МИКРОСХЕМЫ ИНТЕГРАЛЬНЫЕ

1882ВЕ53У

Руководство пользователя

2008

Содержание

1 Введение

Интегральные микросхемы 1882ВЕ53У (далее ИМС) представляют собой высокопроизводительные 8-разрядные микроконтроллеры, имеющие высокие технические характеристики и «сверху - вниз» программно и аппаратно совместимые с микросхемами Н1830ВЕ31, Н1830ВЕ51, а также с ИМС 1882ВЕ52У.

КМОП микроконтроллер 1882ВЕ53У оснащен Flash ПЗУ, которое может быть загружено непосредственно в системе через последовательный SPI интерфейс и совместим по системе команд и по функциональному назначению выводов с аналогичными приборами семейства 80С51. Микроконтроллер содержит 12 Кбайт Flash ПЗУ, 2 Кбайт ЭСППЗУ, 256 байтов ОЗУ, два указателя данных, 32 программируемых линии ввода-вывода, три 16-разрядных таймера/счетчика событий, универсальный асинхронный приемопередатчик (UART), программируемый сторожевой таймер, контроллер прерываний с четырьмя уровнями приоритетов и встроенный тактовый генератор. Содержимое Flash памяти программ может быть защищено от несанкционированной записи/считывания. Тактовая частота микроконтроллера 24 МГц.

2 Назначение и основные технические характеристики микросхем

Микросхемы предназначены для применения во встроенных системах управления комплексами радиосвязи, в системах автоматизации технологических процессов, в системах автоматизированного управления электроприводом, оргтехнике, вычислительной технике, телекоммуникационной технике, для управления робототехническими комплексами и т. п.

Функциональным аналогом микросхем являются изделия AT89S8253 фирмы Atmel. (Информация по аналогу - AT89S8253.pdf)

2.1 Архитектурные характеристики:

- разрядность АЛУ, бит	8
- Flash-память программ, байт	12К
- ЭСППЗУ память данных, байт	2К
- ОЗУ данных, байт	256
- адресуемая память, байт	64К
- таймеры 16-разрядные	3
- сторожевой таймер	1
- последовательный порт	1
- последовательный периферийный интерфейс	1
- порты ввода-вывода 8-разрядные	4
- количество источников прерываний	9

2.2 Конструктивные характеристики

Кристаллы микросхем выполнены по КМОП технологии.

Микросхемы 1882ВЕ53У выпускаются в корпусах Н16.48-2В, 5133.48-3.

Условное графическое обозначение ИМС приведено на рисунке 1.

Функциональное назначение выводов приведено в таблице 1.

2.3 Электрические характеристики

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

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

Динамические характеристики ИМС приведены в таблице 4.

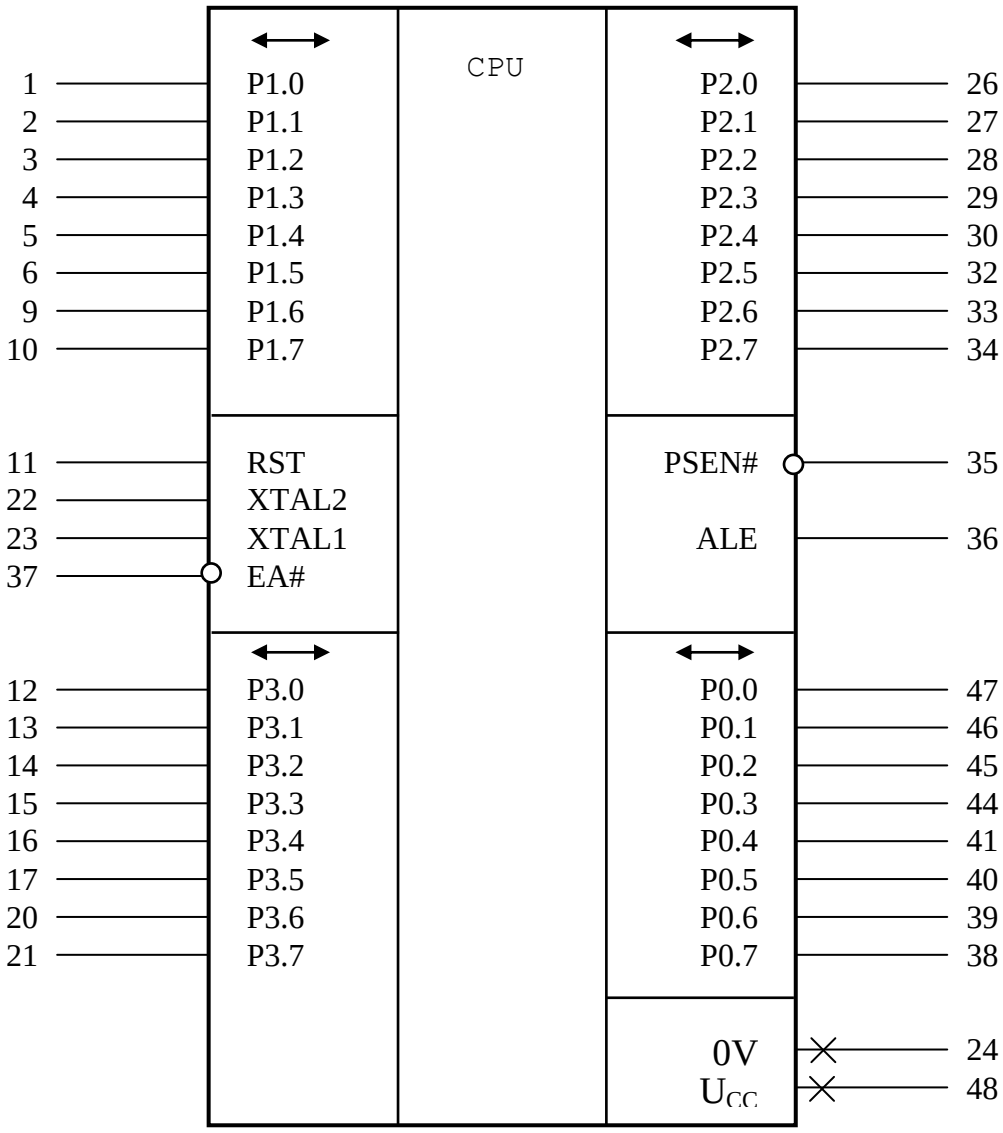


Рисунок 1 - Условное графическое обозначение ИМС

Таблица 1 – Функциональное назначение выводов микросхем

Номер вывода	Обозначение вывода	Функциональное назначение вывода
1-6, 9, 10	P1.0-P1.7	8-разрядный двунаправленный порт P1 с дополнительными функциями:
1	P1.0	вход таймера/счетчика 2: T2
2	P1.1	вход триггера выборки-перезагрузки таймера/счетчика 2: T2EX
3	P1.2	-
4	P1.3	-
5	P1.4	вход выбора последовательного периферийного интерфейса ППИ: SS#
6	P1.5	выход/вход данных ППИ: MOSI
9	P1.6	ввод-вывод данных ППИ: MISO
10	P1.7	ввод-вывод тактового сигнала ППИ: SCK
11	RST	сигнал общего сброса
12-17, 20, 21	P3.0-P3.7	8-разрядный двунаправленный порт P3 с дополнительными функциями:
12	P3.0	последовательные данные приемника RxD
13	P3.1	последовательные данные передатчика TxD
14	P3.2	вход внешнего прерывания 0: INT0#
15	P3.3	вход внешнего прерывания 1: INT1#
16	P3.4	вход таймера/счетчика 0: T0
17	P3.5	вход таймера/счетчика 1: T1
20	P3.6	выход стробирующего сигнала при записи во внешнюю память данных: WR#
21	P3.7	выход стробирующего сигнала при чтении из внешней памяти данных: RD#
22 23	XTAL2 XTAL1	выводы для подключения кварцевого резонатора
24	0V	общий вывод
26-30, 32-34	P2.0-P2.7	8-разрядный двунаправленный порт P2. Выход адреса A8-A15 в режиме работы с внешней памятью
35	PSEN#	разрешение программной памяти
36	ALE	выходной сигнал разрешения фиксации адреса. При программировании РПЗУ подается сигнал PROG#
37	EA#	блокировка работы с внутренней памятью. При программировании РПЗУ подается сигнал U_{PR}
38-41, 44-47	P0.7-P0.0	8-разрядный двунаправленный порт P0. Шина адреса/данных при работе с внешней памятью
48	U_{CC}	вывод питания от источника напряжения плюс 5 В
Примечание - Выводы 7, 8, 18, 19, 25, 31, 42, 43 не задействованы.		

Таблица 2 - Электрические параметры микросхем при приемке и поставке

Наименование параметра, единица измерения	Буквенное обозначение	Норма		Режим измерения
		мин.	макс.	
1	2	3	4	5
1 Выходное напряжение низкого уровня по выводам P1.0 – P1.7, P2.0 – P2.7, P3.0 – P3.7, В	U_{OL1}		0,5	I_{OL}=1,6 мА
2 Выходное напряжение низкого уровня по выводам P0.0 – P0.7, ALE, PSEN#, В	U_{OL2}		0,5	I_{OL}=3,2 мА
3 Выходное напряжение высокого уровня по выводам P1.0 – P1.7, P2.0 – P2.7, P3.0 – P3.7, ALE, PSEN#, В	U_{OH1}	$0,9U_{CC}$ 2,4		I_{OH}= –10 мкА I_{OH}= –60 мкА
4 Выходное напряжение высокого уровня по выводам P0.0 – P0.7, В	U_{OH2}	$0,9U_{CC}$ 2,4		I_{OH}= –80 мкА I_{OH}= –800 мкА
5 Входной ток низкого уровня по выводам P1.0 – P1.7, P2.0 – P2.7, P3.0 – P3.7, мкА	I_{IL}		|–50|	U_{IN}= 0,45 В
6 Токи утечки на входе по выводам P0.0 – P0.7, EA#, мкА	I_{ILH} I_{ILL}		|±10|	0,45 В< U_{IN} < U_{CC}
7 Ток потребления по выводу U_{CC} в активном режиме, мА	I_{OCC1}		25	U_{CC}= 5,5 В f_C=12 МГц
8 Ток потребления по выводу U_{CC} в режиме пониженного потребления, мА	I_{OCC2}		6,5	U_{CC}= 5,5 В f_C =12 МГц
9 Ток потребления по выводу U_{CC} в режиме хранения, мкА	I_{CCS}		100 40	U_{CC}= 6 В U_{CC}= 3 В

Окончание таблицы 2

1		2	3	4	5
10 Функциональный контроль		ФК	–	–	U_{CC}= (4,5 – 5,5) В f_C= 12 МГц
11 Время задержки распространения сигнала ALE относительно сигнала XTAL1, нс		t_{PALE}	–	50	U_{CC}= 4,5 В U_{IN}= 3,0 В U_{IN}= 0 В f_C= 12 МГц
12 Время задержки распространения сигнала PSEN# относительно сигнала XTAL1, нс		t_{PPSEN}	–	50	U_{CC}= 4,5 В U_{IN}= 3,0 В U_{IN}= 0 В f_C= 12 МГц
13 Время переключения на выходе ALE, нс	время нарастания	t_{rALE}	–	20	U_{CC}= 4,5 В U_{IN}= 3,0 В U_{IN}= 0 В f_C= 12 МГц
	время спада	t_{fALE}			
14 Время переключения на выходе PSEN#, нс	время нарастания	t_{rPSEN}	–	20	U_{CC}= 4,5 В U_{IN}= 3,0 В U_{IN}= 0 В f_C= 12 МГц
	время спада	t_{fPSEN}			
15 Функциональный контроль		$ФК_{SP\ Flash}$	–	–	U_{CC}= (4,5 – 5,5) В f_C= 4 МГц

Примечания

1 Нормы параметров приведены в диапазоне питающего напряжения U_{CC} = (5 ± 0,5) В.

2 Показатели по пунктам 11, 12 гарантируют работоспособность микросхем на частоте f_C = 24 МГц.

3 Функциональный контроль Flash – памяти ($ФК_{SP\ Flash}$) проводится в соответствии с алгоритмом, приведенным в «Программе и методике функционального контроля» КФДЛ.431281.032ПМ.

Таблица 3 – Значения предельно допустимых электрических режимов эксплуатации в диапазоне рабочих температур от минус 60 до плюс 85 °С

Наименование параметра, режима, единица измерения	Буквенное обозначение параметра	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, за исключением вывода EA#, В	U_{IL1}	–0,5	$0,2U_{CC}$ –0,1	–1,0	–
3 Входное напряжение низкого уровня по выводу EA#, В	U_{IL2}	–0,5	$0,2U_{CC}$ –0,3	–1,0	–
4 Входное напряжение высокого уровня по выводам, за исключением выводов XTAL1, RST, В	U_{IH1}	$0,2U_{CC}$+0,9	U_{CC} + 0,5	–	7,5
5 Входное напряжение высокого уровня по выводам XTAL1, RST, В	U_{IH2}	$0,7U_{CC}$	U_{CC} + 0,5	–	7,0
6 Период следования импульсов тактового сигнала, нс	T_C	41,6	–	–	–
7 Емкость нагрузки по выводам P0.0 – P0.7, ALE, PSEN#, пФ	C_{L1}	–	100		150*
8 Емкость нагрузки по выводам P1.0 – P1.7, P2.0 – P2.7, P3.0 – P3.7, пФ	C_{L2}	–	80	–	130*
9 Длительность фронтов тактового сигнала, нс для входа XTAL1, для остальных входов	$t_{HL/LH}$	 – –	 10 20	 – –	 40* 80*
10 Напряжение программирования на выводе EA#, В	U_{PP}	11,5	12,5	–	–

Примечание – Время действия предельных режимов по пунктам 1 – 5 не более 10 мс.

* При данных значениях C_{L1}, C_{L2}, $t_{HL/LH}$ динамические параметры не гарантируются.

Таблица 4 − Динамические параметры режима эксплуатации микросхем

Наименование параметра, единица измерения	Буквенное обозначение	Норма		Условное буквенное обозначение на диаграммах рис. 4 – 6
		мин.	макс.	
1	2	3	4	5
1 Время длительности сигнала ALE, нс	$t_W(ALE_H)$	$2T_C$–40	–	t_1
2 Время установления сигналов адреса относительно сигнала ALE при переходе его из состояния высокого уровня в состояние низкого уровня, нс	$t_{SU}(ALE_{HL}\text{-A})$	T_C – 15	–	t_2
3 Время сохранения сигналов адреса относительно сигнала ALE при переходе его из состояния высокого уровня в состояние низкого уровня, нс	$t_V(ALE_{HL}\text{-A})$	T_C – 20	–	t_3
4 Время задержки сигналов команды относительно сигнала ALE при переходе его из состояния высокого уровня в состояние низкого уровня, нс	$t_d(ALE_{HL}\text{-I})$	–	$4T_C$ – 65	t_4
5 Время задержки сигнала PSEN# низкого уровня относительно сигнала ALE при переходе его из состояния высокого уровня в состояние низкого уровня, нс	$t_d(ALE_{HL}\text{-}PSEN_L)$	T_C –15	–	t_5
6 Время длительности сигнала PSEN# низкого уровня, нс	$t_W(PSEN_L)$	$3T_C$ –20	–	t_6
7 Время задержки сигналов команды относительно сигнала PSEN# при переходе его из состояния высокого уровня в состояние низкого уровня, нс	$t_d(PSEN_{HL}\text{-I})$	–	$3T_C$–45	t_7
8 Время удержания сигналов команды относительно сигнала PSEN# при переходе его из состояния низкого уровня в состояние высокого уровня, нс	$t_H(PSEN_{LH}\text{-I})$	0	–	t_8

Продолжение таблицы 4

1	2	3	4	5
9 Время задержки сигналов адреса относительно сигнала PSEN# при переходе его из состояния низкого уровня в состояние высокого уровня, нс	$t_d(PSEN_{LH}\text{-}A)$	$T_C - 5$	–	t_9
10 Время задержки сигналов команды относительно сигналов адреса, нс	$t_d(A\text{-}I)$	–	$5T_C - 55$	t_{10}
11 Время длительности сигнала RD# низкого уровня, нс	$t_W(RD_L)$	$6T_C - 100$	–	t_{11}
12 Время длительности сигнала WR# низкого уровня, нс	$t_W(WR_L)$	$6T_C - 100$	–	t_{12}
13 Время задержки сигналов входных данных относительно сигнала RD# при переходе из состояния высокого уровня в состояние низкого уровня, нс	$t_d(RD_{HL}\text{-}DI)$	–	$5T_C - 90$	t_{13}
14 Время удержания сигналов входных данных относительно сигнала RD# при переходе его из состояния низкого уровня в состояние высокого уровня, нс	$t_H(RD_{LH}\text{-}DI)$	0	–	t_{14}
15 Время задержки сигналов входных данных относительно сигнала ALE при переходе из состояния высокого уровня в состояние низкого уровня, нс	$t_d(ALE_{HL}\text{-}DI)$	–	$8T_C - 150$	t_{15}
16 Время задержки сигналов входных данных относительно сигналов адреса, нс	$t_d(A\text{-}DI)$	–	$9T_C - 165$	t_{16}
17 Время задержки сигнала RD# низкого уровня относительно сигнала ALE при переходе его из состояния высокого уровня в состояние низкого уровня, нс	$t_d(ALE_{HL}\text{-}RD_L)$	$3T_C - 50$	$3T_C + 50$	t_{17}
18 Время задержки сигнала WR# низкого уровня относительно сигнала ALE при переходе его из состояния высокого уровня в состояние низкого уровня, нс	$t_d(ALE_{HL}\text{-}WR_L)$	$3T_C - 50$	$3T_C + 50$	t_{18}

Продолжение таблицы 4

1	2	3	4	5
19 Время установления сигналов адреса относительно сигнала RD# при переходе его из состояния высокого уровня в состояние низкого уровня, нс	$t_{SU}(RD_{HL}\text{-}A)$	$4T_C - 75$	–	t_{19}
20 Время установления сигналов адреса относительно сигнала WR# при переходе его из состояния высокого уровня в состояние низкого уровня, нс	$t_{SU}(WR_{HL}\text{-}A)$	$4T_C - 75$	–	t_{20}
21 Время задержки сигнала ALE высокого уровня относительно сигнала RD# при переходе его из состояния низкого уровня в состояние высокого уровня, нс	$t_d(RD_{LH}\text{-}ALE_H)$	$T_C - 20$	$T_C + 25$	t_{21}
22 Время задержки сигнала ALE высокого уровня относительно сигнала WR# при переходе его из состояния низкого уровня в состояние высокого уровня, нс	$t_d(WR_{LH}\text{-}ALE_H)$	$T_C - 20$	$T_C + 25$	t_{22}
23 Время установления сигналов выходных данных относительно сигнала WR# при переходе его из состояния высокого уровня в состояние низкого уровня, нс	$t_{SU}(WR_{HL}\text{-}DO)$	$T_C - 20$	–	t_{23}
24 Время установления сигналов выходных данных относительно сигнала WR# при переходе его из состояния низкого уровня в состояние высокого уровня, нс	$t_{SU}(WR_{LH}\text{-}DO)$	$7T_C - 120$	–	t_{24}
25 Время сохранения сигналов выходных данных относительно сигнала WR# при переходе его из состояния низкого уровня в состояние высокого уровня, нс	$t_V(WR_{LH}\text{-}DO)$	$T_C - 20$	–	t_{25}

Окончание таблицы 4

1	2	3	4	5
26 Время сохранения сигналов адреса относительно сигнала PSEN# при переходе его из состояния низкого уровня в состояние высокого уровня, нс	t_V ($PSEN_{LH}$-A)	T_C–20	–	t_{26}
27 Время сохранения сигналов адреса относительно сигнала RD# при переходе его из состояния низкого уровня в состояние высокого уровня, нс	t_V (RD_{LH}-A)	T_C–20	–	t_{27}
28 Время сохранения сигналов адреса относительно сигнала WR# при переходе его из состояния низкого уровня в состояние высокого уровня, нс	t_V (WR_{LH}-A)	T_C–20	–	t_{28}

Примечания

1 Нормы динамических параметров приведены в диапазоне питающих напряжений U_{CC} = (4,5 – 5,5) В.

2 Емкости нагрузки по выводам ALE, PSEN#, P0.0 – P0.7 – C_{L1} = 100 пФ. Емкости нагрузки по выводам P2.0 – P2.7, P3.6 (WR#), P3.7 (RD#) – C_{L2} = 80 пФ.

3 Условные обозначения сигналов:

- A – сигналы адреса (выходные);
- I – сигналы команды (входные);
- DI – сигналы входных данных;
- DO – сигналы выходных данных.

4 Допускается уменьшение длительности сигнала PSEN# до величины ($2T_C$–20) нс при холостом цикле чтения кода команды, следующей за инструкцией MOVX, что не отражается на ходе выполнения программы из внешней памяти.

3 Особенности микроконтроллера

Микросхемы имеют следующие основные особенности:

- 8-разрядное микроконтроллерное ядро, оптимизированное для приложений управления;
- 12 Кбайт встроенной программируемой (ISP) Flash-памяти программ с возможностью загрузки через последовательный SPI интерфейс;
- 2 Кбайт встроенной ЭСППЗУ памяти данных с программным и SPI доступом;
- три уровня защиты памяти программ;
- внутреннее ОЗУ (256 × 8) бит;
- 32 программируемые линии ввода-вывода;

- три 16-разрядных таймера/счетчика;
- девять источников прерывания;
- последовательный порт UART с обнаружением покадровых ошибок и автоматическим распознаванием адреса;
- последовательный интерфейс SPI (двойная буферизация Запись/Считывание);
- режимы работы на пониженной мощности в режиме холостого хода и в режиме пониженного потребления;
- выход по прерыванию из режима пониженного потребления;
- программируемый следящий таймер;
- сдвоенный указатель данных;
- гибкое внутрисистемное программирование (ISP) (байтовый и страничный режимы);
- четырехуровневый контроллер прерывания;
- возможность программирования (×2) внутренней тактовой частоты.

