МИКРОСХЕМЫ ИНТЕГРАЛЬНЫЕ
1830ВЕ81Т, 1830ВЕ91Т

Руководство пользователя

2006

СОДЕРЖАНИЕ

1 Введение

ИМС 1830ВЕ81Т, 1830ВЕ91Т – 8-разрядные микроконтроллеры, соответствующие по функциональным возможностям и техническим характеристикам, зарубежным изделиям аналогичного класса. Применение встроенной FLASH памяти программ в ИМС 1830ВЕ91Т обеспечивает потребителю возможность многократного перепрограммирования, гибкость отладки и защиты программ без замены изделия. Разработанные изделия совместимы по системе команд с серийно выпускаемыми микро-ЭВМ Н1830ВЕ31, Н1830ВЕ51.

Развитие микроэлектроники и широкое применение её изделий в промышленном производстве, в устройствах и цифровых системах управления самыми разнообразными объектами и процессами является в настоящее время одним из основных направлений научно-технического прогресса.

Основными областями применения изделия являются встроенные цифровые системы управления.

Применение микроконтроллера качественно улучшит такие характеристики цифровой электроники как:

- производительность аппаратуры;
- потребляемая мощность;
- электромагнитное излучение.

Микросхемы 1830ВЕ81Т, 1830ВЕ91Т представляют собой 8-разрядный микроконтроллер с тактовой частотой 24 МГц и включают в своем составе встроенные ОЗУ – 128 × 8, UART, два 16-разрядных таймер-счётчика, 2 Кбайт программной памяти:

- масочного типа для 1830ВЕ81Т;
- FLASH - типа для 1830ВЕ91Т.

2 Назначение и основные технические характеристики микросхем 1830ВЕ81Т, 1830ВЕ91Т

Функциональным аналогом разрабатываемых микросхем является изделие AT89C2051 фирмы Atmel.

Микросхемы предназначены для применения в системах встроенного управления комплексами радиосвязи, для управления робототехническими комплексами, в системах автоматизации технологических процессов, в системах автоматизированного управления электроприводом, оргтехнике, вычислительной технике, телекоммуникационной технике

и т. п. Особенно перспективно применение микроконтроллера в портативной носимой аппаратуре и приборах, имеющих жесткие ограничения по соотношению быстродействие/потребляемая мощность/стоимость.

2.1 Архитектурные характеристики микроконтроллеров:

Разрядность АЛУ, бит	8
Память программ (встроенная), бит - масочного типа для 1830ВЕ81Т, - FLASH - типа для 1830ВЕ91Т	2К х 8
Регистровое ОЗУ (встроенное), бит	128 х 8
Таймеры, бит	2 х 16
Последовательные порты	1
Режимы энергосбережения	2
Напряжение питания, В	5±0,5
Тактовая частота, МГц	24
Длительность командного цикла, нс	500
Диапазон температур окружающей среды, °С	от минус 60 до 85

2.2 Конструктивные характеристики микросхем 1830ВЕ81Т, 1830ВЕ91Т

Микросхемы разработаны в корпусе 4153.20-5.

Условное графическое обозначение ИМС приведено на рисунке 1.

Функциональное назначение выводов приведено в таблице 1.

2.3 Электрические характеристики микросхем.

Электрические характеристики микросхем при приемке и поставке приведены в таблице 2.

Значения предельно-допустимых электрических режимов эксплуатации в диапазоне рабочих температур приведены в таблице 3.

Динамические характеристики ИМС приведены в таблице 4, а соответствующие временные диаграммы – в разделе 4.

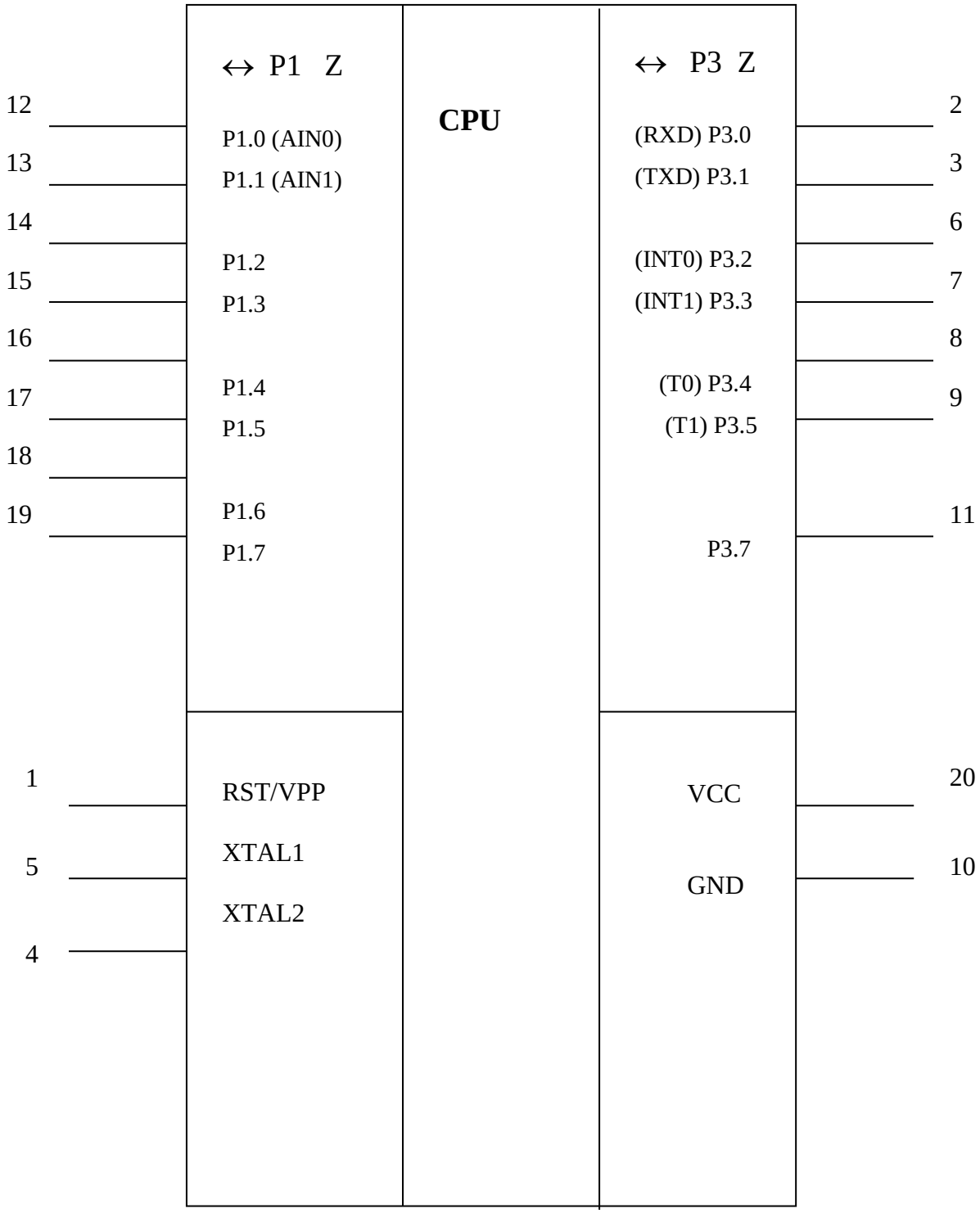


Рисунок 1 - Условное графическое обозначение ИМС 1830ВЕ81Т, 1830ВЕ91Т

Таблица 1 – Функциональное назначение выводов микросхем 8-разрядных микроконтроллеров

Номер вывода	Обозначение вывода	Функциональное назначение вывода
12-19	P1.0-P1.7	8-разрядный двунаправленный порт P1 с дополнительными функциями:
12	P1.0(AIN0)	Компараторный вход 0
13	P1.1(AIN1)	Компараторный вход 1
1	RST/VPP	Сигнал общего сброса / программирования*
2,3 6-9, 11	P3.0-P3.6	7-разрядный двунаправленный порт P3 с дополнительными функциями:
2	(RXD)P3.0	последовательные данные приемника – RXD
3	(TXD)P3.1	последовательные данные передатчика - TXD
6	(INT0)P3.2	вход внешнего прерывания - 0-INT0#
7	(INT1)P3.3	вход внешнего прерывания - 1-INT1#
8	(T0)P3.4	вход таймера / счетчика 0: - T0
9	(T1)P3.5	вход таймера / счетчика 1: - T1
4	XTAL2	выводы для подключения кварцевого резонатора
5	XTAL1	
10	GND	общий вывод
20	VCC	вывод питания от источника напряжения 5 В

* для 1830ВЕ91Т

Таблица 2 - Электрические характеристики микросхем при приемке и поставке

Наименование параметра, единица измерения, режим измерения		Буквенное обозначение параметра	Норма параметра		Температура, ºС
			не менее	не более	
1		2	3	4	5
1 Выходное напряжение низкого уровня по выводам P1.0 - P1.7, P3.0 - P3.7, В I_{OL} = 20 мА		U_{OL}		0,5	– 60 ± 3 25 ± 10 85 ± 3
2 Выходное напряжение высокого уровня по выводам P1.2 - P1.7, P3.0 - P3.7, В I_{OH}= – 80 мкА		U_{OH}	2,4		– 60 ± 3 25 ± 10 85 ± 3
3 Входной ток низкого уровня по выводам P1.2 - P1.7, P3.0 - P3.7, мкА, U_I = 0,45 В		I_{IL}		|– 50|	– 60 ± 3 25 ± 10 85 ± 3
4 Токи утечки на входе по выводам P1.0, P1.1, мкА, 0 < U_1 < U_{CC}		I_{ILH} I_{ILL}	– 10	10	25 ± 10 85 ± 3
5 Напряжение срабатывания компаратора, мВ U_{CC} = 5,0 В, U_{CM} = 2,5 В		U_{OS}		20	– 60 ± 3 25 ± 10 85 ± 3
6 Диапазон входных напряжений компаратора, В U_{CC} = 4,5 В, U_{OS} = 100 мВ		U_{CM}	0	U_{CC}	– 60 ± 3 25 ± 10 85 ± 3
7 Ток потребления в активном режиме, мА U_{CC} = 5,5 В, f_{CI} = 12 МГц		I_{OCC1}		15	– 60 ± 3 25 ± 10 85 ± 3
8 Ток потребления в режиме пониженного потребления, мА U_{CC} = 5,5 В, f_{CI} = 12 МГц		I_{OCC2}		5	– 60 ± 3 25 ± 10 85 ± 3
9 Ток потребления в режиме хранения, мкА U_{CC} = 5,5 В		I_{CCS}		100	– 60 ± 3 25 ± 10 85 ± 3
10 Функциональный контроль U_{CC} = (4,5…5,5) В, f_{CI} = 24 МГц		ФК			– 60 ± 3 25 ± 10 85 ± 3
11 Время переключения на выходе P3.0, нс	время нарастания	$t_{rP3.0}$		20	– 60 ± 3 25 ± 10 85 ± 3
	время спада	$t_{fP3.0}$			

Примечания

1 Нормы параметров приведены при питающем напряжении U_{CC}= (5±0,5) В.

2 Функциональный контроль (ФК) осуществляется в соответствии с системой команд микроконтроллера. Система команд приводится в КФДЛ.431281.025ТО на ИМС.

Таблица 3 – Значения предельно-допустимых электрических режимов эксплуатации в диапазоне рабочих температур

Наименование параметра режима, единица измерения	Буквенное обозначение	Предельно-допустимый режим		Предельный режим	
		не менее	не более	не менее	не более
1	2	3	4	5	6
1 Напряжение источника питания, В	U_{CC}	4,5	5,5		7,5
2 Входное напряжение низкого уровня по выводам P1, P3, RST/VPP, XTAL1, XTAL2, В	U_{IL1}	– 0,5	$0,2U_{CC}$–0,1	–0,7	
3 Входное напряжение высокого уровня по выводам P1, P3, В	U_{IH1}	$0,2U_{CC}$+0,9	U_{CC}+0,5		7,0
4 Входное напряжение высокого уровня по выводам, RST/VPP, XTAL1, XTAL2, В	U_{IH2}	$0,7U_{CC}$	U_{CC}+0,5		7,0
5 Емкость нагрузки по выводам P1.0 – P1.7, P3.0 –P3.7, пФ	C_{L1}		80	-	100
6 Напряжение программирования, В	U_{pp}	11,5	12,5		13
7 Тактовая частота, МГц	f_{CI}	0,2	24		
8 Суммарный ток I_{OL} по выводам P1, P3, мА	ΣI_{OL}		80		
Примечания 1 Длительность фронтов входных сигналов на уровнях $0,1U_{IH}$ и $0,9U_{IH}$ для входа XTAL1 ≤ 10 нс, для остальных входов ≤ 20 нс.					

3 Общая характеристика ИМС

Микроконтроллеры 1830ВЕ81Т, 1830ВЕ91Т имеют следующие основные характеристики:

- 8-разрядный процессор, оптимизированный для приложений управления;
- обширные возможности побитовой обработки;
- встроенная масочная или FLASH память программ;
- двунаправленные и индивидуально адресуемые линии ввода-вывода;
- два 16-разрядных таймера/счётчика;
- полнодуплексный UART;

- структура прерываний (5 источников прерываний);
- экономичные режимы: IDLE и POWER DOWN.

Особенности архитектуры микроконтроллеров.

Отличительной особенностью микроконтроллера 1830ВЕ91Т является применение FLASH-памяти программ. Эта особенность позволяет изменять программный код микроконтроллера, что существенно сокращает цикл разработки. Микроконтроллеры обеспечивают возможность использования наработанных программ и прямой замены. FLASH-память программ делает также возможным изменение программного кода встроенных микроконтроллеров непосредственно у заказчика.

Для обеспечения экономии потребления энергии, микроконтроллеры имеют два программно управляемых режима работы с пониженной мощностью. В режиме IDLE процессор выключен, в то время как оперативная память и встроенные периферийные устройства продолжают функционировать. В этом режиме потребление тока уменьшается приблизительно на 15 процентов от потребления полностью активного устройства. В режиме POWER DOWN все устройства микроконтроллеров выключены, однако данные в оперативной памяти продолжают сохраняться. Кроме того, микроконтроллеры разработаны с применением статической логики, которая не требует непрерывной синхронизации. Поэтому частота тактового генератора может быть уменьшена или же он может быть остановлен в ожидании события, требующего обработки. Это также способствует снижению потребления по питанию.

После выполнения процедуры сброса выполнение программы начинается с адреса 0000Н. С адреса 0003Н располагаются блоки обработки прерываний, занимающие по 8 байтов. Если процедура обработки прерывания занимает не более 8 байтов, она может располагаться в этом блоке. Процедуры обработки прерываний большего размера размещаются в других областях памяти программы, а управление передаётся им из блоков обработки прерываний командами безусловного перехода. Если прерывания в программе не используются, адреса, зарезервированные под блоки обработки прерываний, могут быть заняты кодом программы.

В 128 байтах внутренней памяти первые 32 байта заняты четырьмя банками по 8 регистров, адресуемых командами программы как R0...R7. Выбор банка, в котором адресуются регистры, обеспечивается соответствующей установкой двух битов в слове состояния программы (PSW). Такая архитектура позволяет более эффективно использовать кодовое пространство, так как команды обращения к регистрам короче команд прямой ад-

ресации памяти. Следующие 16 байт после банков регистров образуют блок битовой адресации. Система команд микроконтроллеров включает широкий выбор команд битовой адресации, которые могут непосредственно адресовать 128 битов в этой области. Адресуемые биты имеют адреса 00H...7FH.

Все байты ОЗУ могут адресоваться прямым и косвенным методом адресации. Пространство SFR (регистров специальных функций) включает порты ввода-вывода, регистры таймеров, регистры управления периферийными устройствами и т. д. Эта область может адресоваться только прямой адресацией. Структура пространства идентична структуре аналогичного пространства микроконтроллеров семейства MCS-51™. Некоторые адреса в пространстве SFR могут адресоваться и побайтно и поразрядно.

Система команд микроконтроллеров оптимизирована для 8-разрядных приложений управления, обеспечивая ряд быстрых способов адресации для доступа к внутренней оперативной памяти. Система команд обеспечивает обширную поддержку для однобитовых переменных как отдельного типа данных, позволяя выполнять прямое разрядное манипулирование в управлении и логических системах, которые требуют Булевой обработки.

Слово состояния программы (PSW) содержит биты состояния, которые отражают текущее состояние процессора, и размещается в пространстве SFR. Слово состояния программы содержит бит переноса CY, бит дополнительного переноса (для операций с BCD - двоично-десятичным кодированием) AC, биты выбора банка RS0 и RS1, флаг переполнения OV, бит контроля по четности P и двух определяемых пользователем флажков.

Микроконтроллеры имеют команды прямой и косвенной адресации, регистровые команды и специальные команды для некоторых регистров. В последнем случае код команды непосредственно указывает на регистр, с которым будет производиться операция. При прямой адресации операнд определён 8-разрядным полем адреса в команде. Этим методом может быть адресована оперативная память и пространство SFR. При косвенной адресации в команде указан регистр, который содержит адрес операнда. В качестве регистра адреса для 8-разрядных адресов может быть или указатель вершины стека или регистры R0 или R1 выбранного банка. К банкам регистров, которые содержат регистры R0...R7, можно обращаться командами, чьи коды операции включают 3-разрядную спецификацию регистра. Команды, которые обращаются к регистрам этим способом, обеспечивают эффективное использование кода программы, так как при этом в команде отсутствует байт адреса. Банк, в котором текущей командой адресуется регистр, выбирается соответствующей установкой двух битов в слове состояния программы PSW.

Машинный цикл микроконтроллеров состоит из 6 состояний S1...S6, каждое из которых занимает два такта тактового генератора. Таким образом, длительность машинного цикла составляет 12 тактов тактового генератора, или при тактовой частоте 24 МГц –

0,5 мкс. Команда программы может быть выполнена в течение одного или нескольких машинных циклов.

Микроконтроллеры имеют 5 источников прерываний. Прерывание по каждому из источников может быть индивидуально разрешено или запрещено путём установки или сброса соответствующих битов в регистре разрешения прерываний IE, расположенном в пространстве SFR. В процессе выполнения программы состояние флагов прерываний считываются в 5-м состоянии машинного цикла и опрашиваются в следующем цикле. Для каждого из источников прерываний может быть запрограммирован один из двух уровней приоритета путём установки или сброса соответствующего бита в регистре приоритетов прерываний IP, расположенном в пространстве SFR. Низкоприоритетное прерывание может быть прервано высокоприоритетным прерыванием, но не другим низкоприоритетным прерыванием. Выполнение процедуры высокоприоритетного прерывания не может быть прервано никаким прерыванием. Если одновременно поступило два запроса на прерывание с разными уровнями приоритета, то сначала выполняется процедура высокоприоритетного прерывания. При поступлении запросов на прерывание с одинаковым уровнем приоритета порядок выполнения процедур обработки прерывания определяется внутренней последовательностью опроса. В процессе обработки прерывания аппаратно сгенерированная процедура LCALL помещает содержимое счётчика команд PC в стек и загружает начальным адресом соответствующего блока обработки прерывания. Кроме счётчика команд автоматически в стеке не сохраняются никакие другие регистры. За сохранение других необходимых регистров отвечает программист. В ряде случаев это позволяет сократить время обработки прерывания. В результате, много функций обработки прерываний, которые являются типичными в приложениях управления - переключение вывода порта, перезагрузка таймера, или чтение буфера последовательного порта, могут быть завершены скорее, чем это было бы возможно при другой архитектуре. Многие приложения требуют более двух уровней приоритетности прерываний, обеспечиваемых аппаратными средствами микроконтроллеров. В таком случае возможно применение простого программного кода, с помощью которого эмулируется третий уровень приоритетности прерывания.

Последовательный порт микроконтроллеров - полнодуплексный, с буфером приёмника. Доступ к регистрам приёма и передачи осуществляется через регистр SBUF в пространстве SFR. При выполнении записи в этот регистр загружается регистр передачи,

чтение обеспечивает доступ регистру приёма. Для обеспечения стандартных скоростей обмена 1200...19200 бод необходимо тактировать микроконтроллер с частотой

11,059 МГц, при этом для получения необходимой скорости обмена используется один из таймеров.

Микроконтроллеры имеют два режима работы с пониженным потреблением по питанию: Idle и Power Down, переход в которые обеспечивается установкой соответствующих битов в регистре PCON пространства SFR. В режиме Idle (IDL=1) тактовый генератор продолжает работать, и обеспечивается работа периферийных устройств: таймеров, блока обработки прерываний и последовательного порта, при этом процессор микроконтроллера останавливается в ожидании поступления прерывания. В режиме Power Down (PD=1) останавливается тактовый генератор микроконтроллера, однако, содержимое встроенной памяти и регистров пространства SFR сохраняется. Выход из состояния Power Down возможен только при выполнении аппаратного сброса. При сбросе инициализируются все регистры пространства SFR, однако, содержимое внутренней памяти данных не изменяется. Аппаратный сброс запускает также тактовый генератор микроконтроллера. На время пребывания микроконтроллера в режиме Power Down напряжение питания может быть снижено до 2 В. Однако напряжение питания не должно быть понижено до того, как микроконтроллер перешёл в режим Power Down и должно быть восстановлено перед выполнением процедуры сброса. Сигнал сброса должен быть достаточно длительным для того, чтобы стабилизировалась работа тактового генератора (обычно не менее 10 мс).

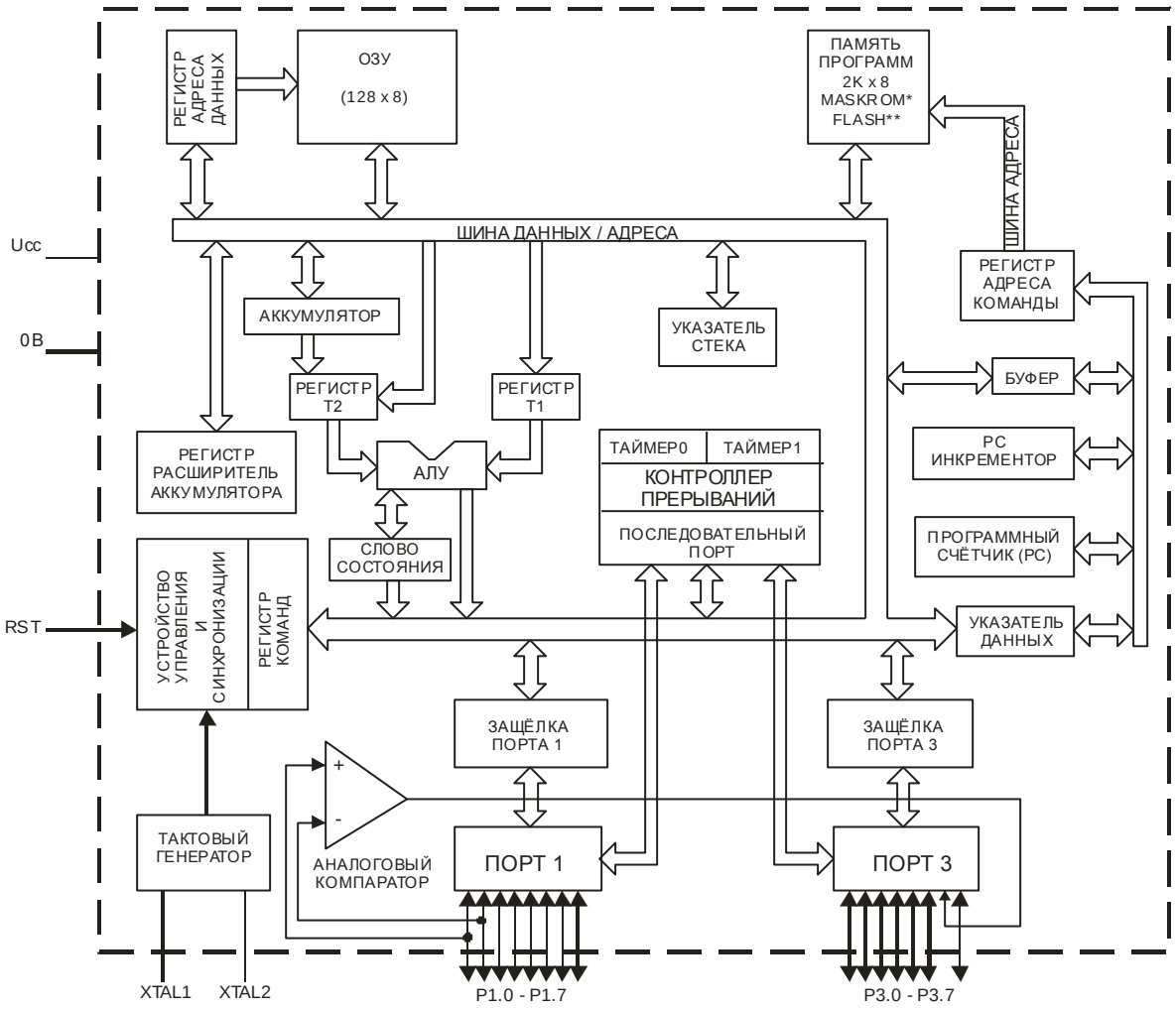
4. Описание устройства

4.1 Структура и организация микроконтроллера

Микроконтроллеры имеют следующую структуру:

- 8-разрядное ЦПУ;
- встроенная масочная (1830ВЕ81Т) или FLASH (1830ВЕ91Т) память программ объемом 2 К байт;
- двунаправленные и индивидуально адресуемые линии ввода-вывода;
- два 16-разрядных таймера/счётчика;
- полнодуплексный UART;
- структура прерываний;
- встроенный тактовый генератор;
- экономичные режимы: IDLE и POWER DOWN.

На рисунке 2 приведена функциональная схема ИМС 1830ВЕ81Т, 1830ВЕ91Т.



* типа MASKROM для 1830ВЕ81Т;

** типа FLASH для 1830ВЕ91Т.

Рисунок 2 – Схема структурная электрическая микроконтроллеров 1830ВЕ81Т, 1830ВЕ91Т.

4.2 Организация ОЗУ, ПЗУ и регистров ИМС

Память программ и данных являются самостоятельными и независимыми друг от друга устройствами, адресуемыми различными командами и управляющими сигналами. Объем памяти программ, расположенной на кристалле ИМС, равен 2 Кбайтам. ИМС обращается к программной памяти при чтении кода операции и операндов (используя счетчик команд PC), а также при выполнении команд переноса байта из памяти программ в аккумулятор. При выполнении команд переноса данных адресация ячейки памяти программ, из которой будут прочитаны данные, осуществляется с использованием счетчика PC.

Память данных, расположенная на кристалле, имеет объем 128 байт (регистровая память). Первые 32 байта организованы в четыре банка регистров общего назначения, обозначаемых соответственно банк 0...3. Каждый банк состоит из 8 регистров R0..R7. В любой момент времени программе доступен только один банк регистров, номер которого содержится в 3-м и 4-м битах слова состояния программы PSW (см. ниже).

Оставшееся адресное пространство может конфигурироваться разработчиком по своему усмотрению - в нем располагаются стек, системные и пользовательские области данных. Обращение к ячейкам памяти данных возможно двумя способами. Первый способ - прямая адресация ячейки памяти. В этом случае адрес ячейки является операндом соответствующей команды. Второй способ - косвенная адресация при помощи регистров R0 или R1 - перед выполнением соответствующей команды в один из этих регистров должен быть занесен адрес ячейки, к которой необходимо обратиться.

