МИКРОСХЕМЫ ИНТЕГРАЛЬНЫЕ Л1874ВЕ36, 1874ВЕ36, Л1874ВЕ36А, 1874ВЕ36А

Руководство пользователя

2012

Содержание

Введение

В настоящем руководстве КФДЛ.431295.008 приведено подробное описание архитектуры, функционального построения, системы команд и особенностей применения ИС серии 1874: Л1874ВЕ36, 1874ВЕ36, Л1874ВЕ36А, 1874ВЕ36А, которые представляют собой СБИС однокристальных 16-разрядных микроконтроллеров с внутренней программной памятью объемом 8 Кбайт.

Микроконтроллеры имеют высокую производительность, развитую систему встроенных функциональных блоков и достаточно мощную систему команд и предназначены для применения во встроенных системах для реализации функций управления и вычисления.

Изделия Л1874ВЕ36, 1874ВЕ36, Л1874ВЕ36А, 1874ВЕ36А выпускаются и поставляются по техническим условиям АЕЯР.431280.169: ИС Л1874ВЕ36, Л1874ВЕ36А – в корпусах 6108.68-1; ИС 1874ВЕ36, 1874ВЕ36А – в корпусах 4235.88-1 .

Микросхемы Л1874ВЕ36, 1874ВЕ36, Л1874ВЕ36А, 1874ВЕ36А могут поставляться как без встроенной памяти программ, так и с внутренним масочным ПЗУ программ.

Функциональным аналогом ИС является микроконтроллер 83С196КВ12 фирмы Intel.

Для разработки и отладки систем на основе микроконтроллеров разработаны и могут поставляться программно-аппаратные средства отладки:

- внутрисхемный эмулятор PICE-196 с соответствующим сменным адаптером (POD для ИС 1874ВЕ36, 1874ВЕ36А) и эмуляционной головкой;
- программный Отладчик-Симулятор PDS-96;
- кросс-компилятор Си МСС-96 и макроассемблер МСА-96.

Все указанные средства поддержки поставляются АО "Фитон", г. Москва.

Настоящее руководство может служить практическим пособием по применению микроконтроллеров для разработчиков систем на основе ИС Л1874ВЕ36, 1874ВЕ36, Л1874ВЕ36А, 1874ВЕ36А.

При обнаружении неточностей и опечаток просьба сообщать разработчикам:

- Тел.: (473) 226-20-35
- Факс: (473) 226-98-95
- E-mail: niiet@niiet.ru
- http://www.niiet.ru

1 Назначение и основные технические характеристики микросхем

Микросхемы представляют собой СБИС однокристальных 16-разрядных микроконтроллеров с внутренней программной памятью объемом 8 Кбайт. Микроконтроллеры предназначены для применения в системах управления и диагностики автомобильных двигателей. В области промышленного производства микроконтроллеры могут быть использованы для управления робототехническими комплексами, в системах автоматизации технологических процессов, в системах автоматизированного управления электроприводом, оргтехнике, вычислительной технике, телекоммуникационной технике и т. п.

1.1 Архитектурные характеристики микроконтроллеров:

- количество 8-разрядных портов ввода-вывода – 5;
- разрядность шины адресов-данных, бит – 8, 16;
- емкость регистрового блока, байт – 232;
- разрядность РАЛУ, бит – 16;
- объем адресуемой памяти, байт – 64К;
- объем внутренней памяти, байт – 8К;
- количество типов команд – 105;
- разрядность команды, байт – 1, …, 8;
- количество 16-разрядных таймеров – 6 (в том числе 4 программных);
- количество источников прерываний/векторов – 28/16;
- полностью дуплексный последовательный порт;
- высокоскоростная подсистема ввода-вывода;
- блок формирования сигналов ШИМ;
- 8-канальный 10-разрядный АЦП;
- режим поддержки мультипроцессорного протокола (HOLD#/HLDA#);
- режимы пониженного потребления мощности и хранения;
- 16-разрядный дежурный таймер.

1.2 Конструктивные характеристики микросхем

Кристалл микросхем ПАКД.757644.231 выполнен по КМОП технологии с поликремниевыми затворами, одним уровнем металлизации и n-карманом.

Размер кристалла (7,84 × 6,74) мм^2.

Микросхемы выполняются в двух вариантах:

1) 68-выводном металлокерамическом корпусе с матричным расположением выводов (PGA) типа 6108.68-1 ("Марево-68") – ИС Л1874ВЕ36, Л1874ВЕ36А;

2) 88-выводном металлокерамическом корпусе с планарным расположением выводов (LCC) типа 4235.88-1 ("Монополия-88") – ИС 1874ВЕ36, 1874ВЕ36А.

1.3 Электрические характеристики микросхем:

- номинальное значение напряжения питания, В – 5,0;
- допустимые отклонения напряжения питания от номинального значения, В – ±0,5;
- потребляемая мощность, мВт – 500;
- частота тактового сигнала, МГц – (3,5 – 20);
- диапазон рабочих температур – от минус 60 до 85 °С.

Примечание – Все значения электрических параметров ИС приведены в технических условиях АЕЯР.431280.169. Значения параметров являются справочными.

Электрические параметры микросхем при приемке и поставке приведены в таблице 1.3.1.

Значения предельно допустимых и предельных режимов эксплуатации микросхем приведены в таблице 1.3.2.

Условное графическое обозначение микросхем приведено на рисунках 1.3.1 и 1.3.2. Функциональное назначение выводов приведено в таблице 1.3.3.

Таблица 1.3.1 – Электрические характеристики микросхем при приёмке и поставке (справочные данные)

Наименование параметра, единица измерения	Буквенное обозначение параметра	Норма		Режим измерения	Примечание
		не менее	не более		
1 Выходное напряжение низкого уровня, В	U_{OL}	–	0,45	I_{OL} = 3,2 мА	1
2 Выходное напряжение высокого уровня, В	U_{OH1}	3,8	–	I_{OH} =–0,03 мА	2
3 Выходное напряжение высокого уровня, В	U_{OH2}	3,8	–	I_{OH} = –3,2 мкА	3
4 Входной ток высокого уровня, мкА	I_{IH}	–650	–	U_{IH} = 2,0 В	8
5 Входной ток низкого уровня, мкА	I_{IL}	–50	–	U_{IL} = 0,45 В	9
6 Ток утечки на входе, мкА	I_{ILH1} I_{ILL1}	–10	10	U_{IH} = 5,2 В U_{IH} = 5,5 В	4 5
7 Ток утечки на входе, мкА	I_{ILH2} I_{ILL2}	–3	3	U_{IH} = 5,5 В	6 7
8 Ток потребления первого источника питания, мА	I_{OCC1}	–	100	U_{CC1} = 5,5 В U_{CC2} =5,5 В	10
9 Ток потребления второго источника питания, мА	I_{OCC2}	–	10	U_{CC1} = 5,5 В U_{CC2} = 5,5 В	11
10 Время задержки распространения сигнала от входа XTAL1 до выхода CLCOUT, не более, нс	t_p (C,LH-CO,H)	–	85	U_{CC1} = 5,0 В U_{CC2} = 5,0 В U_{IH} = 3,0 В	
11 Функциональный контроль	ФК1 ФК2				12 13

Примечания

1 Для выходов AD.0 – AD.15, RD#, WR#, BHE#, INST, HSO.0 – HSO.5, PWM, CLCOUT, P2.0/TXD, RXD (в режиме 0), P1.0 – P1.7, P2.6, P2.7.

2 Для выходов P1.0 – P1.7, P2.6, P2.7.

3 Для выходов AD.0 – AD.15, RD#, WR#, BHE#, INST, HSO.0 – HSO.5, PWM, CLCOUT, P2.0/TXD, RXD (в режиме 0), ALE.

4 Для входов HSI.0 – HSI.3, EA#, READY, BW, NMI, P2.1/RXD, P2.2/EXTINT, P2.3/T2CLK, P2.4/T2RST. На измеряемый вывод подается U_{IH}, а на остальные U_{IL}.

5 Для входов HSI.0 – HSI.3, EA#, READY, BW, NMI, P2.1/RXD, P2.2/ EXTINT, P2.3/T2CLK, P2.4/ T2RST. На измеряемый вывод подается U_{IL}, а на остальные U_{IH}.

6 Для входов P0.0 – P0.7. На измеряемый вход подается U_{IH}, а на остальные U_{IL}.

7 Для входов P0.0 – P0.7. На измеряемый вход подается U_{IL}, а на остальные U_{IH}.

Окончание таблицы 1.3.1

8 Для входов/выходов P1.0 – P1.7, P2.6 , P2.7. На измеряемый вывод подается U_{IH}, а на остальные U_{IL}.

9 Для входов/выходов P1.0 – P1.7, P2.6 , P2.7. На измеряемый вывод подается U_{IL}, а на остальные U_{IH}.

10 Измерение тока потребления первого источника питания при U_{CC1} = 5,5 В, f_C =12/20 МГц*.

11 Измерение тока потребления аналого-цифрового преобразователя при f_C =12/20 МГц*.

12 Функциональный контроль ФК1 проводится при:

- f_C = (3,5 – 20) МГц*;
- U_{IL} по входам: P0.0 – 0,2 В, P0.1 – 0,16 В, P0.2 – 0,32 В, P0.3 – 0,64 В;
- U_{IH} по входам: P0.4 – 2,8 В, P0.5 – 3,2 В, P0.6 – 3,8 В, P0.7 – 4,2 В;
- по остальным входам: уровни U_{IH} = 3 В, U_{IL} = 0 В.

13 Функциональный контроль ФК2 проводится при f_C = 3,5 МГц.

* 20 МГц – для ИС Л1874ВЕ36, 1874ВЕ36; 12 МГц – для Л1874ВЕ36А, 1874ВЕ36А.

Таблица 1.3.2 – Предельно допустимые и предельные режимы эксплуатации микросхем

Наименование параметра режима, единица измерения	Буквенное обозначение параметра	Предельно допустимый режим		Предельный режим
		не менее	не более	
1 Напряжение 1-го источника питания, В	U_{CC1}	4,5	5,5	7,0
2 Напряжение 2-го источника питания, В	U_{CC2}	4,0	5,5	7,0
3 Частота следования импульсов тактового сигнала, МГц	f_C	3,5	12/20*	
4 Входное напряжение низкого уровня, В	U_{IL}	0	0,8	– 0,5
5 Входное напряжение высокого уровня, В	U_{IH}	$0,2U_{CC1}$+1,0	U_{CC1}+0,5	7,0 при U_{CC1}= 7 В
6 Входное напряжение высокого уровня тактового сигнала, В	U_{IHCI}	$0,7U_{CC1}$	U_{CC1}+0,5	7,0 при U_{CC1}= 7 В
7 Входное напряжение высокого уровня сигнала "сброс", В	$U_{IHRESET}$	2,6	U_{CC1}+0,5	7,0 при U_{CC1}= 7 В
8 Выходной ток высокого уровня по выводам RD#, WR#, ALE, BHE#, INST, HSO, P2.5/PWM, CLCOUT, P3.0 – P3.7, P4.0 – P4.7, P2.0/TXD, P2.1/RXD, мА	I_{OH}		–3,2	–10
9 Выходной ток низкого уровня, мА	I_{OL}		3,2	10
10 Емкость нагрузки, пФ	C_L		80	200
11 Длительность фронтов входных сигналов,нс	t_{LH}, t_{HL}	–	7	25

Примечание – Время действия предельных режимов по пунктам 1, 2, 4 – 7 не более 10 мс.

* Показатель в числителе для ИС Л1874ВЕ36А и 1874ВЕ36А, показатель в знаменателе для ИС Л1874ВЕ36 и 1874ВЕ36.

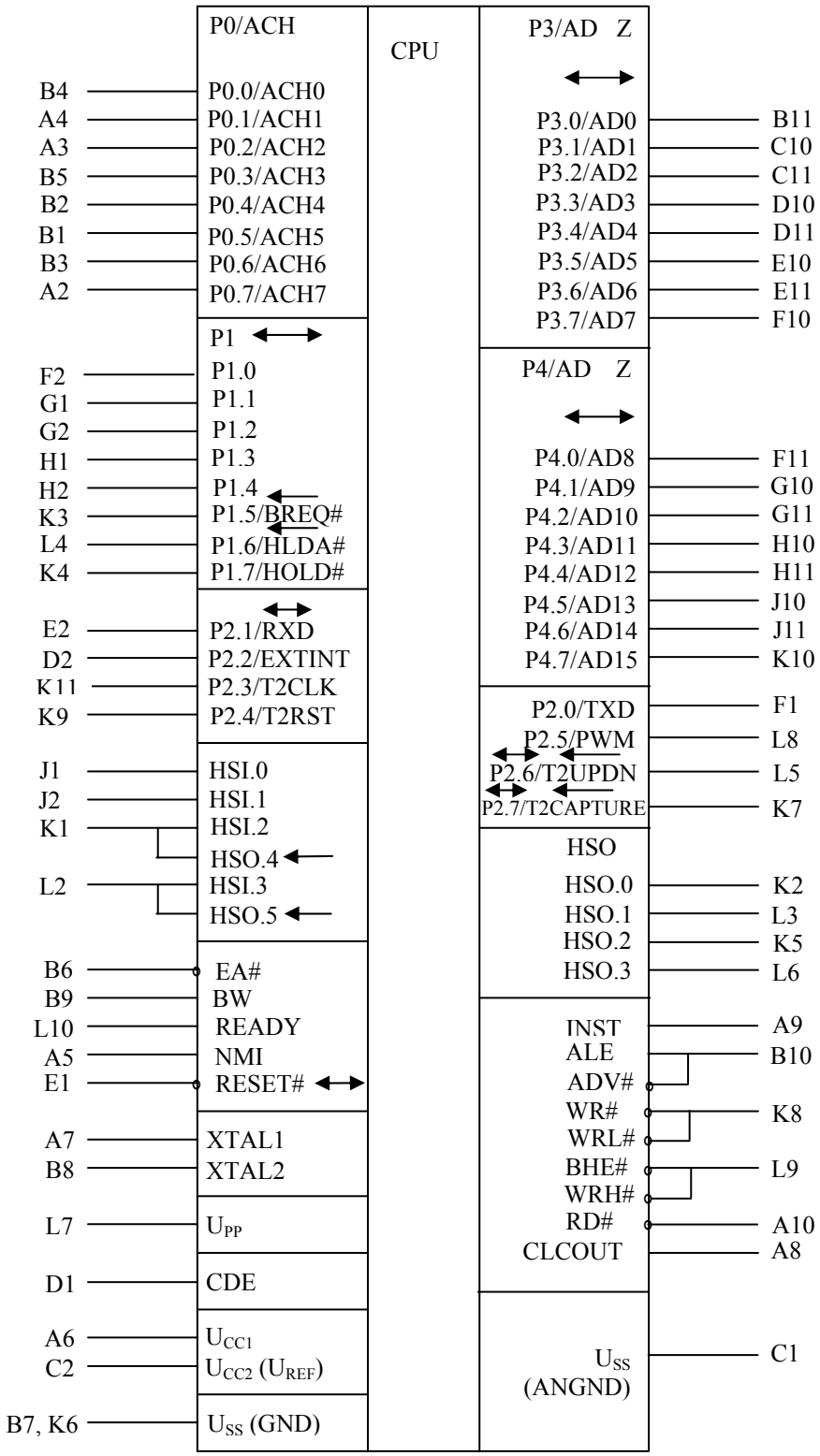


Рисунок 1.3.1 – Условное графическое обозначение микросхем Л1874ВЕ36 Л1874ВЕ36А в корпусе 6108.68-1 ("Марево-68")

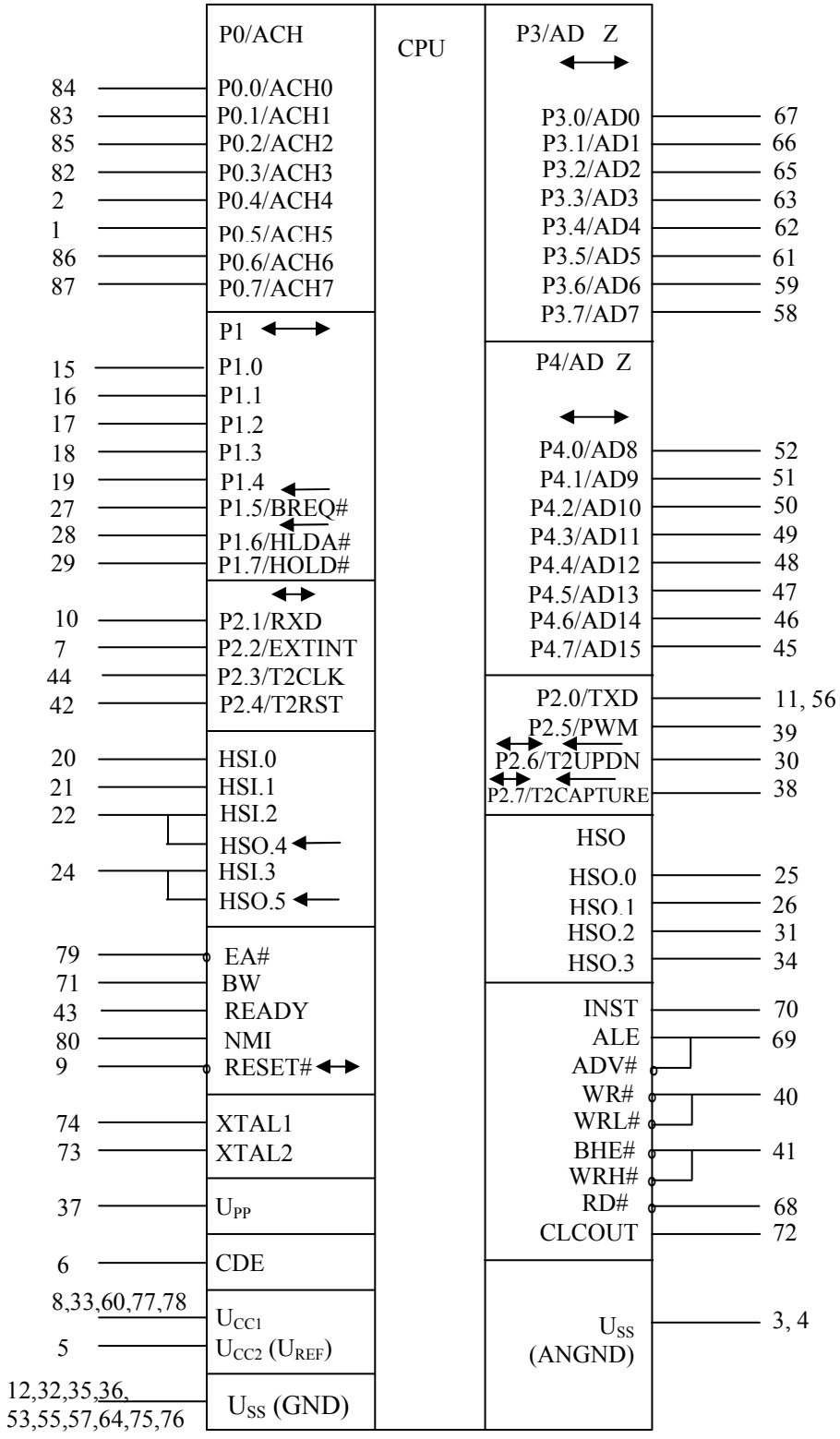


Рисунок 1.3.2 – Условное графическое обозначение микросхем 1874ВЕ36, 1874ВЕ36А в корпусе 4235.88-1 ("Монополия-88")

Таблица 1.3.3 – Функциональное назначение выводов микросхем

Обозначение вывода	Номер вывода, Тип корпуса 6108.68-1	Номер вывода, Тип корпуса 4235.88-1	Функциональное назначение	Тип вывода	Обозначение альтернативной функции вывода
1	2	3	4	5	6
P0.0	B4	84	Вход "порт 0, 0 разряд" Вход АЦП, канал 0	I I	 ACH.0
P0.1	A4	83	Вход "порт 0, 1 разряд" Вход АЦП, канал 1	I I	 ACH.1
P0.2	A3	85	Вход "порт 0, 2 разряд" Вход АЦП, канал 2	I I	 ACH.2
P0.3	B5	82	Вход "порт 0, 3 разряд" Вход АЦП, канал 3	I I	 ACH.3
P0.4	B2	2	Вход "порт 0, 4 разряд" Вход АЦП, канал 4	I I	 ACH.4
P0.5	B1	1	Вход "порт 0, 5 разряд" Вход АЦП, канал 5	I I	 ACH.5
P0.6	B3	86	Вход "порт 0, 6 разряд" Вход АЦП, канал 6	I I	 ACH.6
P0.7	A2	87	Вход "порт 0, 7 разряд" Вход АЦП, канал 7	I I	 ACH.7
P1.0	F2	15	Вход/выход "порт 1, 0 разряд"	I/O	
P1.1	G1	16	Вход/выход "порт 1, 1 разряд"	I/O	
P1.2	G2	17	Вход/выход "порт 1, 2 разряд"	I/O	
P1.3	H1	18	Вход/выход "порт 1, 3 разряд"	I/O	
P1.4	H2	19	Вход/выход "порт 1, 4 разряд"	I/O	
P1.5	K3	27	Вход/выход "порт 1, 5 разряд" Выход "запрос шины"	I/O O	 BREQ#
P1.6	L4	28	Вход/выход "порт 1, 6 разряд" Выход "подтверждение захвата шины"	I/O O	 HLDA#
P1.7	K4	29	Вход/выход "порт 1, 7 разряд" Вход "требование захвата шины"	I/O I	 HOLD#
P2.0	F1	11, 56	Выход "порт 2, 0 разряд" Выход последовательных данных	O O	 TXD
P2.1	E2	10	Вход "порт 2, 1 разряд" Вход/выход последовательных данных	I I/O	 RXD
P2.2	D2	7	Вход "порт 2, 2 разряд" Вход внешнего прерывания	I I	 EXTINT
P2.3	K11	44	Вход "порт 2, 3 разряд" Вход "синхронизация таймера 2"	I I	 T2CLK
P2.4	K9	42	Вход "порт 2, 4 разряд" Вход "сброс таймера 2"	I I	 T2RST
P2.5	L8	39	Выход "порт 2, 5 разряд" Выход ШИМ	O O	 PWM
P2.6	L5	30	Вход/выход "порт 2, 6 разряд" Вход "выбор режима таймера 2"	I/O I	 T2UPDN

Продолжение таблицы 1.3.3

1	2	3	4	5	6
P2.7	K7	38	Вход/выход "порт 2, 7 разряд" Вход "выборка данных таймера 2"	I/O I	 T2CAPTURE
P3.0	B11	67	Вход/выход "порт 3, 0 разряд" Вход/выход "адрес-данные, 0 разряд"	I/O I/O/Z	 AD0
P3.1	C10	66	Вход/выход "порт 3, 1 разряд" Вход/выход "адрес-данные, 1 разряд"	I/O I/O/Z	 AD1
P3.2	C11	65	Вход/выход "порт 3, 2 разряд" Вход/выход "адрес-данные, 2 разряд"	I/O I/O/Z	 AD2
P3.3	D10	63	Вход/выход "порт 3, 3 разряд" Вход/выход "адрес-данные, 3 разряд"	I/O I/O/Z	 AD3
P3.4	D11	62	Вход/выход "порт 3, 4 разряд" Вход/выход "адрес-данные, 4 разряд"	I/O I/O/Z	 AD4
P3.5	E10	61	Вход/выход "порт 3, 5 разряд" Вход/выход "адрес-данные, 5 разряд"	I/O I/O/Z	 AD5
P3.6	E11	59	Вход/выход "порт 3, 6 разряд" Вход/выход "адрес-данные, 6 разряд"	I/O I/O/Z	 AD6
P3.7	F10	58	Вход/выход "порт 3, 7 разряд" Вход/выход "адрес-данные, 7 разряд"	I/O I/O/Z	 AD7
P4.0	F11	52	Вход/выход "порт 4, 0 разряд" Вход/выход "адрес-данные, 8 разряд"	I/O I/O/Z	 AD8
P4.1	G10	51	Вход/выход "порт 4, 1 разряд" Вход/выход "адрес-данные, 9 разряд"	I/O I/O/Z	 AD9
P4.2	G11	50	Вход/выход "порт 4, 2 разряд" Вход/выход "адрес-данные, 10 разряд"	I/O I/O/Z	 AD10
P4.3	H10	49	Вход/выход "порт 4, 3 разряд" Вход/выход "адрес-данные, 11 разряд"	I/O I/O/Z	 AD11
P4.4	H11	48	Вход/выход "порт 4, 4 разряд" Вход/выход "адрес-данные, 12 разряд"	I/O I/O/Z	 AD12
P4.5	J10	47	Вход/выход "порт 4, 5 разряд" Вход/выход "адрес-данные, 13 разряд"	I/O I/O/Z	 AD13
P4.6	J11	46	Вход/выход "порт 4, 6 разряд" Вход/выход "адрес-данные, 14 разряд"	I/O I/O/Z	 AD14
P4.7	K10	45	Вход/выход "порт 4, 7 разряд" Вход/выход "адрес-данные, 15 разряд"	I/O I/O/Z	 AD15
HSI.0	J1	20	Вход "Быстрый ввод, канал 0"	I	
HSI.1	J2	21	Вход "Быстрый ввод, канал 1"	I	
HSI.2	K1	22	Вход "Быстрый ввод, канал 2" Выход "Быстрый вывод, канал 4"	I O	 HSO.4
HSI.3	L2	24	Вход "Быстрый ввод, канал 3" Выход "Быстрый вывод, канал 5"	I O	 HSO.5
HSO.0	K2	25	Выход "Быстрый вывод, канал 0"	O	
HSO.1	L3	26	Выход "Быстрый вывод, канал 1"	O	
HSO.2	K5	31	Выход "Быстрый вывод, канал 2"	O	
HSO.3	L6	34	Выход "Быстрый вывод, канал 3"	O	
EA#	B6	79	Вход "Внешний доступ"	I	
BW	B9	71	Вход "Разрядность внешней шины"	I	

Окончание таблицы 1.3.3

1	2	3	4	5	6
READY	L10	43	Вход "Готовность"	I	
NMI	A5	80	Вход "Немаскируемое прерывание"	I	
RESET#	E1	9	Вход/выход "Сброс"	I/O	
INST	A9	70	Выход "Чтение команды"	O	
ALE	B10	69	Выход "Разрешение записи адреса" Выход "Адрес действителен"	O O	 ADV#
WR#	K8	40	Выход "Запись" Выход "Запись младшего байта"	O O	 WRL#
BHE#	L9	41	Выход "Разрешение записи старшего байта" Выход "Запись старшего байта"	O O	 WRH#
RD#	A10	68	Выход "Чтение"	O	
CLCOUT	A8	72	Выход "Системный тактовый сигнал"	O	
XTAL1	A7	74	Вывод подключения кварцевого резонатора. Вход тактового сигнала	– I	
XTAL2	B8	73	Вывод подключения кварцевого резонатора	–	
U_{PP}	L7	37	Вход "Режим пониженного потребления"	I	
CDE	D1	6	Не используется, подключается к общему выводу	–	
U_{CC1} (U_{CC})	A6	8, 33, 60, 77, 78	Питание 5 В цифровой части микросхемы	–	
U_{CC2} (U_{REF})	C2	5	Питание 5 В аналоговой части микросхемы	–	
U_{SS} (GND)	B7, K6	12, 32, 35, 36, 53, 55, 57, 64, 75, 76	Общий вывод цифровой части микросхемы	–	
U_{SS} (ANGND)	C1	3, 4	Общий вывод аналоговой части микросхемы	–	
NC		13, 14, 23, 54, 81, 88	Незадействованные выводы	–	

Примечание – Принятые условные обозначения в графе «Тип вывода»:
- I – вход,
- O – выход,
- I/O – вход/выход,
- Z – третье состояние.

2 Описание устройства и принцип работы

2.1 Блок-схема микроконтроллеров

Микросхемы представляют собой СБИС высокопроизводительных микроконтроллеров с 16-разрядным CPU и ОЗУ на кристалле 232 байт. Регистровая архитектура не нуждается в аккумуляторе, и большинство операций может быть выполнено с регистрами. С помощью регистровых операций можно управлять периферийными блоками, расположенными на кристалле. В состав периферийных блоков входят: последовательный порт, АЦП, ШИМ, высокоскоростная подсистема ввода-вывода, содержащая два 16-битных таймера, 8-уровневое FIFO выборки и 8-канальный программируемый выходной генератор, сторожевой таймер.

Блок-схема микроконтроллеров представлена на рисунке 2.1.1.

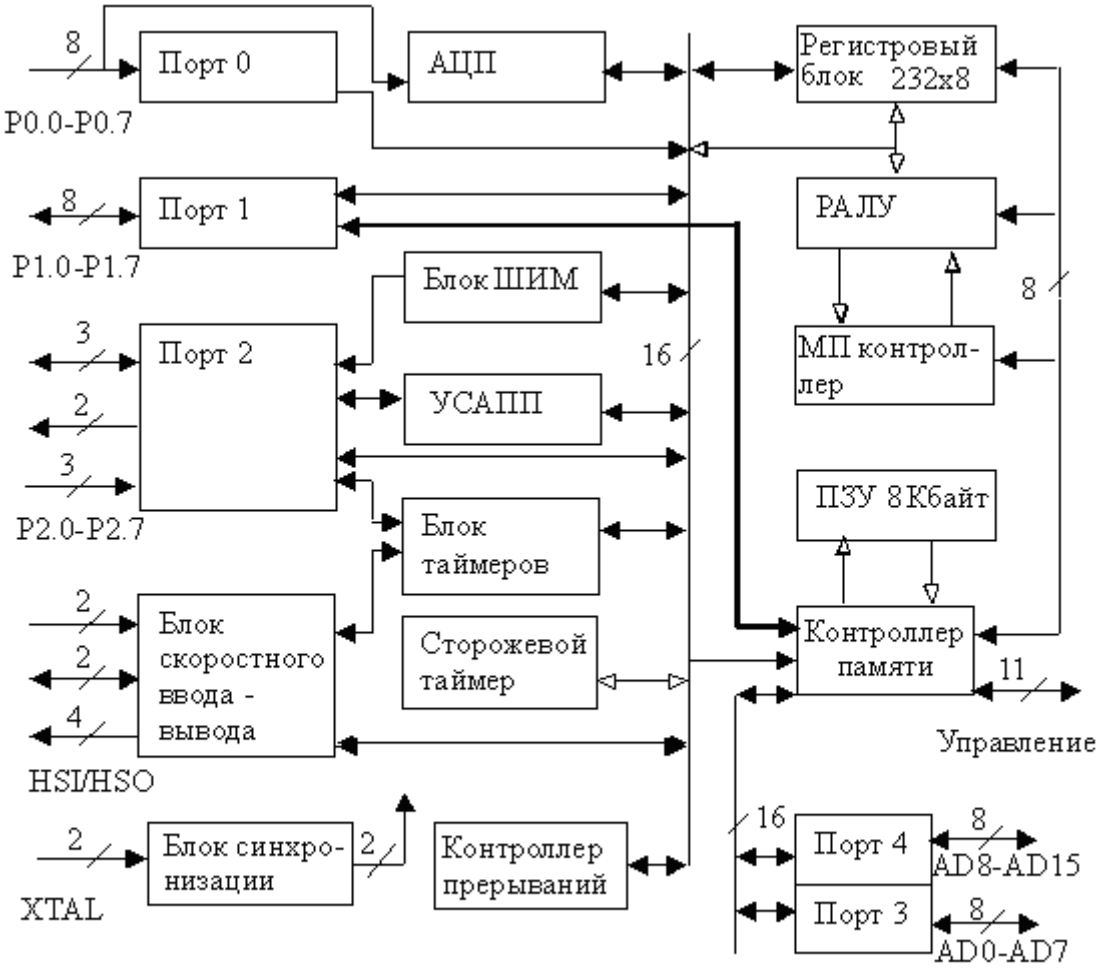


Рисунок 2.1.1 – Блок-схема микроконтроллеров

2.2 Центральный процессор (CPU)

Главными компонентами CPU являются регистровый файл и регистровое арифметическое логическое устройство (РАЛУ). Коммуникации с внешними схемами осуществляются посредством регистров специального назначения (РСН) или через контроллер памяти. РАЛУ не использует аккумулятор. Вместо этого оно оперирует напрямую с 256-байтовым регистровым пространством, состоящим из регистрового файла и регистров специального назначения (РСН). Эффективные операции ввода-вывода возможны благодаря непосредственному управлению вводом-выводом через РСН.

Основные преимущества этой структуры:

- возможность быстрых контекстных переключений;
- отсутствие "узкого места" - аккумулятора;
- быстрые операции ввода-вывода.

16-разрядное CPU соединено с контроллером прерываний и контроллером памяти посредством 16-разрядной шины. Дополнительно имеется 8-разрядная шина, которая передает байты команды из контроллера памяти в CPU, кроме того 16-разрядная шина соединяет CPU с периферийными блоками.

2.3 Контроллер памяти

РАЛУ обменивается данными с внешней памятью через контроллер памяти; в это адресное пространство не входят регистровый файл и регистры специального назначения (РСН). Контроллер памяти содержит контроллер шины, устройство четырехбайтовой очереди команд и подчиненный программный счетчик (Slave PC). Обе внутренние шины ПЗУ и внешней памяти управляются контроллером шины. Запрос доступа к памяти в контроллер шины может придти от РАЛУ или от очереди команд, причем очередь имеет высший приоритет. Запросы из очереди инструкций производятся по адресу из подчиненного программного счетчика. Программный поток (поток команд, программа) считывается из памяти по адресу подчиненного программного счетчика; процессор экономит время, так как адреса редко посылаются в контроллер памяти (перезаписываются). Если адресная последовательность изменяется в результате JUMP, прерываний, CALL (вызова) или возврата из подпрограммы, подчиненный программный счетчик загружается новым значением, очередь заполняется и процесс продолжается. Скорость работы возрастает при использовании очереди, так как обычно сохраняет следующий байт команды доступным. Временные диаграммы выполнения команд приведены без добавления времени ожидания и при выбранной 16-разрядной шине. Перезагрузка подчиненного программного счетчика и запись первого байта нового потока инструкций занимает 4 такта CLKOUT. Это происходит при JUMP (командах перехода).

Когда для отладки кодов используется логический анализатор, нужно представлять, что не возможно определить, когда команда начинает работать после загрузки, так как очередь заполняется до начала исполнения команды.

2.4 Управление CPU

Управление CPU блоком последовательности микрокодов допускает исполнение операций с байтом, словом или двойным словом в 256-битовым регистровом пространстве. Команды для CPU поступают из очереди и временно запоминаются в регистре команд. Блок последовательности микрокодов декодирует команды и генерирует необходимую последовательность событий (сигналов, микрокодов), необходимые для исполнения требуемой функции. На рисунке 2.4.1 приведена блок-схема контроллера памяти, РАЛУ, регистр команд и блок генератора микрокода.

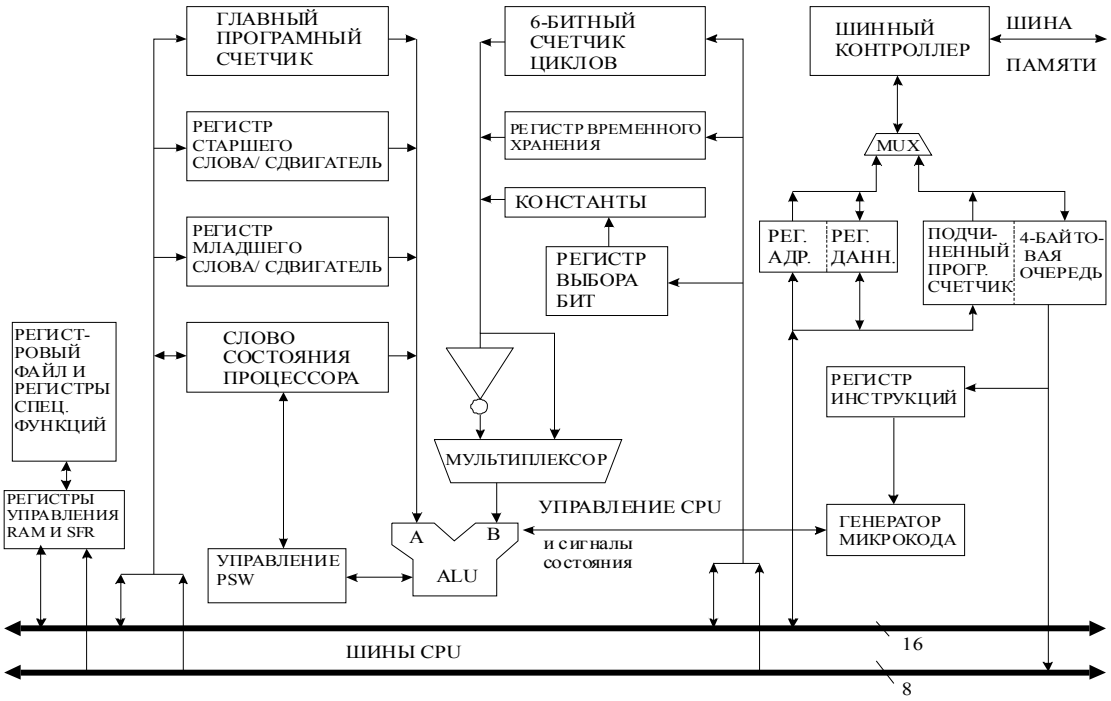


Рисунок 2.4.1 – Блок-схема РАЛУ

2.5 Регистровое АЛУ (РАЛУ)

Главным образом вычисления в микросхемах происходят в РАЛУ. РАЛУ, показанное на рисунке 2.4.1, содержит 17-битное РАЛУ, регистр слова состояния процессора (PSW), программный счетчик, счетчик циклов и три регистра временного хранения. Все регистры 16-разрядные или 17-разрядные (16-разрядов + расширение знаком). Некоторые из регистров способны выполнять простые операции для разгрузки РАЛУ. Отдельный инкремент используется для PC, в то время, когда ему доступны операнды. Однако изменения содержимого PC при JUMP, вызовах, возвратах и прерываниях происходят через РАЛУ. Два временных регистра имеют свою собственную логику сдвигов. Эти регистры используются для операций, требующих логических сдвигов, включая нормализацию, умножение и деление. Регистр "младшего слова" и регистр "старшего слова" используются вместе для 32-битных операций и как временные регистры для большинства команд. Повторные сдвиги управляются 6-битным "счетчиком циклов". Третий временной регистр запоминает второй операнд двухоперандных команд. Он содержит множитель при умножении и делитель при делении. При выполнении вычитания выходные данные этого регистра дополняются до двух перед тем как данные попадают на вход "B" РАЛУ. Несколько констант, такие как 0, 1 и 2, записаны в РАЛУ для ускорения вычислений (создание дополнения числа до двух и выполнение инкремента или декремента). Дополнительно однобитовые маски для команд проверки битов генерируются в регистре констант, основанном на трехбитном регистре выбора бита.

2.6 Внутренние временные соотношения

Микросхемам для функционирования необходим тактовый сигнал на входе XTAL1. XTAL1 и XTAL2, являясь входом и выходом инвертора на кристалле, могут быть использованы для генерации тактового сигнала.

Внутренние операции ИС синхронизируются частотой от внутреннего или внешнего генератора, деленной на 2. Каждые два периода частоты составляют одно "State time", основное время измерения для всех операций ИС при тактовой частоте 20 МГц "State time" (машинный цикл) составляет 100 нс.

Так как микросхемы могут работать на разных частотах, времена даются в "State time", если по-другому не определено. Выходной системный тактовый сигнал, показанный на рисунке 2.6.1, формируется для прослеживания внутреннего машинного цикла при $f_C = 20$ МГц, $t_{CO} = 100$ нс.

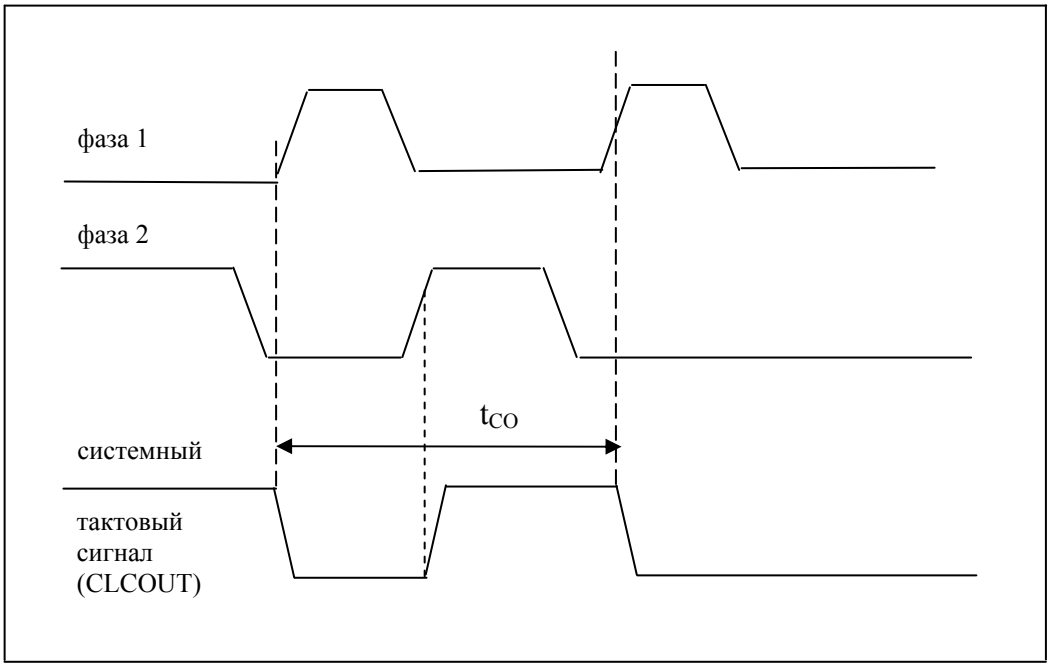


Рисунок 2.6.1 – Внутренние временные соотношения

2.7 Адресное пространство памяти

Адресное пространство микросхем составляет 64 Кбайт, большая часть которого доступна как память программ или память данных. Адреса с 0000H до 00FFH и с 1FFEH до 2080H предназначены для специальных целей. Все другие адреса могут быть использованы для запоминания любых программ или данных или карты памяти периферии. Карта памяти показана на рисунке 2.7.1.

