяти программ (CSEG) и адреса внешней памяти данных (XSEG). В состав устройства входят 16-разрядные буфер BUF, регистр указателя данных DPTR, регистр PC, схема инкремента INC PC, адресный регистр Addr RG и 8-разрядный указатель стека SP (рис. 7).

Буфер BUF осуществляет связь между 16-разрядной внутренней шиной ШВ и 8-разрядной шиной данных ШД, обеспечивая запись, хранение и коммутацию данных.

Регистр указателя данных DPTR служит для хранения 16-разрядного адреса внешней памяти данных. Он состоит из двух 8-разрядных регистров DPH и DPL, расположенных в области регистров специальных функций SFR. Регистры DPH и DPL программно доступны и могут исполь-

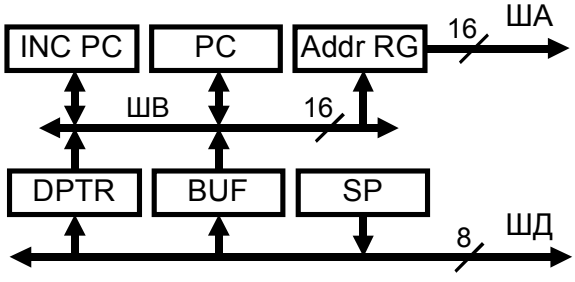


Рис. 7

зоваться в качестве двух независимых РОН, если нет необходимости в хранении 16-разрядного адреса внешней памяти.

Регистр PC адресует память программ и содержит текущий 16-разрядный адрес байта команды.

Схема инкремента INC PC увеличивает текущее значение 16-разрядного адреса памяти программ на единицу.

Регистр адреса памяти Addr RG предназначен для хранения и выдачи на внутреннюю шину адреса ША прямого 16-разрядного адреса памяти программ или 8/16-разрядного адреса внешней памяти данных

Указатель стека SP хранит текущий 8-разрядный адрес вершины стека, расположенного во внутреннем ОЗУ (DSEG). Перед записью байта информации в стек содержимое SP увеличивается на единицу и по этому адресу производится запись. При чтении из стека информация читается по адресу, хранимому в SP, и после чтения содержимое SP уменьшается на единицу.

2.2. Организация памяти

Микроконтроллеры семейства MCS-51 выполнены по Гарвардской архитектуре, в которой, в отличие Принстонской архитектуры фон Неймана, используется принцип независимости сред для хранения программ и данных. Всего имеются 5 типов пространств памяти, из которых 4 являются областями памяти данных [3]:

DSEG (**D**ata **Seg**ment) – пространство внутренней памяти данных,
RSEG (**R**egister **Seg**ment)– пространство регистров,
BSEG (**B**ite **Seg**ment) – пространство битовой памяти данных,
XSEG (e**X**ternal **Seg**ment) – пространство внешней памяти данных,
CSEG (**C**ode **Seg**ment) – пространство программного кода.

2.2.1. Память данных

Рис. 8

Внутренняя память данных DSEG располагается во внутреннем ОЗУ и может иметь объем 128 или 256 байт. В типовой конфигурации DSEG имеет объем 128 байт и располагается в нижней области данных, занимающей адресное пространство 00h…7Fh. В этой области DSEG можно использовать прямую и косвенную адресацию операндов. Область памяти с адресами 80h…FFh занимают регистры специальных функций SFR (**S**pecial **F**unction **R**egisters). В ней разрешена только прямая адресация (рис. 8).

При объеме в 256 байт имеется как нижняя, так и верхняя области данных. Адресные пространства верхнего DSEG и регистров специальных функций SFR совпадают и для их различия использованы разные способы адресации. Обращение к верхней области DSEG осуществляется только командами косвенной адресации, а к регистрам специальных функций – прямой. Карта памяти нижней области DSEG приведена на рис. 10,а.

Пространство регистров RSEG содержит 32 регистра, сгруппированных в 4 регистровых банка (**R**egister **B**anks) RB0...RB3 по 8 регистров (R0...R7) в каждом (рис. 9,б). Все регистры выполняют общецелевые функции промежуточного хранения данных, а два регистра R0 и R1 каждого банка еще и функцию 8-разрядного указателя данных в командах косвенной адресации.

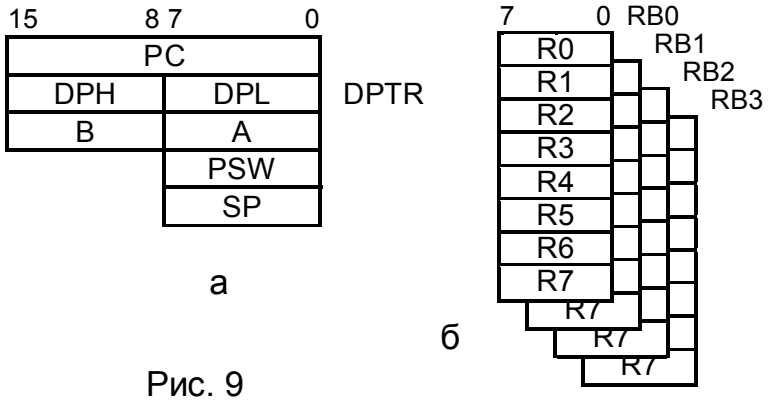


Рис. 9

Регистровые банки переключаются полем RS слова состояния программы PSW. Физически RSEG расположен в области DSEG с адресами 00h...1Fh (рис. 10,а). Такое совмещение позволяет двояко интерпретировать содержимое ячейки, что дает возможность программисту выбрать наиболее подходящий вариант для уменьшения объема и повышения скорости исполнения программы.

В состав регистровой памяти входят также следующие программно-доступные регистры (рис. 9,а):

PC — 16-разрядный счетчик команд (**P**rogram **C**ounter). Содержит адрес ячейки CSEG, подлежащей чтению.

DPTR — 16-разрядный указатель данных (**D**ata **P**oin**t**e**r**). Состоит из двух 8-разрядных регистров, содержащих старший (**H**igh) DPH и младший (**L**ow) DPL байты.

A — Аккумулятор.

B — Регистр общего назначения. Используется также в командах умножения и деления.

PSW — Регистр состояния программы (**P**rogram **S**tatus **W**ord). Содержит признаки, формируемые аккумулятором.

SP — Указатель стека (**S**tack **P**ointer). Содержит адрес вершины стека.

Все эти регистры, кроме PC, расположены в области регистров специальных функций SFR (рис. 10,б, таблица 7). Регистр PC находится в центральном процессоре.

Пространство битов BSEG предназначено для хранения булевых данных. Оно имеет объем в 256 бит и разделено на две области по

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	R0^0	R1^0	R2^0	R3^0	R4^0	R5^0	R6^0	R7^0	R0^1	R1^1	R2^1	R3^1	R4^1	R5^1	R6^1	R7^1
1	R0^2	R1^2	R2^2	R3^2	R4^2	R5^2	R6^2	R7^2	R0^3	R1^3	R2^3	R3^3	R4^3	R5^3	R6^3	R7^3
2	00-07	08-0F	10-17	18-1F	20-27	28-2F	30-37	38-3F	40-47	48-4F	50-57	58-5F	60-67	68-6F	70-77	78-7F
3																
4																
5																
6																
7																

а

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	P0 80-87	SP	DPL	DPH				PCON	TCON 88-8F	TMOD	TL0	TL1	TH0	TH1		
9	P1 90-97								SCON 98-9F	SBUF						
A	P2 A0-A7								IE A8-AF							
B	P3 B0-B7								IP B8-BF							
C	C0-C7								C8-CF							
D	PSW D0-D7								D8-DF							
E	ACC E0-E7								E8-EF							
F	B F0-F7								F8-FF							

б

Рис. 10

128 бит каждая. Одна область BSEG с адресами 00h...7Fh физически совмещена с шестнадцатью ячейками памяти DSEG, имеющими адреса 20h...2Fh, и является областью общего назначения (рис. 10,а). Другая область с адресами 80h..FFh физически совмещена с областью регистров специальных функций SFR, что обеспечивает доступ к отдельным разрядам регистров (рис. 10,б). В битовом пространстве определена только прямая адресация bit и действует свой набор команд, опреде-

ляемых битовым процессором. Роль аккумулятора битового процессора выполняет бит C регистра PSW.

Пространство внешней памяти данных XSEG может иметь объем 64 Кбайт и реализуется внешними средствами (рис. 12). Связь с XSEG поддерживается единственной командой MOVX, имеющей два типа адресации: косвенную регистровую по DPTR и страничную с номерами страниц в порту P2 и смещением в R0, R1. Это позволяет рассматривать организацию внешней памяти как область с линейной или со страничной структурой адресации. Адресное пространство памяти данных внешнего XSEG и внутреннего DSEG не связаны между собой.

2.2.2. Регистры специальных функций

Область регистров специальных функций SFR содержит регистры, обслуживающие порты, таймеры/счетчики, систему прерываний и энергосбережения. Здесь также находится аккумулятор A и регистр B, служащий расширением аккумулятора в командах умножения и деления. В других командах регистр B выполняет общецелевые функции. Список регистров специальных функций с адресами для типовой конфигурации микроконтроллеров семейства MCS-51 приведен в таблице 7, а карта памяти представлена на рис. 10,б.

Таблица 7

A	E0h	Аккумулятор
B	F0h	Регистр B
PSW	D0H	Слово состояния программы
SP	81h	Указатель стека
DPTR		Указатель данных
DPL	82h	Младший байт DPTR
DPH	83h	Старший байт DPTR
P0	80h	Порт P0
P1	90h	Порт P1
P2	A0h	Порт P2
P3	B0h	Порт P3
IP	B8h	Управление приоритетом прерываний
IE	A8h	Управление разрешением прерываний
TMOD	89h	Режимы таймеров/счетчиков
TCON	88h	Управление таймерами/счетчиками
TH0	8Ch	Старший байт TC0
TL0	8Ah	Младший байт TC0
TH1	8Dh	Старший байт TC1
TL1	8Bh	Младший байт TC1
SCON	98h	Управление последовательным портом
SBUF	99h	Буфер данных последовательного порта
PCON	87h	Управление режимами энергосбережения

При включении микроконтроллера или аппаратном сбросе (RST=1) в указатель стека SP записывается значение 07h (для обеспечения совместимости с семейством MCS-48), триггеры всех разрядов параллельных портов устанавливаются в состояние 1, переводя все линии в режим «Чтение». Все используемые разряды остальных регистров устанавливаются в состояние 0, а состояние неиспользуемых (зарезервированных) разрядов и регистра SBUF не определено.

2.2.3. Память программ

Программный счетчик может адресовать память программ CSEG с адресным пространством 64 Кбайт. Часть этой памяти (в базовой конфигурации 4 Кбайт) размещено во внутреннем ПЗУ и образует внутреннюю память программ. Оставшаяся часть может быть реализована внешними средствами и называется внешней памятью программ. Конфигурирование памяти программ осуществляется управлением по входу $\overline{EA}$. При $\overline{EA}$ =0 доступ к внутренней памяти запрещается, и микроконтроллер обращается только к внешней памяти, адрес которой начинается с 0000h (рис. 11,а). При $\overline{EA}$ = 1 адресное пространство внешней памяти программ является продолжением адресного пространства внутренней памяти программ (рис. 11,б). Обращение к внешней памяти происходит автоматически всякий раз при превышении текущим адресом максимального адреса внутренней памяти (0FFFh для объема 4 Кбайт). По этой причине внутренняя и внешняя память программ представляют собой единое линейное пространство.

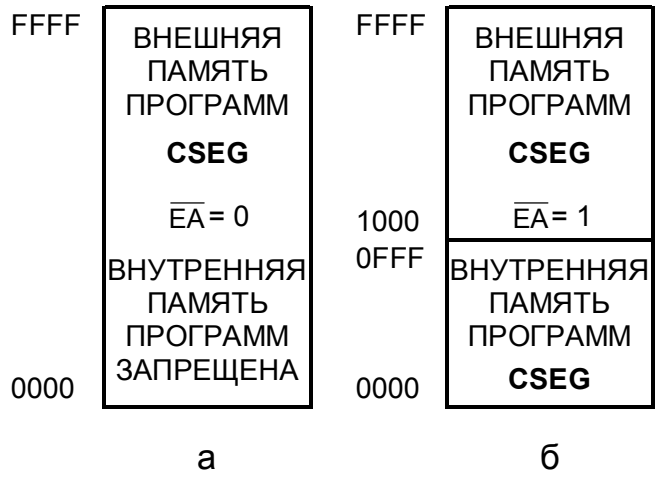


Рис. 11

В начале CSEG расположена таблица векторов прерывания. Каждому источнику прерывания соответствует свой адрес ячейки памяти (вектор прерывания). Он загружается в программный счетчик PC при обслуживании прерывания. Микроконтроллеры базовой конфигурации имеют стартовый адрес и пять векторов прерывания:

RESET	0000h	Стартовый адрес при сбросе микроконтроллера.
EXTI0	0003h	Внешнее прерывание 0.
TIMER0	000Bh	Прерывание таймера/счетчика 0.
EXTI1	0013h	Внешнее прерывание 1.
TIMER1	001Bh	Прерывание таймера/счетчика 1.
SINT	0023h	Прерывание последовательного порта.

2.2.4. Внешняя память программ и данных

Схема включения внешних CSEG и XSEG показана на рис.12. Она содержит 8-разрядный параллельный регистр DD1 на одноступенчатых триггерах с прямым потенциальным управлением, внешние ПЗУ DD2 (CSEG) и ОЗУ DD3 (XSEG). Внешняя 8-разрядная (D[7...0]) шина данных ШД формируется из линий порта P0. Младший байт адреса по стробу ALE фиксируется в регистре DD1 и вместе со старшим байтом, выдаваемым портом P2, образует 16-разрядную шину адреса ША (рис. 13).

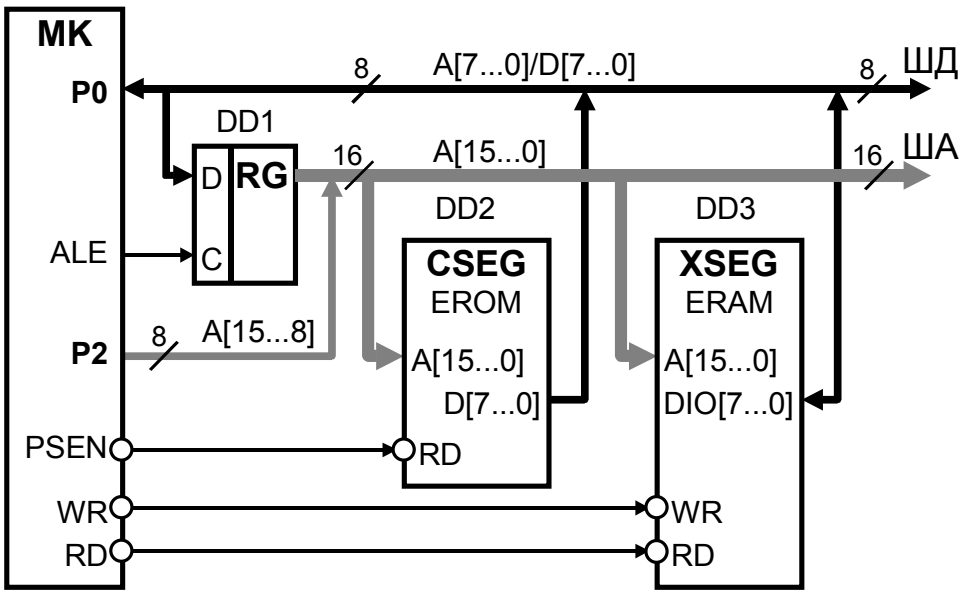


Рис. 12

При низком уровне сигнала $\overline{PSEN}$ выполняется чтение команды из CSEG, а при высоком - выходы DO[7...0] (**D**ata **O**utput) должны перейти в третье состояние (рис. 6,б).

Низким уровнем сигнала $\overline{WR}$ производится запись байта с шины данных ШД в XSEG (рис. 6,в), а низким уровнем сигнала $\overline{RD}$ чтение байта из XSEG на ШД (рис. 6,г). При высоких уровнях сигналов $\overline{WR}$ и $\overline{RD}$ двунаправленные выводы DIO[7...0] (**D**ata **I**nput/**O**utput) внешнего ОЗУ DD3 должны находиться в третьем состоянии.

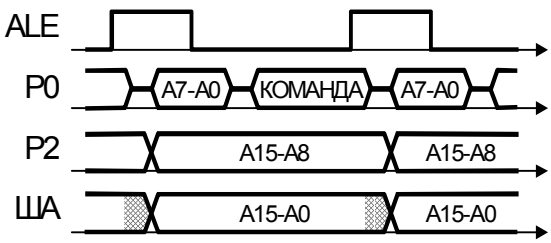


Рис. 13

Данные из CSEG читаются в устройство управления центрального процессора, а XSEG обменивается данными с операционным устройством. Единственная команда MOVC позволяет читать данные из CSEG в аккумулятор операционного устройства. Это позволяет использовать программную память для размещения констант, доступных для операционного устройства.

2.3. Параллельные порты

Базовая конфигурация МК содержит четыре квазидвунаправленных 8-разрядных порта P0...P3 с возможностью независимого индивидуального задания направления передачи каждой линии. Они обеспечивают обмен информацией микроконтроллера с внешними устройствами и выполнение альтернативных функций, таких как обращение к внешней памяти, прием запросов прерываний, управление работой счетчиков/таймеров.

Каждый из портов содержит 8-разрядный параллельный регистр данных с логикой управления и драйвер - выходной каскад, соединяющий порт с внешними линиями. Регистры данных находятся в области BSEG регистров специальных функций и, следовательно, имеют как байтовую, так и битовую адресацию. Это позволяет обращаться к портам как к обычным ячейкам памяти или обслуживать каждую линию порта командами битового процессора независимо от других линий того же порта. С учетом этого обстоятельства схемотехника портов рассматривается на уровне одной линии порта.

2.3.1. Драйверы портов

Схемотехника драйверов портов, кроме порта P0, зависит от используемой технологии изготовления микроконтроллера.

<u>Драйвер разряда порта для n-МОП</u> технологии (рис. 14) содержит выходной ключ на транзисторах VT2 и VT3 и схему ускоренного заряда емкости нагрузки на элементах DD1, DD2, DD3 и транзисторе VT1. Маломощный транзистор VT2 выполняет функцию подтягивающего резистора. Емкость нагрузки представляет суммарную паразитную емкость, подключенную к выходу P1.x драйвера (x=0…7 – номер линии порта). При переходе сигнала на входе 1 драйвера из лог.1 в лог.0 транзистор VT3 запирается и напряжение на выходе P1.x драйвера устанавливается в результате заряда емкости нагрузки через транзистор VT2. Это приводит к затягиванию переходного процесса установления лог.1 на выходе драйвера и, следовательно, к снижению быстродействия микроконтроллера. В этом случае схема ускоренного заряда формирует импульс, который открывает на время $2t_{OSC}$ мощный транзистор VT1, ток которого на два порядка больше тока транзистора VT2, что приводит к быстрому заряду емкости. Разряд емкости нагрузки происходит через открытый транзистор VT3.

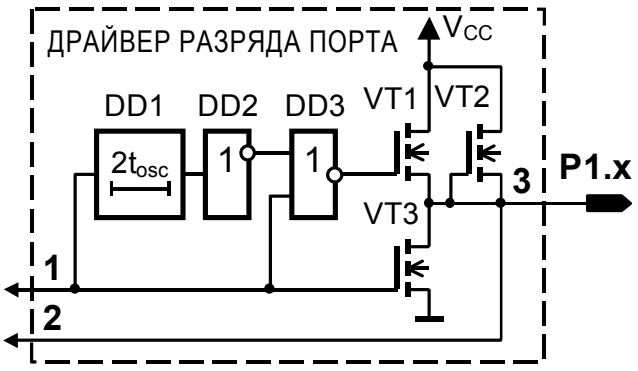


Рис. 14

Данные, считываемые в микроконтроллер с вывода P1.x при открытом транзисторе VT3, будут искажены. При этом может нарушиться

физическая целостность системы. По этой причине при чтении транзистор VT3 должен быть заперт.

Драйвер разряда порта для КМОП технологии (рис. 15) содержит выходной ключ на комплементарных транзисторах VT2 и VT3, схему ускоренного заряда емкости нагрузки (DD1, DD2, DD3 и VT1) и триггер на транзисторе VT4 и инверторе B2. Инвертор B1 восстанавливает фазу сигнала, считываемого с выхода драйвера. Транзисторы ключа рассчитаны на малый ток нагрузки в статическом режиме и не обеспечивают большого тока в переходном режиме. По этой причине здесь также используется схема ускоренного заряда емкости нагрузки. Триггер удерживает состояние 1 на выходе драйвера после запирания транзистора VT1. Ток транзистора VT4 на порядок больше тока транзистора VT2. При чтении сигнала с контакта P1.x в микроконтроллер триггер выполняет функцию приемника с линии. Наличие гистерезиса в его передаточной характеристике позволяет бороться с помехами, уровень которых не превышает порог срабатывания триггера. В режиме чтения транзистор VT3 также должен быть заперт.