Микросхема представляет собой быстродействующий, экономичный 8-разрядный КМОП микроконтроллер с 12 К байт Flash памятью программ и 2 К байт ЭСППЗУ памятью данных. Внутренняя Flash память программ может программироваться внутрисистемно с помощью последовательного SPI интерфейса (последовательное программирование), или используя обычный программатор для энергонезависимой памяти (параллельное программирование).

Микросхема обеспечивает работу с частотой вплоть до нулевой и поддерживает два выбираемых программно режима экономии мощности. Режим холостого хода останавливает ЦПУ, при этом сохраняется работоспособность ОЗУ, таймеров/счетчиков, последовательного порта и системы прерывания. Режим пониженной мощности обеспечивает сохранность содержимого ОЗУ, однако приостанавливает генератор, отключая все остальные функции, имеющиеся на кристалле, до следующего внешнего прерывания или аппаратного сброса. Встроенная Flash и ЭСППЗУ

память имеет, как уже указывалось, доступ через последовательный SPI интерфейс.

4 Описание устройства

4.1 Структура и организация микросхем

На рисунке 2 приведена схема электрическая структурная ИМС 1882ВЕ53У. Обозначения и сокращения, используемые на рисунке 2:

0V – общий вывод.

U_{CC} – вывод питания от источника напряжения плюс 5 В.

XTAL1, XTAL2 - входы для подключения кварцевого резонатора.

RST - вход общего сброса.

PSEN# - разрешение внешней памяти программ; выдается только при обращении к внешнему ПЗУ.

ALE - строб адреса внешней памяти.

EA# - отключение внутренней программной памяти; уровень 0 на этом входе заставляет микроконтроллер выполнять программу только внешнего ПЗУ, игнорируя внутреннее (если последнее имеется).

Порт 0 (P0) - 8-битный двунаправленный порт ввода-вывода информации; при работе с внешними ОЗУ и ПЗУ по линиям порта в режиме временного мультиплексирования выдается адрес внешней памяти, после чего осуществляется передача или прием данных.

Порт 1 (P1) - 8-битный двунаправленный порт ввода-вывода, каждый разряд порта может быть запрограммирован как на ввод, так и на вывод, независимо от состояния других разрядов.

Порт 2 (P2) - 8-битный двунаправленный порт, аналогичный P1, кроме того, выводы порта используются для выдачи адресной информации при обращении к внешней памяти программ или данных (если используется 16-битовая адресация последней). Выводы порта используются для ввода в МК старших разрядов адреса.

Порт 3 (P3) - 8-битный двунаправленный порт, аналогичный P1, кроме того, выводы порта могут выполнять ряд альтернативных функций, которые описаны ниже.

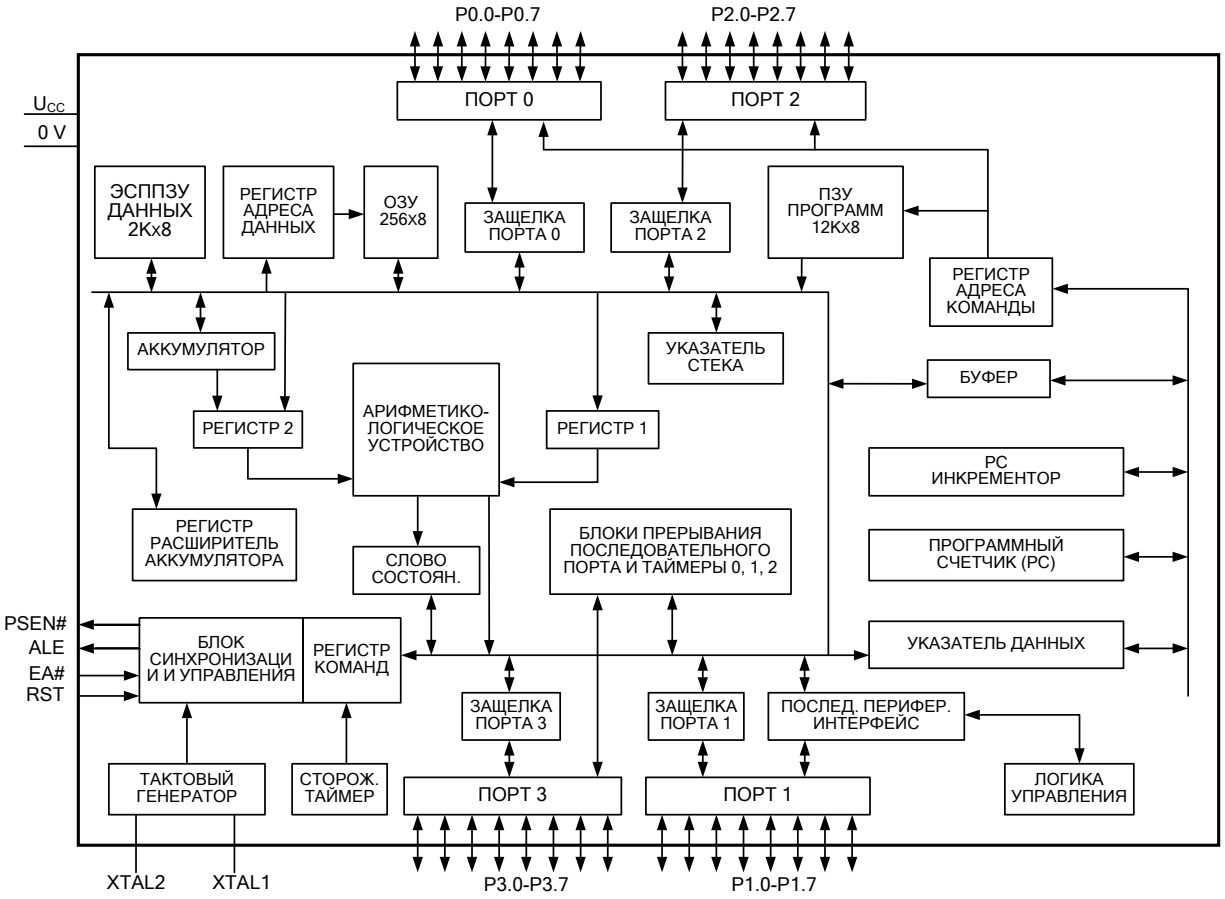


Рисунок 2 - Схема электрическая структурная микросхемы

Регистр-указатель стека SP - 8-битный. Он может адресовать любую область внутренней памяти данных. Стек "наращивается", 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.

Наличие высокого уровня на выводе RST в течение, по крайней мере, двух машинных циклов, пока генератор продолжает работать, вызывает переустановку устройства. В процессе инициализации ИМС после сигнала сброса или при включении питающего напряжения в SP заносится код 07H. Это означает, что первый элемент программы будет располагаться в ячейке памяти с адресом 08H. Описание всех функциональных блоков

микроконтроллера и управляющих регистров приводится в соответствующих разделах руководства.

4.2 Описание выводов

RST - вход сброса. Наличие высокого уровня на этом выводе в течение, по крайней мере, двух машинных циклов, пока генератор продолжает работать, вызывает переустановку устройства.

ALE/PROG - разрешение на защелкивание адреса (Address Latch Enable). ALE/PROG – это выходной импульс для защелкивания младших разрядов адреса (по его заднему фронту) во время доступа к внешней памяти. Этот вывод является также входом программирующего импульса (PROG) во время программирования Flash памяти. В обычном режиме ALE выдается с постоянной частотой, равной 1/6 частоты тактового генератора и может использоваться для внешнего задания времени или тактирования. Однако следует иметь в виду, что один импульс ALE пропускается во время каждого доступа к памяти внешних данных. По желанию, ALE можно отключить путем установки разряда 0 для AUXR SFR в адресе 8EH. С установленным разрядом ALE будет активен только во время команды MOVX или MOVC. В противном случае, на выводе формируется высокий уровень. Описание регистра AUXR приведено в таблице 5.

Таблица 5 - Описание регистра AUXR

-	-	-	-	-	-	I_Pwd_Exit	DISALE
7	6	5	4	3	2	1	0

Символ	Функция
DISALE	Если DISALE = 0, на выходе ALE формируется частота, равная 1/6 частоты синхронизации (за исключением периода MOVX, когда один импульс ALE отсутствует). Когда DISALE = 1, то ALE активен только во время команды MOVX или MOVC.
I_Pwd_Exit	При установленном разряде обеспечивается выход по прерыванию из режима пониженной мощности (по приходу переднего фронта сигнала прерывания). При сброшенном разряде выход по прерыванию осуществляется после интервала времени (обычно 2 мс), отсчитываемого от заднего фронта сигнала прерывания.

PSEN - разрешение внешней памяти программ, выдается только при обращении к внешнему ПЗУ программ (Program Store Enable). PSEN является стробирующим сигналом считывания для внешней памяти программ (active low). Когда микроконтроллер выполняет код из внешней памяти программ, то PSEN за каждый машинный цикл активизируется дважды, за исключением, когда дважды PSEN пропускается во время каждого доступа к внешней памяти данных.

EA#/V_{PP} – выбор режима работы с внутренней памятью программ или с внешней. Вывод EA# должен быть подсоединен к GND для того, чтобы разрешить устройству получать код из ячеек внешней памяти программ с адресами, начинающимися с 0000H и заканчивающимися вплоть до FFFFH. Для исполнения внутренней программы EA# должен быть подсоединен к V_{CC}. На этот вывод поступает также напряжение 12 В (V_{PP}), формирующее разрешение во время параллельного программирования Flash памяти, когда выбирается программирование 12 вольт.

XTAL1 и XTAL2 – выводы для подключения кварцевого резонатора, при этом XTAL1 является входом для инвертирующего усилителя генератора и входом тактовых сигналов при работе в режиме внешней синхронизации, XTAL2 является выходом для инвертирующего усилителя генератора.

Микросхема содержит 4 порта ввода-вывода. Их описание приведено в подразделе «Организация портов ввода-вывода».

4.3 Организация портов ввода - вывода

Все 4 порта ИМС - двунаправленные. Каждый из них содержит регистр-защелку (SFR P0…P3), выходную цепь и входной буфер. Выводы портов помимо обычного ввода информации и её вывода могут выполнять альтернативные функции. Описание этих функций приведено в таблице 1 "Функциональное назначение выводов микросхем".

Альтернативные функции могут быть активизированы только в случае, если в соответствующие биты SFR портов предварительно занесены единицы.

Порт 0. Порт 0 является 8-разрядным двунаправленным портом ввода-вывода с «открытым стоком». На порт 0 поступают также байты кодов во время программирования Flash памяти и выдаются байты кодов во время верификации программы. Во время верификации программы необходимы внешние активные нагрузки.

Нагрузочный транзистор линии порта P0 включен только тогда, когда порт выводит единицу при обращении к внешней памяти. В остальных случаях нагрузочный транзистор отключен. Таким образом, при работе в режиме обычного ввода-вывода информации выводы порта P0 представляют из себя транзисторы с открытым стоком. Запись единицы в соответствующий бит SFR отключает транзистор, что приводит к тому, что вывод оказывается под плавающим потенциалом. Это позволяет использовать линии порта P0 как выводы с высокоимпедансным состоянием.

Порт 1. Порт 1 является 8-разрядным двунаправленным портом ввода-вывода с внутренними активными нагрузками. Выходные буферы Порта 1 могут работать в режиме ввода-вывода. Когда Порт 1 переводится в режим ввода, его выводы переходят в верхний логический уровень с помощью слабых внутренних активных нагрузок. Поэтому при подаче внешних низких логических уровней потребляется небольшой ток благодаря внутренним активным нагрузкам. Некоторые выводы Порта 1 обеспечивают дополнительные альтернативные функции.

Порт 2. Порт 2 является 8-разрядным двунаправленным портом ввода-вывода с внутренними активными нагрузками. Порт 2 передает старший байт адреса во время выборки программы из внешней памяти программ, а также во время доступа к памяти внешних данных, которые используют 16-разрядные адреса (MOVX @ DPTR). В этом режиме применения Порт 2 использует мощные внутренние активные нагрузки когда выводятся логические единицы. Во время доступов к памяти внешних данных, которая использует 8-разрядные адреса (MOVX @ RI), Порт 2 выводит содержимое Регистра Специальной Функции (Special Function Register) P2.

Выводы порта «Порт 2» используются также во время программирования и верификации Flash памяти.

Выходные цепи P0 и P2 вместе с входным буфером P0 используются при обращении к внешней памяти. При этом на выходах P0 младший байт адреса внешней памяти мультиплексируется с вводимым/выводимым байтом. Выходы P2 содержат старший байт адреса внешней памяти, если адрес 16-разрядный. При использовании 8-разрядного адреса на выводах P2 присутствует информация из SFR P2.

Порт 3. Порт 3 является 8-разрядным двунаправленным портом ввода-вывода с внутренними активными нагрузками. Когда на выводы Порта 3 записываются единицы, то они переходят в верхний логический уровень с помощью небольших внутренних активных нагрузок и могут использоваться как входы. Выводы Порта 3 выполняют также и альтернативные функции, указанные в таблице 1. Кроме того, в режиме программирования и верификации Flash памяти на выводах Порта 3 формируются управляющие сигналы.

При обращении к внешней памяти в SFR P0 автоматически заносятся единицы во все биты. Информация в SFR P2 при этом остается неизменной. Для того, чтобы выводы порта P3 выполняли альтернативные функции, необходимо в SFR P3 также занести единицы во все биты.

Выходы триггеров SFR портов P1...P3 выполнены на полевых транзисторах, имеющих внутреннюю нагрузку, в то время как выходы триггеров SFR P0 выполнены на транзисторах с открытым стоком. Каждая линия любого из портов может независимо использоваться как для ввода информации, так и для вывода (для линий портов P0 и P2 это справедливо тогда, когда они не используются для обращения к внешней памяти). 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 выхода микросхемы или транзистора, нагруженного на этот вывод. Поэтому порты

P1...P3 называются "квазидвунаправленные", в отличие от двунаправленного порта P0, переводимого в режиме ввода информации в высокоимпедансное состояние.

Каждый из выводов портов P1...P3 может быть нагружен на стандартные ТТЛ - входы. При использовании в качестве обычного порта каждый из выводов должен быть подключен к шине питания через резистор номиналом 10...20 кОм. При использовании P0 в качестве шины адреса - данных необходимость в этом отпадает.

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.

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

Механизм "чтение – модификация – запись" реализован в командах ANL, ORL, XRL, JBC, CPL, INC, DEC, DJNZ, CLR, SETB, MOV.

4.4 Регистры специальных функций (SFR)

К адресному пространству памяти данных примыкает адресное пространство регистров специальных функций (SFR). Эти регистры занимают часть 256-байтового адресного пространства. Назначение регистров специальных функций приведено в таблице 6. Карта встроенной матрицы памяти, называемая пространством регистров специальных функций (SFR), и адреса, по которым расположены эти регистры, дана в таблице 7.

Таблица 6 – Назначение регистров специальных функций

Символ	Назначение
1	2
* B	Регистр-расширитель аккумулятора
* ACC	Аккумулятор
* PSW	Слово состояния программы
SPCR	Регистр управления SPI
T2CON	Регистр управления-статуса таймера 2
T2MOD	Регистр режимов таймера 2
RCAP2L	Младший байт регистра захвата/перезагрузки таймера 2
RCAP2H	Старший байт регистра захвата/перезагрузки таймера 2
TL2	Младший байт таймера 2
TH2	Старший байт таймера 2
* IP	Регистр приоритетов
SADEN	Регистр маски адресов приемопередатчика
* P3	Порт 3 (SFR P3)
IPH	Старший регистр приоритета прерываний
* IE	Регистр маски прерываний
SADDR	Регистр автоматического распознавания адреса
SPSR	Регистр состояния (статуса) SPI
* P2	Порт 2 (SFR P2)
WDTRST	Регистр режима работы аппаратной части для WDT
WDTCON	Регистр управления таймера WDT
* SCON	Регистр управления приемопередатчиком
SBUF	Регистр данных приемопередатчика (буферный регистр)
* P1	Порт 1 (SFR P1)
EECON	Регистр управления ЭСППЗУ данных
* TCON	Регистр управления-статуса таймеров
TMOD	Регистр режимов таймеров-счетчиков
TL0	Младший байт таймера 0
TL1	Младший байт таймера 1
TH0	Старший байт таймера 0
TH1	Старший байт таймера 1
AUXR	Управление режимом ALE и выходом из прерывания
CLKREG	Управление частотой внутренней синхронизации (×/2)
* P0	Порт 0 (SRF P0)
SP	Регистр-указатель стека
DP0L	Младший байт основного регистра-указателя данных DPTR
DP0H	Старший байт основного регистра-указателя данных DPTR
DP1L	Младший байт вспомогательного регистра-указателя данных DPTR
DP1H	Старший байт вспомогательного регистра-указателя данных DPTR
SPDR	Регистр данных SPI
PCON	Регистр управления мощностью
Примечание – Регистры, имена которых отмечены знаком *, допускают адресацию своих отдельных бит при выполнении команд из группы команд с битами.	

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 информацию в эти незанятые позиции.

Таблица 7 – Регистры специальных функций МК и состояние после RESET

Начальный адрес	Регистры специальных функций МК и их состояние после RESET								Конечный адрес
1	2	3	4	5	6	7	8	9	10
0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000					SPCR 000001xx			0D7H
0C8H	T2CON 00000000	T2MOD xxxxxx00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000			0CFH
0C0H									0C7H
0B8H	IP xx000000	SADEN 00000000							0BFH
0B0H	P3 11111111							IPH xx000000	0B7H
0A8H	IE 0x000000	SADDR 00000000	SPSR 00xxxxxx						0AFH
0A0H	P2 00000000						WDTRST xxxxxxxx	WDTCON 00000000	0A7H
098H	SCON 00000000	SBUF xxxxxxxx							09FH
090H	P1 11111111						EECON xx000011		097H
088H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000	AUXR xxxxxxx0	CLKREG 00xx0000	08fH
080H	P0 11111111	SP 00000111	DP0L 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000	SPDR xxxxxxxx	PCON 0xxx0000	087H
	Примечание – x – неопределенное значение								

4.5 Слово состояния программы

Аккумулятор является источником операнда и местом фиксации результата при выполнении ряда команд. Только с использованием аккумулятора могут быть выполнены команды сдвига, проверки на нуль и ряд других команд. Система команд содержит также большое количество команд пересылок, логических команд и переходов, не использующих аккумулятор. При выполнении определенных команд в арифметико-логическом устройстве (АЛУ) формируются признаки результата - флаги,

которые фиксируются в регистре PSW. Перечень флагов, их символические имена и условия их формирования приведены в таблице 8.

Таблица 8 - Формат слова состояния программы PSW

<table>
<tr><td>Символ</td><td colspan="2">Позиция</td><td colspan="4">Имя и назначение</td></tr>
<tr><td>1</td><td colspan="2">2</td><td colspan="4">3</td></tr>
<tr><td>P</td><td colspan="2">PSW.0</td><td colspan="4">Флаг паритета. Устанавливается и сбрасывается аппаратно в каждом цикле команды и фиксирует нечетное/четное число единичных бит в аккумуляторе.</td></tr>
<tr><td>-</td><td colspan="2">PSW.1</td><td colspan="4">Не используется.</td></tr>
<tr><td>OV</td><td colspan="2">PSW.2</td><td colspan="4">Флаг переполнения. Устанавливается и сбрасывается аппаратно при выполнении арифметических команд.</td></tr>
<tr><td>RS0
RS1</td><td colspan="2">PSW.3
PSW.4</td><td colspan="4">Биты выбора используемого банка регистров. Могут быть изменены программным путем.</td></tr>
<tr><td colspan="3" rowspan="5"></td><td>RS0</td><td>RS1</td><td>Банк</td><td>Границы адресов ОЗУ</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>00H - 07H</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>08H - 0FH</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>10H - 17H</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>18H - 1FH</td></tr>
<tr><td>F0</td><td>PSW.5</td><td colspan="4">Флаг пользователя. Может быть установлен, сброшен или проверен программой пользователя.</td></tr>
<tr><td>AC</td><td>PSW.6</td><td colspan="4">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.</td></tr>
<tr><td>C</td><td>PSW.7</td><td colspan="4">Флаг переноса. Устанавливается и сбрасывается как аппаратно, так и программным путем</td></tr>
</table>

4.6 Организация ОЗУ, ПЗУ

Память программ и данных являются самостоятельными и независимыми друг от друга устройствами, адресуемыми различными командами. Объем памяти программ, расположенной на кристалле МК, равен 12 Кбайт. При обращении к внешней памяти программ МК всегда используется 16-разрядный адрес, что обеспечивает доступ к 64К ПЗУ. При выполнении команд переноса данных адресация ячейки памяти программ, из которой будут прочитаны данные, может осуществляться как с использованием счетчика PC, так и с использованием специального двухбайтового регистра-указателя данных DPTR.

Память данных, расположенная на кристалле, имеет объем 256 байт (ОЗУ) и (2К×8) бит EEPROM данных. Команды с прямой адресацией

обеспечивают доступ к пространству SFR. Например, команда MOV 0A0H, #data; (прямая адресация), обращается к SFR, расположенному по адресу 0A0H (т. е. к P2).

Команда с косвенной адресацией обращается к старшим 128 байтам ОЗУ, например MOV @R0, #data; (косвенная адресация), где R0 содержит 0A0H, обращается к байту с адресом 0A0H в пространстве старших 128 байтов ОЗУ. Работа стека организована с применением косвенной адресации (адрес используемой под стек ячейки памяти размещен в регистре SP), так что старшие 128 байт данных ОЗУ могут быть использованы для размещения в них стека МК.

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.

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. Верхние 128 байт ОЗУ занимают параллельное пространство для Регистров Специальных Функций (Special Function Registers или SFR). Это означает, что верхние 128 байт имеют те же адреса, что и пространство SFR, однако они физически отделены от пространства SFR.