К адресному пространству памяти данных примыкает адресное пространство регистров специальных функций (SFR). Адреса, по которым расположены эти регистры, приведены ниже, в таблице 4. Отметим, что регистры занимают только часть 128-байтового адресного пространства.

Таблица 4 - Блок регистров специальных функций

Символ	Адрес	Наименование
1	2	3
* ACC	0E0H	Аккумулятор
* B	0F0H	Регистр-расширитель аккумулятора
* PSW	0D0H	Слово состояния программы
* P0	080H	Порт 0 (SRF P0)
* P1	090H	Порт 1 (SFR P1)
* P2	0A0H	Порт 2 (SFR P2)
* P3	0B0H	Порт 3 (SFR P3)
SP	081H	Регистр-указатель стека
DPH	083H	Старший байт регистра-указателя данных DPTR

1	2	3
DPL	082H	Младший байт регистра-указателя данных DPTR
TH0	08CH	Старший байт таймера 0
TL0	08AH	Младший байт таймера 0
TH1	08DH	Старший байт таймера 1
TL1	08BH	Младший байт таймера 1
TMOD	089H	Регистр режимов таймеров-счетчиков
* TCON	088H	Регистр управления-статуса таймеров
* IP	0B8H	Регистр приоритетов
* IE	0A8H	Регистр маски прерываний
PCON	087H	Регистр управления мощностью
* SCON	098H	Регистр управления приемопередатчиком
SBUF	099H	Буфер приемопередатчика
Примечание - регистры, имена которых отмечены знаком (*), допускают адресацию своих отдельных бит при выполнении команд из группы команд операций с битами.		

Аккумулятор является источником операнда и местом хранения результата при выполнении многих команд. С использованием аккумулятора могут быть выполнены операции сдвига, проверки на нуль и ряд других. Система команд содержит также команды пересылок, логических операций и переходов, не использующих аккумулятор.

При выполнении ряда команд в арифметико-логическом устройстве (АЛУ) формируются признаки операций - флаги, которые фиксируются в регистре PSW. Перечень флагов, их символические имена и условия их формирования приведены в таблице 5.

Таблица 5 - Формат слова состояния программы PSW

Символ	Позиция	Имя и назначение			
P	PSW.0	Флаг паритета. Устанавливается и сбрасывается аппаратно в каждом цикле команды и фиксирует нечетное/четное число единичных бит в аккумуляторе.			
-	PSW.1	Не используется.			
OV	PSW.2	Флаг переполнения. Устанавливается и сбрасывается аппаратно при выполнении арифметических операций.			
RS0 RS1	PSW.3 PSW.4	Биты выбора используемого банка регистров. Могут быть изменены программным путем.			
		RS0	RS1	Банк	Границы адресов ОЗУ
		0	0	0	00H - 07H
		1	0	1	08H - 0FH
		0	1	2	10H - 17H
		1	1	3	18H - 1FH
F0	PSW.5	Флаг пользователя. Может быть установлен, сброшен или проверен программой пользователя.			
AC	PSW.6	Флаг вспомогательного переноса. Устанавливается и сбрасывается только аппаратными средствами при выполнении команд сложения и вычитания и сигнализирует о переносе или заеме в бите 3 аккумулятора.			
C	PSW.7	Флаг переноса. Устанавливается и сбрасывается как аппаратно, так и программным путем			

Регистр-указатель стека SP в ИМС - 8-битный. Он может адресовать любую область внутренней памяти данных. Стек “растет вверх”, т.е. перед выполнением команды PUSH или CALL содержимое SP инкрементируется, после чего производится запись информации в стек. Соответственно при извлечении информации из стека регистр SP декрементируется.

В процессе инициализации ИМС после сигнала сброса или при включении питающего напряжения в SP заносится код 07H. Это означает, что первый элемент данных будет располагаться в ячейке памяти с адресом 08H.

Две регистровые пары с именами TH0,TL0 и TH1,TL1 представляют из себя регистры, обеспечивающие независимое функционирование двух программно-управляемых 16-битных таймеров-счетчиков. Режимы таймеров-счетчиков задаются с использованием регистра TMOD, а управление ими осуществляется при помощи регистра TCON. Для управления режимами энергопотребления ИМС используется регистр PCON. Регистры IP и IE управляют работой системы прерываний ИМС, а регистры SBUF и SCON - работой приемо-передатчика последовательного порта. Описание таймеров-счетчиков, системы прерываний и приемо-передатчика последовательного порта будет приведено ниже.

4.3 Организация портов ввода – вывода

Оба порта микроконтроллера (P1, P3) - двунаправленные. Каждый из них содержит регистр-защелку (SFR P1, P3), выходную цепь и входной буфер. Выводы порта P1(2..7) имеют внутреннюю нагрузку. P1.0 и P1.1 требуют внешней нагрузки, т.к. альтернативно используются как положительный так и отрицательный входы компаратора соответственно. Порт P1 также принимает коды данных во время программирования и верификации FLASH – памяти.

Порт P3 содержит 7 двунаправленных выводов с внутренней нагрузкой. P3.6 используется как выход с аналогового компаратора и не доступен как общий целевой вывод. Выводы порта P3 помимо обычного ввода информации и вывода ее из SFR могут выполнять альтернативные функции. Описание этих функций приведено в таблице 6.

Таблица 6

Вывод порта	Альтернативная функция
P3.0	RXD - вход последовательного порта
P3.1	TXD - выход последовательного порта
P3.2	$\overline{\text{INT0}}$ - внешнее прерывание 0
P3.3	$\overline{\text{INT1}}$ - внешнее прерывание 1
P3.4	T0 - вход таймера-счетчика 0
P3.5	T1 - вход таймера-счетчика 1

Функциональные схемы регистров-защелок и буферов ввода-вывода каждого из 2-х портов построены следующим образом:

Каждый из битов регистра-защелки SFR является D-триггером, информация в который заносится с внутренней шины данных по сигналу "запись в SFR Pi" (i=0...3) от центрального процессорного элемента (CPU). С неинвертированного выхода D-триггера информация может быть выведена на внутреннюю шину по сигналу "чтение SFR Pi" от CPU, а с вывода микросхемы ("из внешнего мира") - по сигналу "чтение выводов Pi". Одни команды активируют сигнал "чтение SFR Pi", другие - "чтение выводов Pi".

Выходы триггеров SFR портов P1.2 – P1.7, P3 выполнены на полевых транзисторах, имеющих внутреннюю нагрузку, в то время как выходы триггеров SFR P1.0, P1.1 выполнены на транзисторах с открытым стоком. Каждая линия любого из портов может независимо использоваться как для ввода информации, так и для вывода. Для использования любой линии портов P1, P3 в качестве входа необходимо в соответствующий разряд SFR занести 1. При этом выходной полевой транзистор отключается. Внутренний нагрузочный резистор как бы "подтягивает" потенциал вывода к напряжению питания, в то время как внешняя нагрузка может сделать его нулевым.

Поскольку выводы P1, P3 имеют внутреннюю нагрузку, то при переводе их в режим ввода они являются источниками тока для вывода микросхемы или транзистора, нагруженного на этот вывод. Поэтому порты P1, P3 получили название "квазидвунаправленные".

Во время выполнения команд, изменяющих состояние SFR портов, новые значения защелкиваются в триггеры SFR в самом последнем цикле выполнения команды. Но на выводах ИМС данные появляются только в начале выполнения следующей команды.

Если это изменение представляет собой перепад из 0 в 1 линий портов P1, P3, то в начальный момент выполнения команды включается дополнительный нагрузочный транзистор в нагрузке транзистора выходного каскада порта. Дополнительная нагрузка в состоянии обеспечить ток переключения в 100 раз больший, чем номинальная, что резко повышает скорость переключения.

Если на выводе ИМС присутствовала «1», а внешняя нагрузка переключилась в «0», то дополнительный нагрузочный транзистор может закрыться, и порт перейдет в высокоимпедансное состояние. Но при снятии причины, приведшей к этому переключению, основной p-канальный полевой транзистор восстановит исходный потенциал.

Обращение к портам ввода-вывода возможно с использованием команд, оперирующих с байтом, отдельным битом и произвольной комбинацией бит. При этом в тех слу-

чаях, когда порт является одновременно операндом и местом назначения результата, автоматически реализуется специальный режим "чтение – модификация - запись". Этот режим обращения предполагает ввод сигналов не с внешних выводов порта, а из его регистра-защелки SFR, что позволяет исключить неправильное считывание ранее выведенной информации.

Команды типа "чтение – модификация - запись" реализуют считывание из регистра SFR, а не с внешнего вывода порта, что обеспечивает получение правильной информации.

4.4 Устройство управления и синхронизации

К выводам X1 и X2 ИМС подключается кварцевый резонатор.

Устройство управления ИМС формирует машинный цикл фиксированной длительности, равный 12 периодам резонатора, или шести состояниям управляющего устройства (S1...S6). Каждое состояние содержит две фазы сигналов резонатора (P1 и P2). В фазе P1, как правило, выполняется операция в АЛУ, а в фазе P2 выполняется межрегистровая передача. Весь машинный цикл состоит из 12 фаз, начиная от S1P1 и кончая S6P2 . Так осуществляется работа управляющего устройства при выполнении команд различной степени сложности, от однобайтовых, выполняющихся за 1 машинный цикл, до двухбайтовых, работающих с внешней памятью. Количество машинных циклов и тактов, необходимое ИМС для выполнения каждой команды, будет дано при описании системы команд. ИМС оперирует с 13 различными типами команд, имеющими длину от 1 до 3 байт и выполняющимися за 1...2 машинных цикла (за исключением двух однобайтовых команд умножения и деления, выполняющихся за 4 цикла). Таким образом, при работе на частоте 24 МГц подавляющее большинство команд выполняется ИМС за 0,5...1 мкс.

4.5 Таймеры/счетчики

Два программируемых 16-битных таймера/счетчика (T/C0 и T/C1) могут быть использованы как в качестве таймеров, так в качестве счетчиков внешних событий. При работе в качестве таймера содержимое соответствующего T/C инкрементируется в каждом машинном цикле, т.е. через каждые 12 периодов резонатора. При работе в качестве счетчика содержимое T/C инкрементируется под воздействием перехода из 1 в 0 внешнего входного сигнала, подаваемого на соответствующий (T0,T1) вывод ИМС. Опрос значения внешнего входного сигнала выполняется в момент времени S5P2 каждого машинного цикла. Содержимое счетчика будет увеличено на 1 в том случае, если в предыдущем цикле был считан входной сигнал высокого уровня (1), а в следующем - сигнал низкого уров-

ня (0). Новое (инкрементированное) значение счетчика будет сформировано в момент S3P1 в цикле, следующем за тем, в котором был обнаружен переход сигнала из 1 в 0. Так как на распознавание перехода требуется два машинных цикла, то максимальная частота подсчета входных сигналов равна 1/24 частоты резонатора. На длительность периода входных сигналов ограничений сверху нет.

Для гарантированного прочтения входного считываемого сигнала он должен удерживаться как минимум в течение одного машинного цикла.

Для управления режимами работы Т/С и для организации взаимодействия таймеров с системой прерывания используются два регистра специальных функций (TMOD и TCON), описание которых приводится в таблицах 7 и 8 соответственно. Как следует из описания управляющих бит TMOD, для обоих Т/С режимы работы 0,1 и 2 одинаковы. Режимы 3 для Т/С0 и Т/С1 различны.

Таблица 7 - Регистр режима работы таймер/счетчика TMOD

Символ	Позиция	Имя и назначение
GATE	TMOD.7 для Т/С0	Управление блокировкой. Если бит установлен, то для Т/С1 таймер/счетчик "х" разрешен до тех пор, пока и $\overline{\text{TMOD3}}$ на входе "INTx" высокий уровень и бит управления "TRx" установлен. Если бит сброшен, то Т/С разрешается, как только бит управления "TRx" устанавливается
$\overline{\text{C/T}}$	TMOD.6 для Т/С1 и TMOD.2 для Т/С0	Бит выбора режима таймера или счетчика событий. Если бит сброшен, то работает таймер от внутреннего источника сигналов синхронизации Если бит установлен, то работает счетчик от внешних сигналов на входе "Тх"
M1	TMOD.5 для Т/С1и TMOD.1 для Т/С0	Режим работы (см. примечание)
M0	TMOD.4 для Т/С1 и TMOD 0для Т/С0	

Примечание.

M1	M0	Режим работы
0	0	Таймер BE48. "TLx" работает как 5-битный предделитель
0	1	16-битный таймер/счетчик. "THx" и "TLx" включены последовательно
1	0	8-битный автоперезагружаемый таймер/счетчик. "THx" хранит значение, которое должно быть перезагружено в "TLx" каждый раз по переполнению
1	1	Таймер/счетчик 1 останавливается. Таймер/счетчик 0 : TL0 работает как 8-битный таймер/счетчик, и его режим определяется управляющими битами таймера 0. TH0 работает только как 8-битный таймер, и его режим определяется управляющими битами таймера 1

Таблица 8. - Регистр управления/статуса таймера TCON

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

Рассмотрим работу Т/С во всех четырех режимах.

Режим 0. Перевод любого Т/С в режим 0 делает его похожим на таймер, на вход которого подключен 5-битный предделитель частоты на 32. В этом режиме таймерный регистр имеет разрядность 13 бит. При переходе из состояния "все единицы" в состояние "все нули" устанавливается флаг прерывания от таймера TF1. Входной синхросигнал таймера 1 разрешен (поступает на вход Т/С1), когда управляющий бит TR1 установлен в 1 и либо управляющий бит GATE (блокировка) равен 0, либо на внешний вывод запроса прерывания $\overline{INTI}$ поступает уровень 1.

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

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

Режим 2. В режиме 2 работа организована таким образом, что переполнение (переход из состояния "все единицы" в состояние "все нули") 8-битного счетчика TL1 приводит не только к установке флага TF1, но и автоматически перезагружает в TL1 содержимое старшего байта (TH1) таймерного регистра, которое предварительно было задано про-

граммным путем. Перегрузка оставляет содержимое TH1 неизменным. В режиме 2 T/C0 и T/C1 также работают совершенно одинаково.

Режим 3. В режиме 3 T/C0 и T/C1 работают по-разному. T/C1 сохраняет неизменным свое текущее содержимое. Иными словами, эффект такой же, как и при сбросе управляющего бита TR1 в нуль.

В режиме 3 TL0 и TH0 функционируют как два независимых 8-битных счетчика. Работу TL0 определяют управляющие биты T/C0 (C/T, GATE, TR0), входной сигнал $\overline{\text{INT0}}$ и флаг переполнения TF0. Работу TH0, который может выполнять только функции таймера (подсчет машинных циклов ИМС), определяет управляющий бит TR1. При этом TH0 использует флаг переполнения TF1.

Режим 3 используется в тех случаях применения, когда требуется наличие дополнительного 8-битного таймера или счетчика событий. В том случае, если T/C0 используется в режиме 3, T/C1 может быть или выключен, или переведен в режим 0,1 или 2, или может быть использован последовательным портом в качестве генератора частоты передачи.

4.6 Система прерываний

Внешние прерывания $\overline{\text{INT0}}$ и $\overline{\text{INT1}}$ могут быть вызваны либо уровнем, либо переходом сигнала из 1 в 0 на входах ИМС в зависимости от значений управляющих бит IT0 и IT1 в регистре TCON.

От внешних прерываний устанавливаются флаги IE0 и IE1 в регистре TCON, которые инициируют вызов соответствующей программы обслуживания прерывания. Сброс этих флагов выполняется аппаратно только в том случае, если прерывание было вызвано по переходу (срезу) сигнала. Если же прерывание вызвано уровнем входного сигнала, то сбросом флага IE должна управлять соответствующая подпрограмма обслуживания прерывания путем воздействия на источник прерывания с целью снятия им запроса.

Флаги запросов прерывания от таймеров TF0 и TF1 сбрасываются автоматически при передаче управления подпрограмме обслуживания. Флаги запросов прерывания RI и TI устанавливаются блоком управления приемопередатчика аппаратно, но сбрасываться должны программным путем.

Прерывания могут быть вызваны или отменены программой, так как все перечисленные флаги программно - доступны и могут быть установлены/ сброшены программой с тем же результатом, как если бы они были установлены/сброшены аппаратурными средствами.

В блоке регистров специальных функций есть два регистра, предназначенных для управления режимом прерываний и уровнями приоритета. Форматы этих регистров, имеющих символические имена IF и IP, описаны в таблицах 9 и 10 соответственно. Возможность программной установки/сброса любого управляющего бита в этих двух регистрах делает систему прерываний ИМС исключительно гибкой.

Таблица 9 - Регистр масок прерывания (IE)

Символ	Позиция	Имя и назначение
EA	IE.7	Снятие блокировки прерывания. Сбрасывается программно для запрета всех прерываний независимо от состояний IE.4 - IE.0
-	IE.6	Не используется
-	IE.5	Не используется
ES	IE.4	Бит разрешения прерывания от приемопередатчика. Установка/сброс программой для разрешения/запрета прерываний от флагов TI или RI
ET1	IE.3	Бит разрешения прерывания от таймера 1. Установка/сброс программой для разрешения/запрета прерываний от таймера 1
EX1	IE.2	Бит разрешения внешнего прерывания 1. Установка/сброс программой для разрешения/запрета прерывания 1
ET0	IE.1	Бит разрешения прерывания от таймера 0. Установка/сброс программой для разрешения/запрета прерываний от таймера 0
EX0	IE.0	Бит разрешения внешнего прерывания 0. Установка/сброс программой для разрешения/запрета прерывания 0

Таблица 10 - Регистр приоритетов прерываний IP

Символ	Позиция	Имя и назначение
-	IP.7-IP.5	Не используются
PS	IP.4	Бит приоритета приемопередатчика. Установка/сброс программой для присваивания прерыванию от приемопередатчика высшего/низшего приоритета
PT1	IP.3	Бит приоритета таймера 1. Установка/сброс программой для присваивания прерыванию от таймера 1 высшего/низшего приоритета
PX1	IP.2	Бит приоритета внешнего прерывания 1. Установка / сброс программой для присваивания высшего/низшего приоритета внешнему прерыванию $\overline{\text{INT1}}$
PT0	IP.1	Бит приоритета таймера 0. Установка/сброс программой для присваивания прерыванию от таймера 0 высшего/низшего приоритета
PX0	IP.0	Бит приоритета внешнего прерывания 0. Установка/сброс программой для присваивания высшего/низшего приоритета внешнему прерыванию $\overline{\text{INT0}}$

Флаги прерываний опрашиваются в момент S5P2 каждого машинного цикла. Ранжирование прерываний по уровню приоритета выполняется в течение следующего машинного цикла. Система прерываний сформирует аппаратный вызов (LCALL) соответст-

вующей подпрограммы обслуживания, если она не заблокирована одним из следующих условий:

1) в данный момент обслуживается запрос прерывания равного или высокого уровня приоритета;

2) текущий машинный цикл - не последний в цикле выполняемой команды;

3) выполняется команда RETI или любая команда, связанная с обращением к регистрам IE или IP.

Отметим, что если флаг прерывания был установлен, но по одному из перечисленных выше условий не получил обслуживания и к моменту окончания блокировки уже был сброшен, то запрос прерывания теряется и нигде не запоминается.

По аппаратно - сформированному коду LCALL система прерывания помещает в стек только содержимое счетчика команд (PC) и загружает в счетчик команд адрес вектора соответствующей подпрограммы обслуживания. По адресу вектора должна быть расположена команда безусловной передачи управления (JMP) к начальному адресу подпрограммы обслуживания прерывания. Подпрограмма обслуживания в случае необходимости должна начинаться командами записи в стек (PUSH) слова состояния программы (PSW), аккумулятора, расширителя, указателя данных и т.д. и должна заканчиваться командами восстановления из стека (POP). Подпрограммы обслуживания прерывания должны завершаться командой RETI, по которой в счетчик команд перезагружается из стека сохраненный адрес возврата в основную программу. Команда RET также возвращает управление прерванной основной программе, но при этом не снимает блокировку прерываний, что приводит к необходимости иметь программный механизм анализа окончания процедуры обслуживания данного прерывания.

4.7 Последовательный интерфейс

Через универсальный асинхронный приемопередатчик осуществляется прием и передача информации, представленной последовательным кодом (младшими битами вперед), в полном дуплексном режиме обмена. В состав приемопередатчика, называемого часто последовательным портом, входят принимающий и передающий сдвигающие регистры, а также специальный буферный регистр (SBUF) приемопередатчика. Запись байта в буфер приводит к автоматической переписи байта в сдвигающий регистр передатчика и инициирует начало передачи байта. Наличие буферного регистра приемника позволяет совмещать операцию чтения ранее принятого байта с приемом очередного байта. Но если

к моменту окончания приема байта предыдущий байт не был считан из SBUF, то он будет потерян.

Последовательный порт ИМС может работать в четырех различных режимах.

Режим 0. В этом режиме информация и передается и принимается через внешний вывод входа приемника (RXD). Принимаются или передаются 8 бит данных. Через внешний вывод выхода передатчика (TXD) выдаются импульсы сдвига, которые сопровождают каждый бит. Частота передачи бита информации равна 1/12 частоты резонатора.

Режим 1. В этом режиме передаются через TXD или принимаются из RXD 10 бит информации: старт-бит (0), 8 бит данных и стоп-бит (1). При приеме информации в бит RB8 регистра управления-статуса приемопередатчика SCON заносится стоп-бит. Скорость приема/передачи - величина переменная и задается таймером.

Режим 2. В этом режиме через TXD передаются или из RXD принимаются 11 бит информации: старт-бит, 8 бит данных, программируемый девятый бит и стоп-бит. При передаче девятый бит данных может принимать значение 0 или 1, или, например, для повышения достоверности передачи путем контроля по четности в него может быть помещено значение признака паритета из слова состояния программы (PSW.0). При приеме девятый бит данных помещается в бит RB8 SCON, а стоп-бит, в отличие от режима 1, теряется. Частота приема/передачи выбирается программой и может быть равна либо 1/32, либо 1/64 частоты резонатора в зависимости от управляющего бита SMOD.

Режим 3. Режим 3 совпадает с режимом 2 во всех деталях, за исключением частоты приема/передачи, которая является величиной переменной и задается таймером.

Во всех случаях передача инициализируется инструкцией, в которой данные перемещаются в SBUF. Прием инициализируется при обнаружении перепада из 1 в 0 на входе приемника. При этом в режиме 0 этот переход должен сопровождаться выполнением условий RI=0 и REN=1, а для остальных режимов - REN=1.

Регистр управления/статуса приемопередатчика SCON

Управлением режимом работы приемопередатчика осуществляется через специальный регистр с символическим именем SCON.

Этот регистр содержит не только управляющие биты, определяющие режим работы последовательного порта, но и девятый бит принимаемых или передаваемых данных (RB8 и TB8) и биты прерывания приемопередатчика (RI и TI).

Функциональное назначение бит регистра управления/статуса приемопередатчика приводится в таблице 11.

Таблица 11 - Регистр управления/статуса SCON

Символ	Позиция	Имя и назначение
SM0 SM1	SCON.7 SCON.6	Биты управления режимом работы приемопередатчика. Устанавливаются / сбрасываются программно (см. примечание)
SM2	SCON.5	Бит управления режимом приемопередатчика. Устанавливается программно для запрета приема сообщения, в котором девятый бит имеет значение 0
REN	SCON.4	Бит разрешения приема. Устанавливается/сбрасывается программно для разрешения/запрета приема последовательных данных
TB8	SCON.3	Передача бита 8. Устанавливается/сбрасывается программно для задания девятого передаваемого бита в режиме 9-битового передатчика
RB8	SCON.2	Прием бита 8.Устанавливается/сбрасывается аппаратно для фиксации девятого принимаемого бита в режиме 9-битового приемника
TI	SCON.1	Флаг прерывания передатчика. Устанавливается аппаратно при окончании передачи байта. Сбрасывается программно после обслуживания прерывания
RI	SCON.0	Флаг прерывания приемника. Устанавливается аппаратно при приеме байта. Сбрасывается программно после обслуживания прерывания

Примечание

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

Прикладная программа путем загрузки в старшие биты регистра SCON 2-битного кода определяет режим работы приемопередатчика. Во всех четырех режимах работы передача инициализируется любой командой, в которой буферный регистр SBUF указан как получатель байта. Как уже отмечалось, прием в режиме 0 осуществляется при условии, что RI=0 и REN=1, в остальных режимах - REN=1.

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

Флаг прерывания передатчика TI устанавливается аппаратно в конце периода передачи стоп-бита во всех режимах. Соответствующая подпрограмма обслуживания прерывания должна сбрасывать бит TI.

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

Скорость приема / передачи.

Скорость приема/передачи, т.е. частота работы приемопередатчика в различных режимах, определяется различными способами.

В режиме 0 частота передачи зависит только от резонансной частоты кварцевого резонатора f(рез): f = f(рез)/12.За машинный цикл последовательный порт передает один бит информации.

В режимах 1, 2 и 3 скорость приема/передачи зависит от значения управляющего бита SMOD в регистре специальных функций PCON (таблице 12).

Таблица 12 - Регистр управления мощностью PCON

Символ	Позиция	Наименование и функция
SMOD	PCON.7	Удвоенная скорость передачи. Если бит установлен в 1, то скорость передачи вдвое больше, чем при SMOD=0. По сбросу SMOD=0.
-	PCON.6	Не используется
-	PCON.5	Не используется
-	PCON.4	Не используется
GF1	PCON.3	Флаги, специфицируемые пользователем (флаги GF0 PCON.2 общего назначения)
PD	PCON.1	Бит пониженной мощности. При установке бита в 1 ИМС переходит в режим пониженной потребляемой мощности
IDL	PCON.0	Бит холостого хода. Если бит установлен в 1, то ИМС переходит в режим холостого хода
Примечание - При одновременной записи 1 в PD и IDL бит PD имеет преимущество. Сброс содержимого PCON выполняется путем загрузки в него кода OXXXOOOO.		

В режиме 2 при SMOD=0 частота передачи равна (1/64)f(рез), а при SMOD=1 равна (1/32)f(рез).

В режимах 1 и 3 в формировании частоты передачи кроме управляющего бита SMOD принимает участие таймер 1. При этом частота передачи зависит от частоты переполнения f(OVLT1):

f = (2) * (f(OVLT1) / 32

Прерывание от таймера 1 в этом случае должно быть заблокировано. Сам же таймер может работать при этом как в режиме таймера, так и в режиме счетчика. Номер режима (0, 1, 2) роли не играет.

Наиболее типичным является использование таймера в режиме таймера с автоперезагрузкой (старшая тетрада TMOD=0010B).

При этом частота передачи определяется выражением:

SMOD

f = (2) * (f(рез)) / { 32 * 12 * [256 - (TH1)] }

В таблице 13 приводится описание способов настройки Т/С1 для получения типовых частот передачи данных через приемопередатчик.

Таблица 13 - Настройка таймера 1 для управления частотой работы приемопередатчика

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

Предельно низких частот приема / передачи можно достичь при использовании таймера в режиме 1 (16-битовый таймер) и разрешении прерываний от таймера (старший полубайт TMOD = 0001B).

Перезагрузка 16-битового таймера должна осуществляться программным путем.