Внешняя память или устройства ввода-вывода	0FFFFH
Внутреннее ПЗУ или внешняя память	4000H
Резерв	2080H
Старшие 8 векторов прерываний	2040H
Ключ (шифр) доступа ПЗУ	2030H
Резерв	2020H
Байт конфигурации кристалла	2019H
Резерв	2018H
8 младших векторов прерываний + 2 специальных прерывания	2014H
Порты 3 и 4	2000H
Внешняя память или внешние устройства ввода-вывода	1FFEH
Память внутренних данных (регистровый файл, регистры специального назначения, указатель стека)	0100H 0000H

Рисунок 2.7.1 – Карта памяти ИС

2.7.1 Регистровый файл

Адреса с 0000H по 00FFH содержат файл регистров и регистры специального назначения. РАЛУ может оперировать с любым из этих 256 внутренних регистров, однако коды (команд) не могут выполняться из этих регистров. Если сделана попытка запустить инструкции из адресов с 000H по 0FFH, инструкции будут запрашиваться из внешней памяти. Эта секция внешней памяти зарезервирована.

Внутреннее ОЗУ по адресам с 18H до 0FFH содержат регистровый файл. Он содержит 232 байта ОЗУ с возможностью доступа по байтам (8 бит), словам (16 бит) или двойным словам (32 бита). Каждый из этих регистров может быть использован РАЛУ как один из 232 "аккумуляторов". Эта область памяти, как источник данного состояния микросхем, сохраняется нетронутой, пока микросхемы находятся в режиме сохранения мощности (Power down mode).

Ячейки с адресами 18H и 19H составляют указатель стека. Эти ячейки не являются регистрами специального назначения и могут быть использованы как стандартные ячейки ОЗУ, если стековые операции не производятся. РАЛУ может с легкостью оперировать ими. Указатель стека должен быть инициализирован пользовательской программой и может указывать на любое место в 64-Кбайтном адресном пространстве. Работа стека происходит с загрузкой вниз, таким образом, указатель стека должен быть инициализирован на два байта выше верхних стековых адресов и должен быть выровнен по адресам слов.

2.7.2 Регистры специального назначения

Регистры с адресами от 00H до 17H являются регистрами управления вводом-выводом или, по-иному, регистрами специального назначения. Все периферийные устройства ИС (исключая порты 3 и 4) управляются через эти регистры. Как показано на рисунке 2.7.2.1, регистры специального назначения (РСН) микросхем расположены в трех окнах.

Функции РСН представлены в таблице 2.7.2.2, альтернативные функции РСН в окне 15 – в таблице 2.7.2.3.

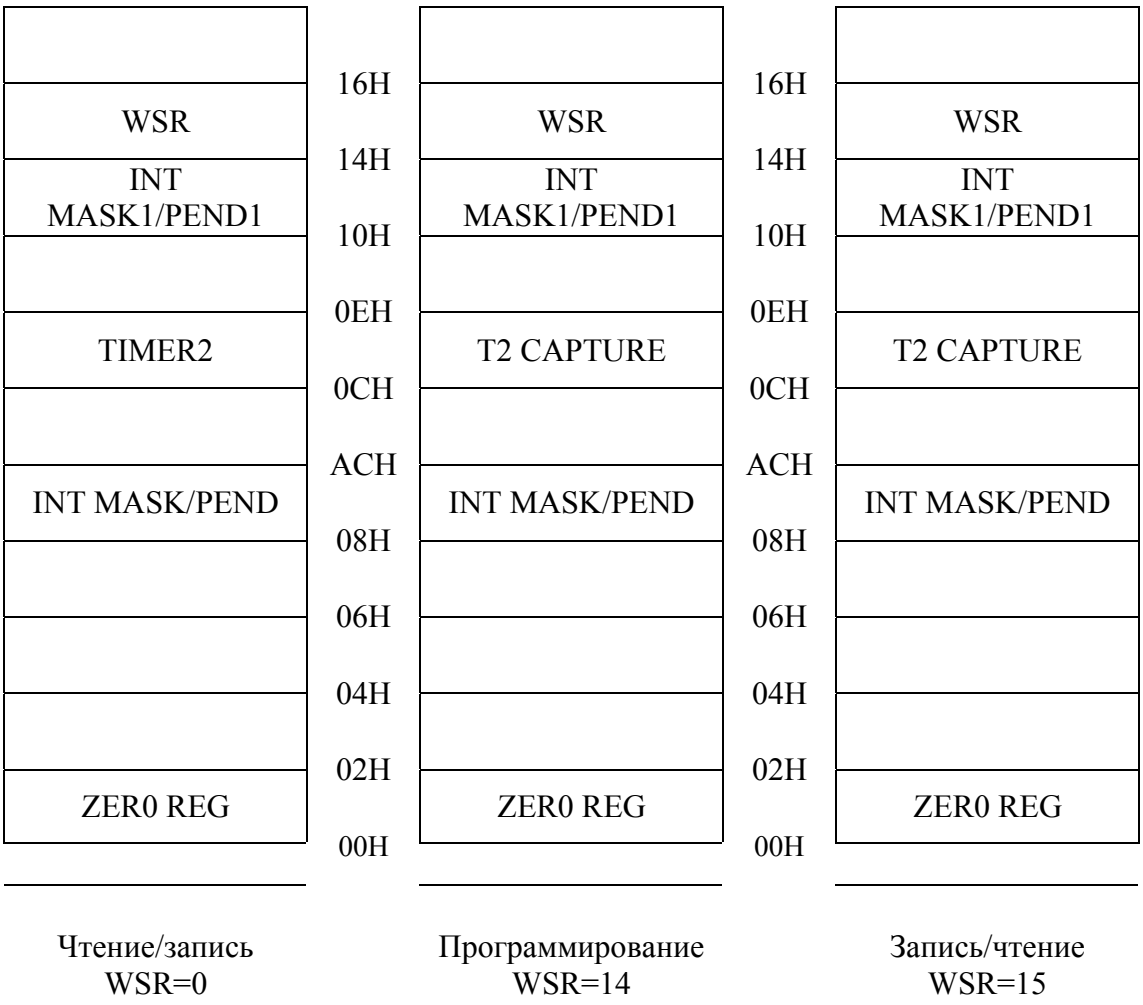


Рисунок 2.7.2.1 - Регистровые окна

Переключения между окнами осуществляются Регистром Выбора Окна (WSR) с адресом 14H для всех окон. Команды PUSHA и POPA загружают и выгружают WSR, таким образом легко сделать выбор между окнами (выбрать одно из трех окон). Только три значения можно записывать в WSR - 0, 14 и 15. Другие значения зарезервированы в будущих реализациях и могут вызывать непредсказуемые действия.

Регистровое окно 0 выбирается при WSR = 0. Как указано в таблице 2.7.2.2, оно (окно) имеет 24 регистра, которые имеют функции, отличающиеся при записи и чтении, т. е. один и тот же регистр может иметь разные функции при записи и чтении.

Управление программированием и тестовые операции - в регистровом окне 14. Регистры, которые не отмечены в этом окне должны быть закрыты от любого чтения или записи. В регистрах окна 15 (WSR = 15) работа PCH изменена. Регистры, которые читаются только в окне 0 записываются только в регистровом окне 15 и наоборот. Причем адреса не меняются. Можно менять окна, не меняя адреса и производить чтение или запись. Как главное исключение - это регистр Таймер 2, он читается и записывается в регистровом окне 0, а в окне 15 по тому же адресу находится регистр захвата Таймера 2, он читается и записывается. Регистры, которые могут читаться и записываться в окне 0 также читаются и записываются в регистровом окне 15, см. таблицу 2.7.2.3.

Таблица 2.7.2.1 – Регистры специального назначения

Указатель стека (SP)	19H	Указатель стека (SP)
IOS2	18H	PWM_CONTROL
IOS1	17H	IOC1
IOS0	16H	IOC0
WSR	15H	WSR
INT_MASK1	14H	INT_MASK1
INT_PEND1	13H	INT_PEND1
SP_STAT	12H	SP_CON
PORT2	11H	PORT2
PORT1	10H	PORT1
PORT0	0FH	BAUD RATE
TIMER2 (HI)	0EH	TIMER2 (HI)
TIMER2 (LO)	0DH	TIMER2 (LO)
TIMER1 (HI)	0CH	IOC2
TIMER1 (LO)	0BH	WATCHDOG
INT_PEND	0AH	INT_PEND
INT_MASK	09H	INT_MASK
SBUF(RX)	08H	SBUF (TX)
HSI_STATUS	07H	HSO_COMMAND
HSI_TIME (HI)	06H	HSO_TIME (HI)
HSI_TIME (LO)	05H	HSO_TIME (LO)
AD_RESULT (HI)	04H	HSI_MODE
AD_RESULT (LO)	03H	AD_COMMAND
ZER0 REG (HI)	02H	ZER0 REG (HI)
ZER0 REG (LO)	01H	ZER0 REG (LO)
	00H	
чтение	WSR=0	запись

РЕЗЕРВ*	10H	
РЕЗЕРВ*	0FH	
РЕЗЕРВ*	0EH	
T2CAPTURE (HI)	0DH	
T2CAPTURE (LO)	0CH	

WSR = 15

Другие SFR в WSR = 15 могут быть считаны, если они были записаны в WSR = 0 и могут быть записаны, если в WSR = 0 они считываются

PPW	04H

WSR = 14

Резервные регистры не записываются и не читаются

Таблица 2.7.2.2 – Функции регистров специального назначения

Регистр	Описание
ZER0 REG	Нулевой регистр всегда считывается как ноль, используется как базовый для индексной адресации и как константа для вычислений и сравнений.
AD_RESULT	АЦП результат (LO/HI) – младший или старший разряды результата преобразования.
AD_COMMAND	АЦП командный регистр – управляет АЦП.
HSI_MODE	Регистр режима HSI – устанавливает режим высокоскоростного ввода.
HSI_TIME	Время HSI старший/младший – содержит время, в течение которого высокоскоростной ввод был включен.
HSO_TIME	Время HSO старший/младший – устанавливает время или счет циклов для высокоскоростного вывода при исполнении команды из командного регистра.
HSO_COMMAND	Командный регистр HSO – определяет, что будет происходить во время, указанное в регистре HSO_TIME.
HSI_STATUS	Регистр состояния HSI – указывает, какой из выводов HSI был переведен в режим HSI, текущее состояние выводов. В окне 15 записываются проверяемые разряды, но не их текущее состояние.
SBUF (TX)	Буфер передачи для последовательного порта – содержит данные для вывода. Последний записанный результат может быть считан в окне 15.
SBUF (RX)	Буфер приема для последовательного порта – сохраняет байт, принятый в последовательный порт. Может быть записан в окне 15.
INT_MASK	Регистр маскирования прерываний – разрешает или запрещает индивидуальные прерывания.
INT_PEND	Регистр ожидания прерываний – указывает, какой сигнал прерывания пришел от одного из источников и не был обслужен (только INT_PENDING).
WATCHDOG	Регистр сторожевого таймера – вызывает автоматический периодический сброс каждые 64К машинных цикла. Возобновляет старший байт счетчика WDT в окне 15.
TIMER1	Таймер 1 старший/младший – старшие/младшие байты таймера 1.
TIMER2	Таймер 2 старший/младший – старшие/младшие байты таймера 2.
IOPORT0	Регистр порта 0 – уровни на выводах порта 0. Резервируются в окне 15.
BAUD_RATE	Регистр, определяющий скорость передачи, загружается периодически. Резервируется в окне 15.
IOPORT1	Регистр порта 1 – используется для чтения/записи в порт 1. Зарезервирован в окне 15.
IOPORT2	Регистр порта 2 – используется для чтения/записи в порт 2. Зарезервирован в окне 15.
SP_STAT	Состояние последовательного порта – указывает состояние последовательного порта.
SP_CON	Управление последовательного порта – используется для установки режима работы последовательного порта.
IOS0	Регистр 0 состояния ввода-вывода – содержит информацию о состоянии HSO. Записывается на выход HSO в окне 15.
IOS1	Регистр 1 состояния ввода-вывода – содержит информацию о состоянии таймеров и HSI.
IOC0	Регистр 0 управления ввода-вывода – управляет альтернативными функциями выводов HSI, источником сброса и синхронизации таймера 2.
IOC1	Регистр 1 управления ввода-вывода – управляет альтернативными функциями порта 2, прерываниями от таймера и прерываниями от HSI.
PWM_CONTROL	Регистр управления ШИМ – управляет длительностью импульсов ШИМ.
INT_PEND1	Регистр ожидания прерываний для 8 векторов прерываний (только INT_PENDING1).
INT_MASK1	Регистр маски прерываний для 8 векторов прерываний.
IOC2	Регистр 2 управления ввода-вывода – управляет функциями таймера 2, АЦП и HSO.
IOS2	Регистр 2 состояния ввода-вывода – содержит информацию о событиях HSO.
WSR	Регистр выбора окна – выбирает регистровое окно.

Таблица 2.7.2.3 – Альтернативные функции PCH в окне 15

Функция	Описание
AD_COMMAND (02H)	Считывание последней записанной команды
AD_RESULT (02H, 03H)	Запись результата в регистр результата АЦП
HSI_MODE (03H)	Чтение значения в HSI_MODE
HSI_TIME (04H, 05H)	Запись в регистр хранения FIFO
HSO_TIME (04H, 05H)	Чтение последнего значения в регистре хранения
HSI_STATUS (06H)	Запись типа события на выводах HSI (разряды HSI_STATUS 0, 2, 4, 6)
HSO_COMMAND (06H)	Чтение последнего значения в регистре хранения
SBUF(RX) (07H)	Запись значения в буфер приема
SBUF(TX) (07H)	Чтение последнего значения из буфера передачи
WATCHDOG (0AH)	Чтение значения старшего байта WDT
TIMER1 (0AH, 0BH)	Запись значения в Timer1
TIMER2 (0CH, 0DH)	Чтение/запись регистра захвата Timer2 (чтение/запись Timer2 при WSR=0)
IOC2* (0BH)	Последняя записанная величина, доступная чтению, исключается бит 7
BAUD_RATE (0EH)	Не может быть считан
PORT0 (0EH)	Нет доступа
SP_STAT (11H)	Установка битов состояния, могут быть установлены биты TI и RI, но они не могут вызывать прерывания
SP_CON (11H)	Чтение текущего байта управления
IOS0 (15H)	Запись в регистр управления выводами HSO. Разряды 6 и 7 недоступны для записи
IOC0* (15H)	Последнее записанное значение разрешено для чтения, за исключением бита 1
IOS1 (16H)	Запись в этот регистр будет устанавливать биты состояния, но не вызывает прерываний, биты 6 и 7 доступны для записи
IOC1 (16H)	Последнее записанное значение доступно для чтения
IOS2 (17H)	Запись в этот регистр устанавливает биты состояния, но не вызывает прерываний
PWM_CONTROL (17H)	Чтение значения рабочего цикла записанного в PWM_CONTROL
_____ * IOC2.7 (CAM CLEAR) и IOC0.1 (T2RST) не защелкиваются и будут читаться как единица (CAM – ассоциативная память модуля HSO).	

Внутри пространства РСН несколько регистров и битов адресов помечены "Зарезервировано". В эти ячейки нельзя производить чтение и запись. В зарезервированные биты всегда записан 0 для совместимости с будущими моделями ИС. Значения, читаемые из этих ячеек могут изменяться от микросхемы к микросхеме или от температуры и напряжения. С регистрами и битами, которые не помечены, нужно обращаться как с резервными регистрами и битами. Заметим, что состояние (по умолчанию) внутреннего регистра – 0, тогда как для внешней памяти – это единичный уровень. Это обусловлено тем, что РСН функции обычно запрещены нулевым значением, тогда как стирание памяти – это перевод в единичное состояние. Необходимо соблюдать предосторожность при использовании РСН, как источника для операций или в качестве базовых или индексных режимов для косвенной или индексной адресации. Возможен нежелательный результат, так как внешние события могут изменить РСН или РСН очистится при чтении. Возможность для РСН по изменению содержимого должна быть учтена при работе с этими регистрами. Это особенно важно, когда используются языки высокого уровня, так как они не всегда учитывают все особенности работы РСН регистров. РСН могут работать с байтами или словами, если иначе не определено.

2.7.3 Зарезервированные области памяти

Адреса 1FFEH и 1FFFH используются для порта 3 и 4, соответственно, позволяя легко перенастраивать эти порты при работе с внешней памятью. Если порты 3 и 4 не надо перенастраивать и внутренняя память не используется, с этими ячейками можно работать как с любыми ячейками внешней памяти. Многие резервные и специальные регистры имеют адреса между 2000H и 2080H. В этой области размещены 18 векторов прерывания, байт конфигурации кристалла и ключ секретности доступа. На рисунке 2.7.3.1 показаны адреса и функции этих регистров. Регистры прерываний, конфигурации кристалла, ключа секретности доступа описаны в разделах 5, 16. С одним исключением, все не определенные по функциям адреса в промежутке с 2000H до 207FH, включая "резервные", зарезервированы для использования при тестировании и для будущих разработок. Они должны быть заполнены шестнадцатеричным значением FFH для обеспечения совместимости с будущими устройствами. Адрес 2019H может содержать 20H для предотвращения возможных конфликтов шины в течение цикла записи ССВ (байт конфигурации кристалла).

Примечание – Эти ситуации возможны только для систем с 16-битной шиной и внешней программной памятью.

"Сброс" ИС вызывает чтение потока команд с адреса 2080H. Этот адрес был выбран для обеспечения возможности системы иметь ОЗУ до 8 Кбайт, являющейся продолжением регистрового файла.

	FFFFH
Внутренняя память или устройства ввода/вывода	4000H
Внутреннее ПЗУ или внешняя память	2080H
Резерв	2074H-207FH
Уровни напряжения	2072H-2073H
Слово сигнатуры	2070H-2071H
Резерв	2040H-206FH
Векторы прерываний	2030H-203FH
Ключ секретности	2020H-202FH
Резерв	2019H-201FH
Байт конфигурации кристалла	2018H
Резерв	2015H-2017H
PPW (регистр длительности импульса программирования, для ИС с EPROM)	2014H
Векторы прерываний	2000H-2013H

Рисунок 2.7.3.1 – Резервное пространство памяти

2.7.4 Внутреннее ПЗУ

При использовании внутреннего ПЗУ программист может определить программу пользователя в области внутренней программной памяти с адресами от 2080H до 3FFFH, а также векторы прерываний, регистр конфигурации кристалла (CCR), ключ секретности в диапазоне адресов от 2000H до 207FH. По адресу 2014H содержится регистр программирования Ширины Импульса (PPW). Этот регистр используется в ИС с СППЗУ и зарезервирован в ИС с масочным ПЗУ.

Инструкции (команды) и данные считываются из внутреннего ПЗУ, если вход EA# установлен в высокий уровень и адрес находится между 2000H и 3FFFH. Во всех других случаях данные считываются из внутреннего ОЗУ данных или внешней памяти и инструкции считываются из внешней памяти. Состояние вывода EA# захватывается по фронту нарастания RESET#. ИС имеют биты защиты ПЗУ, позволяющие программе блокировать чтение внутренней программной памяти. В порядке установления секретности команды нельзя исполнять из последних адресов внутреннего ПЗУ, если биты защиты установлены.

2.7.5 Системная шина

Имеется несколько режимов работы ИС с системной шиной: стандартный режим – 16-битная мультиплексированная шина адрес/данные, 8-битовый режим и режим, когда шина может динамически переключаться между 8-битной и 16-битной. Сигналы HOLD#/HLDA# и READY позволяют создавать разнообразные системы памяти. Сигнал READY необходим для расширения импульсов RD# и WR#, чтобы осуществить доступ к медленной памяти.

3 Программные средства

Система команд ИС содержит полный набор арифметических и логических действий над 8-битными операндами типа BYTE и SHORT-INTEGER и над 16-битными операндами типа WORD и INTEGER. Типы данных DOUBLE WORD и LONG-INTEGER поддерживаются для умножения 16×16, для деления 32 на 16, для операций сдвига и 32-разрядного сравнения. Недостающие операции над 32-битными операндами могут быть выполнены как комбинации над 16-битными операндами.

В дополнение к действиям над различными типами данных поддерживается преобразование между этими типами.

Инструкции микроконтроллеров для сложения, вычитания и сравнения не отличаются для беззнаковых слов и знаковых целых. Условные переходы обеспечивают обработку результатов этих операций как знаковых, так и без знаковых величин.

В данном разделе короткие фрагменты программ будут использоваться для иллюстрации работы. В этих примерах будут использоваться символические имена регистров временного хранения:

AX, BX, CX, DX – 16-битные регистры;
AL – младший байт AX;
AH – старший байт AX;
BL – младший байт BX;
CL – младший байт CX;
DL – младший байт DX.

3.1 Типы операндов

Архитектура ИС обеспечивает поддержку различных типов данных, которые удобны для использования в управляющих системах. При рассмотрении этих типов операндов их имена будут заимствованы из языка PL/M-96. Во избежание разночтений имя типа операнда будем обозначать большими буквами: "BYTE" – это беззнаковая 8-битная переменная; "byte" – восемь бит данных любого типа.

БАЙТЫ (BYTES)

BYTES – беззнаковые 8-битные переменные в диапазоне от 0 до 255. Арифметические и родственные им операции могут использовать операнды типа BYTES, но результат должен быть интерпретирован по модулю 256. Логические операции на операндах типа BYTES работают с отдельными битами. Биты в BYTE-операнде маркированы от 0 до 7, причем бит 0 – младший значащий. Ограничений по размещению для операндов типа BYTE нет; они могут располагаться в любом месте адресного пространства.

СЛОВА (WORDS)

WORDS – беззнаковая 16-битная переменная, которая может иметь значения 0...65535. Арифметические и родственные им операции могут быть применены к операндам типа WORD, но результат должен быть интерпретирован по модулю 65536. Логические операции манипулируют с отдельными битами. Биты пронумерованы с 0 до 15, причем 0 бит – младший значащий. Слова должны быть выровнены по границам четных адресов в адресном пространстве. Младший байт должен находиться по четному адресу, а старший байт – по следующему (нечетному) адресу. Адрес слова – это адрес его младшего байта. Операции со словами, имеющими нечетный адрес, не гарантированны к выполнению.

КОРОТКИЕ ЦЕЛЫЕ (SHORT-INTEGERS)

SHORT-INTEGERS – 8-битная знаковая переменная в диапазоне –128...+127. Арифметические операции, вырабатывающие результаты вне этого диапазона, будут устанавливать флаг переполнения в слове состояния программы (PSW). Действительный числовой результат будет тот же, что и для тех же операций с переменными типа BYTE. SHORT-INTEGERS могут быть размещены в любом месте адресного пространства.

ЦЕЛЫЕ (INTEGERS)

INTEGERS – 16-битные знаковые переменные в диапазоне –32768...+32767. Арифметические операции, вырабатывающие результаты вне этого диапазона, будут устанавливать флаг переполнения в PSW. Действительный числовой результат будет тот же, что для тех же операций с переменными типа WORD. Для INTEGERS существуют те же ограничения по выравниванию и те же правила адресации, что и для WORDS.

БИТЫ (BITS)

BITS – однобитные операнды, которые могут принимать булевы значения "истина" и "ложь". В дополнение к обычной поддержке битовых операндов как компонентов BYTE или WORD, ИС обеспечивают непосредственную проверку любого бита во встроенном регистровом файле. Архитектура микросхем требует, чтобы биты были адресованы как компоненты BYTES или WORDS, она не поддерживает прямую адресацию битов, как у 1830ВЕ51/31.

ДВОЙНЫЕ СЛОВА (DOUBLE-WORDS)

DOUBLE-WORDS – это беззнаковые 32-битные переменные, которые могут принимать значения 0...4294967295. Архитектура ИС непосредственно поддерживает такие операнды для операций сдвига, как делимое при делении на 16-битный операнд, как результат умножения двух 16-битных операндов. Для этих операций переменная типа DOUBLE-WORD должна находиться в регистровом файле и должна быть выровнена по адресу, который без остатка делится на 4. Операнд DOUBLE-WORD адресуется адресом своего младшего байта. Операции с DOUBLE-WORD, которые непосредственно не поддерживаются, могут быть выполнены посредством двух операций с операндами типа WORD. Для совместимости с программным обеспечением пользователь должен выполнять требования по адресации операндов DOUBLE-WORDS.

ДЛИННЫЕ ЦЕЛЫЕ (LONG-INTEGERS)

LONG-INTEGERS – 32-битные переменные в диапазоне –2147483648...+2147483647. Архитектура ИС поддерживает такие операнды непосредственно только для операций сдвига, как делимое при делении на 16-битный операнд, как результат умножения 16-битных операндов, при операциях сравнения двойных слов.

LONG-INTEGERS переменные могут быть также нормализованы. Для этих операций переменная типа LONG-INTEGERS должна находиться в регистровом файле и должна быть выровнена по адресу, который без остатка делится на 4. Переменная типа LONG-INTEGER адресуется адресом ее младшего байта.

Операции с LONG-INTEGERS, которые непосредственно не поддерживаются, могут быть выполнены посредством двух операций с операндами типа INTEGER. Для совместимости с программным обеспечением пользователь должен выполнять требования по адресации LONG операндов.

3.2 Адресация операндов

Доступ к операндам в адресном пространстве ИС возможен посредством шести основных режимов адресации. Данный раздел описывает аппаратную поддержку этих режимов адресации.

Шесть базовых режимов адресации, которые будут описаны, называются: прямая регистровая, косвенная, косвенная с автоинкрементированием, непосредственная, короткая индексная, длинная индексная. Несколько других удобных разновидностей адресации могут быть получены путем комбинирования этих базовых режимов адресации со специфическими регистрами, такими как ZERO [0] или указатель стека [SP].

Прямая регистровая адресация

Используется для прямого доступа к регистру из 256-байтного регистрового файла. Регистр выбирается 8-битным полем в инструкции, причем адрес регистра должен соответствовать требованиям по выравниванию в соответствии с типом операнда. В зависимости от инструкции в вычислениях могут участвовать до трех регистров.

Пример:

```
ADD     AX, BX, CX  ;   AX:=BX+CX
MUL     AX, BX      ;   AX:=AX*BX
INCB    CL          ;   CL:=CL+1
```

Косвенная адресация

Используется для доступа к операнду путем размещения его адреса в форме переменной типа WORD в регистровом файле. Записанный адрес должен удовлетворять ограничениям по выравниванию для операнда требуемого типа. Следует отметить, что косвенная адресация может обращаться к операндам в любом месте адресного пространства ИС, включая регистровый файл. Регистр, содержащий косвенный адрес, выбирается 8-битным полем инструкции. Инструкция может содержать только одну косвенную ссылку, остальные операнды в инструкции должны быть прямо адресуемыми.

Пример:

```
LD      AX, [AX]      ;     AX:=MEM_WORD (AX)
ADDB    AL, BL, [CX]  ;     AL:=BL+MEM_BYTE (CX)
POP     [AX]          ;     MEM_WORD (AX):=MEM_WORD(SP); SP:=SP+2
```

Косвенная адресация с автоинкрементированием

Этот режим ничем не отличается от режима обычной косвенной адресации, за исключением того, что переменная типа WORD, содержащая адрес, будет инкрементироваться после того, как будет использована в качестве адреса. При адресации операндов типа BYTES или SHORT-INTEGERS, инкрементирование будет происходить на 1; при адресации операндов типа WORDS или INTEGERS инкрементирование будет производиться на 2.

Пример:

```
LD          AX, [BX]+     ;     AX:=MEM_WORD (BX); BX:=BX+2
ADDB        AL, BL, [CX]+ ;     AL:=BL+MEM_BYTE (CX); CX:=CX+1
PUSH        [AX]+         ;     SP:=SP-2
                          ;     MEM_WORD (SP):=MEM_WORD (AX)
                          ;     AX:=AX+2
```

Непосредственная адресация

Этот режим адресации позволяет использовать операнд из поля инструкции. При операциях с операндами типа BYTE или SHORT-INTEGER это поле имеет размер 8 бит; при операциях с операндами типа WORD или INTEGER это поле имеет размер 16 бит. Инструкция может иметь только одну непосредственную ссылку, остальные операнды должны быть прямо адресуемыми.

Пример:

```
ADD         AX, #340      ;     AX:=AX+340
PUSH        #1234H        ;     SP:=SP-2; MEM_WORD (SP)=1234H
DIVB        AX, #10       ;     AL:=AX/10; AH:=AX MOD 10
```

Короткая индексная адресация

При этом режиме адресации 8-битовое поле в инструкции выбирает переменную типа WORD в регистровом файле, которая должна содержать адрес. Второе 8-битовое поле в инструкции воспринимается как константа со знаком и суммируется с переменной типа WORD для формирования адреса требуемого операнда. Поскольку это поле имеет знак, адрес операнда может находиться в пределах -128...+127 байт от адреса, определенного переменной типа WORD. Инструкция может иметь только одну ссылку с укороченным индексированием, остальные операнды должны быть прямоадресуемыми.

Пример:

```
LD          AX, 12[BX]    ;     AX:=MEM_WORD (BX+12)
MULB        AX, BL, 3[CX];      AX:=BL*MEM_BYTE (CX+3)
```

Длинная индексная адресация

Этот режим адресации такой же, как короткая индексная адресация за исключением того, что поле индекса 16-разрядное. Инструкция может иметь только одну ссылку с длинной индексной адресацией, остальные операнды должны быть прямо адресуемыми.
Пример:

```
AND     AX, BX, TABLE[CX]       ; AX:=BX.AND.MEM_WORD(TABLE+CX)
ST      AX, TABLE[BX]           ;  MEM_WORD(TABLE+BX)=:AX
ADDB    AL, BL, LOOKUP[CX]      ;  AL:=BL+MEM_BYTE(LOOKUP+CX)
```

Адресация регистром ZERO

Первые два байта в регистровом файле содержат фиксированное значение 0. В дополнение к использованию в качестве константы при вычислениях и сравнениях, этот регистр может использоваться как переменная типа WORD при длинной индексной адресации. Эта комбинация режима адресации и регистра позволяет прямо адресовать любую ячейку памяти.
Пример:

```
ADD     AX, 1234[0]  ;      AX:=AX+MEM_WORD(1234)
POP     5678[0]      ;      MEM_WORD(5678):=MEM_WORD(SP)
                     ;      SP:=SP+2
```

Адресация указателем стека (SP)

Указатель стека может быть доступен как регистр 18H во внутреннем регистровом файле. В дополнение к обычным операциям SP, он также поддерживает доступ к операндам в стеке. Например, вершина стека может быть доступна через SP как переменная типа WORD при использовании косвенной адресации. Или указатель стека может быть использован при короткой индексной адресации для доступа к данным, находящимся в стеке.
Пример:

```
PUSH    [SP]         ;      DUPLICATE TOP_OF STACK
LD      AX, 2[SP]    ;      AX:=NEXT_TO_TOP
```

3.3 Слово состояния программы

Слово состояния программы (PSW) - это набор логических флагов, сохраняющих информацию состояния работы пользовательских программ. В PSW имеется два байта: байт текущего слова состояния и младший байт масок прерывания. Рисунок 3.3.1 показывает биты состояния PSW. PSW может быть сохранен в системном стеке командой PUSHF. Только биты прерываний в PSW доступны для прямой адресации.

7	6	5	4	3	2	1	0
Z	N	V	VT	C	X	I	ST

Рисунок 3.3.1 - PSW регистр

Z: Флаг Z (ZERO) индицирует, что результатом операции является ноль. Для инструкций ADDC и SUBC флаг Z очищается, если результат не ноль, но никогда не устанавливается. Эти две инструкции обычно используются вместе с ADD и SUB для исполнения арифметических операций с повышенной точностью. Такая работа флага Z с инструкциями позволяет ему правильно индицировать результат для всех вычислений повышенной точности.

N: Флаг N (Negative) устанавливается для индикации отрицательного результата операции. Следует отметить, что флаг N устанавливается в алгебраическом корректном состоянии даже и при переполнении. Для сдвиговых операций, включая операции нормализации и всех трех видов сдвигов командами SHL, SHR, SHRA, N-флаг принимает значение старшего бита результата. Это верно и при величине сдвига равной нулю.

V: Флаг V (oVerflow) устанавливается при операциях с результатом, выходящим за диапазон данных (операнда) назначения. Для команд SHL, SHLB и SHLL флаг V установится, если старший бит операнда изменяется в любое время во время сдвига. Для операции деления следующие условия используются для установки флага V:

Для операций:		V установлено, если частное
UNSIGNED BYTE DIVIDE	>	255 (0FFH)
UNSIGNED WORD DIVIDE	>	65535 (0FFFFH)
SIGNED BYTE DIVIDE	<	- 127 (81H) или > 127 (7FH)
SIGNED WORD DIVIDE	<	- 32767 или > 32767 (7FFFH)

VT: Флаг VT (oVerflow Trap) устанавливается, когда установлен флаг V, но очищается только инструкциями, непосредственно работающими с этим флагом, например, CLRVT или JVT JNVT. Работа с флагом VT позволяет проверить, было ли переполнение в конце последовательности арифметических операций. Это более эффективно, чем проверка флага V после каждой инструкции.

C: Флаг C (Carry) устанавливается в соответствии с арифметическим переносом из старшего бита АЛУ для арифметических операций или в соответствии со значением последнего "выдвинутого" из АЛУ бита в операциях сдвига. Арифметический заем после операции вычитания – это инверсное значение флага C (т.е. если операция сгенерировала заем, флаг C сбрасывается в 0).

X: Зарезервирован. Должен всегда очищаться при записи в PSW для совместимости с будущими версиями ИС.

I: Бит глобального запрета прерываний запрещает все прерывания, когда очищен, исключая команды NMI, TRAP и невыполнимые коды команд.

ST: Флаг ST (STicky bit) устанавливается, когда при сдвиге вправо единица была сдвинута сначала во флаг C, а затем была утеряна ("выдвинута"). Флаг ST не определен после операции умножения. Флаг ST может использоваться вместе с флагом C для управления округлением после сдвига вправо. Рассмотрим умножение 8×8 и затем усечение результата до 12 бит:

```
MULUB     AX,CL,DL   ;   AX:=CL*DL
SHR       AX,#4      ;   Правый сдвиг 4 раза
```

Если флаг C установлен после сдвига, это значит, что биты, выдвинутые вправо, имели значение "больше или равно" по отношению к 1/2 младшего бита конечного результата (LSB). В обратном случае флаг C индицирует, что выдвинутые биты имели значение, меньшее 1/2 LSB. Если флаг ST не использовать, округление производится по значению C флага (результат увеличивают на 1, если C установлен). Флаг ST позволяет получить более точное округление при вычислении.

Правила округления с использованием флага ST

C	ST	Значение выдвинутых битов
0	0	Value =0
0	1	0 < Value <1/2 LSB
1	0	Value = 1/2 LSB
1	1	Value > 1/2 LSB

Неточная оценка погрешности может быть главной причиной ошибок в числовых вычислениях; использование флага ST расширяет возможности программиста.

Младшие 8 бит PSW – индивидуальные маски восьми младших источников прерываний ИС. Эти 8 бит могут быть доступны, как 8 бит байта (INT_MASK – адрес 8H) во внутреннем регистровом файле. Отдельный регистр (INT_MASK1 – адрес 13H) содержит управляющие биты для старших восьми прерываний. Логическая "1" в этих битовых позициях разрешает обслуживание соответствующих прерываний. Бит 9 в PSW запрещает все прерывания. Если этот бит очищен, то все прерывания заблокированы. Заметим, что запросы прерываний фиксируются в регистре INT_PEND, даже если они заблокированы. Исполнение соответствующих обслуживающих программ будут происходить в соответствии с их приоритетами (прерываний) по мере их разрешения. Дальнейшая информация по структуре прерываний ИС – в разделе 5.

3.4 Система команд

Система команд ИС содержит полный набор арифметических и логических действий над 8-битовыми операндами BYTE и SHORT-INTEGER и над 16-битными операндами типа WORD и INTEGER.

Типы данных двойное слово (DOUBLE-WORD) и длинное целое (LONG) (32-битные) поддерживаются для результатов 16×16 умножения и деления 32 на 16, а также для сдвиговых операций и 32-битного сравнения. Остальные операции с 32-битными операндами могут быть выполнены комбинацией 16-битных операций.

Например:
```
ADD     AX, CX
ADDC    BX, DX
```
выполняется 32-битное сложение, а последовательность:
```
SUB     AX, CX
SUB     BX, DX
```
выполняет 32-битное вычитание. Операции с REAL (в операциях с плавающей точкой) не поддерживаются аппаратно, но поддерживаются библиотекой операций с плавающей точкой FPAL-96, имеющей набор команд одинарной точности стандарта IEEE для арифметики с плавающей точкой. Использование этого программного обеспечения существенно облегчено наличием команды NORML (нормализацией 32-битных переменных) и наличием флага ST в PSW. В дополнение к действиям над различными типами данных ИС поддерживают преобразование форматов между ними.

Времена выполнения команд приведены в таблицах 3.5.3. и 3.5.4. Эти времена даны для 16-разрядной шины без задержек. Встроенное ПЗУ работает с 16-битной шиной без задержек. Когда идет работа с 8-битной внешней памятью или добавляются циклы ожидания, CPU также вводит циклы ожидания для предварительной загрузки очереди. Производительность для 8-битной внешней шины сложно точно вычислить, однако можно указать ее в пределах от 10 до 30 % от базовой на выполняемом наборе команд. Лучший способ измерения производительности – прогон реальной программы и измерение с использованием эмулятора или с TIMER1. Времена для команд с косвенной и индексной адресацией приведены для двух областей памяти: РСН/внутреннее ОЗУ (0HFFH) и память, с которой работает контроллер памяти (100H-0FFFFH). Любая инструкция, которая использует любой операнд, который запрашивается посредством контроллера памяти (например, ADD r1, 5000H [0]), длится на 2-3 машинных периода больше, чем при работе с РСН/ОЗУ пространством. Любой доступ к данным во внутреннем ПЗУ происходит через контроллер памяти.

Использование регистров

Архитектура ИС позволяет пользоваться 256-байтным регистровым файлом. Часть из этих регистров используются для управления устройствами ввода/вывода и для других специальных функций выполняемых указателем стека и регистром ZERO. Остальные 230 байт доступны для распределения программистом.

Адресация 32-битных операндов

Эти операнды формируются из двух 16-битных слов в памяти. Младшее значащее слово из двойного слова всегда находится по младшему адресу, даже если данные находятся в стеке (это значит, что старшее значащее слово должно быть загружено в стек первым). Двойное слово имеет адрес своего младшего значащего байта. Следует отметить, что архитектура ИС поддерживает некоторые операции с двойными словами. Для этих операций двойное слово должно находиться во внутреннем регистровом файле и должно иметь адрес, который делится на 4 без остатка.

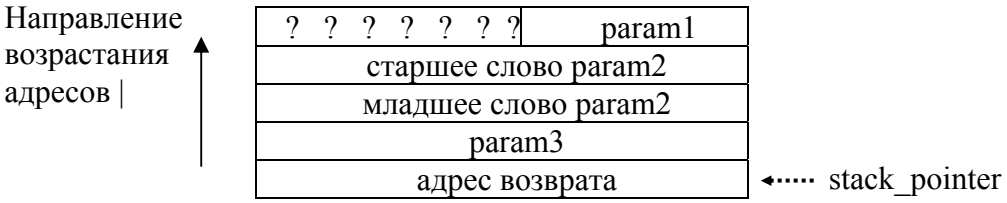
Связь с подпрограммами

Параметры передаются подпрограммам через стек. Параметры записываются в стек в порядке, в котором они перечислены в исходном тексте. Восьмибитные параметры (BYTE или SHORT-INTEGER) заносятся в стек с неопределенным старшим байтом. 32-битные параметры (LONG-INTEGER, DOUBLE-WORD, REAL) заносятся в стек как две 16-битных величины; старшая значащая величина заносится в стек первой.

Рассмотрим, например, следующую процедуру языка PL/M-96:

```
example _ procedure: PROCEDURE
(param1, param2, param3);
DECLARE     param1 BYTE,
            param2 DWORD,
            param3 WORD
```

Когда эта процедура будет вызываться, стек должен содержать параметры в следующем порядке:



Если процедура возвращает значение по команде вызова (в отличие от модифицирования глобальных переменных), тогда результат возвращается в PLMREG. PLMREG затем рассматривается как 8-, 16- или 32-битная переменная в соответствии с типом процедуры.

Стандартный вызов подпрограмм, заимствованный из PL/M-96, должен иметь следующие главные черты.

1 Процедура может в любой момент использовать 8 байт регистрового файла, начиная с PLMREG как ячейки временного хранения данных, находящиеся в теле процедуры.

2 Команда, вызывающая процедуру, должна учитывать, что 8 байт данных в регистровом файле, начиная с PLMREG, модифицируются процедурой.

3 Слово состояния программы не сохраняется и не восстанавливается процедурами, поэтому команда вызова должна учитывать, что флаги состояния (Z, N, V, VT, C, ST) модифицируются процедурой.

4 Результаты выполнения процедуры, всегда возвращаются в PLMREG.

