МИКРОСХЕМЫ ИНТЕГРАЛЬНЫЕ

1874ВЕ76Т, 1874ВЕ06Т, 1874ВЕ05Т

Руководство пользователя

2008

Содержание

Введение 8

1 Назначение и область применения 9

2 Краткое техническое описание ИМС 1874BE76T, 1874BE06T, 1874BE05T 10

2.1 Функциональные параметры: 10
2.2 Электрические параметры микросхем 20

3 Архитектура изделий 23

3.1 Особенности архитектуры 24
3.2 Краткое описание функционирования микросхем комплекта 25
3.2.1 Процессорное ядро 25
3.2.2 Способы адресации 27
3.2.3 Прерывания 27
3.2.4 Периферийные устройства 28
3.2.5 Синхронизация 29а

4 Типы данных и адресация 31

4.1 Типы операндов 30
4.2 Режимы адресации 32
4.2.3 Косвенная адресация с автоинкрементом 32
4.2.4 Непосредственная адресация 33
4.2.5 Короткая Индексная адресация 33
4.2.6 Длинная Индексная адресация 33
4.2.7 Адресация с использованием Нулевого Регистра 34
4.2.8 Адресация с использованием Указателя Стека 34
4.3 Выборы способов адресации на языке ассемблера 34
4.3.1 Прямая адресация 34
4.3.2 Индексная адресация 34
4.4 Стандарты и соглашения программного обеспечения 35
4.4.1 Использование регистров 35
4.4.2 Адресация 32-битных операндов 35
4.4.3 Соединение подпроцедур 35
4.5 Защита и руководящие принципы программного обеспечения 36

5 Распределение памяти 37

5.1 Разделы внешней памяти 38
5.2 Порты 3 и 4 38
5.3 Программная память и память специального назначения 38
5.3.1 Выбор внутренней или внешней памяти 38
5.3.2 Программная память 38
5.3.3 Память специального назначения 38
5.4 Регистровый файл 40
5.4.1 Регистровое ОЗУ общего назначения 41
5.4.2 Указатель стека (SP) 41
5.4.3 Регистры специальных функций (SFRs) 42
5.5 Создание горизонтальных окон 42
5.5.1 Выбор горизонтального окна 42
5.5.2 Горизонтальное окно 0 (HWindow 0) 43
5.5.3 Горизонтальное окно 1 (HWindow 1) 43

5.5.4 Горизонтальное окно 15 (HWindow 15) 43
5.6 Создание вертикальных окон 44
5.6.1 Выбор вертикального окна 45
5.6.2 Создание вертикальных окон и способы адресации 45

6 Прерывания 46

6.1 Обработка прерываний 46
6.1.1 Контроллер прерываний 46
6.1.2 Периферийный Сервер (PTS) 46
6.2 Приоритеты прерываний 48
6.2.1 Модификация приоритетов прерываний 48
6.3 Синхронизация прерываний 51
6.3.1 Время ожидания прерываний 51
6.3.2 Вычисление времени ожидания 52
6.4 Специальные прерывания 53
6.4.1 Неподдерживаемые Коды 53
6.4.2 Программное прерывание 53
6.4.3 Немаскируемое прерывание (NMI) 53
6.5 Программирование прерываний 54
6.5.1 Выбор способа обработки прерываний: через PTS или стандартного 54
6.5.2 Разрешение PTS прерываний 54
6.5.3 Выбор источников прерываний 55
6.5.4 Регистры масок прерываний 56
6.5.5 Регистры ожидания прерываний 56
6.6 Блоки управления PTS 57
6.6.1 Регистр PTSCOUNT 58
6.6.2 Регистр PTSCON 58
6.7 Режим одиночной передачи 60
6.7.1 Пример режима одиночной передачи 60
6.8 Режим блочной передачи 60
6.8.1 Пример режима блочной передачи 61
6.9 Режим сканирования АЦП 61
6.9.1 PTS циклы в режиме сканирования АЦП 62
6.9.2 Режим сканирования АЦП. Пример 1 62
6.9.3 Режим сканирования АЦП. Пример 2 63
6.10 Режим HSI 64
6.10.1 Пример режима HSI 65
6.11 Режим HSO 65
6.11.1 Пример режима HSO 65

7 Порты ввода-вывода 66

7.1 Обзор функциональных возможностей 66
7.1.1 Входной вывод порта 67
7.1.2 Выходной вывод порта 67
7.1.3 Квазидвунаправленный вывод порта 68
7.1.4 Двунаправленный вывод порта с открытым стоком 69
7.2 Программирование портов I/O 70
7.2.1 Входной порт 72
7.2.2 Порт вывода 72
7.2.3 Доступ к портам 3 и 4 73
7.3 Аппаратное подключение к квазидвунаправленным портам 75

8 Последовательный порт ввода-вывода 77

8.1 Режим 0 (MODE 0) 77
8.2 Асинхронные режимы 79
8.2.1 Режим 1 79
8.2.2 Режим 2 79
8.2.3 Режим 3 80
8.2.4 Временные соотношения в Режимах 2 и 3 80
8.2.5 Мультипроцессорные связи 80
8.3 Программирование последовательного порта 80
8.3.1 Выбор режима связи и разрешение проверки чётности 81
8.3.2 Конфигурирование TXD и RXD 82
8.3.3 Разрешение работы по прерываниям для последовательного порта 82
8.3.4 Программирование скорости обмена и источника синхронизации 83
8.4 Состояние последовательного порта 84

9 Устройство высокоскоростного ввода-вывода 86

9.1 Таймеры 86
9.1.1 Функции таймера 1 86
9.1.2 Функции таймера 2 87
9.1.3. Программирование Модуля таймера 2 87
9.1.4 Использование Внешних Входов таймера 2 90
9.1.5 Прерывания таймера 91
9.1.6 Предосторожности при работе с таймером 91
9.2 Модуль высокоскоростного ввода HSI 92
9.2.1 Временные Соотношения для Событий в HSI 93
9.2.2 Чтение Информации о Событии 94
9.2.3 Программирование HSI Модуля 94
9.3 Модуль высокоскоростного вывода HSO 98
9.3.1 Функционирование HSO 98
9.3.2 Программирование HSO Модуля 99
9.3.3 Использование модуля HSO как широтно-импульсного модулятора (PWM) 103
9.3.4 Синхронизация Выхода HSO. 104

10 Аналого-цифровой преобразователь 105

10.1 Обзор функций АЦП 105
10.2 Программирование АЦП 106
10.2.1 Выбор времени запоминания и преобразования 106
10.2.2 Программирование IOC2 Регистра 107
10.2.3 Программирование регистра ADCOMMAND 108
10.3 Чтение результатов преобразования 109
10.4 Интерфейс с АЦ-преобразователем 109
10.4.1 Аналоговая земля и источник опорного напряжения
10.4.2 Совместное использование аналоговых и цифровых входов
10.5 Передаточная функция и источник ошибок АЦП

11 Широтно-импульсный модулятор (PWM) 115

11.1 Функциональный обзор PWM 115
11.2. Программирование рабочего цикла PWM 116

11.3 Разрешение выходов PWM 117
11.4 Формирование аналоговых выходов.....117

12 Специальные режимы работы 118

12.1 Режим IDLE 118
12.1.1 Вход в IDLE Режим 118
12.1.2 Выход из IDLE Режим 118
12.2 Режим POWERDOWN 118
12.2.1 Запрещение Режима POWERDOWN 119
12.2.2 Переход в Режим POWERDOWN 119
12.2.3 Выход из Режима POWERDOWN 120
12.3 Вход в режимы, зарезервированные для тестирования 122

13 Интерфейс внешней памяти 123

13.1 Сигналы интерфейса внешней памяти 123
13.2 Регистр конфигурации кристалла 125
13.3 Разрядность шины и конфигурация памяти 126
13.3.1 Временные требования для сигнала BW 127
13.3.2 Шина разрядностью 16 бит 128
13.3.3 Шина Разрядностью 8 Бит 128
13.4 Управление сигналом READY 129
13.4.1 Временные Требования для Сигнала READY 130
13.5 Протокол захвата шины 130
13.5.1 Разрешение Протокола Захвата Шины 131
13.5.2 Запрещение Протокола Захвата Шины 131
13.5.3 Скрытое состояние HOLD (Hold Latency) 132
13.5.4 Возврат Управления Шины 132
13.6 Режимы управления шиной 132
13.6.1 Стандартный режим управления шиной (Standart Bus - Control) 133
13.6.2 Режим Write Strobe Mode 134
13.6.3 Режим Address Valid Strobe 135
13.6.4 Режим Address Valid with Write Strobe 136
13.7 Временные соотношения системной шины 139
13.7.1 Перечень Принятых Сокращений 139

14 Программирование постоянной памяти 142

14.1 Общее описание 142
14.2 Режимы программирования 142
14.2.1 Выбор режима программирования 143
14.3 Функции выводов в режиме программирования 143
14.4 Распределение памяти 145
14.5 Включение и выключение питания 146
14.5.1 Последовательность включения питания 146
14.5.2 Последовательность выключения питания 146
14.6 Защита памяти 146
14.6.1 Биты кода защиты в CCR 147
14.6.2 Биты защиты UPROM 149
14.6.3 Ключ защиты 150
14.7 Модифицированный быстрый (QUICK_PULSE) алгоритм 151
14.8 Слово-сигнатура 151
14.9 Режим программирования AUTO 152
14.10 Режим программирования SLAVE 154
14.10.1 Режим SLAVE 155
14.10.2 Проверка ключа защиты в режиме SLAVE 156

14.10.3 Алгоритм режима SLAVE.....157
14.10.4 Команды Program Word и Dump Word 157
14.11 Режим ввода содержимого памяти (ROM-DUMP) 162
14.12 Режим RUN-TIME.....162

15 Рекомендации по отладочным средствам ИМС 163

15.1 Программаторы для программируемого варианта ИМС 163
15.2 Описание инструментальных средств для ИМС.....163
15.3 Интегрированный пакет разработки и отладки систем на базе микроконтроллеров. (PROJECT-96).....163
15.3.1 Внутрисхемный эмулятор PICE-196 163
15.4 Отладчик-симулятор микроконтроллеров семейства MCS-196 PDS-96 165
15.5 Кросс-макроассемблер МСА-96.....166
15.6 Кросс-компилятор языка Си МСС-96.....167

16 Заключение.....168

Приложение А (обязательное)
Система команд МК 1874ВЕ76Т (1874ВЕ06Т, 1874ВЕ05Т).....169

Приложение Б (обязательное) Описание сигналов 208

Приложение В (рекомендованное) Регистры……...225

Лист регистрации изменений…….....…….....….....…274

Введение

В настоящем руководстве КФДЛ.431295.019 приведено описание архитектуры, функционального построения, системы команд и особенностей применения микросхем 1874ВЕ76Т, 1874ВЕ06Т и 1874ВЕ05Т. Первые две микросхемы представляют собой СБИС однокристальных 16-разрядных микроконтроллеров с тактовой частотой 20 МГц, ОЗУ (488×8) бит, 8-канальным 10-разрядным АЦП, 3-канальным ШИМ и последовательным портом ввода-вывода, в том числе с внутренней памятью программ типа OTP EPROM (для ИМС 1874ВЕ76Т) объемом (16К×8) бит (функциональные аналоги TN87C196KC-20, TN80C196KC-20 фирмы Intel). Микросхема 1874ВЕ05Т представляет собой стойкий к специальным видам внешних воздействующих факторов вариант микроконтроллера 1874ВЕ06Т без АЦП.

Изделие 1874ВЕ05Т послужит цели развития отечественной элементной базы для применения в различных системах управления и радиосвязи.

Настоящее руководство может служить практическим пособием по применению микроконтроллеров для разработчиков систем на основе ИМС 1874ВЕ76Т, 1874ВЕ06Т и 1874ВЕ05Т.

1 Назначение и область применения

Архитектура микроконтроллеров ориентирована для создания управляющих систем, функционирующих в режиме реального времени с возможностью адаптации и модификации под конкретные приложения. Изделия могут служить элементной базой для систем управления различной аппаратурой, в том числе силовой электроникой, автомобильной техникой и т. д.

Наличие средств инструментальной отладки обеспечивает как эффективное проектирование систем на основе микроконтроллера, так и возможность смены алгоритма работы при создании модификаций систем.

Микросхемы представляют собой комплект высокопроизводительных 16-разрядных микроконтроллеров:

- 1874ВЕ76Т – 16-разрядный микроконтроллер со встроенной памятью программ типа OTP EPROM (16К×8) бит (аналог TN87C196KC-20 фирмы Intel);
- 1874ВЕ06Т – 16-разрядный микроконтроллер без встроенной памяти программ (функциональный аналог TN87C196KC-20 фирмы Intel), предназначенный для применения в системах встроенного управления;
- 1874ВЕ05Т – стойкий к специальным видам внешних воздействующих факторов 16-разрядный микроконтроллер с системой команд микроконтроллера 1874ВЕ36.

2 Краткое техническое описание ИМС 1874ВЕ76Т, 1874ВЕ06Т, 1874ВЕ05Т

2.1 Функциональные параметры:

- разрядность данных, бит 16;
- тактовая частота, МГц 20;
- динамически конфигурируемая шина данных, бит 8 или 16;
- встроенная память программ (однократно программируемая) типа OTP EPROM (только для микросхемы 1874ВЕ76Т. В микросхемах 1874ВЕ06Т и 1874ВЕ05Т отсутствует), бит 16К×8;
- регистровое ОЗУ, бит 488×8;
- адресуемая память, бит 64К×8;
- число источников прерывания 28;
- число параллельных 8-разрядных портов ввода-вывода 5;
- разрядность сторожевого таймера 16;
- число таймеров/счетчиков 2;
- разрядность таймеров/счетчиков 16;
- универсальный последовательный порт 1;
- устройство высокоскоростного ввода-вывода 4/6;
- число разрядов 8-канального аналого-цифрового преобразователя (только для микросхем 1874ВЕ06Т и 1874ВЕ76Т, в микросхеме 1874ВЕ05Т отсутствует) 8, 10;
- блок ШИМ сигналов 3;
- периферийный сервер (PTS) 1.

Микросхемы выполнены в металлокерамическом планарном корпусе с четырехсторонним расположением выводов 4235.88-1 и предназначены для ручной и автоматической сборки в соответствии с ГОСТ РВ 20.39.412-97. Масса микросхемы не более 4,5 г.

Герметизация микросхем осуществляется шовной роликовой сваркой. Показатель герметичности корпуса по скорости утечки гелия не более 5×10^{-3} Па·см^3/с.

Микросхемы не имеют собственных резонансных частот ниже 100 Гц.

Структурная схема микросхем 1874ВЕ76Т, 1874ВЕ06Т приведена на рисунке 2.1. Структурная схема микросхемы 1874ВЕ05Т приведена на рисунке 2.2.

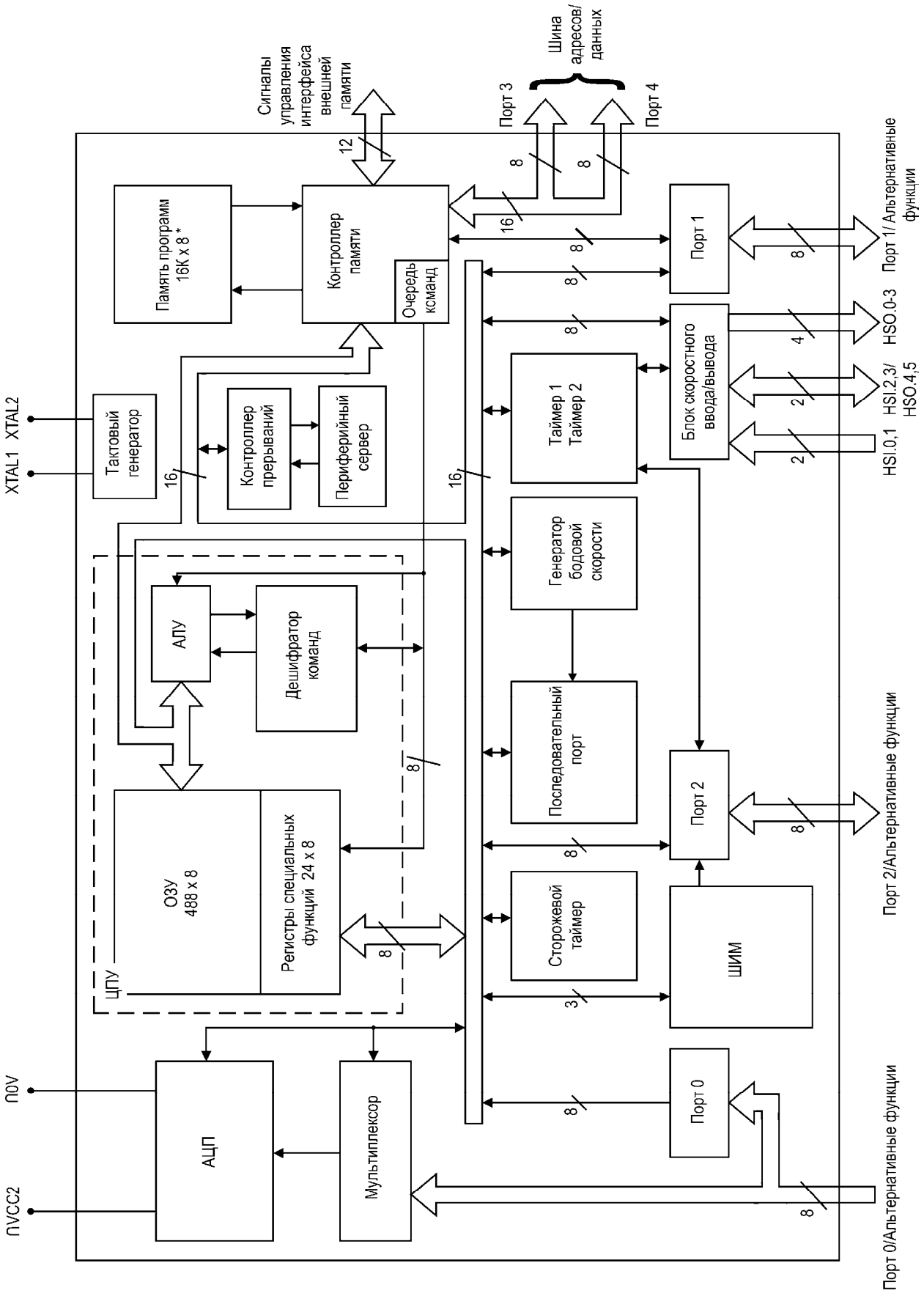


Рисунок 2.1 – Структурная схема микроконтроллеров 1874ВЕ76Т и 1874ВЕ06Т

* Только для 1874ВЕ76Т (со встроенной памятью программ)

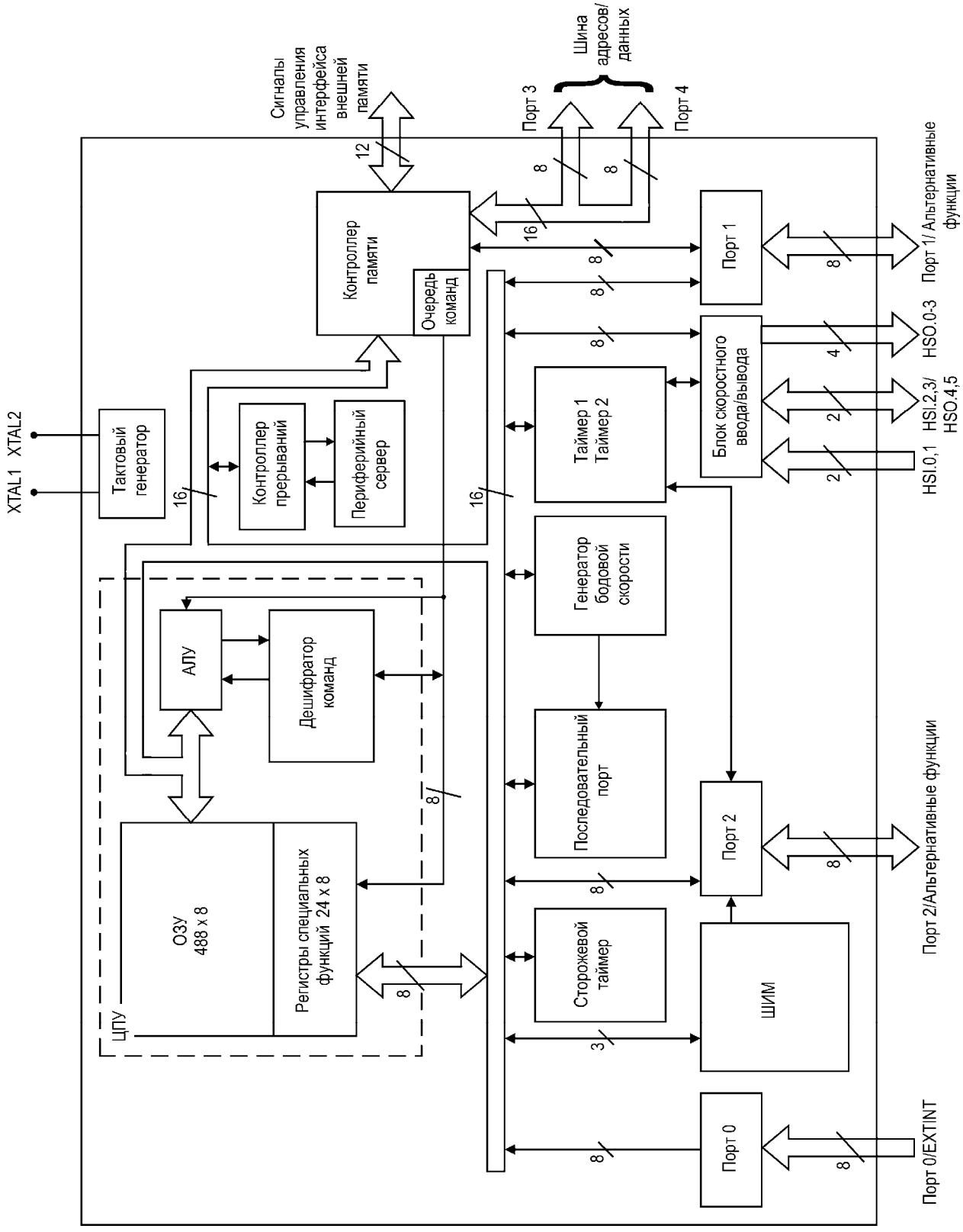


Рисунок 2.2 – Структурная схема микроконтроллера 1874ВЕ05Т

Условные графические обозначения микросхем приведены на рисунках 2.3 и 2.4. Функциональное назначение выводов микросхем приведено в таблице 2.1.