Для того чтобы обеспечить организацию доступа как к внутреннему ЭСППЗУ, так и к внешней памяти данных, в SFR предусмотрены два 16-разрядных регистра указателей данных (Регистры сдвоенного указателя

данных или Dual Data Pointer Registers): DP0, расположенный в ячейках SFR регистров специальных функций с адресом 82H-83H, и DP1, расположенный по адресу 84H-85H. Разряд DPS = 0 в SFR EECON выбирает DP0, а разряд DPS = 1 выбирает DP1.

Необходимо устанавливать в разряде DPS соответствующее значение для доступа в соответствующий регистр указатель данных DP0 или DP1.

Встроенная ЭСППЗУ память данных выбирается путем установки разряда EEMEN в регистре EECON пространства SFR по адресу 96H. Диапазон адресов ЭСППЗУ памяти находится в области от 000H до 7FFH. Для доступа к ЭСППЗУ используются команды MOVX. Для доступа к внешней памяти данных с помощью команд MOVX разряд EEMEN необходимо установить в “0”.

Описание регистра управления ЭСППЗУ данных EECON приведено в таблице 9.

В режиме исполнения программы (с использованием команды MOVX) имеется возможность автоматического стирания на уровне байта. Это означает, что пользователь может обновлять или изменять одну позицию байта ЭСППЗУ в реальном масштабе времени, не влияя при этом на остальные байты.

Перед тем, как записать байт в ЭСППЗУ, разряд EEMWE в регистре EECON необходимо установить в “1”. Программное обеспечение пользователя должно переустановить разряд EEMWE в “0”, если не требуется дальнейшая запись ЭСППЗУ. Циклы записи ЭСППЗУ в режиме последовательного программирования самосинхронизирующиеся. Ход записи ЭСППЗУ можно наблюдать путем считывания разряда RDY/BSY# (только считывание) в SFR EECON. Значение RDY/BSY# = 0 означает, что программирование продолжается, а если RDY/BSY# = 1, то это означает, что цикл записи ЭСППЗУ завершен, и можно начинать другой цикл записи. Разряд EELD в EECON контролирует, будет ли следующая команда MOVX только загружать буфер записи ЭСППЗУ или же она фактически запустит цикл программирования. При установке значения EELD будет происходить только загрузка. До достижения последнего значения MOVX в данной

странице, состоящей из 32 байт, EELD необходимо очистить, чтобы после последнего MOVX можно будет одновременно программировать всю страницу. В результате время программирования 32 байт займет всего лишь порядка 4 мс, вместо 128 мс, которые потребовались бы при побайтовом программировании.

Регистр EECON содержит управляющие разряды для 2K байт встроенного на кристалле ЭСППЗУ данных. Он содержит также управляющие разряды для сдвоенного указателя данных.

Таблица 9 - Регистр управления ЭСППЗУ данных EECON

-	-	EELD	EEMWE	EEMEN	DPS	RDY/BSY#	WRTINH#
7	6	5	4	3	2	1	0

Символ	Функция
EELD	Бит разрешения загрузки ЭСППЗУ данных. Используется для выполнения режима записи страницы (Page Mode Write). Команда записи MOVX в ЭСППЗУ данных не инициирует цикл программирования, если установлен этот разряд, она будет только загружать данные в буфер изменяемых данных памяти данных ЭСППЗУ.
EEMWE	Бит разрешения записи в память данных ЭСППЗУ. При записи байта в ЭСППЗУ с помощью команды MOVX этот разряд должен быть установлен в «1». Пользовательское ПО должно установить этот разряд в «0» после завершения цикла записи ЭСППЗУ.
EEMEN	Бит разрешения доступа к встроенному ЭСППЗУ данных. Если EEMEN = 1, то команда MOVX с DPTR выполнит доступ к встроенному ЭСППЗУ вместо доступа к памяти внешних данных, если используемый адрес меньше чем 2K. Если EEMEN = 0 или если используемый адрес ≥ 2K, то MOVX с DPTR начнет доступ к памяти внешних данных.
DPS	Выбор регистра указателя данных. При DPS = 0 выбирается первый банк регистра указателя данных DP0. При DPS = 1 выбирается второй банк DP1.
RDY/BSY#	Флаг Готово/Занято для памяти данных ЭСППЗУ. Это разряд только считывания, который сбрасывается (очищается) аппаратным способом во время цикла программирования ЭСППЗУ. Он также устанавливается аппаратно, когда программирование завершено. RDY/BSY# сбрасывается после окончания команды MOVX, которая инициировала цикл программирования.
WRTINH#	Бит «Запись/Запрет» считывания (READ-ONLY). Очищается аппаратным способом, когда U_{CC} имеет слишком низкое значение для того, чтобы во встроенном ЭСППЗУ выполнился цикл программирования. Когда этот разряд очистится, то продолжающийся цикл программирования будет остановлен или начнется новый цикл программирования.

Последовательность записи данных в ЭСППЗУ показана на рисунке 3.

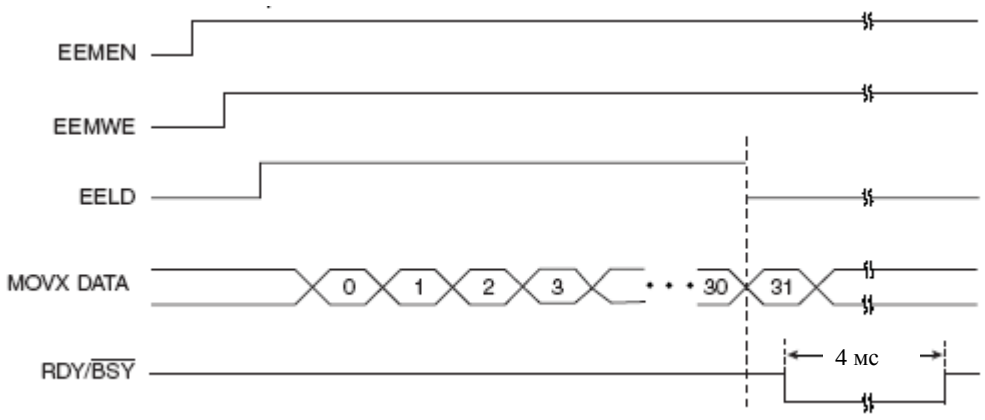


Рисунок 3 - Последовательность записи данных в ЭСППЗУ

4.7 Работа с внешней памятью

При обращении к внешней памяти данных ИМС становятся доступными 64К ОЗУ. Адресация внешнего ОЗУ осуществляется методом косвенной адресации при помощи регистров R0 и R1 или при помощи двухбайтового регистра DPTR.

Обращения к внешней памяти подразделяются на обращения к внешней памяти программ и обращения к внешней памяти данных. В первом случае для формирования сигнала, активирующего ПЗУ с программной памятью используется сигнал PSEN#. Во втором случае используются сигналы RD# и WR#, активирующие ОЗУ с данными.

Сигнал PSEN# (Program Store Enable) является стробирующим сигналом считывания для внешней памяти программ (active low). Когда микроконтроллер считывает из внешней памяти программ, то PSEN# за каждый машинный цикл активизируется дважды.

Если используется 16-битовый адрес, старшие 8 бит выводятся через порт P2, где они сохраняются в течение всего цикла обращения к внешней памяти. Отметим, что выходные буферы порта P2 имеют внутреннюю нагрузку, несколько отличную от P1 и P3, благодаря чему в SFR P2 при выводе адресной информации вовсе не обязательно защелкивать все единицы. Добавим также, что при выводе адресной информации информация

из SFR P2, хотя и не присутствует на выводах МК, но и не теряется, восстанавливаясь на выводах после окончания обращений к внешней памяти (если в процессе этих обращений SFR P2 не был модифицирован).

Если при обращении к внешней памяти данных используется 8-битный адрес, то на выводах порта остается та же информация, которая там была до начала обращения к внешней памяти. Это позволяет организовать постраничную адресацию внешней памяти данных.

Как уже отмечалось выше, на выводах порта P0 младший байт адреса мультиплексируется с данными. Сигналы адреса/данных задействуют оба полевых транзистора выходного буфера P0. Таким образом, в этом случае выводы P0 уже не являются выводами с открытым стоком и не требуют внешних нагрузочных элементов. Сигнал ALE используется для фиксации младшего байта адреса во внешнем регистре-защелке. Адресная информация достоверна в момент окончания сигнала ALE. Разрешение на защелку адреса (Address Latch Enable) ALE/PROG – это выходной импульс для фиксации младших разрядов адреса (по заднему фронту) во время доступа к внешней памяти. Этот вывод является также входом программирующего импульса (PROG) во время программирования Flash памяти. В обычном режиме ALE выдается с постоянной частотой, равной 1/6 частоты тактового генератора и может использоваться для внешнего задания времени или тактирования. Однако следует иметь в виду, что один импульс ALE пропускается во время каждого доступа к памяти внешних данных. По желанию, ALE можно отключить путем установки разряда 0 для AUXR SFR в адресе 8EH. С установленным разрядом ALE будет активен только во время команды MOVX или MOVC. В противном случае на выводе устанавливается высокий уровень.

Выводимый в цикле записи байт заносится в P0 непосредственно перед активацией сигнала WR# и остается неизменным до окончания этого сигнала. В цикле чтения данные на выводах P0 для достоверного считывания должны быть установившимися к моменту окончания сигнала RD#.

Во время обращения к внешней памяти CPU записывает 0FFH в SFR P0, разрушая этим самым хранимую там информацию. Это необходимо учитывать при использовании порта P0 для записи при работе с внешней памятью.

Обращение к внешней памяти программ возможно в двух случаях:

1 Сигнал EA# активен, т. е. имеет нулевой уровень.

2 Программный счетчик PC содержит число, больше 2FFFH.

Когда CPU работает с внешней памятью программ, все линии P2 используются для вывода старшего байта адреса и не могут быть использованы для обычного ввода-вывода информации. При этом, как отмечалось выше, в SFR P2 может быть занесена любая информация - адресная информация, выводимая через P2, не зависит от состояния его SFR.

При обращении к внешней памяти данных используются команды, выполняющиеся за два машинных цикла. В первом машинном цикле сигнал ALE возникает в момент S1P2 и в момент S4P2. По спаду этих сигналов во внешний регистр-защелку должна быть занесена адресная информация, выводимая по линиям порта 0 (DPL OR RI OUT - вывод младшего байта из регистра DPTR либо байта из регистра R0 или R1). При обращении к внешней памяти с использованием регистра DPTR на линиях порта 2 при этом появляется старший байт из регистра DPTR (DPH). Если происходит обращение к внешней памяти данных с использованием регистров R0 или R1, то на линиях порта появляется информация из его SFR (P2 SFR OUT). В момент S1P1 второго машинного цикла активизируется сигнал RD# или WR#.

При этом в режиме чтения линии порта переходят в высокоимпедансное состояние (FLOAT), и к моменту S3P1, когда МК считывает данные, они должны быть установившимися (DATA SAMPLED). В режиме записи в течение всего времени действия сигнала WR# на линиях портов сохраняется выводимая информация. В момент S1P1 второго машинного цикла, как при чтении, так и при записи, сигнал ALE отсутствует,

так как по линиям порта P0 идет обмен данными между МК и внешним ОЗУ, а не выводится адресная информация.

При обращении к внешней памяти программ сигнал ALE формируется как в момент S1P2, так и в момент S4P2, без пропусков. Поскольку при обращении к внешней памяти программ всегда используется двухбайтовый счетчик команд PC, то его младший байт выводится по линиям порта P0 и фиксируется во внешнем регистре-защелке по спаду ALE, а старший байт выводится по линиям порта P2 (PCH OUT). Роль строб - сигнала чтения в этом случае играет сигнал PSEN#.

При вводе информации через порты P0...P3 данные должны быть установившимися к моменту S5P1 и сохраняться неизменными до момента S5P2. При выводе информации из портов новые данные (NEW DATA) появляются на выводах МК в момент S1P1 цикла, следующего за циклом, в котором осуществлялась команда вывода.

Временные диаграммы работы с внешней памятью приведены на рисунках 4 – 6.

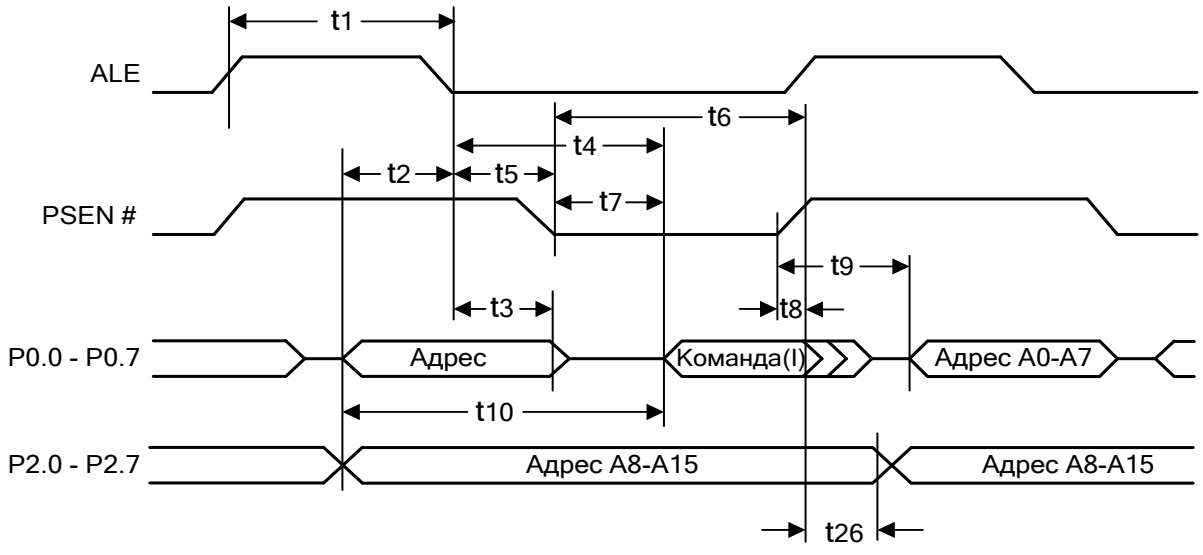


Рисунок 4 – Временные диаграммы сигналов микросхем в цикле «чтение памяти программ»

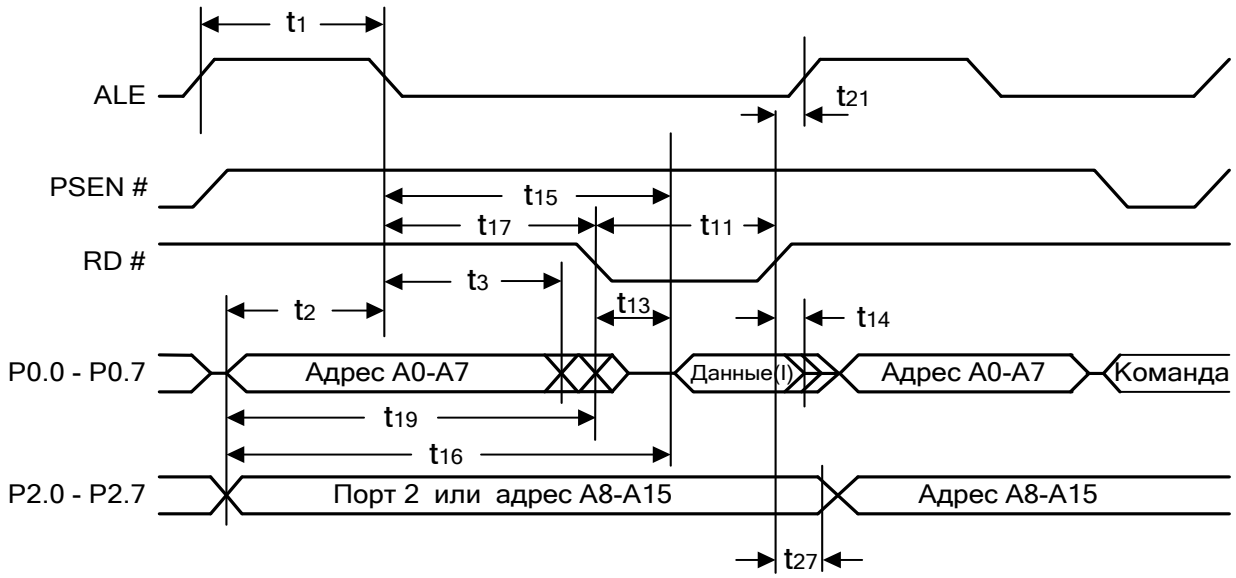


Рисунок 5 − Временные диаграммы сигналов микросхем в цикле «чтение памяти данных»

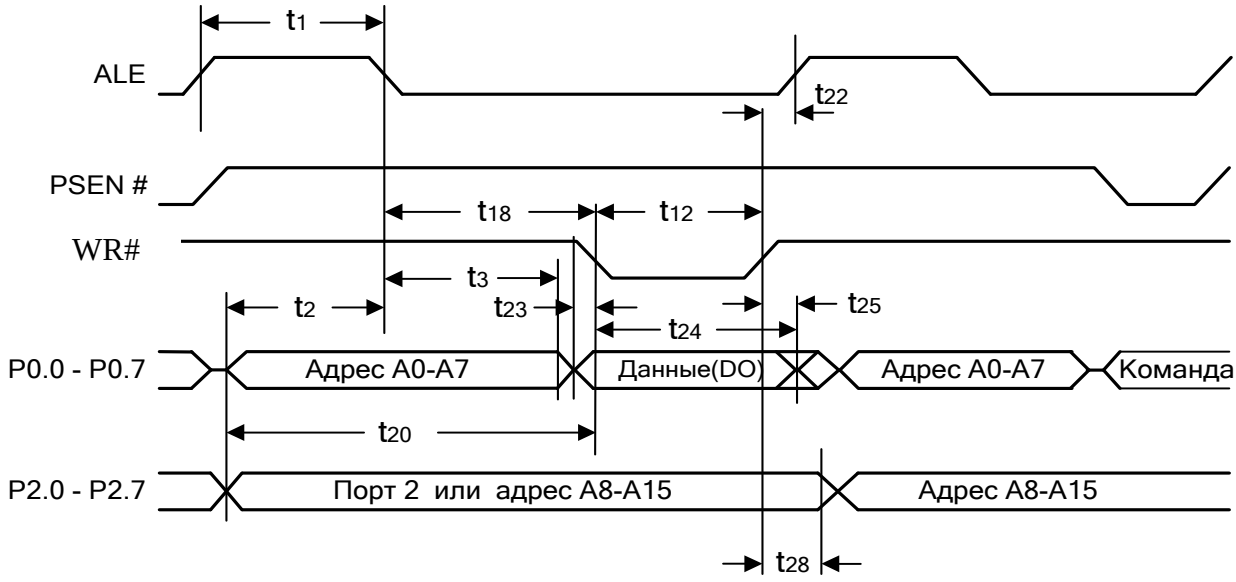


Рисунок 6 − Временные диаграммы сигналов микросхем в цикле «запись памяти данных»

4.8 Устройство управления и синхронизации. Тактовый генератор

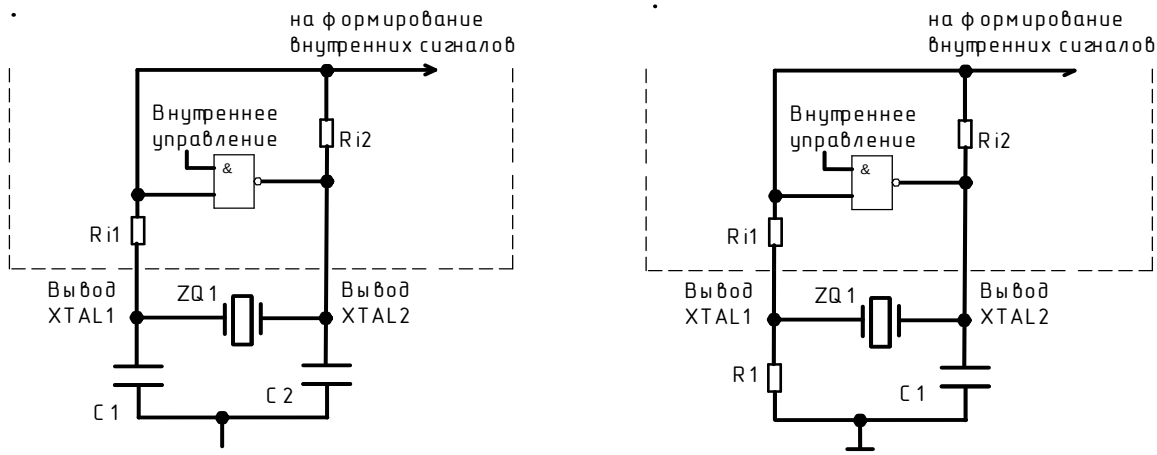
XTAL1 и XTAL2 являются, соответственно, входом и выходом инвертирующего усилителя, который используется в качестве генератора для устройства управления и синхронизации.

Кварцевый резонатор подключается к выводам XTAL1 и XTAL2 МК. 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). Весь машинный цикл состоит из 12 фаз, начиная от S1P1 и кончая S6P2. Внешними сигналами являются сигналы кварцевого резонатора и сигналы ALE. Сигналы ALE используются для защелкивания младшего байта адреса при обращении к внешней памяти.

При работе на частоте 24 МГц подавляющее большинство команд выполняется МК за (0,5 − 1) мкс.

Генератор можно использовать с внешним кварцевым резонатором. Рекомендуемые схемы включения приведены на рисунках 7а), 7б).



а) б)

Рисунок 7 – Тактовый генератор МК

На рисунке 7а) значения элементов следующие: C1 = C2 = (5 ± 5) пФ при f_C = 24 МГц. В случае, если при использовании кварцевого резонатора с частотой, близкой к максимальной, тактовый генератор не всегда запускается стабильно (даже при использовании «не гармониковых» или "fundamental mode" кварцев), рекомендуется замена конденсатора на XTAL1 резистором R1 с сопротивлением около 5 МОм, значение элемента C1 = (5 ± 5) пФ, как показано на рисунке 7б).