PL/M-96 позволяет описывать процедуры INTERRUPT, которые будут исполняться после обслуживания предыдущего прерывания. Эти процедуры не подчиняются правилам обычных процедур. Параметры не могут передаваться и возвращаться. Поскольку эти процедуры могут выполняться в любой момент времени (отсюда термин прерывание – interrupt), эти процедуры обязаны сохранять состояния PSW и PLMREG при входе в процедуру и восстанавливать их перед выходом из процедуры.

3.5 Защита программного обеспечения

Функциональные возможности ИС помогают предотвратить ошибки программного и аппаратного обеспечения. Защита обеспечивается против исполнения запрещенных кодов (прерывания по запрещенному коду операции). Инструкция RST может вызывать сброс, если программный счетчик выходит за пределы допустимого адресного пространства. Эта (RST) инструкция имеет код 0FFH, поэтому, если процессор читает шинные линии, которые имеют предзаряд высокого уровня, то он самостоятельно производит сброс (RESET). Рекомендуется неиспользованные области памяти инструкций заполнять командами NOP (нет операций) и периодически командами JAMP, чтобы переходить к программам обработки ошибок или к инструкциям RST. Это важная особенность в операциях с просмотровой таблицей, если имеет место нежелаемый результат.

Сторожевой таймер (WDT) – дальнейшая защита от программных и аппаратных ошибок. Когда используется WDT для защиты программного обеспечения (программы) желательно сбрасывать его в одном месте в программе, уменьшая этим возможность нежелательного сброса WDT.

Секция программы, которая сбрасывает WDT, должна проверять другие секции программы на правильность работы. Это можно сделать, проверяя на принадлежность их приемлемым значениям. Простое использование программного таймера для сброса WDT каждые 10 мс должно обеспечивать защиту только для катастрофических ошибок.

В таблице 3.5.1 приведена система команд микроконтроллеров.

В таблице 3.5.2 приведена длина команды (в байтах), через дробь указан код операции.

Таблица 3.5.1 – Система команд микроконтроллеров

Мнемоника	Операнды	Операции (примечание 1)	Флаги (примечание II)						Примечания
			Z	N	C	V	VT	ST	
1	2	3	4	5	6	7	8	9	10
ADD/ADDB	2	D ← D + A	√	√	√	√	↑	-	
ADD/ADDB	3	D ← B + A	√	√	√	√	↑	-	
ADDC/ADDCB	2	D ← D + A + C	↓	√	√	√	↑	-	
SUB/SUBB	2	D ← D - A	√	√	√	√	↑	-	
SUB/SUBB	3	D ← B - A	√	√	√	√	↑	-	
SUBC/SUBCB	2	D ← D - A + C - 1	↓	√	√	√	↑	-	
CMP/CMPB	2	D - A	√	√	√	√	↑	-	
MUL/MULU	2	D,D + 2 ← D x A	-	-	-	-	-	-	2
MUL/MULU	3	D,D + 2 ← B x A	-	-	-	-	-	-	2
MULB/MULUB	2	D,D + 1 ← D x A	-	-	-	-	-	-	3
MULB/MULUB	3	D,D + 1 ← B x A	-	-	-	-	-	-	3
DIVU	2	D←(D,D+2)/A,D+2 ←остаток	-	-	-	√	↑	-	2
DIVUB	2	D←(D,D+1)/A,D+1 ←остаток	-	-	-	√	↑	-	3
DIV	2	D←(D,D+2)/A,D+2 ←остаток	-	-	-	√	↑	-	
DIVB	2	D←(D,D+1)/A,D+1 ←остаток	-	-	-	√	↑	-	
AND/ANDB	2	D ← D AND A	√	√	0	0	-	-	
AND/ANDB	3	D ← B AND A	√	√	0	0	-	-	
OR/ORB	2	D ← D OR A	√	√	0	0	-	-	
XOR/XORB	2	D ← D (ecxl. or) A	√	√	0	0	-	-	
LD/LDB	2	D ← A	-	-	-	-	-	-	
ST/STB	2	A ← D	-	-	-	-	-	-	
LDBSE	2	D ← A; D + 1 ← SIGN(A)	-	-	-	-	-	-	3,4
LDBZE	2	D ← A; D + 1 ← 0	-	-	-	-	-	-	3,4
PUSH	1	SP ← SP - 2; (SP) ← A	-	-	-	-	-	-	
POP	1	A ←(SP); SP + 2	-	-	-	-	-	-	
PUSHF	0	SP ← SP - 2; (SP) ← PSW; PSW ← 0000H; I ← 0	0	0	0	0	0	0	
POPF	0	PSW←(SP); SP←SP+2; I← √	√	√	√	√	√	√	
SJMP	1	PC ← PC + 11-бит поле	-	-	-	-	-	-	5
LJMP	1	PC ← PC + 16-бит поле	-	-	-	-	-	-	5
BR (косвенный)	1	PC ← (A)	-	-	-	-	-	-	
SCALL	1	SP ← SP - 2; (SP)←PC; PC←PC+11-бит поле	-	-	-	-	-	-	5
LCALL	1	SP ← SP - 2;(SP)←PC; PC ← PC + 16-бит поле	-	-	-	-	-	-	5
RET	0	PC ← (SP); SP ← SP + 2	-	-	-	-	-	-	
J (условные)	1	PC←PC + 8-бит поле (если выполняется ветвление)	-	-	-	-	-	-	5
JC	1	переход если C=1	-	-	-	-	-	-	5
JNC	1	переход если C=0	-	-	-	-	-	-	5
JE	1	переход если Z=1	-	-	-	-	-	-	5
JNE	1	переход если Z=0	-	-	-	-	-	-	5
JGE	1	переход если N=0	-	-	-	-	-	-	5
JLT	1	переход если N=1	-	-	-	-	-	-	5
JGT	1	переход если N=0 и Z=0	-	-	-	-	-	-	5
JLE	1	переход если N=1 или Z=1	-	-	-	-	-	-	5

Продолжение таблицы 3.5.1

1	2	3	4	5	6	7	8	9	10
JH	1	переход если C=1 и Z=0	-	-	-	-	-	-	5
JNH	1	переход если C=0 или Z=1	-	-	-	-	-	-	5
JV	1	переход если V=1	-	-	-	-	-	-	5
JNV	1	переход если V=0	-	-	-	-	-	-	5
JVT	1	переход если VT=1;очистка VT	-	-	-	-	0	-	5
JNVT	1	переход если VT=0;очистка VT	-	-	-	-	0	-	5
JST	1	переход если ST=1	-	-	-	-	-	-	5
JNST	1	переход если ST=0	-	-	-	-	-	-	5
JBS	3	переход если определ. бит.= 1	-	-	-	-	-	-	5,6
JBC	3	переход если определ. бит.= 0	-	-	-	-	-	-	5,6
DJNZ/ DJNZW	1	D ← D - 1 если D ≠ 0 то PC←PC+8 бит поле	-	-	-	-	-	-	5 10
DEC/DECB	1	D ← D - 1	√	√	√	√	↑	-	
NEG/NEGB	1	D ← 0 - D	√	√	√	√	↑	-	
INC/INCB	1	D ← D + 1	√	√	√	√	↑	-	
EXT	1	D ← D; D + 2 ← Sign(D)	√	√	0	0	-	-	2
EXTB	1	D ← D; D + 1 ← Sign(D)	√	√	0	0	-	-	3
NOT/NOTB	1	D ← Logical Not (D)	√	√	0	0	-	-	
CLR/CLRB	1	D ← 0	1	0	0	0	-	-	
SHL/SHLB/ SHLL	2	C ← ст.бит....мл.бит ← 0 — —	√	√	√	√	↑	-	7
SHR/SHRB/ SHRL	2	0 → ст.бит....мл.бит → C — —	√	√	√	0	-	√	7
SHRA/SHRAB/ SHRAL	2	ст.бит → ст.б....мл.б → C — —	√	√	√	0	-	√	7
SETC	0	C ← 1	-	-	1	-	-	-	
CLRC	0	C ← 0	-	-	0	-	-	-	
CLRVT	0	VT ← 0	-	-	-	-	0	-	
RST	0	PC ← 2080H	0	0	0	0	0	0	8
DI	0	запрет всех прерываний (I←0)	-	-	-	-	-	-	
EI	0	разрешение (I←1)	-	-	-	-	-	-	
NOP	0	PC ← PC + 1	-	-	-	-	-	-	
SKIP	0	PC ← PC + 2	-	-	-	-	-	-	
NORML	2	Сдвиг влево пока ст. бит не станет =1; D ← счет сдвигов	√	√	0	-	-	-	7
TRAP	0	SP ← SP-2; (SP) ← PC; PC ← (2010H)	-	-	-	-	-	-	9
PUSHA	1	SP ← SP-2; (SP) ← PSW; PSW ← 0000H; SP ← SP-2; (SP)←IMASK1/WSR; IMASK1←00H	0	0	0	0	0	0	
POPA	1	IMASK1/WSR←(SP); SP←SP+2 PSW←(SP); SP←SP + 2	√	√	√	√	√	√	
IDLPD	1	Режим IDLE если ключ = 1; Режим энергосбережения если ключ = 2; (Power down) Иначе сброс;	-	-	-	-	-	-	
CMPL	2	D-A	√	√	√	√	↑	-	
BMOV	2	[PTR_HI]+ ← [PTR_LOW]+; до тех пор, пока счетчик не станет= 0.	-	-	-	-	-	-	

Примечание I к таблице 3.5.1

1 Если мнемоника оканчивается на "B" - операции выполняются с байтом, иначе выполняется операция со словом. Операнды D, B и A должны соответствовать правилам выравнивания для требуемых типов операндов. Операнды D и B располагаются в регистровом файле, A может быть расположен в любом месте памяти.
2 D, D + 2 - последовательность слов в памяти; D - двойное слово.
3 D, D + 1 - последовательность бит в памяти; D - слово.
4 Замена байтов словом.
5 Поле дополнения до двух.
6 Определяет бит как 1 из 2048 в регистровом файле.
7 "L" (длинный) суффикс указывает на операцию двойного слова.
8 Инициирует сброс установкой низкого уровня RESET#. Программа будет инициализировать все необходимые регистры с начального адреса 2080H.
9 Ассемблер не допускает такую мнемонику.
10 DJNZW - инструкция не гарантируемая для исполнения.

Примечание II к таблице 3.5.1 (для каждой команды показано, как она изменяет флаги)

1 Символ "√" означает, что флаг устанавливается или сравнивается в соответствии с результатом операции.
2 Символ "-" означает, что флаг не модифицируется.
3 Символ "1" или "0" означает, что флаг устанавливается или сбрасывается всегда.
4 Символ "↑" означает, что команда может устанавливать флаг (в результате вычислений и в соответствии с назначением флага), но не может сбросить его. Тот же смысл только для сброса флага имеет символ "↓".
5 Символ "?" означает, что флаг переходит в неопределенное (случайное) состояние.

Таблица 3.5.2 - Длина команды (в байтах)/код операции

Мнемоника	Прямая	Непо-средст-венная	Косвенная		Индексная	
			нормальная(1)	A - инкре-мент(1)	корот-кая(1)	длин-ная(1)
ADD(3-OP)	4/44	5/45	4/46	4/46	5/47	6/47
SUB(3-OP)	4/48	5/49	4/4A	4/4A	5/4B	6/4B
ADD(2-OP)	3/64	4/65	3/66	3/66	4/67	5/67
SUB(2-OP)	3/68	4/69	3/6A	3/6A	4/6B	5/6B
ADDC	3/A4	4/A5	3/A6	3/A6	4/A7	5/A7
SUBC	3/A8	4/A9	3/AA	3/AA	4/AB	5/AB
CMP	3/88	4/89	3/AB	3/AB	4/8B	5/8B
ADDB(3-OP)	4/54	4/55	4/56	4/56	5/57	6/57
SUBB(3-OP)	4/58	4/59	4/5A	4/5A	5/5B	6/5B
ADDB(2-OP)	3/74	3/75	3/76	3/76	4/77	5/77
SUBB(2-OP)	3/78	3/79	3/7A	3/7A	4/7B	5/7B
ADDCB	3/B4	3/B5	3/B6	3/B6	4/B7	5/B7
SUBCB	3/B8	3/B9	3/BA	3/BA	4/BB	5/BB
CMPB	3/98	3/99	3/9A	3/9A	4/9B	5/9B
MUL(3-OP)	5/(2)	6/(2)	5/(2)	5/(2)	6/(2)	7/(2)
MULU(3-OP)	4/4C	5/4D	4/4E	4/4E	5/4F	6/4F
MUL(2-OP)	4/(2)	5/(2)	4/(2)	4/(2)	5/(2)	6/(2)
MULU(2-OP)	3/6C	4/6D	3/6E	3/6E	4/6F	5/6F
DIV	4/(2)	5/(2)	4/(2)	4/(2)	5/(2)	6/(2)
DIVU	3/8C	4/8D	3/8E	3/8E	4/8F	5/8F
MULB(3-OP)	5/(2)	5/(2)	5/(2)	5/(2)	6/(2)	7/(2)
MULUB(3-OP)	4/5C	4/5D	4/5E	4/5E	5/5F	6/5F
MULB(2-OP)	4/(2)	4/(2)	4/(2)	4/(2)	5/(2)	6/(2)
MULUB(2-OP)	3/7C	3/7D	3/7E	3/7E	4/7F	5/7F
DIVB	4/(2)	4/(2)	4/(2)	4/(2)	5/(2)	6/(2)
DIVUB	3/9C	3/9D	3/9E	3/9E	4/9F	5/9F
AND(3-OP)	4/40	5/41	4/42	4/42	5/43	6/43
AND(2-OP)	3/60	4/61	3/62	3/62	4/63	5/63
OR(2-OP)	3/80	4/81	3/82	3/82	4/83	5/83
XOR	3/84	4/85	3/86	3/86	4/87	5/87
ANDB(3-OP)	4/50	4/51	4/52	4/52	5/53	5/53
ANDB(2-OP)	3/70	3/71	3/72	3/72	4/73	4/73
ORB(2-OP)	3/90	3/91	3/92	3/92	4/93	5/93
XORB	3/94	3/95	3/96	3/96	4/97	5/97
PUSH	2/C8	3/C9	2/CA	2/CA	3/CB	4/CB
POP	2/CC	-	2/CE	2/CE	3/CF	4/CF
LD	3/A0	4/A1	3/A2	3/A2	4/A3	5/A3
LDB	3/B0	3/B1	3/B2	3/B2	4/B3	5/B3
ST	3/C0	-	3/C2	3/C2	4/C3	5/C3
STB	3/C4	-	3/C6	3/C6	4/C7	5/C7
LDBSE	3/BC	3/BD	3/BE	3/BE	4/BF	5/BF
LBSZE	3/AC	3/AD	3/AE	3/AE	4/AF	5/AF

Продолжение таблицы 3.5.2 - Длина команды (в байтах)/код операции

Мнемоника	Длина/код операции
PUSHF	1/F2
POPF	1/F3
PUSHA	1/F4
POPA	1/F5
TRAP	1/F7
LCALL	3/EF
SCALL	2/28-2F(3)
RET	1/F0
LJMP	3/E7
SJMP	2/20-27(3)
BR[]	2/E3
JNST	1/D0
JST	1/D8
JNH	1/D1
JH	1/D9
JGT	1/D2
JLE	1/DA
JNC	1/B3
JC	1/D8
JNVT	1/D4
JVT	1/DC
JNV	1/D5
JV	1/DD
JGE	1/D6
JLT	1/DE
JNE	1/D7

Мнемоника	Длина/код операции
JE	1/DF
JBC	3/30-37
JBS	3/38-3F
DJNZ	3/EO
DJNZW	3/E1(4)
NORML	3/0F
SHRL	3/0C
SHLL	3/0D
SHRAL	3/0E
SHR	3/08
SHRB	3/18
SHL	3/09
SHLB	3/19
SHRA	3/0A
SHRAB	3/1A
CLRC	1/F8
SETC	1/F9
DI	1/FA
EI	1/FB
CLRVT	1/FC
NOP	1/FD
RST	1/FF
SKIP	2/00
IDLPD	1/F6
BMOV	3/C1

Примечания к таблице 3.5.2.

1 Косвенная и косвенная автоинкрементная адресация имеют те же коды операции, как и короткая и длинная индексная адресация. Если второй байт четный, выполняется косвенная или короткая индексная адресация. Если нечетный, выполняется косвенная автоинкрементная или длинная индексная адресация;

2 Код операции для знакового умножения и деления расширен префиксом "FE".

3 3 младших значащих бита кода операции объединяются с 8 битами в 11-битное поле с дополнением до двух.

4 Инструкция DJNZW не гарантируется для исполнения.

Таблица 3.5.3 - Время исполнения команды в машинных циклах(1)

Мнемоника	Прямая	Непосред-ственная	Косвенная		Индексная	
			нормальная*	А-инкремент*	корот-кая*	длин-ная*
ADD(3-OP)	5	6	7/10	8/11	7/10	8/11
SUB(3-OP)	5	6	7/10	8/11	7/10	8/11
ADD(2-OP)	4	5	6/8	7/9	6/8	7/9
SUB(2-OP)	4	5	6/8	7/9	6/8	7/9
ADDC	4	5	6/8	7/9	6/8	7/9
SUBC	4	5	6/8	7/9	6/8	7/9
CMP	4	5	6/8	7/9	6/8	7/9
ADDB(3-OP)	5	5	7/10	8/11	7/10	8/11
SUBB(3-OP)	5	5	7/10	8/11	7/10	8/11
ADDB(2-OP)	4	4	6/8	7/9	6/8	7/9
SUBB(2-OP)	4	4	6/8	7/9	6/8	7/9
ADDCB	4	4	6/8	7/9	6/8	7/9
SUBCB	4	4	6/8	7/9	6/8	7/9
CMPB	4	4	6/8	7/9	6/8	7/9
MUL(3-OP)	16	17	18/21	19/22	19/22	20/23
MULU(3-OP)	14	15	16/19	17/19	17/20	18/21
MUL(2-OP)	16	17	18/21	19/22	19/22	20/23
MULU(2-OP)	14	15	16/19	17/19	17/20	18/21
DIV	26	27	28/31	29/32	29/32	30/33
DIVU	24	25	26/29	27/30	27/30	28/31
MULB(3-OP)	12	12	14/17	13/15	15/18	16/19
MULUB(3-OP)	10	10	12/15	12/16	12/16	14/17
MULB(2-OP)	12	12	14/17	15/18	15/18	16/19
MULUB(2-OP)	10	10	12/15	13/15	12/16	14/17
DIVB	18	18	20/23	21/24	21/24	22/25
DIVUB	16	16	18/21	19/22	19/22	20/23
AND(3-OP)	5	6	7/10	8/11	7/10	8/11
AND(2-OP)	4	5	6/8	7/9	6/8	7/9
OR(2-OP)	4	5	6/8	7/9	6/8	7/9
XOR	4	5	6/8	7/9	6/8	7/9
ANDB(3-OP)	5	5	7/10	8/11	7/10	8/11
ANDB(2-OP)	4	4	6/8	7/9	6/8	7/9
ORB(2-OP)	4	4	6/8	7/9	6/8	7/9
XORB	4	4	6/8	7/9	6/8	7/9
PUSH (внутр.стек)	6	7	9/12	10/13	10/13	11/14
POP (внутр. стек)	8	-	10/12	11/13	11/13	12/14
PUSH (внеш. стек)	8	9	11/14	12/15	12/15	13/16
POP (внеш. стек)	11	-	13/15	14/16	14/16	15/17
LD,LDB	4,4	5,4	5/8	6/8	6/9	7/10
ST,STB	4,4	-	5/8	6/8	6/9	7/10
LDBSE	4	4	5/8	6/8	6/9	7/10
LBSZE	4	4	5/8	6/8	6/9	7/10
BMOV	внутренняя / внутренняя: 6+8 на слово внешняя / внутренняя: 6+11 на слово внешняя / внешняя: 6+14 на слово					

*Время для операндов: РСН и внутреннее ОЗУ (0-1FFH) / контроллер памяти (200H-0FFFFH).

Примечание - Время выполнения команд через контроллер памяти может быть на один или два цикла больше в зависимости от числа байт в предварительной очереди. Внутренний стек 200H-1FFH и внешний стек 200H-0FFFFH.

Таблица 3.5.4 - Время выполнения команд в машинных циклах

Мнемоника	Время	Мнемоника	Время
PUSHF(внутр. стек)	6	PUSHF(внешн. стек)	8
POPF (внутр. стек)	7	POPF (внешн. стек)	10
PUSHA(внутр. стек)	12	PUSHA(внешн. стек)	18
POPA (внутр. стек)	12	POPA (внешн. стек)	18
TRAP (внутр. стек)	16	TRAP (внешн. стек)	18
LCALL(внутр. стек)	11	LCALL(внешн. стек)	13
SCALL(внутр. стек)	11	SCALL(внешн. стек)	13
RET (внутр. стек)	11	RET (внешн. стек)	14
CMPL	7	DEC/DECB	3
CLR/CLRB	3	EXT/EXTB	4
NOT/NOTB	3	NC/INCB	3
NEG/NEGB	3		
LJMP	7		
SJMP	7		
BR (косвенный)	7		
JNST,JST	7		
JNH,JH	4/8 переход не происходит/переход происходит		
JGT,JLE	4/8 переход не происходит/переход происходит		
JNC,JC	4/8 переход не происходит/переход происходит		
JNVT,JVT	4/8 переход не происходит/переход происходит		
JNV,JV	4/8 переход не происходит/переход происходит		
JGE,JLT	4/8 переход не происходит/переход происходит		
JNE,JE	4/8 переход не происходит/переход происходит		
JBC,JBS	5/9 переход не происходит/переход происходит		
DJNZ	5/9 переход не происходит/переход происходит		
DJNZW(1)	5/9 переход не происходит/переход происходит		
NORML	8+1 на сдвиг (9 для 0 сдвиг)		
SHRL	7+1 на сдвиг (8 для 0 сдвиг)		
SHLL	7+1 на сдвиг (8 для 0 сдвиг)		
SHRAL	7+1 на сдвиг (8 для 0 сдвиг)		
SHR/SHRB	6+1 на сдвиг (7 для 0 сдвиг)		
SHL/SHLB	6+1 на сдвиг (7 для 0 сдвиг)		
SHRA/SHRAB	6+1 на сдвиг (7 для 0 сдвиг)		
CLRC	2		
SETC	2		
DI	2		
EI	2		
CLRVT	2		
NOP	2		
RST	15 (включая запись байта конфигурации)		
SKIP	3		
IDLPD	8/25(правильный ключ/неправильный ключ)		

Примечание - Команда DJNZW не гарантируется для исполнения.

4 Периферийные устройства

Этими устройствами являются пять главных периферийных блоков ИС: блок широтно-импульсной модуляции (PWM), Таймер 1 и Таймер 2, устройство высокоскоростных входов-выходов, последовательный порт и аналого-цифровой преобразователь. За исключением высокоскоростных входов и выходов (HSIO), каждое периферийное устройство используется отдельно.

Четыре отдельные секции, входящие в состав HSIO, совместно образуют гибкую систему ввода-вывода, базирующуюся на таймерах/счетчиках. В HSIO входят 16-битный таймер (Таймер 1), 16-битный реверсивный счетчик (Таймер 2), блок программируемых входов (HSI) и блок программируемых высокоскоростных выходов (HSO). С очень малыми затратами ресурсов CPU устройство HSIO может измерять ширину импульса, формировать сигналы и периодически производить прерывания. В зависимости от режима HSIO может производить работу с 18 таймерами/счетчиками и регистрами захвата/сравнения.

4.1 Выход блока широтно-импульсной модуляции (PWM)

Цифро-аналоговое преобразование (D/A) может быть произведено на выходе блока широтно-импульсной модуляции (PWM). Выходной сигнал – импульс с изменяемой скважностью, с повторением каждые 256 или 512 циклов системного тактового сигнала, если подключен делитель на 2. Изменение скважности производится записью в PWM регистр.

4.2 Таймеры

В ИС для пользователя доступны два 16-разрядных таймера. Первый обозначен, как Таймер 1, второй – Таймер 2. Таймер 1 используется для синхронизации событий в реальном масштабе времени, тогда как Таймер 2 тактируется снаружи и синхронизирует действия по отношению к внешним событиям. Таймеры являются времязадающей базой блоков высокоскоростных входов (HSI) и высокоскоростных выходов (HSO) и могут считаться встроенной частью этих блоков.

Таймер 1 – это автономный таймер, который инкрементируется каждые восемь машинных циклов. Таймер 1 может вызывать прерывания при переполнении.

Таймер 2 считает переключения по передним и задним фронтам по любому входу – T2CLK или HSI.1. Таймер 2 может быть читаемым и записываемым, может быть сброшен аппаратно, программно и блоком HSO. Он может использоваться как счетчик с инкременцией/декременцией (при управлении по P2.6) и его значение может быть захвачено T2CAP-регистром. Прерывания могут генерироваться после захвата значения Таймера 2, и если значение Таймера 2 выходит за границы 0FFFFH/0000H или 7FFFH/8000H (при обратном счете).

4.3 Высокоскоростные входы (HSI)

Устройство высокоскоростных входов (HSI) может захватывать значение Таймера 1, когда событие имеет место на одном из четырех входов (HSI.0 – HSI.3). Четыре типа событий могут фиксироваться: положительный фронт, отрицательный фронт, положительный или отрицательный фронт или каждый восьмой положительный фронт. Блок-схема HSI приведена на рисунке 4.3.1.

Когда значение из регистра HSI_TIME считано, следующий FIFO регистр загружается в регистр хранения. Три типа HSI-прерываний могут генерироваться, когда значение передается из FIFO в регистр хранения, когда FIFO (независимо от буферного регистра) запоминает четыре или более событий.

Это позволяет оптимизировать HSI для ожидаемой частоты прерываний. Независимо от HSI операций, состояние HSI выводов отображается 4 битами регистра HSI_STATUS. Также, независимо от HSI операций, HSI.0 может использоваться для внешнего прерывания, если вывод (HSI.0) не установлен как вход HSI.

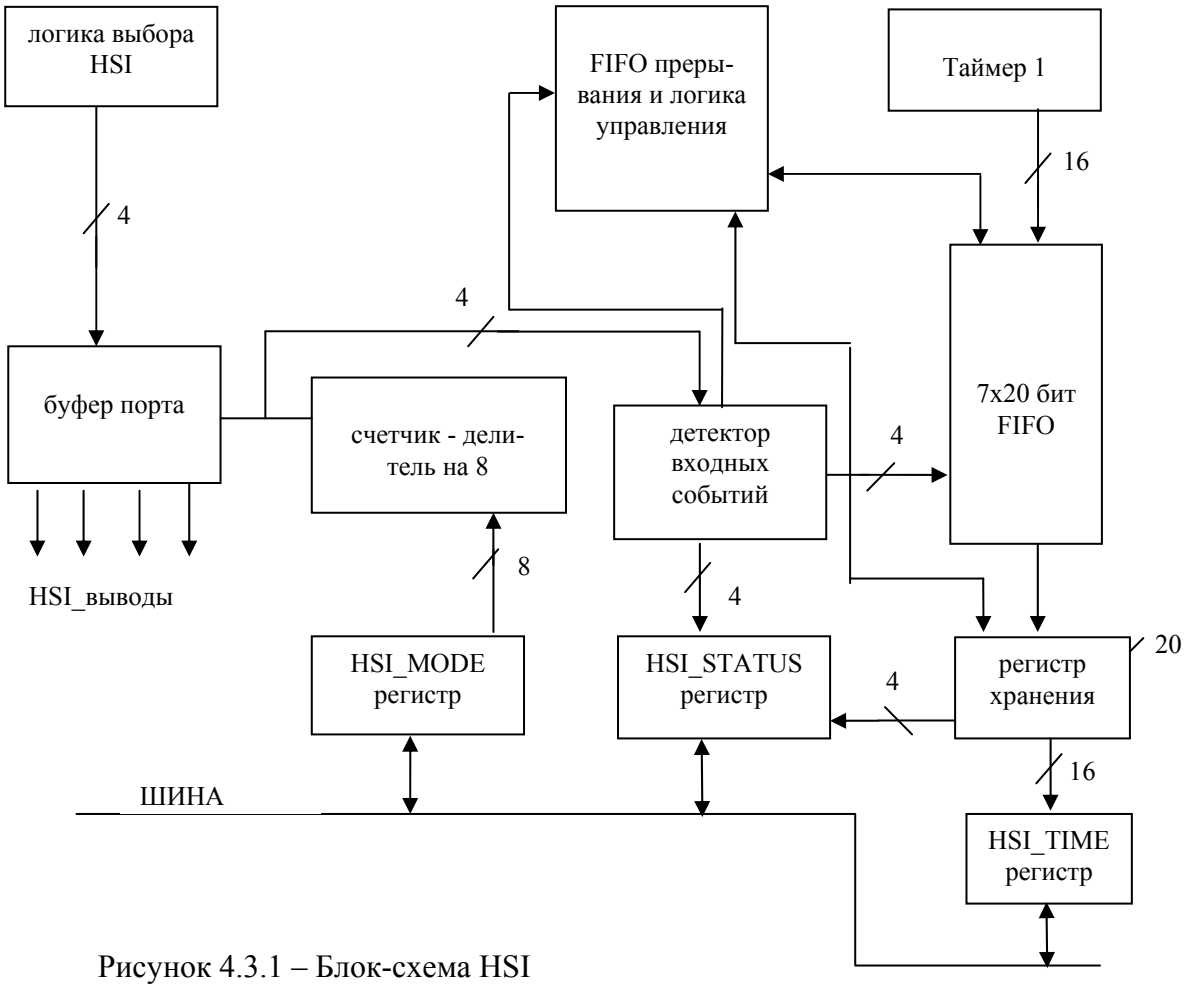


Рисунок 4.3.1 – Блок-схема HSI

4.4 Высокоскоростные выходы (HSO)

Устройство высокоскоростных выходов может генерировать события в определенное время или на базе Таймера 1 или Таймера 2 с минимальным управлением от CPU.

Блок-схема устройства HSO показана на рисунке 4.4.1. До восьми задержанных событий можно запомнить в CAM устройстве HSO одновременно. Команды, поступающие в HSO, записываются в HSO_COMMAND для формирования события и затем в HSO_TIME для согласования со значением таймера.

14 различных типов событий могут обрабатываться HSO: 8 внешних и 6 внутренних. Два вектора прерываний базируются на HSO: один – для внешних событий, другой – для внутренних. Внешние события представляют переключения по одному или нескольким выводам HSO.0 – HSO.5. Внутренние события включают в себя установку до четырех программных таймеров, сброс Таймера 2 и старт аналого-цифрового преобразования. Флаги программных таймеров могут устанавливаться HSO и дополнительно вызывать прерывания.

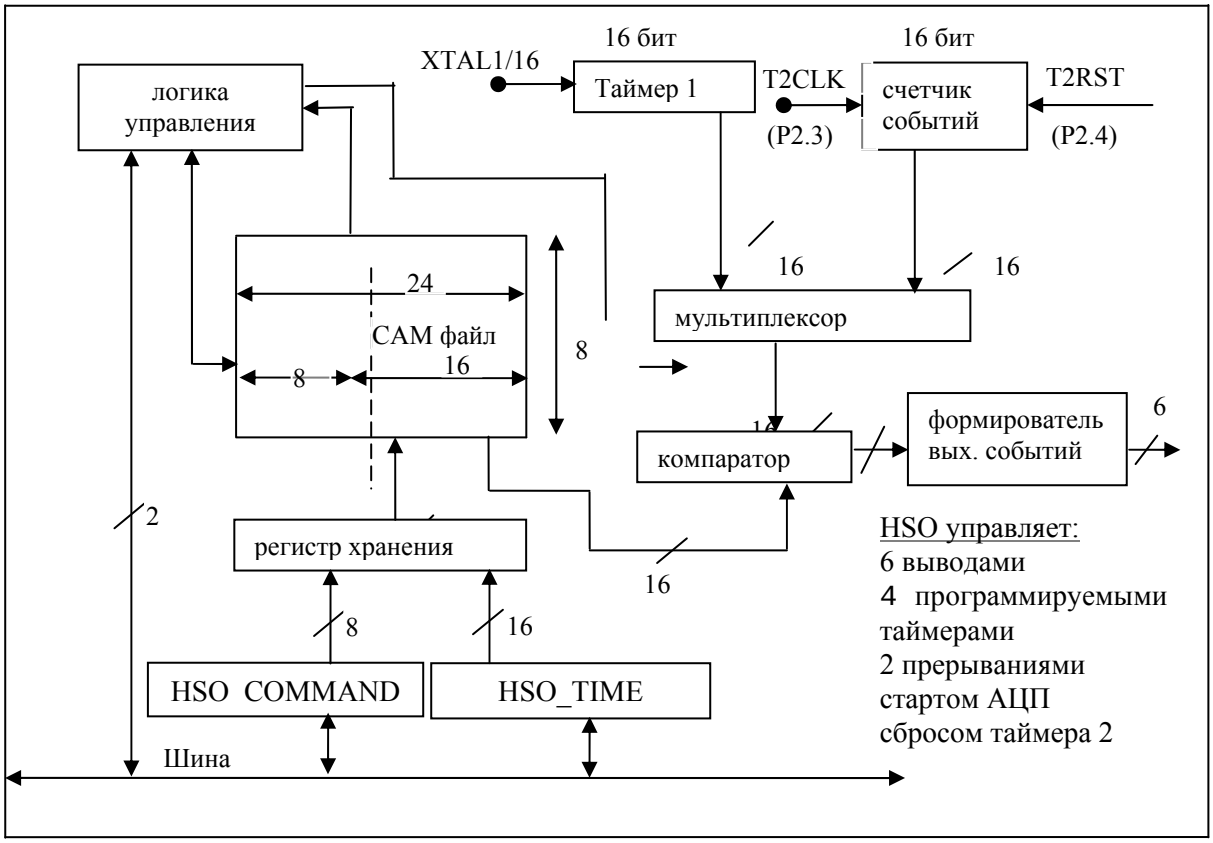


Рисунок 4.4.1 – Блок-схема HSO

4.5 Последовательный порт

Последовательный порт ИС функционально совместим с последовательным портом микроконтроллерного семейства 1830. Имеются один синхронный и три асинхронных режима. Асинхронный режим полностью дуплексный, то есть передача и прием в одно время. Двойная буферизация предусмотрена для приемника таким образом, что второй байт может быть принят раньше, чем первый байт принятой информации считан. Передатчик имеет также двойную буферизацию, позволяющую записывать байты (в буфер), пока идет передача.

Регистр состояния последовательного порта (SP_STAT) содержит биты переполнения, четности, ошибок, прерывания приема и передачи.

Режимы последовательного порта:

1 РЕЖИМ 0 (Mode 0)

Режим 0 – это синхронный режим, обычно используемый для увеличения скорости ввода-вывода с помощью регистра сдвига. В этом режиме TXD выдает набор из восьми импульсов, тогда как RXD служит для передачи и приема данных. Данные передаются по 8 бит, начиная с младшего бита. Следует отметить, что это единственный режим, в котором RXD используется как выход.

2 РЕЖИМ 1 (Mode 1)

Это стандартный асинхронный режим связи. Посылка содержит 10 бит: стартовый бит (0), 8 бит данных (младший – первый), стоповый (1). Если четность разрешена (SPCON.2), то бит четности посылается за восьмым битом данных, и четность проверяется на приеме.

3 РЕЖИМ 2 (Mode 2)

Режим 2 – это асинхронный 9-битный режим. Обычно он используется совместно с режимом 3 для мультипроцессорных коммуникаций.

Данные содержат: стартовый бит (0), 9 бит данных (младший – первый), стоповый бит (1). При передаче девятый бит может быть установлен в 1, через установку бита TB8 в управляющем регистре перед записью в SBUF(TX). Бит TB8 очищается при каждой передаче, поэтому он должен быть установлен перед записью в SBUF(TX) каждый раз, когда это требуется. При приеме прерывание последовательного порта и прерывание приема (RI) не будут устанавливаться, если девятый принятый бит не будет установлен.

4 РЕЖИМ 3 (Mode 3)

Это асинхронный 9-битный режим. Посылка данных идентична посылке для режима 2. Отличие заключается в том, что при передаче бит четности может быть разрешен (PEN = 1), и тогда девятый бит будет принимать четное значение. Бит TB8 может использоваться, если бит четности запрещен (PEN = 0). В режиме 3 прием данных будет всегда вызывать прерывание вне зависимости от состояния 9-ого бита. Если PEN = 0, то девятый бит запоминается и может быть прочитан из RB8. Если PEN = 1, то RB8 станет флагом ошибки четности на приеме (RPE).

4.6 Блок АЦП

Аналоговый интерфейс ИС содержит блок захвата и хранения аналогового сигнала, 8-канальный мультиплексор и блок 10-разрядного аналого-цифрового преобразователя.

Аналоговый сигнал выбирается на одном из 8 аналоговых входов (ACH0 – ACH7), которые входят в порт 0. АЦ-преобразования производят на одном из входов, с использованием последовательной аппроксимации; результат равен отношению входного напряжения к опорному напряжению. Если отношение – единица, то результат – все 1. Преобразование может быть запущено записью в регистр A/D_Command или командой HSO.

4.7 Порты ввода-вывода

ИС имеют пять 8-битных портов ввода-вывода. Часть из них чисто входные, часть работают только на выход, некоторые двунаправленные, а некоторые имеют альтернативные функции. В дополнение к ним выводы устройства быстрого ввода-вывода могут использоваться как линии обычного ввода-вывода, если не нужны специальные возможности устройства быстрого ввода-вывода.

Входные порты соединены с внутренней шиной микроконтроллеров через входные буфера. Выходные порты имеют регистры-защелки для выходных данных и выходные буфера. Двунаправленные порты содержат выходной буфер, регистр-защелку и входной буфер.

Порт 0 – входной; может использоваться как аналоговый вход для АЦП. Порт 1 – квази-двунаправленный. Порт 2 содержит 3 типа выводов: квази-двунаправленные, входные, выходные.

Порты 3 и 4 – двунаправленные; к ним подключена шина внешнего адреса/данных.

4.8 Сторожевой таймер

Сторожевой таймер (WDT) позволяет восстанавливать нормальную работу при сбоях программ. Если WDT разрешен, он будет вызывать аппаратный сброс, если программа не очищает его каждые 64К машинных цикла.

Когда таймер переполняется, он переводит вход RESET# в состояние низкого уровня, сбрасывая ИС и другие устройства, подключенные к выводу RESET#.

Работа WDT разрешается первой же его очисткой. Если она разрешена, то запрещена может быть только аппаратным сбросом.

5 Прерывания

28 источников прерываний реализованы в архитектуре микроконтроллеров. Эти источники используют 15 векторов и дополнительно специальные векторы для NMI, команды TRAP и невыполняемых кодов операций. Рисунок 5.1 показывает связь источников прерываний с их векторами прерываний.

Специальные прерывания

Три специальных прерывания реализованы в ИС: NMI, TRAP и невыполнимые коды операций. Внешний вывод NMI генерирует немаскируемое прерывание для использования в программах, критических к прерываниям.

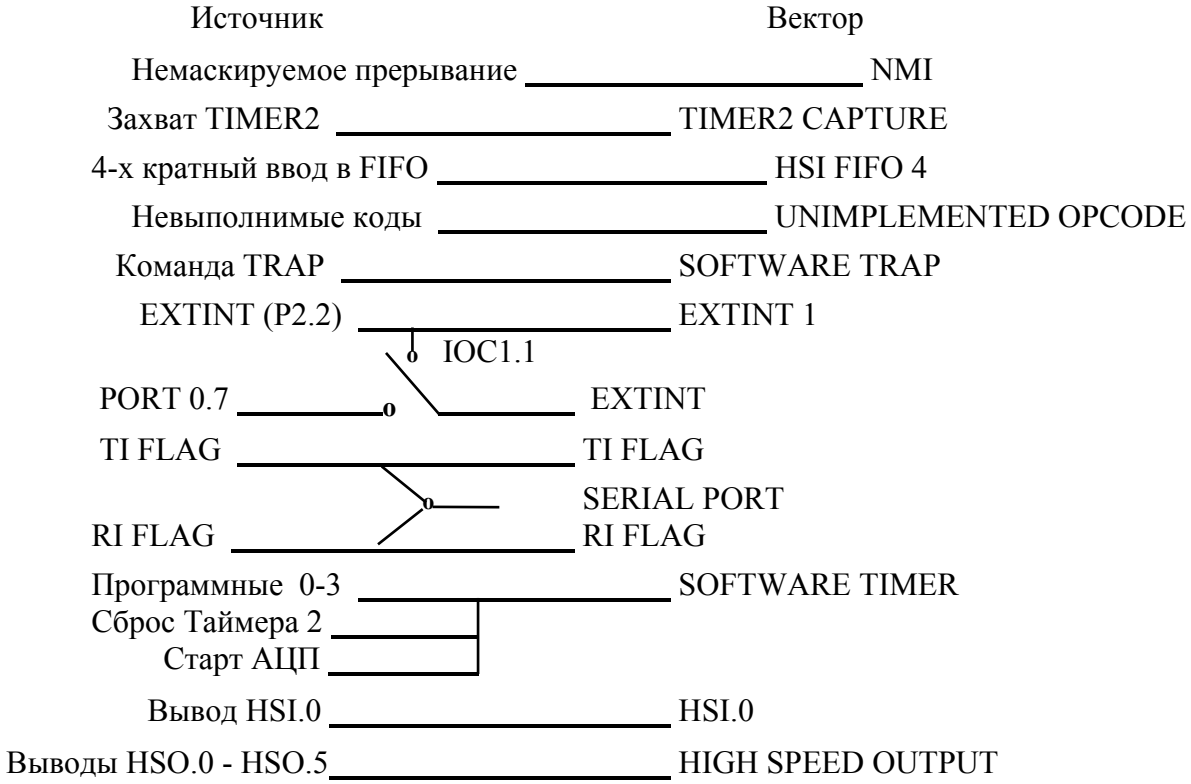
Команда TRAP полезна для разработки отладчиков программного обеспечения или генерации программных прерываний. Прерывание по невыполняемым кодам операций обеспечивает программное возвращение из любой выполняемой программы при возникновении аппаратной или программной ошибки.