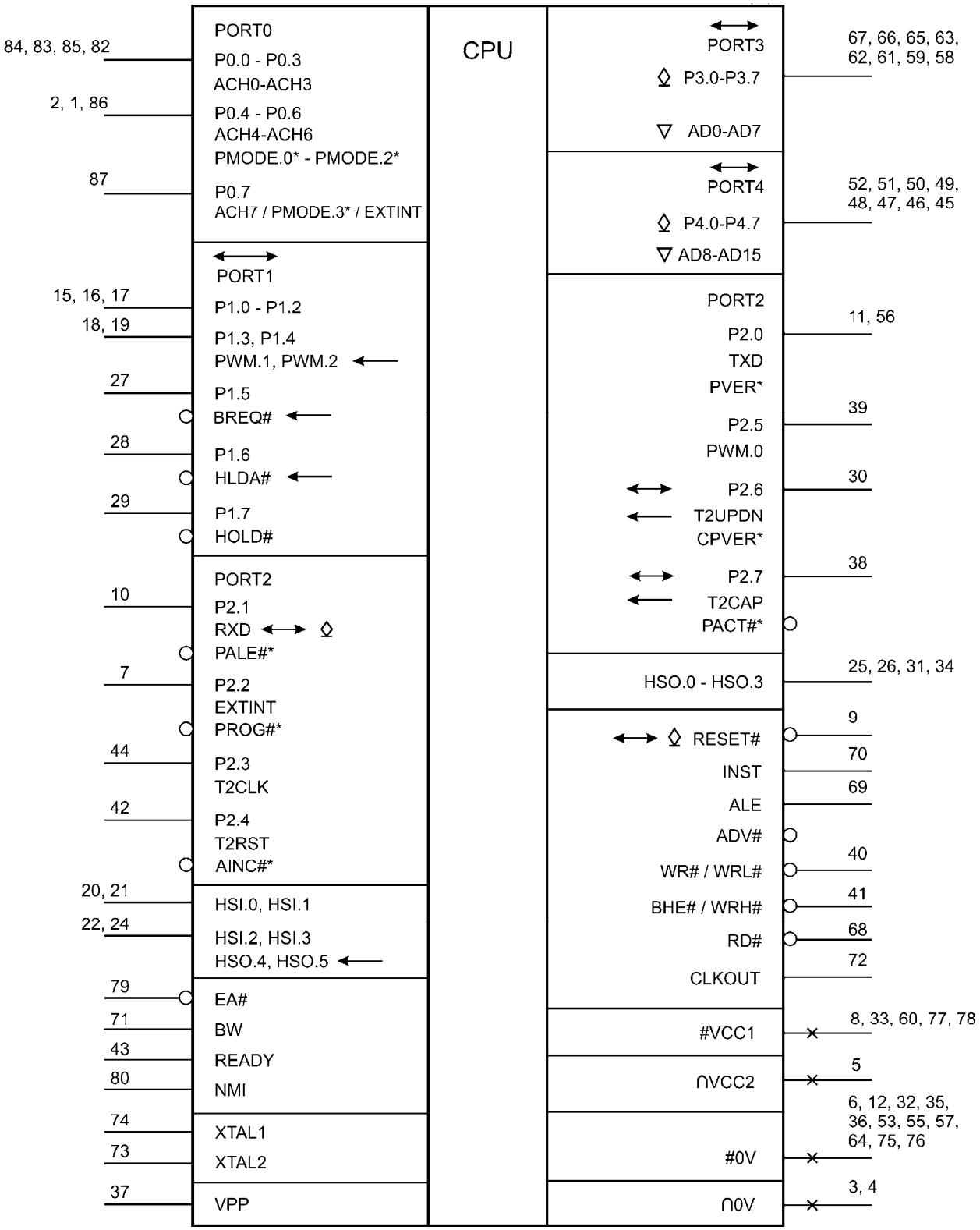


Рисунок 2.3 – Условное графическое изображение микросхем 1874ВЕ76Т, 1874ВЕ06Т

---

* Только для микросхем 1874ВЕ76Т

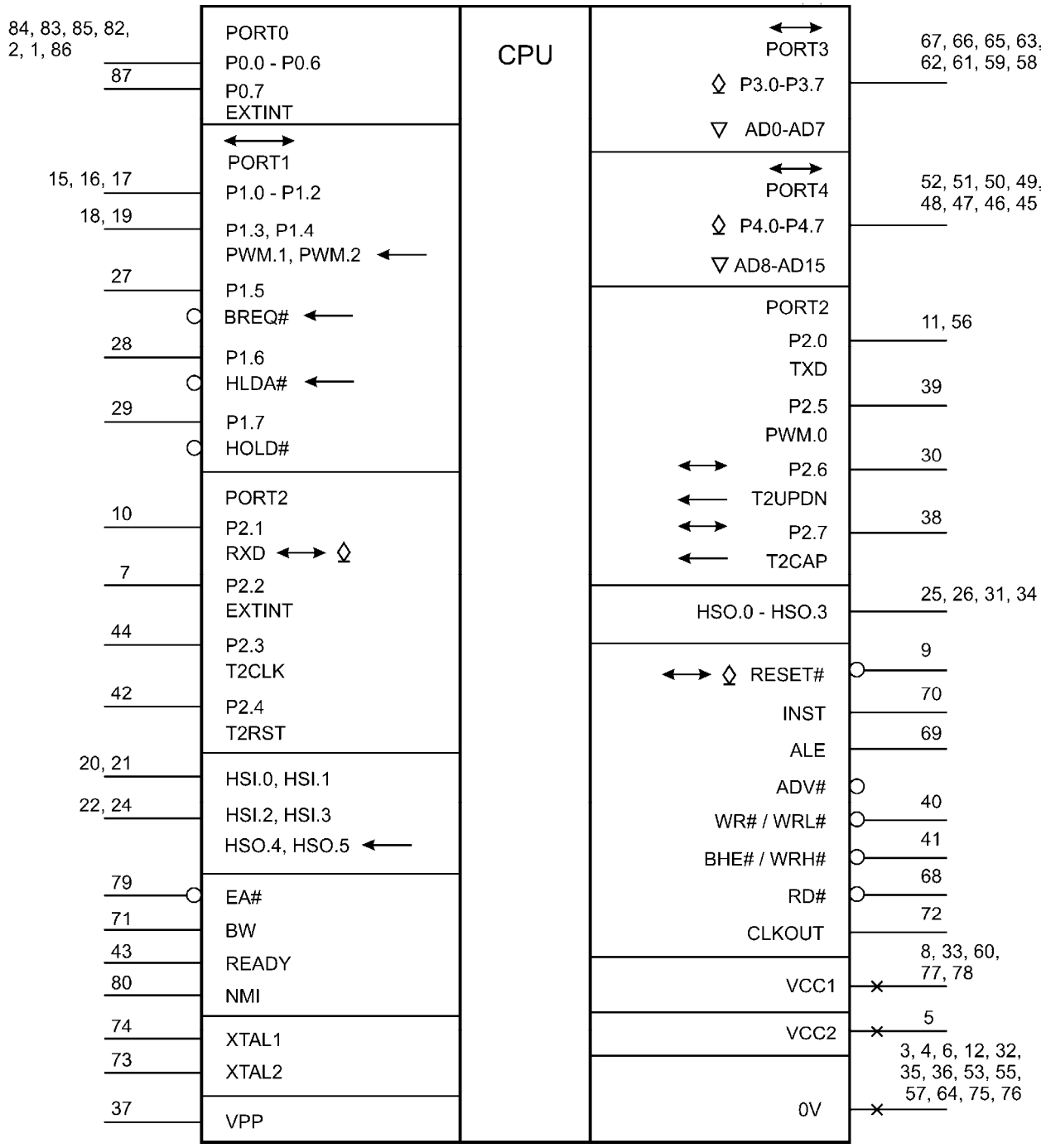


Рисунок 3.2 – Условное графическое изображение микросхем 1874ВЕ05Т

Таблица 2.1 – Функциональное назначение выводов микросхем

Обозначение вывода	Номер вывода	Функциональное назначение	Тип вывода	Обозначение альтернативной функции вывода
1	2	3	4	5
Для микросхем 1874BE76T и 1874BE06T				
P0.0	84	Вход "порт 0, 0 разряд" Вход АЦП, канал 0	I I	 ACH0
P0.1	83	Вход "порт 0, 1 разряд" Вход АЦП, канал 1	I I	 ACH1
P0.2	85	Вход "порт 0, 2 разряд" Вход АЦП, канал 2	I I	 ACH2
P0.3	82	Вход "порт 0, 3 разряд" Вход АЦП, канал 3	I I	 ACH3
P0.4	2	Вход "порт 0, 4 разряд" Вход АЦП, канал 4 Вход "режим программирования, 0 разряд"*	I I I	 ACH4 PMODE.0*
P0.5	1	Вход "порт 0, 5 разряд" Вход АЦП, канал 5 Вход "режим программирования, 1 разряд"*	I I I	 ACH5 PMODE.1*
P0.6	86	Вход "порт 0, 6 разряд" Вход АЦП, канал 6 Вход "режим программирования, 2 разряд"*	I I I	 ACH6 PMODE.2*
P0.7	87	Вход "порт 0, 7 разряд" Вход АЦП, канал 7 Вход "режим программирования, 3 разряд"* Вход "сигнал внешнего прерывания"	I I I I	 ACH7 PMODE.3* EXTINT
P1.0	15	Вход-выход "порт 1, 0 разряд"	I/O	
P1.1	16	Вход-выход "порт 1, 1 разряд"	I/O	
P1.2	17	Вход-выход "порт 1, 2 разряд"	I/O	
P1.3	18	Вход-выход "порт 1, 3 разряд" Выход "ШИМ 1"	I/O O	 PWM.1
P1.4	19	Вход-выход "порт 1, 4 разряд" Выход "ШИМ 2"	I/O O	 PWM.2
P1.5	27	Вход-выход "порт 1, 5 разряд" Выход "запрос шины"	I/O O	 BREQ#
P1.6	28	Вход-выход "порт 1, 6 разряд" Выход "подтверждение захвата шины"	I/O O	 HLDA#
P1.7	29	Вход-выход "порт 1, 7 разряд" Вход "требование захвата шины"	I/O I	 HOLD#
P2.0	11, 56	Выход "порт 2, 0 разряд" Выход последовательных данных Выход "верификация"*	O O O	 TXD PVER *
P2.1	10	Вход "порт 2, 1 разряд" Вход-выход последовательных данных Вход "строб записи"*	I I/O/2 I	 RXD PALE# *
P2.2	7	Вход "порт 2, 2 разряд" Вход "сигнал внешнего прерывания" Вход "программирование"*	I I I	 EXTINT PROG# *
P2.3	44	Вход "порт 2, 3 разряд" Вход "синхронизация таймера 2"	I I	 T2CLK

Продолжение таблицы 2.1

1	2	3	4	5
P2.4	42	Вход "порт 2, 4 разряд" Вход "сброс таймера 2" Вход "автоинкремент"*	I I I	 T2RST AINC# *
P2.5	39	Выход "порт 2, 5 разряд" Выход "ШИМ 0"	O O	 PWM.0
P2.6	30	Вход-выход "порт 2, 6 разряд" Вход "выбор режима таймера 2" Выход "ошибка программирования"*	I/O I O	 T2UPDN CPVER *
P2.7	38	Вход-выход "порт 2, 7 разряд" Вход "выборка данных таймера 2" Выход "подтверждение программирования"*	I/O I O	 T2CAP PACT# *
P3.0	67	Вход-выход "порт 3, 0 разряд" Вход-выход "адрес-данные, 0 разряд"	I/O/2 I/O/Z	 AD0
P3.1	66	Вход-выход "порт 3, 1 разряд" Вход-выход "адрес-данные, 1 разряд"	I/O/2 I/O/Z	 AD1
P3.2	65	Вход-выход "порт 3, 2 разряд" Вход-выход "адрес-данные, 2 разряд"	I/O/2 I/O/Z	 AD2
P3.3	63	Вход-выход "порт 3, 3 разряд" Вход-выход "адрес-данные, 3 разряд"	I/O/2 I/O/Z	 AD3
P3.4	62	Вход-выход "порт 3, 4 разряд" Вход-выход "адрес-данные, 4 разряд"	I/O/2 I/O/Z	 AD4
P3.5	61	Вход-выход "порт 3, 5 разряд" Вход-выход "адрес-данные, 5 разряд"	I/O/2 I/O/Z	 AD5
P3.6	59	Вход-выход "порт 3, 6 разряд" Вход-выход "адрес-данные, 6 разряд"	I/O/2 I/O/Z	 AD6
P3.7	58	Вход-выход "порт 3, 7 разряд" Вход-выход "адрес-данные, 7 разряд"	I/O/2 I/O/Z	 AD7
P4.0	52	Вход-выход "порт 4, 0 разряд" Вход-выход "адрес-данные, 8 разряд"	I/O/2 I/O/Z	 AD8
P4.1	51	Вход-выход "порт 4, 1 разряд" Вход-выход "адрес-данные, 9 разряд"	I/O/2 I/O/Z	 AD9
P4.2	50	Вход-выход "порт 4, 2 разряд" Вход-выход "адрес-данные, 10 разряд"	I/O/2 I/O/Z	 AD10
P4.3	49	Вход-выход "порт 4, 3 разряд" Вход-выход "адрес-данные, 11 разряд"	I/O/2 I/O/Z	 AD11
P4.4	48	Вход-выход "порт 4, 4 разряд" Вход-выход "адрес-данные, 12 разряд"	I/O/2 I/O/Z	 AD12
P4.5	47	Вход-выход "порт 4, 5 разряд" Вход-выход "адрес-данные, 13 разряд"	I/O/2 I/O/Z	 AD13
P4.6	46	Вход-выход "порт 4, 6 разряд" Вход-выход "адрес-данные, 14 разряд"	I/O/2 I/O/Z	 AD14
P4.7	45	Вход-выход "порт 4, 7 разряд" Вход-выход "адрес-данные, 15 разряд"	I/O/2 I/O/Z	 AD15
HSI.0	20	Вход "быстрый ввод, канал 0"	I	
HSI.1	21	Вход "быстрый ввод, канал 1"	I	
HSI.2	22	Вход "быстрый ввод, канал 2" Выход "быстрый вывод, канал 4"	I O	 HSO.4
HSI.3	24	Вход "быстрый ввод, канал 3" Выход "быстрый вывод, канал 5"	I O	 HSO.5
HSO.0	25	Выход "быстрый вывод, канал 0"	O	
HSO.1	26	Выход "быстрый вывод, канал 1"	O	

Продолжение таблицы 2.1

1	2	3	4	5
HSO.2	31	Выход "быстрый вывод, канал 2"	O	
HSO.3	34	Выход "быстрый вывод, канал 3"	O	
EA#	79	Вход "внешний доступ"	I	
BW	71	Вход "разрядность внешней шины"	I	
READY	43	Вход "готовность"	I	
NMI	80	Вход "немаскируемое прерывание"	I	
RESET#	9	Вход-выход "сброс"	I/O/2	
INST	70	Выход "чтение команды"	O	
ALE	69	Выход "разрешение записи адреса" Выход "адрес действителен"	O O	 ADV#
WR#	40	Выход "запись" Выход "запись младшего байта"	O O	 WRL#
BHE#	41	Выход "разрешение записи старшего байта" Выход "запись старшего байта"	O O	 WRH#
RD#	68	Выход "чтение"	O	
CLKOUT	72	Выход "системный тактовый сигнал"	O	
XTAL1	74	Вывод подключения кварцевого резонатора/ вход тактового сигнала	–	
XTAL2	73	Вывод подключения кварцевого резонатора	–	
VPP	37	Вход "возврат из режима пониженного потребления" Вывод "напряжение программирования"	I –	
#VCC1	8, 33, 60, 77, 78	Питание 5 В цифровой части микросхемы	–	
∩VCC2	5	Питание 5 В аналоговой части микросхемы	–	
#0V	6, 12, 32, 35, 36, 53, 55, 57, 64, 75, 76	Общий вывод цифровой части микросхемы	–	
∩0V	3, 4	Общий вывод аналоговой части микросхемы	–	
* Только для микросхемы 1874ВЕ76Т.				
Для микросхемы 1874ВЕ05Т				
1	2	3	4	5
P0.0	84	Вход "порт 0, 0 разряд"	I	
P0.1	83	Вход "порт 0, 1 разряд"	I	
P0.2	85	Вход "порт 0, 2 разряд"	I	
P0.3	82	Вход "порт 0, 3 разряд"	I	
P0.4	2	Вход "порт 0, 4 разряд"	I	
P0.5	1	Вход "порт 0, 5 разряд"	I	
P0.6	86	Вход "порт 0, 6 разряд"	I	
P0.7	87	Вход "порт 0, 7 разряд" Вход "сигнал внешнего прерывания"	I I	 EXTINT
P1.0	15	Вход-выход "порт 1, 0 разряд"	I/O	
P1.1	16	Вход-выход "порт 1, 1 разряд"	I/O	
P1.2	17	Вход-выход "порт 1, 2 разряд"	I/O	
P1.3	18	Вход-выход "порт 1, 3 разряд" Выход "ШИМ 1"	I/O O	 PWM.1
P1.4	19	Вход-выход "порт 1, 4 разряд" Выход "ШИМ 2"	I/O O	 PWM.2