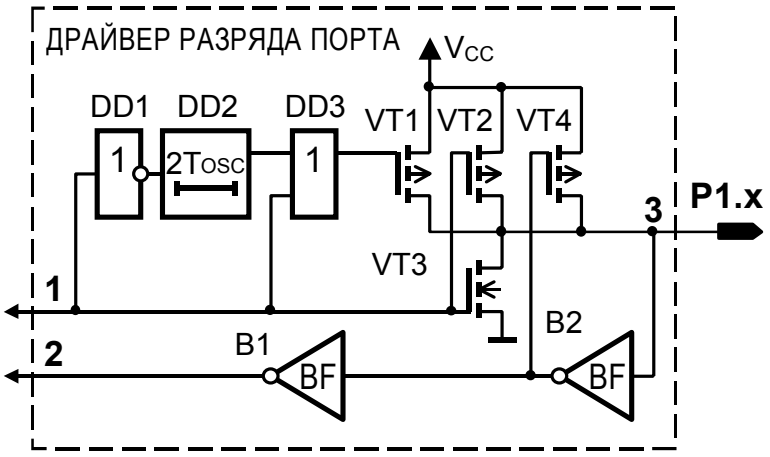


Рис. 15

2.3.2. Особенности архитектуры параллельных портов P0...P3

Каждый разряд порта содержит триггер-защелку (одноступенчатый D-триггер, входящий в состав 8-разрядного регистра данных порта), логику управления и драйвер.

Сигналы управления логикой каждого разряда порта вырабатываются устройством управления ЦП при выполнении команд и подаются в него по выделенным линиям шины управления ШУ (линии «Управление», «Чтение защелки», «Запись в защелку», «Чтение вывода»).

Разрядная «линия внутренней шины данных» соединяет каждый разряд порта с соответствующей разрядной линией встроенной двунаправленной шины данных ШД микроконтроллера для организации двухстороннего обмена.

Для выполнения альтернативных функций каждый разряд порта P0 линиями «Адрес/Данные» соединен с соответствующими разрядами младшего байта шины адреса ША и шины данных ШД. Каждый разряд порта P2 линиями «Адрес» соединен с соответствующими разрядами старшего байта шины адреса ША, а разряды порта P3 - линиями «Альтернативная функция входа» и «Альтернативная функция выхода», об-

разующие шину альтернативных функций ШАФ соединены с соответствующими входами и выходами встроенных периферийных устройств и центрального процессора (рис. 2).

Порт P0 работает как в основном, так и альтернативном режиме. В альтернативном режиме при выполнении команд MOVC и MOVX через него выводится младший байт адреса и производится чтение данных из внешних CSEG и XSEG, а также запись данных в XSEG.

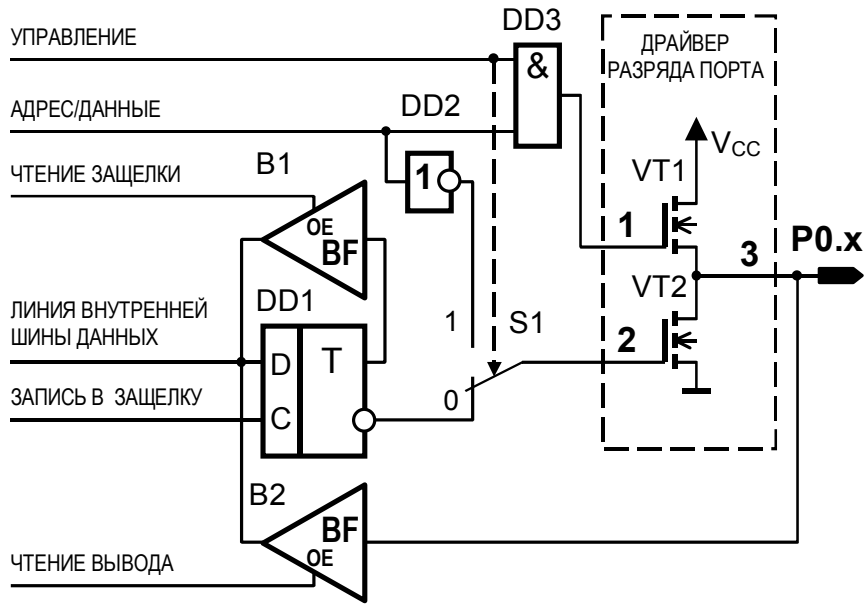


Рис. 16

Каждый разряд порта P0 (рис. 16) содержит триггер-защелку DD1 с логикой управления основной (буферы B1 и B2) и альтернативной (элементы DD2 и DD3) функциями и драйвер (транзисторы VT1 и VT2). Выбор основной или альтернативной функции порта выполняется мультиплексором-переключателем S1 под действием сигнала «Управление».

В основном режиме (сигнал «Управление»=0) вход 2 драйвера соединен с инверсным выходом триггера. Транзистор VT1 заперт и для работы выходного ключа на транзисторе VT2 необходимо включить внешний подтягивающий резистор.

При записи данных в порт внутренним сигналом «Запись в защелку» данные с линии внутренней шины данных записываются в триггер и появляются на выводе P0.x драйвера. Данные в триггере и на выходе драйвера сохраняются до следующей записи.

Состояние триггера может быть прочитано на линию внутренней шины данных сигналом «Чтение защелки», который переводит выход трехстабильного буфера B1 в нормальное состояние. Этот режим чтения используется в командах модификации содержимого порта.

Чтение данных с вывода порта на линию внутренней шины данных выполняется сигналом «Чтение вывода» при закрытом транзисторе VT2. Для запирания транзистора VT2 и перевода разряда порта в режим чтения, необходимо записать в его триггер 1.

Альтернативные функции включаются автоматически лишь во время выполнения команд MOVC и MOVX. В это время сигналом «Управление»=1 устройство управления ЦП переключает вход 2 драй-

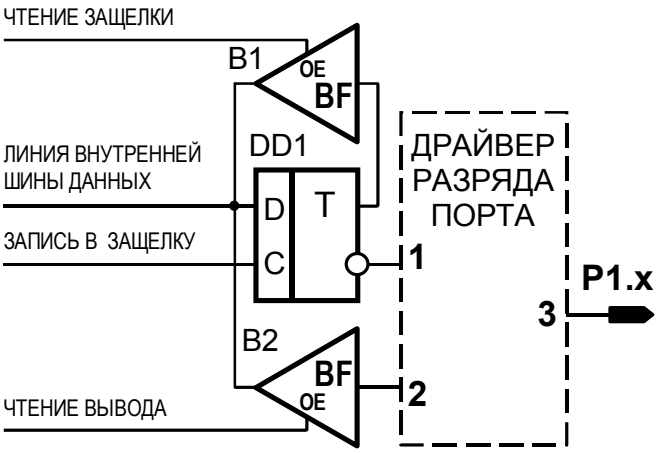


Рис. 17

вера к выходу инвертора DD2 и по 8 младшим разрядам внутренней шины адреса ША[7...0], одной из линий которой является линия «Адрес/Данные», выдает младший байт адреса и байт данных. Если в режиме альтернативных функций выводятся адреса или данные, то работают оба транзистора драйвера, образуя комплементарный ключ. При чтении данных из CSEG или XSEG сигналом «Адрес/Данные»=1 запирается транзистор VT2, переводя драйвер порта в режим ввода. Ввод данных осуществляется обычным образом через буфер B2 сигналом «Чтение вывода». Содержимое триггеров всех разрядов порта P0 при выполнении альтернативной функции не изменяется.

Порт P1 работает только в основном режиме, обеспечивая функции ввода/вывода данных. Каждый разряд порта P1 (рис.17) содержит триггер-защелку с логикой управления и драйвер.

Порт P2 работает как в основном, так и альтернативном режиме (рис. 18). Основной режим порта P2 (сигнал «Управление»=0) аналогичен основному режиму порта P0.

В альтернативном режиме (сигнал «Управление»=1) через порт P2 выдается старший байт адреса при обращении командами MOVX и MOVC к внешним CSEG и XSEG. Линия «Адрес» является одной из линий старшего байта внутренней шины адреса ША[15...8]. Содержимое всех триггеров порта P2 при выполнении альтернативной функции сохраняется.

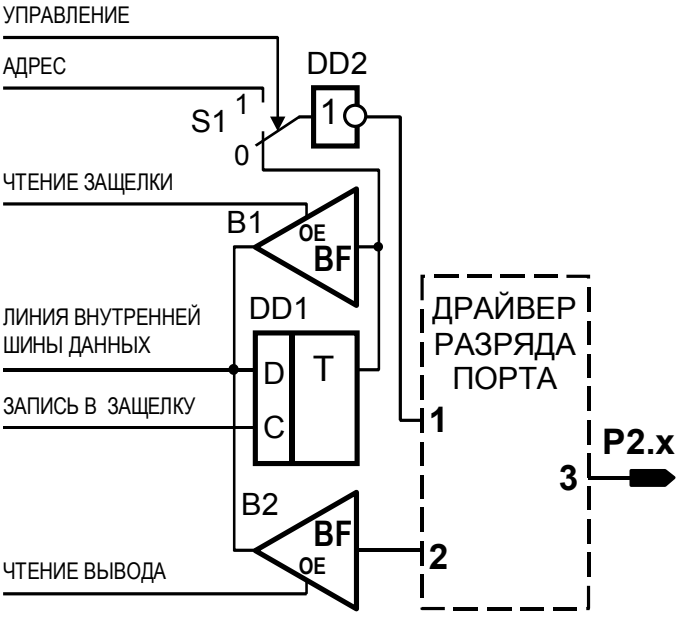


Рис. 18

Порт P3, кроме основной функции, аналогичной основной функции порта P0, выполняет альтернативную функцию по управлению циклами обмена с внешними CSEG и XSEG и другими специальными функциями аппаратного уровня (рис. 19). Альтернативная функция любой линии порта реализуется только в том случае, если в соответствующем этой ли-

нии разряде триггера записана 1. В противном случае на выходе разряда порта будет установлен 0.

Каждая линия порта P3 имеет свою альтернативную функцию:

P3.0	RxD	Вход приемника последовательного канала.
P3.1	TxD	Выход передатчика последовательного канала.
P3.2	$\overline{INT0}$	Вход 0 запроса на прерывание.
P3.3	$\overline{INT1}$	Вход 1 запроса на прерывание.
P3.4	T0	Внешний вход таймера/счетчика 0.
P3.5	T1	Внешний вход таймера/счетчика 1.
P3.6	$\overline{WR}$	Строб записи в XSEG.
P3.7	$\overline{RD}$	Строб чтения XSEG.

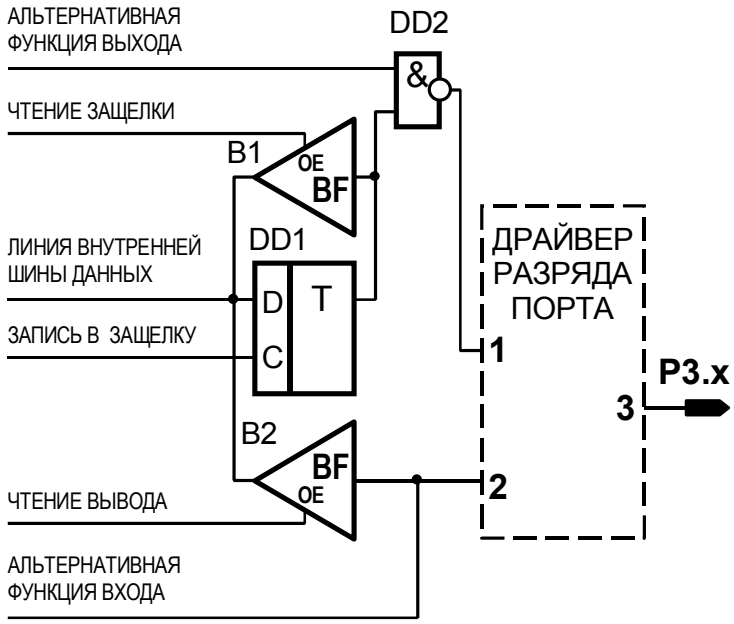


Рис. 19

Вывод альтернативной информации на выход драйвера осуществляется по линии «Альтернативная функция выхода», а ввод – по линии «Альтернативная функция входа».

При аппаратном сбросе (RESET=1) все разряды триггеров портов устанавливаются в состояние 1, а порты - в режим «Ввод».

При отсутствии внешнего CSEG командой MOVC соответствующие альтернативные функции не исполняются.

2.4. Последовательный порт

При работе последовательного порта используются две линии порта P3: P3.1 – RxD (**R**eceiver **D**ata) линия приема данных и P3.2 – TxD (**T**ransmitter **D**ata) линия передачи данных. Входные и выходные данные хранятся в буферном регистре SBUF с адресом 99h, расположенным в области SFR. Вообще, по данному адресу находятся два сдвигающих (параллельно-последовательных) регистра. Один из них – «передатчик» предназначен для передачи байта данных с внутренней шины данных ШД на линию TxD. Он загружается любой командой, использующей SBUF в качестве регистра назначения. Другой – «приемник» служит для чтения данных с линии RxD на шину ШД. К нему обращаются любой командой, в которой SBUF является регистром-источником данных.

Управление работой последовательного порта осуществляется регистром SCON (**S**erial **Con**trol), расположенным в области SFR по ад-

ресу 98h. Обозначение разрядов регистра приведено в таблице 8. Кроме того, в управлении скоростью передачи данных участвует бит SMOD регистра PCON.

Таблица 8. Регистр SCON (98H)

7	6	5	4	3	2	1	0
SM0	**SM1**	**SM2**	**REN**	**TB8**	**RB8**	**TI**	**RI**

Назначение разрядов регистра:

SCON.0	RI	Флаг прерывания приемника.
SCON.1	TI	Флаг прерывания передатчика.
SCON.2	RB8	Девятый бит принимаемых данных.
SCON.3	TB8	Девятый бит передаваемых данных.
SCON.4	REN	Разрешение приема данных.
SCON.5	SM2	Разрешение многопроцессорной работы.
SCON.6	SM1	Младший бит номера режима.
SCON.7	SM0	Старший бит номера режима.

Последовательный порт можно запрограммировать на работу в одном из четырех режимов установкой разрядов SM0, SM1 регистра SCON, как показано в таблице 9. Во всех режимах передача инициируется любой командой, использующей SBUF в качестве регистра назначения.

Таблица 9. Режимы работы последовательного порта

Режим	SM0	SM1	Наименование	Кадр, бит	Скорость передачи
0	0	0	Синхронный	8	$f_{OSC}/12$
1	0	1	Асинхронный	10	Переменная (TC1)
2	1	0	Асинхронный	11	$f_{OSC}/32$ или $f_{OSC}/64$
3	1	1	Асинхронный	11	Переменная (TC1)

Флаг прерывания приемника RI устанавливаются аппаратными средствами после приема 8-го бита в режиме 0 и стоп-бита в остальных режимах. Сбрасывается программными средствами.

Флаг прерывания передатчика TI устанавливаются аппаратными средствами после передачи последнего бита кадра. Сбрасывается программными средствами.

В разряд RB8 в режимах 2 и 3 записывается принятый девятый бит, в режиме 1 - стоп-бит. В режиме 0 он не используется, и в него необходимо записать 0. При приеме устанавливается и сбрасывается аппаратными средствами. Программно доступен.

Разряд TB8 в режимах 2 и 3 содержит девятый бит передаваемых данных. Устанавливается и сбрасывается программным способом.

Бит SM2 вместе с битом REN позволяют управлять приемом данных. Флаг прерывания приемника RI = $REN \cdot \overline{(SM2 \oplus RB8)}$ устанавливается в 1, сигнализируя об успешном завершении приема в SBUF, при REN=1 и равенстве содержимого разрядов SM2 и RB8.

В режиме 0 для RB8=0, необходимо установить SM2=0; в режиме 1 RB8=1 (записан стоп-бит), следовательно, и SM2=1; в режимах 2 и 3 можно селективно принимать данные, устанавливая SM2=0 или SM2=1, что используется при работе в простейшей локальной сети.

2.4.1. Синхронный обмен (режим 0)

Режим 0 предназначен для симплексного синхронного обмена информацией 8-битным кадром со скоростью $f_{M0} = f_{OSC}/12$ бит/с. [Индекс M0 – режим 0 (Mode 0)]. Для тактовой частоты микроконтроллера f_{OSC} = 12 МГц, скорость передачи равна 1 Мбит/с. Для обеспечения работы в режиме 0 необходимо установить RB8=0 и SM2=0.

В синхронном режиме 8 бит информации в последовательном коде принимаются и передаются через двунаправленный вывод RxD. На выводе TxD передатчиком формируется сигнал синхронизации.

В командах, использующих SBUF в качестве регистра назначения, устройство управления формирует сигнал «Запись в SBUF». По этому сигналу, при условии TI=0, данные с внутренней шины данных ШД записываются в регистр SBUF и запускается блок управления передачей. Спустя один машинный цикл, содержимое регистра SBUF-«передатчика» выводится на линию RxD младшим битом вперед (рис. 20,а). После вывода восьмого бита, при условии SM2=0, аппаратно устанавливается флаг TI =1, что является признаком окончания передачи и разрешения загрузки очередного байта. Флаг TI сбрасывается программно.

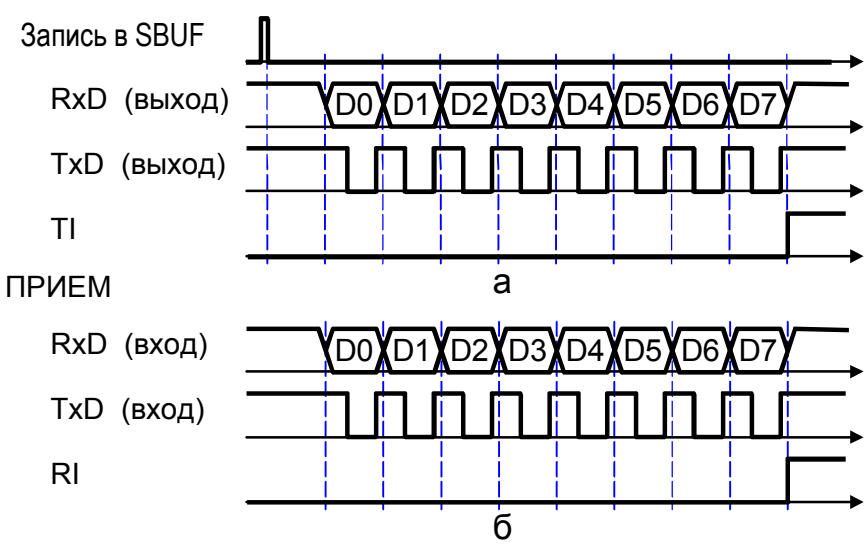


Рис. 20

Прием информации в синхронном режиме возможен при одновременном выполнении условий REN=1 и RI=0. Оба разряда устанавливаются программно. При приеме восьми бит информации в регистр SBUF-«приемник» аппаратно устанавливается RI=1, что является признаком окончания приема и разрешения чтения информации из регистра SBUF (рис. 20,б).

2.4.2. Асинхронный обмен (режимы 1,2,3)

Асинхронный обмен позволяет использовать дуплексный режим работы – одновременно производить прием и передачу информации. Для этого необходимо соединить выход TxD передатчика со входом RxD приемника.

Обмен информацией осуществляется кадром, содержащим 10 (8 бит информации, старт-бит и стоп-бит) или 11 (9 бит информации, старт-бит и стоп-бит) битовых интервалов (рис. 21). Девятый бит D8 в кадре с 11 битами выполняет служебную функцию. При приеме он записывается в разряд RB8, а при передаче читается из разряда TB8 регистра SCON.

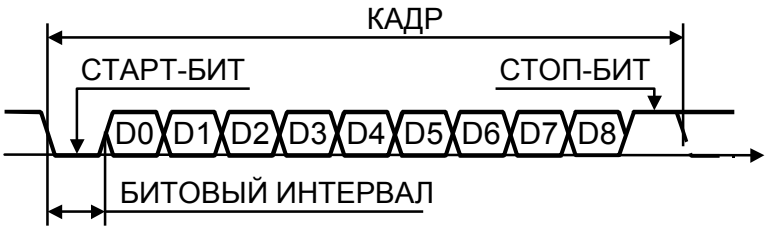


Рис. 21

Скорость обмена информацией определяется частотой внутренних тактовых сигналов синхронизации передатчика Tx и приемника Rx, формирование которых показано на рис. 22. Они формируются из тактовых сигналов частотой f_{IN}. В режиме 2 это тактовые импульсы синхронизации микроконтроллера частотой $f_{OSC}/2$, а в режимах 1 и 3 – таймера/счетчика TC1.

Тактовые сигналы Tx и Rx формируются из сигнала f_1 счетчиками/делителями частоты на шестнадцать CntT и CntR. В зависимости от состояния разряда SMOD регистра PCON частота $f_1 = f_{IN}$ (при SMOD=1) или $f_1 = f_{IN}/2$ (при SMOD=0).

Счетчиком/делителем CntR каждый битовый интервал приемника разбивается на 16 фаз. Бит-детектором в фазах 7, 8 и 9, расположенных в середине битового интервала, опрашивается входная линия RxD. На основании данных трех опросов с помощью мажоритарной функции «два из трех» выносится решение о логическом уровне сигнала на битовом интервале. Результат этого решения подается на вход сдвигающего регистра SBUF-«приемника».