NMI

NMI – внешне немаскируемое прерывание, имеющее наивысший приоритет. Косвенный доступ к вектору осуществляется через ячейку 203EH. Маскирующий бит для NMI создан в регистре INT_MASK 1. Для предупреждения случайного маскирования NMI, этот бит не нужно использовать. Для будущей совместимости маскирующий бит должен быть установлен в ноль.

TRAP

Код операции 0F7H команды TRAP реализует косвенный вызов вектора через ячейку 2010H. Команда TRAP обеспечивает однокомандное прерывание, удобное при разработке программных отладчиков. Команда TRAP предотвращает запросы прерываний до тех пор, пока не выполнится следующая команда.



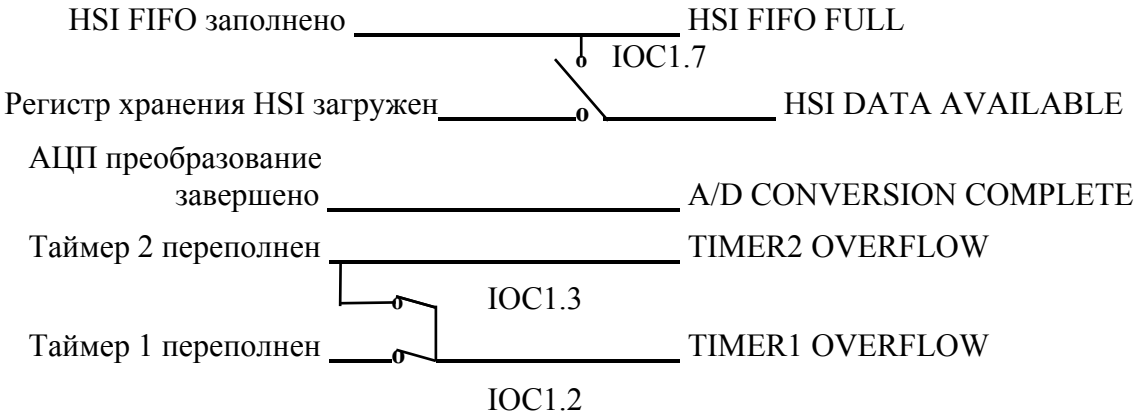


Рисунок 5.1 – Источники прерываний микроконтроллеров

Невыполняемые коды операций

Коды операций, которые не выполняются микроконтроллерами, являются причиной прерывания по вектору с адресом 2012H. Коды команд и аппаратура, которые используют невыполнимые коды операций, могут восстанавливаться программным обеспечением по этому прерыванию. Команда DJNZW не поддерживается ИС, однако остается значащим кодом операции, поэтому не вызывает прерывания.

Программист должен инициализировать таблицу векторов прерываний с соответствующими стартовыми адресами программ обработки прерываний

Пять регистров управляют системой прерываний: INT_PEND, INT_PEND1, INT_MASK, INT_MASK1 и PSW с битом глобального запрета. Блок-схема системы прерываний показана на рисунке 5.2.

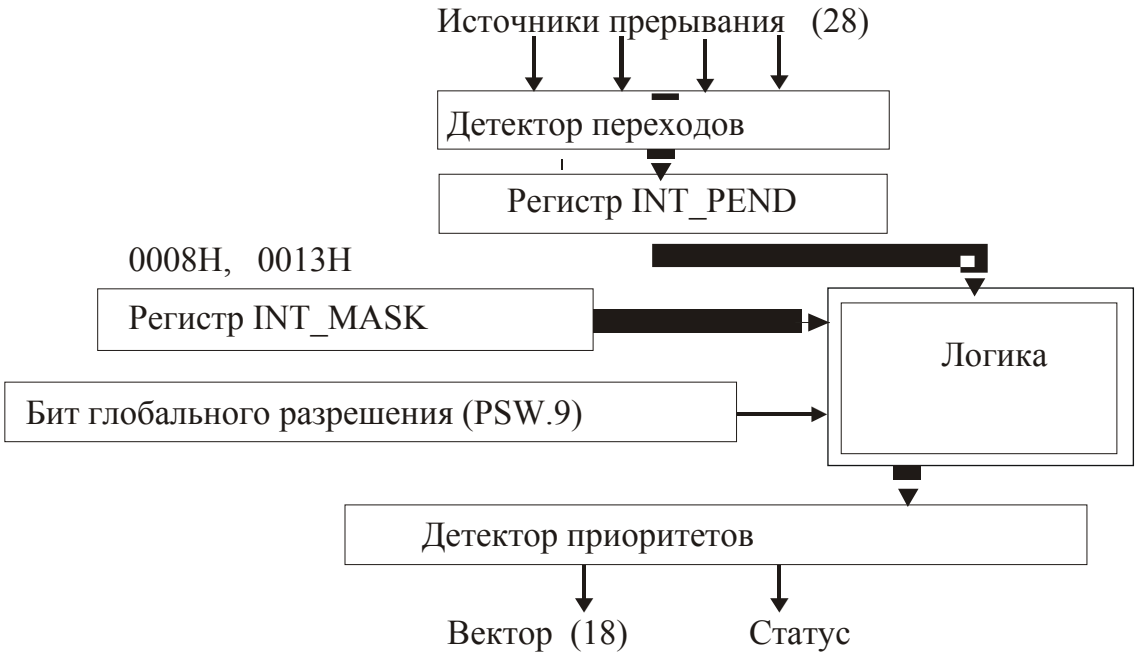


Рисунок 5.2 – Блок-схема системы прерываний

Детектор переходов следит за переходом из 0 в 1 от любого источника прерываний. Внешние источники имеют максимальную скорость переходов, равную одному переключению за один период системного тактового сигнала. Выполнение гарантировано, если уровень сигнала по линии прерывания удержит, по крайней мере, один цикл системного тактового сигнала. Если источник прерываний переключается чаще, чем один раз в течение системного тактового сигнала, то прерывание может быть не обнаружено.

5.1 Управление прерываниями

Регистр задержки прерываний

Когда аппаратное обеспечение определяет одно из 16 прерываний, оно устанавливает соответствующий бит в один из двух регистров задержки прерываний (INT_PEND - 09H и INT_PEND1 - 12H). Когда вектор прерывания выбран, соответствующий бит в регистре задержки прерываний очищается. Эти регистры, формат которых показан на рисунке 5.1.1, могут быть прочитаны или модифицированы, как байтовые регистры. Они могут быть считаны для определения, какие запросы прерываний задержаны, модифицированы для очистки или будут вызывать прерывание. Программное обеспечение INT_PEND регистром должно обеспечивать непрерывность операций. Простейший путь к этому в использовании логических инструкций с двумя или тремя операндами, например:

```
ANDB    INT_PEND, # 11111101B;   очистка прерывания от АЦП
ORB     INT_PEND, # 00000010B;   установка прерывания от АЦП.
```

Осторожно нужно использовать очистку битов в режиме задержанных прерываний. Если запрос прерывания происходит одновременно с очисткой бита, то имеет место "пустое" прерывание в течение пяти циклов системного тактового сигнала. Это сделано потому, что должен продолжаться нормальный поток команд. Эффект такой же, как при выполнении лишних двух инструкций NOP. Это может быть предотвращено использованием для очистки битов логических двухоперандных команд с непосредственной адресацией, так как микроконтроллеры прекращают распознавание прерываний в течение этих инструкций типа "чтение - модификация - запись".

Регистры масок прерываний

Отдельные прерывания могут быть разрешены или запрещены установкой или очисткой битов в регистрах масок прерываний (INT_MASK-(08H) и INT_MASK1-(13H)). Формат этих регистров такой же, как у регистра задержки прерываний INT_PEND, и приведен на рисунке 5.1.1.

INT_PEND (09H)/INT_MASK (08H)

0	Таймер переполнен
1	АЦП завершено
2	HSI данные доступны
3	HSO выводы
4	Вывод HSI.0
5	Программный таймер
6	Последовательный порт
7	Внешнее прерывание

INT_PEND1 (12H)/INT_MASK1 (13H)

0	TI - прерывание при передаче
1	RI - прерывание при приеме
2	HSI FIFO 4 - заполнен
3	Прерывание по выборке Таймера 2 - T2CAP
4	Таймер 2 переполнен
5	EXTINT 1
6	HSI FIFO заполнен
7	NMI - немаскируемое прерывание

Рисунок 5.1.1 – Байтовые регистры

Регистры INT_MASK и INT_MASK 1 могут читаться и записываться, как байтовые регистры. Очистка или установка любого бита запрещает или разрешает соответствующее прерывание. Аппаратура запоминает запросы прерывания установкой битов в регистрах задержанных прерываний, вне зависимости состояния регистров масок прерываний. Регистр INT_MASK это младшие восемь бит регистра PSW, поэтому команды PUSHF и POPF сохраняют и восстанавливают регистр INT_MASK также, как бит общего запрета и арифметические флаги. Оба регистра INT_MASK и INT_MASK 1 могут быть сохранены командами PUSHA и POPA.

Общий запрет прерываний

Выполнение всех прерываний, исключая NMI, TRAP прерывания по невыполнимым кодам операций, могут быть запрещены очисткой бита I в PSW. Установка бита I разрешает прерывания, в соответствии установленными битами в регистрах масок прерываний. Бит I управляется командами EI (разрешить прерывание) и DI (запретить прерывание). Напоминаем, что бит I (регистра PSW) только управляет обслуживанием запросов прерываний. Прерывания, которые имеют место в течение времени опроса прерываний, запоминаются в регистрах задержанных прерываний и выполняются в соответствии с их приоритетами, когда период опроса завершен.

5.2. Приоритеты прерываний

Дешифратор приоритетов обслуживает все прерывания, которые разрешены и запросы которых хранятся в обоих регистрах задержанных прерываний. (INT_PEND, INT_PEND1) и выбирает одно с высшим приоритетом. Приоритеты показаны на рисунке 5.2.1 (15-й - старший, 0-й - младший)

Номер	Источник прерываний	Адрес вектора прерываний	Приоритет
INT 15	NMI	203EH	15
INT 14	HSI FIFO заполнен	203CH	14
INT 13	EXTINT 1	203AH	13
INT 12	Таймер 2 переполнен	2038H	12
INT 11	Выборка Таймера 2	2036H	11
INT 10	4-х кратное заполнение HSI FIFO	2034H	10
INT 09	RI	2032H	9
INT 08	TI	2030H	8
спец-ое	Невыполняемые коды	2012H	-
спец-ое	TRAP	2010H	-
INT 07	EXTINT	200EH	7
INT 06	Последовательный порт	200CH	6
INT 05	Программный таймер	200AH	5
INT 04	Вывод HSI.0	2008H	4
INT 03	HSO	2006H	3
INT 02	HSI данные доступны	2004H	2
INT 01	АЦП завершено	2002H	1
INT 00	Таймер 1 переполнен	2000H	0

Рисунок 5.2.1 - Приоритеты прерываний

Это управление порядком выбора по приоритетам прерываний, запросы которых имеются в регистрах INT_PEND, INT_PEND 1, осуществляются программным обеспечением через вызовы программ, обслуживающих прерывания. Программное обеспечение может создавать свою собственную структуру приоритетов посредством управления регистром масок (INT_MASK, INT_MASK 1). Для просмотра, как это делается, рассмотрим случай программы обработки прерывания последовательного ввода/вывода, которая

должна запускаться с уровнем приоритета меньшим, чем прерывание "HSI данные доступны", однако выше, чем другие источники прерываний. Заголовок и исполняемый код для обработки такого прерывания должны быть такими:

```
Serial_io_isr:
PUSHA       ; сохранить PSW, INT_MASK
            ; INT_MASK 1 и WSR
LDB   INT_MASK, #00000100B
EI          ; снова разрешить прерывание
;    ⎫
;    ⎪
;    ⎬ Программа обслуживания прерывания
;    ⎪
;    ⎭
POPA        ; восстановление
RET .
```

Следует отметить, что адрес 200CH в таблице векторов прерываний должен быть загружен с меткой SERIAL_IO_ISR и прерывание будет доступно для программы обслуживания прерывания.

5.3 Критические ситуации

Рассмотрим очистку бита в регистре INT_PEND как части программы, "не обрабатывающей" прерывания:

```
LDB         AL,   INT_PEND
ANDB        AL,   # bit _mask
STB         AL,   INT_PEND.
```

Эти коды будут вычисляться, если нет других подпрограмм (п/п), работающих конкурентно; но будут выполняться очень нестабильно, если такие п/п есть. (Все программы, которые используют прерывания, должны рассматриваться, как находящиеся в конкурирующем окружении). Чтобы проиллюстрировать эту проблему, рассмотрим случай, когда INT_PEND содержит 00001111B и бит 3 (HSO-прерывание) будет сброшен. Что произойдет, если прерывание от HSI произойдет где-то между инструкциями LDB и STB? До LDB INT_PEND содержит 00001111B, а после LDB то же самое содержит и AL. Если прерывание от HSI произойдет в этой точке, то INT_PEND станет равным 00001011B. Инструкция ANDB изменит AL на 00000111B, а STB изменит значение INT_PEND на 00000111B. Это вызовет генерацию "фальшивого" прерывания от HSI.

Во многих случаях набор инструкций ИС поддерживает модификацию переменной при помощи одной инструкции. Выше приведенные команды могут быть заменены одной инструкцией:

```
ANDB        INT_PEND,   # bit _mask.
```

Изменения в INT_PEND должны быть внесены одной инструкцией, в ином случае биты в этом регистре можно менять, даже если прерывания запрещены. В зависимости от конфигурации системы, другие регистры специального назначения должны изменяться одной инструкцией по сходным причинам.

Когда переменные должны быть изменены без прерываний, а одна инструкция использована быть не может, программист должен создать в программе так называемую "защищенную область", в которой безопасно менять переменные. Один путь – запретить прерывания (DI), изменить переменные, разрешить прерывания (EI). Но такое решение вызывает новую проблему: в конце бит I в PSW всегда установлен, хотя при старте он мог быть и сброшен.

Лучшее решение – войти в "защищенную область" инструкцией PUSHF, которая сохраняет PSW и очищает все флаги прерываний, а выйти из него инструкцией POPF, восстанавливающей предыдущее состояние. Следует отметить, что в некоторых системных конфигурациях может потребоваться большая защита "защищенной области".

Пример тому – система, где более чем один процессор, имеет доступ к общей памяти и устройствам ввода-вывода.

5.4 Временные соотношения при прерываниях

ИС могут иметь четыре различных внешних источника прерываний: NMI, P2.2, HSI.0 и P0.7. Все внешние прерывания детектируются в течение PH1 или по низкому уровню CLKOUT и захватываются внутри. Уровень сигнала внешнего прерывания должен поддерживать не менее одного машинного цикла, чтобы обеспечивать распознавание прерывания.

Внешние прерывания, выбираемые в течение PH1, являются прерываниями, захватываемыми по фронту, в противоположность захватываемым по уровню. Прерывания, захватываемые по фронту, должны генерировать только одно прерывание, если вход удержан в высоком уровне. С другой стороны, прерывания по уровню должны генерировать множественные прерывания, когда удерживается высокий уровень на входе прерываний.

Прерывания не всегда распознаются немедленно. Если сигнал прерывания не установлен за четыре периода системного тактового сигнала до конца выполнения команды, прерывание не может быть распознано до конца выполнения следующей команды. Это происходит потому, что инструкции считываются и готовятся для выполнения в течение нескольких периодов системного тактового сигнала перед непосредственным выполнением.

Имеются шесть инструкций, которые всегда задерживают распознавание прерываний до завершения следующей инструкции.

Это инструкции:

EI, DI – разрешает и запрещает все прерывания посредством управления битом глобального запрета (PSW.9);

PUSHF – сохраняет в стеке пару PSW/INT_ MASK и затем очищает, оставляя INT_MASK и PSW.9 очищенными;

POPF – извлечение PSW и INT_MASK из стека;

PUSHA – выполняет сохранение всего, т.е. выполняет PUSHF, затем сохраняет INT_MASK1 и WSR и очищает INT_MASK1;

POPA – выполняет восстановление из стека всего, т.е. восстанавливает INT_MASK1, WSR и затем выполняет POPF.

Прерывания не могут выполняться непосредственно после выполнения невыполняемых кодов операций:

TRAP – команды программного прерывания;

SIGND – префикс при выполнении команд умножения и деления.

Когда прерывание опознано, соответствующий бит в регистрах INT_PEND или INT_PEND1 очищается и происходит вызов по адресу соответствующего вектора прерывания. Этот вызов происходит после выполнения текущей инструкции, исключение – выше описанные. Процедура считывания вектора и запуска вызова требует 16 периодов системного тактового сигнала. Если стек находится во внешнем ОЗУ, требуется еще два дополнительных периода. Максимальное число циклов (периодов системного тактового сигнала), требуемое для генерации прерывания (не распознавания), пока микроконтроллеры начнут выполнение программы с желаемого адреса – это время наиболее длинной инструкции NORML (нормализация – 39 циклов), плюс четыре цикла перед окончанием предыдущей инструкции, плюс время вызова (16 внутренний стек) или 18 (внешний стек) циклов.

Поэтому максимальное время от запроса до начала выполнения программы обработки прерывания составляет 61 машинный цикл (39+4+18). Это не включает 10 машинных циклов, требуемых для PUSHF, если она используется как первая инструкция в программе обработки прерывания, или добавочной задержки по причине маскирования запрета или прерывания. Это показано на рисунке 5.4.1

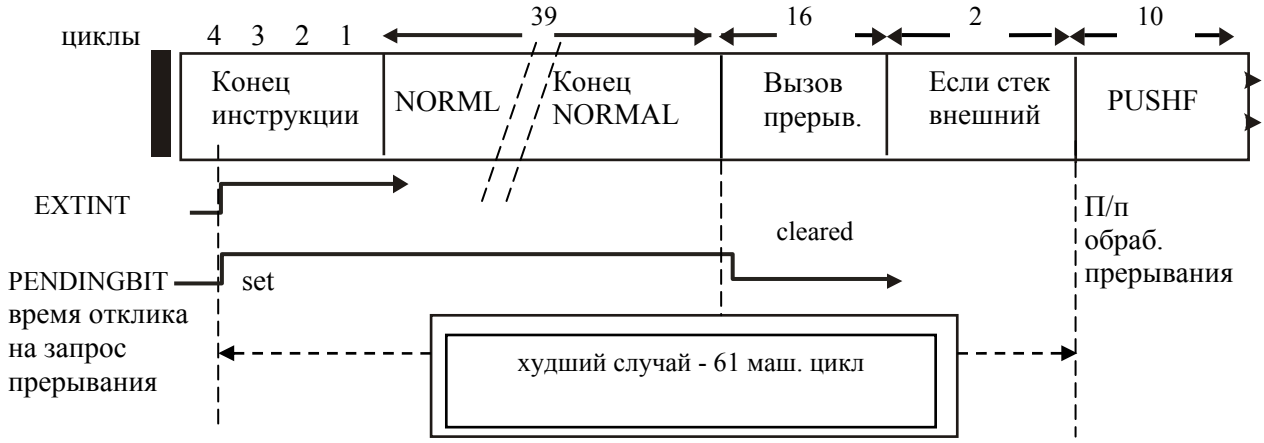


Рисунок 5.4.1 - Время прерывания

Время отклика на прерывание может быть уменьшено тщательной подборкой инструкций в области программы, где прерывания ожидаются. Использование EI непосредственно после длиной инструкций (например MUL, NORML и т.д.) должно увеличивать максимальную задержку на 4 машинных цикла, так как прерывание не может иметь место между EI и инструкцией следующей за EI. Инструкции DI, PUSHF, POPF, PUSHA, POPA и TRAP являются причиной такой же ситуации. Эти инструкции создают задержку, когда одна программа обработки прерывания уже исполняется, т.к. они редко используются в других случаях.

5.5 Общее описание источников прерываний

"EXTINT" и "P0.7"

Микроконтроллер имеет два вектора внешних прерываний EXTINT (200EH) и EXTINT1 (203AH). Вектор EXTINT имеет два альтернативных источника выбираемые посредством IOC1.1, вход внешнего прерывания P2.2 и P0.7. Вход внешнего прерывания P2.2 является единственным источником для вектора прерывания EXTINT1. Вход внешнего прерывания не желательно использовать для обоих векторов. Оба внешних сигнала запрета прерываний защелкиваются по переходу из низкого в высокий уровень.

Прерывания последовательного порта

Последовательный порт генерирует одно из трех возможных прерываний: прерывание при передаче TI(2030H), прерывание при приеме RI(2032H), последовательный порт (200CH).

"HSI FIFO заполнен" и "HSI данные доступны"

Источник прерывания "HSI данные доступны" управляется установками Регистра управления вводом-выводом 1 (IOC1.7). Установка IOC1.7 в ноль должно генерировать прерывание, когда значение времени загружается в регистр хранения. Установка бита в единицу генерирует прерывание, когда FIFO независимо от регистра хранения, заполнен 6-ю событиями. Для прерывания "HSI FIFO заполнен" (203CH) и "HSI данные доступны" (2004H) используются разные адреса векторов прерываний.

"HSI_FIFO_4"

HSI FIFO может генерировать прерывание, когда HSI имеет четыре или более записей в FIFO. Прерывание HSI_FIFO_4 обрабатывается по вектору из ячейки 2034H.

HSI.0 внешнее прерывание.

Фронт сигнала из 0 в 1 по выводу HSI.0 может быть использован в качестве внешнего прерывания. HSI.0 детектируется в течение PH1 или CLKOUT низкого. Исполнение гарантировано, если уровень сигнала удерживается в течении машинного цикла. Прерывание имеет вектор по адресу 2008H. Этот вывод не нужно разрешать, как HSI FIFO вывод для генерирования прерывания.

Таймер_1 и Таймер_2 переполнены

Эти таймеры могут вызывать прерывания по переполнению. Для Таймера_1 вектор прерывания из ячейки 2000H, для Таймера_2 - 2038H. Прерывания могут быть индивидуально разрешены установкой бита 2 или 3 в IOC1: бит 2 для Таймера_1, бит 3 для Таймера_2, конкретный источник прерывания, это можно определить по битам 4 или 5 в регистре IOS1. Таймер_2 переполнен при значении 0H или 8000H.

Выборка Таймера_2

Микроконтроллер может генерировать прерывание при считывание значения Таймера_2 по положительному фронту сигнала (из 0 в 1) на вывод P2.7. Адрес вектора прерывания 2036H.

Высокоскоростные выходы (HSO)

Прерывание высокоскоростных выходов может генерироваться при внешнем событии, запрограммированном в HSO_COMMAND на прерывание. Когда HSO_ COMMAND очищает или устанавливает выводы HSO - это рассматривается, как внешнее событие. Регистр состояния IOS2 отслеживает HSO события, когда они происходят, и это можно использовать для контроля, какой бит HSO_COMMAND явился причиной прерывания. Прерывания обслуживаются через вектор по адресу 2006H.

Программные Таймеры

Когда происходит внутреннее событие, создаваемое командой HSO, оно может иметь прерывание обслуживаемое через вектор прерывания программного таймера, если бит прерывания в команде HSO установлен. Ко внутренним событиям относится старт АЦП, сброс Таймера 2 и программные таймеры. Регистры состояния IOS2 и IOS1 могут использоваться для определения, какое внутреннее HSO событие имеет место. Вектор прерывания программного таймера имеет адрес 200AH.

6 Блок Широтно-Импульсной модуляции (ШИМ)

Цифро-аналоговое преобразование может быть выполнено на входе PWM. Блок-схема приведена на рисунке 6.1. 8-разрядный счетчик инкрементируется каждый машинный цикл. Когда данные в счетчике равны 0, выход ШИМ (PWM) устанавливается в 1, когда содержимое счетчика равно данным в регистре ШИМ, выход ШИМ переключается в 0. Когда счетчик переполняется, выход PWM снова переключается в 1. Типовая форма выходного сигнала показана на рисунке 6.2.

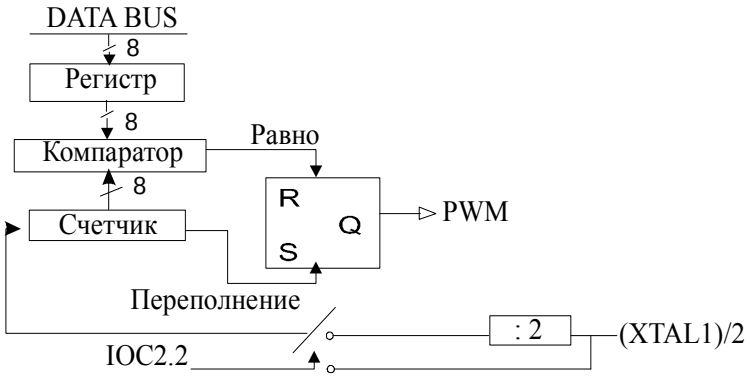


Рисунок 6-1 Выход с ШИМ

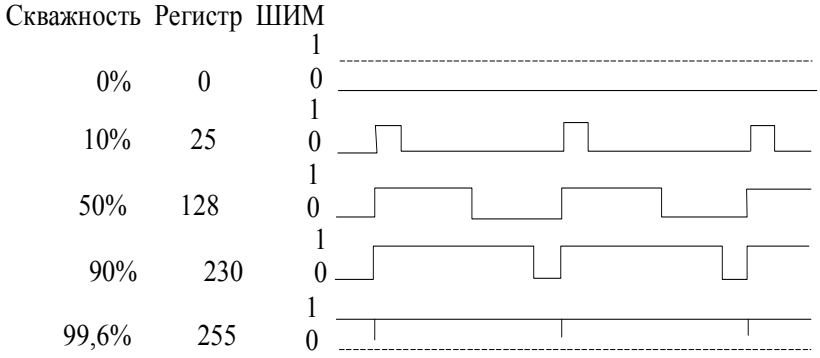


Рисунок 6.2 - Типовые сигналы ШИМ (диаграмма)

Следует отметить, когда в регистре ШИМ записан 0, на выходе PWM всегда низкий уровень. А также регистр ШИМ может быть перегружен только из временной защелки когда счетчик переполнен. Это значит, что компаратор не распознает нового значения регистра ШИМ для сравнения, пока текущий цикл ШИМ не завершен. Устройство ШИМ имеет управляющий бит (деление частоты на 2) IOC 2.2, разрешающий деление при установке его в 1 (IOC 2.2 = 1). Зависимости частоты на выходе PWM от XTAL1 и IOC 2.2 приведены ниже.

XTAL1	8 МГц	10 МГц	12 МГц
IOC 2.2 = 0	15,6 КГц	19,6 КГц	23,6 КГц
IOC 2.2 = 1	7,8 КГц	9,8 КГц	11,8 КГц

Выходной сигнал - это сигнал с переменной скважностью с периодом 256 или 512 машинных цикла. Изменения скважности производится записью данных в регистр по адресу 17H. Содержимое регистра ШИМ может быть прочитано в окне 15 (WSR = 15). Имеется несколько типов двигателей, которые нуждаются в ШИМ сигнале для более эффективной работы. Дополнительно, если ШИМ сигнал проинтегрировать, формируется уровень постоянного напряжения, имеющий 256 градаций. Выход PWM использует вывод 5 в порту 2, поэтому две эти функции не могут использоваться одновременно. Установка разряда IOC1.0 в 1 выбирает функцию ШИМ вместо стандартной функции порта общего назначения.

7 Таймеры

7.1 Таймер_1

Таймер_1 – это 16-разрядный автономный таймер с инкременцией каждые восемь машинных циклов. Прерывание может генерироваться при переполнении. Таймер_1 читается по адресу 0AH в окне 0 и записывается в окне 15. Запись в Таймер_1 должна производиться таким образом, чтобы обеспечить выполнение HSO событий без пропуска и в нужном порядке. Изменения в Таймере_1 должны происходить с учетом событий на входах HSI, если эти входы используются.

7.2 Таймер_2

Таймер_2 может использоваться с блоком HSO, как счетчик с прямым или обратным счетом, для выборки внешних событий или как дополнительный счетчик. Таймер_2 синхронизируется внешними сигналами по выводу T2CLK (ввод P2.3) или выводу HSI.1 в зависимости от состояния IOC0.7. Таймер_2 может инкрементироваться по переднему и заднему фронту. Максимальная скорость переключения – это один раз в восемь машинных циклов (режим быстрого счета). CLKOUT не может использоваться напрямую для синхронизации Таймера_2. Его период должен быть сначала увеличен вдвое. Таймер_2 может читаться и записываться по адресу 0CH в окне 0. Таймер_2 может быть сброшен аппаратно, программно или от блока HSO. T2RST (P2.4) или HSI.0 могут сбросить Таймер_2 в зависимости от установки разряда IOC0.5. Соответствующий управляющий регистр может читаться в окне 15 для определения запрограммированного режима. IOC0.1 (T2RST) не защелкивается и читается 1.

Регистр сбора данных

Значение Таймер_2 может быть загружено в регистр T2CAP по восходящему фронту сигнала P2.7. Уровень сигнала должен удерживаться в течение, по крайней мере, одного машинного цикла. T2CAP размещен по адресу 0CH в окне 15. Прерывание генерируется векторами захвата через ячейку с адресом 2036H.

Режим быстрого счета

Таймер_2 может быть запрограммирован для работы в режиме быстрого счета для подсчета переходов в течение каждого машинного цикла. Установка IOC2.0 программирует Таймер_2 на режим быстрого счета. В этом режиме события, запрограммированные в блоке HSO на базе Таймер_2, не будут выполняться надлежащим образом, так как HSO требует восемь машинных циклов для сравнения каждой ячейки ассоциативной памяти HSO. При использовании Таймер_2 в качестве базового для блока HSO переходы Таймер_2 в течение каждого машинного цикла могут привести к потере запрограммируемых результатов HSO. По этой причине Таймер_2 не должен использоваться в качестве базы для HSO, если переходы происходят чаще, чем один раз в течение восьми машинных циклов.

Таймер_2 не может быть сброшен в режиме быстрого счета. Все сбросы синхронизированы по тактирующему импульсу длительностью восемь машинных циклов.

Режим реверсивного счетчика

Таймер_2 может быть настроен на счет в прямом и обратном направлении при помощи вывода P2.6, если IOC2.1 = 1. Однако следует соблюдать осторожность, когда это используется при работе с HSO. Если Таймер_2 не выполняет полный цикл, возможно наличие результатов в ассоциативной памяти, которые никогда не соответствуют таймеру. Эти результаты будут храниться в ассоциативной памяти пока ассоциативная память (CAM) не очищается или микроконтроллер не сбрасывается.

7.3 Управление Таймером _2_ по внешним выводам

С выводов T2UPDN, T2CLK, T2RST и T2CAPTURE производится выборка в течение PH1. PH1 приблизительно соответствует внешнему сигналу CLKOUT низкого уровня. Для правильной выборки данных входные сигналы должны быть поданы за 30 нс до восходящего фронта сигнала CLKOUT, иначе выборку невозможно произвести до следующего CLKOUT. Если сигнал T2UPDN изменяется и остается постоянным до или же в то же самое время, когда сигнал T2CLK меняет свое значение, то отсчет пойдет в новом направлении.

7.4 Прерывания от таймеров

Как Таймер_1 так и Таймер_2 могут запускать прерывание по переполнению таймера и устанавливать флаг в статусном регистре 1 ввода-вывода (IOS1). Переполнение Таймер_1 управляется посредством установки IOC1.2, а статус прерывания указывается в IOS1.5. Прерывание переполнения таймера (TIMER OVERFLOW) разрешается посредством установки INT_MASK.0.

Условие переполнения Таймер_2 производит прерывание через адрес 2000H посредством установки IOC1.3 и установки INT_MASK.0. Иначе переполнение Таймер_2 может вызывать прерывание через ячейку с адресом 2038H при помощи установки (в состояние "1") INT_MASK.1. Статус прерывания Таймер_2 по переполнению указывается в IOS1.4.

Прерывания могут быть сгенерированы, если Таймер_2 пересекает границу 0FFFFH/0000H или границу 7FFFFH/8000H в любом направлении. При наличии двух точек прерывания является возможным разрешение прерываний, даже если Таймер_2 производит счет в прямом и обратном направлениях относительно одной из точек прерывания. Границы, используемые для управления прерыванием Таймер_2, определяются посредством установки IOC2.5. Будучи установленным, Таймер_2 будет выдавать прерывание на границе 7FFFFH/8000H, иначе на границе 0FFFFH/0000H генерирует прерывание. Прерывание T2CAPTURE разрешается при помощи установки INT_MASK1.3. Вектор прерывания выбирается по адресу 2036H. Необходимо соблюдать осторожность при анализе флагов, так как любой доступ в IOS1 приводит к очистке битов с 0 по 5, включая программные флаги таймера. Поэтому рекомендуется скопировать байты во временный регистр до тестирования битов. Запись в IOS1 в окне 15 приведет к установке в состояние "1" битов состояния, но не приведет к прерываниям. Общее разрешение и запрещение прерываний таймеров управляются битом 0 регистра маски прерывания. Во всех случаях установка бита разрешает функционирование, в то время как очистка бита запрещает его.

8 Модуль быстрого ввода (HSI)

Модуль быстрого ввода (HSI) может регистрировать время события по отношению к Таймер_1. Имеются 4 шины (HSI.0 - HSI.3), которые могут быть использованы в этом режиме и возможна регистрация всего до 8 событий.

HSI.2 и HSI.3 являются двунаправленными выводами, которые также могут быть использованы как HSO.4 и HSO.5. Регистры управления вводом/выводом (IOC0 и IOC1) определяют функции этих выводов. Величины запрограммированные в IOC0 и IOC1 могут быть считаны в окне 15. На рисунке 8.1 дана блок-схема узла HSI.

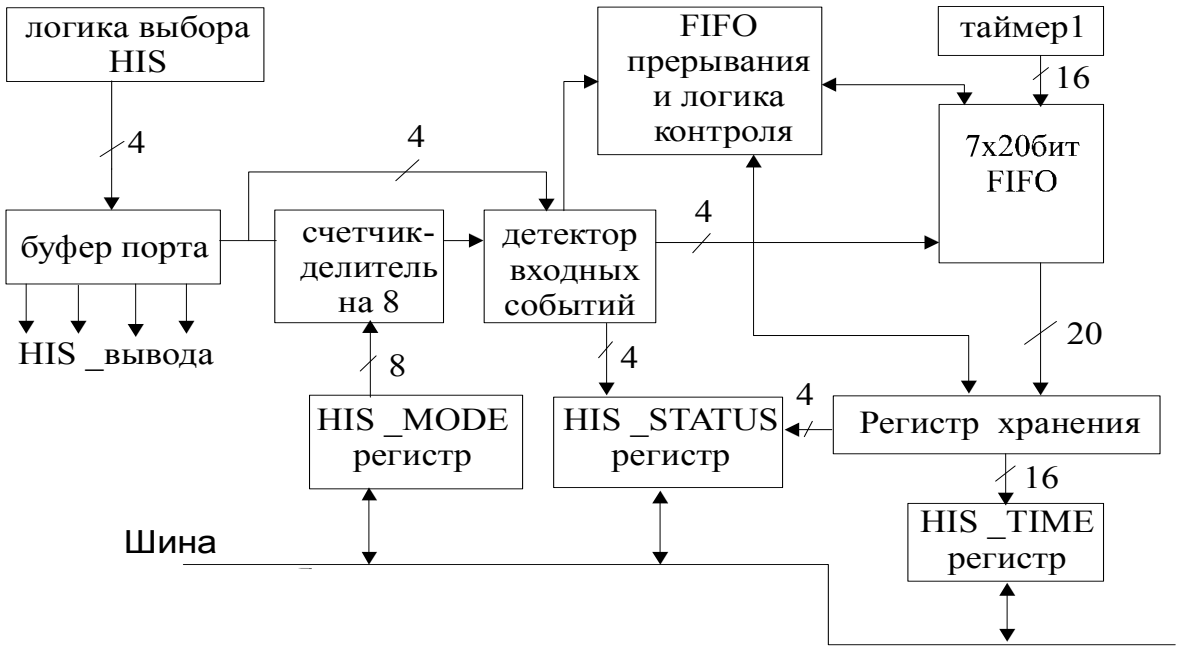
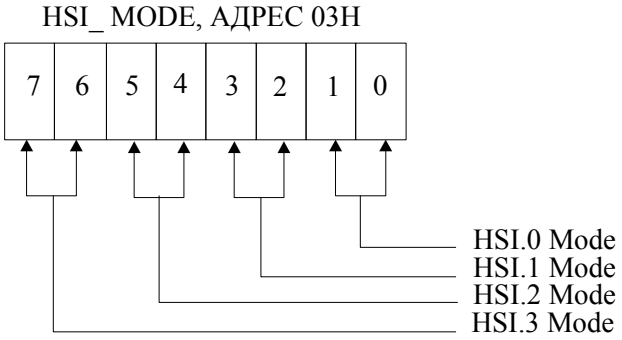


Рисунок 8.1 - Блок-схема HSI

При работе HSI 7x20 FIFO хранит 16 битов Таймер_1 и 4 бита, указывающих, какие фиксированные события на выводах связаны с этим временным признаком. Следовательно, если кратные выводы используются в качестве HSI - выводов, программа должна проверять каждый статусный бит при обработке результата HSI. На кратных выводах могут определяться исходы с одним и тем же временным признаком. Необходимо до 8 машинных циклов для того, чтобы эта информация достигла регистра хранения. По этой причине должно пройти 8 машинных циклов HSI_TIME. Когда FIFO заполнен, одно дополнительное событие для совокупного из 8 событий может хранится посредством рассмотрения, считая регистр хранения частью FIFO. Если FIFO и регистр хранения заполнен, любые дополнительные результаты не будут записаны.

8.1 Режимы работы модуля быстрого ввода

Для каждого из высокоскоростных входов существуют 4 возможных режима работы. Регистр HSI_MODE по адресу 03H определяет, какие входы будут отслеживать события некоего типа. В окне 15 считывание содержимого регистра приводит к повторному считыванию запрограммированного HSI режима. 8-разрядный регистр устанавливается как показано на рисунке 8.1.1.



Каждое двухбитовое управляющее поле определяет один из 4-х режимов:
00 - 8-х положительных фронтов
01 - каждый передний фронт
10 - каждый задний фронт
11 - каждый фронт

Рисунок 8.1.1 - Формат HSI_MODE

Максимальная скорость ввода равна 1 событию каждые 8 машинных циклов, за исключением того случая, когда используется режим 8 переходов; в этом случае существует 1 переход на машинный цикл. Выводы HSI могут быть по одиночке разблокированы и запрещены, используя биты IOC0, как показано на рисунке 8.1.2. Если вывод блокирован, операции перехода не вводятся в FIFO. Однако, входные биты регистра HSI_STATUS всегда действительны к тому, разблокирован ли вывод к FIFO. Это позволяет использовать HSI-выводы в качестве входных выводов общего назначения.

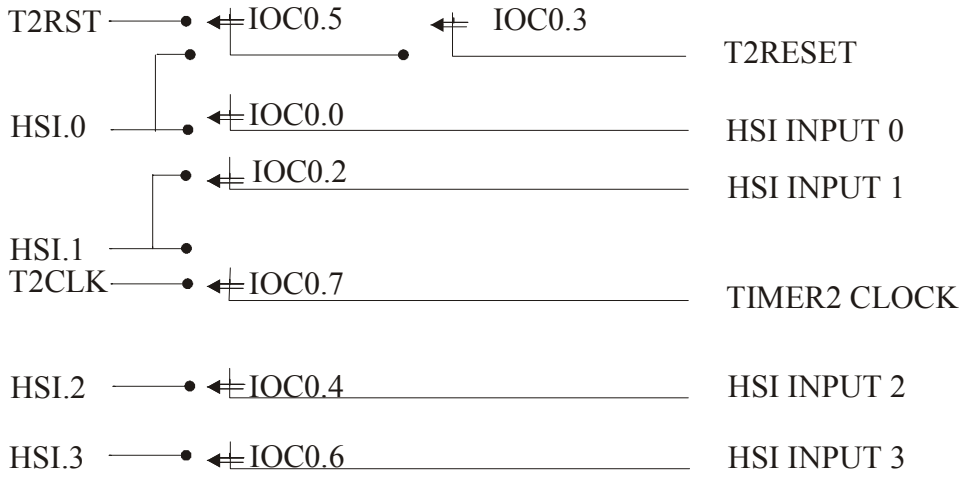


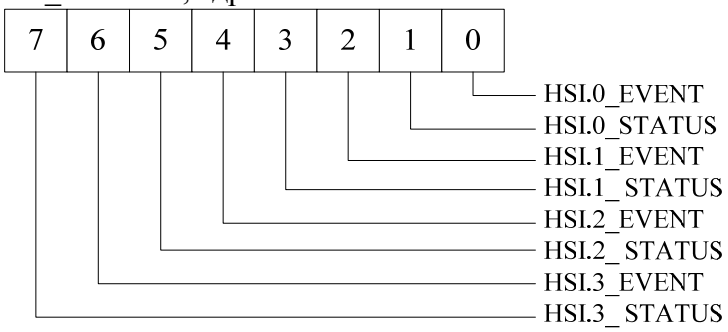
Рисунок 8.1.2 - Управление функциями выводов HSI через IOC0

8.2 Состояния модуля быстрого ввода (HSI)

Разряды 6 и 7 регистра состояния ввода/вывода 1 указывают статус HSI FIFO. Если бит установлен, загружается регистр хранения HSI. FIFO может содержать или нет 1-5 событий. Если бит 6 установлен, FIFO содержит 6 входных сообщений. Если FIFO заполняется, будущие события не будут регистрироваться. Чтение IOS1 очищает биты 0-5, так что следует хранить отображение регистра и тестировать отображение для сохранения всех 6 разрядов. Считывание содержимого регистра хранения HSI должно производиться в определенном порядке. Первым считывается статусный регистр HSI_STATUS согласно рисунка 8.2 для получения статусных и входных битов.