Особенности работы приемопередатчика в различных режимах.

Режим 0. Данные передаются и принимаются через вывод RXD. Через вывод TXD выдаются синхросигналы сдвига. Передаются / принимаются 8 бит младшим битом вперед, частота обмена - фиксированная, равная 1/12 частоте резонатора.

Передача начинается любой командой, по которой в SBUF поступает байт данных. В момент времени S6P2 устройство управления ИМС по сигналу "Запись в буфер" записывает байт в сдвигающий регистр передатчика, устанавливает триггер девятого бита и запускает блок управления передачей, который через один машинный цикл вырабатывает внутренний разрешающий сигнал "Посылка". При этом в момент S6P2 каждого машинного цикла содержимое сдвигающего регистра сдвигается вправо (младшими битами вперед) и поступает на выход RXD. В освобождающиеся старшие биты сдвигающего регистра передатчика записываются нули. При получении от детектора нуля сигнала "Передатчик пуст" блок управления передатчиком снимает сигнал "Посылка" и устанавливает флаг TI (момент S1P1 десятого машинного цикла после поступления сигнала "Запись в буфер").

Прием начинается при условии REN=1 и R1=0. В момент S6P2 следующего машинного цикла блок управления приемником формирует разрешающий сигнал "Прием", по которому на выход TXD передаются синхросигналы сдвига и в сдвигающем регистре приемника начинают формироваться значения бит данных, которые считываются с входа RXD в моменты S5P2 каждого машинного цикла. В момент S1P1 десятого машинного цикла после сигнала "Запись в SCON" блок управления приемником переписывает содержимое сдвигающего регистра в буфер, снимает разрешающий сигнал "Прием" и устанавливает флаг RI.

Режим 1. Через выход TXD приемопередатчик передает, а со входа RXD принимает 10 бит: старт-бит(0), 8 бит данных и стоп-бит (1). При приеме стоп-бит поступает в бит RB8 регистра SCON.

Передача инициируется любой командой, в которой получателем байта является регистр SBUF. Генерируемый при этом управляющий сигнал "Запись в буфер" загружает 1 в девятый бит сдвигающего регистра передатчика, запускает блок управления передачей и в момент времени S1P1 формирует разрешающий сигнал "Посылка".

По этому сигналу на вывод TXD сначала поступает старт-бит, а затем (по внутреннему разрешающему сигналу "Данные") биты данных. Каждый период передачи бита равен 16 тактам внутреннего счетчика-делителя на 16.

Прием начинается при обнаружении перехода сигнала на входе RXD из состояния 1 в состояние 0. Для этого под управлением внутреннего счетчика вход RXD опрашивается 16 раз за период представления бита. Как только переход из 1 в 0 на входе RXD обнаружен, в сдвигающий регистр приемника загружается код 1FFH, внутренний счетчик по модулю 16 немедленно сбрасывается и перезапускается для выравнивания его переходов с границами периодов представления принимаемых бит. Таким образом, каждый период представления бита делится на 16 периодов внутреннего счетчика.

В состояниях 7, 8 и 9 счетчика в каждом периоде представления бита производится опрос сигнала на входе RXD. Считанное значение принимаемого бита - это то, которое было получено по меньшей мере дважды из трех замеров (мажоритарное голосование по принципу "два из трех"). Если значение, принятое в первом такте, не равно 0, то блок управления приемом вновь возвращается к поиску перехода из 1 в 0. Этот механизм обеспечивает подавление ложных (сбойных) старт-бит. Истинный старт-бит сдвигается в регистре приемника, и продолжается прием остальных бит посылки.

Блок управления приемом сформирует сигнал "Загрузка буфера", установит RB8 и флаг RI только в том случае, если в последнем такте сдвига выполняются два условия: бит RI=0, и либо SM2=0, либо принятый стоп-бит равен 1. Если хотя бы одно из этих двух условий не выполняется, то принятая последовательность бит теряется. В это время вне зависимости от того, выполняются указанные условия или нет, блок управления приемником вновь начинает отыскивать переход из 1 в 0 на входе RXD.

Режимы 2, 3. Через вывод TXD приемопередатчик передает или с вывода RXD принимает 11 бит: старт-бит (0), 8 бит данных, программируемый девятый бит и стоп-бит (1). На временных диаграммах (рисунки 13, 14) показана работа приемопередатчика при передаче и приеме данных в режимах 2 и 3. Как видно из них, в режимах 2 и 3 прием данных ничем не отличается от приема в режиме 1 за исключением того, что в бит RB8 в этих режимах заносится не стоп-бит, а девятый бит данных. Естественно, это никак не изменяет временные диаграммы в режимах 2 и 3 в сравнении с режимом 1. Необходимо также отметить, что несколько изменяются условия окончания цикла приема: блок управления приемником сформирует управляющий сигнал "Загрузка буфера", загрузит RB8 и установит флаг RI только в том случае, если в последнем такте сдвига выполняются два условия: бит RI=0 и либо SM2=0, либо значение принятого девятого бита данных равно 1.

При передаче данных в режимах 2 и 3 отличие от режима 1 состоит в том, что передаются не 8, а 9 бит данных, и вследствие этого цикл передачи оказывается на 1 полный период работы счетчика-делителя на 16 длиннее. Кроме того, как отмечалось выше, скорость обмена в режиме 2, в отличие от режимов 1 и 3, фиксирована и равна 1/32 или 1/64 от величины f(рез).

4.8 Работа приемопередатчика в мультимикропроцессорных системах

Архитектурой ИМС обеспечивается работа приемопередатчика в системах децентрализованного управления. Такие системы используются для управления и регулирования в распределенных объектах. При этом возникает задача обмена информацией между

множеством микроконтроллеров, объединенных в локальную вычислительно-управляющую сеть.

В регистре специальных функций SCON ИМС имеется управляющий бит SM2, который в режимах 2 и 3 приемопередатчика позволяет относительно простыми средствами реализовать межпроцессорный обмен информацией в локальных управляющих сетях.

Механизм межпроцессорного обмена информацией через последовательный порт ИМС построен на том, что в режимах 2 и 3 программируемый девятый бит данных при приеме фиксируется в бите RB8. Приемопередатчик может быть запрограммирован таким образом, что при получении стоп-бита прерывание от приемника будет возможно только при условии RB8=1. Это выполняется установкой управляющего бита SM2 в регистре SCON.

Поясним процесс межпроцессорного обмена информацией на примере. Пусть ведущей ИМС требуется передать блок данных некоторой (или нескольким) ведомым. С этой целью ведущая ИМС в протокольном режиме "широковещательной" передачи (всем ведомым) выдает в моноканал байт-идентификатор абонента (код адреса ИМС-получателя), который отличается от байтов данных только тем, что в его девятом бите содержится 1. Программа реализации протокола сетевого обмена информацией должна быть построена таким образом, чтобы при получении байта-идентификатора (RB8=1) во всех ведомых ИМС произошли прерывания прикладных программ и вызов подпрограммы сравнения байта-идентификатора с кодом собственного сетевого адреса. Адресуемая ИМС сбрасывает свой управляющий бит SM2 и готовится к приему блока данных. Остальные ведомые ИМС, адрес которых не совпал с кодом байта-идентификатора, оставляют неизменным состояние SM2=1 и передают управление основной программе. При SM2=1 информационные байты, передаваемые по моноканалу и поступающие в приемопередатчик ведомых ИМС, прерывания не вызывают, т.е. игнорируются.

В режиме 1 приемопередатчиком автономной ИМС управляющий бит SM2 используется для контроля истинности стоп-бита (при SM2=1 прерывание не произойдет до тех пор, пока не будет получено истинное (единичное) значение стоп-бита). В режиме 0 бит SM2 не используется и должен быть сброшен.

4.9 Сброс ИМС

Сброс ИМС осуществляется единичным уровнем сигнала. Этот сигнал должен быть приложен к выводу RST.

Сброс достигается удержанием вывода RST в высоком логическом уровне в течение, по крайней мере, 2-х машинных циклов (24 периодов колебаний) при работающем генераторе. В CPU осуществляется сброс.

Внешний сигнал сброса асинхронен в отношении внутренних сигналов ИМС. Вывод RST опрашивается в течение состояния 5 фазы 2 каждого машинного цикла. Поскольку выводы портов сохраняют свое текущее состояние в течение 19 периодов колебаний после обнаружения единицы на выводе RST, длительность сигнала сброса должна быть большей 19 периодов колебаний после сигнала внешнего сброса.

При сбросе во все регистры специальных функций кроме регистров-защелок портов ввода-вывода, указателя стека и SBUF записываютсят нули. В защелках портов при инициализации установлено 0FFH, в указателе стека - 07H, а содержимое SBUF неопределено. В табл.14 приведен список регистров специальных функций и их значений после прохождения сигнала сброса.

Таблица 14 - Состояние регистров SFR после сброса

Наименование регистра	Значение после сброса
PC	0000H
ACC	00H
B	00H
PSW	00H
SP	07H
DPTR	0000H
P0...P3	0FFH
IP	XXX00000B
IE	0XX00000B
TMOD	00H
TCON	00H
TH0	00H
TL0	00H
TH1	00H
TL1	00H
SCON	00H
SBUF	XXXXXXXXB
PCON (n-МОП)	0XXXXXXXB
PCON (к-МОП)	0XXX0000B

Примечание - X - значение бита неопределено

Сброс не оказывает воздействия на состояние ячеек внутреннего ОЗУ ИМС. Однако необходимо учитывать, что их состояние после включения питающего напряжения не определено.

4.10 Сброс по включении питания

Если подача питающего напряжения не сопровождается достоверным сбросом (т.е. удержанием единичного уровня на входе RST в течение 24 периодов резонатора), то ИМС

может начать выполнение программы до того, как в регистры специальных функций будут занесены значения. При этом нельзя гарантировать корректность выполнения программы. Следовательно, ИМС должна иметь цепи, обеспечивающие автоматическое формирование сигнала сброса при включении питания. Значение емкости для ИМС по цепи сброса составляет значение порядка 1 мкФ.

При включении питания подобная цепь удерживает высокий уровень на входе RST в течение времени, которое зависит от значения емкости и уровня, до которого она заряжена. Для гарантии достоверного сброса этот единичный уровень должен сохраняться дольше, чем генератор выработает 2 машинных цикла.

Пока не запустится генератор ИМС и не выполнится алгоритм сброса, выводы портов P0...P3 будут находиться в неопределенном состоянии.

4.11 Режимы работы ИМС с пониженным энергопотреблением

Во многих вариантах использования ИМС энергопотребление является одним из основных параметров.

ИМС имеет 2 режима с пониженным потреблением:

- режим холостого хода (IDLE);
- режим пониженного напряжения питания ("Power Down").

В режиме холостого хода (IDL=1) генератор ИМС работает, подавая вырабатываемые тактовые сигналы на схему прерываний, последовательный порт и на таймеры-счетчики. Все регистры сохраняют свое значение, на выводах всех портов удерживается то логическое состояние, которое было на них в момент перехода в режим холостого хода. Однако синхросигнал генератора, синхронизирующий CPU, отключается.

В режиме пониженного напряжения питания (PD=1) генератор останавливается. Прекращается тактирование не только CPU, но и последовательного порта, таймеров-счетчиков, схемы прерываний. Как и в режиме холостого хода, состояние регистров, резидентного ОЗУ и выводов портов остается неизменным. Режимы холостого хода и выключенного питания активируются при установке соответствующих битов в специальном регистре - регистре управления мощностью PCON. Адрес этого регистра 87H.

Если одновременно установлены в единицы биты IDL и PD, то бит PD имеет преимущество - ИМС переходит в режим выключенного напряжения питания. Содержимое регистра PCON после сброса - (0XXX0000).

4.12 Режим холостого хода

В режим холостого хода ИМС переводится любой командой, устанавливающей в 1 бит PCON.0. Эта команда оказывается последней в цепочке выполняемых команд - в режиме холостого хода выполнение программы приостанавливается, т.к. на CPU перестает подаваться сигнал с тактового генератора. Однако содержимое внутреннего ОЗУ и регистров специальных функций остается неизменным, выводы портов удерживают значения, которые были на них до перехода в режим холостого хода, на таймеры-счетчики, приемопередатчик и на схему прерывания продолжают поступать тактовые сигналы. Состояние выводов портов зависит от типа ОЗУ, с которым ИМС обменивалась информацией перед переходом в режим холостого хода. При работе с ОЗУ на выводах портов присутствуют данные из соответствующих SFR (естественно, если порт в режиме вывода информации). Прекратить холостой ход возможно двумя способами. Вызов любого из прерываний приведет к аппаратному стиранию PCON.0, прекращающему холостой ход. Прерывание будет обслужено и следующей после RETI выполняемой командой будет та команда, которая следует за командой, приведшей к переходу ИМС в режим холостого хода.

Флаги GF0 и GF1 могут использоваться для индикации того, произошло ли прерывание во время нормальной работы или во время холостого хода. Например, команда, запускающая холостой ход, может также устанавливать один или оба флага. Когда холостой ход прекращен прерыванием, сервисная программа прерывания может проверять состояние флагов.

Другой способ прекращения холостого хода - с помощью аппаратного сброса. Поскольку синхрогенератор продолжает работать, аппаратный сброс должен поддерживаться в активном состоянии только в течение 2-х машинных циклов (24 периодов колебаний).

Сигнал сброса стирает бит PCON.0. В этот момент CPU возобновляет выполнение программы с того места, где оно было остановлено; таким образом, следующая команда - та, что следует за командой, вызвавшей холостой ход. Перед началом отработки алгоритма внутреннего запуска могут иметь место 2 или 3 машинных цикла выполнения программы. Встроенное в микросхему устройство в это время препятствует доступу к внутреннему ОЗУ, но доступ к выводам порта не ограничен. Чтобы исключить возможность появления неопределенных выходных сигналов на выводах порта, команда, следующая за вызывающей холостой ход, не должна быть командой, записывающей информацию в SFR порта или во внешнее ОЗУ данных.

Напомним, что после аппаратного сброса содержимое SFR переопределяется.

4.13 Режим пониженного напряжения питания

Команда, устанавливающая в 1 бит PCON.1, переводит ИМС в режим пониженного напряжения питания. В этом режиме генератор ИМС останавливается. С остановкой синхрогенератора прекращает функционирование не только CPU, но и таймеры-счетчики, приемопередатчик, схема прерываний. При наличии основного или резервного источника питающего напряжения встроенное ОЗУ и регистры SFR сохраняют свое содержимое. Состояние портов не отличается от состояния при переходе в режим холостого хода (см. выше). Единственный способ выйти из этого режима – аппаратный сброс. Он переопределяет содержимое всех SFR, но не меняет содержимого встроенного ОЗУ.

4.14 Встроенный тактовый генератор

Архитектурой микроконтроллера предусмотрена возможность выключения генератора программным путем (записью 1 в бит PD регистра PCON). Сигнал внешнего синхросигнала подается на вход X1 (вход X2 не используется).

Генератор можно использовать с внешними компонентами. Обычно, если элементом обратной связи является кристалл кварца, то C1 = C2 = 30 pF, а если используется керамический резонатор - C1 = C2 = 47 пФ. (рисунки 5, 5.1)

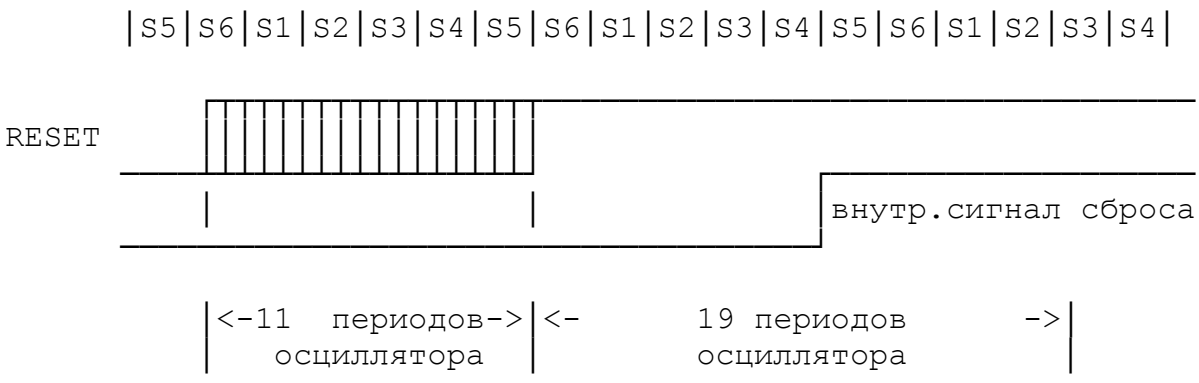


Рисунок 3 - Временные диаграммы сигналов ИМС при сбросе

G – генератор импульсов.

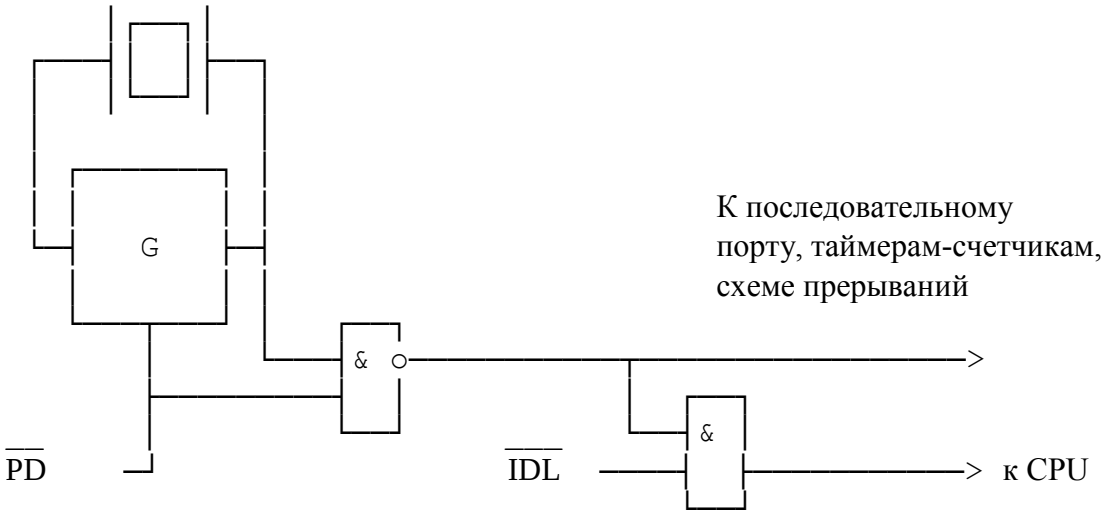


Рисунок 4 - Упрощенная схема аппаратной реализации режимов пониженного энергопотребления

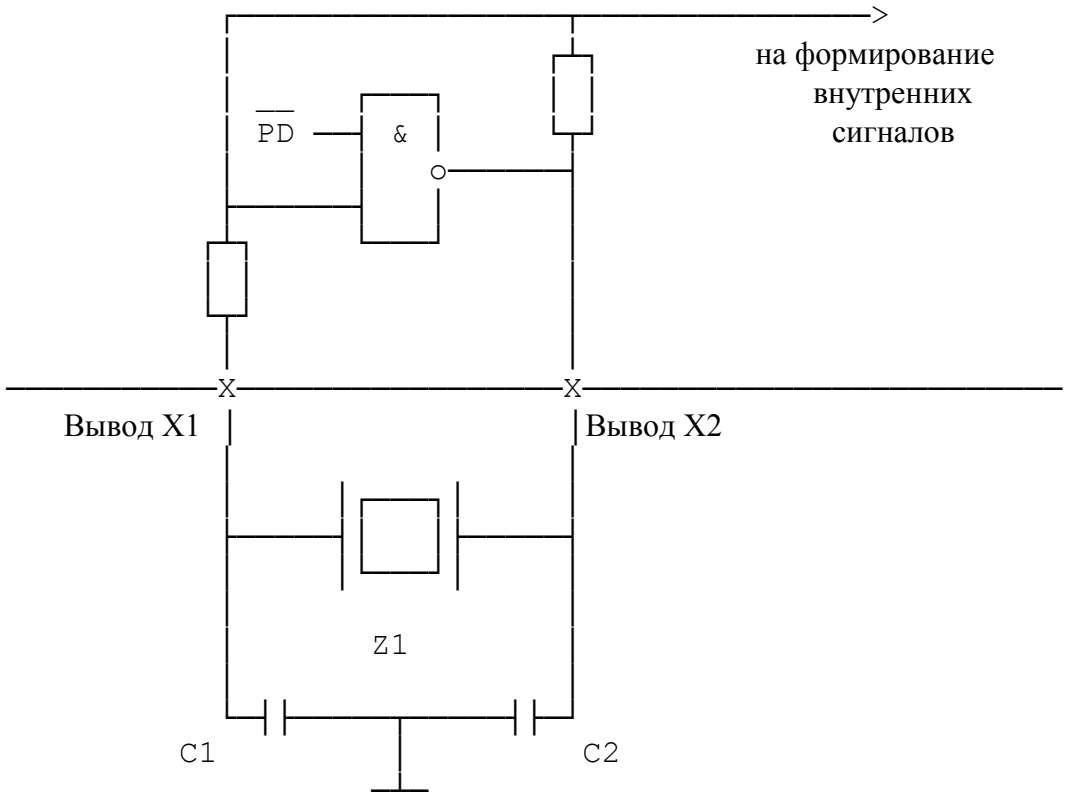
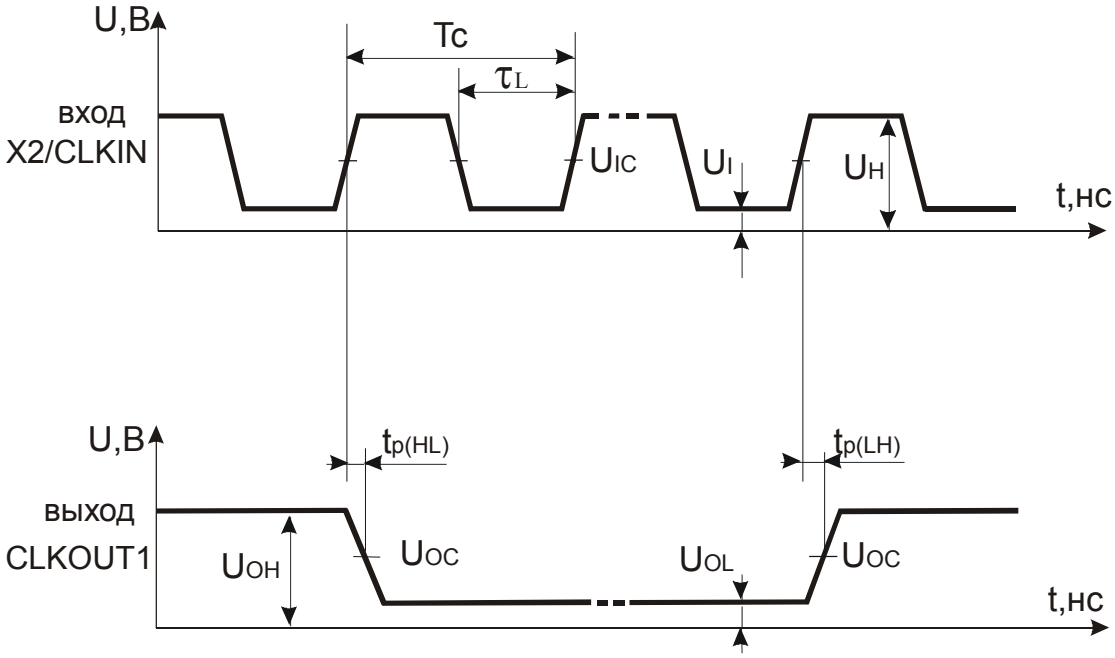


Рисунок 5 - Тактовый генератор ИМС



T_C = 50 нс; τ_L =0,2T_C;
длительность фронтов X2/CLKIN не более 8нс;
t_p – время распространения сигнала: минимум - 20 нс, максимум - 110 нс.
U_{IL} =$U_L \le$0,1 В; U_H=U_{CC}; U_{IH}=(U_{CC} -0,5) В;
U_{OH}, U_{OL} - выходное напряжение высокого и низкого уровня;
U_{IC} = U_{OC} = 1,5 В – уровни измерения t_p.

Рисунок. 5.1 - Временная диаграмма работы тактового генератора

5 Адресация

5.1 Косвенно-регистровая адресация

Косвенно-регистровая адресация используется для обращения к ячейкам внутреннего ОЗУ данных. В качестве регистров-указателей используется регистры R10, R1 выбранного банка регистров.

В командах PUSH и POP используется содержимое указателя стека (SP).

5.2 Непосредственная адресация

Непосредственная адресация позволяет выбрать из адресного пространства памяти программ константы, явно указанные в команде.

5.3 Прямая адресация

Прямая байтовая адресация используется для обращения к ячейкам внутренней памяти (ОЗУ) данных (0-127) и к регистрам специального назначения.

Прямая побитовая адресация используется для обращения к отдельно адресуемым 128 битам, расположенным в ячейках с адресами 20H-2FH и к отдельно адресуемым битам регистров специального назначения.

Старший бит байта кода прямого адреса выбирает одну из двух групп отдельно адресуемых битов, расположенных в ОЗУ или регистрах специального назначения. Прямо адресуемые биты с адресами 0-127 (00H-7FH) расположены в блоке из 16 ячеек внутреннего ОЗУ, имеющих адреса 20H-2FH. Указанные ячейки последовательно пронумерованы от младшего бита младшего байта до старшего бита старшего байта. Отдельно адресуемые биты в регистрах специального назначения пронумерованы следующим образом: пять старших разрядов адреса совпадают с пятью старшими разрядами адреса самого регистра, а три младших - определяют местоположение отдельного бита внутри регистра.

5.4 Регистровая адресация

Регистровая адресация используется для обращения к восьми рабочим регистрам выбранного банка рабочих регистров (эти же регистры могут быть выбраны с помощью прямой адресации и косвенно-регистровой адресации как обычные ячейки внутреннего ОЗУ данных).

Регистровая адресация используется для обращения к регистрам A, B, AB (сдвоенному регистру), DPTR и к флагу переноса C. Использование регистровой адресации позволяет получать двухбайтовый эквивалент трехбайтовых команд прямой адресации.

5.5 Косвенно-регистровая адресация по сумме базового и индексного регистров

Косвенно-регистровая адресация по сумме: базовый регистр, плюс индексный регистр (содержимое аккумулятора A) упрощает просмотр таблиц, зашитых в памяти программ. Любой байт из таблицы может быть выбран по адресу, определяемому суммой содержимого DPTR или PC и содержимого A.

6 Особенности ИМС

ИМС 1830ВЕ81Т, 1830ВЕ91Т являются экономичным и эффективным решением, полностью совместимым с MCS-51 архитектурой. Однако, имеются ограничения на определенные инструкции при программировании этой ИМС.

Все инструкции относительно условных и безусловных переходов должны быть ограничены так, чтобы адрес назначения находился внутри области программной памяти,

равной 2К. Ответственность за это несет программист. Для примера, LJMP 7E0H будет разрешенной инструкцией, в то время, как LJMP 900H не разрешена.

ИМС содержит 128 байт внутренней памяти данных. Т.о. глубина стека ограничена 128 байтами, имеющимися в распоряжении ОЗУ. Внешние, память данных и память программ, не поддерживаются в этом устройстве. Поэтому инструкция MOVX должна быть исключена из программ.