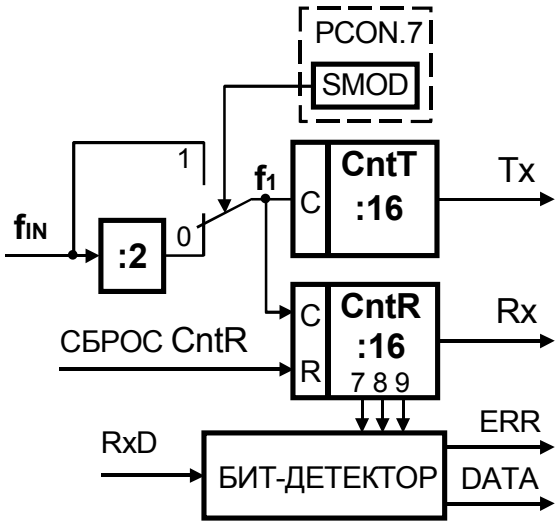


Рис. 22

В отсутствие передачи на выводе TxD установлен высокий уровень сигнала. Передача инициируется любой командой, использующей SBUF в качестве регистра назначения. Вырабатываемый при этом устройством управления ЦП сигнал «Запись в SBUF» загружает данные с внутренней шины дан-

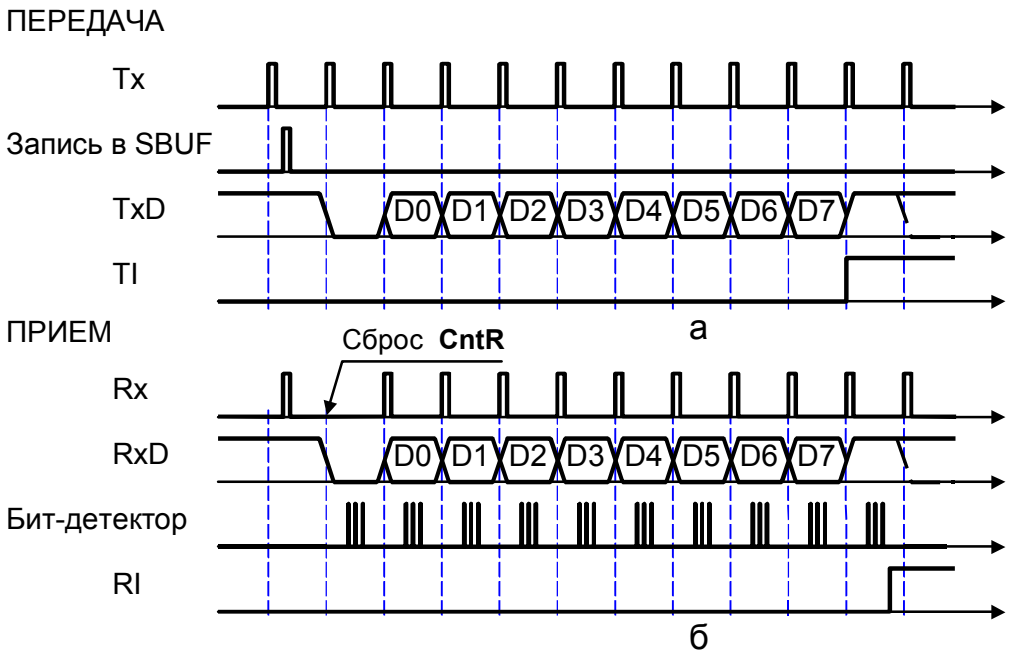


Рис. 23

ных ШД в сдвигающий регистр SBUF-«передатчик» и ближайшим тактовым импульсом Tx (рис. 23,а) начинается передача данных. Первым выдается старт-бит, потом 8 или 9 бит данных младшим разрядом вперед, а затем стоп-бит. После передачи последнего бита данных аппаратно устанавливается TI=1, что является признаком завершения передачи и разрешением загрузки следующего байта. В 11-битовом формате кадра старший бит D8 читается из разряда TB8 регистра SCON, который должен быть предварительно установлен программными средствами.

Прием начинается с перехода сигнала на линии RxD из 1 в 0 (рис. 23,б). Для отслеживания такого перехода в режиме ожидания приема линия RxD аппаратно опрашивается в каждой фазе битового интервала. При обнаружении перехода немедленно сбрасывается счетчик-делитель CntR сигналом «Сброс CntR», что обеспечивает его фазирование относительно бит-интервалов передатчика (рис. 22). Бит-детектором определяется уровень сигнала при приеме старт-бита. Если это 0, то далее принимаются данные. В противном случае приемник переходит в режим ожидания.

После окончания приема кадра в разряд RB8 регистра SCON в режиме 1 записывается стоп-бит, а в режимах 2 и 3 - девятый бит данных D8. Если содержимое разряда RB8 совпадает с установленным значением разряда SM2, то устанавливается RI=1, что является признаком окончания приема и разрешением чтения принятого байта из регистра SBUF. В противном случае признак RI не устанавливается, принятая посылка безвозвратно теряется, и приемник переходит в режим ожидания. При стоп-бите, равном 1, в режиме 1 необходимо установить SM2=1. Значение принятого стоп-бита в режимах 2 и 3 не влияет на содержимое SBUF, RB8 и RI1.

Режимы асинхронного обмена данными различаются форматом кадра и скоростью обмена (таблица 9).

Режим 1 и **3**. Тактовая частота обмена

$f_{M1} = f_{M3} = f_{OSC} \cdot 2^{(SMOD)}/f_{OV}$,

где f_{OSC} – частота тактового генератора микроконтроллера;

(SMOD) – содержимое бита SMOD регистра PCON;

f_{OV} – частота переполнения таймера/счетчика TC1.

При работе таймера/счетчика TC1 в режиме 2 (автозагрузка)

$f_{M1} = f_{M3} = f_{OSC} \cdot 2^{(SMOD)}/(32 \cdot 12 \cdot (256-(TH1)))$,

где (TH1) – содержимое байта TH1 таймера/счетчика TC1.

Режим 2. Тактовая частота обмена

$f_{M2} = f_{OSC} \cdot 2^{(SMOD)}/64$.

2.4.3. Обмен в многопроцессорных системах

Режимы 2 и 3 позволяют организовать локальную сеть для обмена данными в многопроцессорной системе. Каждому микроконтроллеру сети присваивается свой индивидуальный адрес. Выводы TxD и RxD каждого микроконтроллера соединяются и подключаются к общему моноканалу. Очевидно, что в этом случае возможен только симплексный обмен: в каждый момент времени может работать только один передатчик. Обмен данными между микроконтроллерами многопроцессорной системы в режимах 2 и 3 основан на том, что флаг прерывания приемника RI устанавливается лишь в случае, если содержимое разряда SM2 регистра SCON приемника совпадает с логическим уровнем принятого девятого бита D8.

Обмен производится кадрами, содержащими адрес приемника, и данные. В каждой адресной части кадра устанавливается одно, заранее оговоренное значение девятого бита D8, а в области данных - его инверсия. В исходном состоянии разряд SM2 у всех микроконтроллеров сети устанавливается равным биту D8 адресной части посылки.

Когда ведущий микроконтроллер передает данные ведомому, адресную часть принимают все микроконтроллеры и сравнивают принятый адрес с собственным. Лишь один из них, у кого принятый адрес совпадает с собственным, инвертирует содержимое разряда SM2, обеспечивая себе прием данных. Остальные микроконтроллеры разряд SM2 не инвертируют и данные принять не могут. После окончания приема значение разряда SM2 восстанавливается.

2.5. Таймеры/счетчики.

Базовая конфигурация микроконтроллеров содержит два 16-разрядных суммирующих таймера/счетчика TC0 и TC1, предназначенных для подсчета внешних событий, получения программно-управляемых временных задержек и выполнения внутренних времязадающих функций. Таймеры/счетчики построены на программно-доступных регистровых парах (TH0, TL0) и (TH1, TL1), размещенных в области регистров специальных функций SFR. В управлении режи-

мом работы TC0 и TC1 участвует старшая тетрада регистра TCON (**T**imer/Counter **Con**trol) и регистр TMOD (**T**imer/Counter **Mod**e). Младшая тетрада регистра TCON используется в работе системы прерываний. Оба регистра управления расположены в области SFR и программно-доступны, а регистр TCON, находящийся еще и в области BSEG, имеет программно-доступные биты.

Старшая тетрада регистра TCON, назначение разрядов которой приведено в таблице 10, содержит биты разрешения счета TRx (**T**imer

Таблица 10**.** Регистр TCON (88H)

7	6	5	4	3	2	1	0
TF1	**TR1**	**TF0**	**TR0**	**IE1**	**IT1**	**IE0**	**IT0**

Назначение разрядов регистра:

TCON.0	IT0	Бит выбора типа активного сигнала по входу $\overline{INT0}$ ($\overline{INT1}$).
TCON.2	IT1	При ITx=1 активным является переход из 1 в 0, при ITx=0 - низкий уровень входного сигнала. Устанавливается и сбрасывается программно.
TCON.1	IE0	Флаг запроса внешнего прерывания по входу $\overline{INT0}$ ($\overline{INT1}$).
TCON.3	IE1	При подтверждении прерывания и ITx=1 сбрасывается аппаратно.
TCON.4	TR0	Бит запуска TC0 (TC1). При TRx=1 счет разрешен. Уста-
TCON.6	TR1	навливается и сбрасывается программно.
TCON.5	TF0	Флаг переполнения TC0 (TC1) .При подтверждении пре-
TCON.7	TF1	рывания сбрасывается аппаратно.

Run) (x=0,1) и флаги переполнения TFx (**T**imer overflow **F**lag) таймеров/счетчиков TC0 и TC1.

Регистр TMOD разбит на две тетрады, младшая из которых управляет таймером/счетчиком TC0, а старшая – TC1. Обозначение разрядов регистра TMOD приведено таблице 11.

Каждый таймер/счетчик содержит счетный регистр (THx, TLx) с подключенным к нему флагом переполнения TFx и логику управления.

Счетные регистры работают только в режиме суммирования и при переходе из состояния «все единицы» в состояние «все нули» устанавливают флаг переполнения TFx=1. Если прерывание от соответствующего таймера/счетчика TCx разрешено, то установка флага переполнения вызывает прерывание. Флаг переполнения TFx сбрасывается аппаратно при передаче управления прерывающей программе. Флаги TFx программно доступны и могут быть установлены или сброшены программой.

Логика управления устанавливает функцию таймера или счетчика, а также разрешает или останавливает счет программным (установкой бита TRx) или аппаратным (сигналом с вывода Tx микроконтроллера) способом.

Функция таймера заключается в счете импульсов, следующих с фиксированной частотой $f_{OSC}/12$ (при f_{OSC} = 12 МГц период импульсов

Таблица 11. Регистр TMOD (89H)

7	6	5	4	3	2	1	0
GATE1	**C/$\overline{T1}$**	**M1.1**	**M0.1**	**GATE0**	**C/$\overline{T0}$**	**M1.0**	**M0.0**

Назначение разрядов регистра:

TMOD.0	M0.0	Младший бит управления режимом CT0.
TMOD.1	M1.0	Старший бит управления режимом CT0.
TMOD.2	C/$\overline{T0}$	Выбор функции таймера (C/$\overline{T0}$=0) или счетчика (C/$\overline{T0}$=1) для CT0.
TMOD.3	GATE0	Флажок разрешения (GATE=1) управления работой CT0 с внешнего вывода $\overline{INT0}$ при TR0=1. При GATE=0 работа CT0 зависит от состояния бита TR0 регистра TCON.
TMOD.4	M0.1	То же, но для CT1.
TMOD.5	M1.1	То же, но для CT1.
TMOD.6	C/$\overline{T1}$	То же, но для CT1.
TMOD.7	GATE1	То же, но для CT1.

равен 1 мкс), а счетчика – в подсчете числа переходов из 1 в 0 на входах T0 и T1 микроконтроллера.

Таблица 12

Режим	M1.x	M0.x
0	0	0
1	0	1
2	1	0
3	1	1

Оба таймера/счетчика могут работать в трех режимах (0, 1 и 2). Режим каждого таймера/счетчика устанавливается битами M1.x и M0.x. В режиме 3 работает только таймер/счетчик TC0, используя часть бит управления TC1. В таблице 12 приведена комбинация бит M1.x и M0.x для каждого режима работы таймера/счетчика.

Режим 0 (рис. 24). В этом режиме счетный регистр имеет длину 13 разрядов (8 разрядов регистра THx и 5 младших разрядов регистра TLx). Три старших разряда регистра TLx игнорируются. Сигнал переполнения счетчика фиксируется флажком TFx.

Логика управления таймера/счетчика содержит: переключа-

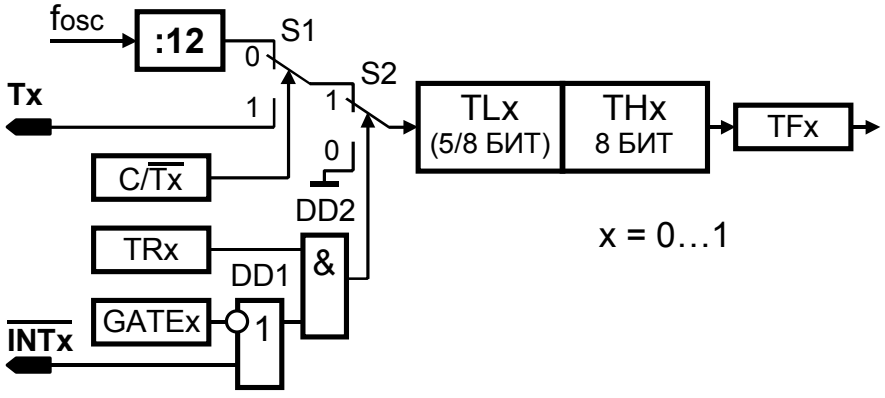


Рис. 24

тель S1 (мультиплексор, управляемый битом C/$\overline{Tx}$), устанавливающий режим таймера (C/$\overline{Tx}$=0) или счетчика (C/$\overline{Tx}$=1); переключатель S2 (мультиплексор, управляемый логикой разрешения счета). Логика разрешения счета содержит ЛЭ DD1 и DD2. Она позволяет выполнить программное (GATEx=0) или аппаратное (GATEx=1) управление счетом.

Программное управление выполняется битом разрешения счета TRx при установке GATEx=0, блокирующим вход $\overline{INTx}$. Счет разрешается при TRx=1.

Аппаратное управление разрешением счета производится внешним управляющим сигналом, подаваемым на вход $\overline{INTx}$ при установке GATEx=1 и TRx=1. Счет разрешается при $\overline{INTx}$ =1.

Режим 0 введен для поддержки программного обеспечения микропроцессоров предыдущего семейства MCS-48.

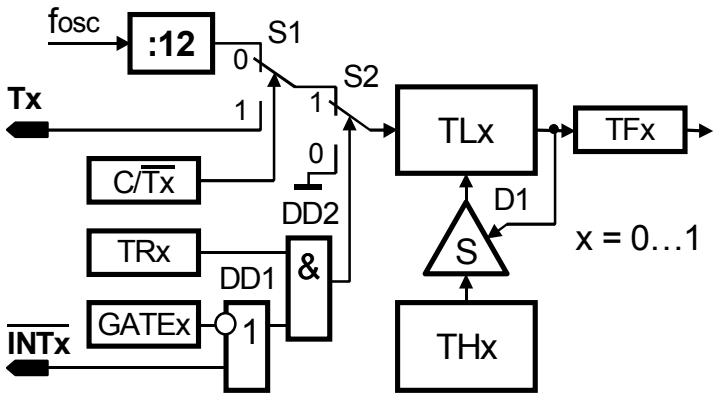


Рис. 25

Режим 1 (рис. 24) отличается от режима 0 использованием 16 разрядного счетного регистра. Логика управления в режимах 1 и 0 одинакова.

Режим 2 отличается от режима 1 использованием автозагрузки счетного регистра (рис. 25). В этом режиме регистр TLx работает как 8-разрядный счетный регистр, а THx – как регистр хранения байта. Сигнал переполнения счетного регистра TLx не только устанавливает бит переполнения TFx, но и перезагружает регистр TLx (открывает ключи D1) содержимым регистра THx, повторяя счет. Содержимое регистра THx при этом не изменяется. При выполнении функции таймера прерывания следуют с периодом, определяемым содержимым регистра TH0. Логика управления в режимах 2 и 0 одинакова.

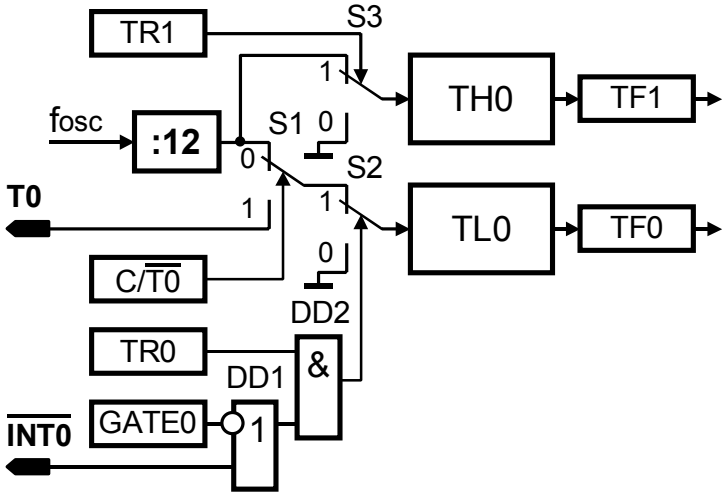


Рис. 26

Режим 3 предназначен для работы только таймера/счетчика TC0. В этом режиме он представляет собой два 8-разрядных счетных регистра TL0 и TH0, работающих независимо друг от друга (рис. 26). Логика управления работой счетного регистра TL0 аналогична логике управления в

режиме 0. При переполнении TL0 устанавливается флажок TF0.

Счетный регистр TH0 работает только в режиме таймера. Для управления счетным регистром TH0 используются биты TR1 и TF1 таймера/счетчика TC1. Сигнал переполнения фиксируется флажком TF1. Программное разрешение счета выполняется переключателем S3 (мультиплексор, управляемый битом TR1). Счет разрешается при TR1=1. Аппаратное управление счетом отсутствует. Неиспользуемые регистры TL1 и TH1 таймера/счетчика TC1 можно использовать как регистры общего назначения.

В режимах 0, 1 и 3 таймер/счетчик, после переполнения, продолжает работать, что может вызвать прерывания через максимальный интервал времени, определяемый используемым режимом. Поэтому в начале прерывающей программы должен содержаться фрагмент перезагрузки таймера/счетчика, если он дальше нужен, или запрещения его работы. Для перезагрузки необходимо приостановить работу таймера/счетчика, установив TRx=0, загрузить регистры TLx и THx и разрешить работу счетчика (TRx=1):

```
CLR     TRx          ;Остановить работу таймера/счетчика
MOV     TLx, DataTLx ;Загрузить
MOV     THx, DataTHx ;       счетный регистр,
SETB    TRx          ;Начать счет
…….                  ;Прерывающая программа
RETI                 ;Выход из прерывающей программы
```

Для запрещения его работы необходимо установить TRx=0:

```
CLR     TRx          ;Остановить работу таймера/счетчика
…….                  ;Прерывающая программа
RETI                 ;Выход из прерывающей программы
```

2.6. Система прерываний

Система прерываний предназначена для обеспечения реакции микроконтроллера на внешние и внутренние запросы прерывания. Базовые микроконтроллеры семейства MCS-51 имеют 5 прерываний: два внешних (входы $\overline{INT0}$ и $\overline{INT1}$) и три внутренних (от таймеров/счетчиков TC1, TC2 и последовательного порта)

Система прерываний позволяет:

- фиксировать запросы от внешних и внутренних источников прерываний до их исполнения;
- программным способом разрешить или запретить работу системы прерываний в целом;
- программным способом маскировать запросы прерываний отдельных источников;
- программным способом устанавливать высокий или низкий приоритет каждому источнику запроса прерываний;
- формировать вектор прерывания;
- аппаратным способом снять запрос на прерывание при его исполнении.

Работу системы прерываний обслуживают регистры IE,IP, а также часть разрядов регистров TCON и SCON.

Логическая схема системы прерываний приведена на рис. 27.

Фиксация запросов двух внешних прерываний выполняется разрядами IE0 и IE1, прерываний таймеров/счетчиков - разрядами TF0 и TF1 регистра TCON, а последовательного порта – разрядами TI и RI регистра SCON. При наличии запроса прерываний от конкретного источника соответствующий ему разряд устанавливается в 1.

Запрос на прерывание от внешних источников с входов $\overline{INT0}$ и $\overline{INT1}$ может реализовываться по уровню или фронту сигнала. Для этой цели на входе установлены мультиплексоры-переключатели S1 и S2, управляемые разрядами IT0 и IT1 регистра TCON.

При ITx=0 (x=0,1) запрос на прерывание от внешнего источника осуществляется по уровню, а при ITx=1 – по фронту.