Продолжение таблицы 2.1

1	2	3	4	5
P1.5	27	Вход-выход "порт 1, 5 разряд" Выход "запрос шины"	I/O O	 BREQ#
P1.6	28	Вход-выход "порт 1, 6 разряд" Выход "подтверждение захвата шины"	I/O O	 HLDA#
P1.7	29	Вход-выход "порт 1, 7 разряд" Вход "требование захвата шины"	I/O I	 HOLD#
P2.0	11, 56	Выход "порт 2, 0 разряд" Выход последовательных данных	O O	 TXD
P2.1	10	Вход "порт 2, 1 разряд" Вход-выход последовательных данных	I I/O/2	 RXD
P2.2	7	Вход "порт 2, 2 разряд" Вход внешнего прерывания	I I	 EXTINT
P2.3	44	Вход "порт 2, 3 разряд" Вход "синхронизация таймера 2"	I I	 T2CLK
P2.4	42	Вход "порт 2, 4 разряд" Вход "сброс таймера 2"	I I	 T2RST
P2.5	39	Выход "порт 2, 5 разряд" Выход "ШИМ 0"	O O	 PWM.0
P2.6	30	Вход-выход "порт 2, 6 разряд" Вход "выбор режима таймера 2"	I/O I	 T2UPDN
P2.7	38	Вход-выход "порт 2, 7 разряд" Вход "выборка данных таймера 2"	I/O I	 T2CAP
P3.0	67	Вход-выход "порт 3, 0 разряд" Вход-выход "адрес-данные, 0 разряд"	I/O/2 I/O/Z	 AD0
P3.1	66	Вход-выход "порт 3, 1 разряд" Вход-выход "адрес-данные, 1 разряд"	I/O/2 I/O/Z	 AD1
P3.2	65	Вход-выход "порт 3, 2 разряд" Вход-выход "адрес-данные, 2 разряд"	I/O/2 I/O/Z	 AD2
P3.3	63	Вход-выход "порт 3, 3 разряд" Вход-выход "адрес-данные, 3 разряд"	I/O/2 I/O/Z	 AD3
P3.4	62	Вход-выход "порт 3, 4 разряд" Вход-выход "адрес-данные, 4 разряд"	I/O/2 I/O/Z	 AD4
P3.5	61	Вход-выход "порт 3, 5 разряд" Вход-выход "адрес-данные, 5 разряд"	I/O/2 I/O/Z	 AD5
P3.6	59	Вход-выход "порт 3, 6 разряд" Вход-выход "адрес-данные, 6 разряд"	I/O/2 I/O/Z	 AD6
P3.7	58	Вход-выход "порт 3, 7 разряд" Вход-выход "адрес-данные, 7 разряд"	I/O/2 I/O/Z	 AD7
P4.0	52	Вход-выход "порт 4, 0 разряд" Вход-выход "адрес-данные, 8 разряд"	I/O/2 I/O/Z	 AD8
P4.1	51	Вход-выход "порт 4, 1 разряд" Вход-выход "адрес-данные, 9 разряд"	I/O/2 I/O/Z	 AD9
P4.2	50	Вход-выход "порт 4, 2 разряд" Вход-выход "адрес-данные, 10 разряд"	I/O/2 I/O/Z	 AD10
P4.3	49	Вход-выход "порт 4, 3 разряд" Вход-выход "адрес-данные, 11 разряд"	I/O/2 I/O/Z	 AD11
P4.4	48	Вход-выход "порт 4, 4 разряд" Вход-выход "адрес-данные, 12 разряд"	I/O/2 I/O/Z	 AD12

Окончание таблицы 2.1

1	2	3	4	5
P4.5	47	Вход-выход "порт 4, 5 разряд" Вход-выход "адрес-данные, 13 разряд"	I/O/2 I/O/Z	 AD13
P4.6	46	Вход-выход "порт 4, 6 разряд" Вход-выход "адрес-данные, 14 разряд"	I/O/2 I/O/Z	 AD14
P4.7	45	Вход-выход "порт 4, 7 разряд" Вход-выход "адрес-данные, 15 разряд"	I/O/2 I/O/Z	 AD15
HSI.0	20	Вход "быстрый ввод, канал 0"	I	
HSI.1	21	Вход "быстрый ввод, канал 1"	I	
HSI.2	22	Вход "быстрый ввод, канал 2" Выход "быстрый вывод, канал 4"	I O	 HSO.4
HSI.3	24	Вход "быстрый ввод, канал 3" Выход "быстрый вывод, канал 5"	I O	 HSO.5
HSO.0	25	Выход "быстрый вывод, канал 0"	O	
HSO.1	26	Выход "быстрый вывод, канал 1"	O	
HSO.2	31	Выход "быстрый вывод, канал 2"	O	
HSO.3	34	Выход "быстрый вывод, канал 3"	O	
EA#	79	Вход "внешний доступ"	I	
BW	71	Вход "разрядность внешней шины"	I	
READY	43	Вход "готовность"	I	
NMI	80	Вход "немаскируемое прерывание"	I	
RESET#	9	Вход-выход "сброс"	I/O/2	
INST	70	Выход "чтение команды"	O	
ALE	69	Выход "разрешение записи адреса" Выход "адрес действителен"	O O	 ADV#
WR#	40	Выход "запись" Выход "запись младшего байта"	O O	 WRL#
BHE#	41	Выход "разрешение записи старшего байта" Выход "запись старшего байта"	O O	 WRH#
RD#	68	Выход "чтение"	O	
CLKOUT	72	Выход "системный тактовый сигнал"	O	
XTAL1	74	Вывод подключения кварцевого резонатора/ вход тактового сигнала	– I	
XTAL2	73	Вывод подключения кварцевого резонатора.	–	
VPP	37	Вход "возврат из режима пониженного потребления "	I	
VCC1	8, 33, 60, 77, 78	Питание периферии микросхемы	–	
VCC2	5	Питание ядра микросхемы	–	
0V	3, 4, 6, 12, 32, 35, 36, 53, 55, 57, 64, 75, 76	Общий вывод микросхемы	–	

Примечания

1 Выводы 13, 14, 23, 54, 81, 88 – свободные, не используются.

2 В графе «Тип вывода»: I – вход, O – выход, Z – третье состояние, 2 – режим открытого стока.

2.2 Электрические параметры микросхем

Электрические параметры микросхем при приёмке и поставке и предельно-допустимые значения параметров приведены в таблицах 2.2 и 2.3 соответственно.

Номинальное значение напряжения питания микросхем 1874ВЕ76Т и 1874ВЕ06Т 5,0 В. Допустимое отклонение напряжения питания ±10 %. Амплитуда пульсаций напряжения питания не более 50 мВ. Напряжение источника опорного напряжения от 4,0 до 5,5 В. Допустимое отклонение напряжения питания от крайних значений минус 1 % для напряжения 4,0 В и плюс 1 % для напряжения 5,5 В.

Номинальное значение напряжения питания ядра микросхемы 1874ВЕ05Т должно быть 3,3 В. Номинальное значение напряжения питания периферии микросхемы должно быть 5,0 В или 3,3 В. Допустимое отклонение напряжения питания ядра от номинального должно быть не более ± 0,3 В, периферии – не более ± 0,5 В при U_{CC1} = 5,0 В или не более ± 0,3 В при U_{CC1} = 3,3 В. Амплитудное значение пульсации напряжения питания ядра должно быть не более 30 мВ, периферии – не более 50 мВ (при U_{CC1} = 5,0 В) или не более 30 мВ (при U_{CC1} = 3,3 В).

Микросхемы устойчивы к климатическим воздействиям и будут сохранять свои параметры в процессе и после воздействия на них климатических факторов, приведенных в таблице 4 ОСТ В 11 0998-99, в том числе:

- пониженной рабочей температуры среды – минус 60 ºС;
- повышенной рабочей температуры среды – 85 ºС;
- повышенной предельной температуры – 150 ºС.

Таблица 2.2 – Электрические параметры при приёмке и поставке в диапазоне рабочих температур окружающей среды от минус 60 ºС до плюс 85 ºС

Наименование параметра, единица измерения	Буквенное обозначение парметра	Норма		Режим измерения
		не менее	не более	
1	2	3	4	5
Для микросхем 1874ВЕ76Т и 1874ВЕ06Т				
1 Выходное напряжение низкого уровня по выводам, В	U_{OL}	–	0,45	I_{OL}=2,8 мА, U_{CC1} = 4,5 В
2 Выходное напряжение высокого уровня по выводам P1.0 – P1.7, 2.6, P2.7, В	U_{OH1}	U_{CC1}–0,7	–	I_{OH} = –30 мкА, U_{CC1} = 4,5 В
3 Выходное напряжение высокого уровня по выводам RD#, WR#, ALE, BHE#, INST, HSO.0 – HSO.3, PWM.0/P2.5, CLKOUT, AD0/P3.0 – AD7/P3.7, AD8/P4.0 – AD15/P4.7, TXD/P2.0, RXD/P2.1 (в режиме 0), В	U_{OH2}	U_{CC1}–0,7	–	I_{OH} = –3,2 мА, U_{CC1} = 4,5 В
4 Токи утечки низкого уровня на входе, кроме выводов P0.0 – P0.7, RESET#, мкА	I_{ILL1}	–10	–	U_{IL} = 0 В, U_{CC1} = 5,5 В

Продолжение таблицы 2.2

1	2	3	4	5
5 Токи утечки высокого уровня на входе, кроме выводов P0.0 – P0.7, NMI, мкА	I_{ILH1}	–	10	$U_{IH} = (U_{CC1} – 0,3)$ В, $U_{CC1} = 5,5$ В
6 Токи утечки низкого уровня на входе по выводам P0.0 – P0.7, мкА	I_{ILL2}	–3	–	$U_{IL} = 0$ В, $U_{CC1} = 5,5$ В
7 Токи утечки высокого уровня на входе по выводам P0.0 – P0.7, мкА	I_{ILH2}	–	3	$U_{IH} = U_{CC1} = U_{CC2} = 5,5$ В
8 Входной ток низкого уровня по выводам P1.0 – P1.7, P2.6, P2.7, мкА	I_{IL}	–70	–	$U_{IL} = 0,45$ В
9 Входной ток высокого уровня по выводам P1.0 – P1.7, P2.6, P2.7, мкА	I_{IH}	–650	–	$U_{IH} = 2,0$ В, $U_{CC1} = 5,5$ В
10 Динамический ток потребления в режиме RESET цифровой части ИМС по выводам VCC1, мА	I_{OCC1}	–	100	$U_{VPP} = U_{CC1} = U_{CC2} = 5,5$ В, $f_{CI} = 12$ МГц
11 Динамический ток потребления аналоговой части ИМС по выводу VCC2, мА	I_{OCC2}	–	10	$U_{VPP} = U_{CC1} = U_{CC2} = 5,5$ В, $f_{CI} = 12$ МГц
12 Функциональный контроль	ФК	–	–	U_{CC1}= (4,5; 5,5) В, U_{CC2} = (4,0; 5,5) В, f_{CI} = (8,0; 20) МГц
13 Время переключения сигнала на выходе CLKOUT: - время нарастания, нс - время спада, нс	 t_r t_f	–	 27 27	
Примечание – Параметры I_{ILL1}, I_{ILH1}, I_{ILL2}, I_{ILH2}, I_{IL}, I_{IH} при температуре минус 60 ºС не измеряются, а гарантируются нормой при температуре (25±10) ºС.				
Для микросхемы 1874BE05T				
14 Выходное напряжение низкого уровня по выводам, В	U_{OL}	–	0,45	$I_{OL} = 2,8$ мА, $U_{CC1} = 5,5$ В, $U_{CC2} = 3,0$ В
15 Выходное напряжение высокого уровня по выводам P1.0 – P1.7, P2.6, P2.7, В	U_{OH1}	U_{CC1}– –0,7	–	$I_{OH} = – 30$ мкА, U_{CC1} [1)] = $U_{CC2} = 3,0$ В
16 Выходное напряжение высокого уровня по выводам: RD#, WR#, ALE, BHE#, INST, HSO, PWM.0/P2.5, CLKOUT, AD0/P3.0 – AD7/P3.7, AD8/P4.0 – AD15/P4.7, TXD/P2.0, RXD/P2.1 (в режиме 0), В	U_{OH2}	U_{CC1}– –0,7	–	$I_{OH} = – 3,2$ мА, U_{CC1} [1)] = $U_{CC2} = 3,0$ В
17 Токи утечки низкого уровня на входах, кроме P0.0 – P0.7, P1.0 – P1.7, P2.6, P2.7, RESET#, мкА	I_{ILL1}	–10	–	$U_{IL} = 0$ В, $U_{CC1} = 5,5$ В
18 Токи утечки высокого уровня на входах, кроме P0.0 – P0.7, P1.0 – P1.7, P2.6, P2.7, NMI, мкА	I_{ILH1}	–	10	$U_{IH} = 5,2$ В $U_{CC1} = 5,5$ В
19 Токи утечки низкого уровня на входах P0.0 – P0.7, мкА	I_{ILL2}	–3	–	$U_{IL} = 0$ В, $U_{CC1} = 5,5$ В, $U_{CC2} = 3,6$ В
20 Токи утечки высокого уровня на входах P0.0 – P0.7, мкА	I_{ILH2}	–	3	$U_{IH} = U_{CC1} = 5,5$ В, $U_{CC2} = 3,6$ В
21 Входной ток низкого уровня по выводам P1.0 – P1.7, P2.6, P2.7, мкА	I_{IL}	– 90	–	$U_{IL} = 0,45$ В, $U_{CC1} = 5,5$ В, $U_{CC2} = 3,6$ В

Окончание таблицы 2.2

1	2	3	4	5
22 Входной ток высокого уровня по выводам P1.0 – P1.7, P2.6, P2.7, мкА	I_{IH}	–650	–	U_{IH} = 2,0 В, U_{CC1} = = 5,5 В, U_{CC2} = 3,6 В
23 Динамический ток потребления периферии ИМС по выводам VCC1 в режиме RESET, мА	I_{OCC1}	–	60	U_{CC1} = 5,5 В, U_{CC2} = 3,6 В, f_{CI} = 12 МГц
24 Динамический ток потребления ядра ИМС по выводу VCC2 в режиме RESET, мА	I_{OCC2}	–	40	U_{CC1} = 5,5 В, U_{CC2} = 3,6 В, f_{CI} = 12 МГц
25 Динамический ток потребления периферии и ядра ИМС по выводам VCC1, VCC2 в режиме RESET, мА	I_{OCC}	–	45	U_{CC1}[1)] = U_{CC2} = 3,6 В, f_{CI} = 12 МГц
26 Функциональный контроль	ФК	–	–	U_{CC1}= (3,0[1)]; 5,5) В, U_{CC2} = (3,0; 3,6) В, f_{CI} = (8,0; 20) МГц
27 Время переключения сигнала на выходе CLKOUT: - время нарастания, нс - время спада, нс	 t_r t_f	–	 24 24	U_{CC1}[1)] = U_{CC2} = 3,0 В, f_{CI} = 8 МГц

Примечание – Параметры I_{ILL1}, I_{ILH1}, I_{ILL2}, I_{ILH2}, I_{IL}, I_{IH} при температуре минус 60 ºС не измеряются, а гарантируются нормой при температуре (25±10) ºС.

[1)] При номинальном напряжении питания U_{CC1} = U_{CC2} = (3,3 ± 0,3) В выводы питания периферии и ядра запитываются от одного источника (U_{CC2}).

Таблица 2.3 – Значения предельно-допустимых электрических режимов эксплуатации в диапазоне рабочих температур

Наименование параметра режима, единица измерения	Буквенное обозначение параметра	Предельно-допустимый режим		Предельный режим	
		не менее	не более	не менее	не более
1	2	3	4	5	6
Для микросхем 1874ВЕ76Т и 1874ВЕ06Т					
1 Напряжение питания цифровой части ИМС, В	U_{CC1}	4,5	5,5	–	7,0
2 Напряжение питания аналоговой части ИМС, В	U_{CC2}	4,0	5,5	–	7,0
3 Напряжение программирования внутреннего OTPROM по выводам VPP, EA#, В (только для ИМС 1874ВЕ76Т)	U_{VPP} U_{EA}	12,25	12,75	–	13,0
4 Входное напряжение низкого уровня, В	U_{IL}	– 0,5	0,8	– 0,6	–
5 Входное напряжение высокого уровня тактового сигнала XTAL1, В	U_{IHCI}	0,7 U_{CC1}	U_{CC1}+0,5	–	U_{CC1}+0,6
6 Входное напряжение высокого уровня по выводу RESET#, В	$U_{IHRESET}$	2,2	U_{CC1}+0,5	–	U_{CC1}+0,6
7 Входное напряжение высокого уровня, В	U_{IH}	0,2U_{CC1}+1,0	U_{CC1}+0,5	–	U_{CC1}+0,6

Окончание таблицы 2.3

1		2	3	4	5	6
8 Выходной ток высокого уровня, мА	RD#, WR#, ALE, BHE#, INST, HSO, PWM.0/P2.5, CLKOUT, AD0/P3.0 – AD7/P3.7, AD8/P4.0 – AD15/P4.7, TXD/P2.0, RXD/P2.1 (в режиме 0)	I_{OH}	–3,2	–	–10	–
	P1.0 – P1.7, P2.6, P2.7		–0,03		–0,1	
9 Выходной ток низкого уровня, мА		I_{OL}	–	2,8	–	10
10 Частота следования импуль-сов тактового сигнала, МГц		f_{CI}	8	20	–	–
11 Длительность фронтов тактового сигнала, нс		t_{LH}, t_{HL}	–	5	–	–
12 Ёмкость нагрузки, пФ		C_L	–	100	–	200
Для микросхемы 1874BE05T						
13 Напряжение питания периферии ИМС, В		U_{CC1}	4,5	5,5	–	6,0
			3,0	3,6		
14 Напряжение питания ядра ИМС, В		U_{CC2}	3,0	3,6	–	4,0
15 Входное напряжение низкого уровня, В		U_{IL}	0	0,8	–0,5	–
16 Входное напряжение высокого уровня тактового сигнала XTAL1, В		U_{IHCI}	$0,7U_{CC1}$	$U_{CC1}+0,3$	–	$U_{CC1}+0,2$
17 Входное напряжение высокого уровня по выводу RESET#, В		$U_{IHRESET}$	2,2	$U_{CC1}+0,3$	–	$U_{CC1}+0,2$
18 Входное напряжение высокого уровня, В		U_{IH}	$0,2U_{CC1}+1,0$	$U_{CC1}+0,3$	–	$U_{CC1}+0,2$
19 Выход-ной ток высокого уровня, мА	RD#, WR#, ALE, BHE#, INST, HSO, PWM.0/P2.5, CLKOUT, AD0/P3.0 – AD7/P3.7, AD8/P4.0 – AD15/P4.7, TXD/P2.0, RXD/P2.1 (в режиме 0)	I_{OH}	–3,2	–	–10	–
	P1.0 – P1.7, P2.6, P2.7		–0,03		–0,1	
20 Выходной ток низкого уровня, мА		I_{OL}	–	2,8	–	10
21 Частота следования импуль-сов тактового сигнала, МГц		f_{CI}	8	20	–	–
22 Длительность фронтов тактового сигнала, нс		t_{LH}, t_{HL}	–	5	–	–
23 Ёмкость нагрузки, пФ		C_L	–	100	–	200

3 Архитектура изделий

16-разрядные микроконтроллеры 1874BE76T, 1874BE06T предназначены для выполнения вычислительных инструкций с повышенным быстродействием и высокоскоростных инструкций ввода-вывода. Они имеют общие архитектуру и набор инструкций с другими микросхемами серии 1874.