Затем считывается содержимое регистра HSI_TIME (04H) для получения временного признака. Считывание HSI_TIME выгружает один из уровней FIFO. Если HSI_TIME считывается до HSI_STATUS, то содержимое HSI_STATUS, связанное с этим признаком HSI_TIME теряется.

HSI_STATUS, адрес 06H



"1" в разрядах 0, 2, 4, 6 – состоялось запрограммированное событие на соответствующем выводе;
"0" в разрядах 0, 2, 4, 6 – событие не состоялось.
Запись в WSR=15 происходит только в вышеуказанные разряды регистра HSI_STATUS.
Разряды 1, 3, 5, 7 являются индикаторами текущего состояния соответствующих выводов, независимо от ранее состоявшихся событий, записанных в разряды 0, 2, 4, 6.

Рисунок 8.2 – Регистр состояния HSI

Если производится считывание из режима HSI_TIME без загрузки регистра хранения, то возвращаемое значение будет неопределяемым. При тех же самых условиях четыре бита в HSI_STATUS, указывающие какие события произошли, также будут неопределенными. Четыре бита HSI_STATUS, указывающие текущее состояние выводов, всегда будут возвращать корректное значение.

Следует отметить, что многие режимы статусного режима изменяются посредством сброса. Запись в HSI_TIME в окне 15 приведет к записи в регистр хранения, HSI_STATUS в окне 15 приведет к установке битов состояния, но не повлияет на входные биты.

8.3 Прерывание HSI

Прерывания могут генерироваться блоком HSI тремя способами: когда значение перемещается из FIFO в регистр хранения; когда FIFO хранит четыре или более событий; когда FIFO содержит шесть или более событий.

Источник для прерывания HSI DATA AVAILABLE управляется IOC1.7. Когда IOC1.7 очищается, HSI будет генерировать прерывание, когда загружается регистр хранения. Прерывание указывает, по крайней мере, одно происшедшее событие HSI и готово к обработке.

Прерывание направляется через адрес 20004H. Прерывание разрешается посредством установки INT_MASK.2. Генерирование прерывания HSI DATA AVAILABLE приведет к установке IOS1.7. Прерывание HSI FIFO FULL будет придавать направление через HSI DATA AVAILABLE, если IOC1.7 установлен. В микроконтроллерах HSI FIFO FULL имеет отдельный вектор прерываний по адресу 203CH.

Прерывание HSI FIFO FULL происходит, когда HSI FIFO имеет шесть или более входных сообщений, не зависящих от регистра хранения. Так как все прерывания запускаются по восходящему фронту, процессор не будет получать повторное прерывание, пока FIFO не будет содержать сначала пять или менее записей, затем шесть или более. Маскирующий бит прерывания HSI FIFO FULL указывается IOS1.6. Более раннее предупреждение о предстоящем заполнении FIFO может быть получено посредством четвертой записи в HSI FIFO.

Прерывание HSI_FIFO_4 генерирует прерывание, когда четыре или более событий хранятся в HSI FIFO не зависимо от регистра хранения. Прерывание разрешается посредством установки INT_MASK1.2. HSI_FIFO_4 направляется косвенным образом через адрес 2034H. Не существует статусного флага, связанного с прерыванием HSI_FIFO_4, так как оно имеет свой собственный независимый вектор прерываний.

Вывод HSI.0 может генерировать прерывание по восходящему фронту, даже если он не разрешен к HSI FIFO. Прерывание, генерированное этим выводом, обслуживается через адрес 2008H.

8.4 Выборка входных данных HSI

Производится внутренняя выборка выводов HSI один раз в течение каждого машинного цикла. Любое значение на этих выводах должно оставаться стабильным в течение одного полного машинного цикла для гарантирования его опознавания. Фактическая выборка происходит во время низкого PH1 или CLKOUT. Входы HSI должны быть действительными, по меньшей мере, за 30 нс до восходящего фронта CLKOUT. Иначе может производиться выборка входного сигнала HSI во время следующего CLKOUT. Следовательно, если информация должна быть синхронизирована с HSI, он должен защелкиваться по восходящему фронту CLKOUT.

8.5 Инициализация

Для запуска HSI должны быть соблюдены следующие шаги и последовательность:

1 Запуск FIFO.

2 Разрешение прерываний HSI.

3 Инициализация и разрешение выводов HSI.

Разблокирование выводов HSI до разрешения прерывания может вызвать состояние блокировки. Например, если HSI выводы разблокированы первыми, то событие могло бы быть загружено в регистр хранения до разрешения прерывания HSI DATA AVAILABLE. Если это имеет место, прерывания HSI DATA AVAILABLE никогда не происходит.

9 Модуль быстрого вывода

Модуль быстрого вывода записывает события в определенные моменты времени с минимальным опережением процессора. События генерируются посредством команд записи в регистр HSO_COMMAND и при помощи относительного времени, в течение которого результаты должны появится в регистре HSO_TIME.

В окне 15 эти регистры считают последнее значение, запрограммированное в регистре хранения. Программируемые события включают в себя: запуск АЦП - преобразования; сброс Таймер_2; установку 4 программных флагов и переключение 6 выходных каналов (HSO.0 до HSO.5).

Формат регистра HSO_COMMAND показан на рисунке 9.1. Команды 0CP и 0DH зарезервированы для будущего использования. До восьми событий могут быть ожидаемы в одно время и прерывания могут генерироваться не зависимо от того, когда эти события запускаются. Выходы HSO.4 и HSO.5 коммутируются с входами HSI.2 и HSI.3 соответственно. Биты 4 и 6 регистра управления вводом/выводом разрешают HSO.4 и HSO.5 в качестве выходов. Содержимое регистров управления может быть считано в окне 15 для определения программируемых режимов HSO. Однако, бит IOC2.7 (CAM CLEAR) не защелкивается и будет считываться. Введенные данные могут заперты в ассоциативной памяти для генерирования периодических событий или форм сигналов.

HSO_COMMAND

7	6	5	4	3	2	1	0
CAM LOCK	TMR2/ TMR1#	SET/ CLEAR#	INT/ INT#	CHANNEL			

06H

CAM LOCK : Загрузка события в CAM по разрешению IOC2/6 (ENA_LOCK)
TMR2/TMR1#: Событие основано на Таймер_2/Таймер_1, если 0
SET/CLEAR# : Установка вывода HSO/очистка вывода HSO, если 0
INT/INT# : Вызов прерывания/нет прерывания, если 0
CHANNEL: 0-5: 0-5 выводы HSO по отдельности
(in Hex)
6: Выводы 0 и 1 HSO вместе
7: Выводы 2 и 3 HSO объединенные
8-B: Программные Таймеры_0-3
C-D: Не используется
E: Сброс Таймер_2
F: Старт АЦП

Рисунок 9.1 – Формат регистра HSO_COMMAND

9.1 Прерывания модуля HSO и программные таймеры

Модуль HSO может генерировать два типа прерываний. Прерывание HSO может генерироваться для HSO команд, которые изменяют один или более из шести выходных выводов. Другое HSO-прерывание является прерыванием, которое может генерироваться любой другой HSO-командой: старт АЦП, сброс Таймер_2 или генерированием программной временной задержки.

Статус HSO - прерываний

Регистр IOS2 по адресу 17H отображает происшедшие HSO-события. IOS2 показан на рисунке 9.1.1. Отображаемые события: с HSO.0 до HSO.5. Сброс Таймер_2 и начало АЦП - прерывания. При доступе IOS2 его содержимое очищается, следовательно содержимое регистра должно быть сохранено в регистре отображения, если проверяется более одного бита. Регистр состояния полезен для определения какие события вызвали генерированное HSO прерывание. Операция записи в этот регистр в окне 15 приведет к установке статусных битов, но не вызовет прерываний. В окне 15 запись в IOS2 может установить линии высокоскоростного вывода в начальное значение. Более подробнее информация, касающаяся окна 15, дана в подразделе 2.7.2.

IOS2:

7	6	5	4	3	2	1	0
START A/D	T2 RESET	HS0.5	HS0.4	HS0.3	HS0.2	HS0.1	HS0.0

17H
Чтение

START A/D: HSO_CMD 15, start A/D
T2RESET: HSO_CMD 14, Timer2 Reset
HSO.0-5: Выводы HSO.0 до HSO.5

Рисунок 9.1.1 - Регистр состояния ввода/вывода 2 (IOS2)

Программируемые таймеры

HSO может быть запрограммирован на генерирование прерываний в данные моменты времени. До четырех таких "программных таймеров" могут функционировать в одно время. Когда достигается каждое предварительно запрограммированное время, блок HSO устанавливает программный флаг таймера. Если бит прерывания в регистре команд HSO установлен, то программное прерывание таймера также будет генерироваться. Программа обслуживания прерываний может затем рассмотреть содержимое статусного регистра ввода/вывода1 для определения того, какой из программируемых таймеров закончил работу и вызвал прерывание. Когда HSO сбрасывает Таймер_2 или запускает АЦП - прерывания, он также может быть запрограммирован на генерацию программного таймера.

Если происходит более одного программного прерывания таймера в течение одного и того же выделенного интервала времени, будут установлены кратные разряды индикации состояния. Любое чтение или тестирование приведет к очистке битов с 0 до 5. Убедитесь в сохранении байта до тестирования его, если только вы не рассматриваете 1 бит.

9.2 Ассоциативное ЗУ модуля высокоскоростного вывода (HSO)

Блок-схема модуля высокоскоростного вывода (HSO) дана на рисунке 4.4.1.Комплект ассоциативного ЗУ является центром управления. Содержимое одного регистра ассоциативного ЗУ сравнивается с значениями таймера в течение каждого машинного цикла, что занимает 8 машинных циклов для сличения всех регистров ассоциативного ЗУ с таймерами. Это определяет время разрешения HSO, равное 8 машинным циклам.

Каждый регистр ассоциативного ЗУ является 24-разрядным. 16 разрядов определяют время, в течение которого выполняется операция, один разряд отведен для байта запирания и 7 разрядов определяют как природу операции, так и то, является ли Таймер_1 или Таймер_2 ссылкой. Формат команд HSO показан на рисунке 9.1. Заметим, что 5 бит игнорируется для командных каналов от 8 до OFH.

Для ввода команды в комплект ассоциативного ЗУ производится запись 8-разрядного "признака команды" в ячейку с адресом 0006H перед моментом времени, когда операция должна быть перенесена в адрес слова 0004H.

Запись временных значений загружает в регистр хранения HSO как временной признак, так и последний записанный командный признак. Команда фактически не записывает файл ассоциативного ЗУ пока пустой регистр ассоциативного ЗУ не станет доступным. Команды, находящиеся в регистре хранения, не будут выполняться, даже если достигается их временной признак. Для того, чтобы выполняться, команды должны находиться в ассоциативном ЗУ. Команды в регистре хранения также могут быть перезаписаны. Т.к. может потребоваться до 8 машинных циклов для пересылки команды из регистра хранения в ассоциативное ЗУ, должны быть предоставлены между последовательными операциями записи в ассоциативном ЗУ.

Для определения правильной синхронизации минимальное время, которое должно быть загружено в Таймер_1, равно Таймер_1 + 2. Меньшие значения могут привести к согласованию Таймер на 65, 636 единичных импульсов счета позже ожидаемого. При использовании Таймер_2 применимо подобное ограничение.

При записи каждого признака необходимо соблюдать осторожность, потому что прерывание может произойти между записью командного признака и загрузкой временного значения. Если программа обслуживания прерываний производит операцию записи в HSO, то командный признак, используемый в программе обработки прерываний, перезапишет признак команды из основной программы. Один из способов избежать эту проблему состоит в блокировке прерываний при записи в блок HSO.

9.3 Состояния HSO

Перед записью в HSO желательно убедиться, что регистр-защелка (пара HSO_COMMAND - HSO_TIME) пуст. Если это не так, запись в HSO будет переписывать данные в регистре-защелке. Биты 6 и 7 в IOS0 показывают статус HSO. Если IOS0.6 равен 0, регистр-защелка пуст и хотя бы один регистр CAM свободен.

Если IOS0.7 равен 0, регистр-защелка пуст. Программист должен определить, какой из этих флагов более пригоден для решения его задачи. Этот регистр также показывает текущее состояние HSO.0 до HSO.5. Выводы HSO могут быть установлены при помощи записи в этот регистр в окне 15. Формат статусного регистра ввода/вывода 0 показан на рисунке 9.3.1.

15 H

0	текущее состояние HSO.0
1	текущее состояние HSO.1
2	текущее состояние HSO.2
3	текущее состояние HSO.3
4	текущее состояние HSO.4
5	текущее состояние HSO.5
6	CAM или регистр хранения заполнены
7	HSO регистр хранения заполнен

Рисунок 9.3.1 - Регистр состояния IOS.0

Конец счета программного таймера с 0 по 4 и переполнение Таймер_1 и Таймер_2 указываются в IOS1. Биты состояния могут быть установлены в окне 15, но не вызывают прерываний. Регистр показан на рисунке 9.3.2.

16H

0	программный Таймер 0 выключен
1	программный Таймер 1 выключен
2	программный Таймер 2 выключен
3	программный Таймер 3 выключен
4	Таймер 2 переполнен
5	Таймер 1 переполнен
6	HSI FIFO заполнен
7	регистр хранения HSI доступен

Рисунок 9.3.2 - Первый регистр состояния (IOS 1)

Когда бы процессор ни считывал содержание этого регистра, все связанные по времени флаги (биты с 5 до 0) очищаются. Это относится не только к явным считываниям, например LDB AL, IOS1 но также к неявным считываниям, таким как JBS IOS1,3, somewhere - else которые очищают переход к somewhere - else, если бит 3 IOS1 установлен. В большинстве случаев эта ситуация может быть наилучшим образом обработана посредством наличия байта в регистровом файле, который поддерживает представление регистра. В любой момент времени, когда происходит аппаратное прерывание таймера или HSO - программное прерывание таймера, содержимое разряда может быть скорректировано: ORB IOS1_image, IOS1 оставляя в IOS1_image в качестве содержимого все флаги, которые были установлены ранее плюс все новые флаги, считанные и очищенные из IOS1. Любая другая программа, которая должна выполнять дискретную выборку флагов, может безопасно проверять IOS1_image. Заметим, что если эти программы должны очищать флаги, на которые они повлияли, то модификация IOS1_image должна быть произведена изнутри критической области.

9.4 Очистка HSO и блокировка введенных данных

Все 8 позиций CAM HSO проверяются перед любым действием. Это дает возможность останавливать внешние действия HSO простой записью команды с противоположным действием в CAM. Однако, если команда + время уже размещена в CAM, она не может быть удалена, пока специфицированное время не совпадает со значением регистра таймера либо пока не будет произведен аппаратный сброс или установлен IOC2.7. IOC2.7 является битом очистки CAM, который очищает все введенные в ассоциативное ЗУ данные.

Внутренние события не могут быть очищены посредством записи противоположного события. Это включает события на каналах HSO с 8 по F. Единственный способ очистки этих событий состоит в сбросе или установке IOC2.7.

Блокирование введенных данных HSO

Запирающий бит ассоциативного ЗУ может быть установлен для хранения команд в ассоциативном ЗУ (CAM), в противном случае команды произведут очистку CAM, как только они вызывают событие. Эта характеристика принимает в расчет генерирование периодических событий на основе Таймер_2 и может быть разрешена посредством установки IOC2.6. Для очистки блокированных событий из CAM вся схема может быть очищена посредством записи единицы в бит очистки CAM IOC2.7. Сброс схемы также произведет очистку CAM.

Блокированные входные данные используются в приложениях, требующих появления периодических или повторяющихся событий.

Таймер_2, используемый для обращения к HSO, может генерировать периодические события с использованием команды T2RST HSO. События HSO, запрограммированные с моментом времени, меньшим, чем время сброса Таймер_2 будут происходить повторно при сбросе 1 Таймер_2. Рекуррентные программные задачи могут распределены посредством блокировки команд программных таймеров в блоке высокоскоростного вывода. Непрерывный выбор АЦП - может выполняться посредством программирования блокировки HSO-команды АЦП. Одной из наиболее полезных характеристик является генерирование множественных PWM на линиях высокоскоростного вывода. Запертые входные данные обеспечивают возможность программирования периодических событий при минимизации. В подразделе 9.6 описано генерирование четырех PWM с использованием блокированных входных данных.

Отдельные внешние события, устанавливающие или очищающие HSO-вывод, могут быть отменены посредством записи противоположного события в CAM. HSO-событие не происходят пока обращение к таймеру не изменит состояние. Событие, запрограммированное на установку и очистку HSO-события, в то же самое время запретят любой другой вывод. Запертые входные данные могут быть соответственно отменены используя этот метод. Однако, введенные данные остаются в HSO CAM и могут быстро заполнить 8 доступных ячеек ЗУ. В качестве альтернативы все введенные данные в HSO CAM могут быть очищены посредством установки IOC2.7.

9.5 Особенности программирования HSO

Таймер_1 инкрементируется всегда 1 раз на каждые 8 машинных циклов. Когда он используется, как ссылочный при работе с HSO, компаратор имеет возможность проверить все 8 регистров CAM перед тем, как Таймер_1 изменяет свое значение. По этой же причине Таймер_2 должен быть синхронизирован не более часто, чем 1 раз на 8 машинных циклов. Запись в Таймер_1, которая разрешена в окне 15, должна производиться с осторожностью. Пользователь должен обеспечить, чтобы запись в Таймер_1 не привела к пропуску запрограммированных событий или появлению в неверной последовательности. Те же самые меры предосторожности относятся к Таймер_2.

HSO требует по меньшей мере 8 машинных циклов для сравнения каждого введенного сообщения в CAM. Следовательно, режим быстрого приращения для Таймер_2 не может быть использован в качестве обращения для HSO, если переходы происходят быстрее, чем один раз каждые 8 машинных циклов.

Обращение к событиям, когда Таймер_2 используется в качестве реверсивного счетчика, могло привести к появлению событий в противоположном порядке или их полному пропуску. В добавлении к этому, блокированные входные данные могут, возможно, присутствовать несколько раз, если Таймер_2 осциллирует около временного признака для входных данных.

При использовании Таймер_2 в качестве ссылки HSO необходимо соблюдать осторожность, чтобы Таймер_2 не произвел сброс прежде наибольшего значения для совпадения Таймер_2 в CAM. Если это совпадение никогда не происходит, событие будет оставаться незаконченным в CAM до тех пор, пока часть не сброшена или CAM не очищен.

9.6 PWM с использованием HSO

Блок HSO может генерировать PWM - сигналы с небольшим "опережением" процессора, используя Таймер_2 для обращения. PWM генерируется посредством программирования HSO - шины в высокое состояние и появления T2RST в тоже время. Малое время HSO программируется в CAM для генерации рабочего цикла PWM.

Повторный сигнал PWM генерируется посредством запирания команд в CAM. Перепрограммирование рабочего цикла или частоты PWM может быть выполнено при помощи генерирования программного прерывания и перепрограммирования высокоуровневых HSO, низкоуровневых HSO- и T2RSN-команд. Мгновенные PWM могут программироваться, используя Таймер_2 для обращения и блокирования входных сообщений CAM. До четырех PWM могут генерироваться посредством блокирования PWM (высокий) и PWM (низкий) в CAM для каждого HSO.0 - HSO.3. Таймер_2 используется для обращения и устанавливается в нуль посредством программирования T2RST- команды в то же самое время, когда HSO-команда устанавливает все шины на высокий уровень. Два входных сообщения CAM программируют четыре PWM (высокий) момента времени посредством установки HSO.0/HSO.1 и HSO.2/HSO.3 в высокое состояние той же самой командой. Четыре входных сообщения в CAM устанавливают каждую из шин HSO в низкое состояние. Одно входное сообщение используется для сброса Таймер_2. Этот метод использует общее из семи входных данных CAM с небольшим "опережением" математического обеспечения или без такого. PWM могут изменять свой рабочий цикл посредством перепрограммирования CAM с различными HSO-уровнями.

Изменение рабочего цикла для каждого PWM требует подавление ненужной информации CAM и перепрограммирование всех семи введенных сообщений CAM. ИС могут подавлять ненужную информацию во всем CAM посредством установки бита 7 в регистре IOC2 (ячейка 16).

Каждый из моментов времени HSO (высокий) и HSO (низкий) должны перепрограммироваться дополнительно к команде сброса Таймер_2. Этот метод дает до четырех PWM без программного "опережения" (software overhead), за исключением того случая, когда производится перепрограммирование рабочего цикла любого отдельного PWM.

9.7 Тактирование выходных сигналов HSO

Изменения на шинах HSO синхронизируются либо с Таймер_1, либо с Таймер_2. Все внешние шины HSO, которые должны изменяться при определенном значении таймера, будут изменяться сразу после приращения таймера. Внутренне таймер изменяется каждые восемь машинных циклов в течение PH1. Вывод HSO должен изменяться именно до ниспадающего фронта сигнала CLKOUT и быть стабильным по его восходящему фронту. Выходящие из HSO данные могут защелкиваться по восходящему фронту сигнала CLKOUT. Внутренние события также происходят, когда происходит приращение таймера.

10 Последовательный порт

Последовательный порт ИС имеет три асинхронных и один синхронный режим. Асинхронные режимы полностью дуплексные, что означает, что передача и прием могут происходить одновременно. Приемник дважды буферизирован, поэтому прием второго байта может начинаться до того, как первый байт будет считан. Передатчик на ИС также обладает двойной буферизацией, разрешая непрерывные передачи. Порт функционально совместим с последовательным портом микроконтроллеров семейства микросхем серии 1830, хотя программное обеспечение, используемое для управления портами, отличается.

SBUF(TX) удерживает данные, готовые для передачи, и SBUF(RX) содержит информацию, полученную последовательным портом. SBUF(TX) и SBUF(RX) могут читаться и записываться в окне 15.

Режим 0, режим синхронного сдвига регистра, предназначен для расширения ввода-вывода через последовательную шину. Режим 1 является стандартным 8-битовым асинхронным режимом, используемым для стандартных передач данных через последовательный порт. Режимы 2 и 3 являются 9-битовыми асинхронными режимами данных, которые в основном используются для межпроцессорных связей. Режим 2 обеспечивает текущий контроль линии связи для 1 в девятом двоичном разряде до вызывания прерывания. Режим 3 вызывает прерывания, независящие от значения девятого бита.

10.1 Состояние и управление последовательного порта

Контроль над работой последовательного порта осуществляется через регистр управления последовательным портом (Serial Port Control, SP_CON) и регистр статуса последовательного порта (Serial Port Status, SP_STAT).

SP_CON:

7	6	5	4	3	2	1	0	
X	X	X	TB8	REN	PEN	M2	M1	11H

TB8 – Устанавливает девятый бит данных для передачи. Очищается после каждой передачи. Если четность разрешена – не действителен.
REN – Разрешает прием.
PEN – Разрешает функцию четности.
M2, M1 – Устанавливают режим. Режим 0 = 00, режим 1 = 01, режим 2 = 10, режим 3 = 11.

SP_STAT

7	6	5	4	3	2	1	0	
RB8/RPE	RI	TI	FE	TXE	OE	X	X	11H

RB8 – Устанавливается, если девятый бит данных высокий на приеме (проверка на четность запрещена).
RPE – Устанавливается, если четность разрешена и встречается ошибка четности.
RI – Устанавливается после приема последнего бита данных.
TI – Устанавливается в начале передачи стопового бита.
FE – Устанавливается, если не найден стоповый бит в конце приема.
TXE – Устанавливается, если два байта могут быть переданы.
OE – Устанавливается, если переполнен буфер приема.

Верхние три разряда SP_CON должны быть записаны как 0 для будущей совместимости. В ИС регистр SP_CON содержит новые биты для индикации получения ошибки переполнения (OE), ошибки кадрирования (FE) и незанятого передатчика (Transmitter Empty) (TXE). SP_CON предназначен только для чтения в окне 0 .

Во всех режимах RI флаг устанавливается после того, как проведена выборка последнего бита данных, приблизительно в середине времени прохождения бита. Данные удерживаются в приемном сдвиговом регистре до тех пор, пока не получен последний информационный разряд, затем информационный байт загружается в SBUF(RX). Приемник ИС также осуществляет проверку правильного бита останова. Если стоповый бит не найден в течение соответствующего времени, то устанавливается бит ошибки кадрирования (FE).

Т.к. приемник имеет двойную буферизацию, прием на втором байте данных может начаться до считывания первого байта. Однако если данные в сдвиговом регистре загружаются в SBUF(RX) до считывания предшествующего байта, устанавливается бит ошибки переполнения (OE). Несмотря на это данные в SBUF(RX) всегда будут последним полученным байтом; они никогда не будут комбинацией двух байтов. Флаги FI, FE и OE очищаются при чтении SP_STAT. Однако RI не обязан очищаться для получения данных последовательным портом. Бит очищенного передатчика (TXE) устанавливается, если буфер передачи пуст и готов к принятию двух знаков TXE; очищается, как только записывается байт в SBUF. Два байта могут быть записаны последовательно, если TXE установлен. Один байт может записываться, если установлен только TI. По определению, если TXE только что установлен, передача завершена и TI будет установлен. Бит TI сбрасывается, когда процессор (CPU) производит считывание регистров SP_STAT. Бит TB8 очищается после каждой передачи, и очищаются как TI, так и RI при чтении SP_STAT. RI и TI статусные биты могут устанавливаться посредством записи в SP_STAT в окне 15, но они не вызовут прерывания. Чтение SP_COM в окне 15 приведет к считыванию последнего записанного значения. Когда бы TXD-вывод не использовался для последовательного порта, он должен быть разрешен посредством установки IOC1.5 в 1. Регистр управления вводом-выводом может считываться в окне 15 для определения установки.

Запуск передачи и приема

В режиме 0, если PEN=1, запись в SBUF(tx) вызовет начало передачи. Передний фронт PEN или очистка RI, когда PEN=1, вызовут прием. Установка PEN=0 остановит текущий прием и запретит дальнейший прием. Для предотвращения частичного или полного нежелательного приема PEN должен быть сброшен до того, как RI будет очищен. Это может быть выполнено подпрограммой прерывания при использовании программных флагов либо в основной программе посредством использования регистра INT_PEND как сигнализатора завершения приема.

В асинхронных режимах запись в SBUF(tx) начнет передачу. Задний фронт RXD начнет прием, если PEN=1. Новые данные, размещенные в SBUF(tx), удерживаются и не передаются, пока конец стопового бита не будет послан.

Во всех режимах флаг RI устанавливается после того, как последний бит данных принимается (примерно в середине длительности бита). Также во всех режимах флаг TI устанавливается после того, как последний бит данных (8-й или 9-й) послан, также в середине длительности бита. Флаги очищаются, когда читается SP_STAT, но не очищаются, когда порт принимает или передает данные.

Запрос на прерывание от последовательного порта устанавливается как логическое "или" между битами TI и RI. Следует отметить, что изменение режима будет сбрасывать последовательный порт и прекращать любую текущую передачу или прием.

Скорость передачи в бодах

Скорости передачи в бодах генерируются на основе либо вывода T2CLK, либо вывода XTAL1. Скорости передачи в бодах вычисляются, используя следующие формулы, где BAUD_REG есть величина, загруженная в регистр скорости передачи:

асинхронные режимы 1, 2 и 3:

$$BAUD\_REG = \frac{XTAL1}{Скорость\ передачи * 16} - 1 \quad \text{или} \quad \frac{T2CLK}{Скорость\ передачи * 8}$$

синхронный режим 0:

$$BAUD\_REG = \frac{XTAL1}{Скорость\ передачи * 2} - 1 \quad \text{или} \quad \frac{T2CLK}{Скорость\ передачи} .$$

Самый старший разряд в бодовом регистре устанавливается в 1 для выбора XTAL1 в качестве источника. Если он равен 0, вывод T2CLK становится источником. В таблице 10.1.1 даны некоторые основные значения скорости передачи.

Таблица 10.1.1 – Скорость передачи и содержимое регистра скорости передачи

Скорость передачи	Частота на XTAL1		
	8,0 МГц	10,0 МГц	12,0 МГц
300	1666/–0,02	2082/0,02	2499/0,00
1200	416/–0,08	520/–0,03	624/0,00
2400	207/0,16	259/0,16	312/–0,16
4800	103/0,16	129/0,16	155/0,16
9600	51/0,16	64/0,16	77/0,16
19,2K	25/0,16	32/1,40	38/0,16

Содержимое регистра скорости передачи / % ошибка

В асинхронных режимах при частоте 12 МГц на XTAL1 наибольшая скорость передачи в 750 Кбайт. Синхронный режим имеет максимальную скорость 3,0 Мбод при частоте тактирующего сигнала 12 МГц. Ячейка 0EH находится в бодовом регистре. Он загружается последовательно в два байта, причем младший байт загружается первым. Этот регистр не может загружаться нулем в режиме 0 последовательного порта.

10.2 Прерывания последовательного порта

Последовательный порт генерирует одно из трех возможных прерываний: прерывания передачи TI(2030H), прерывания приема RI(2032H) и SERIAL(200CH). Когда бит RI установлен, генерируется прерывание через ячейку 200CH, либо 2032H, в зависимости от которой разрешается прерывание. INT_MASK1.1 управляет прерыванием приема последовательного порта через адрес 2032H и INT_MASK.6 управляет прерываниями последовательного порта через адрес 200CH.

Когда бит TI установлен, он может вызвать прерывание через векторы по адресам 200CH или 2030. Прерывание через 2030 определяется INT_MASK1.0.

Прерывания через последовательное прерывание программируются тем же самым битом, что и RI прерывание (INT_MASK.6). Пользователь не должен маскировать прерывание последовательного порта при использовании свойства двойной буферизации передатчика, т.к. это может привести к пропаданию одиночного импульса счета в числе передаваемых байтов.

10.3 Режимы последовательного порта

РЕЖИМ 0 (Mode 0)

Режим 0 – это синхронный режим, обычно используемый для расширения ввода-вывода с помощью регистра сдвига. В этом режиме TXD выдает набор из восьми импульсов, тогда как RXD служит для передачи и приема данных. Данные передаются по восемь бит, начиная с младшего бита. Следует отметить, что это единственный режим, в котором RXD используется как выход.

Временные соотношения режима 0

При работе в режиме 0 вывод TXD выдает последовательность тактирующих импульсов, пока вывод RXD передает или принимает данные. На рисунке 10.3.1 даны формы сигнала и временные соотношения. В этом режиме последовательный порт расширяет возможность ввода-вывода ИС просто добавлением сдвиговых регистров.

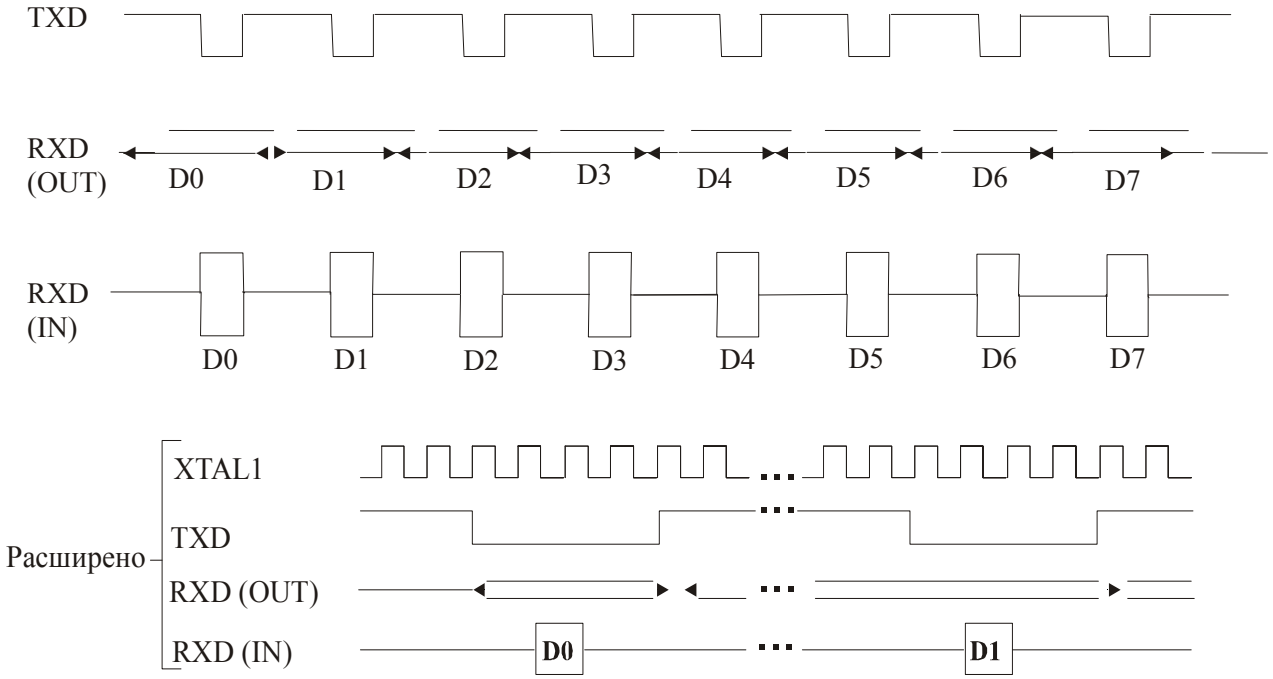


Рисунок 10.3.1 – Формы сигнала и временные соотношения

РЕЖИМ 1 (Mode 1)

Это стандартный асинхронный режим. Посылка содержит 10 бит: стартовый бит (0), 8 бит данных (младший – первый), стоповый (1). Если четность разрешена (бит PEN установлен в 1), то бит четности посылается за восьмым битом данных, и четность проверяется на приеме. Функции передачи и приема управляются разными сигналами. Синхроимпульс передачи запускается, когда инициализируется генератор скорости передачи; генератор тактовых импульсов приема сбрасывается, когда возникает переход из 1 в 0. Следовательно, тактовый импульс передачи может быть не синхронизирован с тактовым импульсом приема, хотя оба они будут работать на одной частоте.

Флаги прерывания передачи (TI) и прерывания приема (RI) устанавливаются для указания завершения операций. TI устанавливается, когда послан последний информационный бит сообщения, а не стоповый бит. Если произведена попытка переслать другой байт до посылки стопового бита, порт будет задерживать передачу, пока не завершен стоповый бит. RI устанавливается, когда получены восемь информационных битов, но не в том случае когда получен стоповый бит. Отметим, что когда производится чтение из регистра состояния последовательного порта, очищаются как TI так и RI.

Следует соблюдать осторожность при использовании последовательного порта для соединения более двух устройств полудуплексным способом. Если принимающий процессор не ждет в течении времени прохождения одного бита после установки RI, до начала передачи, то стоповый бит в линии связи может быть искажен. Это может вызвать сложности в работе других устройств, "прослушивающих" этот канал.

РЕЖИМ 2 (Mode 2)

Режим 2 - это асинхронный режим с опознаванием 9-го бита. Обычно он используется совместно с режимом 3 для мультипроцессорных коммуникаций. Посылка содержит: стартовый бит (0), 9 бит данных (младший - первый), стоповый бит (1). При передаче девятый бит может быть установлен в 1, через установку бита TB8 в управляющем регистре перед записью в SBUF(tx). Бит TB8 очищается при каждой передаче, поэтому он должен быть установлен перед записью в SBUF(tx) каждый раз, когда это требуется. При приеме прерывание последовательного порта и прерывание приема (RI) не будут устанавливаться, если девятый принятый бит не будет установлен. Это обеспечивает простейший и удобный путь к получению селективной восприимчивости к каналу связи. Контроль четности в этом режиме невозможен.

РЕЖИМ 3 (Mode 3)

Это асинхронный 9-ти-битный режим. Посылка данных идентична посылке для режима 2. Отличие заключается в том, что при передаче бит четности может быть разрешен (PEN=1) и тогда 9-й бит будет принимать значение бита четности. Бит TB8 может использоваться, если бит четности запрещен (PEN=0). В режиме 3 прием данных будет всегда вызывать прерывание вне зависимости от состояния 9-ого бита. Если PEN=0, то 9-й бит запоминается и может быть прочитан из RB8. Если PEN=1, то RB8 станет флагом ошибки четности на приеме (RPE).

Временные соотношения режимов 2 и 3

Режимы 2 и 3 работают аналогично режиму 1, см. рисунок 10.3.2. Единственное отличие состоит в том, что данные теперь составляются из 9 битов, так что передаются и принимаются 11 битовые пакеты. Это означает, что TI и RI будут устанавливаться по 9-му информационному биту, а не по 8-му. 9-ый бит может использоваться для контроля четности или для связи между множеством процессоров.

Рисунок 10.3.2 - Работа последовательного порта в режимах 1,2 и 3

10.4 Многопроцессорная связь

Режимы 2 и 3 предназначены для многопроцессорной связи. В режиме 2, если полученный 9-ый бит информации равен нулю, RI-бит не устанавливается и не вызовет прерывания. В режиме 3 RI-бит устанавливается всегда и вызывает прерывание, невзирая на значение 9-го бита. Ниже описан способ использования этого режима в многопроцессорных системах.

Ведущий процессор устанавливается в режим 3, так что он всегда откликается на прерывания из последовательного порта. Ведомые процессоры устанавливаются в режим 2, так что они имеют дело только с прерываниями приема, если установлен 9-ый бит. Используются два типа групп данных: адресные группы данных с установленным 9-ым битом и информационные блоки с очищенным 9-ым битом.

Когда главный процессор передает блок данных одному из нескольких подчиненных процессоров, он, во-первых, передает адрес, который определяет приемный процессор. Передача адреса отличается от передачи данных тем, что при передаче адреса 9-й бит равен 1, а при передаче данных он равен 0. Ни один из подчиненных процессоров не будет прерван байтом данных, если они находятся в режиме 2. Адресный байт будет прерывать все подчиненные процессоры, чтобы они могли определить, адресованы ли они главным процессором. Адресованный процессор переключается на режим 3 и принимает данные, пока не адресованные находятся в режиме 2 и занимаются своим делом.

11 Аналого-цифровой преобразователь (АЦП)

АЦП микроконтроллера (схему см. рисунок 11.1) содержит восьмиканальный аналоговый мультиплексор, устройство выборки-хранения (УВХ), и собственно 10-разрядный АЦП. Можно выбрать любой из 8-ми входов, удержать входное напряжение при помощи УВХ, и преобразовать напряжение в цифровое значение. Каждое преобразование занимает 26,33 мкс, включая время, необходимое для УВХ (при XTAL = 12 МГц). Метод преобразования - последовательное приближение.

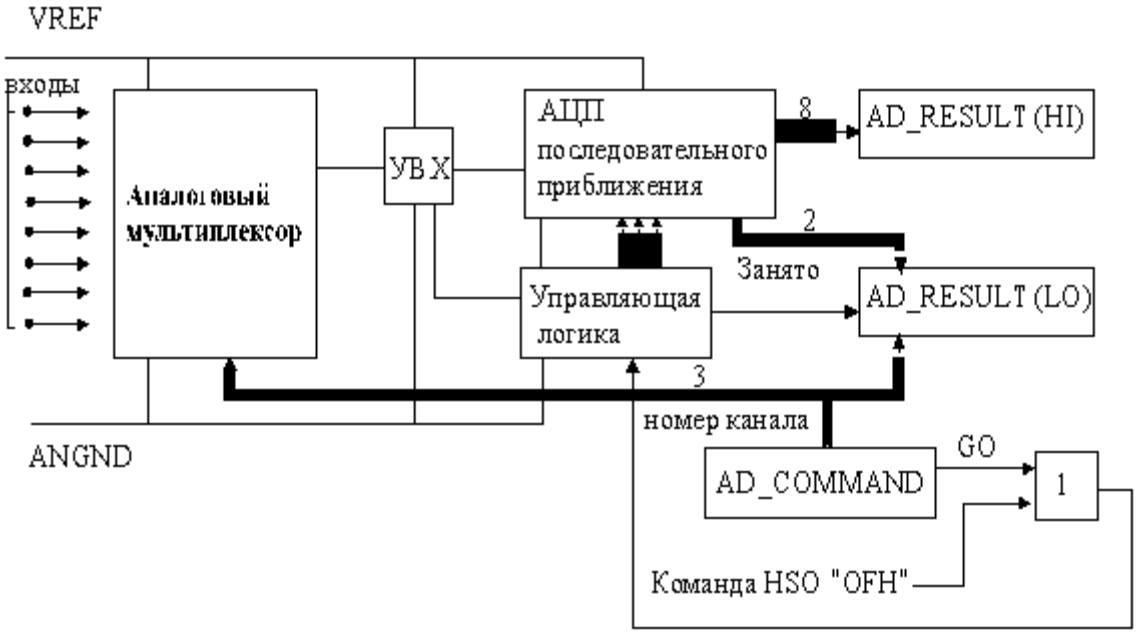


Рисунок 11.1 - Схема АЦП

11.1 Алгоритм работы АЦП

Процесс преобразования запускается исполнением команды 0FH HSO или записью бита GO в управляющий регистр АЦП. Если используется команда HSO, то процесс преобразования начинается при инкрементировании Таймера 1. Это помогает получить "спектрально чистое" преобразование - возможная ошибка по времени не превышает ± 50 нс (при стабильном XTAL1). Преобразование, запускаемое командой GO в AD_COMMAND, будет стартовать в течение трех машинных циклов после завершения инструкции, поэтому рассогласование по времени может достигать 0,50 мкс (при XTAL1 = 12 МГц). Когда АЦП получает сигнал старта, он выдерживает время - 1 машинный цикл - перед выборкой (задержка выборки), в течение которого сбрасывается регистр последовательного приближения и выбирается требуемый канал мультиплексора. После задержки выборки, выход мультиплексора соединяется с запоминающим конденсатором и остается в таком состоянии 4 машинных цикла (время выборки). После этого "окно выборки" закрывается, запоминающий конденсатор отключается от мультиплексора, поэтому дальнейшие изменения входного сигнала не будут влиять на заряд конденсатора при преобразовании. Затем компаратор обнуляется и начинается преобразование. Разброс времени задержки выборки и времени выборки для каждого из них составляет примерно ±50 нс, независимо от тактовой частоты. Для исполнения собственно АЦ - преобразования реализован алгоритм последовательного приближения. Аппаратура АЦП содержит 256-резисторную матрицу, компаратор, делители напряжения, 10-битный регистр последовательного приближения (Successive Approximation Register, SAR) с логикой управления. Резисторная матрица обеспечивает шаг 20 мВ

(при U_{REF} = 5,12 В), тогда как делители напряжения используются для создания 5-милливольтовых ступеней. Таким образом, для сравнения напряжений имеется 1024 напряжения, что позволяет генерировать 10-битный результат. Алгоритм последовательного приближения выполняется посредством сравнения опорных напряжений с напряжением аналогового входа, с последовательным двоичным поиском опорного напряжения, наиболее точно совпадающего с входным. Сперва проверяется напряжение 1/2 полной шкалы. Если напряжение на входе меньше опорного, старший бит 10-битного результата сбрасывается, и проверяется напряжение 1/4 полной шкалы.