6.1 Пространство регистров специальных функций.

Карта встроенной матрицы памяти, называемая пространством специальных функциональных регистров (SFR), дана в табл. 15.

Ячейки, соответствующие незанятым адресам, физически отсутствуют на кристалле микросхемы. Чтение при обращении к этим адресам возвращает случайные данные, запись в такие ячейки также даст неопределенный результат.

Пользовательские программы не должны записывать единицы в эти незанятые позиции, т.к. они будут использованы в новых разработках. Гарантируется, что нулевые значения этих битов всегда будут означать отключение вновь введенных функций. Последнее означает, что программы, рассчитанные на использование в ИМС, не имеющих этих новых функций, будут корректно работать и в новых изделиях, если они не активируют упомянутые биты.

Таблица 15 - Специальные функциональные регистры ИМС

0F8H								
0F0H	B							
0E8H								
0E0H	ACC							
0D8H								
0D0H	PSW							
0C8H								
0C0H								
0B8H	IP							
0B0H	P3							
0A8H	IE							
0A0H								
098H	SCON	SBUF						
090H	P1							
088H	TCON	TMOD	TL0	TL1	TH0	TH1		
080H		SP	DPL	DPH				PCON

6.2 Особенность приемопередатчика

Особенностью приемопередатчика является наличие в приемопередатчике устройства детектирования ошибок формата (Framing Error Detection). Это устройство позволяет последовательному порту проверять правильность стоп-битов в режимах 1, 2 или 3. Потеря стоп-бита может возникнуть, например, из-за шумов на линиях передачи или из-за одновременной передачи информации двумя процессорами мультимикропроцессорной системы.

Если бит остановки потерян, устанавливается бит ошибки формата (FE). Этот бит может быть проверен программно после каждого приема для обнаружения ошибок связи. Будучи однажды установлен, бит ошибки формата может быть сброшен только программным путем. Если после установки бита ошибки формата он не был сброшен программно, а затем в результате следующего цикла обмена информацией получен байт данных с правильным стоп-битом, сброса бита ошибки формата не произойдет.

FE бит расположен в SCON и делит один адрес с SM0. Бит управления SMOD0 в регистре PCON (PCON.6) определяет, к какому биту обращается микропроцессор - к SM0 или к FE. Если SMOD0 = 0, то обращение - к SM0. Если SMOD0 = 1, то обращение - к FE.

6.3 Режим пониженного энергопотребления

ИМС может выходить из режима пониженного энергопотребления как при аппаратном сбросе, так и при внешнем прерывании. Сброс переопределяет все регистры специальных функций, но не меняет содержимого внутреннего ОЗУ. Внешнее прерывание позволяет как регистрам (кроме бита PD в PCON), так и встроенному ОЗУ сохранять свои значения. Чтобы обеспечить правильный выход из этого режима, сброс или внешнее прерывание не должны подаваться прежде, чем VCC восстановит свой нормальный рабочий уровень, и должны удерживаться достаточно долго, чтобы генератор перестартовал и стабилизировался (обычно - менее 10 мс).

При использовании для вывода из режима пониженного энергопотребления внешних прерываний $\overline{\text{INT0}}$ или $\overline{\text{INT1}}$ схема прерываний должна быть конфигурирована на срабатывание по уровню соответствующего сигнала. Удержание вывода при нулевом потенциале перестартует генератор, а перевод вывода в единичный уровень завершит выход. После выполнения команды RETI в подпрограмме обслуживания прерывания следующей будет выполняться та команда, которая идет после той, что перевела ИМС в режим пониженного энергопотребления.

6.4 Флаг отключения питания

Флаг отключения питания (POF), расположенный в PCON.4, устанавливается аппаратно, когда VSS повышается от 0 до примерно 5В. POF может также быть установлен или стерт программно. Это позволяет пользователю различать сбросы "холодного" и "теплого" стартов.

Сброс "холодного" старта совпадает с подачей на ИМС напряжения питания. Сброс "теплого" старта происходит без отключения питания, например при выходе из режима пониженного энергопотребления.

Сразу же после сброса пользовательская программа может проверить состояние бита POF. POF = 1 означает холодный старт.

Программа затем стирает POF и начинает свои задачи. POF = 0 сразу после сброса означает теплый старт.

Для того, чтобы флаг POF удерживался в сброшенном состоянии, необходимо, чтобы напряжение питания не опускалось ниже 3 В.

6.5 Защита внутренней памяти программ

ИМС содержит два бита защиты, которые могут оставаться незапрограммированными (Н) или быть запрограммированы (З) для получения дополнительных свойств.

Таблица 16 - Биты защиты внутреннего ПЗУ

Бит1	Бит2	Дополнительные свойства
Н	Н	Нет дополнительных свойств
З	Н	Дальнейшее программирование FLASH запрещено
З	З	Тоже, что и режим 2, но также запрещена верификация FLASH

Если бит 1 запрограммирован, невозможно дальнейшее программирование FLASH памяти.

Если запрограммированы оба бита, в дополнении к вышеназванной особенности добавляется невозможность верификации программного кода.

Биты защиты могут быть стерты процедурой стирания FLASH.

7. Программирование ИМС 1830ВЕ91Т

ИМС содержит 12-вольтовый интерфейс программирования. Массив FLASH памяти программируется побайтно. Перед перепрограммированием массива необходимо проводить процедуру электрического стирания.

ИМС включает внутренний счетчик адреса, который всегда сбрасывает в 000Н по переднему фронту сигнала RST, и инкрементирует свое значение по переднему фронту импульса на выводе XTAL1.

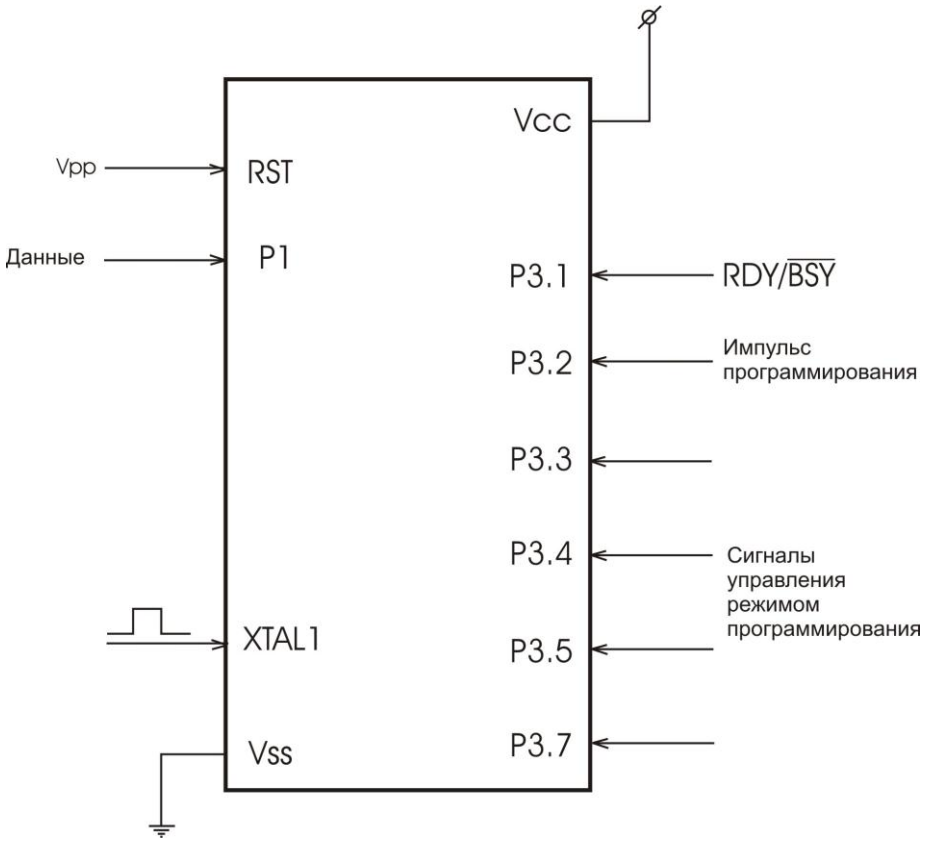


Рисунок 6 - Схема включения микроконтроллера при программировании

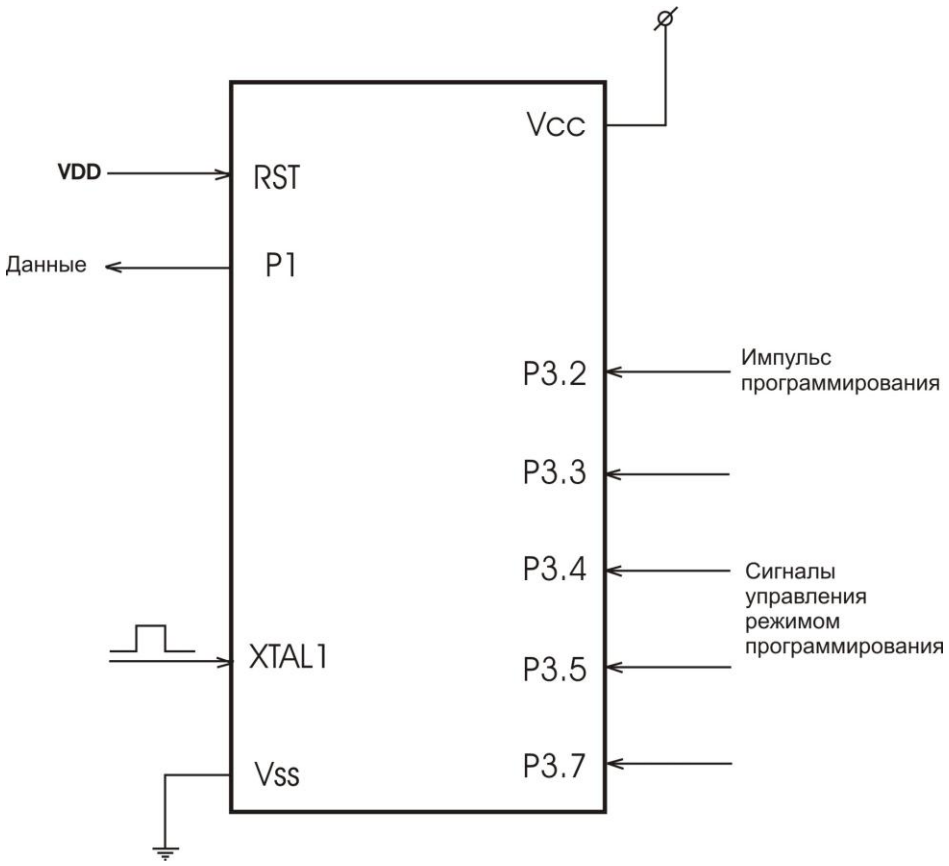


Рисунок 7 - Схема включения микроконтроллера при верификации

Таблица 17 - Сигналы программирования и верификации параллельного режима программирования

Режим		RST/Vpp	P3.2/PROG	P3.3	P3.4	P3.5	P3.7
Запись кода данных		12 В		L	H	H	H
Чтение кода данных		H	H	L	L	H	H
Запись битов защиты		12 В		H	H	H	H
		12 В		H	H	L	L
Стирание		12 В	[1]	H	L	L	L
Чтение байт сигнатуры		H	H	L	L	L	L

где 1 – процедура стирания требует импульса $\overline{\text{PROG}}$ длительностью 10 мс.

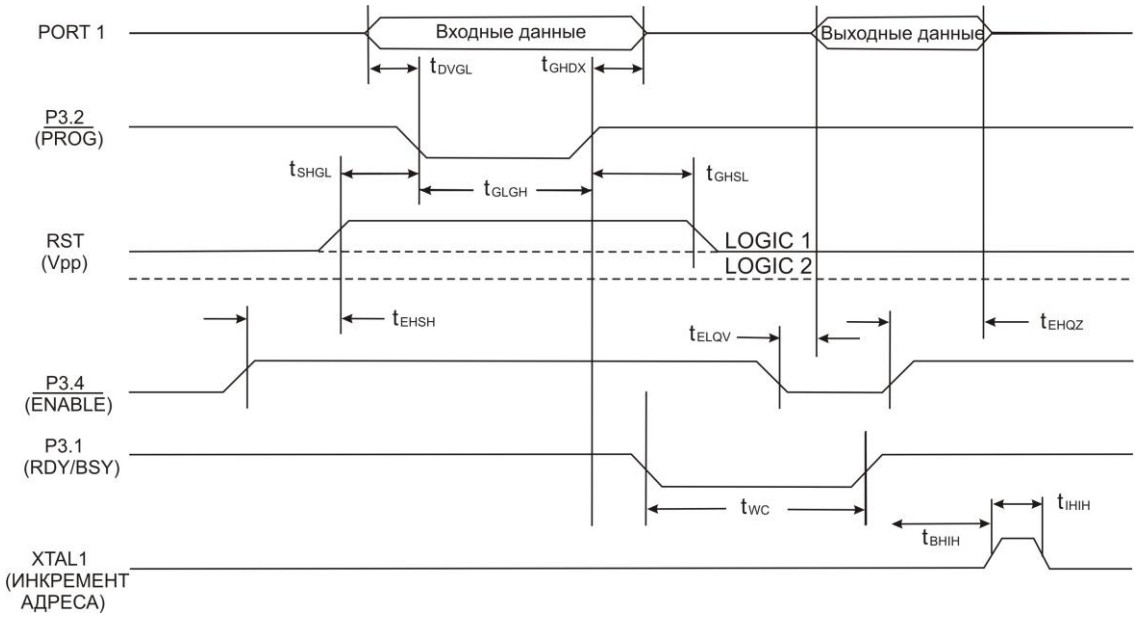


Рисунок 8 - Временная диаграмма сигналов в режимах программирования и верификации

Таблица 18 - Значения тока, напряжения, длительности импульсов и задержек при программировании и верификации FLASH - памяти

Параметр, единица измерения	Буквенное обозначение	Норма	
		Мин.	Макс.
1	2	3	4
1.Напряжение программирования, В	Vpp	12,5	13,5
2. Ток программирования, мА.	Ipp		250
3. Установка данных относительно низкого уровня импульса программирования, мкс	TDVGL	1,0	
4. Удержание данных после завершения импульса программирования, мкс	TGHDX	1,0	
5. Смещение между установленным P3.4 и Vpp, мкс	TEHSX	1,0	
6. Смещение между установленным Vpp и активным импульсом программирования, мкс	TSHGX	10	
7. Удержание Vpp после завершения импульса программирования, мкс	TGHSX	10	
8. Длина импульса программирования, мкс	TGLGH	1	

1	2	3	4
9. Смещение между низким уровнем Enable и верифицируемыми данными, мкс	TELQV		1,0
10. Смещение между началом верифицируемых данных и высоким уровнем Enable	TEHQZ		1,0
11. Смещение между высоким уровнем импульса программирования и низким уровнем сигнала на P3.1, нс	TGHBL		50
12. Время цикла записи байта, мс	TWC		2,0
13. Смещение между высоким уровнем RDY/BSY и инкрементирующим тактовым импульсом, мкс	TBHIH	1,0	
14. Длительность инкрементирующего тактового импульса, нс		200	

Алгоритм программирования состоит из следующих последовательных шагов:

1. Подать напряжение питания на VCC, низкий логический уровень на RST, XTAL1.

2. Подать высокий логический уровень на RST, P3.2.

3. Назначить соответствующую комбинацию высоких и низких уровней на выводы P3.3, P3.4, P3.5, P3.7 для выбора режима программирования.

4. Подать данные на P1.

5. Увеличить напряжение на выводе RST до 12В.

6. Подать импульс программирования на P3.2. Длительность импульса типично составляет 1,2 мс.

7. Для верификации данных снизить уровень напряжения на выводе RST с 12 В до высокого логического уровня и установить соответствующие уровни на выводах P3.3 и P3.7. Верифицируемые данные считывать с порта P1.

8. Для программирования следующего байта подать импульс инкрементирования на XTAL1 и подать следующий байт данных на P1.

9. Повторить шаги с 5 по 8, изменяя данные и инкрементируя адресный счетчик до полного заполнения массива памяти программ или до завершения кода программы.

10. Порядок выключения питания:

- подача на XTAL1 низкого логического уровня;
- подача на RST низкого логического уровня;
- отключение напряжения питания.

8 Рекомендации по отладочным средствам ИМС

8.1 Программаторы для программируемого варианта ИМС

Могут использоваться программаторы любой фирмы, обеспечивающие программирование аналогов разработанной ИМС (AT89C2051 фирмы Atmel), например, типа PicProg+, ChipProg, ChipProg+ ОАО «Фитон», г. Москва.

8.2 Описание инструментальных средств для ИМС

Ниже приводится краткая информация о средствах отладки, поставляемых ООО «Фитон», г. Москва, обеспечивающего отладку систем на базе разработанного микроконтроллера.

Реквизиты: **ООО "Фирма ФИТОН",**

127474, г. Москва, Дмитровское шоссе д.62, кор.2

Тел/факс (095) 904-8631 904-8652

E-mail: phyton@phyton.ru

http://www.phyton.ru

Интегрированный пакет разработки и отладки систем на базе микроконтроллера.

Пакет Project-52 - набор программно-аппаратных средств 4-го поколения, предназначенный для разработки и отладки систем на базе микроконтроллеров семейства 8051 фирмы Intel. Концепция Project-52- объединение внутрисхемного эмулятора, программного отладчика-симулятора, компиляторов, текстового редактора, менеджера проектов и программатора в рамках единой интеллектуальной среды разработки. При наличии одного из программаторов PicProg+, ChipProg, ChipProg+ пакет поддерживает работу и с программатором.

Полная конфигурация пакета называется Project-52/ESA и включает в себя:

- Менеджер проектов;
- Кросс-компиллятор языка ассемблер **MCA-51**;
- Отладчик-симулятор **PDS-52**;
- Внутрисхемный эмулятор **PICE-52**

Внутрисхемный эмулятор для микроконтроллера.

PICE-52 - эмулятор нового поколения, созданный с применением новых технологий разработки аппаратуры и программного обеспечения. Применение программируемых

матриц большой емкости позволило значительно сократить размеры эмулятора без какого-либо ущерба его функциональным возможностям, свести к минимуму отличия в электрических и частотных характеристиках эмулятора от характеристик эмулируемого процессора и, тем самым, добиться максимальной точности эмуляции на частотах до 70 Мгц при напряжениях питания от 1,8В до 5,5 В. Перезагружаемая аппаратная структура эмулятора обеспечивает эмуляцию практически всех микроконтроллеров семейства Intel 8051.

Программная поддержка PICE-52 работает в среде Windows-95/98/ME/NT/2000/XP и предоставляет пользователю обширный сервис как по разработке программ, так и по их отладке. Эмулятор состоит из основной платы размером 95x70x40мм, сменного пода под определенную группу процессоров и сменного адаптера под конкретный тип корпуса. На основной плате реализованы: трассировщик, процессор точек останова. Плата сменного пода содержит эмулирующий процессор под конкретный тип микроконтроллера. Сменные адаптеры обеспечивают установку эмулятора в колодки DIP, PLCC, а также на посадочные места QFP, SOIC и SSOP на плате пользователя. Питание эмулятора осуществляется от блока питания 3,3В, 1А или непосредственно от отлаживаемого устройства. Связь с компьютером - по гальванически развязанному каналу RS-232C на скорости 115 КБод или по каналу USB.

Характеристики аппаратуры

- Точная эмуляция - отсутствие каких-либо ограничений на использование программой пользователя ресурсов микроконтроллера;
- До 1М эмулируемой памяти программ и 512К памяти данных при использовании основной платы MR1-52-05 (до 64К памяти программ и 64К памяти данных при использовании основной платы MR1-52-03);
- Поддержка банкированной модели памяти - до 32 банков размером по 64К;
- Распределение памяти между эмулятором и устройством пользователя с точностью до 256 байт;
- Анализатор использования памяти программ и данных. Отображение информации анализа в физическом и символьном форматах;
- Аппаратная поддержка отладки программ на языках высокого уровня;
- Трассировка 8 произвольных внешних сигналов;
- 4 выхода синхронизации аппаратуры пользователя;
- Трассировщик реального времени с буфером объемом до 64К фреймов по 128 бита при использовании основной платы MR1-52-05 (до 16К фреймов по 128 бита при использовании основной платы MR1-52-03) с доступом "на лету". Трасси-

ровка шины адреса/данных памяти программ, внутренних и внешних данных, EEPROM, сигналов управления, таймера реального времени и 8-ми внешних сигналов пользователя;

- Программируемый фильтр трассировки, до 1M триггеров запуска и останова трассировщика;
- До 1M аппаратных точек останова по доступу к памяти программ;
- До 512K аппаратных точек останова по доступу к внешней памяти данных;

Характеристики программного обеспечения

- Программное обеспечение работает в среде Windows-95/98/ME/NT/2000/XP;
- Поддерживается разработка программ на уровне ведения проектов для макроассемблера MCA-51 нашей фирмы, который входит в комплект поставки, а также для пакетов кросс-средств языка Си и ассемблера фирм IAR Systems, Keil Software и Raisonance. Помимо указанных пакетов, поддерживается полнофункциональная символьная отладка программ, созданных с помощью компиляторов фирм Avocet Systems, Hi-Tech Software, Tasking, Crossware, Intel.
- Автоматическое сохранение и загрузка файлов конфигурации аппаратуры, интерфейса и опций отладки. Обеспечивается совместимость файлов конфигурации с симулятором PDS-52. Обеспечена переносимость проектов между эмулятором PICE-52 и симулятором PDS-52.

Компоненты эмулятора

Для эмулятора PICE-52 существует несколько вариантов основной платы, различающихся по скорости, объему памяти и, соответственно, по цене. Каждый вариант имеет свой номер, присутствующий в конце обозначения платы: MR1-52-XX. Минимальные параметры и цену обеспечивает основная для PICE-52 плата MR1-52-03.

Комплект поставки эмулятора PICE-52

- Руководство пользователя и паспорт (гарантийный талон);
- Компакт-диск с программным обеспечением и документацией;
- Аппаратура эмулятора;
- Кабель связи с компьютером (RS-232C или USB);
- Трассировочный кабель;
- Блок питания;
- Упаковочная коробка.

Отладочные возможности PDS-52

Симулятор PDS-52 представляет собой программно-логическую модель микроконтроллера, имитирующую (симулирующую) работу ядра архитектуры семейства Intel 8051 - памяти, АЛУ, системы команд, регистров (периферийные устройства не поддерживаются). Возможности PDS-52:

- отслеживание выполнения программы по ее исходному тексту;
- просмотр и изменение значений любых переменных;
- встроенный анализатор эффективности программного кода;
- точки останова по сложному условию;
- неограниченное количество точек останова по доступу к ячейкам памяти;

просмотр стека вызовов подпрограмм и функций;

- встроенный строчный ассемблер;
- возможность выполнения программы "назад" на большое количество шагов, а также в непрерывном режиме. При этом состояние модели микроконтроллера полностью восстанавливается;
- точный подсчет интервалов времени и многое другое.

Основные достоинства программно-логической модели микроконтроллера, реализованной в PDS-52 - точная симуляция узлов микроконтроллера и возможность моделировать устройства, подключенные к микроконтроллеру "снаружи" (т.н. моделирование внешней среды), например, внешнюю логику, датчики, клавиатуру, исполнительные устройства (дисплеи), задавать периодические и непериодические воздействия и т.п.

Кросс-макроассемблер. Предназначен для трансляции исходных текстов программ для процессоров семейства 8051 фирмы Intel.

- Кросс-макроассемблер;
- Поддерживает все микроконтроллеры 8051 фирмы Intel;
- Генерирует HEX-файл и подробный листинг ;
- Поддерживает широкий набор директив условной трансляции;
- Предоставляет удобные средства работы с макросами;
- Генерирует подробную символьную информацию для отладчиков;
- Допускает использование русских букв в именах;

9 Общая характеристика системы команд ИМС

Система команд ИМС включает в свой состав 107 основных команд. Длина команд составляет 1, 2 или 3 байта, причем большинство команд (94 %) одно- или двухбайтовые. Все команды выполняются за 1 или 2 машинных цикла (0,5 или 1,0 мкс при тактовой частоте 24МГц соответственно), исключение составляют лишь команды умножения и деления, которые выполняются за 4 машинных цикла (2,0мкс).

В наборе команд имеются следующие арифметические операции: сложение, сложение с учетом флага переноса, вычитание с заемом, инкрементирование, декрементирование, сравнение, десятичная коррекция, умножение и деление.

В АЛУ производятся действия над целыми числами без знака. В двухоперандных операциях: сложение (ADD), сложение с переносом (ADDC) и вычитание с заемом (SUBB) аккумулятор является первым операндом и принимает результат операции. Вторым операндом может быть рабочий регистр выбранного банка рабочих регистров, регистр внутренней памяти данных с косвенно-регистровой и прямой адресацией или байт непосредственных данных. Указанные операции влияют на флаги: переполнения, переноса, промежуточного переноса и флаг четности в слове состояния процессора (PSW).

Использование разряда переноса позволяет многократно повысить точность при операциях сложения (ADDC) и вычитания (SUBB).

Выполнение операций сложения и вычитания с учетом знака может быть осуществлено с помощью программного управления флагом переполнения (OV) регистра PSW. Флаг промежуточного переноса (AC) обеспечивает выполнение арифметических операций в двоично-десятичном коде.

Операции инкрементирования и декрементирования на флаги не влияют.

Операции сравнения не влияют ни на операнд назначения, ни на операнд источника, но они влияют на флаги переноса.

Существуют три арифметические операции, которые выполняются только на аккумуляторе: две команды проверки содержимого аккумулятора A (JZ, JNZ), и команда десятичной коррекции при сложении двоично-десятичных кодов.

При операции умножения содержимое аккумулятора A умножается на содержимое регистра B и результат размещается следующим образом: младший байт в регистре B, старший - в регистре A.

В случае выполнения операции деления целое от деления помещается в аккумулятор A, остаток от деления - в регистр B.

Битовый процессор является частью архитектуры МК и его можно рассматривать как независимый процессор побитовой обработки. Битовый процессор выполняет набор команд, имеет свое побитово-адресуемое ОЗУ и свой ввод-вывод.

Команды, оперирующие с битами, обеспечивают прямую адресацию 128 битов (0-127) в шестнадцати ячейках внутреннего ОЗУ (ячейки с адресами 20H-2FH) и прямую побитовую адресацию регистров специального назначения, адреса которых кратны восьми.

Каждый из отдельно адресуемых битов может быть установлен в "1", сброшен в "0", инвертирован, проверен. Могут быть реализованы переходы: если бит установлен; если бит не установлен; переход, если бит установлен, со сбросом этого бита; бит может быть перезаписан в (из) разряда переноса. Между любым прямоадресуемым битом и флагом переноса могут быть произведены логические операции "И", "ИЛИ", где результат заносится в разряд флага переноса. Команды побитовой обработки обеспечивают реализацию сложных функций комбинаторной логики и оптимизацию программ пользователя.

Система команд позволяет реализовать логические операции: "И", "ИЛИ", "ИСКЛЮЧАЮЩЕЕ ИЛИ" на регистре-аккумуляторе (А) и байте-источнике. Вторым операндом (байтом-источником) при этом может быть рабочий регистр в выбранном банке рабочих регистров; регистр внутреннего ОЗУ, адресуемый с помощью косвенно-регистровой адресации; прямоадресуемые ячейки внутреннего ОЗУ и регистры специального назначения; непосредственная величина.