При прерывании по уровню запрос на прерывание устанавливается низким уровнем сигнала, подаваемого на вход $\overline{INTx}$ (x=0,1). Входной сигнал инвертируется логическим элементом DD1 (или DD2) и устанавливает в 1 разряд IEx (x=0,1). Разряд IEx повторяет, не фиксируя, состояние внешнего входа $\overline{INTx}$, поэтому для надежного обнаружения сигнала запроса прерывания его длительность должна быть не менее длительности машинного цикла. Разряд IEx при обработке прерывания аппаратным способом не сбрасывается. Чтобы исключить повторную обработку одного и того же запроса, источник запроса прерывания должен установить высокий уровень сигнала на входе $\overline{INTx}$ до окончания обработки прерывания либо прерывающая программа должна заканчиваться командой RET (вместо RETI). Команда RET, в отличие от команды RETI, не сбрасывает зафиксированный системой прерывания

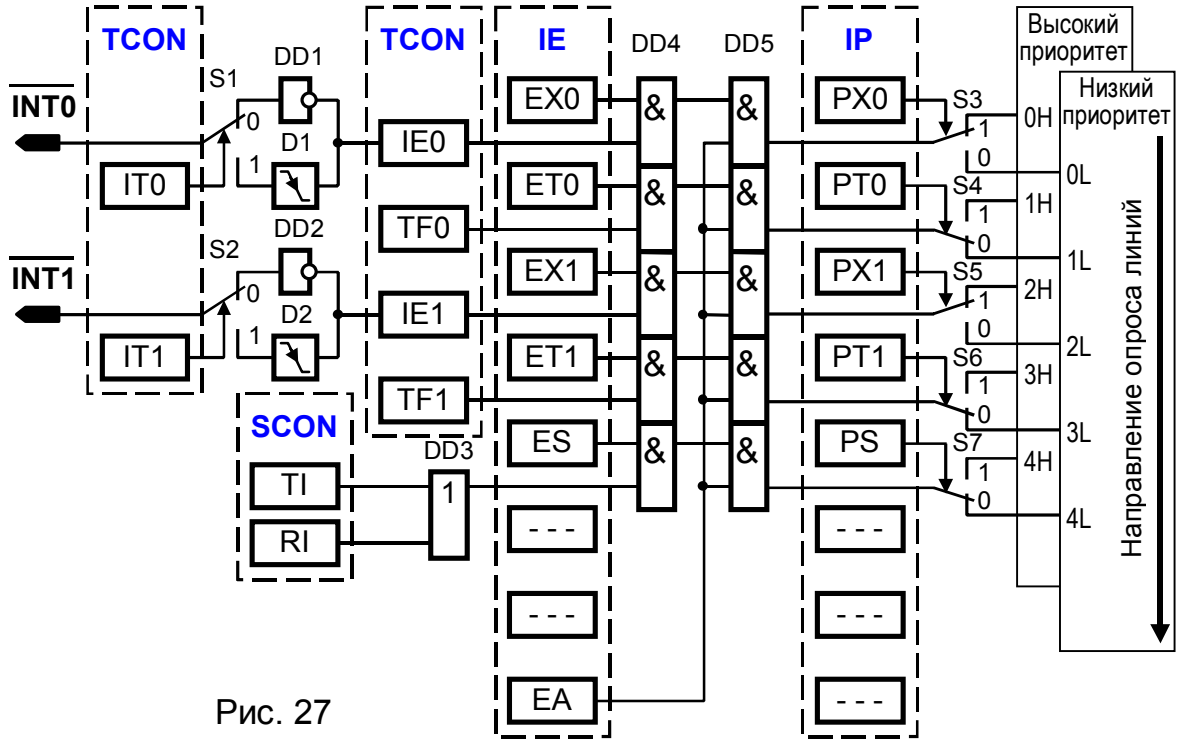


Рис. 27

приоритет запроса обслуживаемого прерывания. Осуществить прерывания далее могут только источники с более высоким приоритетом.

При фиксации запроса прерывания по фронту формирователями D1 (или D2) вырабатывается импульс при переходе сигнала на входе $\overline{INTx}$ из 1 в 0. Этим импульсом устанавливает в 1 разряд IEx (x=0,1). В этом случае разряд IEx сбрасывается аппаратным способом при обработке прерывания.

Разряды TF0 и TF1 устанавливаются таймерами/счетчиками ТС0 и ТС1 соответственно при переходе счетного регистра из состояния «все единицы» в состояние «все нули». Сброс разрядов TF0 и TF1 выполняется аппаратным способом при переходе к подпрограмме обслуживания прерываний.

Разряд TI устанавливается передатчиком последовательного порта после передачи последнего бита кадра - освобождения регистра SBUF, а разряд RI – приемником после приема последнего бита кадра - заполнения регистра SBUF. Разряды RI и TI сбрасываются программным способом обычно в пределах вызванной подпрограммы обработки прерывания.

Все разряды установки запроса прерываний (IE0, IE1, TF0, TF1, RI, TI) могут быть установлены или сброшены программным способом, т.е. прерывание может быть вызвано, а ожидающее обслуживания прерывание сброшено программным способом, а не сигналом запроса прерывания. Кроме того, внешние прерывания по входам $\overline{INTx}$ можно вызвать программной установкой разрядов порта P3.2=0 и P3.3 =0.

Регистр IE (**I**nterrupt **E**nable) предназначен для управления разрешением прерываний. Разряд EA=0 запрещает все прерывания, блокируя передачу запросов прерывания через ЛЭ DD5. При EA=1 прерывания разрешены, однако прерывание от каждого источника или группы источников может быть маскировано (запрещено) установкой 0 в соответствующих разрядах регистра IE, блокирующих передачу запроса прерывания через ЛЭ DD4. Все разряды регистра IE устанавливаются и сбрасываются программным способом. Обозначение разрядов регистра показано в таблице 13.

Таблица 13. Регистр IE (A8H)

7	6	5	4	3	2	1	0
EA	**-**	**-**	**ES**	**ET1**	**EX1**	**ET0**	**EX0**

Назначение разрядов регистра:

IE.0	EX0	Прерывание от внешнего источника $\overline{INT0}$.
IE.1	ET0	Разрешение прерывания от таймера/счетчика ТС0.
IE.2	EX1	Прерывание от внешнего источника $\overline{INT1}$.
IE.3	ET1	Разрешение прерывания от таймера/счетчика ТС1.
IE.4	ES	Разрешение прерывания от последовательного порта.
IE.5	-	Зарезервирован.
IE.6	-	Зарезервирован.
IE.7	EA	Общее управление прерываниями.

Каждому из пяти источников прерываний присваивается один из двух уровней приоритета установкой или сбросом соответствующих разрядов регистра IP (**I**nterrupt **P**riority). Все разряды регистра IP устанавливаются и сбрасываются программным способом. Обозначение разрядов регистра показано в таблице 14.

Таблица 14. Регистр IP (B8H)

7	6	5	4	3	2	1	0
-	-	-	**PS**	**PT1**	**PX1**	**PT0**	**PX0**

Назначение разрядов регистра:

IP.0 PX0 Приоритет внешнего источника $\overline{INT0}$.
IP.1 PT0 Приоритет таймера/счетчика TC0.
IP.2 PX1 Приоритет внешнего источника $\overline{INT1}$.
IP.3 PT1 Приоритет таймера/счетчика TC1.
IP.4 PS Приоритет последовательного канала.
IP.5 - Зарезервирован.
IP.6 - Зарезервирован.
IP.7 - Зарезервирован.

Установка разряда в 1 соответствует высокому уровню приоритета, в 0 – низкому. Каждый разряд регистра IP управляет одним из переключателей – демультиплексоров S3…S7, соединяющих выходы запроса прерываний (выходы ЛЭ DD5) с линиями высокого (0H…4H) и низкого (0L…4L) уровней приоритетов. В конце машинного цикла производится опрос состояния линий высокого приоритета, а затем линий низкого приоритета. Опрос ведется в порядке возрастания номеров линий, что определяет распределение приоритетов вторичного арбитража, приведенного в таблице 15. На каждом уровне наиболее высоким приоритетом обладают нулевые линии, опрашиваемые первыми, а наиболее низким – четвертые, опрашиваемые последними. Наличие вторичного арбитража устраняет конфликтные ситуации, возникающие при одновременном появлении запросов прерывания от нескольких источников одинакового уровня.

Таблица 15

Источник	Приоритет
IE0	0 (высший)
TF0	1
IE1	2
TF1	3
RI $\vee$ TI	4 (низший)

Например, если внешние прерывания имеют высший приоритет, как показано на рис. 27 (PX0=1 и PX1=1) и они не маскированы (EA=1, EX0=1 и EX1=1), то при одновременном появлении запросов прерывания на входах $\overline{INT0}$ и $\overline{INT1}$ первым, в результате вторичного арбитража, будет обслужен запрос по входу $\overline{INT0}$.

Процедура обслуживания низкоуровневого прерывания может быть прервана запросом более высокого уровня. Обработка высокоуровневого прерывания не может быть остановлена.

Опрос состояния системы прерываний производится в конце каждого машинного цикла в фазе S5P2, за исключением команд RETI и любых команд с обращением к регистрам IE и IP. От момента фиксации запроса на прерывания до обслуживания прерывания требуется от 38 до 86 периодов частоты f_{OSC}, в зависимости от фазы поступления запроса и числа машинных циклов команды, во время выполнения которой поступил запрос.

При реализации прерывания аппаратным способом выполняется команда LCALL addr16, обеспечивающая запоминание в стеке текущего состояния программного счетчика (запоминание адреса возврата), и переход к стартовому адресу addr16 соответствующей процедуры обслуживания. С каждым источником запроса прерываний связан свой стартовый адрес (вектор прерывания):

EXTI0	IE0	0003h	Внешнее прерывание $\overline{INT0}$.
TIMER0	TF0	000Bh	Прерывание таймера/счетчика TC0.
EXTI1	IE1	0013h	Внешнее прерывание $\overline{INT1}$.
TIMER0	TF1	001Bh	Прерывание таймера/счетчика TC1.
SINT	RI $\vee$ TI	0023h	Прерывание последовательного порта.

2.7. Методы адресации и система команд семейства MCS-51

Система команд семейства MCS-51 ориентирована на организацию гибкого ввода-вывода данных через универсальные порты P0...P3 и первичную обработку информации. Особое внимание уделено операциям с битами и передаче управления по их значению. Команды, выполняющие такие операции, составляют многочисленную группу и образуют вместе с соответствующими аппаратными средствами так называемый «булев процессор» в составе архитектуры MCS-51.

Система команд предоставляет программисту возможность использовать большинство операций с полным набором методов адресации и программно-доступных ресурсов аппаратуры.

2.7.1. Методы адресации

Каждая команда сообщает процессору выполняемую операцию и методы доступа к операндам. Код команды имеет несколько полей, имеющих определенное функциональное назначение. Важнейшими полями любой команды являются код операции (КОП), определяющий действие команды, и адресная часть. Поля адресной части содержат информацию об адресах операндов и результата операции, а в некоторых случаях информацию об адресе следующей команды.

Если адрес указывает на номер ячейки памяти, в которой находится или куда заносится операнд , то его называют прямым адресом.

Методы адресации представляют собой набор механизмов доступа к операндам. Одни из них просты, приводят к компактному формату команды и быстрому доступу к операнду, но имеют ограниченный объем доступных ресурсов. Другие позволяют оперировать всеми имеющимися в системе ресурсами, но команда получается длинной, на ее

ввод и выполнение тратится много времени. Набор методов адресации в каждой системе команд является компромиссным сочетанием известных механизмов адресации, выбранных проектировщиками архитектуры исходя из набора решаемых задач.

Ниже приведены основные методы адресации, используемые в системе команд семейства MCS-51.

Неявная адресация. В команде не содержится явных указаний об адресе участвующего в операции операнда или адресе, по которому помещается результат операции, но этот адрес подразумевается. В командах наиболее часто неявно адресуется аккумулятор как приемник результата операции. Например, результат сложения содержимого аккумулятора (A) и регистра R1 текущего банка данных командой ADD A,R1 записывается в неявно адресуемый аккумулятор. Вся указанная команда занимает в памяти один байт, в то время как адрес только аккумулятора (8Eh области SFR) содержит один байт.

Непосредственная адресация. В поле адреса команды содержится не адрес операнда, а непосредственно сам операнд. На непосредственную адресацию указывает специальный символ # перед числом. Например, командой MOV A,#15h шестнадцатеричное число 15 (второй байт команды) загружается в аккумулятор. В системе команд непосредственная адресация обозначена как #data, где data – число (data = 00h...FFh).

Прямая адресация. В поле адреса команды указан прямой адрес ячейки памяти данных, в которой находится или куда заносится операнд. Например, командой MOV A,15h содержимое ячейки DSEG с адресом 15h загружается в аккумулятор. Ячейка памяти имеет прямую адресацию, а аккумулятор - неявную. В зависимости от местонахождения адресуемого операнда, прямая адресация подразделяется на прямую регистровую и абсолютную.

Прямая регистровая адресация. В поле адреса команды указан прямой адрес регистра текущего регистрового банка. Регистров в каждом банке восемь, и для их адресации необходим трехбитовый прямой адрес. В мнемонике команд адресуемый регистр обозначен Rn, где n=0...7. Все поля команды умещаются в один байт. Такую адресацию называют короткой. Например, MOV R4,R1.

Прямая абсолютная адресация позволяет обратиться к любой ячейке DSEG и области SFR. Прямой адрес в этом случае занимает один байт, а команда – два байта. В системе команд байт прямого адреса обозначен словом direct (прямой) (direct = 00h…FFh). Например, команда MOV 80h,R2 (или MOV P0,R2) загружает содержимое регистра R2 текущего банка данных в порт P0 (ячейка 80h области SFR). Если оба операнда имеют прямую абсолютную адресацию, то команда становится трехбайтовой (Например, MOV 80h,15h).

Косвенная адресация. В поле адреса указан адрес ячейки памяти, в которой находится прямой адрес операнда. В системе команд на косвенную адресацию указывает специальный символ @. Свойством

хранить прямой адрес обладают регистры R0 и R1 (@Ri, i = 0,1) каждого регистрового банка. Например, если содержимое регистра R1 текущего банка регистров равно 15h, то команда MOV A,@R1 выполнит то же действие, что и приведенная выше команда MOV A,15h – загрузит содержимое ячейки памяти DSEG с адресом 15h в аккумулятор. Однако команда MOV A,@R1 однобайтовая, но самое главное, здесь имеется возможность программным способом изменять адрес, изменяя содержимое регистра R1.

Относительная адресация. При относительной адресации прямой адрес формируется путем сложения базового адреса с адресным полем команды. В качестве базового адреса используется содержимое программного счетчика, а адресное поле команды представляет собой восьмиразрядное смещение rel (relative - относительный). Число rel интерпретируется командой как целое со знаком, представленное в дополнительном коде. Диапазон его представления - (-128...+127). При определении числа rel следует учесть, что программный счетчик указывает на следующую, подлежащую выполнению, команду. Относительная адресация широко используется в командах передачи управления, что позволяет создавать перемещаемые программные модули. Команды передачи управления с относительной адресацией позволяют организовать ветвление относительно текущего положения программного счетчика PC в обе стороны на (-128...+127) байт.

В программах на языке ассемблера в поле смещения можно указать метку, на которую необходимо перейти. В результате трансляции ассемблер вычислит величину смещения, если она не превышает (-128…+127). В противном случае будет выдано сообщение об ошибке.

Базовая адресация представляет разновидность относительной адресации. Прямой адрес в этом случае формируется путем сложения адреса, указанного в команде, с содержимым базового регистра, в котором хранится базовый адрес. Функцию базового регистра в семействе MCS-51 выполняет регистр-указатель данных DPTR или программный счетчик PC. Этот тип адресации особенно удобен при обработке таблиц и массивов данных. В командах MOVC A,@A+DPTR и MOVC A,@A+PC 16-разрядный прямой адрес формируется как сумма содержимого регистров DPTR и A или PC и A.

Страничная адресация. При использовании страничной адресации память разбивается на ряд страниц одинаковой длины. Адресация страниц осуществляется отдельным регистром страниц, а адресация ячеек памяти внутри страницы – адресом, содержащимся в команде. Прямой адрес формируется конкатенацией (присоединением) адреса страниц и адреса ячейки памяти внутри страницы. В команде MOVX A,@Ri функцию регистра страниц выполняет порт P2 (старший байт адреса), а содержимое регистра Ri (младший байт адреса) задает адрес внутри страницы. При этом память разбивается на 256 страниц по 256 ячеек в каждой из них.

Стековая адресация используется в безадресных командах и представляет собой сочетание автоинкрементного и автодекрементного способов адресации, работающее по принципу LIFO (**L**ast **I**nput – **F**irst **O**utput)- «последним вошел – первым вышел». Стек располагается в DSEG и растет в сторону увеличения адреса. Адрес вершины стека содержится в указателе стека SP. При записи байта в стек сначала выполняется инкремент содержимого SP, а затем по этому адресу производится запись. При чтении байта из стека сначала выполняется чтение по адресу, на который указывает SP, а затем - декремент SP. При использовании стека необходимо учитывать, что глубина стека (максимальное число ячеек памяти, занятых под стек) аппаратными средствами не контролируется. При чрезмерном увеличении стека могут быть заняты не предназначенные для него ячейки памяти с потерей информации в них. Аппаратно стек используется для сохранения адреса возврата при обслуживании прерывания.

2.7.2. Система команд семейства MCS-51

Система команд представлена в таблицах П2.1...П2.6 приложения 2. В таблицах указаны наименование команды, ее мнемоника, двоичный код операции, влияние выполняемой команды на флаги C, OV, AC и P, длина команды в байтах (Б) и время выполнения в машинных циклах (Ц), а также содержание преобразования, выполняемого командой. В качестве разделителя адресных полей в командах используется запятая. Для улучшения читаемости можно добавить пробелы после запятой, если их поддерживает используемый ассемблер.

Все множество команд можно разбить на 5 групп: операции передачи данных, арифметические операции, логические операции, операции с битами и операции передачи управления.

Группа команд операций передачи данных (таблица П2.1) содержит команды MOV (передачи данных между DSEG и RSEG), MOVC (между CSEG и A), MOVX (между XSEG и A), команды обращения к стеку PUSH и POP, а также две команды обмена XCH и XCHD. Все команды передачи данных, у которых приемником является аккумулятор, устанавливают флаг паритета P содержимого аккумулятора, а команды с прямой адресацией, у которых приемником является регистр PSW, изменяют все флаги. Наиболее емкой является команда MOV, использующая четыре способа адресации: прямой регистровый (A, Rn, DPTR), прямой (direct), косвенный (@Ri), непосредственный (#data, #data16). Второй операнд команды является источником, первый – приемником. Для указания приемника служат три способа адресации (кроме непосредственного), а для указания источника все четыре. Трехбайтовая команда MOV direct,direct обеспечивает пересылку между двумя любыми ячейками памяти (DSEG и SFR), включая RSEG. Для обмена с RSEG предусмотрены специальные двух- и однобайтовые форматы:

MOV Rn,direct	MOV A,Rn
MOV direct,Rn	MOV Rn,A

Специальная команда MOV DPTR,#data16 позволяет загрузить 16-разрядный указатель DPTR значением data16.

Команда MOVC позволяет считывать информацию из программной памяти CSEG не в регистр команд устройства управления, а в аккумулятор операционного устройства. В команде используются два способа адресации: по базе DPTR и относительно PC. В обоих случаях целое без знака смещение (индекс) хранится в аккумуляторе. Приемником результата также служит аккумулятор. Команда позволяет выполнять быструю перекодировку по таблицам.

Обращение к внешней памяти осуществляется с помощью команды MOVX. Обмен производится по байтам между аккумулятором и внешним XSEG. Ячейка XSEG может быть адресована двумя способами: косвенно через 16-разрядный указатель DPTR и странично косвенно через 8-разрядный указатель Ri, i=0,1. В последнем случае регистром страниц служит регистр P2.

Безадресные команды PUSH и POP обеспечивают передачу данных между DSEG, RSEG и SFR.

Команда обмена XCH обеспечивает двухсторонний обмен байтами, а команда XCHD - младшими тетрадами байтовых операндов.

Группа команд арифметических операций (таблица П2.2) содержит команды сложения ADD, сложения с учетом переноса ADDC, вычитания с учетом заема SUBB, увеличения и уменьшения на единицу INC и DEC, десятичной коррекции сложения в двоично-десятичном (BCD) коде упакованного формата, умножения MUL и деления DIV. Операции выполняются над беззнаковыми целыми числами. В операциях сложения и вычитания первым операндом и приемником результата служит аккумулятор. Для определения второго операнда используется прямая регистровая, прямая абсолютная, непосредственная и косвенная адресации. Операции INC и DEC применимы к аккумулятору, прямо адресуемому регистру, прямо или косвенно адресуемой ячейке памяти. Кроме того, операция INC применима к содержимому 16-разрядного регистра указателя DPTR.

В операциях целочисленного умножения и деления без знака участвуют аккумулятор и регистр B. При умножении 8-разрядное значение A умножается на 8-разрядное значение B, а 16-разрядный результат записывается в пару BA. При этом регистр B хранит старшую часть произведения. Флажок OV устанавливается, если произведение больше 255. При делении 8-разрядного значения A на 8-разрядное значение B частное записывается в A, а остаток в B. При попытке деления на 0 устанавливается флаг переполнения OV.

Команда десятичной коррекции аккумулятора DA размещается после команды сложения. Слагаемые необходимо представить в BCD коде. Коррекция выполняется стандартным способом.

Группа команд логических операций (таблица П2.3) содержит три типовые операции: ANL – логическое И, ORL – логическое ИЛИ, XRL – логическое исключающее ИЛИ. Источником первого операнда

служит либо аккумулятор A, либо прямо адресуемая ячейка памяти. Второй операнд задается одним из четырех основных методов адресации. В состав группы входят также одноместные операции над содержимым аккумулятора: CLR - очистки, CPL – инверсии, а также RL, RLC, RR и RRC – операции циклического и расширенного сдвигов вправо и влево. Сюда же включена операция обмена тетрад в аккумуляторе SWAP,которая может интерпретироваться как циклический сдвиг байта на четыре разряда.

Группа команд операций с битами (таблица П2.6) содержит команды SETB – установки бита в 1, CLR – сброса бита в 0, CPL – инверсии бита, ANL и ORL – логическое И и логическое ИЛИ содержимого флага C и прямо адресуемого бита, MOV – пересылка бита.

В битовых операциях флаг C исполняет роль булевого аккумулятора. В качестве операндов используется содержимое флага C или прямо адресуемого бита bit области BSEG. В операциях ANL и ORL можно использовать содержимое прямо адресуемого бита (bit) или инверсию содержимого (/bit).