Если напряжение на входе больше опорного, 9-й бит SAR устанавливается. Такой двоичный поиск производится 10 раз, результатом его является 10-битное число в SAR, которое может быть считано программным обеспечением. Общее число машинных циклов - 158 (XTAL1 = 12МГц) на 10-битное преобразование. Попытки "укоротить" процесс преобразования за счет раннего считывания AD_RESULT (до установления бита статуса в 0) не рекомендуются.

11.2 Рекомендации по подключению АЦП

Цепи, подключаемые к аналоговому входу, зависят от выполняемой задачи и могут оказывать влияние на характеристики АЦП. При разработке внешних цепей необходимо учитывать такие параметры, как входной ток утечки, емкость запоминающего конденсатора, сопротивление канала мультиплексора (рисунок 11.2.1).

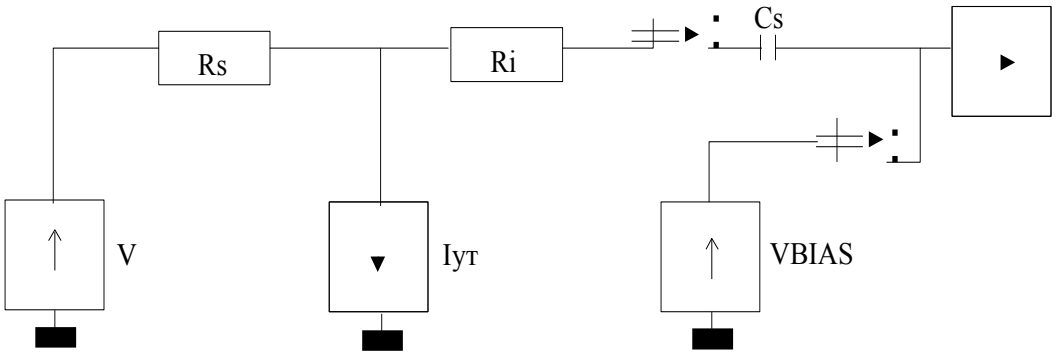


Рисунок 11.2.1 - Идеализированная схема входа АЦП

Внешняя цепь должна быть в состоянии зарядить конденсатор Cs (запоминающий) через последовательно включенный резистор (Ri) до требуемого напряжения, с учетом тока утечки (I). Cs примерно равно 2 пФ, Ri около 5 кОм, I не превышает 3 мкА. При определении необходимого внутреннего сопротивления источника сигнала Rs значение VBIAS не важно. Внешние цепи с внутренним сопротивлением 1 кОм или меньше, могут заряжать запоминающий конденсатор с точностью ±0,61 единицы младшего разряда (1,0 кОм × 3,0 мкА = 3,0 мВ), давая при этом ток утечки. Если внутреннее сопротивление источника сигнала больше 25 кОм, то точность АЦП может уменьшиться за счет неполного разряда запоминающего конденсатора за 1 мкс (при 12 МГц), в течение окна выборки. Важно отметить, что требования к внутреннему сопротивлению источника могут быть ослаблены, если подключить внешний конденсатор ко входу непосредственно. Поскольку внутренний запоминающий конденсатор имеет емкость 2,0 пФ, то внешний конденсатор емкостью 0,005 мкФ (2048 × 2,0 пФ) позволит получить точность входного напряжения ±0,5 единицы младшего разряда. Этот конденсатор в течение длительного времени заряжается через большое внутреннее сопротивление источника, а в течение короткого окна выборки поддерживает входное напряжение постоянным. Если конденсатор имеет ток утечки, он должен иметь большую емкость, чтобы компенсировать ток утечки.

Следует, однако, учитывать, что при большом сопротивлении источника сигнала подключение внешнего конденсатора образует фильтр низких частот, и такое разрешение проблемы возможно, если только входной сигнал изменяется медленно. Поскольку такой фильтр повышает помехозащищенность, рекомендуется всегда подключать внешние конденсаторы максимально возможной емкости для требуемого входного сигнала. Желательно также использование (перед конденсатором) резистора, включенного последовательно и имеющего небольшой номинал, который будет участвовать в создании фильтра и одновременно ограничивать ток при перегрузках по напряжению. Рекомендуемая входная цепь АЦП показана на рисунке 11.2.2, справочные характеристики АЦП см. таблицу 11.5.1.

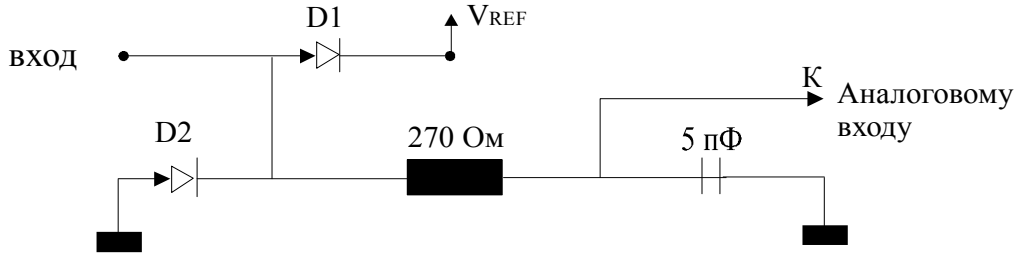


Рисунок 11.2.2 - Входная цепь АЦП

11.3 Подключение опорного напряжения к АЦП

Опорные напряжения сильно влияют на абсолютную точность преобразования. По этой причине рекомендуется, чтобы ANGND был соединен с двумя U_{SS} как можно ближе к микросхеме и наиболее короткими проводниками. Между U_{REF} и ANGND должны быть подключены шунтирующие конденсаторы. ANGND не должен отличаться от U_{SS} больше, чем на 0,1 В.

V_{REF} должно быть хорошо стабилизировано и должно использоваться только для АЦП. Источник U_{REF} должен давать напряжение между 4,5 и 5,5 В и ток 5 мА. Следует отметить, что если требуется невысокая точность, то U_{REF} может быть подключен к U_{CC}. Такое подключение возможно также в том случае, когда АЦП принимает информацию от потенциометрических резисторных датчиков, питающихся от U_{CC}, без потери точности. U_{REF} и ANGND всегда должны быть подключены как указано, даже если АЦП не используется. Помните, что порт 0 получает питание через U_{REF} и ANGND, даже если он используется для цифрового в/в.

11.4 Функция преобразования АЦП

Результат преобразования - это 10-битное представление входного напряжения по отношению к опорному в прямом коде. Цифровое значение преобразования может быть представлено, как: RESULT = INT [1023 x (Vin - ANGND) / (U_{REF} - ANGND)].

Функция преобразования является ступенчатой; выходной код - это дискретная функция входного напряжения. Чем более сложным образом применяется АЦП, тем лучше надо понимать его работу. Для простых применений достаточно знать абсолютную ошибку. Однако, применение АЦП в системах с сервоприводом, имеющих замкнутую обратную связь, требует детальные понимания работы и ошибок АЦП.

Ошибки, которые встречаются при АЦ - преобразовании, следующие:

- ошибка квантования
- ошибка смещения нуля
- ошибка полной шкалы
- дифференциальная нелинейность
- нелинейность.

Это ошибки "передаточной функции". В дополнение к этим ошибкам есть ошибки, вызванные температурным дрейфом преобразования, отклонением напряжения питания, смещением напряжения УВХ во времени, утечками изоляции мультиплексора, межканальным влиянием в мультиплексоре, случайным шумом. Всегда присутствует ошибка, вызванная преобразованием непрерывного напряжения в дискретное целое представление. Эта ошибка, называемая ошибкой квантования, всегда равна + 0,5 LSB (Least Significant Bit, наименее значимый бит). Ошибка квантования - единственная для идеального АЦП и очевидна для всех преобразователей.

11.5 Словарь терминов АЦП

LSB - LEAST SIGNIFICANT BIT - наименее значимый бит - значение напряжения, равное напряжению полной шкалы, деленному на 2 в степени n, где n - это число бит результата. Для 10-битного преобразователя с полной шкалой 5,12 В 1LSB = 5,0 мВ.

АБСОЛЮТНАЯ ОШИБКА - максимальное отличие между соответствующими точками перехода у идеальной и реальной характеристик.

ВЛИЯНИЕ ЦИФРОВОГО ПИТАНИЯ - доля шумов на U_{CC}, которая проникает на вход АЦП.

ДИФФЕРЕНЦИАЛЬНАЯ НЕЛИНЕЙНОСТЬ - разность между ширинами ступеней из идеальной и терминально - базированной характеристик.

ЗАДЕРЖКА ВЫБОРКИ - задержка во времени от момента приема сигнала старта преобразования до открытия окна выборки.

ИДЕАЛЬНАЯ ХАРАКТЕРИСТИКА - характеристика, у которой первая точка перехода имеется при Uвх = 0,5 LSB, точка перехода полной шкалы - при Uвх = (U_{REF} - 1,5 LSB) и ширина всех ступеней равна 1 LSB.

КОД - цифровое значение на выходе АЦП.

НЕЛИНЕЙНОСТЬ - максимальное отличие между точками перехода терминально - базированной характеристики и соответствующими точками перехода идеальной характеристики.

НЕОДИНАКОВОСТЬ КАНАЛОВ МУЛЬТИПЛЕКСОРА - это отличия между соответствующими точками перехода у реальных характеристик преобразования, взятых для разных каналов, но с одинаковыми температурными, электрическими и временными параметрами.

ОКНО ВЫБОРКИ - начинается в момент подключения запоминающего конденсатора к выбранному каналу и заканчивается в момент отключения этого конденсатора.

ОШИБКА ПОЛНОЙ ШКАЛЫ - разница между точками перехода полной шкалы у реальной и идеальной характеристик.

ПОВТОРЯЕМОСТЬ - разница между соответствующими точками перехода у различных реальных характеристик, взятых у одного преобразователя, на одном и том же канале, при одинаковой температуре, напряжении и временных параметрах.

ПРЕДВАРИТЕЛЬНЫЙ РАЗРЫВ (BREAK-BEFORE-MAKE) - свойство мультиплексора, которое гарантирует, что канал, выбранный в предыдущий момент, будет отключен перед тем, как следующий канал будет выбран (т.е. преобразователь никогда не закорачивает входы друг на друга).

ПРОНИКАНИЕ - доля напряжения выбранного канала, проходящая на вход преобразователя при закрытом "окне выборки".

РЕАЛЬНАЯ ХАРАКТЕРИСТИКА - характеристика реального преобразователя. Характеристика данного преобразователя может изменяться в зависимости от температуры, напряжения питания, временных соотношений. Реальная характеристика крайне редко имеет идеальные точки первого или последнего перехода или ширину "ступеньки". Реальная характеристика может также изменяться при многократных преобразованиях, даже если температура, питание и т.п. не изменяются.

СМЕЩЕНИЕ НУЛЯ - разница между первыми точками перехода у реальной и идеальной характеристик.

ТЕРМИНАЛЬНО-БАЗИРОВАННАЯ ХАРАКТЕРИСТИКА - это реальная характеристика, которая повернута и сдвинута так, чтобы сдвиг нуля и ошибка полной шкалы были скомпенсированы.

ТОЧКА ПЕРЕХОДА - напряжение на входе АЦП, при котором выходной код изменяется от значения Q к значению Q + 1.

УТЕЧКИ ИЗОЛЯЦИИ - доля напряжения невыбранного канала мультиплексора, проникающая в выбранный канал.

ХАРАКТЕРИСТИКА - зависимость выходного кода АЦП от входного напряжения.

ЦЕНТР КОДА - напряжение, являющееся средним арифметическим между двумя соседними точками перехода.

ШИРИНА СТУПЕНИ - разность между двумя соседними точками перехода.

Таблица 11.5.1 - Справочные характеристики АЦП при опорном напряжении U_{REF}=5,12 В и тактовой частоте 6 МГц

Параметр	Единица измерения	Типовое	Минимальное	Максимальное
Разрешение	Уровни Биты			1024 10
Абсолютная погрешность	мВ		0	+30
Погрешность смещения полной шкалы	мВ	1,25-2,5		
Смещение нуля	мВ	-1,25-2,50		
Нелинейность	мВ	7,5-12,5	0	+20
Дифференциальная нелинейность	мВ		> -5	+10
Погрешность согласования каналов	мВ	+0,5	0	+5
Воспроизводимость	мВ	+1,25		
Температурный коэффициент: смещения нуля смещения полной шкалы дифференциальной нелинейности	мВ/с мВ/с мВ/с	0,009 0,009 0,009		
Входное сопротивление	Ом		1К	5К
Входная емкость	пФ	3		
Температурный диапазон	^{o}C		-45	+85
Ток потребления	мА			8
Время преобразования	мкс	26,33		

12 Порты ввода-вывода

Микросхемы имеют пять 8-битных портов ввода-вывода. Часть из них чисто входные, часть работают только на выход, некоторые – двунаправленные, а некоторые имеют альтернативные функции. В дополнение к ним выводы устройства быстрого ввода-вывода могут использоваться как линии обычного ввода-вывода, если не нужны специальные возможности устройства быстрого ввода-вывода.

Порт 0 читается по адресу 0EH. Порт 1 является квази-двунаправленным портом и читается или записывается через ячейку 0FH. Три самых старших (двоичных) разряда порта 1 являются управляющими сигналами для выводов HOLD#/HLDA# шины порта. Порт 2 содержит три типа шин порта: квази-двунаправленную, входную и выходную. Порт 2 (см. таблицу 12.1) читается или записывается из адреса 10H. Чтение или запись данных через порт не может производиться в окне 15. На узлах EPROM и ROM чтение или запись через порт 3 и 4 производится через адрес 1FFEH.

Таблица 12.1 – Порт ввода-вывода P2 ИС

Вывод	Функция	Альтернативные функции	Регист управления
P2.0	Выход	TXD (передача последовательного порта)	IOC1.5
P2.1	Вход	RXD (прием последовательного порта)	SPCON.3
P2.2	Вход	EXTINT	IOC1.1
P2.3	Вход	T2CLK (Таймер_2 такт и скорость передачи)	IOC0.7
P2.4	Вход	T2RST (Таймер_2 сброс)	IOC0.5
P2.5	Выход	PWM выход	IOC1.0
P2.6	QBD	Таймер_2 (установка направления счета)	IOC2.1
P2.7	QBD	Таймер_2 (выборка)	N/A
Примечание – QBD = квази-двунаправленный вывод.			

При обсуждении характеристик портов ввода-вывода будут даны некоторые приближенные значения тока и напряжения. Точные технические требования даются в технических условиях на изделие.

12.1 Входные порты

Входные порты и выводы могут быть только считаны. Они не имеют выходных усилителей. Входной ток этих линий находится в диапазоне микроампер. Более конкретные данные находятся в технических условиях на изделия.

Высокоимпедансные входные выводы ИС имеют входной ток утечки в несколько микроампер и имеют емкостную нагрузку порядка 10 пФ.

В дополнение к работе в качестве цифрового входа каждая линия порта 0 может быть выбрана в качестве входа для АЦП. Линии порта 0 имеют входной ток не более 3 мкА (более точно – в ТУ). Входная емкость этих выводов – около 5 пФ, в момент выборки АЦП увеличивается (кратковременно) примерно на 5 пФ.

Выводы порта 0 являются особыми в том смысле, что они могут отдельно использоваться как цифровые входы и аналоговые входы в то же самое время. Вывод порта 0, используемый в качестве цифрового входа, ведет себя, как только что описанные входные порты с высоким импедансом. Однако, при использовании выводов порта 0 в качестве аналоговых входов требуется подача тока на конденсатор внутренней дискретной выборки при начале преобразования. Это означает, что входные характеристики вывода изменятся, если преобразование осуществляется на этом выводе. В любом случае, если порт 0 предполагается использовать как аналоговый или цифровой ввод-вывод, будет необходимо подать питание на этот порт через ввод U_{REF} и контакты ANGND.

12.2 Квази - двунаправленные порты

Порт 1 и порт 2 имеют квази - двунаправленные входы/выходы. При использовании в качестве входов данные на этих выводах должны быть стабильными в течение одного машинного цикла из SFR. Это временное состояние также пригодно для входных выводов порта 2 и аналогично для HSI в том смысле, что дискретная выборка происходит во время нахождения PH1 или CLKOUT на низком уровне. Когда квази - двунаправленные выводы используются как выводы, они будут изменять состояние вскоре после падения CLKOUT. Порт 1, порты 2.6 и 2.7 являются квази-двунаправленными. Это значит, что линия порта имеет высокоомный резистор, всегда подключенный к линии +5В, а также ключ, который включен, когда выводится 0, и выключен, когда выводится 1 (ключ подсоединен между линией порта и линией земли). Когда ключ выключен (записью 1 в данный вывод), логический уровень на данном выводе определяется внешним источником сигнала. Параллельно с высокоомным резистором подключен намного более мощный ключ, который активизируется на 1 машинный цикл, когда вывод изнутри переводится из состояния 0 в 1. Это сделано для ускорения переключения. Этот ключ способен выдавать ток 30 мА (типичное значение).

Когда процессор записывает в квазидвунаправленный порт, он на самом деле записывает в регистр, который управляет ключами на линиях порта. Когда ЦПУ читает этот порт, он обращается к внешнему выводу непосредственно. Если вывод порта используется как вход, то необходимо записать 1 в соответствующий бит порта, что отключит низкоомный ключ и останется только сравнительно высокоомный резистор, который не будет мешать управлению логическим уровнем на выводе.

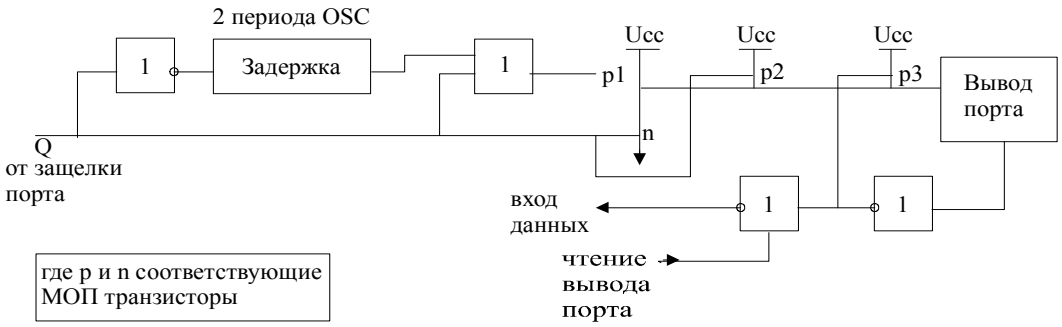


Рисунок 12.2.1 - Конфигурация КМОП квази - двунаправленного порта

12.3 Выходные порты

Выходные порты включают в себя линии управления шиной, линии HSO, часть линий порта 2. Эти выводы могут быть использованы только как выходы, поскольку не имеют входных буферов.

Выходные выводы выводят информацию до восходящего фронта сигнала PH1 и поддерживают ее в течение некоторого времени PH1. На рисунке 12.3.1 показана внутренняя конфигурация выходного порта.

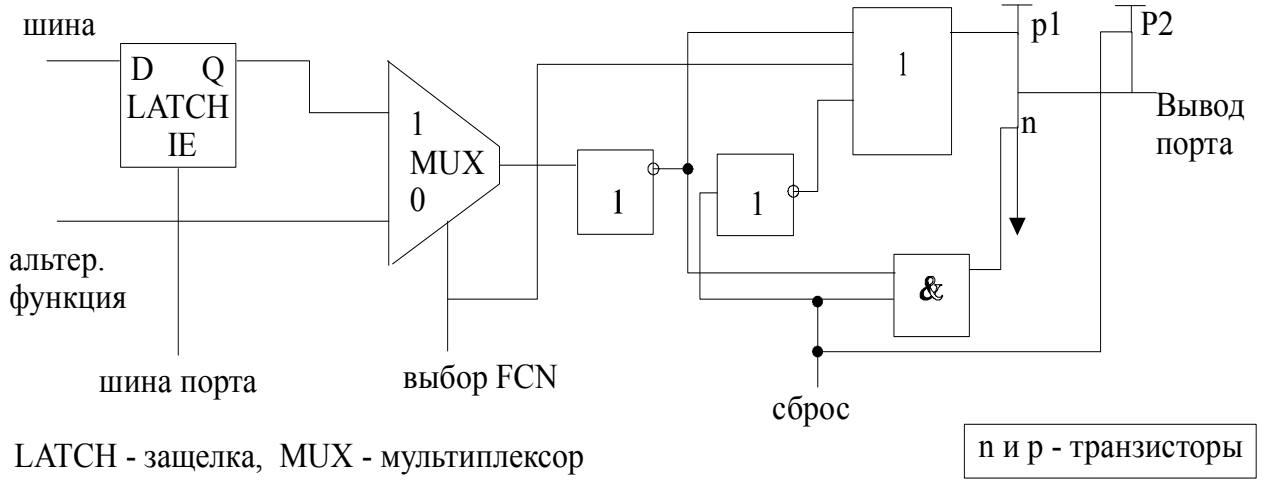


Рисунок 12.3.1 – Конфигурация выходного порта

Порты 3 и 4/AD0 - AD15

Порты 3 и 4 микроконтроллеров являются портами ввода-вывода. Блок-схема выходных буферов, связывающая порты 3 и 4 с выходной шиной, показана на рисунке 12.3.2.

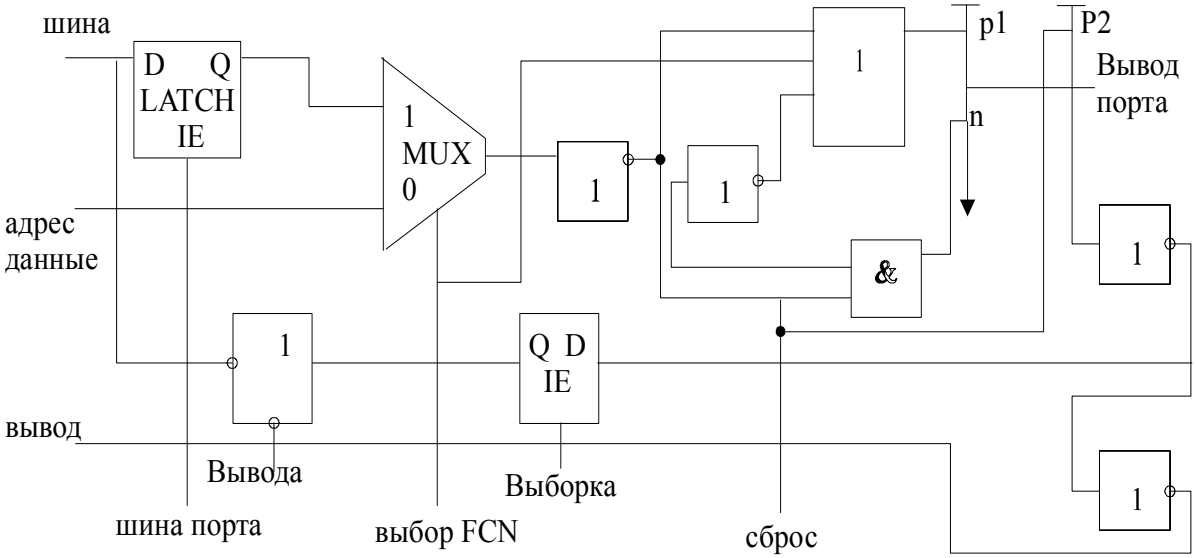


Рисунок 12.3.2 – Выводы портов 3 и 4/AD0 - AD15

13 Схема включения микроконтроллеров

Микроконтроллеры требуют определенных внешних подключений для корректной работы. Должны быть подключены земля и питание, источник тактового сигнала, кроме того должна присутствовать схема сброса. Рассмотрим каждую из этих областей более детально.

"Питание"

"Питание" к микроконтроллерам подается через пять выводов. U_{CC1} – напряжение высокого уровня для цифровых блоков кристалла, U_{REF} (U_{CC2}) запитывает блок АЦП и порт 0 высоким уровнем. Оба типа выводов нуждаются в подключении к 5-вольтовому источнику питания. Когда используется АЦП, желательно подключать U_{CC2} к отдельному источнику питания для обеспечения минимальных шумов в АЦП.

Выводы U_{SS} (GND и ANGND) должны иметь номинальный уровень 0 В. Даже если АЦП не будет использоваться, U_{REF} и ANGND должны быть все равно подключены для функционирования порта 0.

Тактовый генератор и внутренние временные диаграммы

Схема генератора для ИС, как показано на рисунке 13.1, состоит из инвертора с положительной обратной связью (реактивное сопротивление). В таком составе, кристалл работает в основном режиме как индуктивно-реактивный контур в параллельном резонансе с емкостью, внешней по отношению к кристаллу ИС.

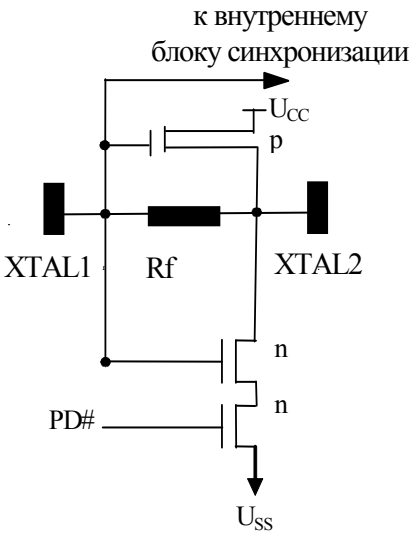


Рисунок 13.1 – Схема генератора

Резистор обратной связи Rf состоит из параллельных n-канального и p-канального транзистора, управляемого сигналом PD# (Powerdown). Rf открыт в режиме PD. Оба XTAL1 и XTAL2 имеют защиту от выбросов напряжения на выводах, которые не показаны на рисунке 13.1.

Соотношения параметров и величина емкостей на рисунке 13.2 некритичны. Керамический резонатор может быть использован вместо кварцевого. Для керамического резонатора уровень емкостей такой же. ИС могут применяться с внешним тактовым генератором, в этом случае тактовый сигнал прикладывается к выводу XTAL1, а вывод XTAL2 остается плавающим.

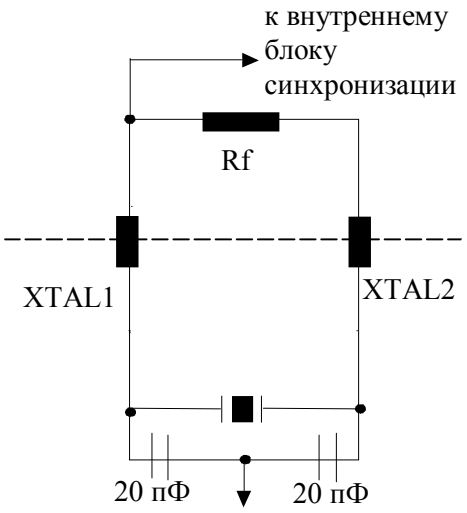


Рисунок 13.2 - Подключение внешнего кварцевого резонатора

Пример такого включения показан на рисунке 13.3. Уровни напряжений на XTAL1 должны соответствовать таблице параметров и иметь стабильные значения. Это важно, так как минимальная продолжительность высокого и низкого уровня сигнала, поступающего на вывод XTAL1 определяет временную длительность периода тактового сигнала. Наиболее длинный сигнал помехи в диапазоне чувствительности определяет наиболее высокую вероятность того, что внешний импульсный шум может быть замечен схемой тактового генератора. Импульсные помехи на внутренней линии тактового сигнала микроконтроллера могут быть причиной ненадежного функционирования.

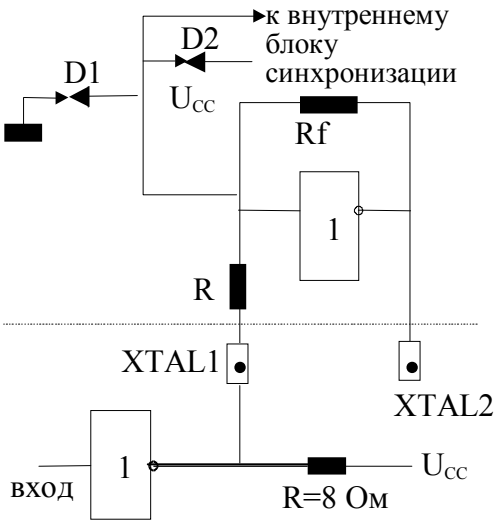


Рисунок 13.3 - Подключение внешнего генератора

Внутренние временные соотношения

Внутренняя работа микроконтроллера основана на частоте колебаний деленной на два, дающей основную временную единицу, известную как "машинный цикл". С 20МГц кварцевым резонатором машинный цикл составляет 100 нс. Микроконтроллер может работать на многих частотах, частота задающего генератора будет определять длительность машинного цикла.

Две неперекрывающиеся внутренние фазы, выдаваемые тактовым генератором (фаза 1 и фаза 2), показаны на рисунке 13.4. CLKOUT генерируется передними фронтами фазы 1 и фазы 2.

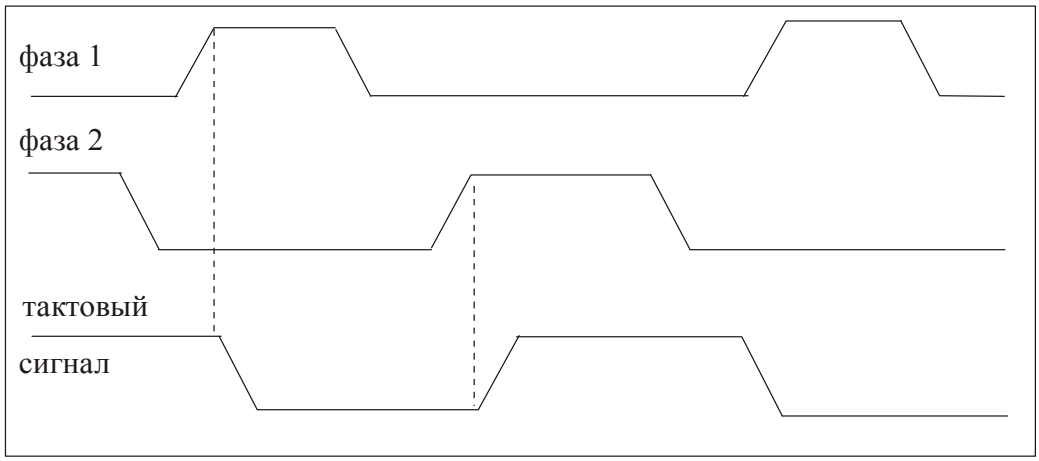


Рисунок 13.4 – Внутренние временные соотношения

Сброс и статус сброса

Начало сброса выключает известные состояния ИС. Для сброса кристалла вывод RESET# должен быть установлен как минимум на четыре машинных цикла после включения питания. Как только вывод RESET# выставлен в низкий уровень, входы/выходы и выводы контроля управляются асинхронно в состоянии сброса.

После того, как сигнал RESET# переходит в высокое состояние, в течение десяти машинных циклов имеет место стартовая последовательность (рисунок 13.5).

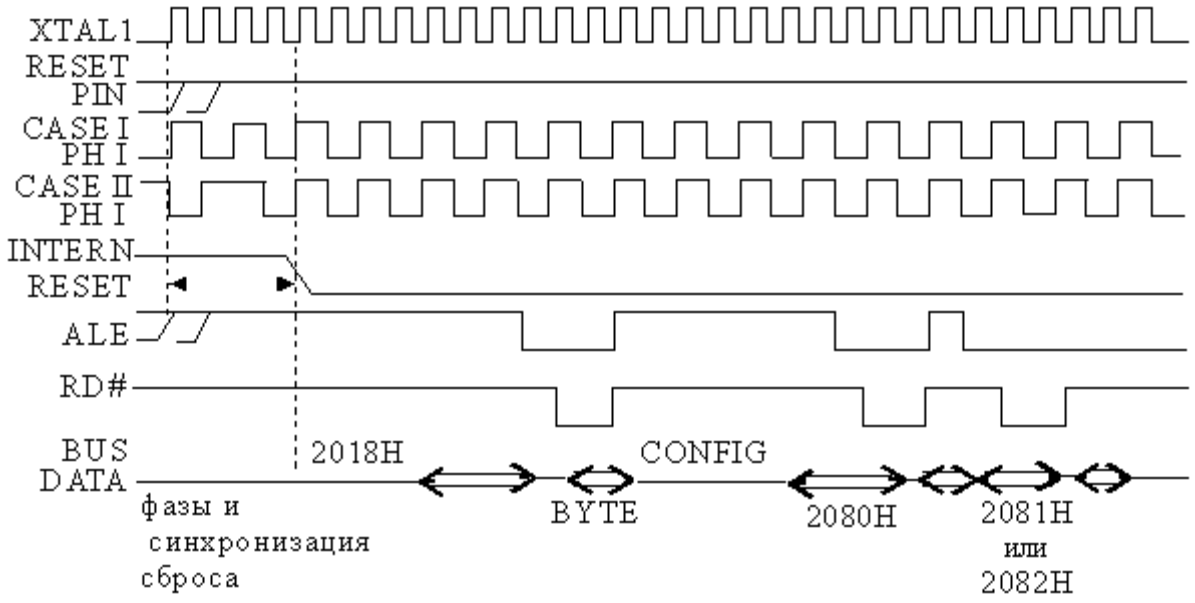


Рисунок 13.5 – Последовательность сброса

В течение этого времени CCB (бит конфигурации кристалла) читается по адресу 2018H и сохраняется в CCR (регистре конфигурации кристалла). Вывод EA# определяет чтение CCB из внешней или внутренней памяти.

Сброс

Существует три пути сброса микроконтроллеров:

- сторожевой таймер: сторожевой таймер будет сбрасывать микроконтроллер, если он не очищен в течение 64К машинных циклов. Сторожевой таймер разрешен с самого начала своей очистки. Для очистки сторожевого таймера записывается "1E" непосредственно следом за "E1" по адресу 0AH;

- команда RST: RST команда также будет сбрасывать микроконтроллер. Код операции для RST команд 0FFH;

- аппаратный сброс: аппаратный сброс микроконтроллера состоит во включении конденсатора между выводами RESET# и U_{SS}. ИС имеют внутреннюю схему установки высокого уровня, которая имеет уровень сопротивления между 6 и 50 кОм. 5 мкФ или большая емкость будет обеспечивать достаточное время сброса, пока включается U_{CC}.

Вывод RESET# функционирует как входной и как выходной при сбросе всей системы, при переполнении сторожевого таймера или посредством команды RST. Лучше всего управлять выводом RESET# при помощи выхода с открытым коллектором. Импульс сброса должен быть удлинен и буферизирован от RESET# только для четырех машинных циклов.

Схема минимального подключения микроконтроллеров

Рисунок 13.8 показывает минимальное подключение, необходимое для обеспечения работы ИС. Необходимо подключить все неиспользуемые выводы к U_{CC} или U_{SS}. Если эти выводы оставлены плавающими, на них наводится средний уровень напряжения и течет непредусмотренный ток. Некоторые выводы, такие как NMI или EXTINT, могут вызывать непредусмотренные прерывания, если их не подключить.

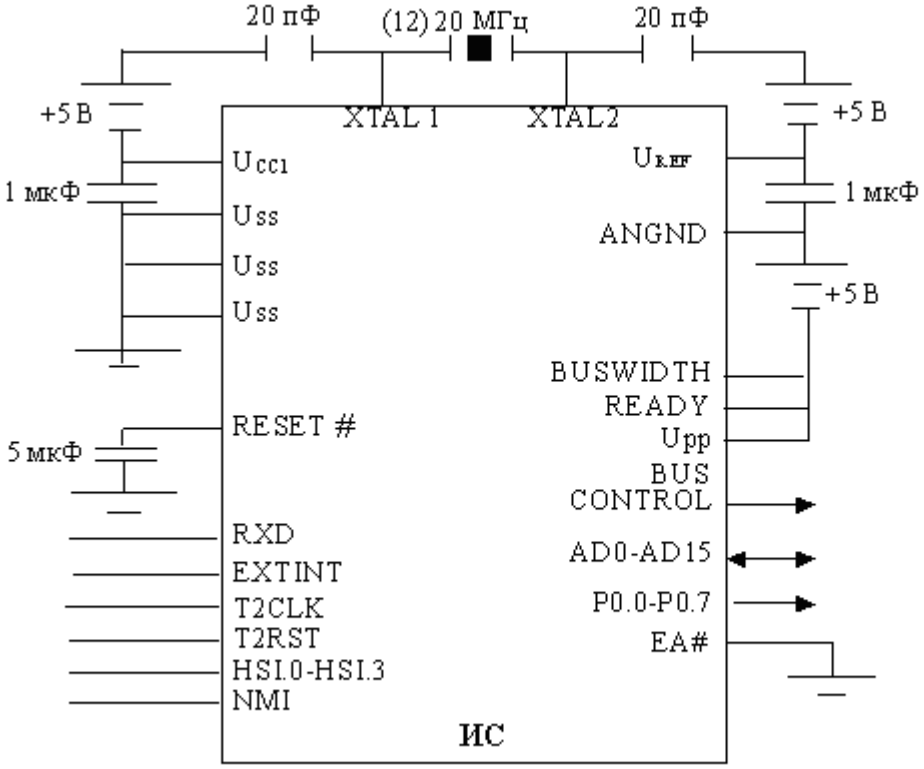


Рисунок 13.8 – Схема минимального подключения микроконтроллеров

14 Специальные режимы работы

Специальные режимы работы

ИС имеют режимы ожидания и энергосбережения, сокращающие ток потребления кристалла. Микроконтроллеры также имеют режим ONCE (внутрисхемная эмуляция), защищающий от сброса в системе.

Режим хранения

Режим хранения включается командой "IDLPD#1". В этом режиме CPU останавливает работу. Таймеры CPU останавливаются в состоянии логического нуля, но периферийные таймеры остаются активными. Энергопотребление в режиме хранения составляет приблизительно 40 % от активного режима.

CPU уходит в режим хранения по любому разрешенному прерыванию или аппаратному сбросу. CPU работает со всей периферией, прерывания могут вырабатываться HSI, HSO, АЦП, последовательным портом и т.д. Когда приходит прерывание, CPU устанавливается в режим хранения, CPU обслуживает полученное прерывание и переходит в активный режим. CPU возвращается из обслуживания прерывания к следующей по порядку команде за командой "IDLPD#1", которая переводит CPU в режим хранения.

В этом режиме выводы контроля системной шины (ALE, RD#, WR#, INST и BHE#) переводятся в неактивное состояние. Порты 3 и 4 будут удерживать предыдущий уровень данных в защелках, если они использовались как порты ввода-вывода. Если эти порты являлись шиной адреса-данных, то выводы будут плавающими.

Это важно для сторожевого таймера, продолжающего работать в режиме хранения. Если этот режим разрешен, то для сохранения работоспособности кристалла необходимо очищать сторожевой таймер каждые 64К машинных циклов, иначе кристалл будет сбрасываться.

Режим энергосбережения

Режим энергосбережения запускается подачей команды "IDLPD#2". В режиме энергосбережения все внутренние таймеры устанавливаются в состояние логического нуля. Все 232 байта регистров и основная периферия сохраняют свое состояние. Энергопотребление сокращается до уровня утечек элементов и лежит в микроамперном диапазоне.

В режиме энергосбережения выводы контроля шины переходят в неактивное состояние. Все выходные выводы будут принимать уровень их регистров данных. Порты 3 и 4 будут продолжать оставаться активными как порты в режиме одиночного кристалла или будут плавающими, если использовались как шина адреса-данных. Предотвращение случайного вхождения в режим энергосбережения может быть обеспечено сбросом бита 0 CCR (регистр конфигурации кристалла). По умолчанию уровень бита 0 CCR равен 1, режим энергосбережения нормально разрешен. Для выхода их режима Powerdown при помощи внешнего прерывания нужно создать высокий уровень на выводе EXTINT или P0.7, при этом прерывание INT7 не должно быть маскировано.

Рисунок 14.1 показывает сброс питания и установку питания при использовании внешнего прерывания.