Микроконтроллеры 1874BE76T, 1874BE06T разработаны для использования в быстродействующих системах управления в режиме реального времени.

В настоящем разделе приводится краткое описание устройства и принципов его работы. Полное описание, включая описание системы команд, встроенных функциональных блоков и особенностей программирования приведено в последующих разделах.

3.1 Особенности архитектуры

Структурная схема микросхем 1874BE76T, 1874BE06T приведена на рисунке 2.1. Структурная схема микросхем 1874BE05T приведена на рисунке 2.2.

Микроконтроллеры 1874BE76T и 1874BE06T являются полными функциональными аналогами изделий TN87C196KC-20 и TN80C196KC-20 соответственно, реализующими базовую архитектуру MCS-196 фирмы Intel.

Это 16-разрядные быстродействующие ИС высокой степени интеграции, ориентированные на решение задач управления процессами в реальном масштабе времени. В результате применения ожидается увеличение функциональности, надежности и производительности встроенных систем управления (не менее чем в 1,6 раза при прямой замене серийно выпускаемых в настоящее время микроконтроллеров 1874BE36) за счёт двукратного увеличения объёмов встроенной памяти программ и регистрового ОЗУ, наличия многофункционального таймера, расширенной системы прерываний.

По общепринятому среди разработчиков аппаратуры мнению однокристальные микроконтроллеры семейства MCS-196 фирмы Intel являются своего рода индустриальным стандартом для 16-разрядных встроенных систем управления, аналогично тому, как популярные микроконтроллеры MCS-51 этой же фирмы в свою очередь давно уже стали базовыми микросхемами для 8-разрядных систем. Это объясняется с одной стороны сочетанием высоких технических показателей и экономической эффективности указанных семейств, а с другой – совершенством архитектуры, обеспечивающей эффективность и, частично, преемственность программных наработок при проектировании систем на их основе.

Архитектура микроконтроллеров идеально ориентирована для создания модификаций систем для реализации функций управления и вычисления в реальном режиме времени под конкретные приложения. Этому способствует высокая производительность микроконтроллеров, развитая система встроенных функциональных блоков, внутренняя программная память объемом 16К (для микросхем 1874BE76T) и мощная система команд. Микросхемы могут служить элементной базой для систем управления различной аппаратурой.

Изделия представляют собой микроконтроллеры – вычислительно-управляющие устройства, предназначенные для выполнения функций контроля и управления систем и периферийным оборудованием.

Ориентация в сторону управления накладывает отпечаток на особенность архитектуры микроконтроллеров.

Ключевые особенности микроконтроллеров:

- Функционирование на частотах до 20 МГц.
- Быстродействующая архитектура типа "регистр-регистр".
- Регистровое ОЗУ емкостью 488 байт.
- Внутренняя однократно программируемая память команд емкостью 16К байт (только для 1874BE76T).
- Динамически конфигурируемая разрядность шины данных - 8 или 16 бит.
- Протокол захваты шины HOLD/HLDA.
- Устройство высокоскоростного ввода-вывода.
- Два 16-разрядных таймера/счетчика с предделителями и режимами квадратурного счета.
- 16-разрядный сторожевой таймер.
- 8 каналов аналого-цифрового преобразователя с разрешением в 8 или 10 бит.
- Пять 8-разрядных портов ввода-вывода.
- Режимы холостого хода (IDLE) и пониженного энергопотребления (POWERDOWN).
- Сервер периферийных транзакций (PTS).

Микроконтроллеры содержат полный набор команд, включающий операции с битами, байтами, словами, двойными словами (беззнаковые 32 бит), длинные операции (32 бит со знаком), работу с флагами, а также переходы и вызовы подпрограмм. Все стандартные логические и арифметические команды работают как с байтами, так и со словами. Команды перехода по установке бита (Jump Bit Set) и очистке бита (Jump Bit Clear) могут работать с каким-либо регистром SFR или с другими байтами регистрового файла. Эти быстрые битовые операции позволяют ускорить функции ввода-вывода.

Операции над байтами и словами составляют основу системы команд. Ассемблер ASM-96 использует суффикс "B" в мнемонике для операций над байтами, иначе мнемоника относится к операциям над словами.

Длинные и двухсловные операции включают операции сдвигов, нормализацию, умножения и деления. В операции “Деление” делится 32-битное число на 16-битное число, что порождает 16-битное частное и 16-битный остаток. В операции “Умножение” умножается 16-битное на 16-битное число с образованием 32-битного результата. Обе операции могут выполняться с числами со знаком или без знака. Команды нормализации и соответствующий флаг обеспечивают аппаратную поддержку пакета программ для выполнения операций над числами с плавающей запятой (FPAL-96).

3.2 Краткое описание функционирования микросхем комплекта

3.2.1 Процессорное ядро

Микроконтроллеры построены по Фон Неймановской архитектуре. Они имеют внешнюю системную магистраль для обмена данными с внешней памятью и дополнительными периферийными устройствами. Система команд микроконтроллеров поддерживает широкий набор методов адресации, в т. ч. битовую адресацию.

Архитектура микроконтроллеров, называемая архитектурой типа "регистр-регистр", принципиально отличается от архитектуры микроконтроллеров других серий. Такая архитектура обеспечивает достижение более высокой производительности и упрощает работу с периферией. Этим, в первую очередь, объясняются возможности эффективного решения на базе микроконтроллеров достаточно сложных задач управления в реальном времени.

Главным отличительным признаком архитектуры микроконтроллеров комплекта является ядро, базирующееся на регистровом файле. Большое число универсальных легко доступных регистров исключает "узкие места", свойственные архитектуре, использующей специальные регистры-аккумуляторы, и обеспечивает быстрое переключение контекста. Все устройства микроконтроллера реализуют операции с байтами, словами, несколько операций с 32-битовыми операндами, а также команды перехода по битам.

Основные компоненты ядра, использованного в микроконтроллерах – регистровый файл и регистровое арифметико-логическое устройство (РАЛУ). Регистровый файл – это адресуемое пространство регистров процессора. Ячейки от 00h до 17h – это управляющие регистры ввода-вывода или регистры специальных функций (Special Function Registers – SFR). Ячейки 18h и 19h содержат указатель стека, они могут использоваться как обычная RAM-память, когда не выполняются стековые операции. Остальные байты регистрового файла служат как обычная RAM-память, доступная как байт, слово или как двойное слово.

ЦПУ выполняет вычисления в РАЛУ. РАЛУ, изображенное на рисунке 3.2, содержит 17-битное арифметическое устройство (АЛУ), слово состояния процессора (PSW), основной счетчик команд (PC), регистр инструкций, процессор микрокоманд, регистр констант (Константы), трёхбитный регистр выбора, счётчик циклов и три вспомогательных регистра (верхнего слова, нижнего слова и регистр 2-го операнда). Все регистры, кроме трёхбитного регистра выбора, либо 16-ти, либо 17-ти битные (16 бит плюс 1 бит знаковое расширение). Некоторые из этих регистров могут уменьшать

нагрузку на АЛУ при выполнении простых операций. Слова вводятся в АЛУ через входы A и B.

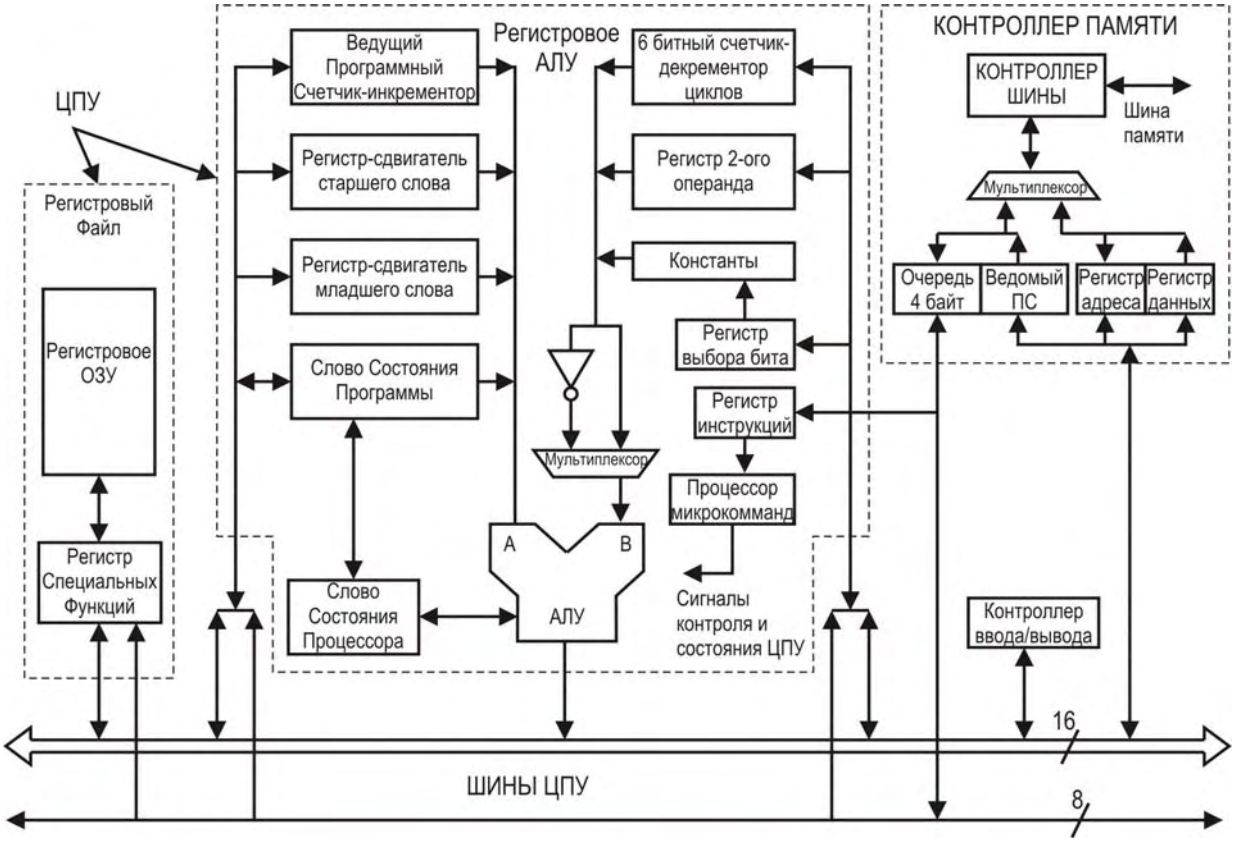


Рисунок 3.2 – Структурная схема ядра микроконтроллеров комплекта

Скорость работы АЛУ при вычислениях увеличивается, если использовать константы, хранимые в регистрах констант (например, константа 0, 1 или 2), так как они постоянно доступны во время операций инвертирования, декрементирования, инкрементирования байтов или слов.

PSW содержит бит (PSW.1), который производит общее разрешение обслуживания всех маскируемых прерываний, бит (PSW.2), разрешающий или запрещающий работу сервера периферийного обмена и шесть битовых флагов, отражающие состояние программы пользователя.

PC содержит адрес следующей команды и автоматически инкрементируется для загрузки следующего адреса из очереди. Если переход, прерывание, вызов подпрограммы или возврат из подпрограммы изменяет адресную последовательность, то АЛУ загружает соответствующий адрес в PC.

Верхний и нижний 16-битные (словные) регистры используются совместно при выполнении 32-х битных команд, а также как вспомогательные регистры во многих командах. Так как они имеют свою собственную сдвиговую логику, РАЛУ использует их в операциях, которые требуют логических сдвигов (таких как нормализация, умножение и деление). Нижний словный регистр используется, когда требуется сдвигать двойные слова, а верхний регистр используется для любых сдвигов. Число сдвигов считается 6-ти битным счётчиком циклов.

Регистр второго операнда хранит второй операнд, когда MCE выполняет двухоперандную команду. Сюда относятся множитель в команде умножения и делитель в команде деления. Во время вычитания выход этого регистра преобразуется в дополнительный код перед подачей его на вход АЛУ.

3.2.2 Способы адресации

Система команд микроконтроллеров содержит следующие типы адресации: прямая регистровая (register direct), косвенная (indirect), косвенная с автоувеличением (indirect with autoincrement) непосредственная (immediate), короткая индексная (short-indexed) и длинная индексная (long-indexed). Эти типы адресации увеличивают гибкость и скорость выполнения команд устройствами микроконтроллера. Каждая команда использует по крайней мере один из способов адресации.

Прямая регистровая адресация и непосредственная адресация выполняются наиболее быстро. Прямая регистровая адресация обеспечивает доступ к файлу регистров и SFR. Непосредственная адресация использует информацию, следующую за кодом команды, как операнд.

Оба режима косвенной адресации используют значение слова в регистре как адрес операнда. Косвенная адресация с автоинкрементом увеличивает адресное слово на единицу после операции с байтом и на два - после операции со словом. Этот способ адресации обеспечивает легкий доступ к справочным таблицам.

Длинная индексная адресация обеспечивает прямой доступ к любой ячейке 64К адресного пространства. Этот способ формирует адрес операнда добавлением 16-битного значения к регистровому слову. Индексирование с нулевым регистром позволяет иметь прямую адресацию к любой ячейке. Короткая индексная адресация формирует адрес операнда добавлением 8-битного значения к регистровому слову.

Кроме того, имеются расширенные версии команд прерываемой и непрерываемой передачи блока. Множество способов адресации делает легким программирование на языке ассемблера и обеспечивает отличную взаимосвязь с языками высокого уровня. Команды ассемблера состоят из мнемоники, за которой следуют адрес или данные.

3.2.3 Прерывания

Основной функцией микроконтроллера является обеспечение управления устройствами в реальном времени. Схема контроллера прерываний позволяет событиям реального времени управлять выполнением программы. Когда событие вызывает прерывание, ядро обслуживает это прерывание перед выполнением следующей команды. Встроенная периферия, внешний сигнал или команда могут выставить запрос на прерывание. В простейшем случае микроконтроллер получает запрос прерывания, осуществляет обслуживание и возвращается к прерванной программе.

В разработанных микроконтроллерах 28 источников прерывания и 16 векторов прерывания, а также 2 дополнительных вектора прерывания Software Trap (Пошаговое выполнение программ) и Unimplemented Opcodes (Неопознанные коды операций), используемых в системах отладки или платах-прототипах. Когда контроллер прерывания определит одно из шестнадцати прерываний, он устанавливает соответствующий бит в один из двух регистров ожидания прерываний. Отдельные прерывания разрешаются либо запрещаются установкой или очисткой бит в регистре маски прерываний. Когда контроллер прерывания производит обработку прерывания, он делает вызов программы обслуживания прерываний (ISR). Соответствующий вектор прерывания содержит адрес ISR. Затем контроллер прерывания очищает соответствующий бит ожидания.

Имеются две возможности обслуживания прерываний: программное обслуживание прерываний через контроллер прерываний и микропрограммное обслуживание прерываний через PTS. Можно выбрать вариант обслуживания прерываний для каждого из маскируемых прерываний. Немаскируемые прерывания (NMI, программное прерывание, прерывание по неподдерживаемому коду) всегда обрабатываются подпрограммами обработки прерываний.

3.2.4 Периферийные устройства

3.2.4.1 Стандартные порты ввода-вывода

Микроконтроллеры имеют пять 8-разрядных портов ввода-вывода. Порт 0 – входной, он же порт аналоговых входов для АЦП (в микросхемах 1874ВЕ76Т и 1874ВЕ06Т). Порт 1 – квазидвунаправленный, он разделяет выводы с двумя выходами модуля ШИМ. Порт 2 содержит три типа линий – квазидвунаправленные, входные и выходные. Другие функции МК используют входные и выходные линии совместно с портом 2. Порты 3 и 4 являются двунаправленными портами с открытым стоком, они могут использоваться как интерфейс шины адресов/данных.

3.2.4.2 Таймеры-счетчики

Микроконтроллер имеет два независимых 16-разрядных многофункциональных таймера-счетчика. В одном из таймеров используется внутренняя синхронизация (режим работы – таймер реального времени), в другом – внешняя (счетчик внешних событий). Содержимое таймеров может быть в любое время считано или программно модифицировано, а также сброшено программно или внешним сигналом. Таймер-счетчик внешних событий подсчитывает положительные или отрицательные фронты входных сигналов и может работать как в режиме прямого счета (инкремента), так и обратного (декремента).

Источником счетных импульсов может быть либо системный синхросигнал с фиксированной частотой (возможно предварительно деленной в заданное число раз), либо один из входов микроконтроллера. В первом случае устройство выполняет функции таймера, так как фактически считает интервалы времени постоянной длительности.

Во втором случае устройство выполняет функции счетчика событий (отрицательных или положительных перепадов) на входе микросхемы. Счет может производиться либо в одном, либо в обоих направлениях. В последнем случае направление счета определяется уровнем сигнала на соответствующем входе микросхемы. При переходе содержимого счетчика из наибольшего состояния в наименьшее и наоборот могут генерироваться соответствующие внутренние запросы на прерывания. Конкретные режимы работы и структура таймеров устанавливаются программно.

Внутренняя синхронизация увеличивает значение таймера 1 и таймера 2 каждый восьмой такт процессора, равный трем периодам тактовой частоты. Таймер 2 может иметь внешнюю синхронизацию. При внешней синхронизации значение таймера 2 увеличивается при каждом положительном или отрицательном перепаде. Любой внешний или внутренний источник может сбросить таймер 2. Таймер 1 и таймер 2 могут сформировать прерывание при переходе границы 0FFFFh/0000h. Микроконтроллер также позволяет использовать таймеры для задания скорости передачи информации последовательным портом и для управления работой сторожевого таймера (Watchdog timer). Это внутренний таймер для сброса системы в случае, если программное обеспечение не может нормально функционировать. Два 16-битных таймера могут работать от внутреннего генератора или внешнего источника. Внешний режим "quadrature clocking" пригоден для наблюдения за скоростью и направлением при позиционном кодировании.

3.2.4.3 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 линию RESET в состояние низкого уровня не менее, чем на 2 машинных цикла, сбрасывая микроконтроллер и другие устройства, подключенные к его выводу RESET.

3.2.4.4 Устройства высокоскоростного ввода (HSI)

HSI может записывать время внешних событий с разрешающей способностью в 9 тактов. HSI может следить за четырьмя независимыми линиями HSI и определять значение таймера 1, когда произойдет событие. Четыре типа событий могут инициализировать захват: нарастающий фронт, спад сигнала, нарастание и спад или каждый восьмой нарастающий фронт. HSI может хранить 8 элементов (значений таймера 1), 7 в 7-уровневой очереди FIFO и 1 в регистре хранения HSI. Чтение регистра хранения HSI не перегружает значения, размещенные в FIFO ранее. Устройства HSI могут сформировать прерывание при загрузке регистра хранения HSI, или когда загружаемое значение является четвертым или шестым элементом FIFO.

3.2.4.5 Высокоскоростное устройство вывода (HSO)

HSO может инициализировать события в определенные моменты, базирующиеся на значениях таймера 1 или таймера 2. Эти программируемые события заключаются в запуске A/D-преобразования, сбросе таймера 2, формировании до четырех программных временных задержек и установке или очистке одной или нескольких линий из шести выходных линий HSO. HSO хранит ожидаемые события и соответствующее время в блоке памяти Content Addressable Memory (CAM). Этот блок хранит до восьми команд. Каждая команда определяет время действия, "природу" действия, имеет ли место прерывание, идет ссылка на таймер 1 или таймер 2. Каждый восьмой такт HSO сравнивает значение ячеек CAM на совпадение с текущим временем. HSO инициализирует определенные события при совпадении времени. Команда стирается из CAM после выполнения или остается в CAM как сохранённая точка входа CAM, тогда команда будет постоянно выполняться при совпадении значения времени для этой точки со значением текущего времени в соответствующем таймере. Сохранённая точка входа полезна в применениях, требующих периодических или повторяющихся событий, таких как повторяющаяся ШИМ последовательность.