В эту группу входят также команды условного перехода с относительным 8-разрядным смещением rel. Условный переход может быть осуществлен как при установленном (команда JB), так и при сброшенном (команда JNB) бите. Особо следует отметить команду JBC, которая при установленном бите реализует ветвление и одновременно с этим сбрасывает бит в 0.

Группа команд передачи управления (таблицы П2.4 и П2.5) содержит команды безусловного перехода AJMP, LJMP, SJMP, JMP, условного перехода JZ, JNZ, CJNE, вызова ACALL, LCALL, возврата RET, RETI и модификации с условным переходом DJNZ. Сюда же включена пустая команда NOP.

В командах передачи управления широко применяется относительная адресация, позволяющая создавать перемещаемые программные модули. В качестве относительного адреса выступает 8-разрядное смещение rel – байт со знаком, обеспечивающее переход на (–128… +127) байт относительно текущего положения PC. Для перехода в любую другую точку 64 Кбайтового адресного пространства может быть использован либо прямой addr16, либо косвенный @A+DPTR адрес. В последнем случае содержимое A интерпретируется как целое без знака. Вариант короткой прямой адресации addr11 внутри 2 Кбайтовой текущей страницы введен для совместимости с семейством MCS-48.

Все эти типы адресации применяются в командах перехода. В командах вызова используются только прямой addr16 и внутристраничный addr11 способы адресации. Во всех условных командах используется только относительная адресация.

Когда микроконтроллер опознает запрос на прерывание, он генерирует команду LCALL addr16, что автоматически обеспечивает запоминание адреса возврата в стеке. Информация о состоянии программы (содержимое регистра PSW) автоматически не сохраняется. При этом

логика прерываний запоминает уровень приоритета обслуживаемого прерывания. При выполнении команды RET уровень приоритета сохраняется и следующим может быть обслужено только прерывание с более высоким уровнем приоритета. Команда RETI отличается от команды RET тем, что она сбрасывает уровень приоритета, что позволяет обслуживать запросы на прерывания с низким уровнем приоритета.

К типовым условным операциям относятся команды JZ и JNZ, JC и JNC. Две последних включены в группу «булевых». В команде CJNE сначала сравниваются, по правилам вычитания целых чисел, два байта и в соответствии с результатом сравнения устанавливается флаг C. Затем, в случае их несовпадения, выполняется ветвление.

В команде DJNZ в качестве счетчика может использоваться не только один из регистров текущего регистрового банка Rn, n=0...7, но и прямо адресуемая ячейка памяти данных DSEG. При исполнении команды сначала выполняется декремент счетчика и, если содержимое счетчика не равно нулю, ветвление.

3. Проектирование микропроцессорных систем

Технология проектирования МПС на основе микроконтроллеров полностью соответствует концепции неразрывности процесса проектирования и отладки аппаратной и программной составляющих, принятой в микропроцессорной технике. Важной особенностью применения микроконтроллеров является работа в реальном масштабе времени, т.е. гарантированная реакция на внешние события в течение определенного интервала времени. Очевидно, что решение задачи совместной отладки аппаратной и программной составляющих в реальном масштабе времени при произвольной структуре и схемотехнике микропроцессорной системы является весьма сложной, дорогостоящей и долговременной работой.

3.1. Этапы проектирования

Особенностью МПС является то, что сами они встраиваются (интегрируются) в некоторый объект. Это предполагает, что перед разработчиком МПС такого рода стоят задачи полного цикла проектирования, начиная от разработки алгоритма функционирования и заканчивая комплексными испытаниями в составе изделия, а возможно, и сопровождением при производстве. Основные этапы проектирования МПС отображены на рис. 28 [1,4].

Технические требования начинают цикл проектирования МПС. Возможность программирования микропроцессорной системы стимулирует заказчика возложить на нее выполнение максимального числа функций. Критерием выбора должна служить экономическая целесообразность любого увеличения объема аппаратных средств, что определяется в результате исследования рынка приборов данного типа, и максимальное улучшение показателя цена/функциональные возможно-

сти. На этом этапе явно или неявно формулируются требования к типу используемого микропроцессора или микроконтроллера.

Этап разработки алгоритма является наиболее ответственным, поскольку ошибки этого этапа обнаруживаются при испытании законченного изделия и приводят к дорогостоящей переработке всей МПС. Прорабатывается несколько вариантов алгоритма, обеспечивающих выполнение технических требований с использованием наработанных ранее функционально-топологических модулей. Основные варианты отличаются соотношением объема программного обеспечения и аппа-

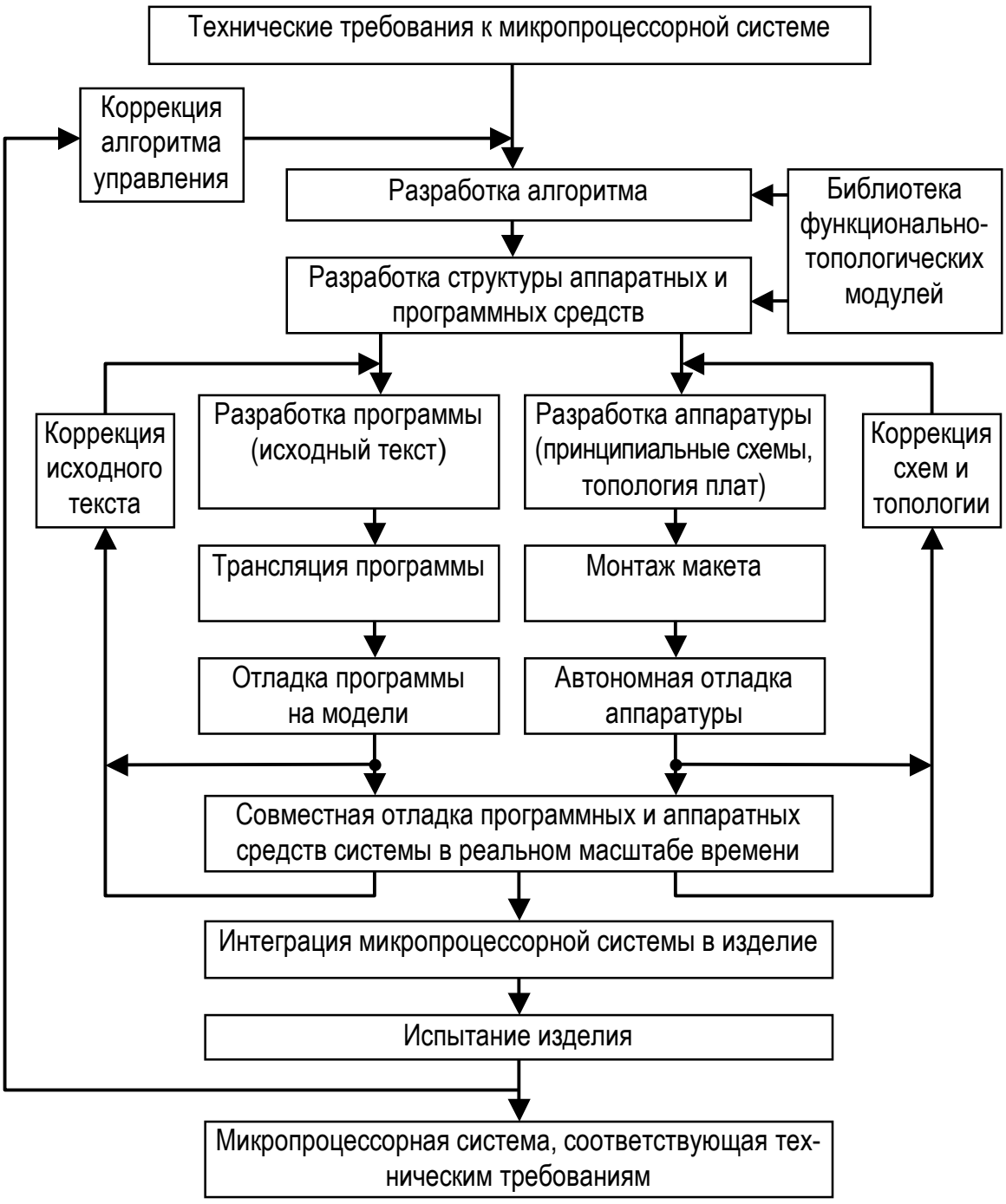


Рис. 28

ратуры. Критерием выбора является реализация максимального числа функций программным способом при наименьших аппаратных затратах, обеспечивающая заданные показатели быстродействия и надежности в полном диапазоне эксплуатационных воздействий. Часто определяющим требованием является возможность размещения кода программы во внутренней памяти микроконтроллера, что позволяет обеспечить его защиту от несанкционированного доступа. На этом этапе окончательно определяется тип микроконтроллера и важнейших схем обрамления (память, ПЛИС, интерфейс, АЦП, и т. п.).

На этапе разработки структуры МПС окончательно определяется состав имеющихся и подлежащих разработке аппаратных модулей, протоколы обмена между модулями, типы разъемов. Поскольку МПС встраивается в изделие, выполняется предварительная проработка конструкции плат. В части программного обеспечения определяются состав и связи программных модулей, язык программирования. Здесь же производится выбор средств проектирования и отладки.

Стадия создания программы образует внутренний, часто повторяющийся цикл. Она состоит из этапов разработки исходного текста, трансляции, отладки программы на модели и коррекции исходного текста. Содержание этих этапов существенно зависит от используемых системных средств. В настоящее время ресурсы микропроцессоров и микроконтроллеров достаточны для поддержки программирования на языках высокого уровня. Это позволяет использовать все преимущества структурного программирования, разрабатывать программное обеспечение как проект с использованием раздельно транслируемых модулей. В настоящее время самым мощным средством разработки программного обеспечения для микропроцессоров и микроконтроллеров являются интегрированные кросс-системы программирования на языках высокого уровня типа Паскаль, Си. Например, интегрированная система разработки программного обеспечения Паскаль-51 содержит в своем составе редактор текста, компилятор с редактором связей, библиотеку стандартных функций периода выполнения и отладчик. Такие системы позволяют значительно сократить затраты времени на выполнение этого внутреннего цикла. Однако программы, написанные на языках высокого уровня, имеют больший объем и более низкое быстродействие, чем аналогичные программы, написанные на языке ассемблера. Поэтому язык ассемблера продолжает широко использоваться, особенно при ограниченных ресурсах МПС и необходимости обеспечить выполнение контролируемых интервалов времени.

На этих этапах обнаруживаются и устраняются синтаксические и логические ошибки программы. Синтаксические ошибки связаны с нарушением синтаксиса команд, директив транслятора и использованием не определенных ранее меток и имен. Логические ошибки приводят к неправильному функционированию программы. Они связаны с ошибками программы (указан неверный переход при ветвлении, записана не

та команда и т.д.) и ошибками алгоритма. Содержание этих этапов при разработке программ на языке ассемблера приведено ниже.

Стадия создания аппаратуры представляет другой внутренний цикл, выполняемый параллельно с первым. Она содержит разработку общей принципиальной схемы, разводку топологии плат, монтаж макета и его автономную отладку. Эти этапы можно считать завершенными после того, как «оживает» системная магистраль МПС и через нее можно обратиться к памяти, устройствам ввода/вывода. Время выполнения этого этапа зависит от имеющегося набора опробованных функционально-топологических модулей и квалификации разработчика. Распространенными системами проектирования, используемыми на этапе ввода принципиальной схемы и разработки топологии, являются PCAD и OrCAD (CAD – computer aided design – автоматизированное проектирование). Эффективность работы с ними значительно зависит от имеющегося в распоряжении разработчика объема библиотек используемых элементов.

Этап совместной отладки аппаратуры и программного обеспечения в реальном масштабе времени является самым трудоемким и обязательно требует использования таких высокопроизводительных средств, как внутрисхемный эмулятор, эмулятор ПЗУ, логический анализатор. Выбор одного из перечисленных средств обусловлен используемым методом отладки. На этом этапе выявляются динамические ошибки, возникающие при взаимодействии программных и аппаратных средств в реальном масштабе времени. Эти ошибки обусловлены различными задержками распространения сигналов по линиям системной магистрали и взаимными помехами между линиями, возникающими при их неудачном взаимном расположении. Динамические ошибки обнаружить значительно сложнее из-за нерегулярности их появления.

Для локализации динамических ошибок используются логические анализаторы. Логические уровни сигналов системной магистрали или отдельных шин и линий в режиме приема постоянно записываются в память типа FIFO. Прекращение записи производится при появлении выбранного события (совпадение заданного и фактического адреса на ША, кодов команд на ШД или появление короткого импульса помехи). В это время в памяти содержится вся предшествующая данному событию информация. Анализируя предысторию события, записанную в памяти, можно определить и причину появления сбоя в работе МПС. Информация на дисплее может быть представлена в графическом виде, в виде двоичного, шестнадцатеричного кода или мнемоники команд. Логические анализаторы состояний выполняют запись с тактовой частотой МПС. Для фиксации быстро протекающих процессов используются временные логические анализаторы, у которых тактовая частота записи в память значительно превышает тактовую частоту МПС.

Совместная отладка аппаратных и программных средств в реальном масштабе времени выполняется с помощью эмуляторов ПЗУ и внутрисхемных эмуляторов под управлением инструментальной ЭВМ.

Внутрисхемный эмулятор (ВСЭ) включается между микроконтроллером (микропроцессором) и остальной частью целевой МПС. Для этого в панельку микроконтроллера вставляется промежуточная панелька, связанная коротким кабелем с ВСЭ, в которую вставляется микроконтроллер. ВСЭ имеет доступ ко всем ресурсам целевой МПС, что позволяет управлять ее работой, отслеживать поведение системной магистрали, памяти, устройств ввода/вывода [8,9,10].

Этап завершается, когда аппаратура и программное обеспечение совместно обеспечивают выполнение всех шагов алгоритма работы системы. В конце этапа код программы записывается с помощью программатора в энергонезависимую память микроконтроллера и проверяется работа МПС без участия эмулятора. На этапе отладки целесообразно использовать микроконтроллеры с репрограммируемой памятью. Отладка на этом этапе ведется в лабораторных условиях с питанием от источников, обеспечивающих максимальную защиту аппаратуры. Часть внешних источников информации может моделироваться.

Этап интеграции контроллера в изделие заключается в повторении работ по совместной отладке аппаратуры и управляющей программы, но при работе в собственном отсеке изделия, при питании от штатного источника, с информацией от штатных устройств и датчиков. Осложнения на данном этапе возникают, как правило, из-за электромагнитной несовместимости исполнительных устройств разрабатываемой МПС. Особенно много времени на этом этапе уходит на ликвидацию одиночных сбоев, их обнаружение и локализацию. Облегчить решение этой задачи помогает использование логических анализаторов. На этом же этапе выполняется калибровка прибора с занесением параметров в память МПС.

Испытание изделия с МПС можно разделить на комплексные и специальные. Особенностью комплексных испытаний является то, что для наблюдение за МПС в реальных условиях не всегда применимы лабораторные средства отладки. Автономные специализированные средства отладки менее развиты и при этом существенно дороже. Специальные испытания (на электромагнитную совместимость, климатические и т. п.) проводятся по стандартным методикам. После успешного проведения испытаний записывается файл с окончательной версией кода управляющей программы для программатора или завода-изготовителя микроконтроллеров, который осуществит масочное программирование внутренней памяти программ.

3.2. Средства проектирования МПС

Комплекс инструментальных средств, необходимых для проектирования МПС включает средства:

- разработки программного обеспечения,
- ввода принципиальной схемы и проектирования топологии печатной платы,
- автономной отладки аппаратуры,

- совместной отладки аппаратуры и программного обеспечения в реальном масштабе времени,
- программирования БИС памяти программ и программируемой логики.

Общепринятым подходом в настоящее время является использование в качестве ведущего процессора (host-processor) инструментальных средств персональной ЭВМ типа IBM PC. Это позволяет разработчику инструментария переложить на нее общесистемные задачи и сосредоточиться в каждом случае на реализации специфических функций проектирования и отладки. Одновременно персональная ЭВМ служит в качестве объединяющего ядра, в том числе конструктивного, и средством коммуникации в вычислительных сетях. Пользователь получает возможность, кроме средств проектирования, пользоваться широким набором прикладного программного обеспечения для MS DOS и Windows.

Для разработки и отладки МПС необходимы программные (кросс-средства разработки и отладки прикладных программ программирования, система проектирования печатных плат) и аппаратные (логический анализатор, программатор, эмулятор ПЗУ, внутрисхемный эмулятор) средства. Аппаратные средства представляют приборы и системы различной категории сложности и стоимости. Их можно реализовать в виде виртуальных измерительных приборов на персональной ЭВМ, что повышает оперативность работы и одновременно значительно сокращает затраты на аппаратуру.

3.3. Разработка программного обеспечения на языке ассемблера

Стадия создания программы на языке ассемблера, образующая внутренний цикл, представлена на рис. 29. Она содержит этапы разработки текста программы исходного модуля, трансляции ассемблером, редактирования связей, загрузки, отладки программы на модели и коррекции исходного текста.

Разработка программы. На этом этапе создается исходный текст программы на языке ассемблера. В ассемблере доступны все ресурсы программируемой МПС, поэтому программы на ассемблере имеют меньший объем и выполняются быстрее, чем те же программы на языках высокого уровня. Однако создание программы на языке ассемблера более трудоемко, чем на языках высокого уровня. Для ввода текста программы используется любой редактор текста, позволяющий сформировать текстовый файл в кодировке ASCII. Текстовые редакторы в состав традиционного инструментального пакета обычно не входят. При записи текста программы на языке ассемблера страница разбивается на четыре поля: поле меток, поле операции, поле операндов и поле комментариев:

Метка	Операция	Операнды	Комментарии
Init_1:	MOV	SP,#30h	;Загрузить начальный адрес стека
	CLR	TR1	;Запретить работу таймера/счетчикаТС1

Поле метки содержит символическое имя (метку) адреса отмеченной команды или операнда. Метка представляет собой буквенно-цифровую комбинацию, начинающуюся с буквы. Используются буквы только английского алфавита. Метка не должна содержать пробела. Для разделения слов используют символ подчеркивания (_). Длина метки определяется используемым ассемблером и для ассемблера X8051 не должна превышать 32 символов. Метка оканчивается двоеточием (:). В качества метки не может быть использована мнемоника команд, ключевые и зарезервированные слова ассемблера.

Поле операции содержит мнемонику команд, имена макросов, директивы ассемблера. Многие ассемблеры поддерживает запись как строчными, так и прописными буквами.

Поле операндов содержит операнды (или операнд), участвующие в операции. Команды могут быть без-, одно- или двухоперандными. Символом разделения операндов является запятая (,). Ввод дополнительного пробела для улучшения читаемости допустим лишь при включении директивы ассемблера SPACES ON. Стандартным режимом является запрещение использования пробелов.

Используемые в качестве операндов символические имена и метки должны быть предварительно определены, а числа представлены с указанием системы счисления. Ассемблеры допускают использование в поле операнда арифметических выражений, вычисляемых в процессе трансляции.

Поле комментария используется для текстового или символического пояснения логической организации прикладной программы. Это поле полностью игнорируется ассемблером, поэтому в нем допустимо использование любых символов. Каждая строка, содержащая комментарии, должна начинаться точкой с запятой (;).

Текстовый файл ассемблерной программы представляет собой исходный модуль (программу), которому необходимо присвоить расширение .asm.

Ассемблер представляет собой специальную программу, предназначенную для трансляции мнемоники команд исходной текстовой программы (исходного модуля), написанной на языке ассемблера в объектную программу (объектный модуль), содержащую машинные коды команд. Объектный модуль представляет собой промежуточную форму, к которой необходимо присоединить библиотечные средства, содержащие стандартные подпрограммы и процедуры, и добавить другие модули, написанные программистами. Он является перемещаемым модулем, не содержащим физических адресов загрузки, т.е. является неподготовленным для загрузки в ПЗУ МПС и выполнения. В нем все изменяемые при перемещении адреса записаны в таблицу.

Создание объектных модулей необходимо при разработке больших проектов группой программистов. Проект в этом случае делится на части, программа каждой из которых самостоятельно транслируется ассемблером и представляется объектным модулем. Положение

каждого модуля в адресном пространстве памяти остается неизвестным. В этом случае формирование единого загрузочного модуля (программы, готовой для загрузки и исполнения) из нескольких объектных модулей и привязка его к фактическим адресам загрузки конкретной МПС осуществляются на следующем этапе программами редактора связей и загрузчика [4,8,9,10].