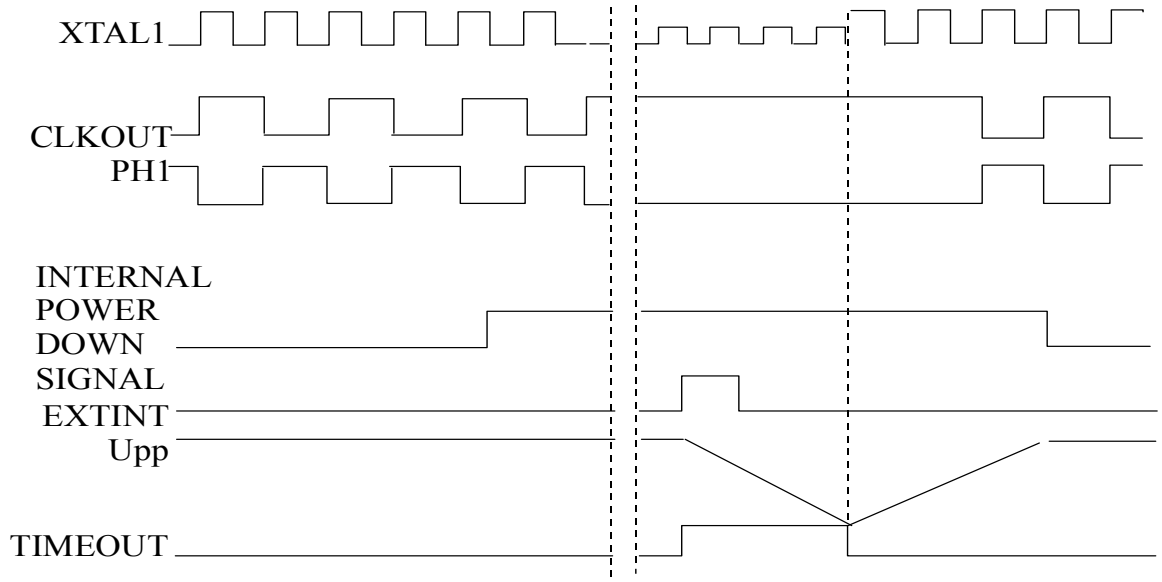


Рисунок 14.1 - Установка и сброс питания

Если внешнее прерывание переводит кристалл в режим энергосбережения, тот будет устанавливать в ожидание регистр прерываний. Если прерывание демаскировано, микроконтроллер немедленно выполняет обслуживание прерывания, возвращает к команде следующей за IDLPD командой, которая переводила кристалл в режим энергосбережения. Если прерывание маскировано, кристалл стартует с команды следующей за командой IDLPD.

Вся периферия будет переводиться в неактивный режим перед переводом в режим энергосбережения. Если АЦП находится в середине преобразования, то он прерывается. Если кристалл переводится в режим энергосбережения внешним прерыванием, последовательный порт будет оставаться выключенным. Последовательный порт работает, если до включения энергосбережения он работал на прием или передачу. SFRS объединенный с АЦП и последовательным портом может также содержать некорректную информацию, когда возвращается из режима энергосбережения. Когда кристалл находится в режиме энергосбережения, работа сторожевого таймера невозможна, так как его счетчик останавливается. Системы, которые должны использовать сторожевой таймер и режим энергосбережения , должны очистить сторожевой таймер перед переводом в режим энергосбережения. Это будет сохранять значение сторожевого таймера во время стабилизации блока синхронизации после выхода из режима энергосбережения.

15 Внешний интерфейс памяти

В стандартном режиме внешняя память адресуется через линии AD0 - AD15, которые составляют 16-разрядную мультиплексируемую шину. Шина адреса-данных связана с выводами 3 и 4 порта. На рисунке 15.1 показаны идеализированные временные диаграммы для внешних сигналов шины. Ключ разрешения адреса (ALE) определяется стробом регистра защелки демультиплексора шины. Во избежание ошибок записываемый сигнал адреса будет называться MA0 - MA15, данные будут иметь имя MD0 - MD15.

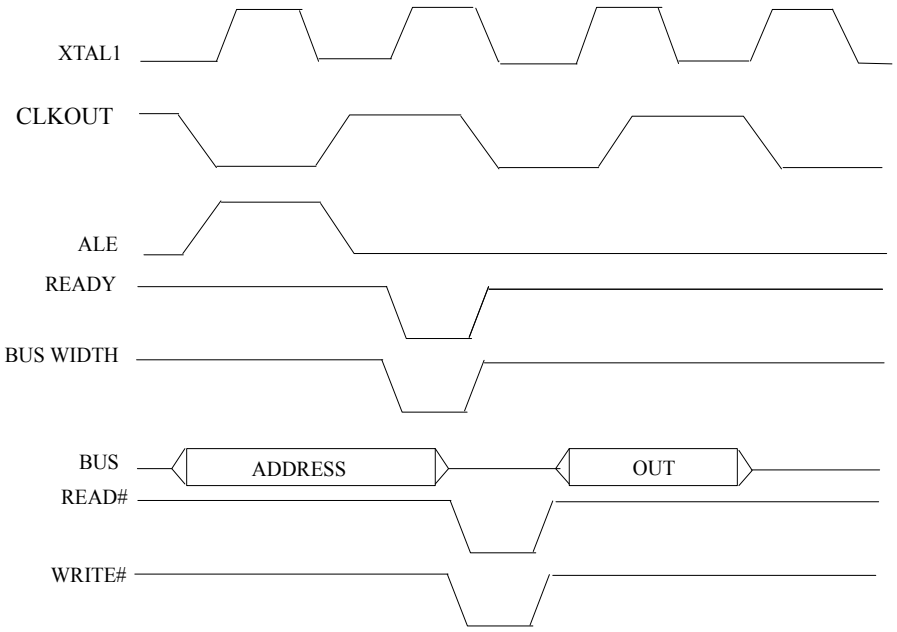


Рисунок 15.1 – Идеализированная диаграмма шины

Данные, поступающие из внешней памяти, должны быть на шине и стабилизироваться за определенное время установки перед появлением фронта RD# (чтение). Фронт сигнала RD# и его конец находятся в одном окне. Запись во внешнюю память связана с сигналом WR# (запись). Данные действительны на MD0 - MD15 по фронту сигнала WR#. В это время данные должны быть захвачены внешней системой. Микроконтроллеры имеют внешние установки и время удержания для записи.

Когда BHE# установлен, память подсоединяется к старшему байту шины данных. Когда MA0 = 0, память подключается к младшему байту шины данных. На этом этапе доступ к 16-разрядной памяти может быть осуществлен только к младшему байту (MA0 = 0, BHE# = 1), только к старшему байту (MA0 = 1, BHE# = 0), обоим байтам (MA0 = 0, BHE# = 0).

Когда блок памяти декодируется только для чтения, система не может декодировать BHE# и MA0. Микроконтроллеры будут сбрасывать этот байт, т.к. он больше не нужен. Для систем, которые записывают во внешнюю память, система должна генерировать раздельные сигналы записи для обоих (старших и младших) байтов памяти.

Все внешние сигналы шины привязаны к переднему и заднему фронту сигнала CLKOUT. Нулевой машинный цикл шины включает два периода CLKOUT. Следовательно, четыре такта составляют цикл работы шины.

Первый задний фронт CLKOUT устанавливает ALE и выставляет адрес на шине. Передний фронт CLKOUT приводит ALE в неактивное состояние. Следующий задний фронт CLKOUT устанавливает RD# и переводит шину в режим чтения. В течение цикла WR# этот фронт устанавливает WR# и выставляет действительные данные на шину.

В последнем переднем фронте CLKOUT данные защелкиваются в микроконтроллерах для цикла чтения или выставляются действительные данные для записи.

Вывод READY (готовность)

Вывод READY может добавлять машинные циклы в цикл шины для связи с медленной памятью или периферией. Машинный цикл составляет два цикла внешнего тактового генератора.

Вывод INST

Вывод INST служит для адресации более чем 64К адресного пространства. Для команд, пришедших из внешней памяти, вывод INST установлен (высокий) для всего цикла шины. Для данных (читаемых и записываемых) вывод INST имеет низкий уровень. Для поступления бита конфигурации кристалла или вектора прерывания вывод INST должен иметь низкий уровень.

15.1 Регистр конфигурации кристалла (CCR)

CCR – это первый байт, поступающий из памяти после сброса кристалла. CCR поступает из CCB (байта конфигурации кристалла), адресуемого вектором 2018H во внешней или внутренней памяти, определяемого состоянием вывода EA#. Сделанная загрузка CCR не может быть изменена пока не будет следующего сброса.

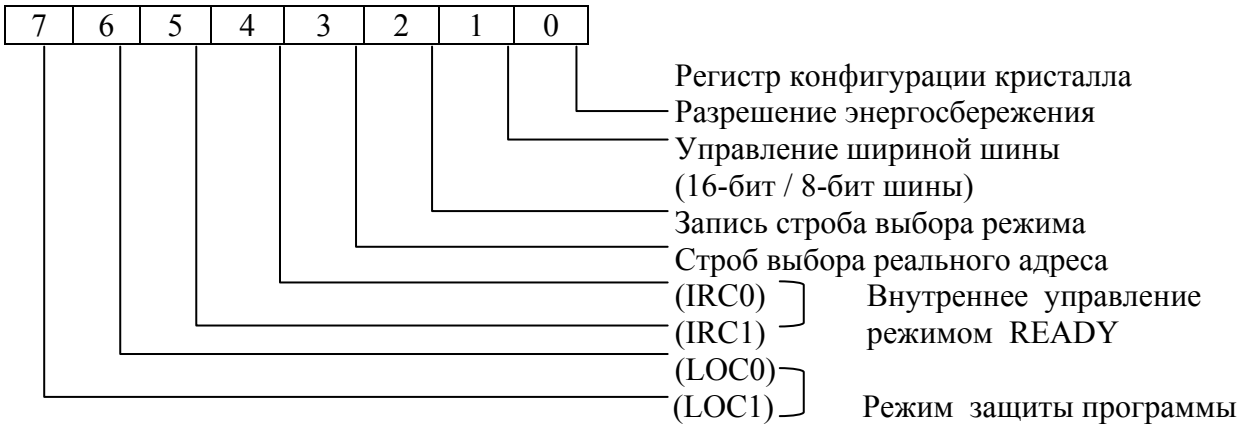


Рисунок 15.1.1 – Конфигурация регистра CCR

CCR показан на рисунке 15.1.1. Два старших бита определяют тип памяти – ROM/EPROM. Следующие два бита контролируют внутренний режим READY. Следующие три бита определяют сигналы контроля шины. Последний бит разрешает или запрещает режим энергосбережения. Перед вызовом CCB, если программная память внешняя, CPU определяет конфигурацию шины как 8-разрядную. В 8-разрядном режиме шина находится в течение всего вызова CCB, адресные линии 8-15 используются только слабым управлением. Однако, в 16-разрядном режиме шины, внешняя память будет управляться верхним байтом шины, пока выводится CCB. Адрес 20H, возмещаемый в 2019H, будет предотвращать неопределенность.

15.2 Управление готовностью (READY контроль)

Для упрощения управления готовностью можно использовать четыре режима работы. Эти режимы выбираются простым конфигурированием битов 4 и 5 в CCR и представлены в таблице 15.2.1.

Внутреннее управление готовностью служит для ограничения числа циклов ожидания, которые медленные устройства могут вставлять в цикл шины.

Если сигнал на выводе READY имеет низкий уровень, то циклы ожидания будут добавляться к циклу шины, пока вывод READY не примет значение логической 1, или пока число циклов ожидания не станет равным числу, определенному CCR.4 и CCR.5. Внутреннее управление готовностью может быть запрещено, если CCR.4 и CCR.5 загрузить логической 1.

Таблица 15.2.1 – Внутреннее управление готовностью

IRC1	IRC0	Описание
0	0	Ограничение 1-м циклом ожидания
0	1	Ограничение 2-мя циклами
1	0	Ограничение 3-мя циклами
1	1	Запрет внутреннего управления готовностью

Управление шиной микроконтроллеров

Используя CCR, микроконтроллеры могут генерировать различные типы сигналов контроля. ALE, WR# и BHE# выводы выполняют двойные функции. Биты 2 и 3 CCR определяют эти функции.

Стандартный режим шины

Если CCR биты 2 и 3 в единице, то генерируются стандартные сигналы ALE, WR# и BHE#, как показано на рисунке 15.2.1. Импульс ALE стартует по выставлении адреса, а задний фронт внешне защелкивает адрес. WR# – управляющий для каждой записи. BHE# и MA0 могут комбинироваться из WRL# и WRH# для четного и нечетного байтов записи.

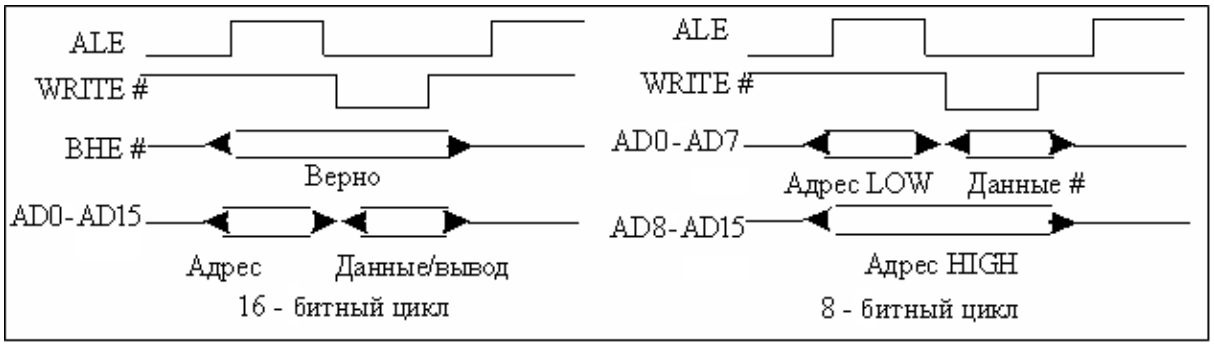


Рисунок 15.2.1 – Диаграмма стандартного режима шины

Режим стробирования записи

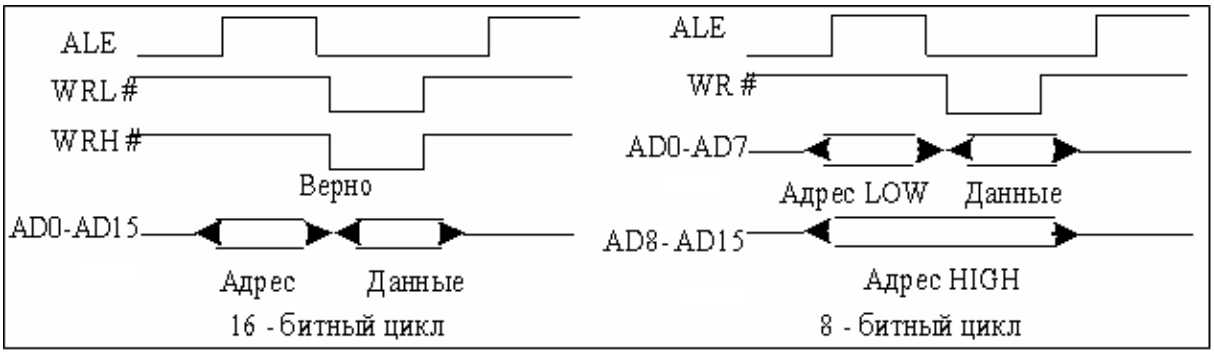


Рисунок 15.2.2 – Режим стробирования записи

Этот режим, приведенный на рисунке 15.2.2, снимает необходимость во внешней декодировке записи в четные/нечетные адреса. Если CCR.2 равен 0, и шина выполняет цикл, находясь в 16-битном режиме, сигналы WRL# и WRH# генерируются вместо WR# и BHE#.

Режим стробирования действительного адреса

WRL# будет активным (иметь низкий уровень) при записи в младшие байты и при записи слов. WRH# будет активным при записи в нечетные байты и при записи слов. WRL# будет активным при всех циклах записи, если шина находится в 8-битном режиме.

Если CCR.3=0, то вместо ALE генерируется сигнал "Адрес Верный" (ADV#). Если этот режим выбран, сигнал ADV# будет принимать низкий уровень, когда адрес установлен. Он будет иметь низкий уровень до конца цикла шины, где он примет высокий уровень (неактивный). Он может быть использован для генерации сигнала "Chip Select" (CS) для внешней памяти.

Режим стробирования действительного адреса приведен на рисунке 15.2.3.

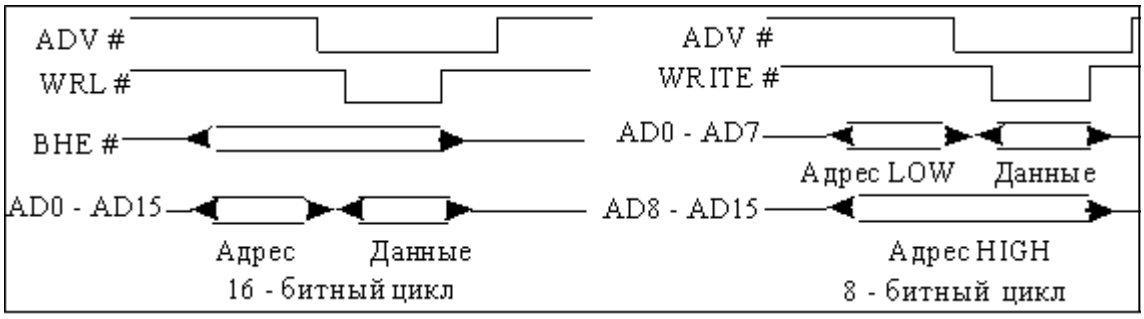


Рисунок 15.2.3 – Режим стробирования действительного адреса

Режим стробирования действительного адреса и записи

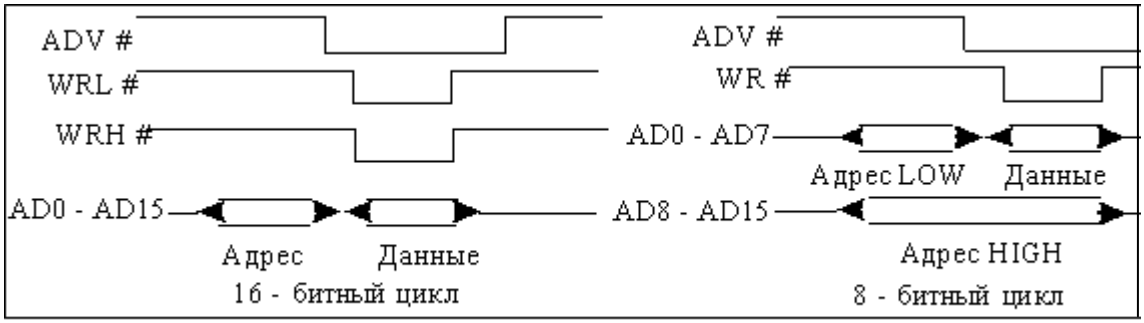


Рисунок 15.2.4 – Режим стробирования действительного адреса и записи (диаграмма)

Если CCR.2 и CCR.3 равны 0, то генерируется и строб правильности адреса, и расширенные стробы записи.

Разрядность шины

Ширина шины может динамически изменяться в процессе работы либо под стандартную 16-битную мультиплексированную шину адреса/данных, либо под 16-битную адресную и 8-битную данных. В течение 16-битных циклов шины, порты 3 и 4 содержат адрес, мультиплексированный с данными, причем ALE используется для стробирования защелок адреса. При 8-битных циклах порт 3 – это мультиплексированные адрес/данные, а порт 4 – это линии адреса с 8 по 15. Адреса на порте 4 действительны в течение всего 8-ми битного цикла шины.

Ширина шины может быть изменена в каждом машинном цикле и может управляться через бит 1 CCR и вывод BUSWIDTH (BW). Если либо CCR.1, либо BUSWIDTH – это 0, то шина становится 8-разрядной. Если оба: и CCR.1, и BUSWIDTH – это 1, то шина 16-разрядная. Доступ к внутренней памяти кристалла всегда 16-разрядный. Ширина шины может быть изменена в каждом машинном цикле, если 1 загружена в CCR.1 при сбросе. Если это так, то изменение уровня на BUSWIDTH будет динамически изменять ширину шины. Если BUSWIDTH – 1, то шина будет 16-разрядной. Для 8-разрядной шины BUSWIDTH должен быть равен 0, BUSWIDTH формируется микроконтроллерами после выставления адреса на шину. BUSWIDTH вывод имеет такую же временную диаграмму, как и вывод READY. Если CCR.1 = 0, микроконтроллеры переходят в режим 8-разрядной шины и вывод BUSWIDTH игнорируется.

15.3 HOLD#/HLDA# протокол

Микроконтроллеры поддерживают обменный протокол шины, допустимый другими устройствами, получающими контроль над шиной. Протокол включает три сигнала HOLD#, HLDA# и BREQ#. HOLD# – выходной сигнал, выставляемый устройством, которое запрашивает шину микроконтроллеров. Рисунок 15.3.1 показывает временную диаграмму сигналов для HOLD#/HLDA#. Микроконтроллеры отвечают освобождением шины и выставлением HLDA#. Когда устройство делает доступным память микроконтроллера, он освобождает шину деактивацией вывода HOLD#. ИС будут убирать свой сигнал HLDA# и принимать контроль шины. Третий сигнал, BREQ#, выставляется микроконтроллерами на протяжении секции хранения, когда он имеет ожидаемый внешний цикл. Микроконтроллеры деактивируют BREQ# во время деактивации HLDA#.

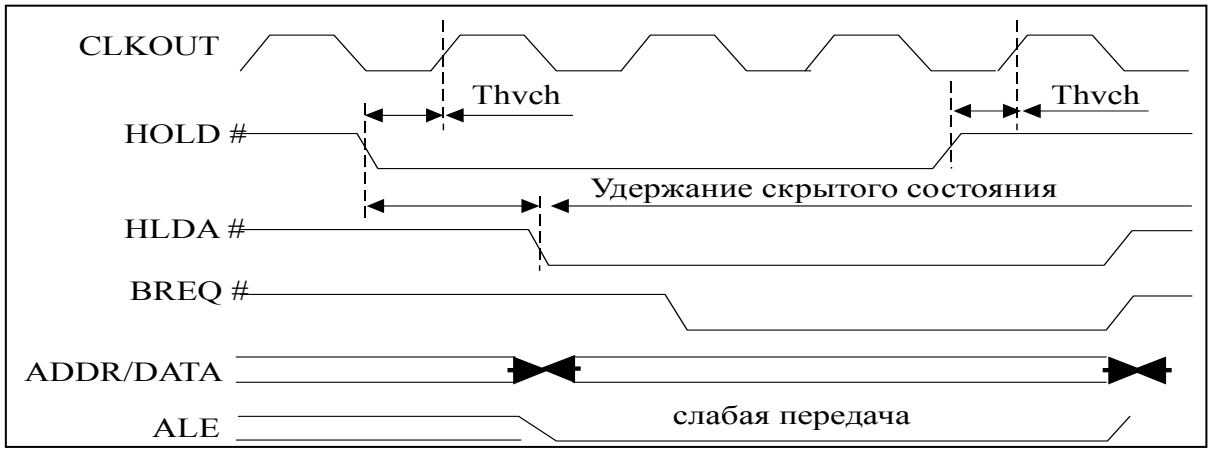


Рисунок 15.3.1 – Временная диаграмма сигналов для HOLD#/HLDA#

HOLD#, HLDA# и BREQ# выводы мультиплицируются с P1.7, P1.6 и P1.5, соответственно. При разрешении HOLD#, HLDA# и BREQ# бит HLDEN (WSR.7) должен быть равен 1. HLDEN очищается при сбросе. Бит должен быть установлен один раз, выводы порта 1 не могут быть переключены, будучи квазидвунаправленными выводами, пока устройство сброшено, но все еще могут считываться. Выводы HOLD#/HLDA# особенные, однако, они могут быть сброшены очисткой бита HLDEN.

HOLD# образуется на фазу 1 или когда код CLKOUT низкого уровня. Когда микроконтроллеры подтверждают запрос выходного буфера для шины адреса/данных, RD#, WR#, BHE# и INST переходят в плавающее состояние. Так как сильный высокий и слабый низкий уровни во время ALE/ADV# запрещены, переключение происходит в слабый низкий уровень. Это обеспечивает возможность связи ALE с другим владельцем шины. Запрос хранения скрытого состояния зависит от состояния контроллера шины.

15.4 Максимальное хранение скрытого состояния

Время между выставлением HOLD# и управлением HLDA# известно как время сохранения скрытого состояния. После опознавания HOLD# микроконтроллеры ожидают окончания любого текущего цикла шины, а затем выставляют HLDA#. Всего существует 3 типа машинных циклов: 8-разрядный внешний цикл, 16-разрядный внешний цикл и пустой цикл. Доступ к ROM/EPROM на кристалле осуществляется в пустом цикле шины.

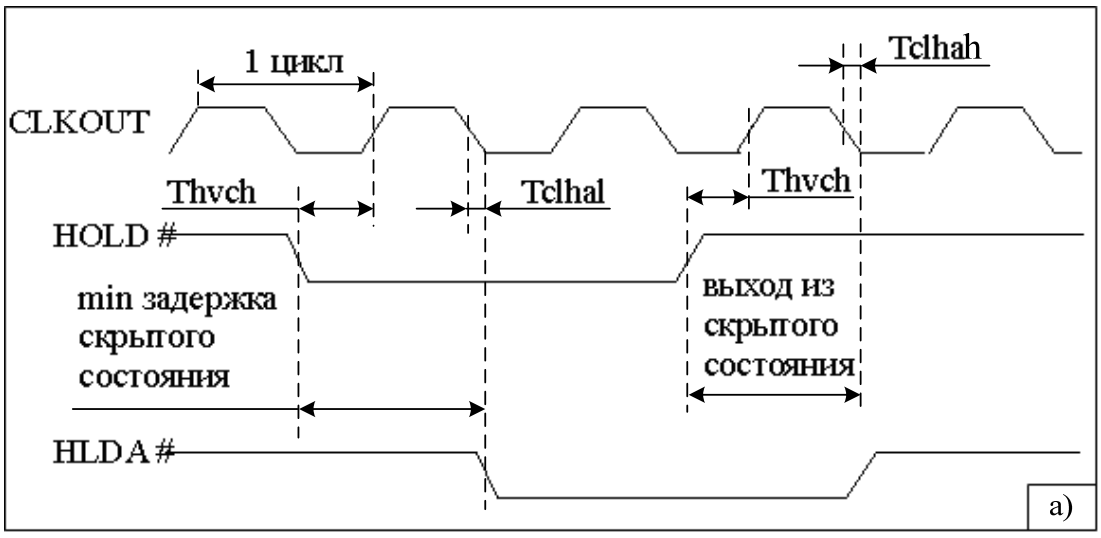
HOLD# – асинхронный вход. Есть две различных системы конфигурирования для установки HOLD#. Микроконтроллеры будут опознавать HOLD# внутренне на следующем тактовом фронте, если система удовлетворяет Thvch (HOLD# имеет действующий уровень на высоком CLKOUT). Если Thvch не удовлетворяет (HOLD# используется асинхронно), HOLD# будет опознан спустя один такт (см. рисунок 15.4.1б). Рисунок 15.4.1а показывает поступающий HOLD#, когда шина свободна. Это минимальное состояние скрытого хранения для обоих (синхронного и асинхронного) случаев. Если Thvch удовлетворяет, HLDA# устанавливается по следующему заднему фронту CLKOUT. Для этого случая, минимальное время хранения скрытого состояния равно Thvch + 0,5 машинного цикла +Tclhal. Если HOLD# устанавливается асинхронным, минимальное время сохранения скрытого состояния увеличивается на один машинный цикл и равно Thvch + 1,5 машинного цикла + Tclhal.

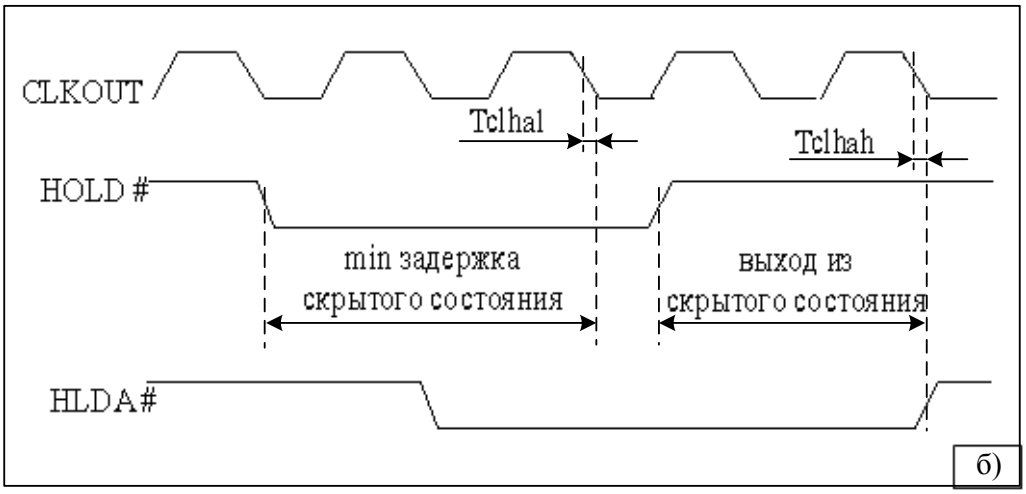
Таблица 15.4.1 суммирует дополнительное время хранения скрытого состояния, добавленного к минимальному скрытому состоянию для 3 типов машинного цикла. Когда доступна внешняя память, добавляется один такт для каждого машинного цикла, вставленного в цикл шины. При 8-разрядной шине – наихудший случай хранения скрытого состояния для чтения или записи слова. Для этого случая, контроллер шины увеличивает время скрытого состояния на два такта. Для хранения выхода минимальное время хранения скрытого состояния применяется для микроконтроллеров, когда будет снят сигнал HLDA# в ответ на сброс HOLD#.

Таблица 15.4.1 – Максимальное удержание скрытого слоя

Пустой цикл шины	Min
16-битный внешний доступ	Min + 1 машинный цикл
8-битный внешний доступ	Min + 3 машинных цикла

Min = Thvch + 0,5 такта + Tclhal, если имеет место Thvch
Thvch + 1,5 такта + Tclhal, для асинхронного HOLD#





а) имеет место Thvch;
б) установлен асинхронный сигнал HOLD#

Рисунок 15.4.1 – Асинхронный ввод HOLD#

15.5 Возвращение контроля шины

Между временем сброса сигнала HLDA# и временем приема контроля шины задержки не существует. После сброса HLDA# микроконтроллеры переводят HLDA# в низкий уровень и возобновляют контроль шины.

BREQ# устанавливается, когда ИС находятся в режиме хранения и нуждаются в выполнении цикла внешней памяти. Цикл внешней памяти может быть для доступа к данным или запроса из предварительной очереди запросов. Запрос может происходить из очереди, когда она содержит два или менее байта. Однажды установленный, он остается активным, пока HOLD# сброшен. В равной степени, BREQ# может быть установлен с сигналом HLDA#.

Как только микроконтроллеры начинают нуждаться в доступе к внешней памяти, то выставляют BREQ# и ожидают, пока HOLD# будет сброшен. В это время, микроконтроллеры не могут отвечать на любой запрос прерываний, пока HOLD# сброшен.

В течение сигнала HOLD# выполняется вывод внешней памяти, микроконтроллеры сохранят работу, пока очередь пустая или необходимо выполнить внешний цикл данных. Микроконтроллеры не могут обслуживать любые прерывания, пока HOLD# сброшен.

Микроконтроллеры будут также реагировать на сохранение очереди в режиме простоя. Скрытое состояние для входной шины сохраняется в режиме простоя также как при выполнении вывода внутренней памяти.

15.6 Запрет запроса HOLD#

Очистка бита (WSR.7) может запрещать HOLD# запрос, когда последовательные циклы памяти осуществляют запрос. Очистка бита HLDEN, однако, не является причиной немедленного взятия контроля над шиной. Микроконтроллеры ожидают окончания текущего запроса HOLD#. Затем этот запрет удерживает шину, являясь причиной того, что любой новый запрос будет игнорироваться, пока бит HLDEN установится опять.

Необходимо вводить задержку вызова кода очистки бита от фактического времени выполнения кода очистки HLDEN, для этого нужно ввести несколько команд для блока, который нужно защитить от запроса HOLD#.

Безопасный путь – это добавить JBC команду задержки статуса вывода HLDA# после кода, очищающего бит HLDEN.

16 Система команд

Обзор набора инструкций

Этот раздел дает описание каждой инструкции ИС. Инструкции приведены по алфавиту по мнемоникам ассемблера. В описании инструкции не всегда показано, как инструкция воздействует на программный счетчик (PC). В этом случае инструкция инкрементирует PC на число, равное числу байт в инструкции. Для каждой инструкции показано, как она изменяет флаги.

Символ "+" означает, что флаг устанавливается в соответствии со своим назначением. Символ "-" означает, что флаг не модифицируется. Символ "1" или "0" означает, что флаг устанавливается либо сбрасывается всегда. Символ "↑↓" означает, что инструкция может установить флаг (в результате вычислений и в соответствии с назначением флага), но не может сбросить его. Тот же смысл только для сброса флага имеет символ "↓". Символ "?" означает, что флаг переходит в неопределенное (случайное) состояние.

Генерация переходов и вызовов п/п Ассемблером АО "Фитон"

Ассемблер обеспечивает правильность генерации переходов и вызовов п/п. Для всех инструкций условного перехода символ "B" может заменить "J", и Ассемблер будет генерировать последовательность кода, которая логически эквивалентна, но может достигнуть любого места в памяти. Команда JH может перейти на +127/-128 байт относительно текущего PC, а BH может достигнуть любого места в памяти. Подобным образом мнемоника "BR" сгенерирует SJMP или LJMP, а CALL - SCALL или LCALL в зависимости от адреса, на который надо перейти. Руководство пользователя по Ассемблеру АО "Фитон" может быть использовано для ознакомления с алгоритмами, по которым Ассемблер генерирует инструкции перехода в вызовах.

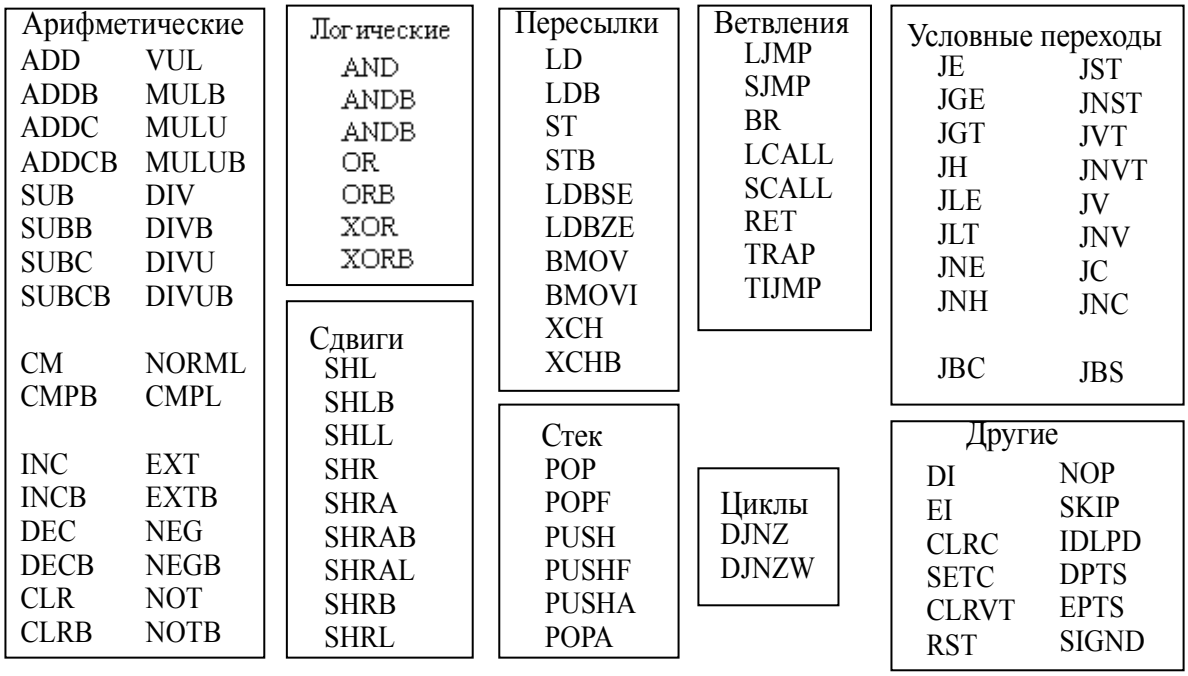


Рисунок 16.1 – Классификация команд микроконтроллеров

Описание системы команд

1 ADD (два операнда) - сложение слов

Действие: сумма двух слов запоминается в операнде назначения.

(DEST) ← (DEST) + (SRC)
Формат ассемблера: DST SRC
ADD wreg, waop
Объектный код: [011001aa] [waop] [wreg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST
+ + + + ↑ -

2 ADD (три операнда) - сложение слов

Действие: сумма второго и третьего операндов помещается в операнд назначения (самый левый).

(DEST) ← (SRC1) + (SRC2)
Ассемблер: DEST SRC1 SRC2
ADD Dwreg, Swreg, waop
Код: [010001aa] [waop] [Swreg] [Dwreg]
Байты: 3 + BEA
Циклы: 5 + CEA
Флаги: Z N C V VT ST
+ + + + ↑ -

3 ADDB (два операнда) - сложение байтов

Действие: сумма двух байтовых операндов запоминается в операнде назначения (левом).

(DEST) ← (DEST) + (SRC)
Ассемблер: DST SRC
ADDB breg, baop
Код: [011101aa] [baop] [breg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST
+ + + + ↑ -

4 ADDB (три операнда) - сложение байтов

Действие: сумма второго и третьего операндов помещается в операнд назначения (левый).

(DEST) ← (SRC1) + (SRC2)
Ассемблер: DST SRC1 SRC2
ADDB Dbreg, Sbreg, baop
Код: [010101aa] [baop] [Sbreg] [Dbreg]
Байты: 3 + BEA
Циклы: 5 + CEA
Флаги: Z N C V VT ST
+ + + + ↑ -

5 ADDC - сложение слов с переносом

Действие: сумма двух слов и флага переноса (C) запоминается в операнде назначения.

(DEST) ← (DEST) + (SRC) + C
Ассемблер: DST SRC
ADDC wreg, waop
Код: [101001aa] [waop] [wreg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST
↓ + + + ↑ -

6 ADDCB - сложение байтов с переносом

Действие: сумма двух байтовых операндов и флага переноса запоминается в операнде назначения.

(DEST) ← (DEST) + (SRC) + C
Ассемблер: DST SRC
ADDCB breg, baop
Код: [101101aa] [baop] [breg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST
↓ + + + ↑ -

7 AND (два операнда) - логическое "И" слов

Действие: результат логического "И" операндов помещается в операнд назначения.

(DEST) ← (DEST) AND (SRC)
Ассемблер: DST SRC
AND wreg, waop
Код: [011000aa] [waop] [wreg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST
+ + 0 0 - -

8 AND (три операнда) - логическое "И" слов

Действие: результат логического "И" второго и третьего операндов помещается в операнд назначения (самый левый).

(DEST) ← (SRC1) AND (SRC2)
Ассемблер: DST SRC1 SRC2
AND Dwreg, Swreg, waop
Код: [010000aa] [waop] [Swreg] [Dwreg]
Байты: 3 + BEA
Циклы: 5 + CEA
Флаги: Z N C V VT ST
+ + 0 0 - -

9 ANDB (два операнда) - логическое "И" байтов

Действие: результат логического "И" двух операндов помещается в операнд назначения.

(DEST) ← (DEST) AND (SRC)
Ассемблер: DST SRC
ANDB breg, baop
Код: [011100aa] [baop] [breg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST
+ + 0 0 - -

10 ANDB (три операнда) - логическое "И" байтов

Действие: результат логического "И" второго и третьего операндов помещается в операнд назначения.

(DEST) ← (SRC1) AND (SRC2)
Ассемблер: DST SRC1 SRC2
ANDB Dbreg, Sbreg, baop
Код: [010100aa] [baop] [Sbreg] [Dbreg]
Байты: 3 + BEA
Циклы: 5 + CEA
Флаги: Z N C V VT ST
+ + 0 0 - -

11 BMOV

Действие: пересылка блока объемом wreg.
[PTR_HI] + ← [PTR_LOW] +
Ассемблер: BMOV Lreg, wreg
Код: [11000001] [wreg] [Lreg]

12 BMOV I - команда отсутствует

13 BR (Indirect) - косвенный переход

Действие: исполнение продолжается с адреса, находящегося в специфицированном регистре длиной 1 слово.

PC ← (DEST)
Ассемблер: BR wreg
Код: [11100011] [wreg]
Байты: 2
Циклы: 8
Флаги: Z N C V VT ST

14 CLR - очистка слова

Действие: значение слова устанавливается в 0.

(DEST) ← 0
Ассемблер: CLR wreg
Код: [00000001] [wreg]
Байты: 2
Слова: 4
Флаги: Z N C V VT ST
1 0 0 0 - -

15 CLRB - очистка байта

Действие: значение байтового операнда устанавливается в 0.

(DEST) ← 0
Ассемблер: CLRB breg
Код: [00010001] [breg]
Байты: 2
Слова: 4
Флаги: Z N C V VT ST
1 0 0 0 - -

16 CLRC - очистка флага переноса

Действие: значение флага переноса устанавливается в 0.

C ← 0
Ассемблер: CLRC
Код: [11111000]
Байты: 1
Слова: 4
Флаги: Z N C V VT ST
- - 0 - - -

17 CLRVT - очистка флага ловушки переполнения

Действие: флаг ловушки переполнения сбрасывается в 0.