3.2.4.6 Последовательный порт

Имеет один синхронный режим (Mode0) и три асинхронных (1, 2 и 3). Асинхронные режимы полностью дуплексные, т. е. они могут передавать и принимать данные одновременно. Приемник на МК буферизован так, что прием второго байта может начаться до считывания первого. Передатчик также дважды буферизован. Наиболее общее использование синхронного Режима 0 – расширение возможностей устройств ввода-вывода микроконтроллеров, использующих регистры сдвигов. Режим 1 – стандартный асинхронный режим, используемый для нормальных последовательных коммуникаций. Структура данных для режима 1 состоит из 10 бит: стартовый, 8 бит данных (LSB первый) и стоповый бит. Если передача бита четности разрешена (PEN=1), бит дополнения до четности посылается вместо восьмого бита данных. Режимы 2 и 3 - девятибитные режимы общего пользования для межпроцессорных коммуникаций. Структура данных, используемых в этих режимах, состоит из 11 бит: стартовый, 9 бит данных (LSB первый) и стоповый бит. Устройства, работающие в режиме 2, будут вырабатывать прерывание при приеме только тогда, когда девятый информационный бит будет установлен. Устройства, работающие в режиме 3, всегда будут вырабатывать прерывание при приеме. Режим 3 позволяет передавать 8 информационных бит плюс бит дополнения до четности.

3.2.4.7 Блок ШИМ (PWM)

Встроенный ШИМ предназначен для генерации широтно-модулированного сигнала на выходе микросхемы без участия процессора и имеет три выхода PWM.

Скважность импульса на выходе PWM является переменной величиной, импульсы повторяются каждые 256 или 512 тактов. PWM может использоваться в различных применениях. Различные типы моторов требуют использование PWM – формирователя импульсов для наиболее эффективной работы. Кроме того, фильтрация

этой ШИМ последовательности будет создавать постоянный уровень, который изменяется с изменением скважности.

3.2.4.8 Аналого-цифровой преобразователь (АЦП)

Содержится только в МК 1874BE76T и 1874BE06T. Конвертирует аналоговый вход в цифровой эквивалент. Разрешающая способность 8 или 10 бит с программируемыми временами захвата и преобразования. Основные компоненты A/D-преобразователя: захватывающий и хранящий конденсатор, 8-канальный мультиплексор и 8- или 10-битный аналого-цифровой преобразователь последовательного приближения. Преобразователь может начать преобразование немедленно, или HSO может инициализировать преобразование в запрограммированное время. При завершении каждого преобразования АЦП вырабатывает прерывание. МК 1874BE76T и 1874BE06T имеют отдельные выводы питания ∩VCC2 и ∩0V, что исключает влияние помех по линиям питания на аналого-цифровое преобразование.

3.2.4.9 Сервер периферийных транзакций (PTS)

Этот функциональный блок предназначен для аппаратной обработки прерываний. Он содержит набор встроенных алгоритмов, исходные данные для которых должны быть размещены программой пользователя во встроенном ОЗУ кристалла. Алгоритмы PTS охватывают, в основном, пересылки данных. Прерывания, обслуживаемые PTS, отрабатываются быстрее, чем те, которые обслуживаются обычным способом. Однако программировать PTS непросто, а отлаживать еще сложнее.

Этот блок предназначен для обеспечения некоторой последовательности событий в заданное время без участия процессора. Заданные события выполняются на микропрограммном уровне. Такими блоками событий могут быть:

- передача блока информации из одного места памяти (или устройства ВВ) в другое;
- последовательный опрос нескольких каналов АЦП;
- передача информации по последовательному каналу связи.

Использование такого механизма обработки событий позволяет снизить загрузку процессора и распараллелить процесс обработки информации.

Сервер периферийных транзакций (PTS) поддерживает обработку запросов на прерывание на микропрограммном уровне, не требуя вмешательства процессора. Специальный режим работы PTS обеспечивает поддержку функций последовательного ввода-вывода (SIO).

3.2.4.10 Энергосберегающие режимы работы

Реализованы режимы работы с пониженным энергопотреблением:

- режим IDLE, когда работают только встроенные функциональные устройства, а микроконтроллер находится в режиме ожидания разрешенного прерывания от внешнего или внутреннего устройства;
- в режиме POWERDOWN вся внутренняя синхронизация замораживается в состоянии логического нуля, и генератор отключается. Внутреннее ОЗУ и большинство периферийной памяти сохраняет своё значение, если сохраняется напряжение питания. В этом случае ток потребления – всего несколько мкА.

3.2.5 Синхронизация

На схему синхроимпульсов (рисунок 3.3) поступает входной тактовый сигнал по выводу XTAL1, задаваемый внешним кварцевым резонатором и делится по частоте на два. Генератор синхроимпульсов принимает входную частоту от схемы "делитель на два" и формирует два не перекрывающихся внутренних сигнала PH1 и PH2.

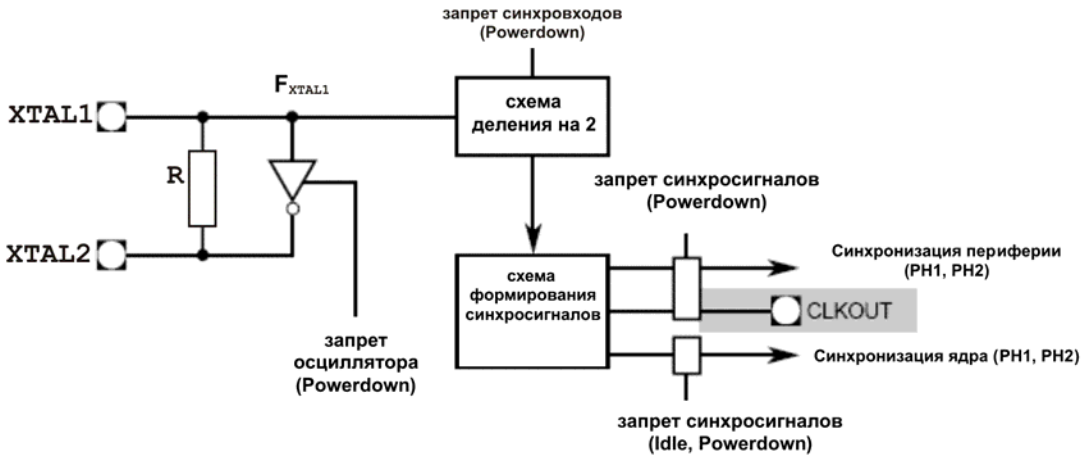


Рисунок 3.3 – Схема формирователя синхроимпульсов

Передние фронты PH1 и PH2 формируют внутренний CLKOUT сигнал. Схема синхроимпульсов отдельно формирует внутренние сигналы синхроимпульсов для центрального процессора и периферийных устройств, чтобы обеспечить гибкость в управлении мощностью потребления от источника питания. В схеме формирователя синхроимпульсов ИС 1874BE05T резистор R отсутствует.

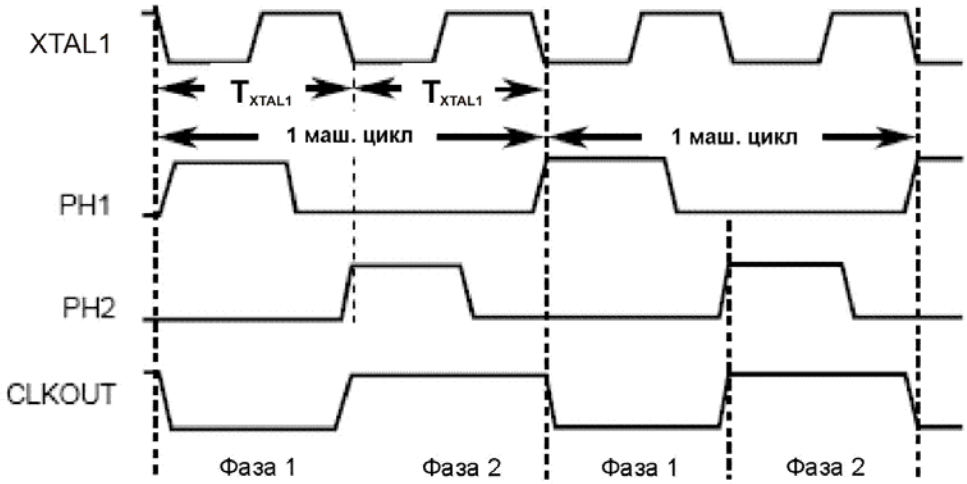


Рисунок 3.4 – Внутренние фазы синхроимпульсов

Комбинация периодов фазы 1 и фазы 2 внутреннего CLKOUT сигнала определяет основную единицу времени микроконтроллера, называемую «машинным циклом».

Следующие формулы позволяют вычислять частоты PH1 и PH2, МГц, – формула (1), длительность машинного цикла, мкс, – формула (2), период синхроимпульсов T_{XTAL1} – (3).

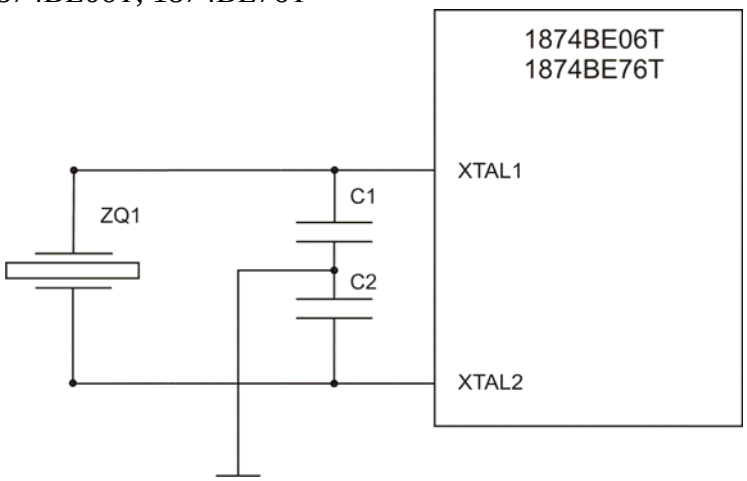
$$PH1 = F_{XTAL1}/2 = PH2, \quad (1)$$

$$\text{длительность машинного цикла} = 2/F_{XTAL1}, \quad (2)$$

$$T_{XTAL1} = 1/F_{XTAL1} \quad (3)$$

Поскольку микроконтроллер может работать на многих частотах, эти вычисления времени позволяют определять время выполнения инструкции в машинных циклах.

Тактовый сигнал микроконтроллера можно получить, подключив к выводам XTAL1 и XTAL2 кварцевый резонатор или подав на вывод XTAL1 тактовый сигнал от внешнего источника. На рисунке 3.5 приведена схема подключения кварцевого резонатора для ИС 1874BE06T, 1874BE76T



Конденсаторы C1=C2=20 пФ.

Рисунок 3.5 – Схема подключения кварцевого резонатора для ИС 1874BE06T, 1874BE76T

Для ИС 1874BE05T запуск внутреннего генератора от кварцевого резонатора является более сложной задачей. Рекомендованная схема подключения приведена на рисунке 3.6.

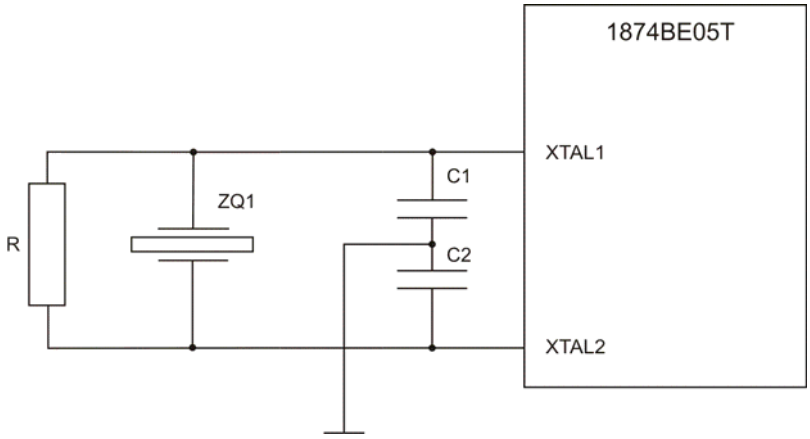
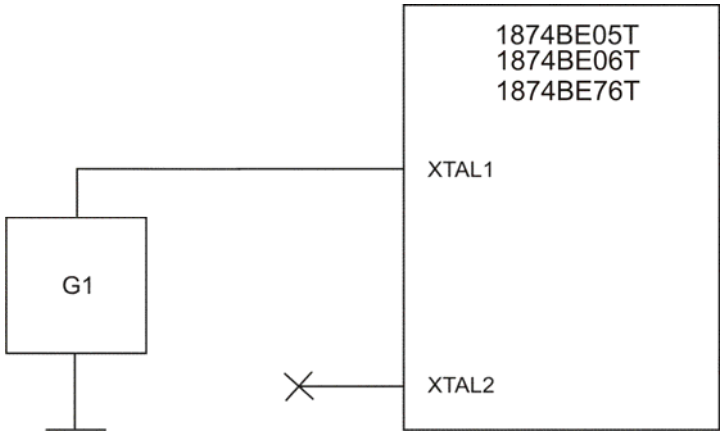


Рисунок 3.6 – Схема подключения кварцевого резонатора для ИС 1874BE05T

Номинал резистора R должен быть от 50 кОм до 1 МОм, номиналы конденсаторов C1, C2 должны быть равны и не превышать 20 пФ. Применять кварцевые резонаторы с высокой добротностью.

Для обеспечения стабильного функционирования микроконтроллера 1874BE05T рекомендуется использовать внешний кварцевый генератор, тактовый сигнал подавать как показано на рисунке 3.7.



G1 – внешний кварцевый генератор

Рисунок 3.6 – Схема подключения кварцевого генератора для ИС 1874BE05T, 1874BE06T, 1874BE76T

Примечание – Вывод XTAL2 при этом остается неподключенным к любому потенциалу.

4 Типы данных и адресация

Данный раздел определяет типы операндов и способы адресации, обеспечиваемые архитектурой микроконтроллеров 1874BE76T, 1874BE06T и 1874BE05T.

4.1 Типы операндов

Система команд поддерживает множество типов данных, которые обычно используются в управляющих алгоритмах.

Типы переменных операндов указаны заглавными буквами во избежание путаницы. Например, BYTE – 8-битная беззнаковая переменная в инструкции, в то время как byte – любая 8-битная единица данных (со знаком или без знака).

Таблица 4.1 даёт общий обзор этих типов операндов. Далее в этом разделе детально рассмотрен каждый из них.

Таблица 4.1 – Определения типа операнда

Тип операнда	Число бит	Знак	Возможные значения	Ограничения адресации
BIT	1	нет	TRUE (1) или FALSE (0)	как компоненты байтов
BYTE	8	нет	от 0 до (2^8–1) (от 0 до 255)	нет
SHORT INTEGER	8	да	от (–$2^{7)}$ до (+2^7–1) от (–128) до (+127)	нет
WORD	16	нет	от 0 до (2^{16}–1) (от 0 до 65 535)	адрес четного байта
INTEGER	16	да	от (–2^{15}) до (+2^{15}–1) (от –32 768 до +32 767)	адрес четного байта
DOUBLE WORD [1)]	32	нет	от 0 до (2^{32}–1) (от 0 до 4 294 967 295)	адрес в младшем регистровом файле, кратный четырем [2)]
LONG INTEGER [1)]	32	да	от (–2^{31}) до (+2^{31}–1) (от –2 147 483 648 до +2 147 483 647)	адрес в младшем регистровом файле, кратный четырем [2)]

[1)] 32-битные переменные поддерживаются только как операнды в инструкциях сдвига, как делимое при инструкции деления 32 на 16 и как результат действия умножения 16 на 16.

[2)] Для совместимости с программными средствами сторонних производителей необходимо придерживаться правил, принятых в программировании на «Си» для адресации 32-битных операндов.

Таблица 4.2 – Эквивалентные типы операндов для ассемблера и языка «Си»

Тип операнда	Эквивалент ассемблера	Эквивалент на языке «Си»
BYTE	BYTE	unsigned char
SHORT INTEGER	BYTE	char
WORD	WORD	unsigned int
INTEGER	WORD	int
DOUBLE WORD	LONG	unsigned long
LONG INTEGER	LONG	long

Операнды BIT

BIT – одноразрядная переменная, которая может иметь булевы значения "true" и "false". Архитектура требует, чтобы к BIT обращались как к компонентам BYTE или WORD, так как прямая адресация BIT не поддерживается.

Операнды BYTE

BYTE – 8-битная переменная без знака, которая может иметь значения от 0 до 255 (28–1). Арифметические и сравнение операторы могут быть применены к операндам BYTE, но результат должен интерпретироваться по модулю 256 арифметически. Логические действия с BYTE осуществляются побитно. Биты в пределах BYTE помечены от 0 до 7, бит 0 – младший бит. Нет никаких ограничений выравнивания для BYTE, так что они могут быть помещены по любому адресу памяти.

Операнды SHORT INTEGER

SHORT INTEGER – 8-битная переменная со знаком, которая может принимать значения от минус 128 (–27) до плюс 127 (27–1). Арифметические действия, которые производят результаты вне диапазона SHORT INTEGER, устанавливают флаги переполнения в слове состояния процессора (PSW). Числовой результат тот же самый, что и результат эквивалентного действия с переменной BYTE. Нет никаких ограничений выравнивания для SHORT INTEGER, так что они могут быть помещены по любому адресу памяти.

Операнды WORD

WORD – 16-битная беззнаковая переменная, которая может иметь значения от 0 до 6 535 (2^{16}–1). Арифметические и сравнение операторы могут быть применены к операндам WORD, но результат должен интерпретироваться по модулю 65536 арифметически. Логические действия с WORD осуществляются побитно. Биты в пределах WORD помечены от 0 до 15, бит 0 – младшего значения бит.

WORD должны быть выровнены по границе четного байта адреса пространства. Младший байт WORD находится по четному адресу байта, а старший байт находится в следующем (нечетном) адресе. Адрес WORD – адрес младшего байта. Действия WORD по нечетным адресам не гарантируются.

Операнды INTEGER

INTEGER – 16-битная переменная со знаком, которая может принимать значения от минус 32 768 ($–2^{15}$) до плюс 32 767 ($+2^{15}$–1). Арифметические действия, которые производят результаты вне диапазона INTEGER, устанавливают флаги переполнения в слове состояния процессора (PSW). Числовой результат тот же самый, как результат эквивалентного действия на переменных WORD.

INTEGER должен быть выровнен по границе четного байта в адресном пространстве. Младший байт INTEGER находится по четному адресу байта, а старший байт находится в следующем выше нечетном адресе. Адрес INTEGER – адрес его младшего байта. Действия INTEGER по нечетным адресам не гарантируются.

Операнды DOUBLE WORD

DOUBLE WORD – 32-битная переменная без знака, которая может принимать значения от 0 до 4 294 967 295 (2^{32}–1). Архитектура непосредственно поддерживает операнды DOUBLE WORD только как операнды в командах сдвига, делимого при делении 32/16 и произведения при умножении 16×16. Для этих действий переменная DOUBLE WORD должна находиться в младшем файле регистров и должна быть выровнена по адресу, который является кратным четырем. Адрес DOUBLE WORD – это адрес младшего байта (четный адрес байта). Младшее слово DOUBLE WORD находится всегда в младшем адресе, даже когда данные находятся в стеке. Это означает, что старшее слово должно быть выдвинуто в стек первым.