Указанные логические операции могут быть реализованы на любом прямоадресуемом регистре внутреннего ОЗУ или регистре специального назначения с использованием в качестве второго операнда содержимого аккумлятора А или непосредственных данных.

Существуют логические операции, которые выполняются только на аккумуляторе: сброс и инвертирование всех восьми разрядов А; циклический сдвиг влево и вправо; циклический сдвиг влево и вправо с учетом флага переноса; обмен местами старшей и младшей тетрад (ниблов) внутри аккумулятора.

Таблицы символов (кодов), зашитые в ПЗУ программы могут быть выбраны с помощью команд передачи данных с использованием косвенной адресации. Байт константы может быть передан в аккумулятор из ячейки памяти программ, адресуемой суммой базового регистра (PC или DPTR) и индексного регистра (содержимого А). Это обеспечивает, например, удобное средство реализации алгоритма преобразования кода ASCII в семисегментный код.

Любая ячейка 256-байтового блока внешнего ОЗУ данных может быть выбрана с использованием косвенно-регистровой адресации через регистры указатели R0 или R1 (выбранного банка рабочих регистров).

Ячейка внутри адресного пространства 64 Кбайт внешнего ОЗУ также может быть выбрана с использованием косвенно-регистровой адресации через регистр-указатель данных DPTR.

Команды передачи между прямоадресуемыми регистрами позволяют заносить величину из порта в ячейку внутреннего ОЗУ без использования рабочих регистров или аккумулятора.

В логическом процессоре любой прямоадресуемый бит может быть помещен в бит переноса и наоборот.

Содержимое аккумулятора может быть обменено с содержимым рабочих регистров (выбранного банка) и с содержимым адресуемых с помощью косвенно-регистровой адресации ячеек внутреннего ОЗУ, а также с содержимым прямо-адресуемых ячеек внутреннего ОЗУ и с содержимым регистров специального назначения.

Младший нибл (разряды 3-0) содержимого аккумулятора, может быть обменен с младшим ниблом содержимого ячеек внутреннего ОЗУ, выбираемых с помощью косвенно-регистровой адресации.

Адресное пространство памяти программ не имеет страничной организации, что позволяет свободно перемещать фрагменты программы внутри адресного пространства, при этом не требуется перезасылка (изменение) номера страницы.

Перемещение отдельных фрагментов программы обеспечивает возможность использования перемещаемых программных модулей различными программами.

Команды 16-разрядных переходов и вызовов подпрограмм позволяют осуществлять переход в любую точку адресного пространства памяти программ объемом 64 Кбайт.

Команды 11-разрядных переходов и вызовов подпрограмм обеспечивают переходы внутри программного модуля емкостью 2 Кбайт. В системе команд имеются команды условных и безусловных переходов относительно начального адреса следующей программы в пределах от (PC)-128 до (ЗС)+127. Команды проверки отдельных разрядов позволяют осуществлять условные переходы по состоянию "0" или "1" прямоадресуемых битов. Команды проверки содержимого аккумулятора (на ноль/не ноль) позволяют осуществлять условные переходы по содержимому А.

Косвенно-регистровые переходы в системе команд обеспечивают ветвление относительно базового регистра (содержимого DPTR или PC) со смещением, находящимся в аккумуляторе А.

В заключении следует отметить, что большинство ассемблеров допускают обобщенную мнемонику JMP для команд безусловного перехода и CALL для команд вызова

подпрограмм. Конкретный тип команды определяется ассемблером исходя из степени "длины" перехода или вызова.

В качестве операндов команд ИМС может использовать: отдельные биты, 4-х битные цифры, байты и двухбайтовые слова. Всего ИМС выполняет 13 типов команд (рисунок 9).

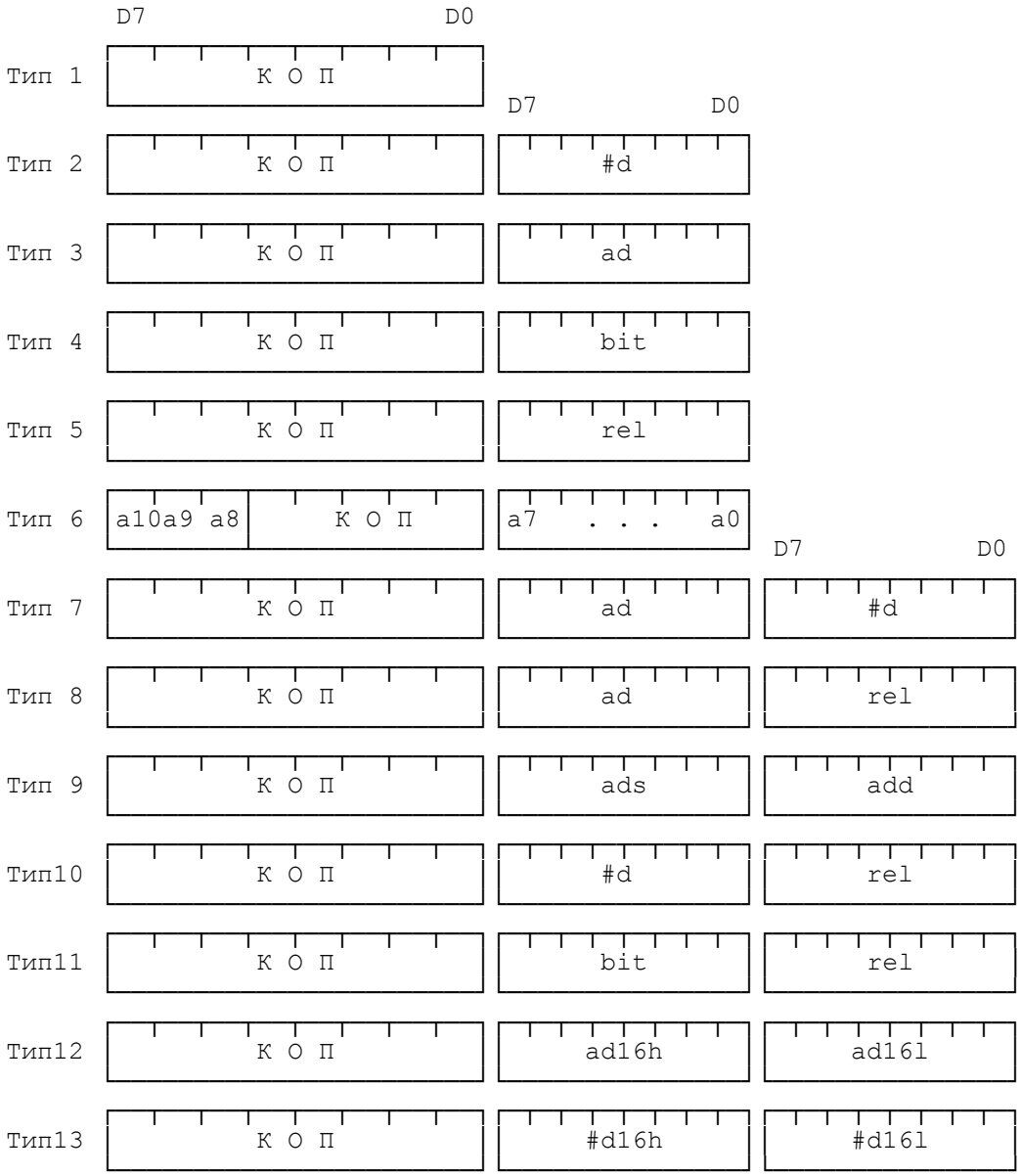


Рисунок 9 - Типы команд ИМС

Как следует из рисунка 9, первый байт команды всегда содержит код операции (КОП), а второй и третий байты (если они присутствуют в команде) содержат адреса операндов или непосредственные значения операндов.

Наиболее существенным является возможность адресовать отдельные биты в резидентной памяти данных. Кроме того, выше уже отмечалось, что отдельные регистры из блока регистров специальных функций так же допускают адресацию отдельных бит.

Карты адресов отдельных бит в резидентной памяти данных и в блоке регистров специальных функций приведены на рисунках 10 и 11.

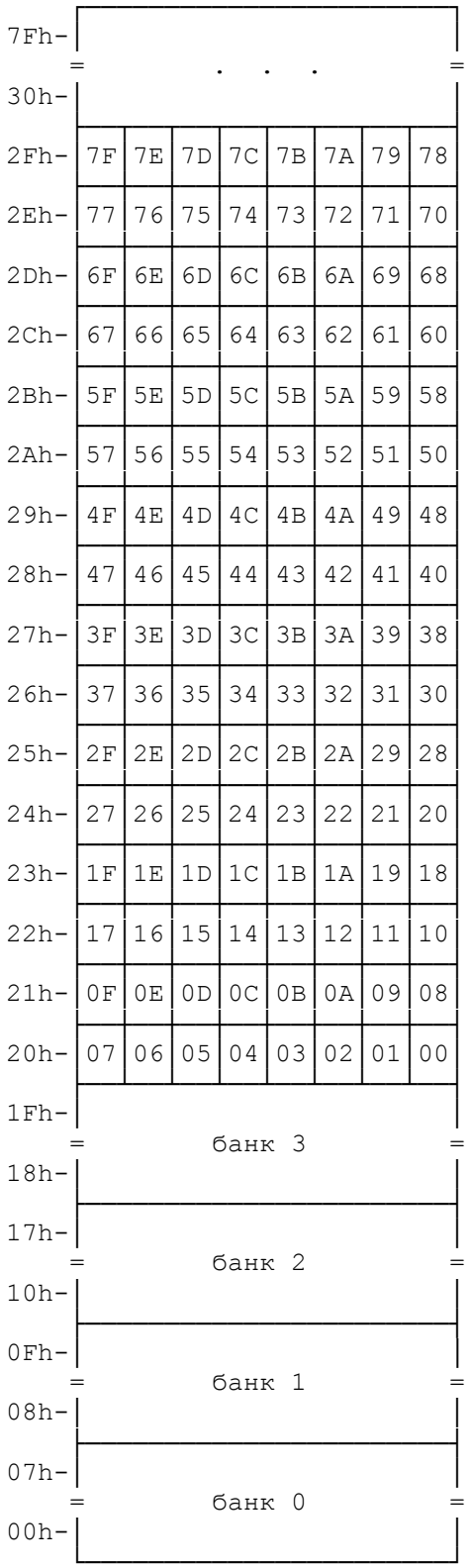


Рисунок 10 - Карта адресуемых бит в резидентной памяти данных

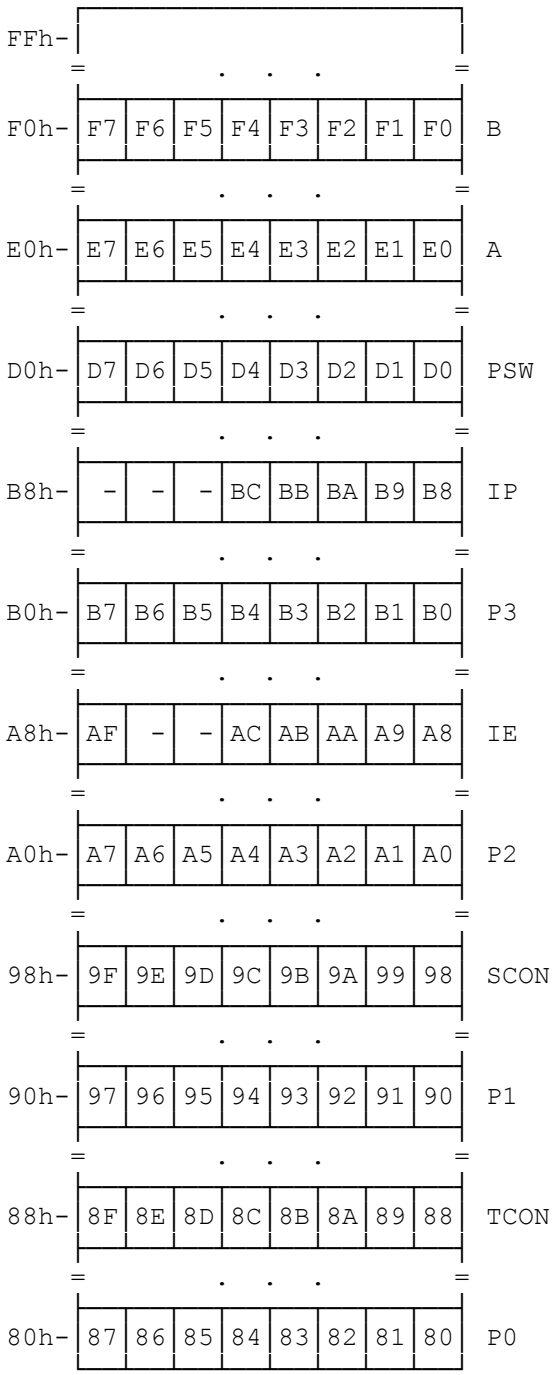


Рисунок 11 - Карта адресуемых бит в блоке регистров специальных функций.

Все множество команд ИМС можно разбить на 5 функциональных групп:

- команды пересылки данных;
- команды арифметических операций;

- команды логических операций;

- команды операций над битами;

- команды передачи управления.

При рассмотрении всех команд будут использоваться следующие обозначения:

Rn (n=0,1,...,7) - регистр общего назначения в выбранном банке регистров;

@Ri (i=0,1) - регистр общего назначения в выбранном банке регистров, используемый в качестве регистра косвенного адреса;

ad - адрес прямоадресуемого байта;

ads - адрес прямоадресуемого байта-источника;

add - адрес прямоадресуемого байта-получателя;

ad11 - 11-разрядный абсолютный адрес перехода;

ad16 - 16-разрядный абсолютный адрес перехода;

rel - относительный адрес перехода;

#d - непосредственный операнд;

#d16 - непосредственный операнд (2 байта);

bit - адрес прямоадресуемого бита;

/bit - инверсия прямоадресуемого бита;

A - аккумулятор;

PC - счетчик команд;

DPTR - регистр указатель данных;

( ) - содержимое ячейки памяти или регистра.

9.1 Группа команд пересылки данных

Данная группа команд представлена 28 командами, краткое описание которых приведено в таблице 19, где так же указаны тип команды (Т), длина команды в байтах (В) и время выполнения команды в машинных циклах (С).

Таблица 19 - Команды пересылки данных

Мнемокод	КОП	Т	В	С	Описание
1	2	3	4	5	6
MOV A,Rn	11101rrr	1	1	1	(A) <-- (Rn)
MOV A,ad	11100101	3	2	1	(A) <-- (ad)
MOV A,@Ri	1110011i	1	1	1	(A) <-- ((Ri))
MOV A,#d	01110100	2	2	1	(A) <-- #d
MOV Rn,A	11111rrr	1	1	1	(Rn) <-- (A)
MOV Rn,ad	10101rrr	3	2	2	(Rn) <-- (ad)
MOV Rn,#d	01111rrr	2	2	1	(Rn) <-- #d
MOV ad,A	11110101	3	2	1	(ad) <-- (A)

1	2	3	4	5	6
MOV ad,Rn	10001rrr	3	2	2	(ad) <-- (Rn)
MOV add,ads	10000101	9	3	2	(add) <-- (ads)
MOV ad,@Ri	1000011i	3	2	2	(ad) <-- ((Ri))
MOV ad,#d	01110101	7	3	2	(ad) <-- #d
MOV @Ri,A	1111011i	1	1	1	((Ri)) <-- (A)
MOV @Ri,ad	0110011i	3	2	2	((Ri)) <-- (ad)
MOV @Ri,#d	0111011i	2	2	1	((Ri)) <-- #d
MOV DPTR, #d16	10010000	13	3	2	(DPTR) <-- #d16
MOVC A,@A+ DPTR	10010011	1	1	2	(A) <-- ((A)+(DPTR))
MOVC A,@A+PC	10000011	1	1	2	(PC) <-- (PC)+1) (A) <-- ((A)+(PC))
MOVX A,@Ri	1110001i	1	1	2	(A) <-- ((Ri))
MOVX A,@DPTR	11100000	1	1	2	(A) <-- ((DPTR))
MOVX @Ri,A	1111001i	1	1	2	((Ri)) <-- (A)
MOVX @DPTR,A	11110000	1	1	2	((DPTR)) <-- (A)
PUSH ad	11000000	3	2	2	(SP) <-- (SP)+1 ((SP)) <-- (ad)
POP ad	11010000	3	2	2	(ad) <-- ((SP)) (SP) <-- (SP)-1
XCH A,Rn	11001rrr	1	1	1	(A) <-> (Rn)
XCH A,ad	11000101	3	2	1	(A) <-> (ad)
XCH A,@Ri	1100011i	1	1	1	(A) <-> ((Ri))
XCHD A,@Ri	1101011i	1	1	1	(A0-3) <-> ((Ri0-3))

По команде MOV выполняется пересылка данных из второго операнда в первый. Команда MOV не имеет доступа ни к внешней памяти данных, ни к памяти программ. Для этих целей служат команды MOVX и MOVC соответственно. Команда MOVX обеспечивает чтение/запись байт из внешней памяти данных, а команды MOVC - чтение байт из памяти программ.

По команде XCH выполняется обмен байтами между аккумулятором и ячейкой РПД, а по команде XCHD - обмен младшими тетрадами (битами 0-3).

Если сравнить группы команд пересылки данных рассматриваемой ИМС и ИМС семейства BE48, то можно выделить два существенных отличия. Во-первых, в рассматриваемой ИМС "исчезли" все команды работы со специальными регистрами: PSW, таймером, портами ввода/вывода. Теперь доступ к ним, как и к другим регистрам специальных функций осуществляется путем задания соответствующего прямого адреса, т.е. эти команды упрятаны в команды типа 3. Например, чтение PSW в аккумулятор может быть выполнено командой MOV A, PSW, которая преобразуется ассемблером к виду MOV A,0D0h (E5 D0), где E5 - код операции, а D0 - операнд (адрес PSW).

Здесь следует отметить, что большинство ассемблеров допускают символические имена для регистров специальных функций, а отдельные биты этих регистров (конечно, если выбранный регистр допускает адресацию отдельных бит) могут адресоваться путем

указания имени регистра и номера бита через точку. Например, к нулевому биту аккумулятора можно обратиться по имени ACC.0. Это означает, что в ИМС аккумулятор имеет два различных имени в зависимости от способа адресации: A - при неявной адресации (например, MOV A,R0) и ACC - при использовании прямого адреса. Первый способ предпочтительнее, однако не всегда возможный.

Вторым существенным отличием является появление команд записи данных в стек PUSH и их чтения из стека POP. Размер стека теперь ограничен лишь размером резидентной памяти данных.

9.2 Группа команд арифметических операций

Данная группа команд состоит из 24 команд, краткое описание которых приведено в таблице 20.

Таблица 20 - Команды арифметических операций

Мнемокод	КОП	Т	В	С	Описание
1	2	3	4	5	6
ADD A,Rn	00101rrr	1	1	1	(A) <-- (A)+(Rn)
ADD A,ad	00100101	3	2	1	(A) <-- (A)+(ad)
ADD A,@Ri	0010011i	1	1	1	(A) <-- (A)+((Ri))
ADD A,#d	00100100	2	2	1	(A) <-- (A)+#d
ADDC A,Rn	00111rrr	1	1	1	(A) <-- (A)+(Rn)+(C)
ADDC A,ad	00110101	3	2	1	(A) <-- (A)+(ad)+(C)
ADDC A,@Ri	0011011i	1	1	1	(A) <-- (A)+((Ri))+(C)
ADDC A,#d	00110100	2	2	1	(A) <-- (A)+#d+(C)
DA A	11010100	1	1	1	десятичная коррекция аккумуляторра
SUBB A,Rn	10011rrr	1	1	1	(A) <-- (A)-(Rn)-(C)
SUBB A,ad	10010101	3	2	1	(A) <-- (A)-(ad)-(C)
SUBB A,@Ri	1001011i	1	1	1	(A) <-- (A)-((Ri))-(C)
SUBB A,#d	10010100	2	2	1	(A) <-- (A)-#d-(C)
INC A	00000100	1	1	1	(A) <-- (A)+1
INC Rn	00001rrr	1	1	1	(Rn) <-- (Rn)+1
INC ad	00000101	3	2	1	(ad) <-- (ad)+1
INC @Ri	0000011i	1	1	1	((Ri)) <-- ((Ri))+1
INC DPTR	10100011	1	1	2	(DPTR) <-- (DPTR)+1

1	2	3	4	5	6
DEC A	00010100	1	1	1	(A) <-- (A)-1
DEC Rn	00011rrr	1	1	1	(Rn) <-- (Rn)-1
DEC ad	00010101	3	2	1	(ad) <-- (ad)-1
DEC @Ri	0001011i	1	1	1	((Ri)) <-- ((Ri))-1
MUL AB	10100100	1	1	4	(B)(A) <-- (A)*(B)
DIV AB	10000100	1	1	4	(A).(B) <-- (A)/(B)

Как следует из таблицы 20, ИМС выполняет достаточно широкий набор команд для организации обработки целочисленных данных, включая команды умножения и деления. По результату выполнения команд ADD, ADDC, SUBB, MUL и DIV устанавливаются флаги PSW, структура которого приведена на рисунке 12.

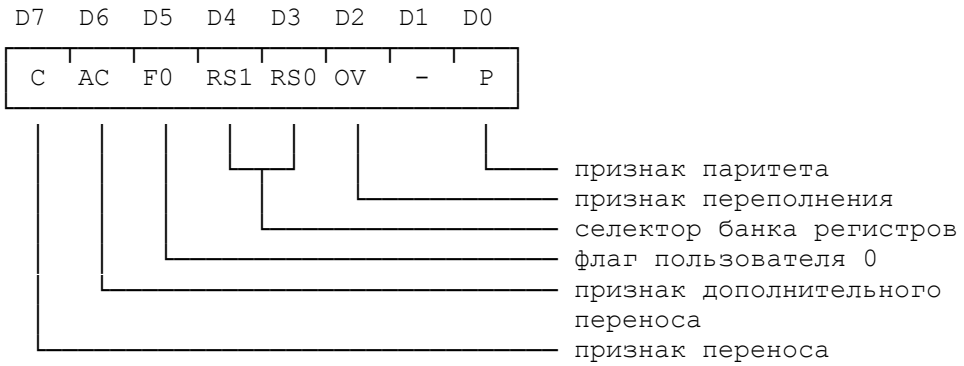


Рисунок 12 - Формат PSW

Флаг C устанавливается при переносе из разряда D7, т.е. когда результат не помещается в 8 разрядов.

Флаг AC устанавливается при переносе из разряда D3 в командах сложения и вычитания и служит для реализации десятичной арифметики. Этот признак используется командой DA A.

Флаг OV устанавливается при переносе из разряда D6, т.е. когда результат не помещается в 7 разрядов и восьмой разряд не может быть интерпретирован как знаковый. Этот признак служит для организации обработки чисел со знаком.

Флаг P устанавливается и сбрасывается аппаратно. Если число единичных бит в аккумуляторе нечетно, то P=1, иначе, P=0.

9.3 Группа команд логических операций

Данная группа команд состоит из 25 команд, краткое описание которых приведено в таблице 21.

Таблица 21 - Команды логических операций

Мнемокод	КОП	Т	В	С	Описание
ANL A,Rn	01011rrr	1	1	1	(A) <-- (A)/\(Rn)
ANL A,ad	01010101	3	2	1	(A) <-- (A)/\(ad)
ANL A,@Ri	0101011i	1	1	1	(A) <-- (A)/\((Ri))
ANL A,#d	01010100	2	2	1	(A) <-- (A)/\#d
ANL ad,A	01010010	3	2	1	(ad) <-- (ad)/\(A)
ANL ad,#d	01010011	7	3	2	(ad) <-- (ad)/\#d
ORL A,Rn	01001rrr	1	1	1	(A) <-- (A)\/(Rn)
ORL A,ad	01000101	3	2	1	(A) <-- (A)\/(ad)
ORL A,@Ri	0100011i	1	1	1	(A) <-- (A)\/((Ri))
ORL A,#d	01000100	2	2	1	(A) <-- (A)\/#d
ORL ad,A	01000010	3	2	1	(ad) <-- (ad)\/A
ORL ad,#d	01000011	7	3	2	(ad) <-- (ad)\/#d
XRL A,Rn	01101rrr	1	1	1	(A) <-- (A) (+) (Rn)
XRL A,ad	01100101	3	2	1	(A) <-- (A) (+) (ad)
XRL A,@Ri	0110011i	1	1	1	(A) <-- (A) (+) ((Ri))
XRL A,#d	01100100	2	2	1	(A) <-- (A) (+) #d
XRL ad,A	01100010	3	2	1	(ad) <-- (ad) (+) A
XRL ad,#d	01100011	7	3	2	(ad) <-- (ad) (+) #d
CLR A	11100100	1	1	1	(A) <-- 0
CPL A	11110100	1	1	1	(A) <-- NOT (A)
SWAP A	11000100	1	1	1	(A0-3) <-> (A4-7)
RL A	00100011	1	1	1	циклический сдвиг влево
RLC A	00110011	1	1	1	сдвиг влево через перенос
RR A	00000011	1	1	1	циклический сдвиг вправо
RRC A	00010011	1	1	1	сдвиг вправо через перенос

Как следует из таблицы 3, данная группа команд позволяет выполнять операции над байтами: логическое И (/\ ), логическое ИЛИ (\/), исключающее ИЛИ ((+)), инверсию (NOT), сброс в нулевое значение и сдвиг. Команды оперирующие отдельными битами описаны далее.

9.4 Группа команд операций над битами.

Данная группа команд состоит из 12 команд, краткое описание которых приведено в таблице 22.

Таблица 22 - Команды операций над битами

Мнемокод	КОП	Т	В	С	Описание
1	2	3	4	5	6
CLR C	11000011	1	1	1	(C) <-- 0
CLR bit	11000010	4	2	1	(bit) <-- 0
SETB C	11010011	1	1	1	(C) <-- 1
SETB bit	11010010	4	2	1	(bit) <-- 1
CPL C	10110011	1	1	1	(C) <-- NOT (C)
CPL bit	10110010	4	2	1	(bit) <-- NOT (bit)
ANL C,bit	10000010	4	2	2	(C) <-- (C)/\(bit)
ANL C,/bit	10110000	4	2	2	(C) <-- (C)/\NOT(bit)

1	2	3	4	5	6
ORL C,bit	01110010	4	2	2	(C) <-- (C)\/(bit)
ORL C,/bit	10100000	4	2	2	(C) <-- (C)\/NOT(bit)
MOV C,bit	10100010	4	2	1	(C) <-- (bit)
MOV bit,C	10010010	4	2	2	(bit) <-- (C)

Как следует из таблицы 22, данная группа команд позволяет выполнять операции над отдельными битами: сброс, установку, инверсию бита, а так же логическое И (/\ ) и логическое ИЛИ (\/).

В качестве "логического" аккумулятора, участвующего во всех операциях с двумя операндами, выступает признак переноса C (разряд D7 PSW). В качестве операндов могут использоваться 128 бит из редентной памяти данных и регистры специальных функций допускающие адресацию отдельных бит.

9.5 Группа команд передачи управления

Группа команд передачи управления представлена командами безусловного и условного переходов, командами вызова подпрограмм и командами возврата из подпрограмм. Краткое описание команд группы приведено в таблице 23.