При работе с внешней синхронизацией сигнал внешнего синхрогенератора должен быть подан на вход XTAL1 (вход XTAL2 при этом не используется).

МК способен выключать свой генератор программным путем (записью 1 в бит PD регистра PCON).

Выбор источника синхронизации осуществляется битами конфигурации FUSE4, как показано в таблице 10.1.

Таблица 10.1 – Биты конфигурации

Название бита	Номер бита	Описание	Значение по умолчанию
FUSE4	4	Разрешение внешней синхронизации / Разрешение кварцевого тактового генератора	1
FUSE3	3	Разрешение / Запрет программирования строки пользователя	1
FUSE2	2	Разрешение встроенного делителя частоты на 2 / Блокировка встроенного делителя частоты на 2	1
FUSE1	1	Разрешение / Запрет последовательного программирования	0

В микроконтроллере имеется возможность выбора режима работы встроенного делителя на 2 тактовой частоты. Эта опция позволяет микроконтроллеру выполнять один машинный цикл за 6 периодов синхронизации вместо 12, что важно с точки зрения снижения электромагнитных помех (EMI). При блокировке встроенного делителя частоты внешняя тактовая частота не должна превышать 12 МГц. Опцию можно активизировать, как аппаратно с помощью конфигурационного бита FUSE2, так и программно. Программное управление осуществляется с помощью регистра CLKREG, описание которого приведено в таблице 10.2.

Таблица 10.2 – Регистр CLKREG

-	-	-	-	-	-	-	X2
7	6	5	4	3	2	1	0

Адрес CLKREG = 8FH Значение переустановки = XXXX XXX0B

Символ	Функция
X2	Если X2 = 0, то частота синхронизации (на выводе XTAL1) делится на 2 и используется в качестве системной частоты. Если X2 = 1, то деления на 2 нет и частота XTAL1 становится системной частотой. Это позволяет пользователю выбирать частоту кварцевого генератора из соображений снижения EMI, например, выбрать частоту 6 МГц вместо 12 МГц.

4.9 Таймеры/счетчики

В микроконтроллере имеется три таймера/счетчика: T/C0, T/C1 и T/C2. Они могут быть использованы как в качестве таймеров, так в качестве счетчиков внешних событий. Таймеры T/C0 и T/C1 микроконтроллера 1882ВЕ53У работают аналогично таймерам T/C0 и T/C1 микроконтроллеров Н1830ВЕ31, Н1830ВЕ51, а также 1882ВЕ52У. При работе в качестве таймера содержимое соответствующего T/Ci инкрементируется в каждом машинном цикле, т. е. через каждые 12 периодов резонатора. При работе в качестве

счетчика содержимое T/Ci 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 уровня (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 удерживать значение 1 как минимум в течение одного машинного цикла ядра МК.

Таймер T/C2 - это 16-битный таймер/счетчик, способный работать и как таймер, и как счетчик событий. Выбор режима работы производится битом $C/\overline{T2}$ в SFR T2CON.

Таймер T/C2 может работать в 3-х различных режимах - режиме защелки (захват), автоперезагрузки (при этом направление счета может быть как вверх, так и вниз, т.е. на увеличение или уменьшение содержимого TL2, TH2) и в режиме генератора скорости передачи в бодах. Управление осуществляется битами в T2CON, как показано в таблице 11, а режимы работы приведены в таблице 12.

Таблица 11 - Регистр T2CON управления таймера/счетчика T/C2

Адрес = 0C8H (Значение переустановки = 0000 0000B)

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	$C/\overline{T2}$	$CP/\overline{RL2}$
7	6	5	4	3	2	1	0

Символ	Функция
1	2
TF2	Флаг переполнения таймера Т/С2 (устанавливается переполнением Т/С2). Сброс его осуществляется программным путем. Флаг не устанавливается, если либо RCLK, либо TCLK установлены в единицы.
EXF2	Внешний флаг таймера Т/С2. Устанавливается, когда захват или перезагрузка вызваны отрицательным переходом на T2EX и когда EXEN2 = 1. Если прерывание от таймера Т/С2 разрешено, то EXF2 = 1 вызывает подпрограмму обработки прерывания таймера Т/С2. EXF2 не вызывает прерывания в режиме работы счетчика вверх/вниз (DCEN = 1).
RCLK	Выбор таймера Т/С2 для задания синхронизации приемника последовательного порта в режимах 1 и 3. При RCLK=1 используется таймер Т/С2, иначе - таймер Т/С1. При установленном RCLK происходит использование последовательным портом импульсов переполнения таймера Т/С2 для получения сигналов синхронизации приемника в режимах 1 и 3 (при сброшенном RCLK – таймера Т/С1).
TCLK	Выбор таймера Т/С2 для задания синхронизации передатчика последовательного порта в режимах 1 и 3. При TCLK=1 используется таймер Т/С2, иначе - таймер Т/С1. При установленном TCLK происходит использование последовательным портом импульсов переполнения таймера Т2 для получения сигналов синхронизации передатчика в режимах 1 и 3 (при сброшенном TCLK – таймера Т1).
EXEN2	Разрешение работы от внешнего сигнала для таймера Т/С2. Его установка позволяет осуществить захват или перезагрузку в результате отрицательного перехода на T2EX (P1.1), если таймер Т/С2 не используется для синхронизации последовательного порта. EXEN2 = 0 приводит к тому, что таймер Т/С2 игнорирует события в T2EX.
TR2	Запуск/остановка таймера Т/С2. TR2=1 запускает таймер Т/С2.
$C/\overline{T2}$	Бит выбора режима работы таймера Т/С2 (таймер или счетчик). $C/\overline{T2}$ = 0 - для функции таймера. $C/\overline{T2}$ = 1 - для счетчика внешних событий (счетчик перепадов из 1 в 0 на входе P1.0).
$CP/\overline{RL2}$	Выбор захвата / перезагрузки. $CP/\overline{RL2}$ = 1 вызывает выполнение захватов при отрицательных переходах у T2EX, если EXEN2 = 1. $CP/\overline{RL2}$ = 0 вызывает выполнение автоматических перезагрузок, когда таймер Т/С2 переполняется или если у T2EX происходят отрицательные переходы, когда EXEN2 = 1. Если либо RCLK или TCLK = 1, то этот разряд игнорируется, и таймер выходит на перезагрузку при переполнении таймера 2.

Таблица 12 - Режимы работы таймера/счетчика Т/С2

RCLK и TCLK	$CP/\overline{RL2}$	TR2	Режим
0 и 0	0	1	16-битный таймер/счетчик с перезагрузкой
0 и 0	1	1	16-битный таймер/счетчик с защелкиванием информации
хотя бы один установлен в 1	любое	1	генератор импульсов
любое	любое	0	выключен

Таймер T/C2 состоит из 2-х 8-битных регистров TH2 и TL2, образующих 16-разрядный регистр. Разряды управления и состояния для таймера T/C2 содержатся в регистрах T2CON и T2MOD. Регистры RCAP2H и RCAP2L являются регистрами захвата/перезагрузки для таймера 2 в режиме 16-разрядного захвата или 16-разрядной перезагрузки.

В режиме таймера 16-разрядный регистр TH2 + TL2 инкрементируется в каждом машинном цикле. Так как машинный цикл состоит из 12 периодов колебаний, скорость счета равна 1/12 частоты генератора, то есть в этом режиме происходит подсчет таймером выполненных машинных циклов.

Работа таймера T/C2 в режиме счетчика

В режиме счетчика регистр инкрементируется в ответ на перепад из 1 в 0 на соответствующем контакте внешнего ввода P1.0. В этом режиме внешний ввод анализируется в момент S5P2 каждого машинного цикла. Когда анализ показывает наличие единичного уровня в одном цикле и нулевого в следующем, содержимое счетчика инкрементируется. Новое значение счетчика появляется в регистре в момент S3P1 цикла, следующего за тем, в котором переход был обнаружен. Так как обнаружение этого перехода занимает 2 машинных цикла (24 периода колебаний), то максимальная скорость счета составляет 1/24 частоты генератора. Для гарантии того, что заданный уровень идентифицирован МК перед тем, как он изменится, он должен удерживаться по крайней мере в течение одного полного машинного цикла.

Работа таймера T/C2 в режиме захвата.

В режиме захвата есть два подрежима, выбираемых битом EXEN2 в T2CON. Если EXEN2 = 0, то таймер T/C2 - это 16-битный таймер или счетчик, при переполнении которого устанавливается бит TF2 в T2CON. Этот бит затем может использоваться для вызова прерывания.

Если EXEN2 = 1, то таймер T/C2 продолжает делать то же самое, но при этом перепад из 1 в 0 на внешнем вводе P1.1 вызывает защелкивание

текущих значений TH2 и TL2 в RCAP2H и RCAP2L соответственно. Кроме того, этот перепад вызывает установку в 1 бита EXF2 в T2CON. Бит EXF2, как и TF2, может вызвать прерывание.

Работа таймера T/C2 в режиме автоперезагрузки

Если таймер T/C2 работает в режиме 16-битного таймера/счетчика с автоперезагрузкой, то он может быть настроен на счет вверх или вниз (увеличение или уменьшение содержимого регистров TL2, TH2). Этот режим вызывается установкой в 1 бита DCEN (Down Counter Enable), расположенного в T2MOD (см. таблицу 13). При сбросе бит DCEN устанавливается в 0, так что таймер T/C2 по умолчанию считает вверх. Когда DCEN установлен, таймер T/C2 может считать вверх или вниз, в зависимости от значения на выводе P1.1 (T2EX).

В режиме автоматического подсчета вверх DCEN = 0. В этом режиме разрядом EXEN2 в T2CON выбираются две опции. Если EXEN2 = 0, то таймер 2 выполняет подсчет до 0FFFFH, а затем устанавливает разряд TF2 при переполнении. Переполнение вызывает также перезагрузку регистров таймера на 16 – разрядное значение в RCAP2H и RCAP2L. Значения в RCAP2H и RCAP2L предварительно устанавливаются программно. Если EXEN2 = 1, то 16-разрядная перезагрузка может быть запущена либо с помощью переполнения, либо за счет перехода с 1 на 0 на внешнем входе T2EX. Этот переход устанавливает также разряд EXF2. Разряды TF2 и EXF2 при разблокировании могут генерировать прерывание. Установка разряда DCEN позволяет таймеру 2 выполнять счет вверх или вниз. В этом режиме вывод T2EX управляет направлением подсчета. Появление логической 1 на T2EX заставляет таймер 2 выполнять счет по возрастающей. Таймер переполнится при достижении 0FFFFH и установит разряд TF2. Это переполнение также вызовет появление 16-разрядного значения в RCAP2H и RCAP2L для последующей перегрузки в регистры таймеров, соответственно, в TH2 и TL2.

Таблица 13 - Регистр режима T2MOD таймера T/C2

T2MOD адрес = 0C9H. Величина переустановки = XXXX XX00B

-	-	-	-	-	-	T2OE	DCEN
7	6	5	4	3	2	1	0

Символ	Функция
-	Не используется
T2OE	Бит разрешения выхода таймера T/C2.
DCEN	При установке бита DCEN таймер T/C2 конфигурируется на счет как вверх, так и вниз.

Работа таймера T/C2 в режиме счёта на увеличение

В этом режиме (при DCEN = 0) имеются 2 подрежима, выбираемых битом EXEN2 в T2CON. Если EXEN2 = 0, то таймер T/C2 считает вверх до 0FFFFH и затем устанавливает в 1 бит переполнения TF2. Переполнение вызывает также перезагрузку регистров таймера 16-битным значением в RCAP2H и RCAP2L. Значения RCAP2H и RCAP2L предварительно должны быть установлены программно. Если EXEN2 =1, 16-битная перезагрузка может срабатывать как от переполнения, так и от перепада из 1 в 0 на внешнем входе P1.1. Этот перепад также устанавливает в 1 бит EXF2. Оба бита, TF2 и EXF2, могут вызвать прерывание, если оно разрешено.

Установка бита DCEN в 1 переводит таймер T/C2 в режим счета вверх и вниз. В этом режиме вывод P1.1 управляет направлением счета. Единичный уровень на P1.1 заставляет таймер T/C2 считать вверх. Таймер переполняется по достижении значения 0FFFFH и устанавливает в 1 бит TF2. Это переполнение также вызовет перезагрузку 16-битного значения из RCAP2H и RCAP2L в регистры таймера TH2 и TL2, соответственно.

Нулевой уровень на P1.1 заставляет таймер T/C2 считать вниз. Теперь таймер «антипереполняется» (недозаполняется), когда содержимое регистров TH2 и TL2 равно значениям, хранящимся в регистрах RCAP2H и RCAP2L. «Антипереполнение» устанавливает бит TF2 в 1 и вызывает перезагрузку 0FFFFH в регистры таймера.

Бит EXF2 переключается при переполнении или антипереполнении таймера. Этот бит можно при необходимости использовать как 17-й бит таймера-счетчика. В этом режиме EXF2 не является флагом прерывания и его установка в 1 не вызывает соответствующей подпрограммы.

Работа таймера T/C2 в режиме синхронизации последовательного порта

Таймер T/C2 превращается в генератор скорости обмена для последовательного порта установкой в 1 битов TCLK и/или RCLK в T2CON.

При этом скорости в бодах для передачи и приема могут отличаться. Это достигается, например, при использовании таймера 2 для синхронизации приема, а таймера 1 - для синхронизации передачи (или наоборот). Режим генератора скорости передачи в бодах имеет общее с режимом автоперезагрузки в том, что переполнение содержимого регистров таймера/счетчика TL2 и TH2 вызывает перезагрузку регистров TL2 и TH2 16-битным значением из регистров RCAP2H и RCAP2L, предварительно установленным программно.

Скорость передачи в бодах в режимах 1 и 3 последовательного порта определяется выражением: скорость = (скорость переполнения таймера 2)/16.

Формула скорости передачи в бодах в зависимости от значений RCAP2L, RCAP2H выглядит следующим образом:

$$\text{Скорость} = \frac{\text{Частота генератора}}{32 \times (65536 - (RCAP2H, RCAP2L))},$$

где (RCAP2H, RCAP2L) - содержимое RCAP2H и RCAP2L, взятое как 16-битное целое без знака. При этом переполнение TH2 не устанавливает TF2 и не вызывает прерывания. Кроме того, если EXEN2=1, то перепад из 1 в 0 в P1.1 установит в 1 флаг EXF2, но не вызовет перезагрузки из (RCAP2H, RCAP2L) в (TH2, TL2). Таким образом, когда таймер T/C2 используется как генератор скорости передачи в бодах, вход P1.1 может использоваться при необходимости как вход дополнительного внешнего прерывания.

При этом таймер может работать как в режиме таймера, так и в режиме счетчика. В большинстве случаев выбирают режим таймера ($C/\overline{T2} = 0$).

Особенностью работы таймера/счетчика Т/С2 при использовании его в качестве генератора скорости передачи в бодах является то, что как таймер, он инкрементирует свое содержимое в каждом машинном цикле с частотой, равной 1/12 частоты генератора. Однако, как генератор скорости передачи в бодах, он инкрементирует свое значение в 6 раз быстрее с частотой, равной 1/2 частоты генератора).

Следует отметить, что при работе таймера/счетчика Т/С2 в "таймерном" режиме генератора скорости передачи в бодах нельзя пытаться считывать или записывать в TH2 или TL2. В этом случае таймер инкрементируется 6 раз в течение каждого машинного цикла, и результаты чтения или записи будут неопределенными. Регистры RCAP2 можно читать, но нельзя в них записывать, так как запись может отменить ближайшую по времени перезагрузку и вызвать ошибки приема или передачи. Таймер должен быть отключен (TR2=0) перед обращением к регистрам TH2, TL2 или к регистрам RCAP2H, RCAP2L.

Вывод программируемого сигнала меандра

Вывод Р1.0 можно запрограммировать на использование в качестве генератора меандра (генератора прямоугольных импульсов с 50 % скважностью). У МК этот вывод, помимо основной своей функции - функции ввода-вывода - может выполнять еще две дополнительные. Как упоминалось выше, он играет роль внешнего счетного входа для таймера/счетчика Т/С2. Кроме того, он может быть источником прямоугольных импульсов с 50 % скважностью и частотой в диапазоне от 61 Гц до 4 МГц при тактовой частоте генератора МК 16 МГц.

Для использования вывода Р1.0 в этом режиме бит $C/\overline{T2}$ (T2CON.1) должен быть сброшен в 0, а бит T2OE (T2MOD.1) установлен в единицу. Бит TR2 (T2CON.2) запускает и останавливает таймер.

Выходная частота генератора меандра на выходе Р1.0 зависит от частоты генератора МК и значения, загруженного в регистры защелок (RCAP2H, RCAP2L) таймера 2, в соответствии с формулой:

$$\text{Частота меандра} = \frac{\text{Частота генератора}}{4 \times (65536 - \text{RCAP2H, RCAP2L})},$$

где (RCAP2H, RCAP2L) - двухбайтовое содержимое регистровой пары, взятое как целое без знака. В описываемом режиме переполнение таймера/счетчика T/C2 не вызовет прерывания. Это похоже на режим использования таймера T/C2 как генератора скорости передачи в бодах. Можно использовать таймер T/C2 как генератор скорости передачи в бодах и как генератор меандра одновременно. Отметим, однако, что частоты передачи в бодах и генератора меандра нельзя определять независимо друг от друга, так как обе они используют RCAP2H и RCAP2L.

Для управления режимами работы таймер/счетчиков T/C0 и T/C1 и для организации взаимодействия таймеров с системой прерывания используются два регистра специальных функций (TMOD и TCON), описание которых приводится в таблицах 14, 15 и 16 соответственно. Как следует из описания управляющих бит TMOD, для обоих T/C режимы работы 0, 1 и 2 одинаковы. Режимы 3 для T/C0 и T/C1 различны. Рассмотрим кратко работу T/C во всех четырех режимах.

Таблица 14 - Регистр TMOD режима работы таймер/счетчика

Символ	Позиция	Имя и назначение
1	2	3
GATE	TMOD.7 для T/C1 и TMOD.3 для T/C0	Управление блокировкой. Если бит установлен, то T/C 0 или 1 разрешен, когда и на входе $\overline{INT0}$ или $\overline{INT1}$ высокий уровень, и бит управления "TR0" или "TR1" установлен. Если бит сброшен, то T/C 0 или 1 разрешается, как только бит управления "TR0" или "TR1"устанавливается.
$\overline{C/T}$	TMOD.6 для T/C1 и TMOD.2 для T/C0	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 "T0" или "T1".
M1	TMOD.5 для T/C1 и TMOD.1 для T/C0	Режим работы.
M0	TMOD.4 для T/C1 и TMOD.0 для T/C0	Режим работы.

Таблица 15 - Режим работы таймер/счетчиков Т/С0 и Т/С1 в зависимости от значений М1 и М0

M1	M0	Режим работы.
0	0	Регистр таймера "TLx" работает как 5-битный предделитель.
0	1	16-битный таймер/счетчик. Регистры "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.
"THx" и "TLx" – соответственно старший и младший байты таймерного регистра.		

Таблица 16 - Регистр TCON управления/статуса таймеров

Символ	Позиция	Имя и назначение
1	2	3
TF1	TCON.7	Флаг переполнения таймера 1. Устанавливается аппаратно при переполнении таймера/счетчика 1. 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 или $\overline{INT1}$. 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 программно для спецификации запроса $\overline{INT0}$ (срез/низкий уровень).

Режим 0. Перевод таймер/счетчика 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 T/C1), когда управляющий бит TR1 установлен в 1 и либо управляющий бит GATE (блокировка) равен 0, либо на внешний вывод запроса прерывания $\overline{INT1}$ поступает уровень 1. При этом установка бита GATE в 1 позволяет использовать таймер для измерения длительности импульсного сигнала, подаваемого на вход запроса прерывания.

Режим 1. Работа таймер/счетчика 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 программным путем. Перегрузка оставляет содержимое TH1 неизменным. В режиме 2 таймеры T/C0 и T/C1 также работают совершенно одинаково.

Режим 3. В режиме 3 таймеры T/C0 и T/C1 работают по-разному. T/C1 сохраняет неизменным свое текущее содержимое. В режиме 3 TL0 и TH0 функционируют как два независимых 8-битных счетчика. Работу таймерного регистра TL0 определяют управляющие биты T/C0 (C/T, GATE, TR0), входной сигнал $\overline{INT0}$ и флаг переполнения TF0. Работу таймерного регистра TH0, который может выполнять только функции таймера (подсчет машинных циклов МК), определяет управляющий бит TR1. При этом TH0 использует флаг переполнения TF1.

4.10 Сторожевой таймер

Микроконтроллер содержит аппаратный сторожевой (следящий) таймер (WDT).

Таймер WDT подсчитывает циклы команд. Разряды устройства масштабирования PS0, PS1 и PS2 в SFR WDTCON (WDT Control Register) используются для установления периода сторожевого таймера от 16K до 2048K циклов команд. Период отключения WDT зависит от частоты внешней синхронизации. WDT отключается путем переустановки включения питания и во время режима пониженной мощности. Если работа таймера разрешена,

он через заданный установками в соответствующем регистре WDTCON временной промежуток (от 16 до 2048 мс) генерирует сигнал RST пересброса микроконтроллера для переустановки ЦПУ. Избежать пересброса можно, если до этого момента установить соответствующим сигналом в 0 содержимое счетчика сторожевого таймера. Обычно для этого разработчики включают в основной цикл программы последовательность команд, периодически обнуляющую счетчик таймера. "Зависание" программы останавливает выдачу этих команд и через заданный промежуток времени после этого микроконтроллер «пересбрасывается» и вновь начинает нормально функционировать с начального адреса управляющей программы, что существенно повышает надежность работы устройства.

Сторожевой таймер выключается после сброса и в режиме Power Down.

Сторожевой таймер обслуживается регистром WDTCON (0A7H), см. таблицу 17, а выбор периода отключения таймера осуществляется в соответствии с таблицей 18.