Действия DOUBLE WORD, которые непосредственно не поддерживаются, могут быть легко осуществлены с двумя операндами WORD. Например, следующие последовательности инструкций с 16 битами исполняют 32-битное сложение и 32-битное вычитание соответственно.

```
ADD     REG1, REG3; ( 2-операндное сложение)
```

```
ADDC    REG2, REG4
SUB     REG1, REG3; ( 2-операндное вычитание)
SUBC    REG2, REG4
```

Операнды LONG INTEGER

LONG INTEGER – 32-битная переменная со знаком, которая может принимать значения от минус 2 147 483 648 (-2^{31}) до плюс 2 147 483 647 ($+2^{31}-1$). Архитектура непосредственно поддерживает операнды LONG INTEGER только как операнды в командах сдвига, как делимое при делении 32/16 и как произведение при умножении 16×16. Для этих действий переменная LONG INTEGER должна быть в младшем файле регистров и должна быть выровнена по адресу, который является кратным четырем. Адрес LONG INTEGER – это его младший байт (адрес четного байта).

Инструкции с LONG INTEGER, которые непосредственно не поддерживаются, могут быть легко осуществлены с двумя операндами INTEGER. См. пример в “Операнды DOUBLE WORD”.

4.2 Режимы адресации

Косвенная адресация осуществляет доступ к операнду, адрес которого размещен в переменной типа WORD Регистрового Файла. Вычисляемый адрес должен соответствовать правилам выравнивания. Следует заметить, что косвенная адресация позволяет обращаться к операнду в любом месте адресного пространства MCS-96, включая Регистровый Файл. 8-и битное поле внутри команды выбирает регистр, который содержит косвенный адрес. Команда может содержать только одну косвенную ссылку, обращение к добавочным операндам осуществляется с помощью прямой регистровой адресации.

Пример косвенной адресации:

```
LD       AX, [AX]    ; AX ← MEM_WORD(AX)
ADDB AL, BL, [CX]; AL ← BL + MEM_BYTE(CX) (ADDB_3op)
POP [AX]             ; MEM_WORD(AX) ← MEM_WORD(SP)
                     ; SP ← SP + 2
```

Рабочие регистры:
AX, CX -16-и битные регистры.
AL ,SL - младшие байты регистров AX, BX.

4.2.1 Косвенная адресация с автоинкрементом

Косвенная адресация с автоинкрементом - это то же самое, что и косвенная адресация, но в дополнение переменная типа WORD, содержащая косвенный адрес, инкрементируется после использования для адресации операнда. Младший значащий бит регистра типа WORD поразному интерпретируется в зависимости от наличия или отсутствия автоинкремента в косвенной адресации. Если команда работает с переменными типа BYTE или SHORT-INTEGER, адрес инкрементируется на 1, а если команда работает с переменными типа WORD или INTEGER, то адрес инкрементируется на 2.

Пример косвенной адресации с автоинкрементом:

```
LD              AX, [BX]+   ; AX ← MEM_WORD(BX)
                            ; BX ← BX + 2
ADDB AL, BL, [CX]+          ; AL ← BL + MEM_BYTE(CX)
                            ; CX ← CX + 1 (ADDB_3op)
PUSH [AX]+                  ; SP ← SP – 2
                            ; MEM_WORD(SP) ← MEM_WORD(AX)
                            ; AX ← AX + 2
```

Рабочие регистры:
AX, BX, CX – 16-битные регистры.
AL, BL - младшие байты регистров AX, BX.

4.2.2 Непосредственная адресация

Непосредственная адресация позволяет брать операнд непосредственно из поля команды. Для операций с переменными типа BYTE или SHORT-INTEGER это будет 8-и разрядное поле. Для операций с переменными типа WORD или INTEGER - 16-и разрядное поле. Команда может содержать только один непосредственный операнд; для других операндов необходимо использовать Прямую Регистровую адресацию.

Пример Непосредственной адресации:

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

Рабочие регистры:
AX - 16-и битный регистр.
AL - младший байт регистра AX.
AH - старший байт регистра AX.

4.2.3 Короткая Индексная адресация

При Короткой Индексной адресации адрес одного из операндов вычисляется с помощью двух 8-и битных полей. Одно 8-и битное поле выбирает переменную типа WORD в Регистровом Файле, которая содержит адрес. Второе 8-и битное поле в команде имеет знаковое расширение и суммируется с переменной типа WORD для формирования исполнительного адреса операнда. Исполнительный адрес может быть на 128 байт впереди от текущего адреса переменной типа WORD и на 127 байт после него. Команда может содержать только один такой операнд, другие операнды задаются с помощью Прямой Регистровой адресации.

Пример Короткой Индексной адресации:

```
LD   AX, 12[BX]         ; AX ← MEM_WORD(BX+12)
MULB AX, BL, 3[CX]      ; AX ← BL * MEM_BYTE(CX+3)
; (MULB_3op)
```

Рабочие регистры:
AX, BX, CX - 16-и битные регистры.
BL - младший байт регистра BX.

4.2.4 Длинная Индексная адресация

Длинная Индексная адресация похожа на Короткую Индексную адресацию за исключением того, что 16-и битное поле берётся из команды и добавляется к переменной типа WORD для формирования адреса операнда. Нет необходимости в знаковом расширении. Такая адресация может применяться только для одного операнда в команде, для других операндов необходимо использовать Прямую Регистровую адресацию.

Пример Длинной Индексной адресации:

```
AND AX, BX, TABLE[CX] ;     AX ← BX AND MEM_WORD(TABLE+CX)
;                           (AND_3op)
ST AX, TABLE[BX]      ; MEM_WORD(TABLE_BX) ← AX
ADDB AL, BL, LOOKUP[CX]   ; AL ← BL + MEM_BYTE(LOOKUP+CX)
;                           (ADDB_3op)
```

Рабочие регистры:
AX, BX, CX - 16-и битные регистры.
AL, BL - младшие байты регистров AX, BX.

4.2.5 Адресация с использованием Нулевого Регистра

Два первых байта в регистровом файле образуют Нулевой Регистр (ZERO_REG). Эти байты всегда равны нулю. Кроме источника фиксированного нуля для расчётов и сравнения, Нулевой Регистр может использоваться как переменная типа WORD в длинной индексной адресации. Эта комбинация способа адресации и регистра позволяет адресоваться к любому участку памяти. Так как этот способ использует индексную адресацию, то доступ осуществляется медленнее, чем при Прямой Регистровой адресации.

Пример адресации с использованием Нулевого Регистра:

```
ADD AX, 1234[ZERO_REG]; AX ← AX + MEM_WORD(1234) (ADD_2op)
POP 5678[ZERO_REG]          ; MEM_WORD(5678) ← MEM_WORD(SP)
                                ; SP ← SP + 2
```

Рабочие регистры:
AX - 16-и битный регистр.

4.2.6 Адресация с использованием Указателя Стека

Байты 18h и 19h нижнего Регистрового Файла содержат Указатель Стека (SP), к которому адресуются по адресу 18h. Кроме обычных операций Указатель Стека (SP) может также использоваться как переменная типа WORD в Косвенной адресации для доступа к вершине стека или в Короткой Индексной адресации для доступа к данным внутри стека.

Пример адресации с использованием Указателя Стека:

```
PUSH [SP]          ; Duplicate TOP_OF_STACK
                   ; (скопировать вершину стека)
LD   AX, 2[SP]     ; AX ← NEXT_TO_TOP
```

Рабочие регистры:
AX – 16-и битный регистр.

4.3 Выборы способов адресации на языке ассемблера

Ассемблер упрощает выбор режимов адресации. Это упрощает задачу программирования, и эти возможности ассемблера следует применять там, где только это возможно.

4.3.1 Прямая адресация

Ассемблер выбирает между прямой регистровой адресацией и адресацией с использованием нулевого регистра, в зависимости от местоположения операнда в памяти. Необходимо обратиться к операнду с его символическим именем. Если операнд находится в младшем файле регистров, ассемблер выбирает прямую адресацию. Если операнд находится в другой области памяти, ассемблер выбирает адресацию с нулевым индексом.

4.3.2 Индексная адресация

Ассемблер выбирает между короткой индексной и длинной индексной адресациями в зависимости от значения индекса. Если значение может быть выражено в восьми битах, язык ассемблера выбирает короткую индексную адресацию. Если значение больше 8-битного, выбирается длинная индексная адресация.

4.4 Стандарты и соглашения программного обеспечения

При разработке программного обеспечения рекомендуется, чтобы использовались соглашения, принятые для редактирования на основе языка «Си». Эти стандарты применимы и для ассемблера, и для программной среды языка «Си», и они обеспечивают совместимость между этими средами.

4.4.1 Использование регистров

В 256-байтном младшем файле регистров содержатся регистры специальных функций центрального процессора и указатель стека. Остаток младшего файла регистров и старший файл регистров доступны для использования. Периферийные регистры специальных функций (SFRs) и распределенные в карте памяти SFR расположены в верхней области адресного пространства. Периферийные SFR могут быть перемещены через «окна» в младший файл регистров для прямого доступа. Распределенные в карте памяти SFR не могут быть использованы в режиме «окон». Можно использовать косвенную или индексную адресацию, чтобы получить доступ к ним. Все SFR оперируют как BYTE или WORD, если не определено иначе.

Язык программирования «Си» адаптирован к простой эффективной стратегии, размещает восемь байтов, начинающиеся в адресе 1CH, во временное хранение и рассматривает оставшуюся область в файле регистров как сегмент запоминающего устройства, размещённый там, где необходимо.

Примечание – Использование любого SFR как базового или индексного регистра для косвенных или индексных инструкций может вызвать непредсказуемые результаты, потому что внешние события могут изменить содержимое SFRs. Так как некоторые SFRs очищаются при чтении, допустимо использование SFR как операнда в инструкциях "чтение-модификация-запись" (например, XORB).

4.4.2 Адресация 32-битных операндов

32-битные операнды (DOUBLE WORD и LONG INTEGER) сформированы двумя смежными 16-битными словами в запоминающем устройстве. Младшее слово DOUBLE WORD находится всегда в младшем адресе, даже когда данные находятся в стеке (это означает, что старшее слово должно быть выдвинуто в стек первым). Адрес 32-битного операнда – адрес его младшего байта.

Аппаратные средства поддерживают 32-разрядные типы данных как операнды в инструкциях сдвига, делимое в делении 32/16 и произведение в умножении 16×16. Для этих действий 32-битный операнд должен располагаться в младшем файле регистров и должен быть выровнен по адресу, кратному четырем.

4.4.3 Соединение подпроцедур

Параметры передаются к подпроцедурам через стек. Параметры выдвигаются в стек от самого правого параметра налево. Параметры на 8 битов выдвигаются в стек со старшим неопределенным байтом. 32-битные параметры выдвигаются в стек как два 16-битных; старшая часть параметра выдвигается в стек первой. Как пример рассмотрим следующую процедуру:

Void example_procedure (char param1, long param2, int param3)

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

param3
low word of param2
high word of param2
undefined; param1
return address ← Stack Pointer

Если процедура возвращает значение к коду запроса (в противоположность изменению более глобальных переменных), результат возвращается в область временного хранения (TMPREG0 в этом примере) начинающуюся в 1CH. TMPREG0 рассматривается или как 8-, 16- или 32-битная переменная, в зависимости от типа процедуры.

Стандарт вызова, принятый языком «Си», имеет несколько следующих ключевых особенностей.

Процедуры могут всегда предполагать, что восемь байтов файла регистров, начинающиеся в 1CH, могут использоваться для временного хранения в пределах тела процедуры.

Код, который вызывает процедуру, должен учитывать, что процедура изменяет восемь байтов файла регистров, начинающиеся в 1CH.

Код, который вызывает процедуру, должен предполагать, что процедура изменяет слово состояния процессора (PSW), потому что процедуры не сохраняют и не восстанавливают PSW.

Результат процедур всегда возвращается в переменную TMPREG0.

Язык «Си» позволяет определять прерывание процедур, которые выполняются. Прерывание процедур не соответствует правилам нормальных процедур. Параметры нельзя передать к этим процедурам, и они не могут возвратить результаты. Так как прерывание процедуры можно выполнить по существу в любое время, они должны сохранить и восстанавливать значения PSW и TMPREG0.

4.5 Защита и руководящие принципы программного обеспечения

В микроконтроллере реализовано несколько механизмов восстановления после ошибок программного обеспечения и аппаратных средств. Прерывание по невыполнимому коду инструкции обеспечивает защиту от выполнения некорректного кода инструкции. Команда аппаратного сброса (RST) может вызвать сброс, если программный счетчик выходит за границы. Код инструкции RST – FFH, поэтому процессор сбросит себя, если попробует выполнить инструкцию от незапрограммированных ячеек в энергонезависимом запоминающем устройстве или из шины, линии которой подключены к высокому уровню. Сторожевой таймер (WDT) может также сбросить микроконтроллер в случае аппаратной или программной ошибки.

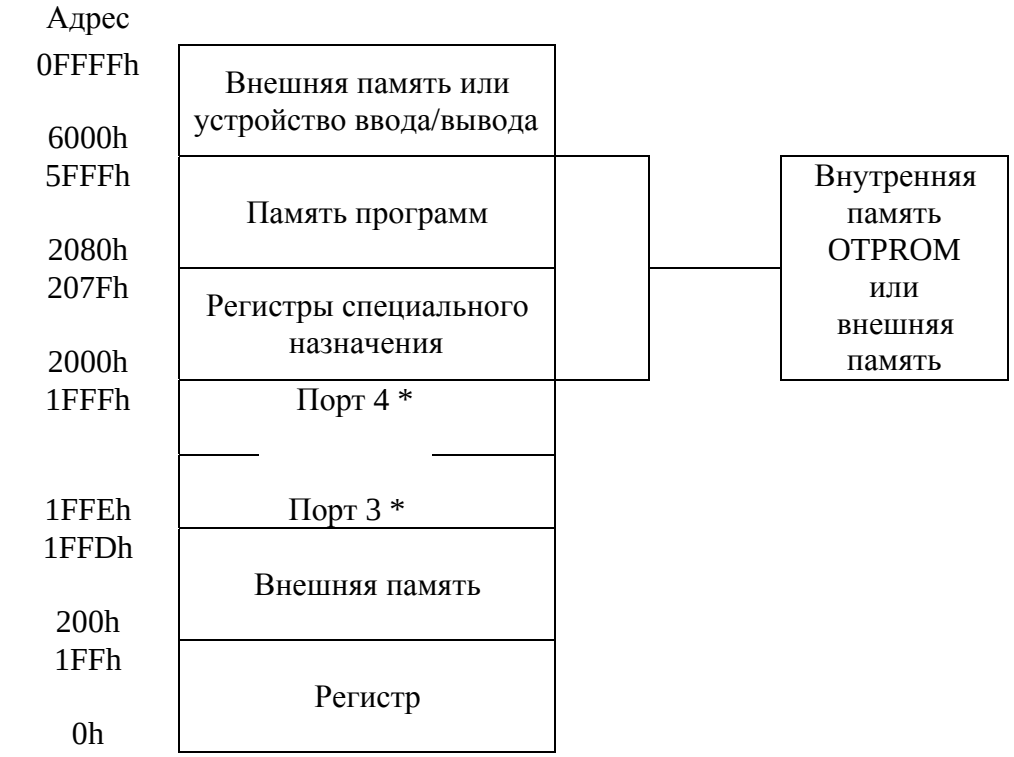
Рекомендуется заполнить неиспользованные области кодом с NOPs и периодическими переходами к процедуре обслуживания ошибок или RST инструкции. Это особенно важно для кодов окружения таблиц поиска. Везде, где место позволяет, необходимо окружить каждую таблицу семью NOPs (потому что самая длинная инструкция устройства имеет семь байтов) и RST или переходами к процедуре обслуживания ошибок. Так как RST – однобайтовая инструкция, NOPs не нужны, если RSTs используются вместо переходов к процедуре ошибки. Это гарантирует быстрое восстановление от ошибки программного обеспечения.

При использовании сторожевого таймера (WDT) для защиты программного обеспечения мы рекомендуем сбрасывать WDT только в одном месте программы, сокращая возможность нежелательного сброса WDT. Секция кода, который сбрасывает WDT, должна контролировать другие секции кода для правильного выполнения инструкций. Это может быть сделано проверкой переменных, чтобы удостовериться, что они – в пределах разумных значений. Использование программного таймера, чтобы сбросить WDT каждые 10 миллисекунд, обеспечивает защиту только от катастрофических ошибок.

5 Распределение памяти

Этот раздел описывает адресное пространство памяти микроконтроллеров. Изделия имеют 64К байт адресного пространства, большая часть которого предназначена для программ или запоминания данных. Каждый элемент памяти – ячейка памяти – содержит один байт.

Рисунок 5.1 показывает основные разделы памяти.



* Порты 3 и 4 всегда читаются и записываются как одно слово.

Рисунок 5.1 – Карта адресов памяти

В таблице 5.1 приводятся начальные и конечные адреса в 16-ричной и 10-тичной системах исчисления. В последующих подразделах описывается каждый раздел памяти.

Таблица 5.1 – Старшие адреса памяти МК

Обозначение	Диапазон адресов	
	16-ричные	10-тичные
Внешняя память или устройства ввода-вывода	6000h -0FFFFh	24576 - 65635
Память программ *	2080h - 5FFFh	8320 - 24575
Память специального назначения *	2000h - 207Fh	8192-8319
Порты 3 и 4 **	1FFEh - 1FFFh	8190-8191
Внешняя память	200h - 1FFDh	512-8189
Регистровый файл (включая SFR)	0h - 1FFh	0-511

* Расположены во внутреннем OTPROM или внешней памяти.
** Порты 3 и 4 только пословно-адресуемые.

5.1 Разделы внешней памяти

Верхняя часть карты памяти и раздел прямо над регистровым файлом всегда назначаются на внешнюю память или порты ввода-вывода (I/O) (смотрите рисунок 5.1 и таблицу 5.1). МК может сопрягаться со множеством устройств внешней памяти.

5.2 Порты 3 и 4

Порты 3 и 4 читаются и записываются как слово по адресу 1FFEh (Порт 3 – младший байт; Порт 4 – старший байт). Эти порты работают либо как порты ввода-вывода, либо как мультиплексированная шина адрес/данные, либо их комбинация, в зависимости от EA# сигнала при сбросе. Для получения более подробной информации смотрите раздел 6.

5.3 Программная память и память специального назначения

Программная память и память специального назначения занимают раздел памяти 16К байт (смотрите рисунок 5.1 и таблицу 5.1). Эти разделы могут постоянно располагаться либо во внутреннем OTPROM, либо во внешней памяти.

5.3.1 Выбор внутренней или внешней памяти

Доступ к адресам программной памяти и адресам памяти специального назначения осуществляется в зависимости от того, где находится данный раздел: во внутреннем OTPROM или во внешней памяти, причём он не может находиться там и там. Значение вывода внешнего доступа (EA#) во время переднего фронта сигнала RESET# определяет вид доступа: внутренний или внешний. Это значение запоминается в устройстве и не может быть изменено до сброса устройства. Если EA# имеет низкий уровень, то внутреннее OTPROM недоступно, и все обращения к этому адресному диапазону направляются прямо во внешнюю память. Если EA# имеет высокий уровень, то обращение к этому адресному диапазону осуществляется во внутреннее OTPROM.

Примечание – Вывод EA# должен иметь низкий уровень для надежного функционирования ЦПУ без внешней памяти.

5.3.2 Программная память

Программная память располагается по адресам 2080h – 5FFFh. После сброса устройства ЦПУ выбирает и выполняет команду, находящуюся по адресу 2080h.

Примечание – Так как исходное значение ячеек внутренней памяти OTPROM равно 0FFh, то рекомендуется записывать 0FFh в неиспользуемые ячейки программной памяти.

5.3.3 Память специального назначения

Память специального назначения расположена по адресам 2000h – 207Fh (смотрите рисунок 5.2). Она содержит несколько зарезервированных ячеек памяти, векторы для периферийного сервера (PTS) и стандартных прерываний, ключ защиты и байт конфигурации кристалла (CCB). Таблица 5.2 даёт список адресов памяти специального назначения и начальные и конечные адреса в 16-ричной и 10-тичной системах.