Таблица 23 - Команды передачи управления

Мнемокод	КОП	Т	В	С	Описание
1	2	3	4	5	6
LJMP ad16	00000010	12	3	2	длинный безусловный переход по всей памяти
AJMP ad11	a10a9a800001	6	2	2	безусловный переход в пределах страницы 2 Кбайт
SJMP rel	10000000	5	2	2	безусловный переход в пределах страницы 256 байт
JMP @A+DPTR	01110011	1	1	2	безусловный переход по косвенному адресу
JZ rel	01100000	5	2	2	переход если нуль
JNZ rel	01110000	5	2	2	переход если не нуль
JC rel	01000000	5	2	2	переход если бит переноса установлен
JNC rel	01010000	5	2	2	переход если бит переноса не установлен
JB bit,rel	00100000	11	3	2	переход если бит установлен
JNB bit,rel	00110000	11	3	2	переход если бит не установлен
JBC bit,rel	00010000	11	3	2	переход если бит установлен со сбросом бита
DJNZ Rn,rel	11011rrr	5	2	2	команда цикла
DJNZ ad,rel	11010101	8	3	2	команда цикла
CJNE A,ad,rel	10110101	8	3	2	сравнение аккумулятора с байтом и переход если не равно
CJNE A,#d,rel	10110100	10	3	2	сравнение аккумулятора с константой и переход если не равно

1	2	3	4	5	6
CJNE Rn,#d,rel	10111rrr	10	3	2	сравнение регистра с константой и переход если не равно
CJNE @Ri,#d,rel	1011011i	10	3	2	сравнение байта памяти с константой и переход если не равно
LCALL ad16	00010010	12	3	2	длинный вызо подпрограммы во всей памяти
ACALL ad11	a10a9a810001	6	2	2	вызов подпрограммы в пределах страницы 2 Кбайт
RET	00100010	1	1	2	возврат подпрограммы
RETI	00110010	1	1	2	возврат подпрограммы обработки прерывания
NOP	00000000	1	1	1	пустая операция

Команда безусловного перехода LJMP (L-long-длинный) осуществляет переход по абсолютному 16-битному адресу, указанному в теле команды, т.е. команда обеспечивает переход в любую точку памяти программ. Действие команды AJMP (A-absolute-абсолютный) аналогично команде LJMP, однако в теле команды указаны лишь 11 младших разрядов адреса. Поэтому переход осуществляется в пределах страницы размером в 2 Кбайт, при этом надо иметь в виду, что сначала содержимое счетчика команд увеличивается на 2 и только потом заменяются младшие 11 разрядов адреса.

В отличие от предыдущих в команде SJMP (S-short-короткий) указан не абсолютный, а относительный адрес перехода. Величина смещения rel рассматривается как число со знаком, а следовательно переход возможен в пределах -128...+127 байт относительно адреса команды следующей за командой SJMP.

Команда косвенного перехода JMP @A+DPTR позволяет вычислять адрес перехода в процессе выполнения самой программы.

Команды условного перехода позволяют проверять следующие условия:

JZ - аккумулятор содержит нулевое значение;

JNZ - аккумулятор содержит не нулевое значение;

JC - бит переноса C установлен;

JNC - бит переноса C не установлен;

JB - прямоадресуемый бит равен единице;

JNB - прямоадресуемый бит равен нулю;

JBC - прямоадресуемый бит равен единице и сбрасывается в нулевое значение при выполнении команды.

В отличие от ИМС семейства BE48, все команды условного перехода содержат короткий относительный адрес, т.е. переход может осуществляться в пределах -128...+127 байт относительно следующей команды.

Команда DJNZ предназначена для организации программных циклов.

Регистр Rn или байт по адресу ad, указанные в теле команды, содержат счетчик повторений цикла, а смещение rel – относительный адрес перехода к началу цикла. При выполнении команды содержимое счетчика уменьшается на единицу и проверяется на нуль. Если значение счетчика не равно нулю, то осуществляется переход на начало цикла, иначе выполняется следующая команда.

Команда CJNZ удобна для реализации процедур ожидания внешних событий. В теле команды указаны "координаты" двух байт и относительный адрес перехода rel. В качестве двух байт могут быть, например, использованы значения аккумулятора и прямоадресуемого байта или косвенно адресуемого байта и константы. При выполнении команды значения указанных двух байт сравниваются и, в случае, если эти значения не равны, осуществляется переход. Например, команда

WAIT: CJNE A,P0,WAIT будет выполняться до тех пор, пока значения на линиях порта P0 не совпадут с аккумулятором.

Действие команд вызова процедур полностью аналогично командам безусловного перехода. Единственное отличие состоит в том, что эти команды сохраняют в стеке адрес возврата.

Команда возврата из подпрограммы RET восстанавливает из стека значение счетчика команд, а команда возврата из процедуры обработки прерывания RETI кроме того разрешает прерывание обслуженного уровня. Команды RET и RETI не различают, какой командой LCALL или ACALL подпрограмма была вызвана, т.к. и в том и в другом случае в стеке сохраняется полный 16 разрядный адрес возврата.

Полный перечень команд приведен ниже.

10 Описание системы команд микроконтроллера

Система команд ИМС предоставляет большие возможности обработки данных, обеспечивает реализацию логических, арифметических операций, а также управление в режиме реального времени. Реализована побитовая, потетрадная (4 бита), побайтовая (8 бит) и 16-разрядная обработка данных.

Основные функциональные блоки ИМС: ПЗУ, ОЗУ, регистры специального назначения, АЛУ и внешние шины имеют байтовую организацию. Двухбайтовые данные используются только регистром-указателем (DPTR) и счетчиком команд (PC). Следует отметить, что регистр-указатель данных может быть использован как двухбайтовый регистр DPTR или как два однобайтовых регистра специального назначения DPH и DPL. Счетчик команд всегда используется как двухбайтовый регистр.

Набор команд ИМС имеет 42 мнемонических обозначения команд для конкретизации 33 функций этой системы.

Синтаксис большинства команд ассемблерного языка состоит из мнемонического обозначения функции, вслед за которым идут операнды, указывающие методы адресации и типы данных. Различные типы данных или режимы адресации определяются установленными операндами, а не изменениями мнемонических обозначений.

Систему команд условно можно разбить на пять групп:

- Арифметические команды
- Логические команды
- Команды передачи данных
- Команды битового процессора
- Команды ветвления и передачи управления

Существуют следующие типы адресации операндов-источников:

- Регистровая адресация
- Прямая адресация
- Косвенно-регистровая адресация
- Непосредственная адресация
- Косвенно-регистровая адресация по сумме базового и индексного регистров.

Обозначения и символы, принятые при описании системы команд, приведены в таблице 24.

Таблица 24 - Обозначения и символы, используемые в системе команд.

Обозначение, символ	Назначение
1	2
A	Аккумулятор
Rn	Регистры текущего выбранного банка регистров
r	Номер загружаемого регистра, указанного в команде
direct	Прямо адресуемый 8-битовый внутренний адрес ячейка данных, который может быть ячейкой внутреннего ОЗУ данных (0-127) или SFR (128-255)
@Rr	Косвенно адресуемая 8-битовая ячейка внутреннего ОЗУ данных
data8	8-битовое непосредственное данное, входящее в КОП
dataH	Старшие биты (15-8) непосредственных 16-битовых данных
dataL	Младшие биты (7-0) непосредственных 16-битовых данных
addr11	11-битовый адрес назначения
addrL	Младшие биты адреса назначения
disp8	8-битовый байт смещения со знаком

1	2
bit	Бит с прямой адресацией, адрес которого содержит КОП, находящийся во внутреннем ОЗУ данных или SFR
a15, a14...a0	Биты адреса назначения
(X)	Содержимое элемента X
((X))	Содержимое по адресу, хранящемуся в элементе X
(X)[M]	Разряд M элемента X
Операции: + - * / AND OR XOR /X	Операции: сложения вычитания умножения деления логического умножения (операция И) логического сложения (операция ИЛИ) сложения по модулю 2 (исключающее ИЛИ) инверсия элемента X

Мнемонические обозначения функций однозначно связаны с конкретными комбинациями способов адресации и типами данных. Всего в системе команд возможно 111 таких сочетаний. В таблице 25 приведен перечень команд, упорядоченных по алфавиту.

Таблица 25 - Перечень команд, упорядоченных по алфавиту.

Мнемоника	Функция	Флаги
1	2	3
Команда ACALL <addr 11>	Абсолютный вызов подпрограммы	
Команда ADD A, <байт-источник>	Сложение	AC, C, OV
Команда ADDC A, <байт-источник>	Сложение с переносом	AC, C, OV
Команда AJMP <addr 11>	Абсолютный переход	
Команда ANL <байт-назначения>, <байт-источника>	Логическое "И"	
Команда ANL C, <байт-источника>	Логическое "И" для переменных-битов	C
Команда CJNE <байт-назначения>, <байт-источник>, <смещение>	Сравнение и переход, если не равно	C
Команда CLR A	Сброс аккумулятора	
Команда CLR <bit>	Сброс бита	C, bit
Команда CPL A	Инверсия аккумулятора	
Команда CPL <bit>	Инверсия бита	C, bit
Команда DA A	Десятичная коррекция аккумулятора для сложения	AC, C
Команда DEC <байт>	Декремент	
Команда DIV AB	Деление	C, OV

1	2	3
Команда DJNZ <байт>, <смещение>	Декремент и переход, если не равно нулю	
Команда INC <байт>	Инкремент	
Команда INC DPTR	Инкремент указателя данных	
Команда JB <bit>, <re18>	Переход, если бит установлен	
Команда JBC <bit>, <re18>	Переход, если бит установлен и сброс этого бита	
Команда JC <re18>	Переход, если перенос установлен	
Команда JMP @A+DPTR	Косвенный переход	
Команда JNB <bit>, <re18>	Переход, если бит не установлен	
Команда JNC <re18>	Переход, если перенос не установлен	
Команда JNZ <re18>	Переход, если содержимое аккумулятора не равно нулю	
Команда JZ <re18>	Переход, если содержимое аккумулятора равно 0	
Команда LCALL <addr16>	Длинный вызов	
Команда LJMP <addr16>	Длинный переход	
Команда MOV <байт-назначения>, <байт-источника>	Переслать переменную-байт	
Команда MOV <бит-назначения>, <бит-источника>	Переслать бит данных	C
Команда MOV DPTR,#data16	Загрузить указатель данных 16-битовой константой	
Команда MOVC A, @A+DPTR	Переслать байт из памяти программ	
Команда MOVX <байт приемника>, <байт источника>	Переслать во внешнюю память (из внешней памяти) данных	
Команда MUL AB	Умножение	C, OV
Команда NOP	Нет операции	PC
Команда ORL <байт-назначения>, <байт-источника>	Логическое "ИЛИ" для переменных-байтов	
Команда ORL C, <бит источника>	Логическое "ИЛИ" для переменных-битов	C
Команда POP <direct>	Чтение из стека	
Команда PUSH <direct>	Запись в стек	
Команда RET	Возврат из подпрограммы	
Команда RETI	Возврат из прерывания	
Команда RL A	Сдвиг содержимого аккумулятора влево	
Команда RLC A	Сдвиг содержимого аккумулятора влево через флаг переноса	
Команда RR A	Сдвиг содержимого аккумулятора вправо	
Команда RRC A	Сдвиг содержимого аккумулятора вправо через флаг переноса	C

1	2	3
Команда SETB <bit>	Установить бит	C
Команда SJMP <метка>	Короткий переход	
Команда SUBB A, <байт источника>	Вычитание с заемом	AC, C, OV
Команда SWAP A	Обмен тетрадами внутри аккумулятора	
Команда XCH A, <байт>	Обмен содержимого аккумулятора с переменной-байтом	
Команда XCHD A,@Ri ; где i=0,1	Обмен тетрадой	
Команда XRL <байт-назначения>, <байт-источника>	Логическое "ИСКЛЮЧАЮЩЕЕ ИЛИ" для переменных-байтов	

Команда ACALL <addr 11>

Команда "абсолютный вызов подпрограммы" вызывает безусловно подпрограмму, размещенную по указанному адресу. При этом счетчик команд увеличивается на 2 для получения адреса следующей команды, после чего полученное 16-битовое значение PC помещается в стек (сначала следует младший байт), и содержимое указателя стека также увеличивается на два. Адрес перехода получается с помощью конкатенации старших бит увеличенного содержимого счетчика команд, битов старшего байта команды и младшего байта команды.

Ассемблер: ACALL <метка>

Код: A10 A9 A8 1 0 0 0 1 A7 A6 A5 A4 A3 A2 A1 A0

Время; 2 цикла

Алгоритм:
(PC) : = (PC) + 2
(SP) : = (SP) + 1
((SP)) : = (PC [7 - 0])
(SP) : = (SP) + 1
((SP)) : = (PC [15 - 8])
(PC [10 - 0]) : = A10A9A8 II A7A6A5A4A3A2A1A0,
где II - знак конкатенации (сцепление)

Пример:
```
;ДО ВЫПОЛНЕНИЯ КОМАНДЫ ACALL
;(SP)=07H
;метка MT1 соответствует адресу: 0345H,
;т.е. (PC)=0345H
ACALL MT1 ;расположена по адресу 028DH, т.е.
;(3C)=028DH
;ПОСЛЕ ВЫПОЛНЕНИЯ КОМАНДЫ
;(SP)=09H, (PC)=0345H,
;ОЗУ [08]=8FH, ОЗУ [09]=02H.
```

Команда ADD A, <байт-источник>

Эта команда ("сложение") складывает содержимое аккумулятора A с содержимым байта-источника, оставляя результат в аккумуляторе. При появлении переносов из разрядов 7 и 3, устанавливаются флаги переноса (C) и дополнительного переноса (AC) соответственно, в противном случае эти флаги сбрасываются. При сложении целых чисел без знака флаг переноса "C" указывает на появление переполнения. Флаг переполнения (OV) устанавливается, если есть перенос из бита 6 и нет переноса из бита 7, или есть перенос из бита 7 и нет - из бита 6, в противном случае флаг OV сбрасывается. При сложении целых чисел со знаком лаг OV указывает на отрицательную величину, полученную при суммировании операндов или на положительную сумму для двух отрицательных операндов.

Для команды сложения разрешены следующие режимы адресации байта-источника:

1. регистровый
2. косвенно-регистровый
3. прямой
4. непосредственный

Рассмотрим их.

1.

Ассемблер: ADD A,Rn ; где n=0-7

Код: 0 0 1 0 1 rrr , где rrr=000-111

Время: 1 цикл

Алгоритм: (A) : = (A) + (Rn), где n=0-7
C : = X, OV : =X, AC : =X, где X=(0 или 1)

Пример:
```
        ;(A)=C3H, (R6)=AAH
ADD  A,R6  ;(A)=6DH, (R6)=AAH
        ;(AC)=0, (C)=1, (OV)=1
```

2.

Ассемблер: ADD A,@Ri ; где i=0,1

Код: 0 0 1 0 0 1 1 i , где i=0,1

Время: 1 цикл

Алгоритм: (A) : +(A) = ((Ri)), где i=0,1
C : = X, OV : = X, AC : = X, где X=(0 или1)

Пример:
```
        ;(A)=95H, (R1)=31H, (ОЗУ [31])=4CH
ADD  A,@R1  ;(A)=E1H, (ОЗУ [31])=4CH,
        ;(C)=0, (AC)=1, (OV)=0
```

3.

Ассемблер: ADD A,<direct>

Код: 0 0 1 0 0 1 0 1 direct address

Время: 1 цикл

Алгоритм: (A) : =(A)+(direct)
C : =X, OV : =X, AC : =X, где X=(0 или 1)

Пример:
```
;(A)=77H, (ОЗУ [90])=FFH
ADD  A,90H ;(A)=76H, (ОЗУ [90})=FFH,
;(C)=1, (OV)=0, (AC)=1
```

4.

Ассемблер: ADD A, <#data>

Код: 0 0 1 0 0 1 0 0 #data

Время: 1 цикл

Алгоритм: (A) : = (A)+#data
C : =X, OV : =X, AC : = X, где X=(0 или 1)

Пример:
```
;(A)=09H
ADD  A,#0D3H ;(A)=DCH,
;(C)=0, (OV)=0, (AC)=0
```

Команда ADDC A, <байт-источник>

Эта команда ("сложение с переносом") одновременно складывает содержимое байта-источника, флаг переноса и содержимое аккумулятора A, оставляя результат в аккумуляторе. При этом флаги переноса и дополнительного переноса устанавливаются, если есть перенос из бита 7 или бита 3, и сбрасываются в противном случае. При сложении целых чисел без знака флаг переноса указывает на переполнение. Флаг переполнения (OV) устанавливается, если имеется перенос бита 6 и нет переноса из бита 7 или есть перенос из бита 7 и нет - из бита 6, в противном случае OV сбрасывается. При сложении целых чисел со знаком OV указывает на отрицательную величину, полученную при суммировании двух положительных операндов или на положительную сумму от двух отрицательных операндов.

Для этой команды разрешены следующие режимы адресации байта-источника:

1. регистровый
2. косвенно-регистровый
3. прямой
4. непосредственный

Рассмотрим их.

1.

Ассемблер: ADDC A,Rn ; где n=0-7

Код: 0 0 1 1 1 rrr , где rrr=000-111

Время: 1 цикл

Алгоритм: (A) : = (A) + (C) = (Rn)
(C) : =X, (AC) : =X, (OV) : =X, где X=(0 или 1)

Пример:
```
;(A)=B2H, (R3)=99,
;(C)=1
ADDC A,R3  ;(A)=4CH, (R3)=99
;(C)=1, (AC)=0, (OV)=1
```

2.

Ассемблер: ADDC A,@Ri ; где i=0,1

Код: 0 0 1 1 0 1 1 i , где i=0,1

Время: 1 цикл

Алгоритм: (A) : =(A) + (C) + ((Ri))
(C) : =X, (AC) : =X, (OV) : = X, где X=(0 или1)

Пример:
```
;(A)=D5H, (R0)=3AH,
;(ОЗУ [3A])=1AH, (C)=1
ADDC A,@R0  ;(A)=F0H, (ОЗУ [3A])=1AH,
;(C)=0, (AC)=1, (OV)=0
```

3.

Ассемблер: ADDC A,<direct>

Код: 0 0 1 1 0 1 0 1 direct address

Время: 1 цикл

Алгоритм: (A) : =(A)+(C)+(direct)
(C) : =X, (AC) : =X, (OV) : =X, где X=(0 или 1)

Пример:
```
;(A)=11H, (ОЗУ [80])=DFH, (C)=1
ADDC A,80H ;(A)=F1H, (C)=0, (AC)=1 (OV)=0
```

4.

Ассемблер: ADDC A, <#data>

Код: 0 0 1 1 0 1 0 0 #data

Время: 1 цикл

Алгоритм: (A) : = (A)+(C)+ #data
(C) : =X, (AC) : = X, (OV) : =X, где X=(0 или 1)

Пример:
```
;(A)=55H, (C)=0
ADDC  A,#55H ;(A)=AAH, (C)=0, (AC)=0, (OV)=1
```

Команда AJMP <addr 11>

Команда "абсолютный переход", передает управление по указанному адресу, который получается при конкатенации пяти старших бит счетчика команд PC (после увеличения его на два), 7-5 битов кода операции и второго байта команды. Адрес перехода должен находиться внутри одной страницы объемом 2 Кбайт памяти программы, определяемой пятью старшими битами счетчика команд.

Ассемблер: AJMP <метка>

Код: A10 A9 A8 0 0 0 0 1 A7 A6 A5 A4 A3 A2 A1 A0

Время: 2 цикла

Алгоритм: (PC [15-0]) : = (PC [15-0]) + 2,
(PC [10 - 0]) : = <addr11>

Пример:
```
;(PC)=028FH
;Метке MT2 соответствует адрес 034AH
AJMP  MT2 ;(PC)=034AH
```

Команда ANL <байт-назначения>,<байт-источника>

Команда "логическое "И" для переменных-байтов" выполняет операцию логического "И" над битами указанных переменных и помещает результат в байт-назначения. Эта операция не влияет на состояние флагов.

Для операнда обеспечивают следующие комбинации шести режимов адресации:

- байтом назначения является аккумулятор (A):
  1. регистровый
  2. прямой
  3. косвенно-регистровый
  4. непосредственный
- байтом назначения является прямой адрес (direct):
  5. прямой аккумуляторный
  6. непосредственный (байт-источник равен константе)

Рассмотрим их.

1.

Ассемблер: ANL A,Rn ; где n=0-7

Код: 0 1 0 1 1 rrr , где rrr=000-111

Время: 1 цикл

Алгоритм: (A) : = (A) AND (Rn)

Пример:
```
;(A)=FEH, (R2)=C5H
```

ANL A,R2 ;(A)=C4H, (R2)=C5H

2.

Ассемблер: ANL A,<direct>

Код: 0 1 0 1 0 1 0 1 direct address

Время: 1 цикл

Алгоритм: (A) : =(A) AND (direct)

Пример: ;(A)=A3H, (PSW)=86H
ANL A,PSW ;(A)=82H, (PSW)=86H

3.

Ассемблер: ANL A,@Ri ; где i=0,1

Код: 0 1 0 1 0 1 1 i , где i=0,1

Время: 1 цикл

Алгоритм: (A) : =(A) AND (Ri)

Пример: ;(A)=BCH, (ОЗУ [35])=47H, (R0)=35H,
ANL A,@R0 ;(A)=04H, (ОЗУ [35])=47H

4.

Ассемблер: ANL A, #data

Код: 0 1 0 1 0 1 0 0 #data8

Время: 1 цикл

Алгоритм: (A) : = (A)AND #data

Пример: ;(A)=36H
ANL A,#0DDH ;(A)=14H

5.

Ассемблер: ANL <direct>, A

Код: 0 1 0 1 0 0 1 0 direct address

Время: 1 цикл

Алгоритм: (direct) : = (direct) AND (A)

Пример: ;(A)=55H, (P2)=AAH
ANL P2,A ;(P2)=00H, (A)=55H

6.

Ассемблер: ANL <direct>, #data

Код: 0 1 0 1 0 0 1 1 direct address #data8

Время: 2 цикла

Алгоритм: (direct) : = (direct) AND #data

Пример: ;(P1)=FFH
ANL P1,#73H ;(P1)=73H

Команда ANL C, <байт-источника>

Команда "логическое "И" для переменных-битов", выполняет операцию логического "И" над указанными битами. Если бит-источник равен "0", то происходит сброс флага переноса, в противном случае флаг переноса не изменяет текущего значения. "/" перед операндом в языке ассемблера указывает на то, что в качестве значения используется логическое отрицание адресуемого бита, однако сам бит источника при этом не изменяется. На другие флаги эта команда не влияет.

Для операнда-источника разрешена только прямая адресация к битам.

1.

Ассемблер: ANL C, <bit>

Код: 1 0 0 0 0 0 1 0 bit address

Время: 2 цикла

Алгоритм: (C) : = (C) AND (bit)

Пример: ;(C)=1, P1[0]=0
ANL C,P1.0 ;(C)=0, P1[0]=0

2.

Ассемблер: ANL C,</bit>

Код: 1 0 1 1 0 0 0 0 bit address

Время: 2 цикла

Алгоритм: (C) : =(C) AND (/bit)

Пример: ;(C)=1, (AC)=O
ANL C,/AC ;(C)=1, (AC)=0

Команда CJNE <байт-назначения>,<байт-источник>, <смещение>

Команда "сравнение и переход, если не равно" сравнивает значения первых двух операндов и выполняет ветвление, если операнды не равны. Адрес перехода (ветвления) вычисляется при помощи сложения значения (со знаком), указанного в последнем байте команды, с содержимым счетчика команд после увеличения его на три.

Флаг переноса "С" устанавливается в "1", если значение целого без знака <байта назначения> меньше, чем значение целого без знака <байта источника>, в противном слу-

чае перенос сбрасывается (если значения операндов равны, флаг переноса сбрасывается). Эта команда не оказывает влияния на операнды.

Операнды, стоящие в команде, обеспечивают комбинации четырех режимов адресации:

- если байтом-назначения является аккумулятор:

1. прямой
2. непосредственный

- если байтом-назначения является любая ячейка ОЗУ с косвенно-регистровой или регистровой адресацией:

3. непосредственный к регистровому
4. непосредственный к косвенно-регистровому

Рассмотрим их.

1.

Ассемблер: CJNE A, <direct>, <метка>

Код: 1 0 1 1 0 1 0 1 direct address rel8

Время: 2 цикл

Алгоритм: (PC) : =(PC)+3
если (direct) < (A) то (PC) :=(PC)+<rel8>, C:=0
если (direct) > (A) то (PC) :=(PC)+<rel8>, C:=1

Пример:

```
;(A)=97H, (P2)=F0H, (C)=0
   CJNE  A,P2,MT3
   ...
MT3: CLR   A      ;(A)=97H, (P2)=F0H, (C)=1
             ;Адрес, соответствующий метке
             ;MT3 вычисляется, как
             ;(PC):=(PC)+3+(rel8)
```

2.

Ассемблер: CJNE A, #data, <метка>

Код: 1 0 1 1 0 1 0 0 #data8 rel8

Время: 2 цикла

Алгоритм: (PC) :=(PC)+3,
если #data< (A), то (PC)+<rel8>, C:=0
если #data8>(A), то (PC) :=(PC)+<rel8>, C:=1

Пример:

```
;(A)=FCH, (C)=1
  CJNE  A,#0BFH,MT4
   ...
MT4: INC   A      ;(A)=FDH, (C)=0
             ;(PC):=(PC)+3+(rel8)
```

3.

Ассемблер: CJNE Rn,#data, <метка>; где n=0-7

Код: 1 0 1 1 1 rrr #data8 rel8

Время: 2 цикла

Алгоритм: (PC) : = (PC)+3,
если #data <(Rn), то (PC) :=(PC)+<rel8>, C:=0
если #data8 >(Rn), то (PC)+<rel8>, C:=1

Пример:
```
;(R7)=80H, (C)=0
  CJNE  R7,#81H,MT5
    ...
MT5: NOP          ;(R7)=80H, (C)=1
                  ;(PC):=(PC)+3+(rel8)
```

4.

Ассемблер: CJNE @Ri, #data, <метка>; где i=0,1

Код: 1 01 1 1 0 1 1 i #data8 rel8

Время: 2 цикла

Алгоритм: (PC) :=(PC)+3,
если #data <((Ri)), то (PC)+<rel8>, C:=0
если #data8>((Ri)), то (PC)+<rel8>, C:=1

Пример:
```
;(R0)=41H, (C)=1, (ОЗУ[41])=57H
   CJNE  @R0,#29H,MT6
   ...
MT6: DEC   R0      ;(ОЗУ[41])=57H, (C)=0
                   ;(PC):=(PC)+3+(rel8)
```

Команда CLR A

Команда "сброс аккумулятора" сбрасывает (обнуляет) содержимое аккумулятора A. На флаги команда не влияет.

1.

Ассемблер: CLR A

Код: 1 1 1 0 0 1 0 0

Время: 1 цикл

Алгоритм: (A) : = 0

Пример:
```
          ;(A)=6DH, (C)=0, (AC)=1
CLR   A   ;(A)=00H, (C)=0, (AC)=1
```

Команда CLR <bit>

Команда "сброса бита" сбрасывает указанный бит в нуль. Эта команда работает с флагом переноса "C" или любым битом с прямой адресацией.