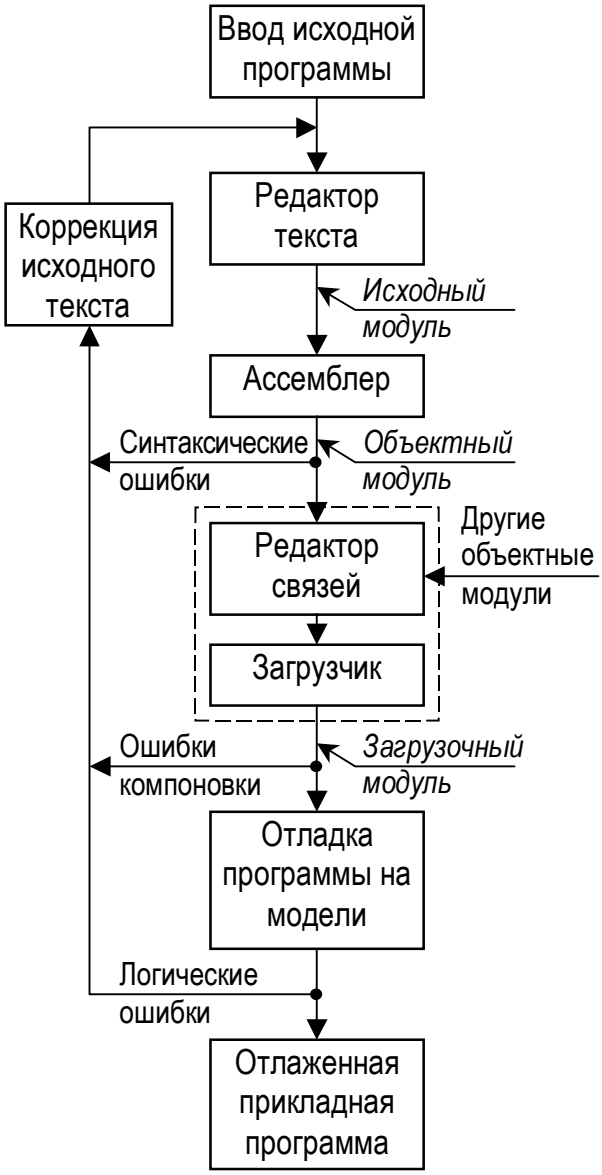


Рис. 29

Ассемблер позволяет получить объектный модуль – файл с расширением .obj и файл листинга трансляции с расширением .lst. Наличие листинга позволяет проанализировать синтаксические ошибки программы и исправить их в исходном текстовом файле.

Редактор связей является специальной программой, обеспечивающей объединение (компоновку) нескольких самостоятельно транслированных объектных модулей, в том числе и библиотечных, в единый модуль с установкой внутренней связи между ними. Поэтому редактор связей часто называют компоновщиком.

Типичные ошибки этого этапа заключаются в неправильном задании начальных адресов модулей, что приводит к перекрытию программ в адресном пространстве памяти и превышению допустимой длины перехода между ними в командах с относительной адресации.

Загрузчик преобразует скомпонованный модуль в программу, предназначенную для использования в конкретной системе. Часто программы компоновщика и загрузчика объединены в одну неделимую программу. Загрузчики могут формировать выходные файлы в различных форматах. Наиболее распространенными являются шестнадцатеричный формат файла фирмы Intel с расширением .hex и двоичный с расширением .tsk. Шестнадцатеричный формат имеет простейшие средства контроля ошибок и наиболее часто используется при загрузке в программаторы ПЗУ. Для загрузки в память МПС в этом случае необходимо использовать специальную программу загрузчика. Двоичный файл с

расширением .tsk представляет машинные коды, которые можно непосредственно загрузить в память МПС.

Простейшие ассемблеры, предназначенные для преобразования единственной программы проекта, содержат в своем составе как неделимое целое редактор связей и загрузчик. Часто они формируют двоичный выходной файл, предназначенный для непосредственной загрузки в память МПС, который имеет расширение не .tsk , а .obj.

Отладка программы на модели. Разработка прикладных программ – это сложный процесс с неизбежными ошибками, которые необходимо обнаружить и исправить. Ошибки могут появиться на разных этапах разработки прикладной программы. Простейшие синтаксические ошибки и ошибки компоновки выявляются на этапе трансляции и компоновки программы. Более сложно обнаруживаются статические логические ошибки, выявляемые лишь при исполнении программы. Для их локализации применяют специальные средства отладки, которые разделяют на резидентные и кросс-средства. Резидентные средства отладки программ имеют инструментальную ЭВМ, тип микропроцессора которой совпадает с целевым, для которого написана отлаживаемая программа. Команды программы исполняются инструментальной ЭВМ. В кросс-средствах отладки тип инструментального и целевого микропроцессоров не совпадают. Команды целевого микропроцессора моделируются инструментальной ЭВМ. Команды выполняются более медленно, но при этом на одной инструментальной ЭВМ можно выполнять отладку программ для различных микропроцессоров, используя соответствующие программы отладчиков (эмуляторов). По этой причине кросс-средства отладки прикладных программ получили наиболее широкое распространение. При отладке фиксируется и анализируется состояние всех регистров и ячеек памяти после выполнения каждой команды, что позволяет проверить правильность вычислений и переходов в программе.

В приложении 3 приведено описание эмулятора EMU-51, предназначенного для отладки прикладных программ микроконтроллеров семейства MCS-51. Эмулятор разработан для учебных целей на кафедре радиотехнических систем РГРТА.

Приложение 1

Условное графическое обозначение и основные электрические характеристики микроконтроллеров подсемейств 51 и С51

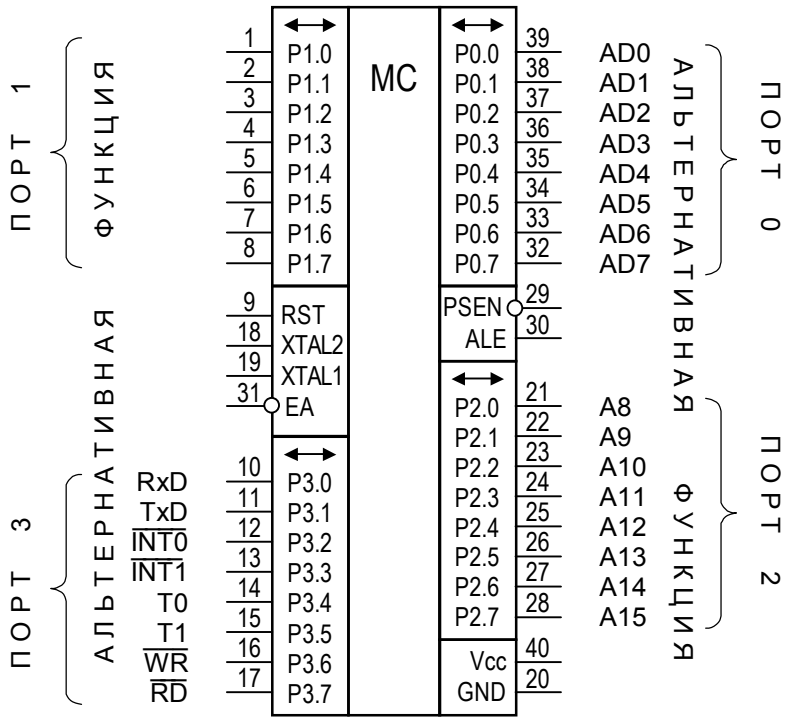


Рис. П1.1

Таблица П1.1. Основные электрические характеристики.

№ п/п	Параметры	Обозначение	Значение параметров			
			n-МОП		КМОП	
			мин	макс	мин	макс
1	Выходное напряжение высокого уровня, В	U^1_{OUT}	2,4		2,4	
2	Выходное напряжение низкого уровня, В	U^0_{OUT}		0,45		0,45
3	Выходной ток высокого уровня, мА P1(0...7), P2(0...7), P3(0...7) : P0(0...7), ALE, $\overline{PSEN}$	I^1_{OUT}	-0,08 -0,4		-0,08 -0,4	
4	Выходной ток низкого уровня, мА P1(0...7), P2(0...7), P3(0...7): P0(0...7), ALE, $\overline{PSEN}$	I^0_{OUT}		1,6 3,2		1,6 3,2
5	Входное напряжение высокого уровня, В	U^1_{IN}	2,0		2,0	
6	Входное напряжение низкого уровня, В	U^0_{IN}	-0,5	0,8	-0,5	0,8
7	Входной ток высокого уровня, мА	I^1_{IN}		-0,50		-0,50
8	Входной ток низкого уровня, мА P0(0...7), P1(0...7), P2(0...7), P3(0...7) : XTAL2	I^0_{IN}		-0,80 -2,5		-0,05
9	Ток потребления, мА	I_{CC}		150,0		18,0
10	Ток потребления в режиме холостого хода, мА	I_{CCIDL}				4,2
11	Ток в режиме микропотребления, мА	I_{CCPD}				0,05
12	Емкость линий ввода/вывода, пФ	$C_{I/O}$		20,0		20,0

Таблица П1.2. Назначение выводов микроконтроллеров подсемейств 51 и С51

№выв.	Обозначен.	Наименование	Тип
1…8	P1.0…P1.7	8-разрядный двунаправленный порт P1	Вход/ Выход
9	RST	Сигнал общего сброса	Вход
10…17	P3.0…P3.7	8-разрядный двунаправленный порт P3 с альтернативными функциями	Вход/ Выход
	P3.0	Последовательные данные приемника RxD	Вход
	P3.1	Последовательные данные передатчика TxD	Выход
	P3.2	Вход внешнего прерывания 0 $\overline{INT0}$	Вход
	P3.3	Вход внешнего прерывания 1 $\overline{INT1}$	Вход
	P3.4	Вход таймера/счетчика 0 T0	Вход
	P3.5	Вход таймера/счетчика 1 T1	Вход
	P3.6	Строб записи во внешний сегмент данных $\overline{WR}$	Выход
	P3.7	Строб чтения из внешнего сегмента данных $\overline{RD}$	Выход
18	XTAL2	Выводы для подключения	Выход
19	XTAL1	кварцевого резонатора	Вход
20	GND	Общий вывод питания	
21…28	P2.0…P2.7	8-разрядный двунаправленный порт P2. Альтернативная функция - шина старшего байта адреса A8…A15 для внешней памяти	Вход/ Выход
29	$\overline{PSEN}$	Чтение из внешнего сегмента памяти CSEG	Выход
30	ALE	Разрешение фиксации адреса	Выход
31	$\overline{EA}$	Блокировка работы с внутренней памятью CSEG	Вход
32…39	P0.7…P0.0	8-разрядный двунаправленный порт P0. Альтернативная функция - шина адреса/данных для внешней памяти	Вход/ Выход
40	Ucc	Вывод питания +5 В	

Приложение 2

Система команд микроконтроллеров семейства MCS-51

Условные обозначения, используемые в командах

A – Аккумулятор
AB – Аккумулятор и регистр B в командах MUL и DIV
C – Флажок C регистра PSW. Выполняет функцию аккумулятора в битовых командах
Rn – Регистр R0…R7 текущего банка регистров
n – Номер регистра текущего банка регистров (n = 0…7)
Ri – Регистр R0, R1 – указатель данных текущего банка регистров
DPTR – 16-разрядный регистр-указатель данных
PC – 16-разрядный программный счетчик
direct – Прямой 8-разрядный адрес байта нижней области DSEG (0…127) или области SFR (128…255)
@Rn – Косвенно адресуемая 8-разрядная ячейка DSEG
#data – Непосредственный 8-разрядный операнд
#data16 – Непосредственный 16-разрядный операнд
addr11 – 11-разрядный адрес в командах ACALL и AJMP
addr16 – 16-разрядный адрес
rel – 8-разрядный байт смещения со знаком (-128…+127)
bit – Прямой 8-разрядный адрес бита BSEG
/bit – Прямой 8-разрядный адрес бита DSEG. Считанный бит инвертируется
– – Признак не устанавливается
+ – Признак устанавливается
* – Признак устанавливается при адресации PSW
(X) – Содержимое элемента X
[X] – Адрес, указанный элементом X
([X]) – Содержимое по адресу, указанному элементом X
[(X)+(Y)] – Адрес равен сумме содержимого элементов X и Y
[(X):(Y)] – Адрес равен конкатенации содержимого элементов X и Y
$(X)_{3...0}$ – Содержимое младшей тетрады элемента X

Таблица П2.1. **КОМАНДЫ ОПЕРАЦИЙ ПЕРЕДАЧИ ДАННЫХ**

НАИМЕНОВАНИЕ КОМАНДЫ	МНЕМОНИКА	КОД	ПРИЗНАКИ C	OV	AC	P	ЧИСЛО Б	Ц	СОДЕРЖАНИЕ КОМАНДЫ
ПЕРЕСЫЛКА в аккумулятор содержимого									
регистра (n = 0...7)	**MOV A, Rn**	11101rrr	–	–	–	+	**1**	**1**	A ← (Rn)
прямо адресуемого байта	**MOV A, direct**	11100101	–	–	–	+	**2**	**1**	A ← (direct)
косвенно адресуемого байта (i = 0,1)	**MOV A, @Ri**	1110011i	–	–	–	+	**1**	**1**	A ← ([Ri])
константы	**MOV A, #data**	01110100	–	–	–	+	**2**	**1**	A ← #data
ПЕРЕСЫЛКА в регистр (n = 0...7) содержимого									
аккумулятора	**MOV Rn, A**	11111rrr	–	–	–	–	**1**	**1**	Rn ← (A)
прямо адресуемого байта	**MOV Rn, direct**	10101rrr	–	–	–	–	**2**	**2**	Rn ← (direct)
константы	**MOV Rn, #data**	01111rrr	–	–	–	–	**2**	**1**	Rn ← #data
ПЕРЕСЫЛКА по прямому адресу содержимого									
аккумулятора	**MOV direct, A**	11110101	*	*	*	–	**2**	**1**	direct ← (A)
регистра (n = 0...7)	**MOV direct, Rn**	10001rrr	*	*	*	–	**2**	**2**	direct ← (Rn)
прямо адресуемого байта	**MOV direct, direct**	10000101	*	*	*	–	**3**	**2**	direct ← (direct)
косвенно адресуемого байта (i = 0,1)	**MOV direct, @Ri**	1000011i	*	*	*	–	**2**	**2**	direct ← ([Ri])
константы	**MOV direct, #data**	01110101	*	*	*	–	**3**	**2**	direct ← #data
ПЕРЕСЫЛКА по косвенному адресу содержимого									
аккумулятора	**MOV @Ri, A**	1111011i	–	–	–	–	**1**	**1**	[Ri] ← (A)
прямо адресуемого байта	**MOV @Ri, direct**	1010011i	–	–	–	–	**2**	**2**	[Ri] ← (direct)
константы	**MOV @Ri, #data**	0111011i	–	–	–	–	**2**	**1**	[Ri] ← #data
ЗАГРУЗКА указателя данных	**MOV DPTR, #data16**	10010000	–	–	–	–	**3**	**2**	DPTR ← #data16
ПЕРЕСЫЛКА в аккумулятор байта									
из CSEG по базовому адресу в DPTR	**MOVC A, @A + DPTR**	10010011	–	–	–	+	**1**	**2**	A ← CSEG[(A) + (DPTR)]
из CSEG по базовому адресу в PC	**MOVC A, @A + PC**	10000011	–	–	–	+	**1**	**2**	PC ← (PC) + 1; A ← CSEG[(A)+(PC)]
из страницы XSEG	**MOVX A, @Ri**	1110001i	–	–	–	+	**1**	**2**	A ← XSEG[(P2):(Ri)]
из области XSEG	**MOVX A, @DPTR**	11100000	–	–	–	+	**1**	**2**	A ← XSEG[DPTR]
ПЕРЕСЫЛКА байта из аккумулятора									
в страницу XSEG	**MOVX @Ri, A**	1111001i	–	–	–	–	**1**	**2**	XSEG[(P2):(Ri)] ← (A)
в область XSEG	**MOVX @DPTR, A**	11110000	–	–	–	–	**1**	**2**	XSEG[DPTR] ← (A)
ЗАПИСЬ прямо адресуемого байта в стек	**PUSH direct**	11000000	–	–	–	–	**2**	**2**	SP ← (SP) + 1; [SP] ← (direct)
ЧТЕНИЕ из стека в прямо адресуемую ячейку	**POP direct**	11010000	*	*	*	–	**2**	**2**	direct ← (SP); SP ← (SP) - 1
ОБМЕН									
аккумулятора с регистром (n = 0...7)	**XCH A, Rn**	11001rrr	–	–	–	+	**1**	**1**	(A) ↔ (Rn)
аккумулятора с прямо адресуемым байтом	**XCH A, direct**	11000101	*	*	*	+	**2**	**1**	(A) ↔ (direct)
аккумулятора с байтом из DSEG (i = 0,1)	**XCH A, @Ri**	1100011i	–	–	–	+	**1**	**1**	(A) ↔ ([Ri])
младших тетрад аккумулятора и байта из DSEG	**XCHD A, @Ri**	1101011i	–	–	–	+	**1**	**1**	$(A)_{3...0}$ ↔ $([Ri])_{3...0}$

Таблица П2.2. **КОМАНДЫ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ**

НАИМЕНОВАНИЕ КОМАНДЫ	МНЕМОНИКА	КОД	ПРИЗНАКИ				ЧИСЛО		СОДЕРЖАНИЕ КОМАНДЫ
			C	OV	AC	P	Б	Ц	
СЛОЖЕНИЕ аккумулятора									
с регистром (n = 0...7)	**ADD A, Rn**	00101rrr	+	+	+	+	**1**	**1**	A ← (A) + (Rn)
с прямо адресуемым байтом	**ADD A, direct**	00100101	+	+	+	+	**2**	**1**	A ← (A) + (direct)
с косвенно адресуемым байтом (i = 0,1)	**ADD A, @Ri**	0010011i	+	+	+	+	**1**	**1**	A ← (A) + ([Ri])
с константой	**ADD A, #data**	00100100	+	+	+	+	**2**	**1**	A ← (A) + #data
СЛОЖЕНИЕ С ПЕРЕНОСОМ аккумулятора									
с регистром (n = 0...7)	**ADDC A, Rn**	00111rrr	+	+	+	+	**1**	**1**	A ← (A) + (Rn) + (C)
с прямо адресуемым байтом	**ADDC A, direct**	00110101	+	+	+	+	**2**	**1**	A ← (A) + (direct) + (C)
с косвенно адресуемым байтом (i = 0,1)	**ADDC A, @Ri**	0011011i	+	+	+	+	**1**	**1**	A ← (A) + ([Ri]) + (C)
с константой	**ADDC A, #data**	00110100	+	+	+	+	**2**	**1**	A ← (A) + #data + (C)
ВЫЧИТАНИЕ С ЗАЁМОМ из аккумулятора									
регистра (n = 0...7)	**SUBB A, Rn**	10011rrr	+	+	+	+	**1**	**1**	A ← (A) - (C) - (Rn)
прямо адресуемого байта	**SUBB A, direct**	10010101	+	+	+	+	**2**	**1**	A ← (A) - (C) - (direct)
косвенно адресуемого байта (i = 0,1)	**SUBB A, @Ri**	1001011i	+	+	+	+	**1**	**1**	A ← (A) - (C) - ([Ri])
константы	**SUBB A, #data**	10010100	+	+	+	+	**2**	**1**	A ← (A) - (C) - #data
ИНКРЕМЕНТ									
аккумулятора	**INC A**	00000100	–	–	–	+	**1**	**1**	A ← (A) + 1
регистра (n = 0...7)	**INC Rn**	00001rrr	–	–	–	–	**1**	**1**	Rn ← (Rn) + 1
прямо адресуемого байта	**INC direct**	00000101	–	–	–	–	**2**	**1**	direct ← (direct) + 1
косвенно адресуемого байта (i = 0,1)	**INC @Ri**	0000011i	–	–	–	–	**1**	**1**	[Ri] ← ([Ri]) + 1
указателя данных	**INC DPTR**	10100011	–	–	–	–	**1**	**2**	DPTR ← (DPTR) + 1
ДЕКРЕМЕНТ									
аккумулятора	**DEC A**	00010100	–	–	–	+	**1**	**1**	A ← (A) - 1
регистра (n = 0...7)	**DEC Rn**	00011rrr	–	–	–	–	**1**	**1**	Rn ← (Rn) - 1
прямо адресуемого байта	**DEC direct**	00010101	–	–	–	–	**2**	**1**	direct ← (direct) - 1
косвенно адресуемого байта (i = 0,1)	**DEC @Ri**	0001011i	–	–	–	–	**1**	**1**	[Ri] ← ([Ri]) - 1
УМНОЖЕНИЕ аккумулятора на регистр B	**MUL AB**	10100100	0	+	–	+	**1**	**4**	(B)(A) ← (A) x (B)
ДЕЛЕНИЕ аккумулятора на регистр B	**DIV AB**	10000100	0	+	–	+	**1**	**4**	(A),(B) ← (A)/(B)
ДЕСЯТИЧНАЯ КОРРЕКЦИЯ аккумулятора	**DA A**	11010100	+	–	–	+	**1**	**1**	Если $(A_3...A_0) > 9$ или (AC) = 1, то $(A_3...A_0) \leftarrow (A_3...A_0) + 6$; если $(A_7...A_4) > 9$ или (C) = 1, то $(A_7...A_4) \leftarrow (A_7...A_4) + 6$