Таблица 17 - Регистр WDTCON (WDT Control Register)

PS2	PS1	PS0	WDIDLE	DISRTO	HWDT	WSWRST	WDTEN
7	6	5	4	3	2	1	0

WDTCON адрес = A7H. Значение переустановки = 0000 0000B

Символ	Функция
1	2
PS2 PS1 PS0	Разряды масштабирования WDT. Когда все три разряда сброшены в 0, то номинальный период для сторожевого таймера составляет 16K машинных циклов, (т. е. 16 мс для частоты XTAL, равной 12 МГц в нормальном режиме или 6 МГц в режиме ×2). Когда все три разряда установлены в 1, то номинальный период составляет 2048K машинных циклов (т. е. 2048 мс при тактовой частоте 12 МГц в нормальном режиме или 6 МГц в режиме ×2).
WDIDLE	Разрешение/отключение сторожевого таймера в режиме IDLE mode. Когда WDIDLE = 0, то WDT продолжает подсчет в данном режиме. Когда WDIDLE = 1, то WDT приостанавливается до тех пор, пока продолжается режим IDLE mode.
DISRTO	Разрешение/отключение переустановки выключения сторожевого таймера (WDT приводит в действие вывод RST). Когда DISRTO = 0, то вывод RST переводится в верхний уровень после отключения WDT, и система переустанавливается. Когда DISRTO = 1, то вывод RST выполняет только функцию входа, и WDT переустанавливает только микроконтроллер после того, как заканчивает работу WDT.

Окончание таблицы 17

1	2
HWDT	Режим работы аппаратной части для WDT. Когда HWDT = 0, то WDT может быть включен/выключен просто путем установки или сброса WDTEN (для WDT это программный режим). Когда HWDT = 1, то WDT должен устанавливаться путем записи последовательности 1EH/E1H в регистр WDTRST (с адресом 0A6H) и после такой установки WDT не может быть отключен, за исключением переустановки, «горячей» или «холодной» (это аппаратный режим для WDT). Для предотвращения того, что аппаратная часть WDT переустановит всю систему, та же последовательность 1EH/E1H должна быть записана в тот же WDTRST SFR до истечения интервала отключения (пересброса).
WSWRST	Бит сброса сторожевого таймера. Когда HWDT = 0 (т. е. WDT находится в программно управляемом режиме), и когда он устанавливается программно, то это бит сброса WDT. После установления программным способом WSWRST переустанавливается аппаратным способом во время следующего машинного цикла. Если HWDT = 1, то он не будет сбрасываться аппаратным способом.
WDTEN	Бит разрешения программного режима сторожевого таймера. Когда HWDT = 0 (т. е. WDT находится в программно управляемом режиме), этот разряд разрешает установку WDT в «1» и отключает WDT при сбросе в «0» (в этом случае WDT не переустанавливается и его счетчик остается в существующем состоянии). Когда HWDT = 1, то этот разряд обеспечивает только считывание (READ-ONLY) и отражает состояние WDT (независимо от того, работает он или нет).

Таблица 18 - Выбор периода отключения сторожевого таймера

Разряды масштабирования WDT			Период пересброса, мс (для f_C = 12 МГц)
PS2	PS1	PS0	
0	0	0	16
0	0	1	32
0	1	0	64
0	1	1	128
1	0	0	256
1	0	1	512
1	1	0	1024
1	1	1	2048

4.11 Последовательный порт UART

Микроконтроллер содержит расширенный универсальный асинхронный приемопередатчик (UART) с поддержкой обнаружения ошибок посылки и автоматическим распознаванием адреса. Через UART осуществляется прием и передача информации, представленной последовательным кодом (младшими битами вперед), в полном дуплексном

режиме обмена. В состав приемопередатчика 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 команду 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. При использовании функции фиксации ошибок посылки UART в процессе связи следит за наличием всех стоп-бит. При обнаружении пропущенного бита устанавливается флаг FE в регистре SCON. Функция автоматического распознавания адреса позволяет UART распознать некоторые адреса в последовательном битовом потоке на аппаратном уровне, выполняя сравнение.

Последовательный порт МК может работать в четырех различных режимах.

Режим 0. В этом режиме информация передается и принимается через внешний вывод входа приемника (RxD). Принимаются или передаются 8 бит данных. Через внешний вывод выхода передатчика (TxD) выдаются синхросигналы сдвига, которые сопровождают каждый бит. 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 МК 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 RI=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. 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 (биты RCLK и TCLK регистра T2CON, таблица 11).

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.

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.

При передаче данных отличие от режима 1 состоит в том, что передаются не 8, а 9 бит данных, и вследствие этого цикл передачи оказывается на один полный период работы счетчика-делителя длиннее.

Скорость обмена в режиме 2, в отличие от режимов 1 и 3, фиксирована и равна 1/32 или 1/64 от значения частоты резонатора.

Режим 3. Режим 3 совпадает с режимом 2, за исключением частоты приема/передачи, которая является величиной переменной и задается таймером, как в режиме 1.

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

Управление 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 19.

Таблица 19 - Регистр управления/статуса SCON

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

Адрес SCON =98H Значение переустановки = 0000 0000B

<table>
<tr><td>Символ</td><td>Функция</td></tr>
<tr><td>1</td><td>2</td></tr>
<tr><td>FE</td><td>Разряд ошибки кадрирования. При обнаружении ошибок кадрирования в сообщениях определяются отсутствующие разряды останова и устанавливается разряд FE при их обнаружении. FE не сбрасывается и должен очищаться программно. Необходимо установить разряд SMOD0 для того, чтобы разрешить доступ к разряду FE. Устанавливается FE независимо от состояния SMOD0.</td></tr>
<tr><td>SM0</td><td>Разряд 0 режима последовательного порта, (SMOD0 должно быть равно 0 для доступа к разряду SM0)</td></tr>
<tr><td>SM1</td><td>Разряд 1 режима последовательного порта
<table>
<tr><td>SM0</td><td>SM1</td><td>Режим</td><td>Описание</td><td>Скорость передачи</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Сдвиговый регистр</td><td>f/12</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>8-бит UART</td><td>переменная</td></tr>
<tr><td>1</td><td>0</td><td>2</td><td>9-бит UART</td><td>f/64 или f/32</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>9-бит UART</td><td>переменная</td></tr>
</table></td></tr>
</table>

Окончание таблицы 19

1	2
SM2	Разрешает функцию автоматического распознавания адреса в режимах 2 или 3. Если SM2=1, то RI не будет установлен до тех пор, пока полученный 9-й разряд данных (RB8) равен 1, и полученный байт является «Заданным» или «Широковещательным Адресом». В режиме 1, если SM2=1, то RI не будет активизирован до получения достоверного разряда останова, а получаемый байт должен являться «Заданным» или «Широковещательным Адресом». В режиме 0 SM2 должен равняться 0.
REN	Разрешает последовательный прием. Устанавливается программно для разрешения приема. Сбрасывается программно для отключения приема.
TB8	9-й разряд данных, который передается в режимах 2 и 3. Устанавливается или сбрасывается программно.
RB8	В режимах 2 и 3, 9-й разряд данных, который был получен. В режиме 1, если SM2 = 0, то RB8 является разрядом останова, который был получен. В режиме 0 RB8 не используется.
TI	Флаг прерывания передачи. Устанавливается в конце периода передачи стоп-бита в режиме 0 или в начале периода приема стоп-бита в других режимах при последовательной передаче. Сбрасывается программно.
RI	Флаг прерывания приема. Устанавливается аппаратным способом в конце периода приема восьмого бита данных в режиме 0 или в середине периода приема стоп-бита в других режимах при любом последовательном приеме (см. SM2). Сбрасывается программно. В случае установленного бита RI прием данных в сдвиговый регистр невозможен.
Примечание - SMOD0 расположен в PCON.6.	

Режим работы приемопередатчика задается программно, загрузкой в старшие биты регистра SCON 2-битного кода. Во всех четырех режимах работы передача инициируется 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 согласно таблице 20.

Таблица 20 - Регистр специальных функций PCON (Регистр Управления Мощностью)

SMOD1	SMOD0	-	POF	GF1	GF0	PD	IDL
7	6	5	4	3	2	1	0

Адрес PCON =87H Значение переустановки = 00xx 0000B

Символ	Функция
SMOD1	Разряд удвоенной скорости передачи. Удваивает скорость передачи UART в режимах 1, 2 или 3. Если бит установлен в 1, то скорость передачи вдвое больше, чем при SMOD=0. По сбросу SMOD=0.
SMOD0	Выбор ошибки кадра. Когда SMOD0=1, SCON.7 – это SM0. Когда SMOD0 = 1, SCON.7 – это FE. FE будет установлен после ошибки кадра независимо от состояния SMOD0.
POF	Флаг отключения мощности. POF устанавливается в “1” во время возрастания мощности (т. е. во время «холодной» перезагрузки). Он может быть установлен или переустановлен программно и на него не влияет RST или BOD (т. е. «горячие» перезагрузки).
GF1, GF0	Флаги общего назначения
PD	Разряд уменьшения мощности. Установка этого разряда активирует уменьшение мощности.
IDL	Разряд холостого хода. Установка этого разряда активирует режим работы холостого хода.

Дополнительные функции приемопередатчика

В дополнение ко всем своим обычным режимам работы, UART может выполнять:

- обнаружение ошибок кадрирования;

- автоматическое распознавание адреса.

Обнаружение ошибок кадрирования (framing error detection) осуществляется путем просмотра отсутствующих разрядов останова. При обнаружении ошибок кадрирования, UART ищет в сообщениях отсутствующие разряды останова. Отсутствующий разряд установит бит FE в регистре SCON. Если SMOD0 установлен, то SCON.7 работает как FE. SCON.7 работает как SM0, если SMOD0 очищен. При использовании как FE SCON.7 может быть сброшен только программным способом.

Автоматическое распознавание адреса является важной особенностью, благодаря которой UART может распознавать определенные адреса в последовательном потоке двоичных данных, используя для сравнения аппаратные средства. Подобное свойство сохраняет значительную часть ресурсов программного обеспечения, поскольку при этом устраняется потребность анализа каждого последовательного адреса, который проходит через последовательный порт. Эта функция включается путем установки разряда SM2 в SCON. В 9-разрядных режимах UART (режим 2 и режим 3) флаг получения прерывания (RI) будет установлен автоматически, если получаемый байт содержит либо “данный” адрес, либо адрес “широко-вещательной рассылки”. В 9-разрядном режиме требуется, чтобы 9-й информационный бит равнялся 1 для того, чтобы показать, что полученная информация является адресом, а не данными. 8-разрядный режим называется режимом 1. В этом режиме флаг RI будет установлен в том случае, если SM2 разрешен и полученная информация имеет действующий бит останова, за которым следуют 8 бит адресов и информация является либо данным адресом, либо широковещательным адресом. Режим 0 является режимом сдвигового регистра и SM2 игнорируется. Использование режима автоматического распознавания адреса позволяет задающему устройству выборочно общаться с одним или несколькими подчиненными устройствами путем вызова адреса или адресов данных устройств. Ко всем подчиненным устройствам можно обращаться, используя широковещательный адрес. Для определения адреса подчиненного устройства используются два

специальных регистра SADDR и маски адреса SADEN. SADEN используется для того, чтобы определить, какие разряды в SADDR должны использоваться, а какие разряды “не играют роли”. Маску SADEN можно логически связать через функцию AND с SADDR для создания “Данного” адреса, который задающее устройство будет использовать для обращения к каждому из подчиненных устройств.

«Широковещательный Адрес» для каждого подчиненного устройства создается путем выбора логического ИЛИ из SADDR и SADEN. Нули в полученном результате рассматриваются как безразличные состояния. В большинстве случаев, при рассмотрении безразличных состояний в виде единиц, широковещательный адрес будет в виде шестнадцатиричного FF. При переустановке SADDR (SFR, адрес 0A9H) и SADEN (SFR, адрес 0B9H) загружаются нулями. При этом режим автоматического распознавания адреса отключается.

4.12 Последовательный порт SPI

Микроконтроллер содержит расширенный последовательный интерфейс SPI (двойная буферизация чтения/записи).

Последовательный периферийный интерфейс (SPI) предназначен для быстрого синхронного обмена информацией между микроконтроллером и периферией или между двумя микроконтроллерами. Кроме того, SPI обеспечивает возможность внутрисистемного перепрограммирования и загрузки программы во встроенное Flash ПЗУ или информации во внутреннее EEPROM данных. В расширенном режиме буфер записи SPI хранит следующий передаваемый байт. Это позволяет передать несколько байт при минимальных задержках между передачами байт, при условии, что ЦПУ имеет возможность поддерживать буфер записи заполненным.

Обмен информацией может быть полностью дуплексным и осуществляться с использованием трех линий. Максимальная частота, на которой происходит передача, равна четверти значения частоты генератора

(для 24-мегагерцевой частоты микроконтроллера - 6 МГц). Завершение передачи сопровождается установкой в 1 соответствующего флага.

Основные особенности SPI:

- полнодуплексная, 3-х проводная синхронная передача данных;
- работа в режиме Master или Slave;
- максимальная частота передачи = f/4 (f/2 при работе в режиме ×2);
- передача данных либо по принципу «младший бит вперед» либо «старший бит вперед»;
- четыре программируемых скорости передачи данных в режиме задающего устройства (Master Mode);
- флаг прерывания окончания передачи.

Межсоединение между ведущим и ведомым ЦПУ с SPI показано на рис. 9. Четырьмя выводами в интерфейсе являются Master-In/Slave-Out (MISO), Master-Out/Slave-In (MOSI), Shift Clock (SCK) и Slave Select (SS). Вывод SCK является выходом синхронизации в режиме ведущего устройства, работая в то же время в качестве входа синхронизации в подчиненном режиме. Разряд MSTR в SPCR определяет направления MISO и MOSI. При этом MOSI подсоединено к MOSI, а MISO к MISO. В режиме ведущего устройство SS/P1.4 игнорируется, и вывод может быть использован в качестве обычного универсального входа и выхода. В режиме ведомого устройства на SS должен быть установлен низкий уровень для того, чтобы выбирать каждое отдельное устройство в качестве ведомого. Если на SS установлен высокий уровень, то порт ведомого устройства SPI деактивируется, и вывод MOSI/P1.5 может использоваться как универсальный вход.

Регистры управления SPCR (SPI Control Register) и состояния (статуса) SPSR (SPI Status Register) описаны соответственно в таблицах 21 и 22.

Таблица 21 - Регистр управления SPCR

SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
7	6	5	4	3	2	1	0

Адрес SPCR = D5H Значение переустановки = 0000 0100B

<table>
<tr><th>Символ</th><th>Функция</th></tr>
<tr><td>SPIE</td><td>Разрешение прерывания от SPI – SPI Interrupt Enable. Вместе с битом ES в регистре IE этот бит разрешает прерывание SPI. Если они установлены в 1, то прерывание от SPI разрешено, а если хотя бы один из них сброшен в 0 - запрещено.</td></tr>
<tr><td>SPE</td><td>Бит включения SPI в рабочий режим (SPI Enable). Его установка в 1 соединяет выводы SS#, MOSI, MISO и SCK с выводами P1.4, P1.5, P1.6 и P1.7 соответственно. Установка бита в 0 отключает канал SPI.</td></tr>
<tr><td>DORD</td><td>Порядок передачи битов по каналу SPI (Data Order). Если DORD установлен в 1, то вначале выдается младший бит (LSB), в противном случае – старший (MSB).</td></tr>
<tr><td>MSTR</td><td>Выбор ведущего/ведомого. Установка в 1 сообщает микроконтроллеру, что он выполняет функцию ведущего. Нулевое значение бита переводит микроконтроллер в режим ведомого.</td></tr>
<tr><td>CPOL</td><td>Полярность сигнала тактирования. Полярность тактового сигнала SCK в режиме ожидания находится на высоком уровне при установленном в состояние 1 бите CPOL и на низком уровне при сброшенном бите CPOL.</td></tr>
<tr><td>CPHA</td><td>Фазировка сигнала тактирования. Разряд CPHA совместно с разрядом CPOL управляют соотношениями тактовой частоты и данных между основным и ведомым устройством.</td></tr>
<tr><td>SPR1
SPR0</td><td>Эти два бита устанавливают частоту тактирования SPI (она устанавливается только для ведущего микроконтроллера). Соотношение между значениями битов и частотой тактирования определяется:
<table>
<tr><th>SPR1</th><th>SPR0</th><th>SCK=FOSC, деленное на:</th></tr>
<tr><td>0</td><td>0</td><td>4 (2 в режиме ×/2)</td></tr>
<tr><td>0</td><td>1</td><td>16 (8 в режиме ×/2)</td></tr>
<tr><td>1</td><td>0</td><td>64 (32 в режиме ×/2)</td></tr>
<tr><td>1</td><td>1</td><td>128 (64 в режиме ×/2)</td></tr>
</table>
</td></tr>
</table>

Таблица 22 - Регистр состояния SPSR (SPI Status Register)

SPIF	WCOL	LDEN	-	-	-	DISSO	ENH
7	6	5	4	3	2	1	0

Адрес SPCR = AAH Значение переустановки = 000X XX00B

Символ	Функция
1	2
SPIF	Флаг прерывания SPI. Устанавливается аппаратно при завершении передачи и создается прерывание, если SPIE=1 и ES=1. Сбрасывается путем считывания регистра состояния SPI, за которым следует считывание/запись регистра данных SPI.
WCOL	Когда ENH = 0, то WCOL - флаг ошибки при записи данных. Разряд WCOL устанавливается, если регистр данных SPI записывается во время передачи данных. Во время передачи данных результат считывания регистра SPDR может быть неверным, и запись в него не дает эффекта. Разряд WCOL (и разряд SPIF) сбрасываются путем считывания регистра состояния SPI, за которым следует считывание/запись регистра данных SPI. Если ENH = 1, то WCOL является расширением буфера Tx при его заполнении. При записи, когда WCOL = 1 в расширенном режиме, будет происходить запись, замещая данные, уже находящиеся в буфере Tx. В этом режиме WCOL не переустанавливается при переустановке SPIF.

Окончание таблицы 22

1	2
LDEN	Бит разрешения загрузки для буфера Tx в расширенном SPI режиме. Когда ENH установлен, можно загружать буфер Tx, если LDEN = 1 и WCOL = 0.
DISSO	Бит блокировки выхода ведомого устройства. При его установке на выводе MISO создается третье состояние, поэтому один интерфейс может быть использован более, чем одним ведомыми устройствами с одним ведущим устройством. Обычно первый байт в передаче является адресом ведомого устройства, и только выбранное ведомое устройство должно сбрасывать свой бит DISSO.
ENH	Бит выбора расширенного режима SPI. Если ENH = 0, то SPI находится в нормальном режиме, т. е. без двойного буферирования записи. Если ENH = 1, то SPI находится в расширенном режиме с двойным буферированием записи. Буфер Tx использует совместно тот же адрес с регистром SPDR.

Регистр SPDR (таблица 23) представляет собой регистр с возможностью чтения/записи и предназначен для пересылки данных между регистровым файлом и сдвиговым регистром SPI. Запись в регистр SPDR инициирует передачу данных, считывание регистра приводит к чтению сдвигового регистра приема.

Таблица 23 - Регистр данных SPDR

Адрес SPDR = 86H Значение переустановки = 00H (после «холодной» переустановки), не изменено (после «горячей» переустановки)

SPD7	SPD	SPD	SPD	SPD	SPD	SPD	SPD
7	6	5	4	3	2	1	0

При запуске SPI необходимо установить режим синхронизации до выдачи разрешения для SPI, затем установить в SPCR все необходимые разряды, за исключением разряда SPE, после чего установить SPE. Работа ведущего устройства SPI должна быть разрешена раньше, чем ведомого устройства. Ведомое устройство будет повторять ведущее устройство во время следующего Tx, если оно не загружается новыми данными.

Структурная схема SPI-интерфейса микроконтроллера изображена на рисунке 8, а схема соединения узлов SPI двух различных устройств - на рисунке 9. Запись в регистр данных SPI (SPDR) ведущего устройства запускает его тактовый генератор, разрешая прохождение сигналов через делитель частоты. Данные из регистра бит за битом поступают на вывод MOSI ведущего, соединенного с одноименным входом ведомого. Каждый импульс тактовой частоты SCK, вырабатываемый ведущим узлом, сопровождается передачей бита. Ведомый принимает бит со своего входа MOSI и заносит его в свой сдвиговый регистр. Одновременно с этим через выход MISO он бит за битом передает ведущему узлу содержимое своего регистра сдвига, который принимает его по одноименному входу. Таким образом, после прохождения восьми импульсов тактового генератора содержимое сдвиговых регистров ведущего и ведомого обменивается местами. Завершение передачи устанавливает флаг SPIF у ведущего и при взведенных SPIE и ES вызывает соответствующее прерывание.

Вывод PB1 (SCK) является выходом тактового сигнала ведущего микроконтроллера и входом тактового сигнала ведомого. По записи ведущим CPU данных в SPI регистр начинает работать тактовый генератор SPI, и записанные данные сдвигаются через вывод выхода PB2 (MOSI) ведущего микроконтроллера на вывод входа PB2 (MOSI) ведомого микроконтроллера. После сдвига одного байта тактовый генератор SPI останавливается, устанавливая флаг окончания передачи (SPIF). Если в регистре SPCR будет установлен бит разрешения прерывания SPI (SPIE), то произойдет запрос прерывания. Вход выбора ведомого PB0 (SS#), для выбора индивидуального SPI устройства в качестве ведомого устанавливается на низкий уровень. Два сдвиговых регистра ведущего и ведомого микроконтроллеров можно рассматривать как один разнесенный 16-разрядный циклический сдвиговый регистр. При сдвиге данных из ведущего микроконтроллера в ведомый одновременно происходит сдвиг данных из ведомого микроконтроллера в ведущий, т. е. в течение одного цикла сдвига происходит обмен данными

между ведущим и ведомым микроконтроллерами. Если к одному ведущему узлу подключено несколько ведомых, то перед началом обмена первый должен установить в 0 вход SS# (P1.4) того ведомого, который должен принимать информацию. При этом у невыбранных ведомых выводы MOSI и MISO оказываются подключенными к линии как входы, чтобы на линиях не было конфликтов данных. Формат передаваемых данных при различных комбинациях битов CPHA и CPOL приведен на рисунках 10 и 11.