1.

Ассемблер: CLR C
Код: 1 1 0 0 0 0 1 1
Время: 1 цикл
Алгоритм: (C) : = 0
Пример:
```
;(C)=1
CLR   C    ;(C)=0
```

2.

Ассемблер: CLR <bit>
Код: 1 1 0 0 0 0 1 0 bit address
Время: 1 цикл
Алгоритм: (bit) : = 0
Пример:
```
;(P1)=5EH (01011110B)
CLR P1.3   ;(P1)=56H (01010110B)
```

Команда CPL A

Команда "инверсия аккумулятора" каждый бит аккумулятора инвертирует (изменяет на противоположный). Биты, содержащие "единицы", после этой команды будут содержать "нули", и наоборот. На флаги эта операция не влияет.

1.

Ассемблер: CPL A
Код: 1 1 1 1 0 1 0 0
Время: 1 цикл
Алгоритм: (A) : = /(A)
Пример:
```
;(A)=65H (01100101B)
CPL A      ;(A)=9AH (10011010B)
```

Команда CPL <bit>

Команда "инверсия бита" (изменяет на противоположное значение) указанный бит. Бит, который был "единицей", изменяется в "нуль" и наоборот. Команда CPL может работать с флагом переноса или с любым прямо адресуемым битом. На другие флаги команда не влияет.

1.

Ассемблер: CPL <bit>
Код: 1 0 1 1 0 0 1 0 bit address
Время: 1 цикл
Алгоритм: (bit) : = /(bit)
Пример:
```
;(P1)=39H (00111001B)
CPL P1.1
CPL P1.3   ;(P1)=33H (00110011B)
```

2.

Ассемблер: CPL C
Код: 1 0 1 1 0 0 1 1
Время: 1 цикл
Алгоритм: (C) : = /(C)
Пример:
```
;(C)=0, (AC)=1, (OV)=0
CPL C      ;(C)=1, (AC)=1, (OV)=0
```

Примечание - Если эта команда используется для изменения информации на выходе порта, значение, используемое как исходные данные, считывается из "защелки" порта, а не с выводов БИС.

Команда DA A

Команда десятичная коррекция аккумулятора для сложения" упорядочивает 8-битовую величину в аккумуляторе после выполненной ранее команды сложения двух переменных (каждая в упакованном двоично-десятичном формате). Для выполнения сложения может использоватться любая из типов команд ADD или ADDC. Если значение битов 3-0 аккумулятора (A) превышает 9 (XXXX 1010-XXXX 1111) или, если флаг AC равен "1", то к содержимому (A) прибавляется 06, получая соответствующую двоично-десятичную цифру в младшем полубайте. Это внутреннее побитовое сложение устанавливает флаг переноса, если перенос из поля младших четырех бит распространяется через все старшие биты, а в противном случае - не изменяет флага переноса. Если после этого флаг переноса равен "1", или значение четырех старших бит (7-4) превышает 9 (1010 XXXX - 1111 XXXX), значения этих старших бит увеличивается на 6, создавая соответствующую двоично-десятичную цифру в старшем полубайте. И снова при этом флаг переноса устанавливается, если перенос получается из старших битов, но не изменяется в противном случае. Таким образом, флаг переноса указывает на то, что сумма двух исходных двоично-десятичных переменных больше чем 100. Эта команда выполняет десятичное преобразо-

вание с помощью сложения 06, 60, 66 с содержимым аккумулятора в зависимости от начального состояния аккумулятора и слова состояния программы (PSW).

1.

Ассемблер: DA A

Код: 1 1 0 1 0 1 0 0

Время: 1 цикл

Алгоритм: если ((A[3-0])>9 или (AC)=1), то A[3-0]:=A[3-0]+6
если ((A[7-4])>9 или (C)=1), то A[7-4]:=A[7-4]+6

Пример:
```
a)      ;(A)=56H, (R3)=67H, (C)=1
ADDC A,R3
DA   A    ;(A)=24H, (R3)=67H, (C)=1

b)      ;(A)=30H, (C)=0
ADD A, #99H
DA   A    ;(A)=29, (C)=1
```

Примечание - Команда DA A не может просто преобразовать шестнадцатеричное значение в аккумуляторе в двоично-десятичное представление и не применяется, например, для десятичного вычитания.

Команда DEC <байт>

Команда "декремент" производит вычитание "1" из указанного операнда. Начальное значение 00H перейдет в 0FFH. Команда DEC не влияет на флаги. Этой командой допускается четыре режима адресации операнда:

1. к аккумулятору
2. регистровый
3. прямой
4. косвенно-регистровый

Рассмотрим их.

1.

Ассемблер: DEC A

Код: 0 0 0 1 0 1 0 0

Время: 1 цикл

Алгоритм: (A) : =(A)-1

Пример:
```
        ;(A)=11H, (C)=1, (AC)=1
DEC A   ;(A)=10H, (C)=1, (AC)=1
```

2.

Ассемблер: DEC Rn ; где n=0-7

Код: 0 0 0 1 1 rrr где rrr=000-111

Время: 1 цикл

Алгоритм: (Rn) :=(Rn)-1

Пример:
```
        ;(R1)=7FH,
        ;(ОЗУ[7F])=40H, (ОЗУ[7F])=00H
 DEC @R1
 DEC R1
 DEC @R1    ;(R1)=7EH,
        ;(ОЗУ[7F])=3FH, (ОЗУ[7F])=FFH
```

3.

Ассемблер: DEC <direct>

Код: 0 0 0 1 0 1 0 1 direct address

Время: 1 цикл

Алгоритм: (direct) : = (direct)-1

Пример:
```
        ;(SCON)=A0H, (C)=1, (AC)=1
DEC SCON    ;(SCON)=9FH, (C)=1, (AC)=1
```

4.

Ассемблер: DEC @Ri ; где i=0,1

Код: 0 0 0 1 0 1 1 i

Время: 1 цикл

Алгоритм: ((Ri) :=((Ri)-1)

Пример:
```
        ;(R1)=7FH,
        ;(ОЗУ[7F])=40H, (ОЗУ[7F])=00H
DEC  @R1
DEC  R1
DEC  @R1    ;(R1)=7EH,
        ;(ОЗУ[7F])=3FH, (ОЗУ[7F])=FFH
```

Примечание - Если эта команда используется для изменения информации на выходе порта, значение, используемое как исходные данные, считывается из "защелки" порта, а не с выводов БИС.

Команда DIV AB

Команда "деление" делит 8-битовое целое без знака из аккумулятора A на 8-битовое целое без знака в регистре B. Аккумулятору присваивается целая часть частного (старшие разряды), а регистру B - остаток. Флаги переноса (C) и переполнения (OV) сбра-

сываются. Если (A)<(B), то флаг дополнительного переноса (AC) не сбрасывается. Флаг переноса сбрасывается в любом случае.

1.

Ассемблер: DIV AB

Код: 1 0 0 0 0 1 0 0

Время: 4 цикла

Алгоритм: (A):=((A)/(B))[15-8],
(B):=((A)/(B))[7-0]

Пример: Пусть аккумулятор содержит число 251 (0FBH или 11111011B), а регистр B - число 18 (12H или 00010010B). После выполнения команды

DIV AB

в аккумуляторе будет число 13 (0DH или 00001101B), а в регистре B - число 17 (11H или 00010001B), т.к. 251=(13*18)+17. Флаги C и OV будут сброшены.

Примечание - Если B содержит 00, то после команды DIV содержимое аккумулятора A и регистра B будут не определены. Флаг переноса сбрасывается, а флаг переполнения устанавливается в "1".

Команда DJNZ <байт>, <смещение>

Команда "декремент и переход, если не равно нулю" выполняет вычитание "1" из указанной ячейки и осуществляет ветвление по вычисляемому адресу, если результат не равен нулю. Начальное значение 00H перейдет в 0FFH. Адрес перехода (ветвления) вычисляется сложением значения смещения (со знаком), указанного в последнем байте команды, с содержимым счетчика команд, увеличенным на длину команды DJNZ. На флаги эта команда не влияет и допускает следующие режимы адресации:

1. регистровый
2. прямой

1.

Ассемблер: DJNZ Rn, <метка> ; где n=0-7

Код: 1 1 0 1 1 rrr re18

Время: 2 цикла

Алгоритм: (PC) : = (PC)+2,
(Rn) : = (Rn)-1,
если ((Rn)>0 или (Rn)<0), то (PC):=(PC)+<re18>

Пример:
```
;(R2)=08H, (P1)=FFH (11111111B)
LAB4:  CPL  P1.7
DJNZ R2,LAB4 ;(R2)={07-00}
```

```
;Эта последовательность команд переключает P1.7
;восемь раз и приводит к появлению четырех импульсов
;на выводе БИС, соответствующем биту P1.7
```

2.

Ассемблер: DJNZ <direct>, <метка>

Код: 1 1 0 1 0 1 0 1 direct address re18

Время: 2 цикла

Алгоритм: (PC) : = (PC)+3,
(direct) : = (direct)-1,
если ((вшкусе)>0 или (direct)<0), то
(PC) : = (PC)+<re18>

Пример:

```
                ;(ОЗУ[40])=01H, (ОЗУ[50])=80H,
                ;(ОЗУ[60])=25H
    DJNZ  40H, LAB1 ;(ОЗУ[40]):=00H
    DJNZ  50H, LAB2 ;(ОЗУ[50]):=7FH
    DJNZ  60H, LAB3 ;(ОЗУ[60]):=25H
    ...
LAB1: CLR   A
    ...
LAB2: DEC   R1      ;осуществился переход на
                    ;метку LAB2
```

Примечание - Если команда DJNZ используется для изменения выхода порта, значение, используемое как операнд, считывается из "защелки" порта, а не с выводов БИС.

Команда INC <байт>

Команда "инкремент" выполняет прибавление "1" к указанной переменной и влияет на флаги. Начальное значение 0FFH перейдет в 00H. Эта команда допускает четыре режима адресации:

1. к аккумулятору
2. регистровый
3. прямой
4. косвенно-регистровый

Рассмотрим их.

1.

Ассемблер: INC A

Код: 0 0 0 0 0 1 0 0

Время: 1 цикл

Алгоритм: (A) : =(A)+1

Пример:

```
            ;(A)=1FH, (AC)=0
INC A    ;(A)=20H, (AC)=0
```

2.

Ассемблер: INC Rn ; где n=0-7

Код: 0 0 0 0 1 rrr где rrr=000-111

Время: 1 цикл

Алгоритм: (Rn) :=(Rn)+1

Пример:
```
;(R4)=FFH, (C)=0, (AC)=0
INC  R4   ;(R4)=00H, (C)=0, (AC)=0
```

3.

Ассемблер: INC <direct>

Код: 0 0 0 0 0 1 0 1 direct address

Время: 1 цикл

Алгоритм: (direct) : = (direct)+1

Пример:
```
;(ОЗУ[43])=22H
INC  43H   ;(ОЗУ[43])=23H
```

4.

Ассемблер: INC @Ri ; где i=0,1

Код: 0 0 0 0 0 1 1 i , где i=0,1

Время: 1 цикл

Алгоритм: ((Ri) :=((Ri))+1

Пример:
```
;(R1)=41H, (ОЗУ[41])=4fH, (AC)=0
INC  @R1   ;(R1)=41H, (ОЗУ[41])=50H, (AC)=0
```

Примечание - При использовании команды INC для изменения содержимого порта, величина, используемая как операнд, считывается из "защелки" порта, а не с выводов БИС.

Команда INC DPTR

Команда "инкремент указателя данных" выполняет инкремент (прибавление "1") содержимого 16-битового указателя данных (DPTR). Прибавление "1" осуществляется к 16 битам, причем переполнение младшего байта указателя данных (DPL) из FFH в 00H приводит к инкременту старшего байта указателя данных (DPH). На флаги эта команда не влияет.

Ассемблер: INC DPTR

Код: 1 0 1 0 0 0 1 1

Время: 2 цикла

Алгоритм: (DPTR):=(DPTR)+1

Пример: ;(DPH)=12H, (DPL)=FEH
INC DPTR
INC DPTR
INC DPTR ;(DPH)=13H, (DPL)=01H

Команда JB <bit>, <re18>

Команда "переход если бит установлен" выполняет переход по адресу ветвления, если указанный бит равен "1", в противном случае выполняется следующая команда. Адрес ветвления вычисляется с помощью прибавления относительного смещения со знаком в третьем байте команды (re18) к содержимому счетчика команд после прибавления к нему 3. Проверяемый бит не изменяется. Эта команда на флаги не влияет.

Ассемблер: JB (bit), <метка>
Код: 0 0 1 0 0 0 0 0 bit address re18
Время: 2 цикла
Алгоритм: (PC):=(PC)+3
если (bit)=1, то (PC):=(PC)+<re18>
Пример: ;(A)=96H, (10010110B)
JB ACC.2,LAB5 ;эта команда обеспечивает переход
;на метку LAB5
...
LAB5: INC A

Команда JBC <bit>, <re18>

Команда "переход, если бит установлен и сброс этого бита", выполняет ветвление по вычисляемому адресу, если бит равен "1". В противном случае выполняется следующая за JBC команда. В любом случае указанный бит сбрасывается. Адрес перехода вычисляется сложением относительного смещения со знаком в третьем байте команды (re18) и содержимого счетчика команд после прибавления к нему 3. Эта команда не влияет на флаги.

Ассемблер: JBC (bit), <метка>
Код: 0 0 0 1 0 0 0 0 bit address re18
Время: 2 цикла
Алгоритм: (PC):=(PC)+3
если (bit)=1, то (bit):=0, (PC):=(PC)+<re18>
Пример: (A)=76H (0111 0110B)
JBC ACC.3,LAB6 ; Перехода на LAB6 нет, т.к.
; (A[3])=0
JBC ACC.2,LAB7 ; (A)=72H (0111 0010B) и переход
; на адрес, соответствующий
; метке LAB7

Примечание. Если эта команда используется для проверки бит порта, то значение, используемое как операнд, считывается из "защелки" порта, а не с вывода БИС.

Команда JC <re18>

Команда "переход, если перенос установлен" выполняет ветвление по адресу, если флаг переноса равен "1", в противном случае выполняется следующая команда. Адрес ветвления вычисляется с помощью сложения относительного смещения со знаком во втором байте команды re18) и содержимого счетчика команд, после прибавления к нему 2. Эта команда на флаги не влияет.

Ассемблер: JC <метка>

Код: 0 1 0 0 0 0 0 0 re18

Время: 2 цикла

Алгоритм: (PC):=(PC)+2
если (C)=1, то (PC):=(PC)+<re18>

Пример:
```
        ;(C)=0
    JC   LAB8 ;нет перехода на LAB8
    CPL C     ;(C):=1
LAB8: JC LAB9   ;переход на метку LAB9, т.к. (C)=1
    ...
LAB9: NOP
```

Команда JMP @A+DPTR

Команда "косвенный переход" складывает 8-битовое содержимое аккумулятора без знака с 16-битовым указателем данных (DPTR) и загружает полученный результат в счетчик команд, содержимое которого является адресом для выборки следующей команды. 16-битовое сложение выполняется по модулю 216, перенос из младших восьми бит распространяется на старшие биты программного счетчика. Содержимое аккумулятора и указателя данных не изменяется. Эта команда на флаги не влияет.

Ассемблер: JMP @A+DPTR

Код: 0 1 1 1 0 0 1 1

Время: 2 цикла

Алгоритм: (PC):=(A)[7-0]+(DPTR)[15-0]

Пример:
```
        ;(PC)=034EH, (A)=86H, (DPTR)=0329H
  JMP @A+DPTR ;(PC)=03AEH, (A)=86H, (DPTR)=0329H
```

Команда JNB <bit>, <re18>

Команда "переход, если бит не установлен" выполняет ветвление по адресу, если указанный бит равен "нулю", в противном случае выполняется следующая команда. Адрес

ветвления вычисляется с помощью сложения относительного смещения со знаком в третьем байте команды (re18) и содержимого счетчика команд после прибавления к нему 3. Проверяемый бит не изменяется. Эта команда на флаги не влияет

Ассемблер: JNB (bit), <метка>

Код: 0 0 1 1 0 0 0 0 bit address re18

Время: 2 цикла

Алгоритм: (PC):=(PC)+3
если (bit)=0, (PC):=(PC)+<re18>

Пример:
```
                ;(P2)=CAH (11001010B)
                ;(A)=56H (0101 0110B)
      JNB  P1.3,LAB10  ;нет перехода на LAB10
      JNB  ACC.3,LAB11 ;переход на метку LAB11
      ...
LAB11: INC   A
```

Команда JNC <re18>

Команда "переход, если перенос не установлен" выполняет ветвление по адресу, если флаг переноса равен нулю, в противном случае выполняется следующая команда. Адрес ветвления вычисляется с помощью сложения относительного смещения со знаком во втором байте команды (re18) и содержимого счетчика команд после прибавления к нему 2. Флаг переноса не изменяется. Эта команда на другие флаги не влияет

Ассемблер: JNC <метка>

Код: 0 1 0 1 0 0 0 0 re18

Время: 2 цикла

Алгоритм: (PC):=(PC)+2
если (C)=0, то (PC):=(PC)+<re18>

Пример:
```
            ;(C)=1
      JNC   LAB12 ;нет перехода на LAB12
      CPL C
LAB12: JNC LAB13   ;переход на метку LAB13
```

Команда JNZ <re18>

Команда "переход, если содержимое аккумулятора не равно нулю" выполняет ветвление по адресу, если хотя бы один бит аккумулятора равен "1", в противном случае выполняется следующая команда. Адрес ветвления вычисляется сложением относительного смещения со знаком во втором байте команды (re18) и содержимого счетчика (PC) после прибавления к нему 2. Содержимое аккумулятора не изменяется. Эта команда на флаги не влияет.

Ассемблер: JNZ <метка>

Код: 0 1 1 1 0 0 0 0 re18

Время: 2 цикла

Алгоритм: (PC):=(PC)+2
если (A)□0, то (PC):=(PC)+<re18>

Пример:
```
        ;(A)=00H
    JNC   LAB14 ;нет перехода на LAB14
    INC   A
LAB14: JNZ   LAB15 ;переход на метку LAB15
    ...
LAB15: NOP
```

Команда JZ <re18>

Команда "переход, если содержимое аккумулятора равно 0" выполняет ветвление по адресу, если все биты аккумулятора равны "0", в противном случае выполняется следующая команда. Адрес ветвления вычисляется сложением относительного смещения со знаком во втором байте команды (re18) и содержимым счетчика команд после прибавления к нему 2. Содержимое аккумулятора не изменяется. Эта команда на флаги не влияет.

Ассемблер: JZ <метка>

Код: 0 1 1 0 0 0 0 0 re18

Время: 2 цикла

Алгоритм: (PC):=(PC)+2
если (A)□0, то (PC):=(PC)+<re18>

Пример:
```
        ;(A)=01H
    JZ   LAB16 ;нет перехода на LAB16
    DEC  A
LAB16: JZ   LAB17 ;переход на метку LAB17
    ...
LAB17: CLR  A
```

Команда LCALL <addr16>

Команда "длинный вызов" вызывает подпрограмму, находящуюся по указанному адресу. По команде LCALL к счетчику команд (PC) прибавляется 3 для получения адреса следующей команды и после этого полученный 16-битовый результат помещается в СТЕК (сначала следует младший байт, за ним - старший), содержимое указателя СТЕКа (SP) увеличивается на 2. Затем старший и младший байты счетчика команд загружаются соответственно вторым и третьим байтами команды LCALL. Выполнение программы продолжается командой, находящейся по полученному адресу. Подпрограмма, следовательно,

может начинаться в любом месте адресного пространства памяти программ объемом до 64 Кбайт. Эта команда на флаги не влияет.

Ассемблер: LCALL <метка>

Код: 0 0 0 1 0 0 1 0 addr [15-8] addr [7-0]

Время: 2 цикла

Алгоритм: (PC):=(PC)+3
(SP):=(SP)+1
((SP)):=(PC[7-0])
(SP):=(SP)+1
((SP)):=(PC[15-8])
(PC):=<adddr[15-0]>

Пример:
```
        ;(SP)=07H,
        ;метке PRN соответствует адрес 1234H,
        ;по адресу 0126H находится команда
        ;LCALL
 LCALL  PRN     ;(SP)=09H, (PC)=1234H,
        ;(ОЗУ[08])=26H, (ОЗУ[09])=01H
```

Команда LJMP <addr16>

Команда "длинный переход" выполняет безусловный переход по указанному адресу, загружая старший и младший байты счетчика команд (PC) соответственно вторым и третьим байтами, находящимися в коде команды. Адрес перехода, таким образом, может находиться по любому адресу пространства памяти программ в 64 Кбайт. Эта команда на флаги не влияет.

Ассемблер: LJMP <метка>

Код: 0 0 0 0 0 0 1 0 addr [15-8] addr [7-0]

Время: 2 цикла

Алгоритм: (PC):=<addr[15-0]>

Команда MOV <байт-назначения>,<байт-источника>

Команда "переслать переменную-байт" пересылает переменную-байт, указанную во втором операнде, в ячейку, указанную в первом операнде. Содержимое байта источника не изменяется. Эта команда на флаги и другие регистры не влияет. Команда "MOV" допускает 15 комбинаций адресации байта-источника и байта-назначения.

1.

Ассемблер: MOV A,Rn; где n=0-7

Код: 1 1 1 0 1 rrr где rrr=000-111

Время: 1 цикл

Алгоритм: (A) : =(Rn)

Пример: ;(A)=FAH, (R4)=93H
MOV A,R4 ;(A)=93H, (R4)=93H

2.

Ассемблер: MOV A, <direct>

Код: 1 1 1 0 0 1 0 1 direct address

Время: 1 цикл

Алгоритм: (A) :=(direct)

Пример: ;(A)=93H, (ОЗУ[40])=10H, (R0)=40H
MOV A,40H ;(A)=10H, (ОЗУ[40])=10H, (R0)=40H

3.

Ассемблер: MOV A,@Ri; где i=0,1

Код: 1 1 1 0 0 1 1 i

Время: 1 цикл

Алгоритм: (A) : = ((Ri))

Пример: ;(A)=10H, (R0)=41H, (ОЗУ[41])=0CAH
MOV A,@R0 ;(A)=CAH, (R0)=41H, (ОЗУ[41])=0CAH

4.

Ассемблер: MOV A,#data

Код: 0 1 1 1 0 1 0 0 #data8

Время: 1 цикл

Алгоритм: (A) :=<#data8>

Пример: ;(A)=C9H (11001001B)
MOV A,#37H ;(A)=37H (00110111B)

5.

Ассемблер: MOV Rn ,A; где n=0-7

Код: 1 1 1 1 1 rrr где rrr=000-111

Время: 1 цикл

Алгоритм: (Rn) : =(A)

Пример: ;(A)=38H, (R0)=42H
MOV R0,A ;(A)=38H, (R0)=38H

6.

Ассемблер: MOV Rn, <direct>; где n=0-7

Код: 1 0 1 0 1 rrr direct address

Время: 2 цикла

Алгоритм: (Rn) :=(direct)

Пример:
```
;(R0)=39H, (P2)=0F2H
MOV R0,P2  ;(R0)=F2H
```

7.

Ассемблер: MOV Rn,#data; где n=0-7

Код: 0 1 1 1 1 rrr #data8

Время: 1 цикл

Алгоритм: (Rn) :=<#data8>

Пример:
```
;(R0)=0F5H
MOV  R0,#49H ;(R0)=49H
```

8.

Ассемблер: MOV <direct>,A

Код: 1 1 1 1 0 1 0 1 direct address

Время: 1 цикл

Алгоритм: (direct) :=(A)

Пример:
```
;(P0)=FFH, (A)=4BH
MOV P0,A   ;(P0)=4BH, (A)=4BH
```

9.

Ассемблер: MOV <direct>, Rn ; где n=0-7

Код: 1 0 0 0 1 rrr direct address

Время: 2 цикла

Алгоритм: (direct) :=(Rn)

Пример:
```
;(PSW)=C2H, (R7)=57H
MOV PSW,R7 ;(PSW)=57H, (R7)=57H
```

10.

Ассемблер: MOV <direct>, <direct>

Код: 1 0 0 0 0 1 0 1 direct address direct address

Время: 2 цикла

Алгоритм: (direct) :=(direct)

Пример:
```
;(ОЗУ[45])=33H, (ОЗУ[48])=0DEH
MOV 48H,45H ;(ОЗУ[45])=33H, (ОЗУ[45])=33H
```

11.

Ассемблер: MOV <direct>,@Ri ; где i=0,1

Код: 1 0 0 0 0 1 1 i direct address

Время: 2 цикла

Алгоритм: (direct) :=((Ri))

Пример: ;(R1)=49H, (ОЗУ[49])=0E3H
MOV 51H,@R1 ;(ОЗУ[51])=0E3H, (ОЗУ[49])=0E3H

12.

Ассемблер: MOV <direct>, #data

Код: 0 1 1 1 0 1 0 1 direct address #data8

Время: 2 цикла

Алгоритм: (direct) :=<#data8>

Пример: ;(ОЗУ[5F])=9BH
MOV 5FH,#07H ;(ОЗУ[5F])=07H

13.

Ассемблер: MOV @Ri,A; где i=0-7

Код: 1 1 1 1 0 1 1 i где i=0,1

Время: 1 цикл

Алгоритм: ((Ri)) : =(A)

Пример: ;(R1)=51H, (ОЗУ[48])=75H, (A)=0BDH
MOV @R1,A ;(ОЗУ[48])=0BDH

14.

Ассемблер: MOV @Ri, <direct>, где i=0,1

Код: 1 0 1 0 0 1 1 i direct address

Время: 2 цикла

Алгоритм: ((Ri)) :=(direct)

Пример: ;(R0)=51H, (ОЗУ[51])=0E3H, (P0)=0ACH
MOV @R0,P0 ;(A)=10H, (ОЗУ[51])=0ACH

15.

Ассемблер: MOV Ri,#data ; где i=0,1

Код: 0 1 1 1 0 1 1 i

Время: 1 цикл

Алгоритм: ((Ri)) : =<#data8>

Пример: ;(ОЗУ[7E])=67H, (R1)=7EH
MOV @R1,#0A9H ;(ОЗУ[7E])=0A9H, (R1)=7EH

Команда MOV <бит-назначения>,<бит-источника>

Команда "переслать бит данных" пересылает битовую переменную, указанную во втором байте, копирует в разряд, который указан в первом операнде. Одним из операндов должен быть флаг переноса C, а другим может быть любой бит, к которому возможна прямая адресация.

1.

Ассемблер: MOV C, <bit>
Код: 1 0 1 0 0 0 1 0 bit address
Время: 1 цикл
Алгоритм: (C) : = (bit)
Пример:
```
;(C)=0,  (P3)=D5H (11010101B)
MOV C,P3.0 ;C:=1
MOV C,P3.3 ;C:=0
MOV C,P3.7 ;C:=1
```

2.