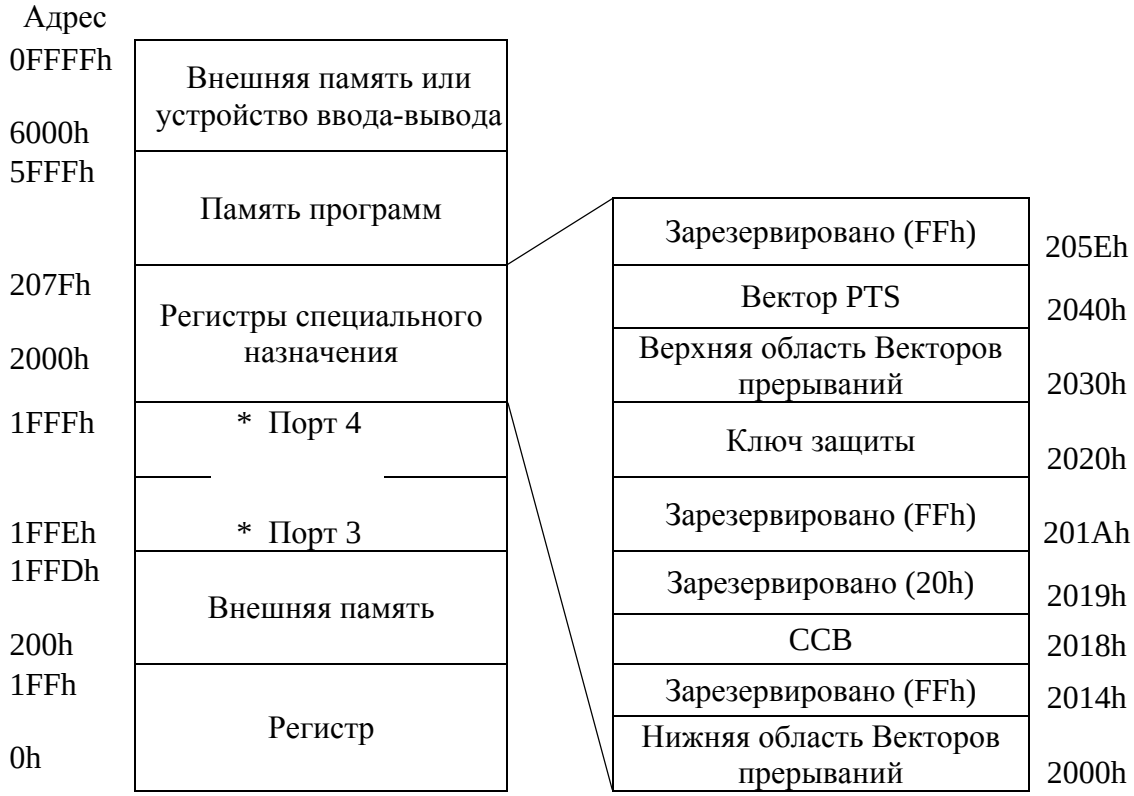


Рисунок 5.2 – Карта памяти специального назначения

Зарезервированные ячейки памяти

Несколько ячеек памяти зарезервировано для тестирования или будущего использования. Пользователь не может читать или записывать в эти зарезервированные ячейки памяти. Содержимое и/или функции этих ячеек могут меняться в будущих версиях, поэтому не гарантируется работа программного обеспечения при использовании зарезервированных ячеек

При программировании OTPROM всегда записывают 20h по адресу 2019h и 0FFh по всем другим зарезервированным адресам.

Векторы Прерываний

Верхняя и нижняя области Векторов Прерываний содержат адреса подпрограмм обслуживания прерываний. PTS-векторы содержат адреса управляющих блоков PTS.

Таблица 5.2 – Адреса памяти специального назначения

Обозначения	Адресный диапазон	
	16-ричные	10-ричные
Зарезервировано (должен содержать 0FFh)	205Eh – 207Fh	8286 – 8319
Векторы PTS	2040h – 205Dh	8256 – 8285
Старшие векторы прерывания	2030h – 203Fh	8240 – 8255
Ключ безопасности	2020h – 202Fh	8224 – 8239
Зарезервировано (должен содержать 0FFh)	201Ah – 201Fh	8218 – 8223
Зарезервировано (должен содержать 20h)	2019h	8217
CCB	2018h	8216
Зарезервировано (должен содержать 0FFh)	2014h – 2017h	8212 – 8215
Младшие векторы прерывания	2000h – 2013h	8192 – 8211

Ключ Защиты

Ключ безопасности защищает МК от несанкционированного чтения и записи в OPTROM. Если security lock bits (блокировочные биты) в регистре конфигурации кристалла (CCR.6 и CCR.7) очищаются, то каждый байт 16-байтного программируемого кода защиты сравнивается с соответствующими байтами во внешней памяти. Если внешние данные не совпадают с ключом защиты, устройство не входит в запрашиваемый режим программирования.

Байт Конфигурации Кристалла (CCB)

Байт Конфигурации Кристалла (CCB) – первый байт, выбранный из памяти после сброса устройства. Последовательность сброса загружает CCB во внутренний регистр, называемый регистром конфигурации кристалла (CCR). CCR управляет защитой внутренней памяти, режимом READY, сигналом управления шиной, разрядностью шины и режимом POWERDOWN (режимом пониженного энергопотребления).

5.4 Регистровый файл

Регистровый файл разделён на верхний и нижний регистровые файлы (смотрите рисунок 5.3). Верхний регистровый файл содержит регистровое ОЗУ общего назначения. Нижний регистровый файл содержит регистровое ОЗУ общего назначения, Указатель стека (SP) и регистры специальных функций (SFR). В таблице 5.3 приведены начальные и конечные адреса в 16-ричной и 10-тичной системах.

* Нижний Регистровый Файл

Рисунок 5.3 – Карта памяти регистрового файла

Примечание - Регистровый Файл не должен содержать коды команд. Попытка выполнить команду из регистрового файла (0h – 01FFh) ведёт к автоматической выборке контроллером команды из внешней памяти. Эта секция внешней памяти зарезервирована для использования средствами разработки.

Таблица 5.3 – Адреса памяти регистрового файла

Обозначение	Адресный диапазон	
	16-ричные	10-ричные
Старший Регистровый Файл *	100h - 1FFh	256-511
Регистровое RAM (ОЗУ) Общего назначения **	1Ah - 0FFh	26-255
Указатель Стека (SP) **	18h - 19h	24-25
Регистры специального назначения (SFR) **	0h - 17h	0-23

_____

* Содержит регистровое RAM (ОЗУ) общего назначения (доступно с помощью косвенной или индексной адресации, если только нет использования вертикальных окон).

** Расположены в нижнем регистровом файле (доступны с помощью прямой регистровой, косвенной или индексной адресации).

5.4.1 Регистровое ОЗУ общего назначения

Ячейки 1Ah – 0FFh содержат регистровое RAM (ОЗУ) общего назначения. Ячейки указателя стека 18h – 19h могут также использоваться в качестве регистрового ОЗУ общего назначения, когда операции со стеком не проводятся. РАЛУ может прямо обращаться к этой памяти, используя прямую регистровую адресацию. Верхний Регистровый Файл также содержит регистровое ОЗУ общего назначения (100h – 1FFh). Обычно РАЛУ обращается к этой памяти, используя косвенную или индексную адресацию. Технология Vertical windowing (создание вертикальных окон) позволяет РАЛУ использовать прямую регистровую адресацию для доступа к ОЗУ в верхнем регистровом файле (смотрите раздел 3, где описаны различия между прямой регистровой и индексной адресациями). Создание вертикальных окон обеспечивает быстрое переключение на задачи по прерываниям и ускоряет выполнение программ (смотрите подраздел 5.6).

5.4.2 Указатель стека (SP)

Ячейки памяти 18h и 19h содержат указатель стека (SP). SP содержит адрес вершины стека. SP должен указывать на адрес слова (четный), который на два байта больше, чем желаемый стартовый адрес. Перед тем, как ЦПУ выполнит вызов подпрограммы или программы обслуживания прерывания, оно дважды декрементирует содержимое SP и копирует (PUSH) адрес следующей команды из программного счётчика в стек. Затем загружает адрес подпрограммы или программы обслуживания прерываний в программный счётчик. После завершения выполнения подпрограммы или программы обслуживания прерывания, ЦПУ возвращается из подпрограммы (командой RET) и загружает (командой POP) содержимое вершины стека (т. е. адрес возврата) в программный счётчик.

Подпрограммы могут быть вложенными. Это значит, что каждая подпрограмма может вызывать другую подпрограмму. ЦПУ засылает содержимое программного счётчика в стек каждый раз при выполнении вызова подпрограммы. Стек растёт вниз по мере добавления содержимого. Глубину стека ограничивает только величина доступной памяти. Каждый раз при возврате из подпрограммы ЦПУ выгружает адрес из вершины стека, и потом вершина стека перемещается на следующий адрес возврата. Когда операции со стеком не проводятся, ячейки SP могут использоваться в качестве регистрового ОЗУ общего назначения.

5.4.2.1 Инициализация указателя стека

Пользователь программ должен загрузить двухбайтный адрес в указатель стека. Выбор адреса, на два байта большего, чем желаемый стартовый адрес, делается потому, что указатель стека автоматически декрементируется перед тем, как ЦПУ засылает первый байт адреса возврата в стек. Следует помнить, что стек растёт вниз, что требует

достаточного количества ячеек для максимального числа адресов, занесённых в стек. Стек может находиться либо во внутреннем Регистровом Файле, либо во внешнем ОЗУ.

5.4.3 Регистры специальных функций (SFRs)

Ячейки 00h - 17h обеспечивают доступ к регистрам специальных функций (SFR) ЦПУ через три горизонтальных окна (HWindow 0, 1 и 15, смотрите подраздел 5.5.). РАЛУ осуществляет прямое управление всеми периферийными модулями, кроме Портов 3 и 4, через SFR.

При использовании SFR в качестве базового или индексного регистра при косвенной или индексной адресациях следует отдавать себе отчёт в том, что содержимое SFR непредсказуемо. Внешние события могут менять содержимое SFR, и некоторые SFR очищаются при считывании.

Функции многих SFR различны в зависимости от того, считывается ли из них или записывается в них информация. По этой причине никогда не следует использовать SFR в качестве операнда в команде чтение-модификация-запись. Не следует использовать зарезервированные SFR; запишите в них 0 или оставьте их в исходном состоянии. При считывании зарезервированные биты и SFR могут возвращать неопределённые значения.

5.5 Создание горизонтальных окон

Микроконтроллеры используют технологию horizontal windowing (создания горизонтальных окон), которая переключает три 24-байтных блока памяти (HWindow 0, 1 и 15) внутри младших 24 байтов в нижнем Регистровом Файле (смотрите рисунок 5.4). Каждое HWindow (горизонтальное окно) обеспечивает чтение или запись в определенные SFR. Некоторые регистры доступны как отдельный байт; другие как слово (два байта). Некоторые регистры, такие как регистр выбора окна (WSR, 14h), доступны во всех 3-х окнах, другие могут быть доступны для записи в одном окне, а для считывания в другом.

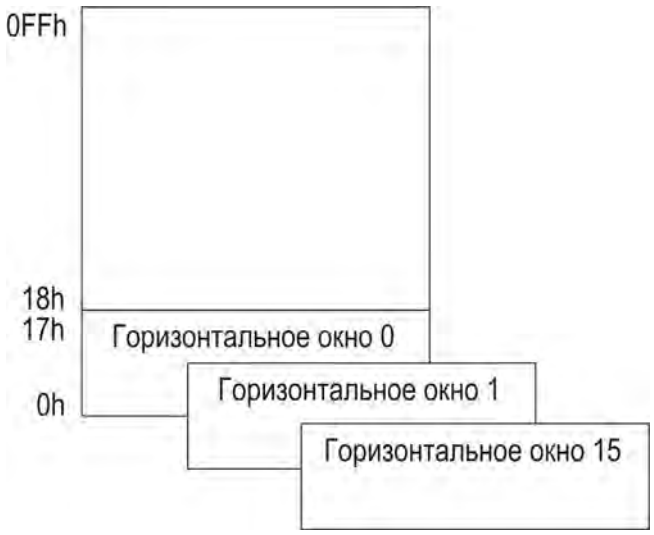


Рисунок 5.4 – Создание горизонтальных окон

5.5.1 Выбор горизонтального окна

Регистр выбора окна (WSR,14h) обеспечивает доступ к горизонтальным и вертикальным окнам (смотрите подраздел 5.6). Для выбора HWindow необходимо занести номер желаемого окна в WSR.0 – WSR.3 и очистить WSR.4 – WSR.6. Доступны только окна HWindow 0, 1 и 15. Все другие окна HWindow зарезервированы. Таблица 5.4 показывает необходимое содержимое WSR для выбора каждого HWindow.

Таблица 5.4 – Выбор горизонтальных окон

HWindow	Содержимое WSR
0	X000 0000B = 00h
1	X000 0001B = 01h
15	X000 1111B = 0Fh

5.5.2 Горизонтальное окно 0 (HWindow 0)

HWindow 0 – начальное окно. Оно обеспечивает доступ к чтению из 19 регистров и записи в 21 регистр (смотрите рисунок 5.5). Некоторые регистры (например, INT_MASK1) доступны для чтения и для записи внутри HWindow 0. Другие (например, IOS1) могут быть доступны или только для чтения, или только для записи внутри HWindow 0. Для проведения недостающих функций в этих регистрах следует выбирать Hwindow 15.

5.5.3 Горизонтальное окно 1 (HWindow 1)

HWindow 1 обеспечивает доступ для чтения и для записи в 12 регистров (смотрите рисунок 5.5). Некоторые регистры также доступны в двух окнах – Hwindow 0 и HWindow 15.

Адрес регистра в окне	HWindow 0 чтение	HWindow 0 запись	HWindow 1 чтение/запись
17h	IOS2		
16h	IOS1	IOC1	
15h	IOS0	IOC0	Reserved
14h	WSR	WSR	WSR
13h			
12h			
11h			Reserved
10h	IOPORT2	IOPORT2	Reserved
0Fh	IOPORT1	IOPORT1	Reserved
0Eh	IOPORT0		Reserved
0Dh	TIMER2(HI)	TIMER2(HI)	Reserved
0Ch	TIMER2(LO)	TIMER2(LO)	IOC3
0Bh	TIMER1(HI)	IOC2	Reserved
0Ah	TIMER1(LO)	WATCHDOG	Reserved
09h			
08h			
07h	SBUF(RX)	SBUF(TX)	PTSSRV(HI)
06h			PTSSRV(LO)
05h			PTSSEL (HI)
04h			PTSSEL (LO)
03h			
02h			Reserved
01h			
00h			

Рисунок 5.5 – HWindow 0 и HWindow 1

5.5.4 Горизонтальное окно 15 (HWindow 15)

HWindow 15 обеспечивает доступ к тем же регистрам, что и HWindow 0, кроме байтов 0Ch – 10h. Через байты 0Ch – 0Dh в HWindow 0 осуществляется доступ к 16-разрядному регистру TIMER2, а в HWindow 15 – к 16-разрядному регистру

T2CAPTURE. Через байты 0Eh – 10h в HWindow 0 осуществляется доступ к регистрам IOPORT0, IOPORT1, IOPORT2, а в HWindow 15 они зарезервированы. Регистры, доступные только для чтения в HWindow 0, доступны только для записи в HWindow 15 и наоборот.

Адрес регистра в окне	HWindow 15 чтение	HWindow 15 запись
17h	PWM0_CONTROL	IOS2
16h	IOC1	IOS1
15h	IOC0	IOS0
14h	WSR	WSR
13h	INT_MASK1	INT_MASK1
12h	INT_PEND1	INT_PEND1
11h	SP_CON	SP_STAT
10h	Reserved	Reserved
0Fh	Reserved	Reserved
0Eh	Reserved	Reserved
0Dh	T2CAPTURE(HI)	T2CAPTURE(HI)
0Ch	T2CAPTURE(LO)	T2CAPTURE(LO)
0Bh	IOC2	TIMER1 (HI)
0Ah	WATCHDOG	TIMER1 (LO)
09h	INT_PEND	INT_PEND
08h	INT_MASK	INT_MASK
07h	SBUF(TX)	SBUF(RX)
06h	HSO_COMMAND	HSI_STATUS
05h	HSO_TIME (HI)	HSI_TIME (HI)
04h	HSO_TIME (LO)	HSI_TIME (LO)
03h	HSI_MODE	AD_RESULT (HI)
02h	AD_COMMAND	AD_RESULT (LO)
01h	ZERO_REG (HI)	ZERO_REG (HI)
00h	ZERO_REG (LO)	ZERO_REG (LO)

Рисунок 5.6 – HWindow 15

5.6 Создание вертикальных окон

Создание вертикальных окон преобразует области верхнего регистрового файла в блоки старших ячеек нижнего регистрового файла, размером в 32-, 64- или 128 байт, известные как вертикальные окна – VWindows. Микроконтроллеры комплекта имеют 16 32-байтных VWindows, 8 64-байтных VWindows и 4 128-байтных VWindows. Пример 128-байтного окна показан на рисунке 5.7.

Рисунок 5.7 – Создание вертикальных окон

5.6.1 Выбор вертикального окна

Регистр Выбора Окна (WSR, 14h) обеспечивает доступ к горизонтальным – HWindow и вертикальным - VWindow окнам. Необходимо установить WSR.4, WSR.5 или WSR.6 для выбора 128-, 64- или 32-байтного VWindows соответственно (смотрите рисунок 5.8). Следует записать номер VWindows в младшие биты WSR для выбора вертикального окна, показанного на рисунке 5.7, загрузить 17h в WSR.

Рисунок 5.8 – Установка битов регистра выбора окна

5.6.2 Создание вертикальных окон и способы адресации

При создании вертикальных окон возможны:

- команды с прямой регистровой адресацией, которые используют адрес внутри окна нижнего Регистрового Файла, в действительности получают доступ к VWindow в верхнем регистровом файле;
- команды с косвенной или индексной адресацией, использующие адрес внутри окна в нижнем Регистровом Файле или в VWindow, получают доступ к действительной ячейке памяти.

Примечание - Косвенные операции сдвига выполняются неправильно, если WSR = X100 0000.

Расположенные ниже команды иллюстрируют различие между прямой регистровой и индексной адресациями при использовании вертикальных окон:

```
PUSHA              ; pushes the contents of WSR onto the stake
                   ; (засылает содержимое WSR в стек)
LDB WSR, #17h      ; select VWindow 7, a 128-byte block
                   ; (выбор VWindow 7, 128-байтный блок)
                   ; The next instruction uses register-direct addr
                   ; (следующая команда использует прямую регистровую
                   ; адресацию)

ADD 40h, 80h       ; mem_word(40h) ← mem_word(40h) +mem_word(380h)
                   ;The next two instructions use indirect addr
                    ; (следующие две команды используют косвенную адресацию)
ADD 40h, 80h       ; mem_word(40h) ← mem_word(40h) + mem_word(80h+0)
                   ;
ADD 40h, 380h[0]   ; mem_word(40h) ← mem_word(40h) +mem_word(380h+0)
                ;
POPA ; reloads the previous contents into WSR ; (перегружает в WSR
                   первоначальную информацию)
```

6 Прерывания

Основной функцией микроконтроллера является обеспечение управления устройствами в реальном времени. Схема контроллера прерываний позволяет событиям реального времени управлять выполнением программы. Когда событие вызывает прерывание, ЦПУ обслуживает это прерывание перед выполнением следующей команды. Встроенная периферия, внешний сигнал или команда могут выставить запрос на прерывание. В простейшем случае МК получает запрос прерывания, осуществляет обслуживание и возвращается к прерванной программе. Данный раздел описывает схему управления прерываниями, схему приоритетов и синхронизацию. Здесь также описывается три специальных прерывания и объясняется программирование и управление прерываниями.