VT ← 0
Ассемблер: CLRVT
Код: [11111100]
Байты: 1
Слова: 4
Флаги: Z N C V VT ST
- - - - 0 -

18 CMP - сравнение слов

Действие: операнд-источник вычитается из операнда-приемника. Флаги изменяются, но операнды не изменяют своего значения.

(DEST) ← (SRC)
Ассемблер: DST SRC
CMP wreg, waop
Код: [100010aa] [waop] [wreg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST
+ + + + ↑ -

19 CMPB - сравнение байтов

Действие: то же, что и у операции CMP, но для байтов.

(DEST) ← (SRC)
Ассемблер: DST SRC
CMPB breg, baop
Код: [100110aa] [baop] [breg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST
+ + + + ↑ -

20 CMPL - команда отсутствует

21 DEC - декрементирование слова

Действие: значение операнда-слова уменьшается на 1.

(DEST) ← (DEST) - 1
Ассемблер: DEC wreg
Код: [00000101] [wreg]
Байты: 2
Слова: 4
Флаги: Z N C V VT ST
+ + + + ↑ -

22 DECB - декрементирование байта

Действие: значение операнда-байта уменьшается на 1.

(DEST) ← (DEST) - 1
Ассемблер: DECB breg
Код: [00010101] [breg]
Байты: 2
Циклы: 4
Флаги: Z N C V VT ST
+ + + + ↑ -

23 DI - запрещение прерываний

Действие: прерывания запрещаются. После этой инструкции не может быть вызвана п/п обслуживания прерывания.

(PSW.9) ← 0
Ассемблер: DI
Код: [11111010]
Байты: 1
Циклы: 4
Флаги: Z N C V VT ST

24 DIV - деление целых чисел (INTEGER)

Действие: инструкция делит содержимое операнда назначения типа LONG-INTEGER на содержимое операнда источника типа INTEGER, используя знаковую арифметику.

Младшее слово операнда назначения (имеющее меньший адрес) будет содержать частное, старшее слово - остаток.

(low word DEST) ← (DEST)/(SRC)
(high word DEST) ← (DEST)MOD(SRC)
(Эти две строки исполняются одновременно)
Ассемблер: DST SRC
DIV lreg, waop
Код: [11111110] [100011aa] [waop] [lreg]
Байты: 2 + BEA
Циклы: 29 + CEA
Флаги: Z N C V VT ST
- - - ? ↑ -

25 DIVB - деление коротких целых (SHORT-INTEGER)

Действие: содержимое операнда назначения типа INTEGER делится на содержимое операнда источника типа SHORT-INTEGER; используется знаковая арифметика. Младший байт операнда назначения будет содержать частное, старший байт - остаток.

(low byte DEST) ← (DEST)/(SRC)
(high byte DEST) ← (DEST)MOD(SRC)
(Эти две строки исполняются одновременно.)
Ассемблер: DST SRC
DIVB wreg, baop
Код: [11111110] [100111aa] [baop] [wreg]
Байты: 2 + BEA
Циклы: 21 + CEA
Флаги: Z N C V VT ST
- - - ? ↑ -

26 DIVU - деление слов

Действие: содержимое операнда назначения типа DOUBLE-WORD делится на содержимое операнда источника типа WORD; используется беззнаковая арифметика. Младшее слово операнда назначения будет содержать частное, старшее - остаток.

(low word DEST) ← (DEST)/(SRC)
(high word DEST) ← (DEST)MOD(SRC)
(Эти строки исполняются одновременно.)
Ассемблер: DST SRC
DIVU lreg, waop
Код: [100011aa] [waop] [lreg]
Байты: 2 + BEA
Циклы: 25 + CEA
Флаги: Z N C V VT ST
- - - + ↑ -

27 DIVUB - деление байтов

Действие: содержимое операнда назначения типа WORD делится на содержимое операнда источника типа BYTE; используется беззнаковая арифметика. Младший байт операнда назначения будет содержать частное, старший - остаток.

(low byte DEST) ← (DEST)/(SRC)
(high byte DEST) ← (DEST)MOD(SRC)
(Эти строки исполняются одновременно.
Ассемблер: DST SRC
DIVUB wreg, baop
Код: [100111aa] [baop] [wreg]
Байты: 2 + BEA
Циклы: 17 + CEA
Флаги: Z N C V VT ST
- - - + ↑ -

28 DJNZ - декрементирование и переход, если не ноль

Действие: значение байтового операнда декрементируется на 1. Если результат не ноль, то к PC прибавляется смещение, которое может быть в диапазоне -128...+127, что является переходом. Если равен 0, управление передается следующей инструкции.

(COUNT) ← (COUNT) - 1
if (COUNT) ↔0 then
PC ← PC + disp (расширяется до 16 бит)
end_if
Ассемблер: DJNZ breg, cadd
Код: [11100000] [breg] [disp]
Байты: 3
Циклы: если нет перехода- 5
если есть переход- 9
Флаги: Z N C V VT ST

29 DJNZW - декрементирование и переход если слово не ноль

Действие: значение операнда декрементируется не1. Если результат не ноль, то к PC прибавляется смещение, которое может быть в диапазоне -128...+127, что является переходом. Если равен нулю, управление передается следующей команде.

(COUNT) ← (COUNT) - 1
if (COUNT) ↔0 then
PC ← PC + disp
end_if
Ассемблер: DJNZ wreg, cadd
Код: [11100000] [wreg] [disp]
Байты: 3
Циклы: если нет перехода- 5
если есть переход- 9
Флаги: Z N C V VT ST

30 DPTS - команда отсутствует

31 EI - разрешение прерываний.

Действие: прерывания разрешаются после исполнения следующей инструкции. Обслуживание прерывания (вызов п/п) не может начаться сразу после данной инструкции.

Ассемблер: EI
Код: [11111011]
Байты: 1
Циклы: 4
Флаги: Z N C V VT ST

32 EPTS - команда отсутствует

33 EXT - знаковое расширение целого к длинному целому.

Действие: младшее слово операнда назначения расширяется в старшее слово операнда назначения без потери знака (операнд в дополнительном коде).

if (low word DEST) ←8000H then
(high word DEST) ← 0
else
(high word DEST) ← 0FFFFH
end_if
Ассемблер: EXT lreg

Код: [00000110] [lreg]
Байты: 2
Циклы: 4
Флаги: Z N C V VT ST
+ + 0 0 - -

34 EXTB - знаковое расширение короткого целого к целому

Действие: то же, что и у EXT, только для байтов.

if (low byte DEST) < 80H then
(high byte DEST) ← 0
else
(high byte DEST) ← 0FFH
end_if
Ассемблер: EXT wreg
Код: [00010110] [wreg]
Байты: 2
Циклы: 4
Флаги: Z N C V VT ST
+ + 0 0 - -

35 INC - инкремент слова

Действие: значение операнда-слова увеличивается на 1.

Ассемблер: INC wreg
Код: [00000111] [wreg]
Байты: 2
Циклы: 4
Флаги: Z N C V VT ST
+ + + + ↑ -

36 INCB - инкремент байта

Действие: значение байтового операнда увеличивается на 1.

Ассемблер: INCB breg
Код: [00010111] [breg]
Байты: 2
Циклы: 4
Флаги: Z N C V VT ST
+ + + + ↑ -

37 IDLPD - команда отсутствует

38 JBC - переход, если бит очищен

Действие: если специфицированный бит равен 0, то к PC прибавляется смещение (disp), которое может быть в диапазоне от минус 128 до плюс 127. В ином случае управление передается на следующую инструкцию.

if (specified bit) = 0 then
PC <- PC+disp (расширенное до 16 бит)
Ассемблер: JBC breg, bitno, cadd
Код: [00110bbb] [breg] [disp]
(bbb - номер бита в специфицированном регистре).
Байты: 3
Циклы: если нет перехода - 5; если есть переход - 9
Флаги: Z N C V VT ST

39 JBS - переход, если бит установлен

Действие: если специфицированный бит равен 1, то к PC прибавляется значение смещения (в диапазоне от минус 128 до плюс 127). В ином случае управление передается на следующую инструкцию.

if (specified bit) = 1 then
PC ← PC+disp (расширенное до 16 бит)
Ассемблер: JBS breg, bitno, cadd
Код: [00111bbb] [breg] [disp]
(bbb - номер бита в специфицированном регистре).
Байты: 3
Циклы: если нет перехода - 5; если есть переход - 9
Флаги: Z N C V VT ST

40 JC - переход, если флаг переноса установлен

Действие: если флаг переноса установлен, то к PC прибавляется смещение (в диапазоне от минус 128 до плюс 127). В ином случае управление передается на следующую инструкцию.

if C=1 then
PC ← PC + disp (расширенное до 16 бит).
Ассемблер: JC cadd
Код: [11010011] [disp]
Байты: 2
Циклы: если нет перехода - 4; если есть переход - 8
Флаги: Z N C V VT ST

41 JE - переход, если равно

Действие: если установлен индикатор 0 (флаг Z), то к PC прибавляется смещение (в диапазоне от минус 128 до плюс 127). В ином случае управление передается на следующую инструкцию.

if Z=1 then
PC ← PC + disp (расширенное до 16 бит).
Ассемблер: JE cadd
Код: [11011111] [disp]
Байты: 2
Циклы: если нет перехода - 4; если есть переход - 8
Флаги: Z N C V VT ST

42 JGE - переход, если знаковое число больше или равно

Действие: если флаг N очищен, то к PC прибавляется смещение (в диапазоне от минус 128 до плюс 127). В ином случае управление передается на следующую инструкцию.

if N=1 then
PC ← PC + disp (расширенное до 16 бит).
Ассемблер: JGE cadd
Код: [11010110] [disp]
Байты: 2
Циклы: если нет перехода - 4; если есть переход - 8
Флаги: Z N C V VT ST

43 JGT - переход, если знаковое число больше

Действие: если сброшены флаги N и Z, то к PC прибавляется смещение (в диапазоне от минус 128 до плюс 127). В ином случае управление передается на следующую инструкцию.

if N=0 AND Z=0 then
PC ← PC + disp (расширенное до 16 бит).
Ассемблер: JGT cadd
Код: [11010010] [disp]
Байты: 2
Циклы: если нет перехода - 4; если есть переход - 8
Флаги: Z N C V VT ST

44 JH - переход, если беззнаковое число больше

Действие: если флаг переноса установлен, а индикатор нуля - нет, то к PC прибавляется смещение (в диапазоне от минус 128 до плюс 127). В ином случае управление передается на следующую инструкцию.

if C=1 AND Z=0 then
PC ← PC + disp (расширенное до 16 бит).
Ассемблер: JH cadd
Код: [11011001] [disp]
Байты: 2
Циклы: если нет перехода - 4; если есть переход - 8
Флаги: Z N C V VT ST

45 JLE - переход, если знаковое число меньше или равно

Действие: если установлен в 1 хотя бы один из флагов N, Z, то к PC прибавляется смещение (в диапазоне от минус 128 до плюс 127). В ином случае управление передается на следующую инструкцию.

if N=1 OR Z=1 then
PC ← PC + disp (расширенное до 16 бит).
Ассемблер: JLE cadd
Код: [11011010] [disp]
Байты: 2
Циклы: если нет перехода - 4; если есть переход - 8
Флаги: Z N C V VT ST

46 JLT - переход, если знаковое число меньше

Действие: если установлен флаг N, то к PC прибавляется смещение (в диапазоне от минус 128 до плюс 127). В ином случае управление передается на следующую инструкцию.

if N=1 then
PC ← PC + disp (расширенное до 16 бит).
Ассемблер: JLT cadd
Код: [11011010] [disp]
Байты: 2
Циклы: если нет перехода - 4; если есть переход - 8
Флаги: Z N C V VT ST

47 JNC - переход, если флаг переноса очищен

Действие: если флаг переноса очищен, то к PC прибавляется смещение (в диапазоне от минус 128 до плюс 127). В ином случае управление передается на следующую инструкцию.

if C=0 then
PC ←PC + disp (расширенное до 16 бит).
Ассемблер: JNC cadd
Код: [11010011] [disp]
Байты: 2
Циклы: если нет перехода - 4; если есть переход - 8
Флаги: Z N C V VT ST

48 JNE - переход, если не равно

Действие: если флаг Z очищен, то к PC прибавляется смещение (в диапазоне от минус 128 до плюс 127). В ином случае управление передается на следующую инструкцию.

if Z=0 then
PC ← PC + disp (расширенное до 16 бит).
Ассемблер: JNE cadd
Код: [11010111] [disp]
Байты: 2
Циклы: если нет перехода - 4; если есть переход - 8
Флаги: Z N C V VT ST

49 JNH - переход, если беззнаковое число не больше

Действие: если флаг переноса очищен и/или индикатор нуля установлен, то к PC прибавляется смещение (в диапазоне от минус 128 до плюс 127). В ином случае управление передается на следующую инструкцию.

if C=0 OR Z=1 then
PC ← PC + disp (расширенное до 16 бит).
Ассемблер: JNH cadd
Код: [11010001] [disp]
Байты: 2
Циклы: если нет перехода - 4;
если есть переход - 8
Флаги: Z N C V VT ST

50 JNST - переход, если очищен флаг ST

Действие: если флаг ST очищен, то к PC прибавляется смещение (в диапазоне от минус 128 до плюс 127). В ином случае управление передается на следующую инструкцию.

if ST=0 then
PC ← PC + disp (расширенное до 16 бит).
Ассемблер: JNST cadd
Код: [11010000] [disp]
Байты: 2
Циклы: если нет перехода - 4;
если есть переход - 8
Флаги: Z N C V VT ST

51 JNV - переход, если нет переноса

Действие: если флаг переноса очищен, то к PC прибавляется смещение (в диапазоне от минус 128 до плюс 127). В ином случае управление передается на следующую инструкцию.

if V=0 then
PC ← PC + disp (расширенное до 16 бит).
Ассемблер: JNV cadd
Код: [11010101] [disp]
Байты: 2
Циклы: если нет перехода - 4; если есть переход - 8
Флаги: Z N C V VT ST

52 JNVT - переход, если флаг ловушки переполнения очищен

Действие: если флаг ловушки переполнения очищен, то к PC прибавляется смещение (в диапазоне от минус 128 до плюс 127). В ином случае управление передается на следующую инструкцию.

if VT=0 then
PC ← PC + disp (расширенное до 16 бит).
Ассемблер: JNVT cadd
Код: [11010100] [disp]
Байты: 2
Циклы: если нет перехода - 4; если есть переход - 8
Флаги: Z N C V VT ST

53 JST - переход, если флаг ST установлен

Действие: если флаг ST установлен, то к PC прибавляется смещение (в диапазоне от минус 128 до плюс 127). В ином случае управление передается на следующую инструкцию.

if ST=1 then
PC ← PC + disp (расширенное до 16 бит).
Ассемблер: JST cadd
Код: [11011000] [disp]
Байты: 2
Циклы: если нет перехода - 4; если есть переход - 8
Флаги: Z N C V VT ST

54 JV - переход, если переполнение

Действие: если флаг переполнения установлен, то к PC прибавляется смещение (в диапазоне от минус 128 до плюс 127). В ином случае управление передается на следующую инструкцию.

if V=1 then
PC ← PC + disp (расширенное до 16 бит).
Ассемблер: JV cadd
Код: [11011101] [disp]
Байты: 2
Циклы: если нет перехода - 4; если есть переход - 8
Флаги: Z N C V VT ST

55 JVT - переход, если установлен флаг ловушки переполнения

Действие: если флаг ловушки переполнения установлен, то к PC прибавляется смещение (в диапазоне -128...+127). В ином случае управление передается на следующую инструкцию.

if VT=1 then
PC ← PC + disp (расширенное до 16 бит).
Ассемблер: JVT cadd
Код: [11011100] [disp]
Байты: 2
Циклы: если нет перехода - 4; если есть переход - 8
Флаги: Z N C V VT ST

56 LCALL - длинный вызов п/п

Действие: содержимое PC (адрес возврата) помещается в стек; к PC прибавляется 16-битное смещение.

SP ← SP-2
(SP) ← PC
PC ← PC + disp
Ассемблер: LCALL cadd
Код: [11101111] [disp-low] [disp-high]
Байты: 3
Циклы: внутренний стек - 4; внешний стек - 8
Флаги: Z N C V VT ST

57 LD - загрузка слова

Действие: значение операнда-источника (правого) запоминается в операнде-приемнике (левом).

(DEST) <- (SRC)
Ассемблер: DST SRC
LD wreg,waop
Код: [101000aa] [waop] [wreg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST

58 LDB - загрузка байта

Действие: значение операнда-источника (правого) запоминается в операнде-приемнике (левом).

(DEST) ← (SRC)
Ассемблер: DST SRC
LDB breg,baop
Код: [101100aa] [baop] [breg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST

59 LDBSE - загрузка целого (INTEGER) из короткого целого (SHORT)

Действие: значение операнда-источника (правого) подвергается знаковому расширению и запоминается в операнде-приемнике (левом) длиной одно слово.

(low byte DEST) ← (SRC)
if (SRC) ← 80H then
(high byte DEST) ← 0
else
(high byte DEST) ← 0FFH
end_if
Ассемблер: DST SRC
LDBSE wreg,baop
Код: [101111aa] [baop] [wreg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST

60 LDBZE - загрузка слова байтом

Действие: значение байтового операнда-источника (правого) подвергается расширению нулем и запоминается в операнде-приемнике (левом) длиной одно слово.

(low byte DEST) ← (SRC)
(high byte DEST) ← 0
Ассемблер: DST SRC
LDBZE wreg,baop
Код: [101011aa] [baop] [wreg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST

61 LJMP - длинный переход

Действие: смещение прибавляется к PC. Длинный переход может достигнуть любого места адресного пространства.

PC ←Д PC + disp
Ассемблер: LJMP cadd
Код: [1110011] [disp-low] [disp-high]
Байты: 3
Циклы: 8
Флаги: Z N C V VT ST

62 MUL (два операнда) - перемножение целых чисел (INTEGER)

Действие: два операнда типа INTEGER перемножаются при использовании знаковой арифметики, 32-битный результат помещается в оператор назначения типа LONG-INTEGER.

(DEST) ←Д (DEST) * (SRC)
Ассемблер: DST SRC
MUL lreg, waop
Код: [11111110] [011011aa] [waop] [lreg]
Байты: 3 + BEA
Циклы: 29 + CEA
Флаги: Z N C V VT ST

63 MUL (три операнда) - перемножение целых

Действие: второй и третий операнды типа INTEGER перемножаются с использованием знаковой арифметики, 32-битный результат помещается в операнд назначения (левый) типа LONG-INTEGER.

(DEST) ←Д (SRC1) * (SRC2)
Ассемблер: DST SRC1 SRC2
MUL lreg, wreg, waop
Код: [11111110] [010011aa] [waop] [wreg] [lreg]
Байты: 4 + BEA
Циклы: 30 + CEA
Флаги: Z N C V VT ST

64 MULB (два операнда) - перемножение коротких целых

Действие: два операнда типа SHORT - INTEGER перемножаются с использованием знаковой арифметики и 16-битный результат помещается в операнд назначения (левый) типа INTEGER.

(DEST) ←Д (DEST) * (SRC)
Ассемблер: DST SRC
MULB wreg, baop
Код: [11111110] [011111aa] [baop] [wreg]
Байты: 3 + BEA
Циклы: 21 + CEA
Флаги: Z N C V VT ST
` ?

65 MULB (три операнда) - перемножение коротких целых

Действие: второй и третий операнды типа SHORT-INTEGER перемножаются с использованием знаковой арифметики и 16-битный результат помещается в операнд назначения (левый) типа INTEGER.

(DEST) ←Д (SRC1) * (SRC2)
Ассемблер: DST SRC1 SRC2
MULB wreg, breg, baop
Код: [11111110] [010111aa] [baop] [breg] [wreg]
Байты: 4 + BEA
Циклы: 22 + CEA
Флаги: Z N C V VT ST
?

66 MULU (два операнда) - перемножение слов

Действие: два операнда типа WORD перемножаются с использованием беззнаковой арифметики, а 32-битный результат запоминается в операнде назначения (левом) типа DOUBLE-WORD.

(DEST) ←Д (DEST) * (SRC)
Ассемблер: DST SRC
MULU lreg, waop
Код: [011011aa] [waop] [lreg]
Байты: 2 + BEA
Циклы: 25 + CEA
Флаги: Z N C V VT ST
?

67 MULU (три операнда) - перемножение слов

Действие: второй и третий операнды типа WORD перемножаются с использованием беззнаковой арифметики, 32-битный результат помещается в операнд назначения (левый) типа DOUBLE-WORD.

(DEST) ←Д (SRC1) * (SRC2)
Ассемблер: DST SRC1 SRC2
MULU lreg, wreg, waop
Код: [010011aa] [waop] [wreg] [lreg]
Байты: 3 + BEA
Циклы: 26 + CEA
Флаги: Z N C V VT ST
?

68 MULUB (два операнда) - перемножение байтов

Действие: два операнда типа BYTE перемножаются с использованием беззнаковой арифметики, результат типа WORD помещается в операнд назначения (левый).

(DEST) ←Д (DEST) * (SRC)
Ассемблер: DST SRC
MULUB wreg, baop
Код: [011111aa] [baop] [wreg]
Байты: 2 + BEA
Циклы: 17 + CEA
Флаги: Z N C V VT ST
?

69 MULUB (три операнда) - перемножение байтов

Действие: второй и третий операнды типа BYTE перемножаются с использованием беззнаковой арифметики, результат типа WORD помещается в операнд назначения (левый).

(DEST) ← Д (SRC1) * (SRC2)
Ассемблер: DST SRC1 SRC2
MULUB wreg, breg, baop
Код: [010111aa] [baop] [breg] [wreg]
Байты: 3 + CEA
Циклы: 18 + CEA
Флаги: Z N C V VT ST
?

70 NEG - изменить знак целого

Действие: значение операнда типа INTEGER изменяет знак.

(DEST) ← Д (DEST)
Ассемблер: NEG wreg
Код: [00000011] [wreg]
Байты: 2
Циклы: 4
Флаги: Z N C V VT ST
+ + + + ↑ -

71 NEGB - изменить знак короткого целого

Действие: значение операнда типа SHORT-INTEGER изменяет знак.

(DEST) ← Д (DEST)
Ассемблер: HEGB breg
Код: [00010011] [breg]
Байты: 2
Циклы: 4
Флаги: Z N C V VT ST
+ + + + ↑ -

72 NOP - нет действия

Действие: ничего не делается, управление передается на следующую инструкцию.

Ассемблер: NOP
Код: [11111101]
Байты: 1
Циклы: 4
Флаги: Z N C V VT ST

73 NORML - нормализация длинного целого

Действие: операнд типа LONG-INTEGER нормализуется, т.е. смещается влево, пока самый значимый бит не станет равным 1. Если старший бит будет сохранять значение 0 после 31 сдвига, то процесс останавливается и флаг Z устанавливается. Число сдвигов запоминается во втором операнде.

(COUNT) ← Д 0
do while (MSB (DEST)=0) AND ((COUNT)<31)
(DEST) ← Д (DEST) * 2
(COUNT) ← Д (COUNT) + 1
end while
Ассемблер: NORML lreg, breg
Код: [00001111] [breg] [lreg]
Байты: 3
Циклы: 11 + число сдвигов
Флаги: Z N C V VT ST
+ ? 0 - - -

74 NOT - инверсия слова

Действие: значение операнда типа WORD подвергается побитной логической инверсии.

(DEST) <Д NOT (DEST)
Ассемблер: NOT wreg
Код: [00000010] [wreg]
Байты: 2
Циклы: 4
Флаги: Z N C V VT ST
+ + 0 0 - -

75 NOTB - инверсия байта

Действие: значение операнда типа BYTE подвергается логической инверсии.

(DEST) <Д NOT (DEST)
Ассемблер: NOTB breg
Код: [00010010] [breg]

Байты: 2
Циклы: 4
Флаги: Z N C V VT ST
+ + 0 0 -

76 OR - логическое "ИЛИ" слов

Действие: операнд - источник типа WORD подвергается логическому побитному "ИЛИ" с операндом - приемником (левым) типа WORD, результат помещается в операнд - приемник.

(DEST) ←Д (DEST) OR (SRC)
Ассемблер: DST SRC
OR wreg, waop
Код: [100000aa) [waop] [wreg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST
+ + 0 0 - -

77 ORB - логическое "ИЛИ" байтов

Действие: то же, что и у OR, но для операндов типа BYTE.

(DEST) ←Д (DEST) OR (SRC)
Ассемблер: ORB breg, baop
Код: [100100aa] [baop] [breg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST
+ + 0 0 - -

78 POP - извлечь из стека слово

Действие: слово вершины стека помещается в операнд назначения, SP увеличивается на 2.

(DEST) ←Д (SP)
SP ←Д SP + 2
Ассемблер: POP waop
Код: [110011aa] [waop]
Байты: 1 + BEA
Циклы: внутренний стек: 12 + CEA
внешний стек: 14 + CEA
Флаги: Z N C V VT ST

79 POPA - извлечь из стека все

Действие: извлечь из стека все.
SP ← SP + 2
Ассемблер: POPA
Код: [11110101]
INT_MASKI / WSR← (SP)
PSW / INT_MASK ← (SP)
Флаги: Z N C V VT ST
+ + + + + +

80 POPF - извлечь из стека флаги

Действие: слово из вершины стека извлекается и размещается в PSW. Указатель стека увеличивается на 2. Прерывание не может произойти сразу за этой инструкцией.

(PSW) ←Д (SP)
SP ←Д SP + 2
Ассемблер: POPF
Код: [11110011]
Байты: 1
Циклы: внутренний стек: 9
  внешний стек: 13
Флаги: Z N C V VT ST
         + + + + + +

81 PUSH - поместить слово в стек

Действие: специфицированный операнд помещается в стек.
SP ←Д SP - 2
(SP) ←Д (DEST)
Ассемблер: PUSH waop
Код: [110010aa] [waop]
Байты: 1 + BEA
Циклы: внутренний стек: 8 + CEA
  внешний стек: 12 + CEA
Флаги: Z N C V VT ST

82 PUSHA - поместить все в стек

Действие: поместить все в стек.
SP ←SP - 2
(SP) ← PSW / INT_MASK
PSW / INT_MASK ← 0
SP ←SP - 2
(SP) ← INT_MASKI / PSR
INT_MASK ← 0
Ассемблер: PUSHA
Код: [11110100]
Флаги: Z N C V VT ST
         0 0 0 0 0 0

83 PUSHF - поместить флаги в стек

Действие: PSW помещается в вершину стека, а затем обнуляется. Все прерывания запрещаются. Вызовы п/п обслуживания прерываний невозможны сразу за этой инструкцией.

SP ← Д SP - 2
(SP) ←Д PSW
PSW ←Д 0
Ассемблер: PUSHF
Код: [11110010]
Байты: 1
Циклы: внутренний стек: 8
  внешний стек: 12
Флаги: Z N C V VT ST
         0 0 0 0 0 0

84 RET - возврат из п/п

Действие: PC извлекается из вершины стека.

PC ←Д (SP)
SP ←Д SP + 2
Ассемблер: RET
Код: [11110000]
Байты: 1
Циклы: внутренний стек: 12
  внешний стек: 16
Флаги: Z N C V VT ST

85 RST - сброс процессора

Действие: PSW инициализируется значением 0, PC инициализируется значением 2080H. Регистры в/в инициализируются как при аппаратном сбросе. На выводе RESET\ процессора возникает импульс сброса.

PSW ← Д 0
PC ← Д 2080H
Ассемблер: RST
Код: [11111111]
Байты: 1
Циклы: 16
Флаги: Z N C V VT ST
0 0 0 0 0 0

86 SCALL - короткий вызов п/п

Действие: содержимое PC (адрес возврата 0 помещается в стек, затем к PC прибавляется смещение (-1024...+1023)).

SP ← Д SP - 2
(SP) ←Д PC
PC ← Д PC + disp (расширенное до 16 бит)
Ассемблер: SCALL cadd
Код: [00101xxx] [disp - low]
(где xxx - три старших бита смещения)
Байты: 2
Циклы: внутренний стек: 13
  внешний стек: 16
Флаги: Z N C V VT ST

87 SETC - установить флаг переноса

Действие: флаг переноса устанавливается.

C ←Д 1
Ассемблер: SETC
Код: [11111001]
Байты: 1
Циклы: 4
Флаги: Z N C V VT ST
1

88 SHL - сдвиг слова влево

Действие: операнд назначения (левый) длиной 1 слово сдвигается влево столько раз, сколько определено операндом - счетчиком (правым). Счетчиком может быть непосредственное значение от 0 до 15 или содержимое регистра. Правые биты результата заполняются нулями. Последний выдвинутый бит запоминается в флаге переноса.

Temp ←Д (COUNT)
do while Temp ↔ 0
C ←Д старший бит (DEST)
(DEST) ←Д (DEST) * 2
Temp ←Д Temp - 1
end while
Ассемблер: SHL wreg, #count или SHL wreg, breg
Код: [00001001] [cnt/breg] [wreg]
Байты: 3
Циклы: 7 + число сдвигов, если число сдвигов равно 0, то 8 циклов
Флаги: Z N C V VT ST
+ ? + + ↑ -

89 SHLB - сдвиг байта влево

Действие: то же, что SHL, но для операнда назначения длиной 1 байт.

Temp ←Д (COUNT)
do while Temp ↔0
C ←Д старший бит (DEST)
(DEST) ←Д (DEST) * 2
Temp ←Д Temp - 1
end while
Ассемблер: SHLB breg, #count
или SHLB breg, breg
Код: [00011001] [cnt/breg] [breg]
Байты: 3
Циклы: 7 + число сдвигов, если число сдвигов равно 0, то 8 циклов
Флаги: Z N C V VT ST
+ ? ++ ↑ -

90 SHLL - сдвиг двойного слова влево

Действие: то же, что и SHL, но для операнда назначения длиной 2 слова.

Temp ← Д (COUNT)
do while Temp ↔0
C ← Д старший бит (DEST)
(DEST) ← Д (DEST) * 2
Temp ← Д Temp - 1
end while
Ассемблер: SHLL lreg, #count
или SHLL lreg, breg
Код: [00001101] [cnt/breg] [lreg]
Байты: 3
Циклы: 7 + число сдвигов,
если число сдвигов - 0, то 8 циклов
Флаги: Z N C V VT ST
+ ?+ + ↑ -

91 SHR - логический сдвиг вправо слова

Действие: операнд назначения (левый) типа WORD сдвигается вправо столько раз, сколько определено операндом - счетчиком (правым). Операнд - счетчик может быть определен как непосредственное значение в диапазоне 0...15 или как содержимое регистра. Левые биты результата заполняются нулями. Последний выдвинутый бит помещается во флаг переноса. Флаг ST обнуляется в начале исполнения, а устанавливается на следующем сдвиге после сдвига, во время которого была выдвинута 1.

Temp ← Д (COUNT)
do while Temp ↔ 0
C ← Д младший бит (DEST)
(DEST) ← Д (DEST)/2 (беззнаковое деление)
Temp ← Д Temp - 1 end while
Ассемблер: SHR wreg, #count
или SHR wreg, breg
Код: [00001000] [cnt/breg] [wreg]
Байты: 3
Циклы: 7 + число выполненных сдвигов
0 сдвигов займет 8 циклов
Флаги: Z N C V VT ST
+ 0 + 0 - +

92 SHRA - арифметический сдвиг вправо слова

Действие: операнд назначения (левый) типа INTEGER сдвигается вправо столько раз, сколько определено операндом - счетчиком (правым). Операнд - счетчик может быть определен как непосредственное значение в диапазоне 0...15, или как содержимое регистра. Если в начале операции старший бит был равен 0, то вдвигаются нули. Если в начале операции старший бит операнда назначения был равен 1, то вдвигаться будут единицы. Последний выдвинутый бит остается во флаге переноса. Флаг ST обнуляется в начале операции, устанавливается на следующем сдвиге после того, как была выдвинута единица.

Temp ←Д (COUNT)
do while Temp ↔ 0
C <Д младший бит (DEST)
(DEST) ←Д (DEST)/2 (знаковое деление)
TEMP ←Д TEMP -1
end while
Ассемблер: SHRA wreg, #count, или SHRA wreg, breg
Код: [00001010] [cnt/breg] [wreg]
Байты: 3
Циклы: 7 + число сдвигов, но не менее 8
Флаги: Z N C V VT ST
+ + + 0 - +

93 SHRAB - арифметический сдвиг вправо байта

Действие: то же, что у SHRA, но для операнда назначения типа SHORT-INTEGER.

Temp ←Д (COUNT)
do while Temp ↔0
C ←Д младший бит (DEST)
(DEST) ←Д (DEST)/2 (знаковое деление)
Temp ←Д Temp -1
end while
Ассемблер: SHRAB breg, #count
или SHRAB breg, breg

Код: [00011010] [cnt/breg] [breg]
Байты: 3
Циклы: 7 + число выполненных сдвигов, но не менее 8
Флаги: Z N C V VT ST
+ + + 0 - +

94 SHRAL - арифметический сдвиг вправо двойного

Действие: то же, что и SHRA, но для операнда назначения типа LONG-INTEGER.

Temp ←Д (COUNT)
do while Temp ↔0
C ←Д младший бит (DEST)
(DEST) ←Д (DEST)/2 (знаковое деление)
Temp ←Д Temp -1
end while
Ассемблер: SHRAL lreg, #count
или SHRAL lreg, breg
Код: [00001110] [cnt/breg] [lreg]
Байты: 3
Циклы: 7 + число выполненных сдвигов, но не менее 8
Флаги: Z N C V VT ST
+ + + 0 - +

95 SHRB - логический сдвиг вправо байта

Действие: то же, что и SHR, но для операнда назначения типа BYTE.

Temp ←<Д (COUNT)
do while Temp ↔0
C ←Д младший бит (DEST)
(DEST) ←Д (DEST)/2 (беззнаковое деление)
Temp ←Д Temp -1
end while
Ассемблер: SHRB breg, #count
или SHRB breg, breg
Код: [00011000] [cnt/breg] [breg]
Байты: 3
Циклы: 7 + число выполненных циклов, но не менее 8
Флаги: Z N C V VT ST
+ 0 + 0 - +

96 SHRL - логический сдвиг вправо двойного слова

Действие: то же, что и SHR, но для операнда назначения типа DOUBLE-WORD.

Temp ←Д (COUNT)
do while Temp ↔ 0
C ←Д младший бит (DEST)
(DEST) ←Д (DEST)/2 (беззнаковое деление)
Temp ←Д Temp -1
end while
Ассемблер: SHRL lreg, #count
или SHRL lreg, breg
Код: [00001100] [cnt/breg] [lreg]
Байты: 3
Циклы: 7 + число выполненных циклов, но не менее 8
Флаги: Z N C V VT ST
+ 0 + 0 - +

97 SJMP - короткий переход

Действие: к программному счетчику прибавляется смещение (-1024...1023).

PC ←Д PC + disp (расширенное до 16 бит)
Ассемблер: SJMP cadd
Код: [00100xxx] [disp-low], где xxx - три старших бита смещения
Байты: 2
Циклы: 8
Флаги: Z N C V VT ST

98 SKIP - два байта "нет операции"

Действие: ничего не делается. То же самое, что и две инструкции NOP, но второй байт может иметь любое значение, он просто игнорируется.

Ассемблер: SKIP breg
Код: [00000000] [breg]
Байты: 2
Циклы: 4
Флаги: Z N C V VT ST

99 ST - запомнить слово

Действие: значение левого операнда - слова помещается в правый операнд.
(DEST) <Д (SRC)
Ассемблер: SRC DST
ST wreg, waop
Код: [110000aa] [waop] [wreg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST

100 STB - запомнить байт

Действие: значение левого операнда - байта помещается в правый операнд.

(DEST) ←Д (SRC)
Ассемблер: SRC DST
STB breg, baop
Код: [110001aa] [baop] [breg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST

101 SUB (два операнда) - вычитание слов

Действие: значение операнда - источника (правого), длиной 1 слово, вычитается из значения операнда - приемника (левого), длиной 1 слово. Результат помещается в операнд - приемник. Флаг переноса устанавливается в значение, инверсное заему.

(DEST) ←Д (DEST) - (SRC)
Ассемблер: DST SRC
SUB wreg, waop
Код: [011010aa] [waop] [wreg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST
+ + + + ↑ -

102 SUB (три операнда) - вычитание слов

Действие: значение третьего операнда - слова вычитается из значения второго операнда - слова, результат помещается в операнд назначения. Флаг переноса устанавливается в значение, инверсное заему.

(DEST) ←Д (SRC1) - (SRC2)
Ассемблер: DST SRC1 SRC2
SUB wreg, wreg, waop
Код: [010010aa] [waop] [Swreg] [Dwreg]
Байты: 3 + BEA
Циклы: 5 + CEA
Флаги: Z N C V VT ST
+ + + + ↑ -

103 SUBB (два операнда) - вычитание байтов

Действие: то же, что и SUB, но для байт (два операнда).

(DEST) ←Д (DEST) - (SRC)
Ассемблер: DST SRC
SUBB breg, baop
Код: [011110aa] [baop] [breg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST
+ + + + ↑ -

104 SUBB (три операнда) - вычитание байтов

Действие: то же, что и SUB (три операнда), но для байтов.

(DEST) ←Д (SRC1) - (SRC2)
Ассемблер: DST SRC1 SRC2
SUBB breg, Sbreg, baop
Код: [010110aa] [baop] [Sbreg] [Dbreg]
Байты: 3 + BEA
Циклы: 5 + CEA
Флаги: Z N C V VT ST
+ + + + ↑ -

105 SUBC - вычитание слов с учетом заема

Действие: операнд - источник (правый 0 длиной 1 слово, вычитается из операнда - приемника длиной 1 слово (левого). Если флаг C очищен, то из этого результата вычитается 1. Результат помещается в операнд - приемник. Флаг переноса C устанавливается в значение, инверсное заему.

(DEST) ←Д (DEST) - (SRC) - (1 - C)
Ассемблер: DST SRC
SUBC wreg, waop
Код: [101010aa] [waop] [wreg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST
+ + + + ↑ -

106 SUBCB - вычитание байтов с учетом заема

Действие: то же, что и SUBC, но для байтов.

(DEST) ←Д (DEST) - (SRC) - (1 - C)
Ассемблер: DST SRC
SUBCB breg, baop
Код: [101110aa] [baop] [breg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST
+ + + + ↑ -

107 TIJMP - команда отсутствует

108 TRAP - программное прерывание

Действие: происходит вызов п/п прерывания, вектор которой находится по адресу 2010H. На эту инструкцию не влияет состояние флага разрешения прерываний в PSW. Вызов другой п/п прерывания невозможен сразу после исполнения этой инструкции. Эта инструкция предназначена для использования в Intel Development Tools. Эти средства не поддерживают использование инструкции TRAP пользователем.

SP ←Д SP - 2
(SP) ←Д PC
PC ←Д (2010H)
Ассемблер: в версии 1.0 Ассемблера для 8096 не поддерживается.
Код: [11110111]
Байты: 1
Циклы: внутренний стек 21
внешний стек 24
Флаги: Z N C V VT ST

109 XOR - логическое исключающее "ИЛИ" слов

Действие: операнд - источник (правый 0 длиной 1 слово подвергается побитному исключающему "ИЛИ" с операндом - приемником (левым) длиной 1 слово. Результат помещается в операнд - приемник.

(DEST) ←Д (DEST) XOR (SRC)
Ассемблер: DST SRC
XOR wreg, waop
Код: [100001aa] [waop] [wreg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST
+ + 0 0 - -

110 XCH - замена слова- команда отсутствует

111 XCHB - замена байта- команда отсутствует

112 XORB - логическое исключающее "ИЛИ" байтов

Действие: то же, что XOR, но для байтов.

(DEST) ←Д (DEST) XOR (SRC)
Ассемблер: DST SRC
XORB breg, baop
Код: [100101aa] [baop] [breg]
Байты: 2 + BEA
Циклы: 4 + CEA
Флаги: Z N C V VT ST
+ + 0 0 - -

Заключение

В настоящем руководстве КФДЛ.431295.008 приведено подробное описание архитектуры, функционального построения, системы команд и особенностей применения ИС Л1874ВЕ36, 1874ВЕ36, Л1874ВЕ36А, 1874ВЕ36А, которые представляют собой СБИС однокристальных 16-разрядных микроконтроллеров с внутренней программной памятью объемом 8 Кбайт. Микросхемы предназначены для применения в системах управления, в области промышленного производства. ИС могут быть использованы для управления робототехническими комплексами, в системах автоматизации технологических процессов, в системах автоматизированного управления электроприводом, телекоммуникационной технике, оргтехнике, вычислительной технике и т. п.

Все значения электрических параметров ИС приведены в технических условиях АЕЯР.431280.169 на изделия, значения параметров, приведенные в техническом описании, являются справочными.

Настоящее руководство может служить практическим пособием по применению микроконтроллеров для разработчиков систем на основе ИС Л1874ВЕ36, 1874ВЕ36, Л1874ВЕ36А, 1874ВЕ36А и программистов.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Номер изме-нения	Номер страниц				Номер доку-мента	Дата внесе-ния изм.	Дата введе-ния изм.	Всего лис-тов в док.	Под пись
	изме-нен-ных	заменен-ных	новых	Анну-лиро-ван.					
-	-	-	все			27.08.02		120	
1		56,57,60,61,87				10.06.05		-	
2		Тит. лист,2-13,15,16,17,19-27,29-33,40-46,48,49,53,54,57,58,59,64-68,76,78,79,81,82,83,85-92,119		Лист ут-верж-де-ния		30.08.12		120	