При разрешенном SPI направления данных выводов MOSI, MISO, SCK и SS# настраиваются в соответствии с таблицей 24.

Таблица 24 - Настройка выводов MOSI, MISO, SCK и SS#

Вывод	Направление, ведущий SPI	Направление, ведомый SPI
MOSI	Определяется пользователем	Вход
MISO	Вход	Определяется пользователем
SCK	Определяется пользователем	Вход
SS#	Определяется пользователем	Вход

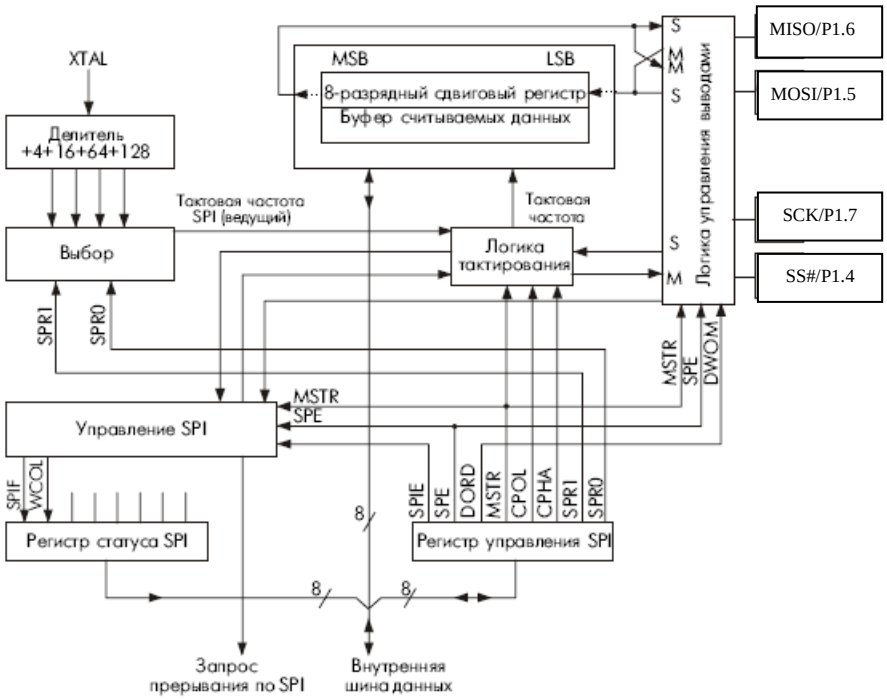


Рисунок 8 - Блок-схема последовательного порта SPI

Буфер записи данных используется только в расширенном режиме SPI. Для SPI имеется два режима работы: нормальный (не буферированная

запись) и расширенный (буферированная запись). В нормальном режиме запись в регистр данных SPI (SPDR) для основного ЦПУ запускает тактовый генератор SPI, и записанные данные сдвигаются из вывода MOSI на вывод MOSI подчиненного ЦПУ. Передача может начаться после некоторой начальной задержки, пока тактовый генератор ожидает следующего полного битового интервала для указанной скорости передачи. После сдвига на один байт тактовый генератор останавливается, устанавливая при этом окончание флага передачи (SPIF) и передавая полученный байт в буфер считывания (SPDR). Если устанавливаются оба разряда, как разряд разрешения прерывания SPI (SPIE), так и разряд разрешения прерывания последовательного порта (ES), то выдается запрос на прерывание. SPDR обращается либо к буферу записи данных, либо к буферу считывания данных, в зависимости от того, является доступ считыванием или записью. В обычном режиме, поскольку буфер прозрачен, (и доступ к записи в SPDR будет направлен в сдвиговый буфер), любая попытка выполнить запись в SPDR, когда выполняется передача, приведет к ошибке и конфликту записи с устройством WCOL. В то же время, передача будет завершаться нормально, но новый байт будет игнорироваться, и будет необходим новый доступ записи к SPDR. Расширенный режим аналогичен нормальному, за исключением того, что буфер записи удерживает следующий байт от передачи. Запись в SPDR загружает буфер записи, и устанавливает WCOL таким образом, чтобы сообщалось о том, что буфер заполнен и любая дальнейшая запись перезапишет буфер. WCOL очищается аппаратным способом при загрузке буферированного байта в сдвиговый регистр, после чего начинается передача. Если ведущее устройство SPI в данный момент работает в режиме холостого хода, т. е., если это первый байт, то после загрузки SPDR начинается передача байта, и WCOL немедленно очищается. Пока передается этот байт, в SPDR может записываться следующий байт. Флаг разрешения загрузки (LDEN) в SPSR может быть использован для определения начала передачи. LDEN подтверждается во время первых

четырех битовых интервалов передачи SPI. Ведущий ЦПУ должен сначала проверить, что LDEN установлено, и что WCOL очищается перед загрузкой следующего байта. В расширенном режиме, если WCOL установлен по завершению перемещения, т. е. следующий байт доступен, тогда SPI сразу же загружает буферированный байт в сдвиговый регистр, переустанавливает WCOL и продолжает передачу без остановки и перезапуска тактового генератора. В течение периода времени, пока ЦПУ может поддерживать буфер записи заполненным подобным образом, байты могут быть перемещены с минимальным временем ожидания между байтами.

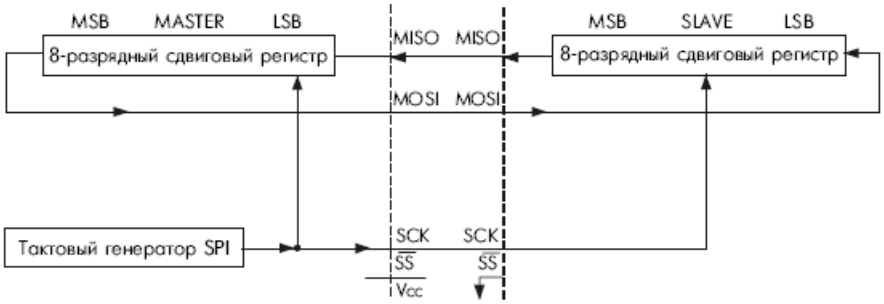


Рисунок 9 - Межсоединения ведущего и ведомого SPI

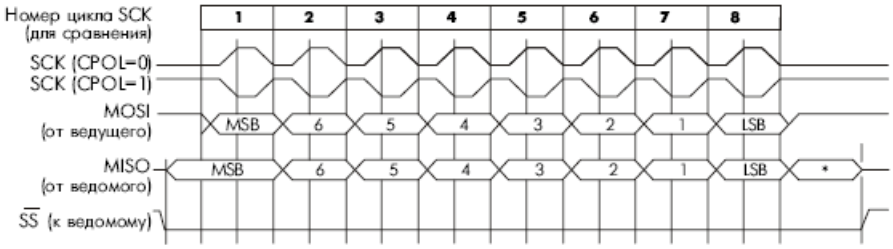


Рисунок 10 - Формат передаваемых данных SPI при CPHA = 0

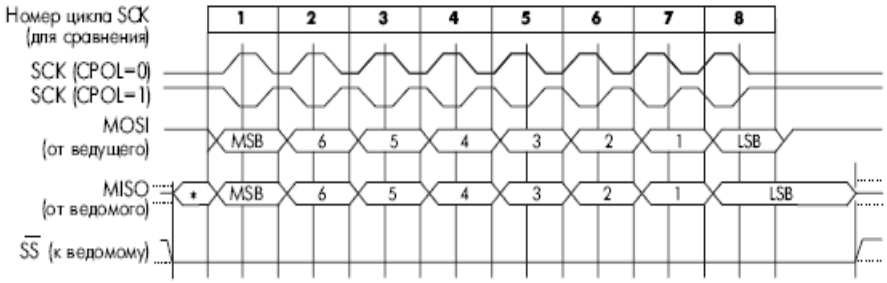


Рисунок 11 - Формат передаваемых данных SPI при CPHA = 1

Биты CPHA, CPOL и SPR в SPCR управляют формой и скоростью SCK. Два разряда SPR обеспечивают четыре возможных тактовые частоты, когда

SPI находится в режиме ведущего устройства. В режиме ведомого устройства SPI будет работать с частотой поступающего SCK до тех пор, пока не превысит максимальную скорость передачи данных. В части последовательных данных существует также четыре возможных комбинации фазы и полярности SCK. CPHA и CPOL определяют вид формата, используемого для передачи. Для предотвращения сбоев на SCK из-за прерывания интерфейса, CPHA, CPOL и SPR должны быть установлены до включения интерфейса, а ведущий прибор должен быть включен до включения ведомого устройства.

5 Система прерываний

Микроконтроллер содержит четырехуровневый расширенный контроллер прерываний. Каждый источник прерываний может раздельно программироваться на один из четырех уровней приоритета путем установки или сброса бит в регистре приоритета прерываний (IP) и в старшем регистре приоритета прерываний (IPH). Старший регистр приоритета прерываний показывает значения бит и уровни приоритетов, связанные с каждой комбинацией.

Микроконтроллер имеет шесть векторов прерывания: два внешних прерывания (INT0 и INT1), три прерывания таймера (Таймеры 0, 1 и 2) и прерывание последовательного порта. Прерывание от последовательного порта является логическим ИЛИ (OR) разрядов RI и TI в регистре SCON, а также разрядом SPIF в SPSR (если SPIE установлено в SPCR). При этом необходимо программно определять источник прерывания: от UART или от SPI. 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. IE содержит также разряд глобального отключения EA, который отключает все прерывания одновременно.

Низкоприоритетное прерывание может быть прервано высокоприоритетным прерыванием, но не другим низкоприоритетным прерыванием. Если одновременно поступило два запроса на прерывание с разными уровнями приоритета, то сначала выполняется процедура высокоприоритетного прерывания.

Прерывание Таймера 2 инициируется логическим ИЛИ (OR) разрядов TF2 и EXF2 в регистре T2CON. Прерывание от последовательного порта является логическим ИЛИ (OR) разрядов RI и TI в регистре SCON, а также разрядом SPIF в SPSR. Флаги таймеров T/C0 и T/C1 (соответственно TF0 и TF1) устанавливаются в момент S5P2 цикла, в котором переполняются таймеры. Значения опрашиваются в следующем цикле. Флаг таймера T/C2 TF2 устанавливается по S2P2 и опрашивается в том же цикле, в котором переполняется таймер. В процессе обработки прерывания аппаратно сгенерированная процедура LCALL помещает содержимое счётчика команд PC в стек и загружает его с начальным адресом соответствующего блока обработки прерывания. Кроме счётчика команд автоматически в стеке не сохраняются никакие другие регистры. Адреса векторов прерываний приведены в таблице 25.

Таблица 25 - Адреса векторов прерываний

Прерывание	Источник (Бит запроса прерывания)	Адрес вектора
Системный сброс	RST или POR или BOD	0000H
Внешнее прерывание 0	IE0	0003H
Переполнение счетчика T/C0	TF0	000BH
Внешнее прерывание 1	IE1	0013H
Переполнение счетчика T/C1	TF1	001BH
Последовательный порт UART или SPI	RI, TI	0023H
Переполнение счетчика T/C2	TF2	002BH

Необходимо отметить, что POR – «холодная» переустановка или сброс, вызванные снижением напряжения питания, влияет на флаг POF. BOD – «горячая» переустановка или сброс, инициируемые устройством обнаружения снижения уровня мощности, не оказывает влияния на флаг POF.

Внешние прерывания $\overline{\text{INT0}}$ и $\overline{\text{INT1}}$ могут быть вызваны либо уровнем, либо переходом сигнала из 1 в 0 на входах МК 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 и очищением битов IE0 и IE1 регистра TCON. 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 аппаратными средствами.

В блоке регистров специальных функций есть регистры, предназначенные для управления режимом прерываний и уровнями приоритета. Форматы этих регистров, имеющих символические имена IF, IP и IPH, описаны в таблицах 26, 27 и 28 соответственно. Возможность программной установки/сброса любого управляющего бита в этих двух регистрах делает систему прерываний МК исключительно гибкой.

Таблица 26 - Регистр разрешения прерываний IE

EA	-	ET2	ES	ET1	EX1	ET0	EX0
7	6	5	4	3	2	1	0

Адрес IE = A8H Значение переустановки = 0X00 0000B

Бит разрешения = 1 разрешает прерывание

Бит разрешения = 0 отключает прерывание

Символ	Функция
EA	Бит запрета прерываний. Если EA#= 0, то все прерывания запрещены. Если EA#= 1, то каждый источник прерывания разрешается индивидуально или запрещается путем установки или очистки разряда, который его разрешает. Сбрасывается программно для запрета всех прерываний.
ET2	Бит разрешения прерываний от таймера T/C2. Установка/сброс программой для разрешения/запрета прерываний от таймера 2.
ES	Бит разрешения прерывания от UART или SPI. Установка/сброс программой для разрешения/запрета прерываний от флагов TI или RI и SPI.
ET1	Бит разрешения прерывания от таймера 1. Установка/сброс программой для разрешения/запрета прерываний от таймера 1.
EX1	Бит разрешения внешнего прерывания 1. Установка/сброс программой для разрешения/запрета прерывания 1.
ET0	Бит разрешения прерывания от таймера 0. Установка/сброс программой для разрешения/запрета прерываний от таймера 0.
EX0	Бит разрешения внешнего прерывания 0. Установка/сброс программой для разрешения/запрета прерывания 0.

Таблица 27 - Регистр приоритетов прерываний IP

-	-	PT2	PS	PT1	PX1	PT0	PX0
7	6	5	4	3	2	1	0

Адрес IP = B8H Значение перестановки = XX00 0000B

Символ	Функция
PT2	Низкий приоритет прерывания Таймера 2
PS	Низкий приоритет прерывания последовательного порта
PT1	Низкий приоритет прерывания Таймера 1
PX1	Низкий приоритет внешнего прерывания $\overline{\text{INT}1}$
PT0	Низкий приоритет прерывания Таймера 0
PX0	Низкий приоритет внешнего прерывания $\overline{\text{INT}0}$

Таблица 28 - Регистр высокого приоритета прерывания IPH

-	-	PT2	PS	PT1	PX1	PT0	PX0
7	6	5	4	3	2	1	0

Адрес IPH = B7H Значение перестановки = XX00 0000B

Символ	Функция
PT2H	Высокий приоритет прерывания Таймера 2
PSH	Высокий приоритет прерывания последовательного порта
PT1H	Высокий приоритет прерывания Таймера 1
PX1H	Высокий приоритет внешнего прерывания 1
PT0H	Высокий приоритет прерывания Таймера 0
PX0H	Высокий приоритет внешнего прерывания 0

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

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

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

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

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.

Прерывание таймера 2 формируется логикой И битов TF2 и EXF2 в регистре T2CON. Оба этих флага должны сбрасываться программным путем, аппаратного сброса они не имеют. Подпрограмма прерывания должна самостоятельно определить, какой из этих флагов установлен, и сбросить его, выполняя после этого действия, предусмотренные в качестве реакции МК на вызов прерывания.

6 Сброс МК

Сброс МК осуществляется единичным уровнем сигнала. Этот сигнал должен быть приложен к выводу RST. Сброс достигается удержанием вывода RST вверху в течение, по крайней мере, 2-х машинных циклов (24 периодов колебаний) при работающем генераторе. CPU отвечает выработкой внутреннего сброса по определенному алгоритму.

Внешний сигнал сброса асинхронен в отношении внутренних сигналов МК. RST вывод опрашивается в течение состояния 5 фазы 2 каждого машинного цикла. Пока вывод RST имеет единичный уровень, выводы ALE и PSEN# превращаются во входы, и потенциал на них медленно нарастает.

После возврата RST в нулевой уровень требуется от 1 до 2 машинных циклов, чтобы ALE и PSEN# начали синхронизироваться. Подача на выводы ALE и PSEN# сигнала нулевого уровня в момент сброса может перевести МК в неопределенное состояние.

Алгоритм внутреннего сброса записывает исходную информацию во все регистры специальных функций. В таблице 7 приведен список регистров специальных функций и их значений после прохождения сигнала сброса.

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

Микроконтроллер содержит внутреннюю схему сброса при подаче питания. Для обеспечения устойчивой ее работы рекомендуется подключение вывода RST через конденсатор емкостью порядка 1 мкФ к положительному полюсу U_{CC} источника питания 5 В. Кроме того, должна быть исключена возможность попадания помех и выбросов по указанному выводу RST. Пока не запустится генератор МК и не выполнится описанный выше алгоритм сброса, выводы портов P0...P3 будут находиться в неопределенном состоянии.

7 Режимы работы МК с пониженным энергопотреблением

Во многих вариантах использования МК энергопотребление является одним из основных параметров. МК имеет два режима с пониженным потреблением - режим ожидания или холостого хода ("Idle") и режим отключения или микропотребления ("Power Down Mode").

В режиме холостого хода Idle ЦПУ переходит в «спящее» состояние, но при этом все имеющиеся на кристалле периферийные устройства остаются в активном состоянии. Этот режим управляется с помощью программного обеспечения. Содержимое встроенного ОЗУ и всех регистров специальных функций остается неизменным во время этого режима. Режим холостого хода может быть прерван с помощью любого разрешенного прерывания или переустановки аппаратной части (сброс). При этом устройство обычно возобновляет исполнение программ с того места, с которого оно было прервано. Для устранения возможности непредвиденной записи на выводе порта, в случае, когда режим холостого хода заканчивается сбросом, команда, следующая за командой, вызывающей режим холостого хода, не должна вести запись в вывод порта или во внешнюю память. Состояние внешних выводов во время режимов холостого хода или пониженной мощности приведено в таблице 29.

Таблица 29 - Состояние внешних выводов во время режимов холостого хода или пониженной мощности

Режим	Программная память	ALE	$\overline{\text{PSEN}}$	PORT0	PORT1	PORT2	PORT3
Idle	Внутренняя	1	1	Данные	Данные	Данные	Данные
Idle	Внешняя	1	1	«Плавающее» состояние	Данные	Адрес	Данные
Power Down	Внутренняя	0	0	Данные	Данные	Данные	Данные
Power Down	Внешняя	0	0	«Плавающее» состояние	Данные	Данные	Данные

В режиме пониженной мощности Power Down Mode генератор останавливается и исполняется команда, которая инициирует режим пониженной мощности. При остановке тактового генератора 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.

Встроенное ОЗУ и Регистры Специальных Функций (SFR) сохраняют свои значения до окончания действия режима пониженной мощности. Выход из режима пониженной мощности может быть инициирован либо с помощью сброса, либо с помощью прерывания. Переустановка переопределяет SFR, однако, не изменяет встроенное ОЗУ. Для выхода из режима пониженной мощности через прерывание вывод внешнего прерывания P3.2 или P3.3 необходимо поддерживать на нижнем логическом уровне в течение промежутка времени, необходимого для требуемого периода для пуска кварцевого генератора. После этого начинается процедура прерывания во время периода фронта нарастания сигнала на выводе внешнего прерывания, при условии, что устанавливается разряд SFR, равный AUXR.1. Если AUXR.1 переустанавливается (сбрасывается), то исполнение

начинается после синхронизированного интервала в 2 мс (номинальное значение) от фронта спада с внешнего вывода прерывания. Пользователь не должен вводить повторно режим пониженной мощности в течение, как минимум, 4 мс.

Режимы холостого хода и микропотребления активируются при установке соответствующих битов в специальном регистре - регистре управления мощностью PCON. Адрес этого регистра 87H.

Следует отметить следующие особенности этого регистра.

Если одновременно установлены в единицы биты IDL и PD, то бит PD имеет преимущество - МК переходит в режим микропотребления. Содержимое регистра PCON после сброса: (0XXX0000).

МК может выходить из режима пониженного энергопотребления как при аппаратном сбросе, так и при внешнем прерывании. Чтобы обеспечить правильный выход из этого режима, сброс или внешнее прерывание не должны подаваться прежде, чем U_{CC} 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 $\overline{INT0}$ или $\overline{INT1}$ схема прерываний должна быть конфигурирована на срабатывание по уровню соответствующего сигнала. После выполнения команды RETI в подпрограмме обслуживания прерывания следующей будет выполняться та команда, которая идет после той, что перевела МК в режим пониженного энергопотребления.

Флаг отключения питания (POF), расположенный в PCON.4, устанавливается аппаратно, когда питающее напряжение повышается от 0 до примерно 5 В. POF может также установлен или стерт программно.

8 Программирование ИМС

Микроконтроллер имеет 12K байт внутрисистемной перепрограммируемой Flash памяти программ и 2K байта ЭСППЗУ памяти

данных. При поставке микроконтроллера блоки памяти поставляются в стертом состоянии, (содержимое каждой ячейки = FFH). Этим обеспечивается готовность блоков памяти к программированию.

В микроконтроллере реализована страничная запись в ЭСППЗУ в процессе работы ЦПУ. Это даёт возможность одновременной записи одной страницы в ЭСППЗУ (32 байта). При этом на запись 32 байт потребуется только 4 мс вместо 128 мс, требуемых при побайтном программировании. В процессе выполнения программы (с помощью инструкции MOVX) поддерживается возможность автоматического стирания на байтном уровне. Это полезно, в случае необходимости, модифицировать содержимое только одного байта ЭСППЗУ в реальном времени, а остальные оставить незатронутыми. Память программ и память данных могут программироваться в страничном режиме (1 страница программного кода = 64 байта; страница данных = 32 байта). Страничный режим доступен как при параллельном, так и при последовательном программировании. Доступ к встроенной Flash - памяти и ЭСППЗУ организован через последовательный интерфейс SPI. При переводе входа сброса в активное состояние интерфейс SPI переходит в режим последовательного программирования и позволяет считать или запрограммировать память программ при условии отсутствия защиты программирования. Алгоритм программирования микроконтроллера предусматривает раздельное программирование памяти программ и памяти данных. В микроконтроллере поддерживаются режимы как параллельного, так и последовательного программирования. Режим последовательного программирования дает возможность перепрограммирования устройства внутри самой пользовательской системы. Режим параллельного программирования совместим с программаторами Flash и ЭСППЗУ памяти, описанными в настоящем руководстве.