Таблица П2.3. **КОМАНДЫ ЛОГИЧЕСКИХ ОПЕРАЦИЙ**

НАИМЕНОВАНИЕ КОМАНДЫ	МНЕМОНИКА	КОД	ПРИЗНАКИ C	OV	AC	P	ЧИСЛО Б	Ц	СОДЕРЖАНИЕ КОМАНДЫ
ЛОГИЧЕСКОЕ "И" аккумулятора									
с регистром (n = 0...7)	**ANL A, Rn**	01011rrr	–	–	–	+	**1**	**1**	$A \leftarrow (A) \wedge (Rn)$
с прямо адресуемым байтом	**ANL A, direct**	01010101	–	–	–	+	**2**	**1**	$A \leftarrow (A) \wedge (direct)$
с косвенно адресуемым байтом (i = 0,1)	**ANL A, @Ri**	0101011i	–	–	–	+	**1**	**1**	$A \leftarrow (A) \wedge ([Ri])$
с константой	**ANL A, #data**	01010100	–	–	–	+	**2**	**1**	$A \leftarrow (A) \wedge$ #data
ЛОГИЧЕСКОЕ "И" прямо адресуемого байта									
с аккумулятором	**ANL direct, A**	01010010	–	–	–	–	**2**	**1**	direct $\leftarrow$ (direct) $\wedge$ (A)
с константой	**ANL direct, #data**	01010011	–	–	–	–	**3**	**2**	direct $\leftarrow$ (direct) $\wedge$ #data
ЛОГИЧЕСКОЕ "ИЛИ" аккумулятора									
с регистром (n = 0...7)	**ORL A, Rn**	01001rrr	–	–	–	+	**1**	**1**	$A \leftarrow (A) \vee (Rn)$
с прямо адресуемым байтом	**ORL A, direct**	01000101	–	–	–	+	**2**	**1**	$A \leftarrow (A) \vee (direct)$
с косвенно адресуемым байтом (i = 0,1)	**ORL A, @Ri**	0100011i	–	–	–	+	**1**	**1**	$A \leftarrow (A) \vee ([Ri])$
с константой	**ORL A, #data**	01000100	–	–	–	+	**2**	**1**	$A \leftarrow (A) \vee$ #data
ЛОГИЧЕСКОЕ "ИЛИ" прямо адресуемого байта									
с аккумулятором	**ORL direct, A**	01000010	–	–	–	–	**2**	**1**	direct $\leftarrow$ (direct) $\vee$ (A)
с константой	**ORL direct, #data**	01000011	–	–	–	–	**3**	**2**	direct $\leftarrow$ (direct) $\vee$ #data
"ИСКЛЮЧАЮЩЕЕ ИЛИ" аккумулятора									
с регистром (n = 0...7)	**XRL A, Rn**	01101rrr	–	–	–	+	**1**	**1**	$A \leftarrow (A) \oplus (Rn)$
с прямо адресуемым байтом	**XRL A, direct**	01100101	–	–	–	+	**2**	**1**	$A \leftarrow (A) \oplus (direct)$
с косвенно адресуемым байтом (i = 0,1)	**XRL A, @Ri**	0110011i	–	–	–	+	**1**	**1**	$A \leftarrow (A) \oplus ([Ri])$
с константой	**XRL A, #data**	01100100	–	–	–	+	**2**	**1**	$A \leftarrow (A) \oplus$ #data
"ИСКЛЮЧАЮЩЕЕ ИЛИ" прямо адресуемого байта									
с аккумулятором	**XRL direct, A**	01100010	–	–	–	–	**2**	**1**	direct $\leftarrow$ (direct) $\oplus$ (A)
с константой	**XRL direct, #data**	01100011	–	–	–	–	**3**	**2**	direct $\leftarrow$ (direct) $\oplus$ #data
СДВИГ циклический аккумулятора									
влево	**RL A**	00100011	–	–	–	–	**1**	**1**	$(A_7 \leftarrow A_6 \leftarrow A_5 \leftarrow ... \leftarrow A_1 \leftarrow A_0 \leftarrow A_7)$
влево через перенос	**RLC A**	00110011	+	–	–	+	**1**	**1**	$(C \leftarrow A_7 \leftarrow A_6 \leftarrow ... \leftarrow A_1 \leftarrow A_0 \leftarrow (C))$
вправо	**RR A**	00000011	–	–	–	–	**1**	**1**	$(A_0 \rightarrow A_7 \rightarrow A_6 \rightarrow ... \rightarrow A_2 \rightarrow A_1 \rightarrow A_0)$
вправо через перенос	**RRC A**	00010011	+	–	–	+	**1**	**1**	$((C) \rightarrow A_7 \rightarrow A_6 \rightarrow ... \rightarrow A_1 \rightarrow A_0 \rightarrow C)$
СБРОС аккумулятора	**CLR A**	11100100	–	–	–	+	**1**	**1**	$(A) \leftarrow 0$
ИНВЕРСИЯ аккумулятора	**CPL A**	11110100	–	–	–	+	**1**	**1**	$A \leftarrow \overline{(A)}$
ПЕРЕСТАНОВКА тетрад в аккумуляторе	**SWAP A**	11000100	–	–	–	–	**1**	**1**	$(A_7...A_4) \leftrightarrow (A_3...A_0)$

Таблица П2.4. **КОМАНДЫ ОПЕРАЦИЙ ПЕРЕДАЧИ УПРАВЛЕНИЯ**

НАИМЕНОВАНИЕ КОМАНДЫ	МНЕМОНИКА	КОД	ПРИЗНАКИ C OV AC P	ЧИСЛО Б	ЧИСЛО Ц	СОДЕРЖАНИЕ КОМАНДЫ
ПЕРЕХОД безусловный,						
длинный внутри CSEG [0...64 Кбайт]	**LJMP addr16**	00000010	– – – –	**3**	**2**	PC ← addr16
абсолютный внутри страницы в 2 Кбайт	**AJMP addr11**	aaa00001	– – – –	**2**	**2**	$(PC_{10}...PC_0)$ ← addr11
короткий относительный [–128...+127]	**SJMP rel**	10000000	– – – –	**2**	**2**	PC ← (PC) + 2; PC ← (PC) + rel
косвенный относительный	**JMP @A + DPTR**	01110011	– – – –	**1**	**2**	PC ← (A) + (DPTR)
ПЕРЕХОД условный,						
если аккумулятор равен нулю	**JZ rel**	01100000	– – – –	**2**	**2**	PC ← (PC) + 2; если (A) = 0, то PC ← (PC) + rel
если аккумулятор не равен нулю	**JNZ rel**	01110000	– – – –	**2**	**2**	PC ← (PC) + 2; если (A) ≠ 0, то PC ← (PC) + rel
если перенос равен единице	**JC rel**	01000000	– – – –	**2**	**2**	PC ← (PC) + 2; если (C) = 1, то PC ← (PC) + rel
если перенос равен нулю	**JNC rel**	01010000	– – – –	**2**	**2**	PC ← (PC) + 2; если (C) = 0, то PC ← (PC) + rel
если бит равен единице	**JB bit, rel**	00100000	– – – –	**3**	**2**	PC ← (PC) + 3; если (bit) = 1, то PC ← (PC) + rel
если бит равен нулю	**JNB bit, rel**	00110000	– – – –	**3**	**2**	PC ← (PC) + 3; если (bit) = 0, то PC ← (PC) + rel
если бит установлен, с последующим сбросом бита	**JBC bit, rel**	00010000	– – – –	**3**	**2**	PC ← (PC) + 3; если (bit) = 1, то PC ← (PC) + rel и (bit) ← 0
ДЕКРЕМЕНТ регистра и **ПЕРЕХОД,** если не нуль	**DJNZ Rn, rel**	11011rrr	– – – –	**2**	**2**	PC ← (PC) + 2, Rn ← (Rn) - 1; если (Rn) ≠ 0, то PC ← (PC) + rel
ДЕКРЕМЕНТ прямо адресуемого байта и **ПЕРЕХОД,** если не нуль	**DJNZ direct, rel**	11010101	– – – –	**3**	**2**	PC ← (PC) + 3, direct ← (direct) - 1; если (direct) ≠ 0, то PC ← (PC) + rel

Таблица П2.5. **КОМАНДЫ ОПЕРАЦИЙ ПЕРЕДАЧИ УПРАВЛЕНИЯ**

НАИМЕНОВАНИЕ КОМАНДЫ	МНЕМОНИКА	КОД	ПРИЗНАКИ				ЧИСЛО		СОДЕРЖАНИЕ КОМАНДЫ
			C	OV	AC	P	Б	Ц	
СРАВНЕНИЕ аккумулятора с прямо адресуемым байтом и **ПЕРЕХОД**, если не равно	**CJNE A, direct, rel**	10110101	+	–	–	–	**3**	**2**	PC ← (PC) + 3; если (A) ≠ (direct), то PC ← (PC) + rel; причем если (A) < (direct), то C ← 1; если (A) > (direct), то C ← 0
СРАВНЕНИЕ аккумулятора с константой и **ПЕРЕХОД**, если не равно	**CJNE A, #data, rel**	10110100	+	–	–	–	**3**	**2**	PC ← (PC) + 3; если (A) ≠ #data , то PC ← (PC) + rel; причем если (A) < #data , то C ← 1; если (A) > #data , то C ← 0
СРАВНЕНИЕ регистра с константой и **ПЕРЕХОД**, если не равно	**CJNE Rn, #data, rel**	10111rrr	+	–	–	–	**3**	**2**	PC ← (PC) + 3; если (Rn) ≠ #data, то PC ← (PC) + rel; причем если (Rn) < #data , то C ← 1; если (Rn) > #data , то C ← 0
СРАВНЕНИЕ байта из DSEG с константой и **ПЕРЕХОД**, если не равно	**CJNE @Ri, #data, rel**	1011011i	+	–	–	–	**3**	**2**	PC ← (PC) + 3; если ([Ri]) ≠ #data, то PC ← (PC) + rel; причем если ([Ri]) < #data, то C ← 1; если ([Ri]) > #data, то C ← 0
ВЫЗОВ подпрограммы длинный внутри CSEG [0...64 Кбайт]	**LCALL addr16**	00010010	–	–	–	–	**3**	**2**	PC ← (PC) + 3; SP ← (SP) + 1, [SP] ← (PC_7...PC_0); SP ← (SP) + 1, [SP] ← (PC_{15}...PC_8) PC ← addr16
ВЫЗОВ подпрограммы абсолютный внутри страницы CSEG [0...2 Кбайт]	**ACALL addr11**	aaa00001	–	–	–	–	**2**	**2**	PC ← (PC) + 2; SP ← (SP) + 1, [SP] ← (PC_7...PC_0); SP ← (SP) + 1, [SP] ← (PC_{15}...PC_8) (PC_{10}...PC_0) ← addr11
ВОЗВРАТ из подпрограммы	**RET**	00100010	–	–	–	–	**1**	**2**	(PC_{15}...PC_8) ← ([SP]), SP ← (SP) - 1 (PC_7...PC_0) ← ([SP]), SP ← (SP) - 1
ВОЗВРАТ из подпрограммы обработки прерывания	**RETI**	00110010	–	–	–	–	**1**	**2**	(PC_{15}...PC_8) ← ([SP]), SP ← (SP) - 1 (PC_7...PC_0) ← ([SP]), SP ← (SP) - 1
ХОЛОСТАЯ КОМАНДА	**NOP**	00000000	–	–	–	–	**1**	**1**	PC ← (PC) + 1

Таблица П2.6. **КОМАНДЫ ОПЕРАЦИЙ С БИТАМИ**

НАИМЕНОВАНИЕ КОМАНДЫ	МНЕМОНИКА	КОД	ПРИЗНАКИ C	OV	AC	P	ЧИСЛО Б	Ц	СОДЕРЖАНИЕ КОМАНДЫ
СБРОС									
переноса	**CLR C**	11000011	0	–	–	–	**1**	**1**	C ← 0
бита	**CLR bit**	11000010	*	*	*	–	**2**	**1**	bit ← 0
УСТАНОВКА									
переноса	**SETB C**	11010011	1	–	–	–	**1**	**1**	C ← 1
бита	**SETB bit**	11010010	*	*	*	–	**2**	**1**	bit ← 1
ИНВЕРСИЯ									
переноса	**CPL C**	10110011	$\overline{C}$	–	–	–	**1**	**1**	C ← $(\overline{C})$
бита	**CPL bit**	10110010	*	*	*	–	**2**	**1**	bit ← $\overline{(bit)}$
ЛОГИЧЕСКОЕ "И"									
бита и переноса	**ANL C, bit**	10000010	+	–	–	–	**2**	**2**	C ← (C) ∧ (bit)
инверсии бита и переноса	**ANL C, /bit**	10110000	+	–	–	–	**2**	**2**	C ← (C) ∧ $\overline{(bit)}$
ЛОГИЧЕСКОЕ "ИЛИ"									
бита и переноса	**ORL C, bit**	01110010	+	–	–	–	**2**	**2**	C ← (C) ∨ (bit)
инверсии бита и переноса	**ORL C, /bit**	10100000	+	–	–	–	**2**	**2**	C ← (C) ∨ $\overline{(bit)}$
ПЕРЕСЫЛКА									
бита в перенос	**MOV C, bit**	10100010	+	–	–	–	**2**	**1**	C ← (bit)
переноса в бит	**MOV bit, C**	10010010	–	*	*	–	**2**	**2**	bit ← (C)

Приложение 3.
Отладка программ эмулятором EMU-51

Эмулятор (отладчик) представляет собой системную программу, предназначенную для локализации и исправления логических ошибок в пользовательской программе.

Рассматриваемый эмулятор EMU-51 работает в среде MS-DOS. Запустить его можно из Windows, открыв файл emu51.exe. Переключение между полноэкранным и оконным режимами работы в MS-DOS производится клавишами ALT+ENTER. Желательно использовать полноэкранный режим. Перед выходом из MS-DOS необходимо завершить работу эмулятора, нажав F10.

Все числа в эмуляторе вводятся в шестнадцатеричной системе счисления (по умолчанию).

Перед запуском эмулятора необходимо сформировать из текстового файла с расширением .asm, содержащего отлаживаемую программу на языке ассемблера, исполняемый файл, т.е. файл, который будет загружен в ПЗУ микропроцессора. Отлаживаемый файл должен иметь расширение .tsk или .obj и находиться в одной папке с программой эмулятора.

После загрузки эмулятора на экран дисплея выводятся 4 окна: сегмент регистров и регистры специальных функций «RSEG & SFR», файл программы «ФАЙЛ : », сегмент памяти данных «DSEG» и «КОММЕНТАРИИ», а также строка состояния в нижней части экрана (рис. П3.1). В строке состояния указываются клавиши, активные для данного состояния эмулятора. Кроме фиксированных окон в режиме редактирования выводятся окна программного кода «CSEG», внешней памяти данных «XSEG» и регистров управления из области SFR.

Эмулятор может находиться в основном режиме или в режиме редактирования содержимого окон. В основном режиме функциональные клавиши выполняют следующие действия:

F1 – вызов контекстной помощи,
F3 – перезагрузка эмулятора,
F4 – переход в режим редактирования содержимого окон,
F5 – ввод адреса в программный счетчик,
F6 – непрерывный или пошаговый режим работы эмулятора,
F7 – включить или выключить отображение данных в окнах,
F8 – выполнение команды, выделенной курсором в окне «ФАЙЛ : »,
F10 – выход из эмулятора

В окно «**КОММЕНТАРИИ**» выводятся сообщения о работе эмулятора. После запуска эмулятора в первой строке окна предлагается ввести имя исполняемого файла с расширением. Расширение отделяется от имени файла точкой и должно быть .tsk или .obj. На рис. П3.1 показан ввод файла test.tsk. После завершения ввода имени файла и исправления возможных ошибок в его имени или расширении необходимо нажать клавишу ENTER. Во время работы эмулятора в первой строке отображается число выполненных машинных циклов, во второй – режим его работы (F6), в третьей – состояние окон (F7), а в

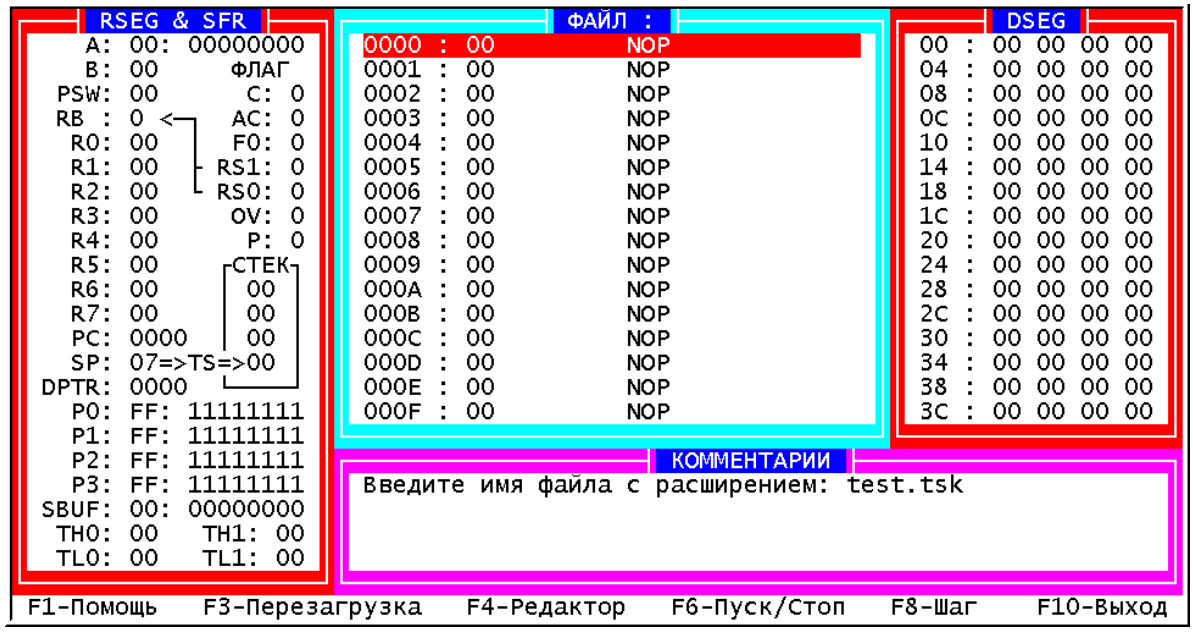


Рис. П3.1

режим его работы (F6), в третьей – состояние окон (F7), а в четвертой – комментируются результаты действий и возможные ошибки.

В окне «**ФАЙЛ :** » после успешной загрузки отлаживаемого файла отображаются в шестнадцатеричной системе счисления текущие 16 адресов команд, сами команды, а также их мнемоника. Имя окна дополнится именем загружаемого файла (рис. П3.2). Курсором окна выделяется команда, которая будет выполнена после нажатия клавиши F8 - «Шаг».

При выполнении программы в окне регистров «**RSEG & SFR**» отображается состояние регистров **R0...R7** текущего регистрового банка, номер которого указан в строке **RB**. Содержимое аккумулятора **A,**

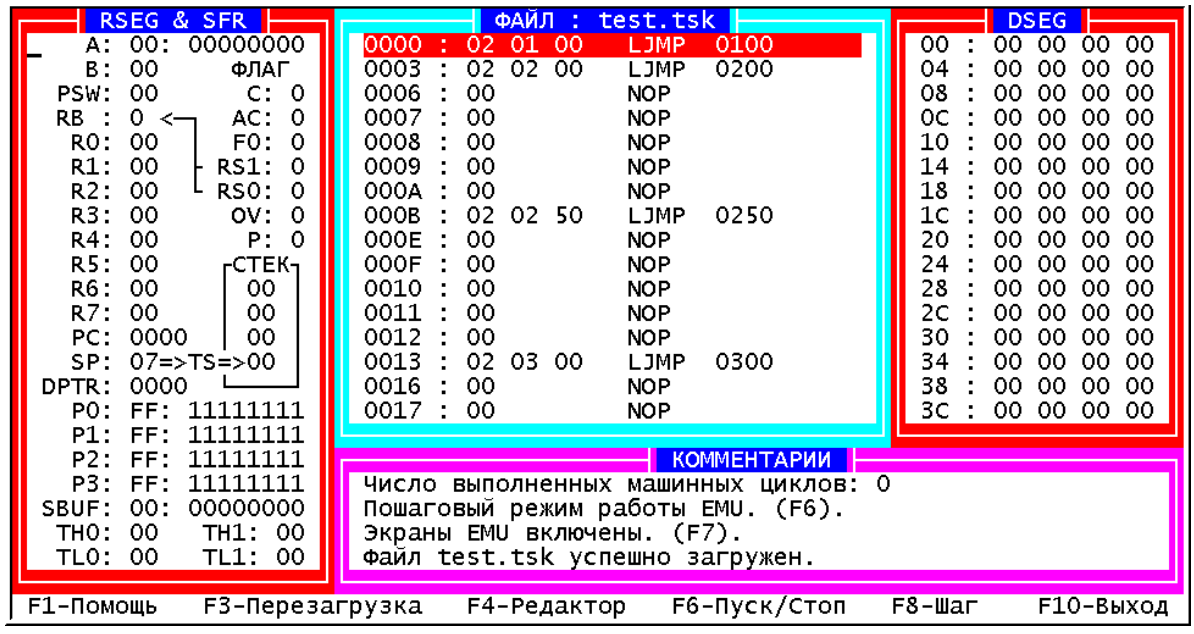


Рис. П3.2

портов **P0**, **P1**, **P2**, **P3** и буфера последовательного порта **SBUF** представлено как в байтовом, так и битовом форматах. Отображается состояние программного счетчика **PC,** указателя вершины стека **SP** и регистра-указателя **DPTR**. В окошко **СТЕК** выводится текущее состояние 4 соседних ячеек стека, начиная с вершины стека **TS**, на которую указывает **SP**. Стек заполняется в сторону увеличения адресов.

Слово состояния **PSW** представлено как байтом, так и отдельными битами в поле «ФЛАГ»:

C – флаг переноса из старшего разряда аккумулятора,
AC – флаг переноса из младшей тетрады аккумулятора в старшую,
F0 – флаг общего назначения, определяемый пользователем,
OV – флаг переполнения в арифметических операциях,
P – флаг паритета (четность числа единиц в байте).

Кроме того, в этом поле выведено содержимое младшего (**RS0**) и старшего (**RS1**) бита номера регистрового банка, указанного в строке **RB**.

В нижней части окна показано содержимое младшего (**TL**) и старшего (**TH**) байт таймеров/счетчиков TC0 и TC1.

В окне «**DSEG**» отображается текущее состояние 64 ячеек памяти. Окно содержит поле адресов и поле данных. В каждой строке поля адресов указан адрес первой в данной строке ячейки памяти.