Ассемблер: MOV <bit>,C
Код: 1 0 0 1 0 0 1 0 bit address
Время: 2 цикла
Алгоритм: (bit) : =(C)
Пример:
```
;(C)=1, (P0)=2OH  (00100000B)
MOV  PO.1,C
MOV  P0.2,C
MOV  P0.3,C ;(C)=1, (P0)=2EH  (00101110B)
```

Команда MOV DPTR,#data16

Команда "загрузить указатель данных 16-битовой константой" загружает указатель данных DPTR 16-битовой константой, указанной во втором и третьем байтах команды. Второй байт команды загружается в старший байт указателя данных (DPH), а третий байт - в младший байт указателя данных (DPL). Эта команда на флаги не влияет и является единственной командой, которая одновременно загружает 16 бит данных.

1.

Ассемблер: MOV DPTR, #<data16>
Код: 1 0 0 1 0 0 0 0 #data[15-8] #data[7-0]
Время: 2 цикла
Алгоритм: (DPTR) :=#[15-0], причем DPH:=#data[15-8] , DPL:=#data[7-0]

Пример: ;(DPTR)=01FDH
MOV DPTR,#1234H ;(DPTR)=1234H,
;(DPH)=12H, (DPL)=34H

Команда MOVC A,@A+(<R16>)

<R16> -16-разрядный регистр.

Команда "переслать байт из памяти программ" загружает аккумулятор байтом кода или константой из памяти программы. Адрес считываемого байта вычисляется как сумма 8-битового исходного содержимого аккумулятора без знака и содержимого 16-битового регистра. В качестве 16-битового регистра может быть:

1. указатель данных DPTR
2. счетчик команд PC

В случае, когда используется PC, он увеличивается до адреса следующей команды перед тем, как его содержимое складывается с содержимым аккумулятора. 16-битовое сложение выполняется так, что перенос из младших восьми бит может распространяться через старшие биты. Эта команда на флаги не влияет.

1.

Ассемблер: MOVC A, @A+DPTR
Код: 1 0 0 1 0 0 1 1
Время: 2 цикла
Алгоритм: (A) : = ((A)+(DPTR))
Пример: ;(A)=1BH, (DPTR)=1020H,
;(ПЗУ[103B])=48H,
MOVC A,@A+DPTR ;(A)=48H, (DPTR)=1020H

2.

Ассемблер: MOVC A, @A+PC
Код: 1 0 0 0 0 0 1 1
Время: 2 цикла
Алгоритм: (A) : =((A)+(PC))
Пример: ;(A)=FAH, (PC)=0289
;(ПЗУ[0384])=9BH
MOVC A, @A+PC ;(A)=9BH, (PC)=028AH

Команда MOVX <байт приемника>,<байт источника>

Команда "переслать во внешнюю память (из внешней памяти) данных" пересылает данные между аккумулятором и байтом внешней памяти данных. Имеется два типа ко-

манд, которые отличаются тем, что обеспечивают 8-битовый или 16-битовый косвенный адрес к внешнему ОЗУ данных.

В первом случае содержимое R0 или R1 в текущем банке регистров обеспечивает 8-битовый адрес, который мультиплексируется с данными порта P0. Для расширения дешифрации ввода-вывода или адресации небольшого массива ОЗУ достаточно восьми бит адресации. Если применяются ОЗУ, немного больше чем 256 байт, то для фиксации старших битов адреса можно использовать любые другие выходы портов, которые переключаются командой, стоящей перед командой MOVX.

Во втором случае, при выполнении команды MOVX указатель данных DPTR генерирует 16-битовый адрес. Порт P2 выводит старшие 8 бит адреса (DPH), а порт P0 мультиплексирует младшие 8 бит адреса (DPL) с данными. Эта форма является эффективной при доступе к большим массивам данных (до 64К байт), так как для установки портов вывода не требуется дополнительных команд.

1.

Ассемблер: MOVX A@Ri; где i=0,1
Код: 1 1 1 0 0 0 1 i
Время: 2 цикла
Алгоритм: (A) : = ((Ri))
Пример:
```
;(A)=32H,  (R0)=83H, ячейка
;внешнего ОЗУ по адресу 83H
;содержит B6H
MOVX  A,@R0  ;(A)=B6H,  (R0)=83H
```

2.

Ассемблер: MOVX A, @DPTR
Код: 1 1 1 0 0 0 0 0
Время: 2 цикла
Алгоритм: (A) : =((DPTR))
Пример:
```
;(A)=5CH, (DPTR)=1ABEH,
;ячейка внешнего ОЗУ по адресу
;1ABEH содержит 72H
MOVX  A, @DPTR ;(A)=72H, (DPTR)=2ABEH
```

3.

Ассемблер: MOVX Ri@A; где i=0,1
Код: 1 1 1 1 0 0 1 i
Время: 2 цикла

Алгоритм: ((Ri)) : = (A)

Пример:
```
;(A)=95H,  (R1)=FDH,
;ячейка внешнего ОЗУ с адресом
;FDH содержит 00
MOVX  R1,@A  ;(A)=95H,  (R1)=FDH,
;ячейка внешнего ОЗУ с адресом
;FDH содержит 95H
```

4.

Ассемблер: MOVX @DPTR,A

Код: 1 1 1 1 0 0 0 0

Время: 2 цикла

Алгоритм: ((DPTR)) : =(A)

Пример:
```
;(A)=97H, (DPTR)=1FFFH,
;ячейка внешнего ОЗУ с адресом
;1FFFH содержит 00
MOVX  @DPTR, A ;(A)=97H,
;ячейка внешнего ОЗУ с адресом
;1FFFH содержит 97H
```

Команда MUL AB

Команда "умножение" умножает 8-битовые целые числа без знака из аккумулятора и регистра B. Старший байт 16-битового произведения помещается в регистр B, а младший - в аккумулятор A. Если результат произведения больше, чем 0FFH(255), то устанавливается флаг переполнения (OV), в противном случае он сбрасывается. Флаг переноса всегда сбрасывается.

Ассемблер: MUL AB

Код: 1 0 1 0 0 1 0 0

Время: 4 цикла

Алгоритм: (A)[7-0]=(A)*(B),
(B)[15-8]=(A)*(B)

Пример:
```
;(A)=50H (50H=80 DEC), (C)=1,
;(B)=0A0H (A0H=160 DEC), (OV)=0
MUL  AB   ;(A)=00H, (B)=32H, (C)=0, (OV)=1
;(A)=2HH, (OV)=1, (B)=06H, (C)=1
MUL  AB   ;(A)=0D8H, (B)=00H, (OV)=0, (C)=0
```

Команда NOP

Команда "нет операции" выполняет холостой ход и не влияет на регистры и флаги, кроме как на счетчик команд (PC).

Ассемблер: NOP

Код: 0 0 0 0 0 0 0 0

Время: 1 цикл

Алгоритм: (PC):=(PC)+1

Пример: Пусть требуется создать отрицательный выходной импульс на порте P1[6] длительностью 3 цикла. Это выполнит следующая последовательность команд:

```
CLR  P1.6  ;P1[6]:=0
NOP
NOP
NOP
SETB  P1.6  ;P1[6]:=1
```

Команда ORL <байт-назначения>,<байт-источника>

Команда "логическое "ИЛИ" для переменных байтов" выполняет операцию логического "ИЛИ" над битами указанных переменных, записывая результат в байт назначения. Эта команда на флаги не влияет. Допускается шесть комбинаций режимов адресации:

- если байтом назначения является аккумулятор :

1. регистровый
2. прямой
3. косвенно-регистровый
4. непосредственный

- если байтом назначения является прямой адрес :

5. к аккумулятору
6. к константе

Рассмотрим их.

1.

Ассемблер: ORL A,Rn ; где n=0-7

Код: 0 1 0 0 1 rrr , где rrr=000-111

Время: 1 цикл

Алгоритм: (A) := (A) OR (Rn),
где OR - операция логического "ИЛИ"

Пример:

```
;(A)=15H, (R5)=6CH
ORL  A,R5  ;(A)=7DH, (R5)=6CH
```

2.

Ассемблер: ORL A,<direct>

Код: 0 1 0 0 0 1 0 1 direct address

Время: 1 цикл

Алгоритм: (A) : =(A) OR (direct)

Пример:
```
;(A)=84H, (PSW)=C2H
ORL  A,PSW ;(A)=C6H, (PSW)=C2H
```

3.

Ассемблер: ORL A,@Ri ; где i=0,1

Код: 0 1 0 0 0 1 1 i

Время: 1 цикл

Алгоритм: (A) : =(A) OR ((Ri))

Пример:
```
;(A)=52H,  (R0)=6DH, (ОЗУ [6D])=49H
ORL  A,@R0  ;(A)=58H, (ОЗУ [6D])=49H
```

4.

Ассемблер: ORL A, #data

Код: 0 1 0 0 0 1 0 0 #data8

Время: 1 цикл

Алгоритм: (A) : = (A) OR #data

Пример:
```
;(A)=FOH
ORL  A,#0AH  ;(A)=FAH
```

5.

Ассемблер: ORL <direct>, A

Код: 0 1 0 0 0 0 1 0 direct address

Время: 1 цикл

Алгоритм: (direct) : = (direct) OR (A)

Пример:
```
;(A)=34H, (IP)=23H
ORL  IP,A  ;(IP)=37H, (A)=34H
```

6.

Ассемблер: ORL (direct), #<data>

Код: 0 1 0 0 0 0 1 1 direct address #data8

Время: 2 цикла

Алгоритм: (direct) : = (direct) OR #<data>

Пример:
```
;(P1)=00H
ORL P1,#0C4H ;(P1)=11000100B (C4H)
```

Примечание - Если команда используется для работы с портом, величина, используемая в качестве исходных данных порта, считывается из "защелки" порта, а не с выводов БИС.

Команда ORL C, <бит источника>

Команда "логическое "ИЛИ" для переменных битов" устанавливает флаг переноса C, если булева величина равна логической "1", в противном случае устанавлиается флаг C в "0". Косая дробь ("/") перед операндом на языке ассемблера указывает на то, что в качестве операнда используется логическое отрицание значения адресуемого бита, но сам бит источника не изменяется. Эта команда на другие флаги не влияет.

1.

Ассемблер: ORL C,<bit>

Код: 0 1 1 1 0 0 1 0 bit address

Время: 2 цикла

Алгоритм: (C) : =(C) OR (bit)

Пример:
```
;(C)=0, (P1)=53H  (01010011B)
ORL  C,P1.4 ;(C)=1, (P1)=53H  (01010011B)
```

2.

Ассемблер: ORL C,/ <bit>

Код: 1 0 1 0 0 0 0 0 bit address

Время: 2 цикла

Алгоритм: (C) : = (C) OR /(bit)

Пример:
```
;(C)=0, (ОЗУ[25])39H (0011100B)
ORL  C,/2A   ;(C)=1, (ОЗУ[25])39H (0011100B)
```

Команда POP <direct>

Команда "чтение из стека" считывает содержимое ячейки, которая адресуется с помощью указателя стека, в прямо адресуемую ячейку ОЗУ, при этом указатель стека уменьшается на единицу.

Эта команда не воздействует на флаги и часто используется для чтения из стека промежуточных данных.

Ассемблер: POP <direct>

Код: 1 1 0 1 0 0 0 0 direct address

Время: 2 цикла

Алгоритм: (direct):=((SP)),
(SP):=(SP)-1

Пример:
```
;(SP)=32H, (DPH)=01, (DPL)=ABH,
;(ОЗУ[32])=12H, (ОЗУ[32])=56H,
POP  DPH
POP  DPL   ;(SP)=30H, (DPH)=12H, (DPL)=56H,
;(ОЗУ[32])=12H, (ОЗУ[31])=56H
POP  SP    ;(SP)=20H, (ОЗУ[30])=20H
```

Команда PUSH <direct>

Команда "запись в стек" увеличивает указатель стека на единицу и после этого содержимое указанной прямо адресуемой переменной копируется в ячейку внутреннего ОЗУ, адресуемого с помощью указателя стека. На флаги эта команда не влияет и используется для записи промежуточных данных в стек.

Ассемблер: PUSH <direct>

Код: 1 1 0 0 0 0 0 0 direct address

Время: 2 цикла

Алгоритм: (SP):=(SP)+1

Пример:
```
;(SP)=09H, (DPTR)=1279H
PUSH  DPL
PUSH  DPH  ;(SP)=0BH, (DPTR)=1279H,
;(ОЗУ[0A])=79H, (ОЗУ[0B])=12H
```

Команда RET

Команда "возврат из подпрограммы" последовательно выгружает старший и младший байты счетчика команд из стека, уменьшая указателя стека на 2. Выполнение основной программы обычно продолжается по адресу команды, следующей за ACALL или LCALL. На флаги эта команда не влияет.

Ассемблер: RET

Код: 0 0 1 0 0 0 1 0

Время: 2 цикла

Алгоритм: (PC)[15-8]:=((SP)),
(SP):=(SP)-1,
(PC)[7-0]:=((SP)),

(SP):-(SP)-1

Пример: ;(SP)=0DH, (ОЗУ[0C])=93H, (ОЗУ[0D])=02H
RET ;(SP)=0BH, (PC)=0293H

Команда RETI

Команда "возврат из прерывания" выгружает старший и младший байты счетчика команд из стека и устанавливает "логику прерываний", разрешая прием других прерываний с уровнем приоритета, равным уровню приоритета только что обработанного прерывания. Указатель стека уменьшается на 2. Слово состояния программы (PSW) не восстанавливается автоматически. Выполнение основной программы продолжается с команды, следующей за командой, на которой произошел переход к обнаружению запроса на прерывание. Если при выполнении команды RETI обнаружено прерывание с таким же или меньшим уровнем приоритета, то одна команда основной программы успевает выполниться до обработки такого прерывания.

Ассемблер: RETI

Код: 0 0 1 1 0 0 1 0

Время: 2 цикла

Алгоритм: (PC)[15-8]:=((SP)),
(SP):=(SP)-1,
(PC)[7-0]:=((SP)),
(SP):-(SP)-1

Пример: ;(SP)=0BH, (ОЗУ[0A])=2AH, (ОЗУ[0B])=03H,
;(PC)=УУУУН, где У=0-FH
RETI ;(SP)=09H, (PC)=032AH

Команда RL A

Команда "сдвиг содержимого аккумулятора влево", сдвигает восемь бит аккумулятора на один бит влево, бит 7 засылается на место бита 0. Н а флаги эта команда не влияет.

Ассемблер: RL A

Код: 0 0 1 0 0 0 1 1

Время: 1 цикл

Алгоритм: (A[N+1]):=(A[N]), где N=0-6
(A[0]):=(A[7])

Пример: ;(A)=0D5H, (11010101B), (C)=0,
RL A ;(A)=0ABH, (10101011B), (C)=0

Команда RLC A

Команда "сдвиг содержимого аккумулятора влево через флаг переноса" сдвигает восемь бит аккумулятора и флаг переноса влево на один бит. Содержимое флага переноса помещается на место бита 0 аккумулятора, а содержимое бита 7 аккумулятора переписывается в флаг переноса. На другие флаги эта команда не влияет.

Ассемблер: RL A

Код: 0 0 1 1 0 0 1 1

Время: 1 цикл

Алгоритм: (A[N+1]):=(A[N]), где N=0-6
(A[0]):=(C)
(C):=(A[7])

Пример:
```
;(A)=56H, (01010110B), (C)=1
RLC  A ;(A)=0ADH, (10101101B), (C)=0
```

Команда RR A

Команда "сдвиг содержимого аккумулятора вправо" сдвигает вправо на один бит все восемь бит аккумулятора. Содержимое бита 0 помещается на место бита 7. На флаги эта команда не влияет.

Ассемблер: RR A

Код: 0 0 0 0 0 0 1 1

Время: 1 цикл

Алгоритм: (A[N]):=(A[N+1]), где N=0-6
(A[7]):=(A[0])

Пример:
```
;(A)=0D6H, (11010110B), (C)=1
RR   A ;(A)=6BH, (01101011B), (C)=1
```

Команда RRC A

Команда "сдвиг содержимого аккумулятора вправо через флаг переноса" сдвигает восемь бит аккумулятора и флаг переноса на один бит вправо. Бит 0 перемещается в флаг переноса, а исходное содержимое флага переноса помещается в бит 7. На другие флаги эта команда не влияет.

Ассемблер: RRC A

Код: 0 0 0 1 0 0 1 1

Время: 1 цикл

Алгоритм: (A[N]):=(A[N+1]), где N=0-6
(A[7]):=(C),
(C):=(A[0])

Пример: ;(A)=95H (10010101B), (C)=0
RRC A ;(A)=4AH (01001010B), (C)=1

Команда SETB <bit>

Команда "установить бит" устанавливает указанный бит в "1". Адресуется:

1. к флагу переноса (C);
2. к биту с прямой адресацией.

1.

Ассемблер: SETB C
Код: 1 1 0 1 0 0 1 1
Время: 1 цикл
Алгоритм: (C) : = 1
Пример: ;(C)=0
SETB C ;(C)=1

2.

Ассемблер: SETB (bit)
Код: 1 1 0 1 0 0 1 0 bit address
Время: 1 цикл
Алгоритм: (bit) : = 1
Пример: ;(P2)=38H (00111000B)
SETB P2.0
SETB P2.7 ;(P2)=B9H (10111001B)

Команда SJMP <метка>

Команда "короткий переход" выполняет безусловное ветвление в программе по указанному адресу. Адрес ветвления вычисляется сложением смещения со знаком во втором байте команды с содержимым счетчика команд после прибавления к нему 2.

Таким образом, адрес перехода должен находиться в диапазоне от 128 байт, предшествующих команде, до 127 байт, следующих за ней.

Ассемблер: SJMP <метка>
Код: 1 0 0 0 0 0 0 0 re18
Время: 2 цикла
Алгоритм: (PC):=(PC)+2,
(PC):=(PC)+(re18)
Пример: ;(PC)=0418H,

```
        ;метка MET1 соответствует адресу 039AH
SJMP MET1 ;(PC)=039AH, где (re18)=80h=-128 DEC
SJMP MET2 ;(PC)=041AH, где метка MET2 соответствует
        ;адресу 041AH,
        ;(re18)=7DH=+125 DEC
```

Команда SUBB A, <байт источника>

Команда "вычитание с заемом" вычитает указанную переменную вместе с флагом переноса из содержимого аккумулятора, засылая результат в аккумулятор. Эта команда устанавливает флаг переноса (заема), если при вычитании для бита 7 необходим заем, в противном случае флаг переноса сбрасывается. Если флаг переноса установлен перед выполнением этой команды, то это указывает на то, что заем необходим при вычитании с увеличенной точностью на предыдущем шаге, поэтому флаг переноса вычитается из содержимого аккумулятора вместе с операндом источника. (AC) устанавливается, если заем необходим для бита 3 и сбрасывается в противном случае. Флаг переполнения (OV) устанавливается, если заем необходим для бита 6, но его нет для бита 7, или есть для бита 7, но нет для бита 6.

При вычитании целых чисел со знаком (OV) указывает на отрицательное число, которое получается при вычитании отрицательной величины из положительной, или положительное число, которое получается при вычитании положительного числа из отрицательного.

Операнд источника допускает четыре режима адресации:

1. регистровый;
2. прямой;
3. косвенно-регистровый;
4. непосредственный (к константе).

1.

Ассемблер: SUBB A, Rn; где n=0-7

Код: 1 0 0 1 1 rrr где r=000-111

Время: 1 цикл

Алгоритм: (A) : = (A)-(C)-(Rn);
(C):=X, (AC):=X, (OV):=X, где X=(0 или 1)

Пример:

```
        ;(A)=C9H,  (R2)=54H, (C)=1
SUBB  A,R2   ;(A)=74H,  (R2)=54H, (C)=0
        ;(AC)=0, (OV)=1
```

2.

Ассемблер: SUBB A, <direct>

Код: 1 0 0 1 0 1 0 1 direct address

Время: 1 цикл

Алгоритм: (A) : =(A)-(C)-(direct);
(C):=X, (AC):=X, (OV):=X, где X=(0 или 1)

Пример:
```
;(A)=97H, (B)=25H, (C)=9
SUBB  A,B ;(A)=72H, (B)=25H, (C)=0,
;(AC)-0, (OV)=1
```

3.

Ассемблер: SUBB A@Ri; где i=0,1

Код: 1 0 0 1 0 1 1 i

Время: 1 цикл

Алгоритм: (A) : = (A)-(C)-((Ri));
(C):=X, (AC):=X, (OV):=X, где X=(0 или 1)

Пример:
```
;(A)=49H,  (C)=1, (R0)=33H,
;(ОЗУ[33])=68H
SUBB  A,@R0  ;(A)=E0H,  (C)=1, (AC)=0, (OV)=0
```

4.

Ассемблер: SUBB A,#data

Код: 1 0 0 1 0 1 0 0 #data8

Время: 1 цикл

Алгоритм: (A):=(A)-(C)-(#data8);
(C):=X, (AC):=X, (OV):=X, где X=(0 или 1)

Пример:
```
;(A)=0BEH, (C)=0
SUBB  A,#3FH   ;(A)=7FH, (C)-0, (AC)=1, (OV)=1
```

Команда SWAP A

Команда "обмен тетрадами внутри аккумулятора" осуществляет обмен между младшими четырьмя и старшими четырьмя битами аккумулятора (между старшей и младшей тетрадами).

Эта команда может рассматриваться так же, как команда четырехбитового циклического сдвига. На флаги эта команда не влияет.

Ассемблер: SWAP A

Код: 1 1 0 0 0 1 0 0

Время: 1 цикл

Алгоритм: (A[3-0]):=(A[7-4]), (A[7-4]):=(A[3-0])

Пример: ;(A)=0D7H (11010111B)
SWAP A ;(A)=7DH (01111101B)

Команда XCH A, <байт>

Команда "обмен содержимого аккумулятора с переменной-байтом" осуществляет обмен содержимого аккумулятора с содержимым источника, указанным в команде. Операнд источника может использовать следующие режимы адресации:

1. регистровый;
2. прямой;
3. косвенно-регистровый;

1.

Ассемблер: XCH A, Rn; где n=0-7
Код: 1 1 0 0 1 rrr где r=000-111
Время: 1 цикл
Алгоритм: (A) : = (Rn), (Rn):=(A)
Пример: ;(A)=3CH, (R4)=15H
XCH A,R4 ;(A)=15H, (R4)=3CH

2.

Ассемблер: XCH A, <direct>
Код: 1 1 0 0 0 1 0 1 direct address
Время: 1 цикл
Алгоритм: (A) : =(direct), (direct):=(A)
Пример: ;(A)=0FEH, (P3)=0DAH
XCH A,P3 ;(A)=0DAH, (P3)=0FEH

3.

Ассемблер: XCH A@Ri; где i=0,1
Код: 1 1 0 0 0 1 1 i
Время: 1 цикл
Алгоритм: (A) : = ((Ri)), ((Ri)):=(A)
Пример: ;(R1)=39H, (ОЗУ[39])=44H, (A)=0BCH
XCH A,@R1 ;(ОЗУ[39])=0BCH, (A)=44H

Команда XCHD A, @R1

Команда "обмен тетрадой" выполняет обмен младшей тетрады (биты 3-0) аккумулятора с содержимым младшей тетрады (биты 3-0) ячейки внутреннего ОЗУ, косвенная адресация к которой производится с помощью указанного регистра. На старшие биты (биты 7-4) эта команда не влияет (так же, как и на флаги).

Ассемблер: XCHD A,@Ri ; где i=0,1

Код: 1 1 0 1 0 1 1 i

Время: 1 цикл

Алгоритм: (A[3-0]):=((Ri[3-0])),
((Ri[3-0])):=(A[3-0])

Пример:
```
;(R0)=55H, (A)=89H, (ОЗУ[55])=0A2H
XCHD A,@R0 ;(A)=82H, (ОЗУ[55])=0A9H
```

Команда XRL <байт-назначения>,<байт-источника>

Команда "логическое " ИСКЛЮЧАЮЩЕЕ ИЛИ" для переменных-байтов" выполняет операцию "ИСКЛЮЧАЮЩЕЕ ИЛИ" над битами указанных переменных, записывая результат в байт назначения. Эта команда на флаги не влияет. Допускается шесть комбинаций режимов адресации:

- если байтом назначения является аккумулятор:

2. регистровый
3. прямой
4. косвенно-регистровый
5. непосредственный

- если байтом назначения является прямой адрес :

6. к аккумулятору
7. к константе

Рассмотрим их.

1.

Ассемблер: XRL A,Rn ; где n=0-7

Код: 0 1 1 0 1 rrr , где rrr=000-111

Время: 1 цикл

Алгоритм: (A) : = (A) XOR (Rn)

Пример:
```
;(A)=C3H, (R6)=0AAH
XRL  A,R6  ;(A)=69H, (R6)=0AAH
```

2.

Ассемблер: XRL A,<direct>

Код: 0 1 1 0 0 1 0 1 direct address

Время: 1 цикл

Алгоритм: (A) : =(A) XOR (direct)

Пример: ;(A)=0FH, (P1)=0A6H
XRL A,P1 ;(A)=A9H, (P1)=0A6H

3.

Ассемблер: XRL A,@Ri ; где i=0,1

Код: 0 1 1 0 0 1 1 i

Время: 1 цикл

Алгоритм: (A) : =(A) XOR ((Ri))

Пример: ;(A)=55H, (R1)=77H, (ОЗУ [77])=5AH
XRL A,@R1 ;(A)=0FH, (ОЗУ [77])=5AH

4.

Ассемблер: XRL A, #data

Код: 0 1 1 0 0 1 0 0 #data8

Время: 1 цикл

Алгоритм: (A) : = (A) XOR <data>

Пример: ;(A)=0C3H
XRL A,#0F5H ;(A)=36H

5.

Ассемблер: XRL <direct>, A

Код: 0 1 1 0 0 0 1 0 direct address

Время: 1 цикл

Алгоритм: (direct) : = (direct) XOR (A)

Пример: ;(A)=31H, (P1)=82H
XRL P1,A ;(A)=31H, (P1)=B3H

6.

Ассемблер: XRL <direct>, #data

Код: 0 1 1 0 0 0 1 1 direct address #data8

Время: 2 цикла

Алгоритм: (direct) : = (direct) XOR #data

Пример:
```
;(IP)=65H
XRL IP,#65H ;(IP)=00H
```

Примечание - Если эта команда используется для работы с портами, то значение, используемое в качестве операнда, считывается из "защелки" порта, а не с выводов БИС.

11 Заключение

В настоящем РП приведено подробное описание архитектуры, функционального построения, системы команд и особенностей применения ИМС 1830ВЕ81Т, 1830ВЕ91Т которые представляют собой СБИС однокристального 8-разрядного микроконтроллера с внутренней программной памятью объемом 2К.

Все значения электрических параметров ИМС приведены в ТУ на изделие, значения параметров, приведенные в РП, являются справочными.

Настоящее РП может служить практическим руководством по применению микроконтроллера для разработчиков систем на основе ИМС 1830ВЕ81Т, 1830ВЕ91Т и программистов.

Применение микроконтроллеров в цифровых системах управления, для управления робототехническими комплексами, в системах автоматизации технологических процессов, в системах автоматизированного управления электроприводом, оргтехнике, вычислительной технике, телекоммуникационной технике и т.д. позволит создавать более совершенные в техническом отношении и надежные в эксплуатации изделия.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум	№ докум.	Подп.	Дата
	измененных	замененных	новых	аннулированных				
-	-	-	все	-	108	6604		14.09.06
1	-	4, 5, 107	-	-	-	8052		19.02.13