Режимы программирования (справочные данные) приведены в таблицах 30 и 31.

Таблица 30 - Коды режимов программирования памяти программ и данных

Режим			RST	PSEN#	ALE	EA#	P3.3	P3.4	P3.5	P3.6	P3.7	Данные входа/ выхода P0.7-P0.0	Адрес P2.5-P0.0, P1.7-P1.0
Режимы последовательного программирования			H	H	H								
Стирание кристалла			H	L	1,0 мкс	12 В	H	L	H	L	L	X	X
Запись страницы		12К код	H	L	1,0 мкс	12 В	L	H	H	H	H	D1	ADDR
Считывание		12К код	H	L	H	12 В	L	L	H	H	H	D0	ADDR
Запись страницы		2К данные	H	L	1,0 мкс	12 В	L	H	L	H	H	D1	ADDR
Считывание		2К данные	H	L	H	12 В	L	L	L	H	H	D0	ADDR
Запись бит блокировки		Разряд 01	H	L	1,0 мкс	12 В	H	L	H	H	L	D0=0	X
		Разряд 02										D1=1	X
		Разряд 03										D2=0	X
Считывание бит блокировки		Разряд 01	H	L	H	12 В	H	H	H	L	L	D0	X
		Разряд 02										D1	X
		Разряд 03										D2	X
Запись страницы		Строка пользователя	H	L	1,0 мкс	12 В	H	L	H	H	H	D1	0-3FH
Считывание		Строка пользователя	H	L	H	12 В	L	L	H	L	H	D0	0-3FH
Считывание		Строка сигнатуры	H	L	H	12 В	L	L	H	L	L	D0	0-3FH
Запись бит конфигурации	Бит 1	Разрешение / Запрет последовательного программирования	H	L	1,0 мкс	12 В	L	H	H	L	H	D0=0	X
												D0=1	X
	Бит 2	Разрешение синхронизации ×2										D1=0	X
		Блокировка синхронизации ×2										D1=1	X
	Бит 3	Разрешение / Запрет программирования строки пользователя										D2=0	X
												D2=1	X
	Бит 4	Разрешение внешней синхронизации										D3=0	X
		Разрешение тактового генератора										D3=1	X
Считывание бит конфигурации		Последовательное программирование (перемычка 1)	H	L	H	12 В	H	H	H	L	H	D0	X
		Синхронизация (перемычка 2)										D1	X
		Программирование строки пользователя (перемычка 3)										D2	X
		Выбор синхронизации (перемычка 4)										D2	X

Таблица 31- Режимы программирования памяти программ и данных

Символ	Параметр	Мин	Макс	Единица измерения
V_{PP}	Напряжение разрешения программирования	11,5	12,5	В
$1/t_{CLCL}$	Частота генератора	3	24	МГц
t_{PWRUP}	Высокий логический уровень RST	10		мкс
t_{RHX}	Высокий логический уровень RST до запуска XTAL	10		мкс
t_{OSTL}	Время установки генератора	10		мкс
t_{HSTL}	Время установки высокого напряжения	10		мкс
t_{MSTP}	Время установки режима к моменту переключения PROG# в состояние низкого логического уровня	1		мкс
t_{ASTP}	Время установки адреса к моменту переключения PROG# в состояние низкого логического уровня	1		мкс
t_{DSTP}	Время установки данных к моменту переключения PROG# в состояние низкого логического уровня	1		мкс
t_{PGW}	Ширина импульса PROG#	1		мкс
t_{AHLD}	Удержание данных после импульса PROG#	1		мкс
t_{DHLD}	Удержание данных после импульса PROG#	1		мкс
t_{BLT}	Период загрузки байта	1	150	мкс
t_{PHBL}	Время между переключением PROG# в высокое логическое состояние и BUSY# в низкое логическое состояние		256	мкс
t_{WC}	Длительность цикла записи		4,5	мкс
t_{MHLD}	Удержание режима после переключения BUSY# в низкое логическое состояние	10		мкс
t_{VFY}	Время предустановки адреса для верификации достоверных данных		1	мкс
t_{PSTP}	Предустановка PROG# к моменту подачи высокого логического уровня на U_{PP}	10		мкс
t_{PHLD}	Удержание PROG# после переключения U_{PP} в низкое логическое состояние.	10		мкс
t_{PLX}	Удержание PROG# в низком логическом состоянии до остановки XTAL	1		мкс
t_{XRT}	Остановка XTAL до момента переключения RST в низкое логическое состояние	1		мкс
t_{PWRDN}	Переключение RST в низкое логическое состояние к моменту отключения питания.	1		мкс

Матрицы памяти кодов и данных адресуются через отдельные адресные пространства в параллельных и последовательных режимах программирования: с 0000H по 2FFFH для памяти программ и с 000H по 7FFH для памяти данных. Память программ и данных в микроконтроллере программируются побайтовым или постраничным способом в любом из

режимов программирования. Для перепрограммирования любого байта в параллельном или последовательном режиме пользователю необходимо сначала вызвать команду «Стирание Кристалла» (Chip Erase) для того, чтобы стереть обе матрицы, поскольку отсутствует встроенная функция автоматического стирания. Для блокировки программирования памяти программно устанавливается «UsrRowProEn» = 1 (как в параллельном, так и в последовательном режимах программирования).

8.1 Параллельное программирование

Время программирования (страничный способ) составляет: 0,96 с для Flash-памяти программ (сброс=1); 0,32 с для ЭСППЗУ (сброс=1) и 0,256 с для ЭСППЗУ (сброс=0, работа ЦПУ).

Выводы, используемые в качестве сигналов управления при параллельном программировании - P3.3, P3.4, P3.5, P3.6 и P3.7.

Для индикации в режиме параллельного программирования (программирование байта продолжается) используется вывод RDY/$\overline{\text{BSY}}$ (P3.0). При необходимости перепрограммирования любого байта после выполнения программирования потребуется выполнение команды "Chip Erase" (стирание кристалла).

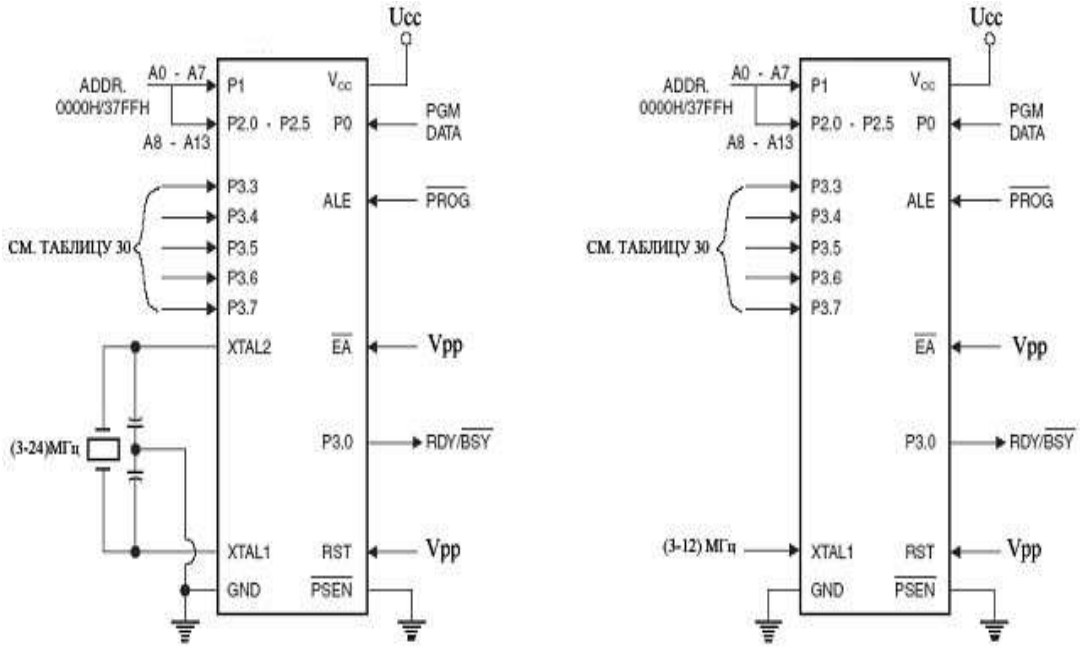
За процессом побайтового программирования в режиме параллельного программирования можно наблюдать, используя выходной сигнал RDY/$\overline{\text{BSY}}$. Вывод P3.0 переходит на нижний логический уровень после перехода ALE на высокий логический уровень во время программирования для индикации BUSY. P3.0 снова переходит на высокий уровень (High), если выполнено программирование для индикации READY. Для P3.0 необходим внешний резистор при функционировании в качестве RDY/$\overline{\text{BSY}}$.

Если не запрограммированы биты защиты LB1 и LB2, то программируемый байт Кода или Данных может быть обратно считан через линии адреса и данных для проверки. Состояние разрядов защиты можно

также проверить непосредственно в режимах параллельного и последовательного программирования.

Обе матрицы памяти (Flash и ЭСППЗУ) можно стирать одновременно. В режиме параллельного программирования «Стирание Кристалла» запускается с помощью соответствующих сигналов управления. В матрицы кодов и данных записываются во время процедуры стирания во все ячейки «1» (FFh). Строку Пользователя можно также стирать, если «UsrRowProEn» = 0 (разрешенное состояние).

Схема включения микросхемы при параллельном программировании приведена на рисунке 12.



а) б)

Рисунок 12 - Схема включения микросхемы при параллельном программировании: а) – с кварцевым резонатором;
б) – с внешним синхросигналом.

Для программирования и верификации ИМС в параллельном режиме может использоваться следующий алгоритм:

1. Подключение питания:

- подключить питание и общий вывод к выводам U_{CC} и GND соответственно;

- установить на выводе RST высокий логический уровень;

- подключить источник тактовых импульсов с частотой от 3 МГц до 24 МГц к выводу XTAL1 и выждать не менее 10 мс.

2. Установить на выводе $\overline{\text{PSEN}}$ низкий логический уровень, установить на выводе ALE высокий логический уровень, установить на выводе $\overline{\text{EA}}$ и всех других выводах высокий логический уровень.

3. Увеличить напряжение на выводе $\overline{\text{EA}}/V_{PP}$ до 12 В для разрешения программирования Flash памяти, стирания или верификации. Подключить к выводу P3.0 подтягивающий резистор (10 кОм) для RDY/$\overline{\text{BSY}}$;

4. Подать соответствующую кодовую комбинацию на выводы P3.3 – P3.7 для выбора одного из режимов программирования, показанных в таблице режимов программирования.

5. Подать байт адреса на выводы P1.0 – P1.7 и P2.0 – P2.5. Подать байт данных на выводы P0.0 – P0.7.

6. Подать импульс ALE/PROG для загрузки памяти программ, памяти данных или битов блокировки.

7. Повторить шаги 5-6, изменяя адрес и данные для 64 байт в странице памяти программ или 32 байт в странице памяти данных. Когда производится загрузка страницы, интервал между загружаемыми байтами не должен превышать 150 мкс. Иначе устройство воспримет затянувшуюся паузу как окончание загрузки.

8. После загрузки последнего байта текущей страницы необходимо выждать 5 мс или контролировать состояние вывода RDY/$\overline{\text{BSY}}$, пока он не переключится в высокое логическое состояние.

9. Для верификации последнего байта только что запрограммированной страницы подать низкое логическое состояние на вывод P3.4 и прочитать запрограммированные данные на выводах P0.0 – P0.7.

10. Повторить шаги с 4 по 7, изменяя адреса и данные для заполнения всего массива или до окончания программируемого файла.

11. Порядок выключения питания:

- перевести в состояние высокого импеданса шины адреса и данных;
- снять нагрузку вывода P3.0, используемую для RDY/$\overline{\text{BSY}}$;
- установить на выводе XTAL1 низкое логическое состояние;
- установить выводы RST и $\overline{\text{EA}}$ в низкое логическое состояние;
- отключить питание с вывода U_{CC}.

За процессом побайтового программирования в режиме параллельного программирования можно следить, используя выходной сигнал RDY/$\overline{\text{BSY}}$. Вывод P3.0 переходит на нижний логический уровень после перехода ALE на высокий логический уровень во время программирования для индикации BUSY. P3.0 снова переходит на высокий уровень, если выполнено программирование для индикации READY. Для P3.0 необходим внешний резистор около 10 кОм при функционировании в качестве RDY/$\overline{\text{BSY}}$.

Если не запрограммирована блокировка разрядов LB1 и LB2, то программируемый байт Кода или Данных может быть обратно считан через линии адреса и данных для проверки. Состояние разрядов запирания можно также проверить непосредственно в режимах параллельного и последовательного программирования.

Обе матрицы памяти можно стирать одновременно. В режиме параллельного программирования действие «Стирание Кристалла» запускается с помощью соответствующих сигналов управления. В режиме последовательного программирования стирание кристалла инициируется при выдаче команды «Стирание Кристалла». В этом режиме стирание занимает примерно 8 мс. Во время действия команды «Стирание Кристалла» последовательное считывание из любого места расположения адреса возвращает 00H к выходам данных.

Для блокировки Последовательного Программирования имеется программируемая перемычка. Перемычка Последовательного Программирования может быть задействована/блокирована как в Параллельном, так и в Последовательном режимах программирования.

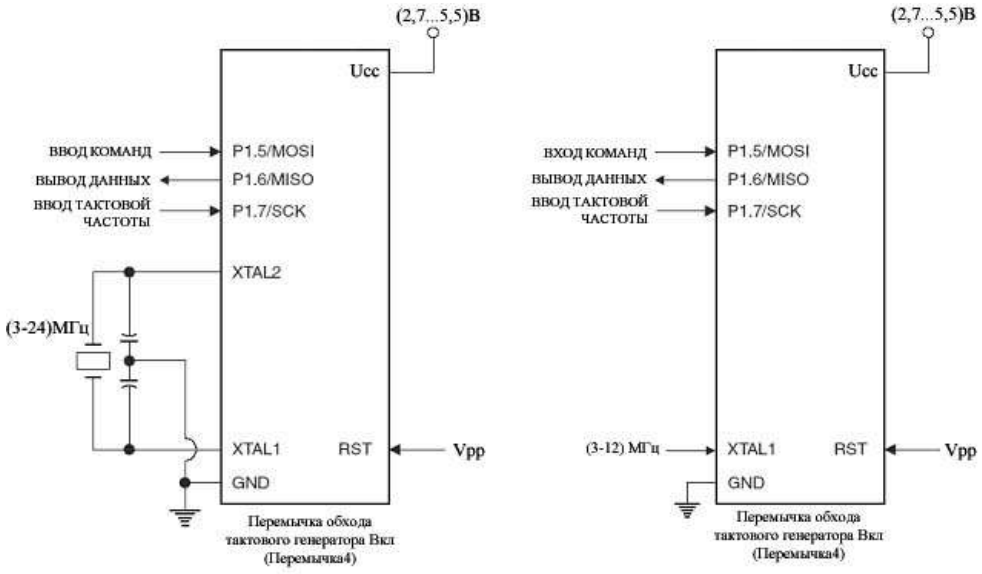
Считывание байтов сигнатур осуществляется с помощью процедуры, аналогичной проверке адресов 030H и 031H, за исключением того, что P3.6 и P3.7 должны быть переведены в состояние логического нуля. Ответные значения выглядят: (030H) = 1EH.

8.2 Последовательное программирование

При поставке микроконтроллер подготовлен к работе в режиме последовательного программирования. При последовательном программировании реализован четырехбайтный последовательный протокол.

Последовательный интерфейс состоит из выводов SCK, MOSI (вход) и MISO (выход). После перевода RST в высокое логическое состояние необходимо исполнить команду «Разрешение Программирования» прежде, чем другие команды. Процедура «Стирание Кристалла» превращает содержимое каждого адреса памяти в блоках программ и данных в состояние «FFH».

Схема включения микросхемы при последовательном программировании приведена на рисунке 13.



а) б)

Рисунок 13 - Схема включения микросхемы при последовательном программировании: а) – с кварцевым резонатором;
б) – с внешним синхросигналом.

Набор команд программирования приведен в таблице 32.

Таблица 32 - Набор команд программирования

Команда	Формат команды					Действие
	Байт 1	Байт 2	Байт 3	Байт 4	Байт n	
Разрешение программирования	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx		Разрешает последовательное программирование, когда RST находится в высоком логическом состоянии.
Стирание кристалла	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx		Стирание кристалла для матриц памяти 12 Кбайт и 2 Кбайт.
Запись данных в память программ (побайтовый режим)	0100 0000	xxA13A12 A11A10A9A8	A7A6A5A4 A3A2A1A0	D7D6D5D4 D3D2D1D0		Запись данных в память программ – побайтовый режим.
Считывание памяти программ (побайтовый режим)	0010 0000	xxA13A12 A11A10A9A8	A7A6A5A4 A3A2A1A0	D7D6D5D4 D3D2D1D0		Считывание данных из памяти программ – побайтовый режим.
Запись данных в память программ (постраничный режим)	0101 0000	xxA13A12 A11A10A9A8	A7A600 0000	Байт 0 … Байт 63		Запись данных в память программ – побайтовый режим (64 байта).
Считывание памяти программ (постраничный режим)	0011 0000	xxA13A12 A11A10A9A8	A7A600 0000	Байт 0 … Байт 63		Считывание данных из памяти программ – побайтовый режим (64 байта).
Запись данных в память программ (побайтовый режим)	1100 0000	xxxx xA10A9A8	A7A6A5A4 A3A2A1A0	D7D6D5D4 D3D2D1D0		Запись данных в память программ – побайтовый режим.
Считывание памяти программ (побайтовый режим)	1010 0000	xxxx xA10A9A8	A7A6A5A4 A3A2A1A0	D7D6D5D4 D3D2D1D0		Считывание данных из памяти программ – побайтовый режим.
Запись данных в память программ (постраничный режим)	1101 0000	xxxx xA10A9A8	A7A6A50 0000	Байт 0 … Байт 31		Запись данных в память программ – побайтовый режим (32 байта).
Считывание памяти программ (постраничный режим)	1011 0000	xxxx xA10A9A8	A7A6A50 0000	Байт 0 … Байт 31		Считывание данных из памяти программ – побайтовый режим (32 байта).
Запись пользовательских перемычек	1010 1100	0001 FUSE4 FUSE3 FUSE2 FUSE1	xxxx xxxx	xxxx xxxx		Запись разрядов пользовательских бит конфигурации.
Считывание пользовательских перемычек	0010 0001	xxxx xxxx	xxxx xxxx	xxxx FUSE4 FUSE3 FUSE2 FUSE1		Обратное считывание состояния разрядов пользовательских бит конфигурации.
Запись разрядов бит блокировки	1010 1100	1110 0LB3LB2LB1	xxxx xxxx	xxxx xxxx		Запись бит блокировки (запись «0» для запирания).
Считывание разрядов бит блокировки	0010 0100	xxxx xxxx	xxxx xxxx	xxxx xLB3LB2LB1		Считывание состояния обратного тока для бит блокировки (программируемый разряд запирания считывает обратно как «0»).
Запись байта сигнатуры пользователя	0100 0010	xxxx xxxx	xxA5 A4A3A2A1A0	D7D6D5D4 D3D2D1D0		
Считывание байта сигнатуры пользователя	0010 0010	xxxx xxxx	xxA5 A4A3A2AA0	D7D6D5D4 D3D2D1D0		
Запись страницы сигнатуры пользователя	0101 0010	xxxx xxxx	xxxx xxxx	Байт 0 … Байт 63		
Считывание страницы сигнатуры пользователя	0011 0010	xxxx xxxx	xxxx xxxx	Байт 0 … Байт 63		
Считывание байтов сигнатуры	0010 1000	xxxx xxxx	xxA5 A4A3A2A1A0	D7D6D5D4 D3D2D1D0		Считывание байта сигнатуры.

Блоки памяти программ и данных можно программировать, используя последовательную шину SPI в случае, если RST переведено в состояние U_{CC}.

Матрицы памяти программ и памяти данных имеют раздельные адресные пространства: с 0000H по 2FFFH для памяти кодов и с 000H по 7FFH для памяти данных. Внешняя синхронизация системы передается на вывод XTAL1. Максимальная последовательная тактовая частота (SCK) должна составлять не менее 1/16 от частоты кварцевого генератора. Для тактового генератора частотой 24 МГц максимальная частота SCK составляет 1,5 МГц.

В режиме последовательного программирования процедура стирания памяти инициируется также при команде «Стирание Кристалла». В этом режиме стирание памяти автоматически синхронизируется и занимает примерно 8 мс.

Как правило, используется следующий алгоритм последовательного программирования:

- подать питание на выводы U_{CC} и GND;

- установить вывод RST в состояние “H”. Если к выводам XTAL1 и XTAL2 не подсоединен кварцевый резонатор, то тактовый сигнал с частотой от 3 до 24 МГц подать на вывод XTAL1;

- выждать, по меньшей мере, 10 мс, при этом вывод RST должен находиться на высоком логическом уровне, а P1.7 (SCK) на низком логическом уровне;

- начать процедуру последовательного программирования, подав на вывод MOSI/P1.5. команду «Разрешение Программирования». Частота тактового генератора сдвига, подаваемого на вывод SCK/P1.7, должна быть меньше, чем тактовая частота ЦПУ на выводе XTAL1, поделенная на 16;

- матрица памяти программ или данных программируется путем передачи информации об адресе и о данных одновременно с соответствующей командой «Запись». Цикл записи самосинхронизирующийся и занимает менее 4,0 мс;

- можно проверить любой адрес памяти, используя команду «Считывание», которая возвращает содержимое в выбранном адресе на последовательный выход MISO/P1.6;

- в завершение сеанса программирования на RST установить нижний логический уровень для того, чтобы начать обычный рабочий цикл.