Редактирование содержимого регистров и памяти

Для входа в режим редактирования содержимого окон надо нажать клавишу F4. Строка состояния имеет следующий вид (рис. П3.3):

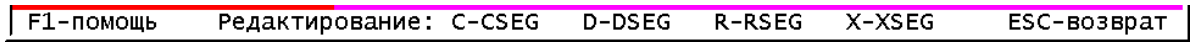


Рис. П3.3

Для редактирования сегмента программного кода (CSEG) нажать клавишу **C**, сегмента данных (DSEG) – **D**, сегмента регистров (RSEG) - **R**, сегмента внешней памяти данных (XSEG) - **X**, регистров управления (IP, IE, TMOD, TCON, SCON, PCON) – **K**.

Строка состояния в режиме редактирования указывает на способы навигации в редактируемом окне и на действия для ввода или отмены введенных изменений:

← - на позицию влево,	HOME - в начало строки,
→ - на позицию вправо,	END - в конец строки,
↑ - на строку вверх,	PgUp - на страницу вверх,
↓ - на строку вниз,	PgDn - на страницу вниз,
ENTER - ввод изменений,	Ctrl+PgUp - в начало памяти,
ESC - отмена изменений,	Ctrl+PgDn - в конец памяти.

Новые значения вводятся в позицию курсора и выделяются инверсным цветом. После ввода (ENTER) или отмены (ESC) выделение снимается, а содержимое окна обновляется.

Редактирование сегмента регистров (нажать клавишу **R**). Для перемещения курсора между полями шестнадцатеричного и двоичного

представлений содержимого регистров и полем «ФЛАГ» нажать клавишу табуляции (Tab).

Содержимое программного счетчика PC, указателя стека SP**,** номера регистрового банка RB и окошка стека не редактируется**.** Курсор в их позициях не устанавливается. Необходимый регистровый банк RB можно установить изменением содержимого младшего (RS0) или старшего (RS1) бита его номера в поле «ФЛАГ». На рис. П3.4 в регистры R1, R4 и R6 записаны числа 2A, 1F и 2D, а в младшую тетраду регистра R7 - 0. Изменения не введены.

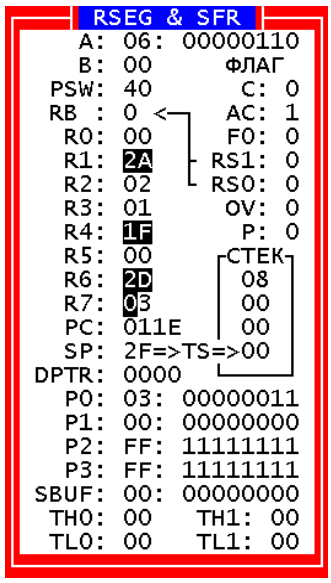


Рис. П3.4

Редактирование сегментов данных (DSEG, XSEG) и программного кода (CSEG) становится доступным при нажатии клавиши D, X или C. Окно «**DSEG**» постоянно находится на экране, а окна «**CSEG**» и «**XSEG**» отображаются только при их редактировании. Каждое окно состоит из двух полей: поля адресов и поля данных, разделенных двоеточием (:). В поле адресов отсутствует автоматическое смещение курсора после ввода символа, поэтому нужно воспользоваться клавишами сдвига вправо или влево. В поле данных курсор смещается автоматически.

Если курсор находится в одном из полей и в него введены изменения (изменения выделены инверсным цветом), то перейти в другую область можно лишь после их ввода (ENTER) или отмены (ESC).

При отсутствии изменений перейти из области адресов в область данных можно клавишами смещения вправо и END, а назад - клавишами смещения влево и HOME, причем END устанавливает курсор в последнюю позицию строки поля данных, а HOME – в первую позицию поля адресов этой строки. При наличии изменений, выделенных инверсным цветом, клавишей END курсор смещается в последнюю, а клавишей HOME – в первую позицию строки поля данных.

Для отображения в окне любого сегмента памяти необходимо набрать начальный адрес сегмента в любой строке области адреса и нажать клавишу ENTER.

Когда курсор находится в поле данных, в соответствующей его положению строке поля адреса инверсным цветом указывается адрес ячейки памяти, в которой он находится. На рис. П3.5 показано редактирование сегмента данных «DSEG». Курсор находится в позиции ячейки памяти с адресом 1Eh. В ячейки памяти 10h…15h записаны (но не введены) числа 20h…25h соответственно, в старшие тетрады ячеек 16h и 1Eh записаны числа 2 и 3. Все они выделены инверсным цветом.

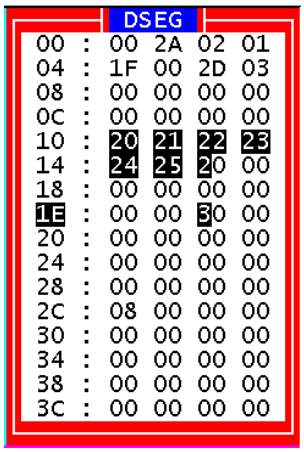


Рис. П3.5

На рис. П3.6 показано редактирование сегмента программного кода «CSEG». Курсор находится в позиции младшей тетрады ячейки памяти с адресом 011Eh, в старшую тетраду которой записано число Ah. В младшие тетрады ячеек памяти с адресами 011Bh и 011Dh записаны (но не введены) числа 5h и Dh соответственно. Они выделены инверсным цветом. После введения (ENTER), изменения дизассемблируются и показываются в окне «ФАЙЛ : » в виде кода и мнемоники. При редактировании кода программы необходимо выполнить следующие условия.

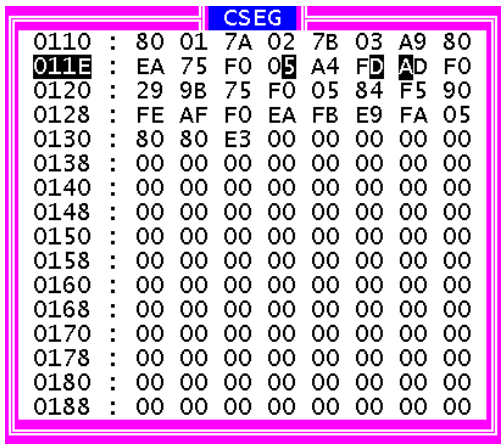


Рис. П3.6

Если заменяется одна команда на другую такой же длины (имеющей то же число байт), то вместо кодов прежней команды записываются коды новой команды.

Если новая команда короче (имеет меньшее число байт), то лишние байты надо заполнить нулями, т.е. ввести команду NOP. Если ситуация противоположная, то новый фрагмент кода нужно разместить после всей программы, введя команду безусловного перехода LJMP <B2,B3> (<B2,B3> - адрес начала нового фрагмента). Если заменяется однобайтовая команда, то трехбайтовая команда безусловного перехода может занять ячейки одной или двух следующих команд. Их нужно внести в новый фрагмент, заполнив нулями оставшиеся не занятыми байты после записи команды LJMP. В конце нового фрагмента командой безусловного перехода необходимо вернуться назад.

Редактирование регистров управления (IP, IE, TMOD, TCON, SCON, PCON) становится доступным при нажатии клавиши **K** в режиме редактирования (рис. П3.7). Содержимое каждого регистра представлено в двух полях с шестнадцатеричной и двоичной системой счисления. Переключение между полями выполняется клавишей табуляции «Tab» только в случае отсутствия выделенных изменений. При навигации курсор находится в поле с выделенными изменениями. Навигация в окне не отличается от рассмотренных выше случаев.

На рис. П3.7 курсор находится в позиции регистра IE. В разряды IE.2 (флаг EX1) и IE.3 (флаг ET1) записаны (но не введены) соответственно 0 и 1. При вводе этих изменений будут разрешены (IE.7 = 1) внешнее прерывание INT0 (IE.0 = 1), прерывания от таймеров/счетчиков TC0 (IE.1 = 1) и TC1 (IE.3 = 1). Внешнее прерывание INT1 (IE.2 = 0) и последовательного порта (IE.4 = 0) запрещены. Для ввода изменений нажать ENTER, а для их отмены – ESC.

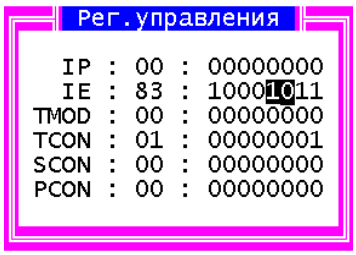


Рис. П3.7

Выполнение прерываний

Прерывания, если они разрешены, можно выполнить как в пошаговом, так и непрерывном режимах. Для разрешения прерываний в программе необходимо установить бит IE.7 (EA= 1) и разрешить прерывание от выбранных источников. При запуске эмулятора регистр IE обнуляется, что запрещает все прерывания.

Запрос на прерывание вводится в любой момент времени набором Alt+<S>, где S – буквенное имя прерывания. В эмуляторе EMU-51 внешним прерываниям присвоены следующие буквенные имена:

i – внешнее прерывание INT0;
j – внешнее прерывание INT1;
k – прерывание таймера/счетчика TC0;
u – прерывание таймера/счетчика TC1.

Например, для ввода запроса на прерывание INT0 необходимо при удерживаемой в нажатом состоянии клавише Alt нажать буквенную клавишу i (выполнить Alt+i**)**. В 4-й строке окна «КОММЕНТАРИИ» появится сообщение «Установлен запрос прерывания INT0 по фронту (IT0=1)», если прерывание INT0 разрешено (рис. П3.8). После выполнения очередной команды по адресу 011Eh (нажать F8) содержимое программного счетчика PC будет 0120h, что соответствует адресу первого байта следующей команды. Сообщение в 4-й строке изменилось на «Прерывание INT0 готово к выполнению.» и счетчик машинных циклов увеличился на 2 (рис. П3.9).

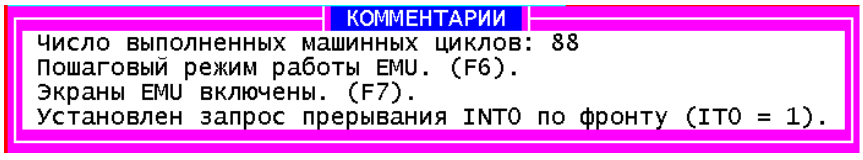


Рис. П3.8

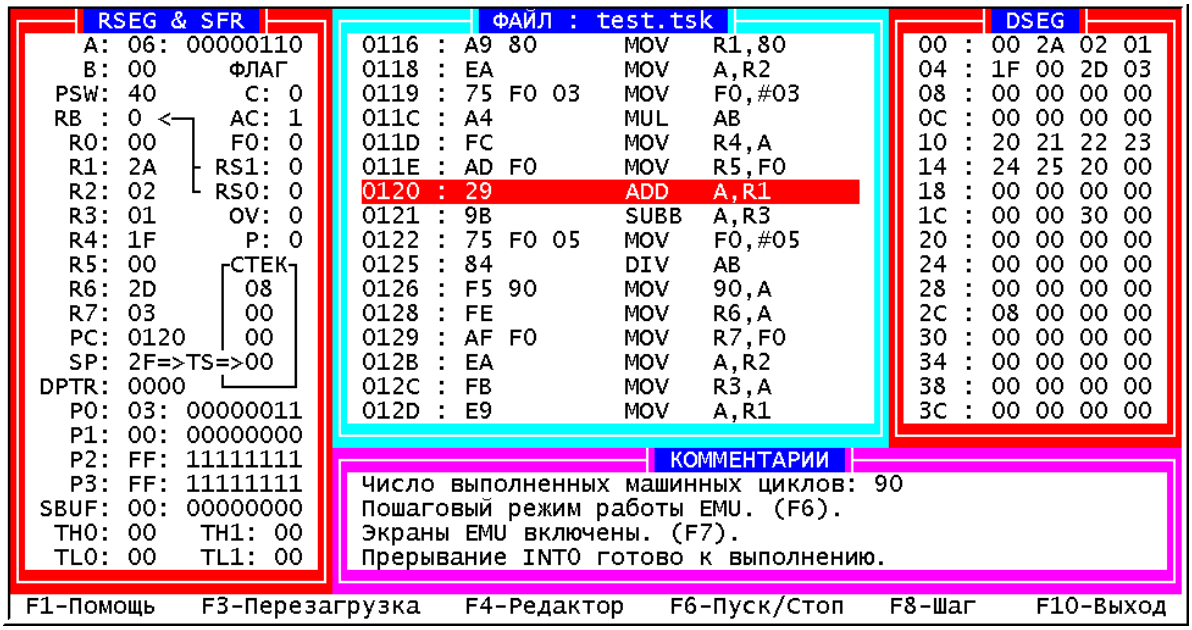


Рис. П3.9

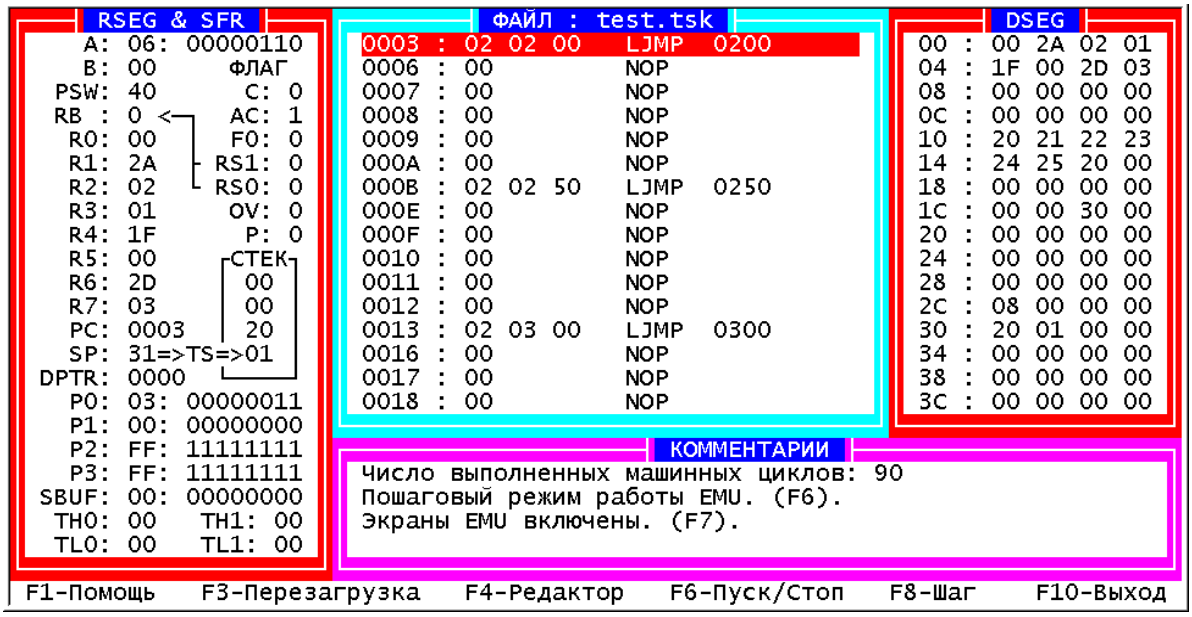


Рис. П3.10

Следующий шаг (F8) – начало обработки прерывания: в стек записывается адрес возврата (содержимое PC = 0120h), а в программный счетчик – адрес прерывания INT0 (PC = 0003h) (рис. П3.10).

Еще один шаг (F8) приведет к началу выполнения прерывающей программы, расположенной по адресу 0200h. На рис. П3.11 показано состояние перед выходом из прерывающей программы. Кроме адреса возврата, в стек было записано содержимое PSW (40h) и аккумулятора A (06h). В прерывающей программе используется RB = 1.

Для наглядности на рис. П3.10 область стека отображена в окне «**DSEG**». В микропроцессорах семейства MCS-51 при записи двухбайтовых чисел в память по старшему адресу записывается старший байт. Стек заполняется в сторону увеличения адресов.

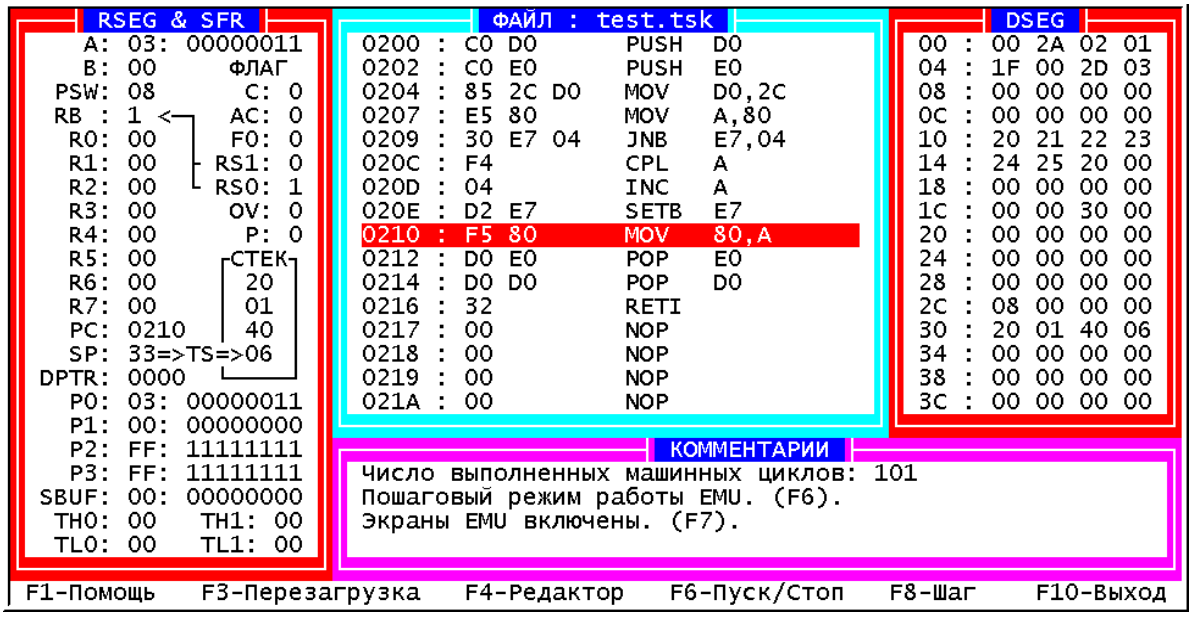


Рис. П3.11

Отладка программы

Для отладки программы необходимо рассчитать несколько контрольных точек по исходным уравнениям (в этом случае можно обнаружить ошибки алгоритма) или по алгоритму. Выходные величины рассчитываются как для положительных, так и отрицательных входных воздействий при определенном начальном состоянии. В качестве промежуточных контрольных точек удобно брать выходные значения подпрограмм, модулей, макросов.

Перед запуском отлаживаемой программы необходимо установить начальные значения регистров и ячеек памяти в соответствии с начальным состоянием, принятым при расчете контрольных точек.

Эмулятор надо запустить в пошаговом режиме, сопоставляя содержимое регистров и ячеек памяти в контрольных точках с расчетными значениями. В случае появления отклонений проанализировать причину их появления. Если они вызваны логическими ошибками программы, то их можно исправить в эмуляторе.

Для коррекции программы надо перейти в режим редактирования сегмента программного кода «CSEG» и исправить содержимое ячеек памяти, соответствующих ошибочным командам, как описано выше. Адреса ячеек памяти можно определить в окне «ФАЙЛ : ».

Проверку работы отдельных фрагментов программы и просмотр дизассемблированной программы можно выполнить, запуская эмулятор с произвольного адреса. Для этого нажать клавишу **F5** и ответить на приглашение вводом начального адреса фрагмента. При вводе адреса необходимо проявить осторожность: вводиться должен адрес первого байта команды. Для корректного выполнения программы в этом случае необходимо перед ее запуском ввести соответствующие проверяемому фрагменту начальные условия.

После окончания отладки программы все изменения необходимо внести в исходный текстовый файл с расширением .asm, заново выполнить ассемблирование, компановку и отладку.

Библиографический список

1. Бродин В.Б., Шагурин М.И. Микроконтроллеры. Архитектура, программирование, интерфейс :Справочник М.: ЭКОМ, 1999.

2. Однокристальные микроЭВМ. :Справочник /А.В. Боборыкин, Г.П. Липовецкий, Г.В. Литвинский и др. М.: МИКАП, 1994.

3. Щелкунов Н.Н., Дианов А.П. Микропроцессорные средства и системы. М.: Радио и связь, 1989.

4. Сташин В.В., Урусов А.Б., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах. М.: Энергоатомиздат, 1990.

5. Цифровая и вычислительная техника: Учебн. для вузов. /Э.Ю. Евреинов, Ю.Т. Бутыльский, И.А. Мамзелев и др.; Под ред. Э.Ю. Евреинова. М.: Радио и связь, 1991.

6. Сальников Н.И. Микроконтроллеры 8051 в устройствах управления радиоэлектронных приборов : Учеб. пособие. Рязань: Рязан. гос. радиотехн. акад. 1998.

7. Гребнев В.В. Однокристальные микроЭВМ семейства MCS-51 фирмы Intel. СПб.: ЭФО, 1997.

8. Микропроцессоры: системы проектирования и отладки. /В.А. Мясников, М.Б. Игнатьев, Л.Л. Кочкин и др.; Под ред. В.А. Мясникова, М.Б. Игнатьева. М.: Энергоатомиздат, 1985.

9. Уильямс Г.С. Отладка микропроцессорных систем. Пер. с англ. М.: Мир, 1988.

10. Фергусон Дж., Макари Л., Уилльямз П. Обслуживание микропроцессорных систем. Пер. с англ. М.: Мир, 1989.

Оглавление