В случае необходимости, последовательность отключения выглядит следующим образом:

- установить XTAL1 на “L” (если не используется кварцевый генератор);

- установить RST на “L”;

- отключить напряжение питания U_{CC}.

После перехода сигнала переустановки в высокое логическое состояние, SCK должен находиться в низком логическом состоянии, по крайней мере, в течение 64 системных тактов перед тем, как он переходит в высокое логическое состояние для синхронизации в байтах разрешения данных. При этом не требуется подача импульсов сигнала переустановки. SCK не должен превышать 1/16 сигнала синхронизации на выводе XTAL1.

Временные диаграммы при последовательном программировании (SPI, Режим 1 ––> CPOL = 0, CPHA = 1) приведены на рисунке 14, а соответствующие справочные режимные характеристики – в таблице 33.

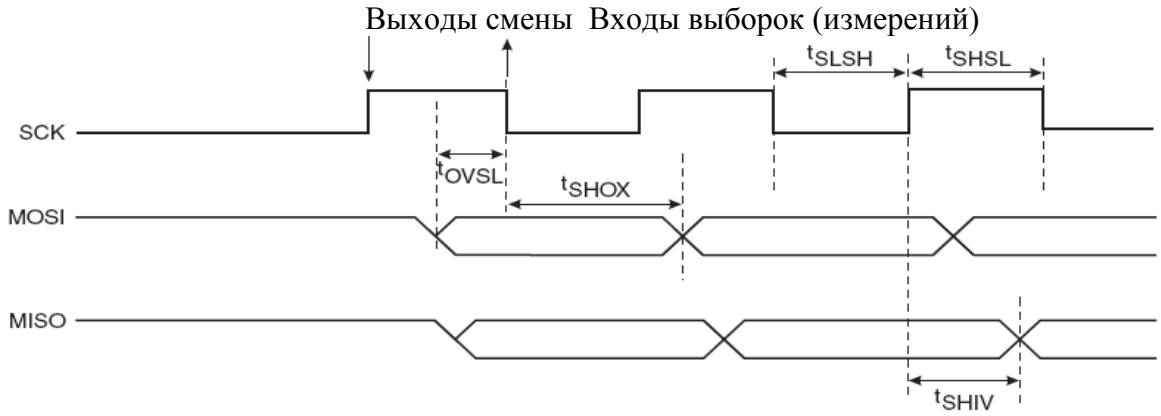


Рисунок 14 - Временные диаграммы при последовательном программировании

Для страницы Запись/Считывание данные всегда должны начинаться с байта 0 до 31 или 63. После того, как байт команды и байт верхнего адреса заблокированы, каждый последующий байт обрабатывается как данные, до тех пор, пока все 32 или 64 байта не переместятся в устройство или из него. Затем будет готова к дешифрации следующая команда.

Таблица 33 - Характеристики последовательного программирования

Символ	Параметр	Мин	Тип	Макс	Единицы
1	2	3	4	5	6
$1/t_{CLCL}$	Частота генератора	3		24	МГц
t_{CLCL}	Период генератора	41,6		33,3	нс
t_{SHCH}	Ширина положительного полупериода имульса SCK	8 t_{CLCL}			нс
t_{SLSH}	Ширина отрицательного полупериода имульса SCK	8 t_{CLCL}			нс
t_{OVSL}	Предустановка MOSI к моменту переключения SCK в состояние низкого логического уровня	t_{CLCL}			нс
t_{SHOX}	Удержание MOSI после момента переключения SCK в состояние низкого логического уровня	2 t_{CLCL}			нс
t_{SHIV}	Предустановка SCK до появлении на MISO достоверных данных	10		32	нс
t_{ERASE}	Длительность цикла стирания			9	мс
t_{SWC}	Длительность цикла последовательной записи страницы			4,5	мс
U_{CC} = (4,5 – 5,5) В					

8.3 Режим записи в ЭСППЗУ

На рисунке 15 показаны диаграммы режима записи в ЭСППЗУ при малой частоте синхронизации и при высокой частоте синхронизации.

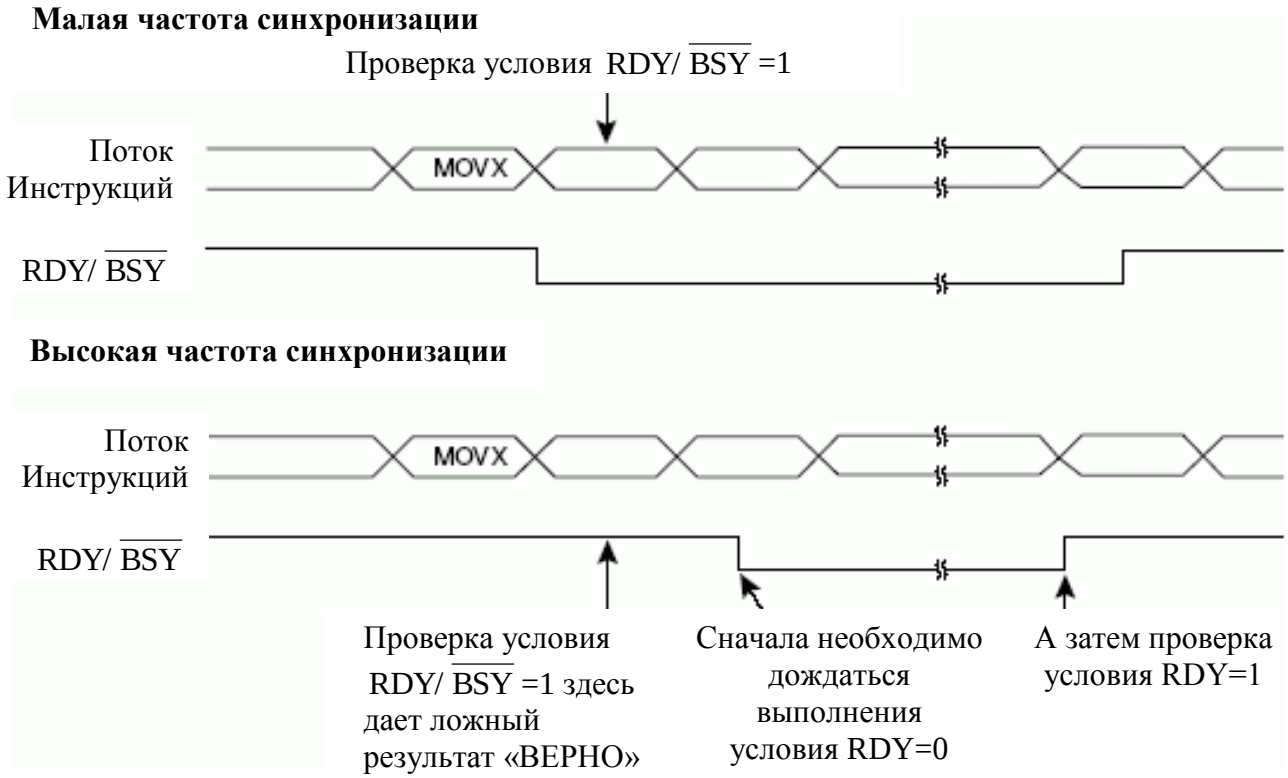


Рисунок 15- Режим записи в ЭСППЗУ

При малой частоте синхронизации может использоваться следующий код для записи одной страницы данных в ЭСППЗУ 1882ВЕ53У:

```
EEMWE EQU 00010000B     ; Бит разрешения записи данных в ЭСППЗУ
EEMEN EQU 00001000B     ; Бит разрешения доступа к внутреннему
ЭСППЗУ
EELD EQU 00100000B      ; Бит разрешения загрузки ЭСППЗУ
ORL EECON,#EEMEN        ; Разрешаем доступ к ЭСППЗУ
ORL EECON,#EEMWE        ; Разрешаем запись в ЭСППЗУ
ORL EECON,#EELD         ; Разрешаем загрузку страницы ЭСППЗУ
MOV R7,#1FH             ;0x1FH = 31
PAGELOAD:
MOV A,#055H             ; помещаем записываемый байт в
аккумулятор
MOVX @DPTR,A            ; помещаем этот байт в буфер данных
INC DPTR                ; инкрементируем DPTR(адрес ЭСППЗУ)
DJNZ R7,PAGELOAD        ; проверяем на загрузку 31 байта
; ЗАГРУЗКА ПОСЛЕДНЕГО БАЙТА И ИНИЦИАЦИЯ СТРАНИЧНОЙ ЗАПИСИ
MOV A,#055H             ; помещаем записываемый байт в
аккумулятор
XRL EECON,#EELD         ; следующая инструкция MOVX вызовет
запуск записи страницы
MOVX @DPTR,A            ; загрузка байта в буфер
```

При более высоких частотах, когда опрашивается состояние бита $\mathrm{RDY}/\overline{\mathrm{BSY}}$ (готовность/занято) в процессе записи ЭСППЗУ микросхемы 1882ВЕ53У, необходимо определять как начало, так и завершение цикла записи ЭСППЗУ. При этом может использоваться следующий код для записи одной страницы данных в ЭСППЗУ микросхемы 1882ВЕ53У:

```
WRITESTART:          ; ожидание запуска цикла записи (RDY/BSY#=0)
MOV VAR,096H         ;096H - адрес регистра EECON в SFR
JB VAR.1,Writestart
WAIT:                ; ожидаем завершения цикла записи
MOV VAR,096H
JNB VAR.1,Wait
```

8.4 Защита внутренней памяти программ

Микроконтроллер имеет три разряда защиты, которые могут быть оставлены незапрограммированными (Н) или могут быть запрограммированы (З) для обеспечения дополнительных возможностей.

Если Бит 1 запрограммирован, логический уровень на выводе $\overline{\mathrm{EA}}$ опрашивается и фиксируется во внутренней защелке во время сброса.

Биты защиты могут быть перепрограммированы после стирания кристалла в параллельном или последовательном режиме.

В таблице 34 приведено описание бит защиты внутреннего ПЗУ.

Таблица 34 - Биты защиты внутреннего ПЗУ

Биты программной защиты				Тип защиты
Режим	LB1	LB2	LB3	
1	Н	Н	Н	Отсутствует блокировка внутренней памяти.
2	З	Н	Н	Команда, выбранная из внешней памяти, не имеет доступа к внутренней памяти программ (для команды MOVC). Дальнейшее программирование невозможно.
3	З	З	Н	Аналогично Режиму 2, но при этом параллельная или последовательная проверка внутренней памяти также блокируются.
4	З	З	З	Аналогично Режиму 3, но невозможно выполнение программ из внешнего ПЗУ.

9 Методы адресации

Реализованы следующие методы адресации:

- косвенно-регистровая адресация;

- непосредственная адресация;
- прямая байтовая адресация;
- регистровая адресация;
- косвенно-регистровая адресация по сумме.

Косвенно-регистровая адресация используется для обращения к ячейкам внутреннего ОЗУ данных. В качестве регистров-указателей используются регистры R10, R1 выбранного банка регистров. В командах PUSH и POP используется содержимое указателя стека (SP). Косвенно-регистровая адресация используется также для обращения к внешней памяти данных. В этом случае с помощью регистров-указателей R0 и R1 (выбранного банка рабочих регистров) выбирается ячейка из блока в 256 байт внешней памяти данных. Номер блока предварительно задается содержимым порта P2. 16-разрядный указатель данных (DPTR) может быть использован для обращения к любой ячейке адресного пространства внешней памяти данных объемом до 64К байт.

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

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.

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.

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.

10 Общая характеристика системы команд

Система команд МК включает в свой состав 111 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 24 МГц соответственно), исключение составляют лишь команды умножения и деления, которые выполняются за 4 машинных цикла (2,0 мкс).

11 Программаторы и отладочные средства

11.1 Программаторы для программирования микроконтроллера

Могут использоваться программаторы любой фирмы, обеспечивающие программирование аналогов микроконтроллера (AT89S8253 ф. Atmel), например, типа PicProg+, ChipProg, ChipProg+ ООО «Фитон» и др.

11.2 Описание инструментальных средств для ИМС

Инструментальные средства разработки и отладки систем на базе микроконтроллеров (аналог ИМС AT89S8253 ф. Atmel) поставляются многими фирмами. Важной особенностью внутрисхемных эмуляторов является то, что архитектура микроконтроллера (в том числе и аналога) не позволяет построение на его основе отладочного устройства, эмулирующего все порты ввода-вывода информации. В поставляемых аппаратных отладчиках не обеспечивается эмуляция портов P0 и P2 микроконтроллера. Программист в каждом конкретном случае должен сам решать вопросы использования портов P0 и P2 при отладке систем.

Ниже приводится краткая информация о средствах отладки, поставляемых ООО «Фирма Фитон», г. Москва, обеспечивающих отладку систем на базе разработанного микроконтроллера при условии модернизации жгута подключения (адаптер) под корпусы Н16.48-2В, 5133.48-3.

Реквизиты ООО "Фирма ФИТОН":

Адрес: Москва, 127015, ул. Новодмитровская, д. 5а, 11 этаж, оф. 1103

Телефон/факс: (495) 730-75-84 (многоканальный)

Электронная почта: PHYTON@phyton.ru

Интернет-телефон (Skype): phytonmsk

Пакет Project-51 – набор программно-аппаратных средств, предназначенный для разработки и отладки систем на базе микроконтроллеров, совместимых с семейством 8051. Концепция Project-51: объединение внутрисхемного эмулятора, программного отладчика-симулятора, компиляторов, текстового редактора, менеджера проектов и программатора в рамках интеллектуальной единой среды разработки. При наличии одного из программаторов PicProg+, ChipProg, ChipProg+ пакет поддерживает работу и с программатором. Встроенные многооконный редактор, менеджер проектов и большое количество сервисных возможностей облегчают труд разработчика. Полная конфигурация пакета называется Project-51/ESA и включает в себя:

- менеджер проектов;
- кросс-компилятор языка ассемблер MCA-51;
- отладчик-симулятор PDS-51;
- внутрисхемный эмулятор PICE-51;
- описание внутрисхемного эмулятора.

Программная поддержка PICE-51 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, сменного пода под определенную группу процессоров и сменного адаптера под конкретный тип корпуса. На основной плате реализованы трассировщик и процессор точек останова. Плата сменного пода содержит эмулирующий процессор под конкретный тип микроконтроллера. Сменные адаптеры обеспечивают установку эмулятора на посадочные места на плате пользователя. Питание эмулятора осуществляется от блока питания или непосредственно от отлаживаемого устройства. Связь с компьютером - по гальванически развязанному каналу RS-232C на скорости 115 кбод или по каналу USB.

Эмулятор обеспечивает:

- 256 Кбайт эмулируемой памяти программ и данных. Поддержка банкированной модели памяти. Распределение памяти между эмулятором и устройством пользователя с точностью до 1-го байта;
- аппаратные точки останова по доступу к памяти программ и данных;
- аппаратную поддержку для отладки программ на языках высокого уровня;
- трассировку 8 произвольных внешних сигналов;
- 4 выхода синхронизации аппаратуры пользователя;
- трассировку адреса, данных, сигналов управления, таймера реального времени и 8-ми внешних сигналов пользователя. Трассировщик

реального времени имеет буфер объемом от 16 до 64 Кфреймов по 64 бита с доступом "на лету";

- программируемый фильтр трассировки;
- аппаратный процессор точек останова с возможностью задания сложного условия останова эмуляции по комбинации сигналов адреса, данных, управления, 8-ми внешних сигналов, таймера реального времени, счетчиков событий и таймера задержки;
- четыре комплексных точки останова, которые могут быть использованы независимо или в комбинациях по условиям AND/OR/IF-THEN;
- 48-разрядный таймер реального времени;
- прозрачную эмуляцию - доступ "на лету" к эмулируемой памяти, точкам останова, процессору точек останова, буферу трассировки, таймеру реального времени;
- управляемый генератор тактовой частоты для эмулируемого процессора с возможностью плавного изменения тактовой частоты от 500 кГц до 40 МГц;
- гальванически развязанный от компьютера канал связи RS-232C со скоростью обмена 115 кбод;
- встроенную диагностику аппаратуры эмулятора.

Характеристика программного обеспечения:

- программное обеспечение ориентировано на работу в среде Windows;
- поддерживается разработка программ на уровне ведения проектов для макроассемблера MCA-51, который входит в комплект поставки, а также для пакетов кросс-средств фирм Keil Software и IAR Systems;
- PICE-51 обеспечивает символьную отладку и отладку по исходному тексту для программ, созданных с помощью следующих компиляторов:
- ассемблер ASM51 фирмы Intel;
- компилятор PL/M фирмы Intel;

- ассемблер и компилятор Си фирмы IAR Systems;
- ассемблер и компилятор Си фирмы Avocet Systems Inc./HiTech;
- ассемблер и компилятор Си фирмы Keil Software Inc.;
- автоматическое сохранение и загрузка файлов конфигурации аппаратуры, интерфейса и опций отладки. Обеспечивается совместимость файлов конфигурации с симулятором PDS-51. Обеспечена переносимость проектов между эмулятором PICE-51 и симулятором PDS-51;
- возможность настройки цветов, шрифтов и других параметров для всех окон одновременно и для каждого окна в отдельности;
- для обновления версий PICE-51 достаточно обновить его программное обеспечение.
- эмулятор снабжен печатным руководством по эксплуатации и контекстным электронным руководством, в которых детально описаны его принципы работы, команды, меню, горячие клавиши.

Компоненты эмулятора

Для эмулятора PICE-51 существует несколько вариантов основной платы, различающихся по скорости, объему памяти и, соответственно, по цене.

Комплект поставки эмулятора PICE-51:

- 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.

Симулятор PDS-51

Симулятор PDS-51 представляет собой программно-логическую модель микроконтроллера. Возможности PDS-51:

- 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-51 - точная симуляция узлов микроконтроллера и возможность моделировать устройства, подключенные к микроконтроллеру "снаружи", например, внешнюю логику, датчики, клавиатуру, исполнительные устройства (дисплеи), задавать периодические и непериодические воздействия и т. п.

Кросс-макроассемблер

Кросс-макроассемблер предназначен для трансляции исходных текстов программ и обеспечивает:

- генерацию HEX-файла и подробного листинга;
- поддержку набора директив условной трансляции.

Поставляется кросс-макроассемблер как в составе пакета Project-51, так и отдельно.

12 Заключение

В руководстве КФДЛ.431281.032 приведено описание архитектуры, функционального построения, системы команд и особенностей применения 8-разрядного микроконтроллера 1882ВЕ53У с внутренней программной Flash-памятью объемом 12 Кбайт и 2 Кбайт ЭСППЗУ памятью данных.

Все значения электрических параметров ИМС приведены в базовых технических условиях АЕЯР.431280.286 и технических условиях исполнения АЕЯР.431280.286-02 на изделия. Значения параметров, приведенных в руководстве, являются справочными.

Настоящее руководство может служить практическим пособием по применению микроконтроллеров 1882ВЕ53У для разработчиков систем на их основе.

Приложение А

(обязательное)

Система команд ИМС

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

Основные функциональные блоки МК: ПЗУ, ОЗУ, регистры специальных функций, 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 МК 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 и 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 А.1.

Таблица А.1 - 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
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
 + - * / AND OR XOR /X	Действия: сложения вычитания умножения деления логического умножения (операция И) логического сложения (операция ИЛИ) сложения по модулю 2 (исключающее ИЛИ) инверсия элемента X

В таблице А.2 приведен перечень команд, упорядоченных по алфавиту.

Таблица А.2 - 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
Команда DJNZ <байт>, <смещение>	Декремент и переход, если не равно нулю	
Команда INC <байт>	Инкремент	
Команда INC DPTR	Инкремент указателя данных	
Команда JB <bit>, <re1 8>	Переход, если бит установлен	
Команда JBC <bit>, <re1 8>	Переход, если бит установлен и сброс этого бита	
Команда JC <re18>	Переход, если перенос установлен	
Команда JMP @A+DPTR	Косвенный переход	
Команда JNB <bit>, <re1 8>	Переход, если бит не установлен	
Команда JNC <re1 8>	Переход, если перенос не установлен	
Команда JNZ <re1 8>	Переход, если содержимое А не равно нулю	

Продолжение таблицы А.2

1	2	3
Команда JZ <re1 8>	Переход, если содержимое аккумулятора равно 0	
Команда LCALL <addr 16>	Длинный вызов	
Команда LJMP <addr 16>	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
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	

Окончание таблицы А.2

1	2	3
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	

Мнемонические обозначения функций однозначно связаны с конкретными комбинациями способов адресации и типами данных. Всего в системе команд возможно 111 таких сочетаний.

Описание команд

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 Ф с содержимым байта-источника, оставляя результат в аккумуляторе. При появлении переносов из разрядов 7 и 3 устанавливаются флаги переноса (С) и дополнительного переноса (АС) соответственно, в противном случае эти флаги сбрасываются. При сложении целых чисел без знака флаг переноса "С" 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 флаг 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 2К байт 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 "C" устанавливается в "1", если значение целого без знака <байта назначения> меньше, чем значение целого без знака <байта источника>, в противном случае 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 А. 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 "С" 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 использоваться 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), значение 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 преобразование 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) сбрасываются. 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 ИМС.

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>, <re1 8>

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>, <re1 8>

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 <re1 8>

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>, <re1 8>

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 <re1 8>

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 <re1 8>

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 <re1 8>

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 <addr 16>

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 К байт. 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 <addr 16>

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 команд, 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 64 Кбайт), 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 устанавливается 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. На 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 во 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 во 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) и на флаги эта команда не влияет.

```
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:

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: 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.

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ докум.	Подп.	Дата
	изменен-ных	заменен-ных	новых	анну-лиро-ванных				
-	-	-	Все	-	143			07.02.08
1	-	79, 87-92	-	-	-			03.09.08
2	-	68	-	-	-			29.09.08
3	-	33, 34	-	-	-			02.06.09
4	-	10-13, 31-34	-	-	-			04.06.10
5	-	3	-	-	-			20.02.13
6	-	4, 13, 87	-	-	-			06.05.14
7	-	51, 64	-	-	-			03.10.14
8	-	4, 8. 9, 87	-	-	-			25.11.15