6.1 Обработка прерываний

В микроконтроллерах 1874BE76T, 1874BE06T, 1874BE05T имеются две возможности обслуживания прерываний: программное обслуживание прерываний через контроллер прерываний и микропрограммное обслуживание прерываний через PTS. Можно выбрать вариант обслуживания прерываний для каждого из маскируемых прерываний (смотрите подраздел 6.5.1). Немаскируемые прерывания (NMI, программное прерывание, прерывание по неподдерживаемому коду) всегда обрабатываются подпрограммами обработки прерываний. Рисунок 6.1 иллюстрирует процесс обработки прерываний.

6.1.1 Контроллер прерываний

Контроллер прерываний обслуживает прерывания совместно с подпрограммами обработки прерываний. Когда аппаратура обнаруживает прерывание, она генерирует и выполняет специальный вызов подпрограммы обслуживания прерывания. При этом в стек помещается содержимое программного счётчика, а затем счетчик команд (PC) загружается значением, соответствующим вектору прерывания. Верхние и нижние области векторов прерываний, расположенные в специально отведённой памяти (смотрите раздел 5), содержат адреса подпрограмм обслуживания прерываний. ЦПУ выполняет подпрограмму обслуживания прерывания. После выполнения этой подпрограммы PC перезагружается значением из стека, и выполнение прерванной программы продолжается.

6.1.2 Периферийный Сервер (PTS)

PTS является микропрограммным аппаратным обработчиком прерываний. Он может использоваться вместо стандартных подпрограмм обработки прерываний для любых маскируемых прерываний. PTS обслуживает прерывания с небольшой задержкой; он не модифицирует стек или PSW, что позволяет нормально продолжить выполнение прерванной программы. По этим причинам PTS может обслуживать прерывание за время, необходимое для выполнения одиночной команды.

PTS работает в пяти специальных микропрограммных режимах, которые позволяют PTS выполнять отдельные задачи гораздо быстрее, чем подпрограммы обслуживания прерывания. Смотрите описание режимов PTS в подразделе 6.6.

Каждое прерывание PTS требует блока данных, называемого блоком управления PTS (PTSCB). Если имеет место прерывание PTS, то декодировщик приоритетов выбирает соответствующий вектор и вызывает PTSCB.

PTSCB определяет режим, общее число передач (при необходимости), общее число циклов, которые будут выполняться пока PTS не потребует дополнительного обслуживания и источник и/или приёмник данных при передаче (при необходимости). Каждое прерывание PTS генерирует один цикл PTS. Рисунок 6.2 показывает цикл работы PTS.

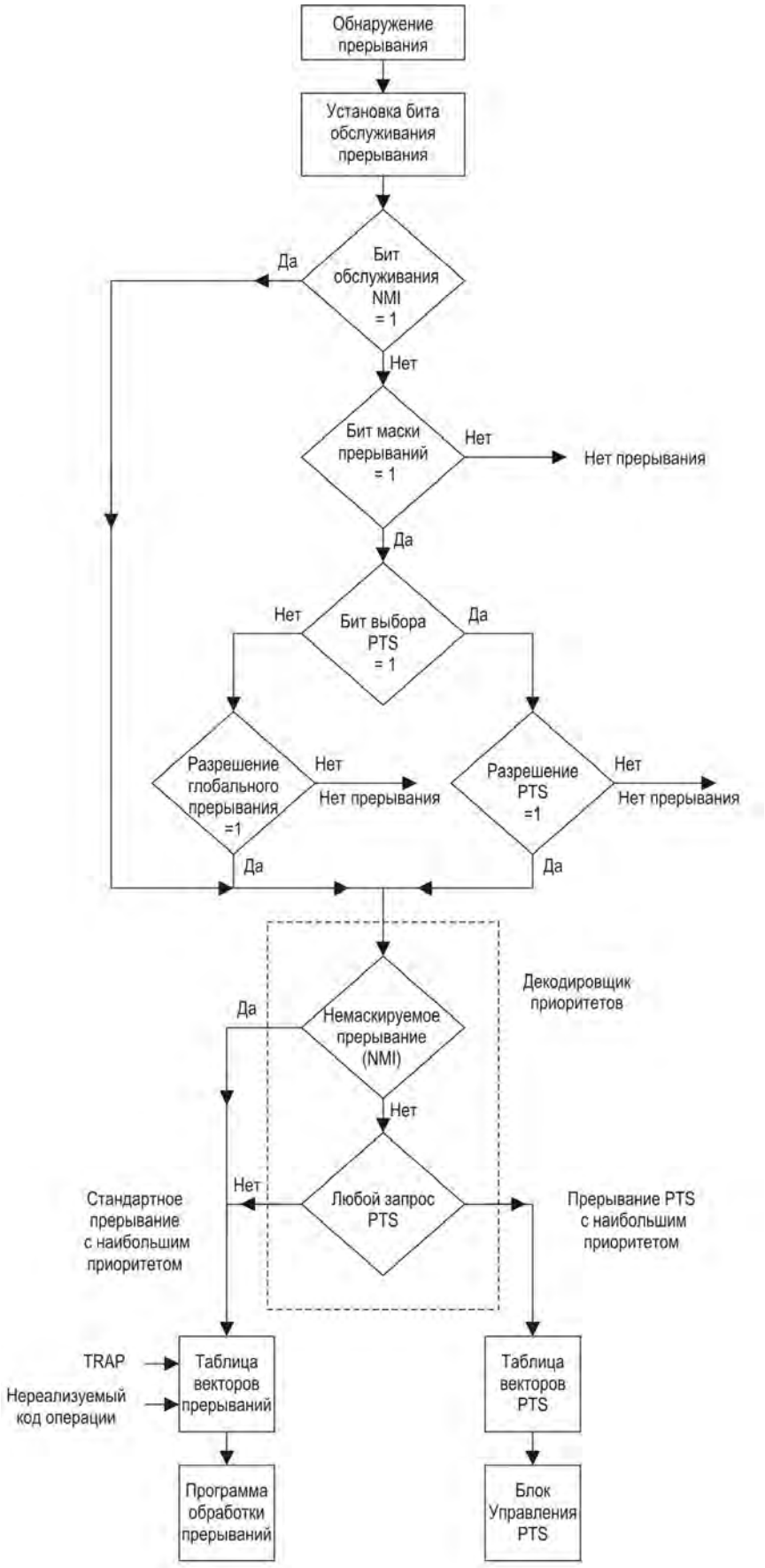


Рисунок 6.1 – Блок-схема стандартного и PTS прерываний

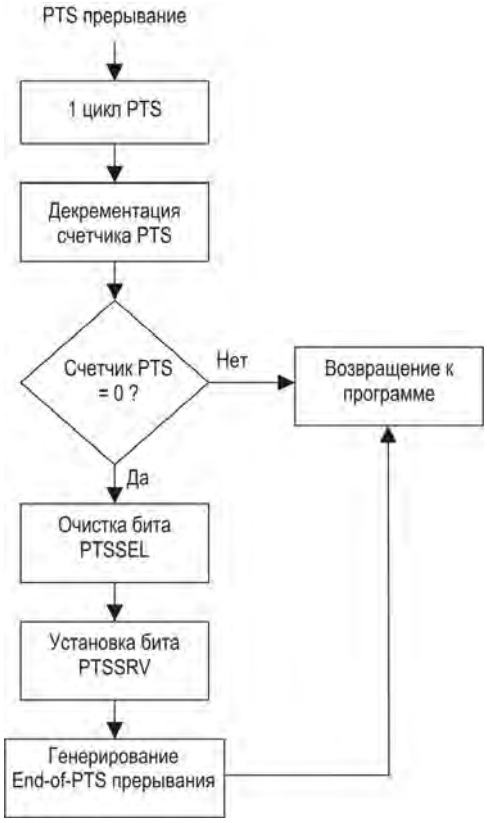


Рисунок 6.2 – Блок-схема цикла PTS

6.2 Приоритеты прерываний

Прерывания по неподдерживаемым командам и программное прерывание не имеют приоритетов; они напрямую проходят в контроллер прерываний для обслуживания. Контроллер прерываний выбирает соответствующий вектор, расположенный в специально отведённой памяти (смотрите раздел 5). Вектор содержит стартовый адрес программы обработки прерываний.

Декодировщик приоритетов определяет приоритеты всех других ожидаемых запросов прерывания. Таблица 6.1 показывает установленные по умолчанию приоритеты прерываний (15 – высший и 1 – низший). NMI имеет высший приоритет над всеми прерываниями, имеющими приоритет. Если зафиксирован NMI, декодировщик приоритетов определяет его как запрос с высшим приоритетом, и контроллер прерываний выбирает соответствующий вектор, расположенный в специально отведённой памяти (смотрите раздел 5). Вектор содержит стартовый адрес соответствующей подпрограммы обработки прерываний.

Любой запрос прерывания PTS имеет больший приоритет, чем любой запрос маскируемого прерывания. Если NMI не установлен, декодировщик приоритетов устанавливает высший приоритет для запроса прерываний PTS, и контроллер прерываний выбирает соответствующий вектор PTS, расположенный в специально отведённой памяти. Вектор содержит стартовый адрес соответствующего блока управления PTS (PTSCB).

Если нет запроса ни от NMI, ни от PTS, декодировщик приоритетов устанавливает наивысший приоритет запросу стандартных прерываний, и контроллер прерываний выбирает соответствующий вектор, расположенный в специально отведённой памяти. Вектор содержит стартовый адрес соответствующей подпрограммы обработки прерываний.

6.2.1 Модификация приоритетов прерываний

Программное обеспечение может модифицировать установленные по умолчанию приоритеты маскируемых прерываний через регистры маски прерываний (INT_MASK и

INT_MASK1). Например, можно определить, какие прерывания, если это необходимо, могут обслуживаться подпрограммой обработки прерывания, а какие в цикле PTS. Следующая программа показывает возможный вариант запрещения всех прерываний, кроме EXTINT (приоритет 7), в подпрограмме обработки прерывания при приёме в последовательном канале (приоритет 9).

```
SERIAL_RI_ISR:
    pusha                          ; Save PSW, INT_MASK, & WSR
    di                             ; Disable all interrupts except EXTINT
    ldb INT_MASK1, #00100000B      ;
    ei                             ; Enable interrupt servicing

                                   ; Service the RI interrupt

    popa                           ; Restore PSW, INT_MASK, & WSR regsret
    ret
```

Ячейка, которая расположена с адреса 2032h в таблице векторов прерывания, загружается значением метки SERIAL_RI_ISR, для разрешения прерывания приёма.

Таблица 6.1 – Источники, расположение и приоритеты векторов прерывания

Номер	Вектор Прерывания	Источник	Ячейка Вектора Прерывания	Ячейка Вектора PTS	Приоритеты *
Специальный	Неподдерживаемый код	Неподдерживаемый код	2012h		
Специальный	Программное прерывание	Команда TRAP	2010h		
INT15	NMI **	NMI	203Eh		15
Каждое из следующих маскируемых прерываний может быть назначено на PTS. Любой из PTS прерываний имеет приоритет над всеми другими маскируемыми прерываниями.					
INT14	HSI FIFO Full	Буфер FIFO HSI полный	203Ch	205Ch	14
INT13	EXTINT **	P2.2	203Ah	205Ah	13
INT12	Timer 2 Overflow	Переполнение таймера 2	2038h	2058h	12
INT11	Timer 2 Capture **	Захват значения таймера 2	2036h	2056h	11
INT10	HSI FIFO 4	Четвёртый вход в FIFO HSI	2034h	2054h	10
INT09	Receive	RI Флаг ***	2032h	2052h	9
INT08	Transmit	TI Флаг ***	2030h	2050h	8
INT07	EXTINT **	P2.2 или P0.7	200Eh	204Eh	7
INT06	Serial Port	RI Флаг и TI Флаг ****	200Ch	204Ch	6
INT05	Software Timer	Программный таймер 0-3, сброс таймера 2, запуск АЦП	200Ah	204Ah	5
INT04	HSI.0 **	Вход HSI.0	2008h	2048h	4
INT03	HSO	HSO.0 - HSO.5	2006h	2046h	3
INT02	HSI Data Available	Заполнение FIFO HSI или загрузка фиксирующего регистра HSI	2004h	2044h	2
INT01	A/D Conversion Complete	Завершение A/D преобразования	2002h	2042h	1
INT00	Timer Overflow	Таймер 1 или 2	2000h	2040h	0

Окончание таблицы 6.1

_____

* Прерывание по неподдерживаемым кодам и программное прерывание не имеют приоритетов. Они проходят напрямую в контроллер прерывания на обработку. NMI имеет наивысший приоритет из всех прерываний с приоритетом. Любое PTS-прерывание имеет больший приоритет, чем остальные маскируемые прерывания.

** Данные прерывания могут быть сконфигурированы как независимые внешние прерывания.

*** Если прерывание Serial Port замаскировано, а прерывания Receive и Transmit разрешены, RI флаг и TI флаг генерируют отдельные прерывания Receive и Transmit. Если не требуется совместимости с 8096BH, то эта конфигурация предпочтительней.

**** Если прерывания Receive и Transmit замаскированы, а прерывание Serial Port разрешено, оба флага RI и TI генерируют прерывания Serial Port. Данная конфигурация обеспечивает совместимость с 8096BH.

Все программы обработки прерываний работают следующим образом:

1 После аппаратного детектирования и определения приоритетов запроса прерывания генерируется и выполняется вызов специального прерывания. Он помещает PC в стек и затем загружает его значением вектора прерывания, соответствующего наибольшему приоритету, если нет немаскируемого прерывания. Аппаратное обеспечение не может вызвать следующее прерывание, пока не выполнится первая команда подпрограммы обработки прерывания.

2 Команда PUSHA, которая гарантированно выполняется, сохраняет значение PSW, INT_MASK1 и WSR в стеке, а затем очищает PSW и INT_MASK1. Дополнительно к арифметическим флагам PSW содержит регистр INT_MASK, бит глобального разрешения прерываний (I) и бит разрешения PTS (PSE). Обнуление регистров PSW и INT_MASK1 маскирует все маскируемые прерывания, запрещает обработку стандартных прерываний и запрещает PTS. Команда PUSHA препятствует вызову прерывания, пока не выполнится следующая команда.

3 Команда LDB INT_MASK1 разрешает те прерывания, которые вы выбрали в программе обработки прерывания. В приведенном примере только EXTINT может прерывать программу обработки прерывания по приему информации. Разрешением и запрещением прерываний программное обеспечение устанавливает свои приоритеты обработки прерываний.

4 Команда EI разрешает обработку прерываний после выполнения следующей за ней команды.

5 Действующая программа обработки прерываний выполняется в соответствии со структурой приоритетов, установленной программным обеспечением.

6 В конце программы обработки команда POPA восстанавливает исходное содержимое регистров PSW, INT_MASK1 и WSR. Так как вызов прерывания не может произойти сразу же за командой POPA, то последняя команда RET будет выполнена до того, как произойдет другой вызов прерывания. Рассмотрим что будет, если команда POPA разрешает прерывания. Если контроллер прерываний отработал принятое прерывание, прежде чем команда RET выполнена, то адрес возврата в прерванную программу, которая выполнялась до того, как случилось данное прерывание, останется в стеке. Несмотря на то, что это не является проблемой для выполнения программы, всё же это может привести к переполнению стека, если прерывания происходят на большой частоте.

Следует заметить, что описанные “пролог” и “эпилог” не сохраняют и не восстанавливают регистры ОЗУ. Для программ обработки прерываний принято сохранять установки регистров пользователя из нижнего Регистрового Файла. Это возможно благодаря доступности 232 байт ОЗУ в нижнем Регистровом Файле. В дополнение к этому

ОЗУ из верхнего Регистрового Файла доступно через режим вертикальных окон (смотрите раздел 5).

6.3 Синхронизация прерываний

Пять внешних источников могут прервать микроконтроллер. Входная схема фиксирует прерывания по нарастающему фронту на входе. Сигнал на входе прерывания должен поддерживаться в высоком состоянии на время минимального импульса для уверенного распознавания. Одни прерывания фиксируются во время Phase1 (низкий CLKOUT); другие во время Phase2 (высокий CLKOUT). В таблице 6.2 приведены длительность минимального импульса и фаза синхронизации для каждого внешнего прерывания.

Таблица 6.2 – Минимальная длительность импульса прерывания и фаза приёма

Источник прерывания	Вектор прерывания	Номер	Минимальная Ширина Импульса	Фаза Фиксации
HSI.0	Вывод HSI.0	INT04	>2 тактов	Фаза 1
NMI	NMI	INT15	>1 тактов	Фаза 2
P0.7	EXTINT	INT07	>2 тактов	Фаза 1
P2.2	EXTINT	INT07	>2 тактов	Фаза 2
P2.2	EXTINT1	INT13	>2 тактов	Фаза 2
Захват значения таймера 2	Timer 2 Capture	INT11	>2 тактов	Фаза 2

Таблица 6.3 описывает шесть команд, которые всегда блокируют прерывания до тех пор, пока не выполнится следующая за ней команда.

Таблица 6.3 – Команды, блокирующие прерывания

Команда	Описание
DI	Запрещает прерывания
EI	Разрешает прерывания после выполнения следующего состояния
POPA	Считывает два слова из верхушки стека и помещает первое в регистровую пару INT_MASK1/WSR, а второе – в регистровую пару PSW/INT_MASK
POPF	Считывает слово из стека и помещает в регистровую пару PSW/INT_MASK
PUSHA	Записывает содержимое регистров PSW, INT_MASK, INT_MASK1 и Window Select в верхушку стека, затем очищает регистры PSW, INT_MASK и INT_MASK1
PUSHF	Записывает содержимое пары регистров PSW/INT_MASK в верхушку стека, затем очищает их.

Выполнение любой из следующих операций также блокирует прерывания, пока не выполнится следующая команда:

- Знаковый префикс команды (FE) для двухбайтных команд и команд знакового умножения и деления.
- Прерывание по неподдерживаемым кодам.
- Прерывание пошагового исполнения (программное прерывание).

6.3.1 Время ожидания прерываний

Время ожидания прерывания – это общая задержка между временем, когда произошла генерация прерывания (без подтверждения) и временем, когда МК начнёт выполнять или подпрограмму обработки прерывания, или цикл PTS. Задержка имеется между временем детектирования прерывания и временем его подтверждения.

Прерывание не получит подтверждения, пока текущая команда не будет выполнена. Если прерывание пришло менее чем за четыре такта до завершения текущей команды, то оно не будет подтверждено, пока не закончится следующая команда. Эта дополнительная задержка имеет место, так как команды предварительно выбираются и подготавливаются несколько периодов перед выполнением. Эта максимальная задержка между генерацией прерывания и его подтверждением составляет четыре такта плюс время выполнения следующей команды.

После завершения текущей команды, когда стандартное прерывание подтверждено, аппаратно очищается бит ожидания прерывания и принудительно вызывается адрес, содержащийся в соответствующем векторе прерывания. Процедура, которая выбирает вектор и производит вызов, требует 16 тактов. Если стек находится во внешнем ОЗУ, дополнительно требуется два такта для ожидания шины. Если подтверждено прерывание PTS, то сразу осуществляется переход по вектору PTSCB и начинается выполнение цикла PTS.

6.3.2 Вычисление времени ожидания

Максимальное время ожидания получается, если прерывание пришло слишком поздно для подтверждения во время текущей команды. Следующий наихудший случай получается, когда текущая команда блокирует прерывания (смотрите таблицу 6.3). Для подсчёта времени ожидания суммируются следующие условия:

- четыре такта для завершения команды;
- общее количество тактов следующей команды; наибольшая команда (NORMAL) содержит 39 тактов;
- время реакции (16 тактов при внутреннем стеке и 18 при внешнем) только для стандартных прерываний.

6.3.2.1 Время Ожидания Стандартного Прерывания

Максимальная задержка при стандартном прерывании составляет 61 такт (4 + 39 + 18). Эта задержка не включает время, необходимое для обработки первой команды в программе обработки прерываний. Рисунок 6.3 иллюстрирует пример наихудшего развития событий.

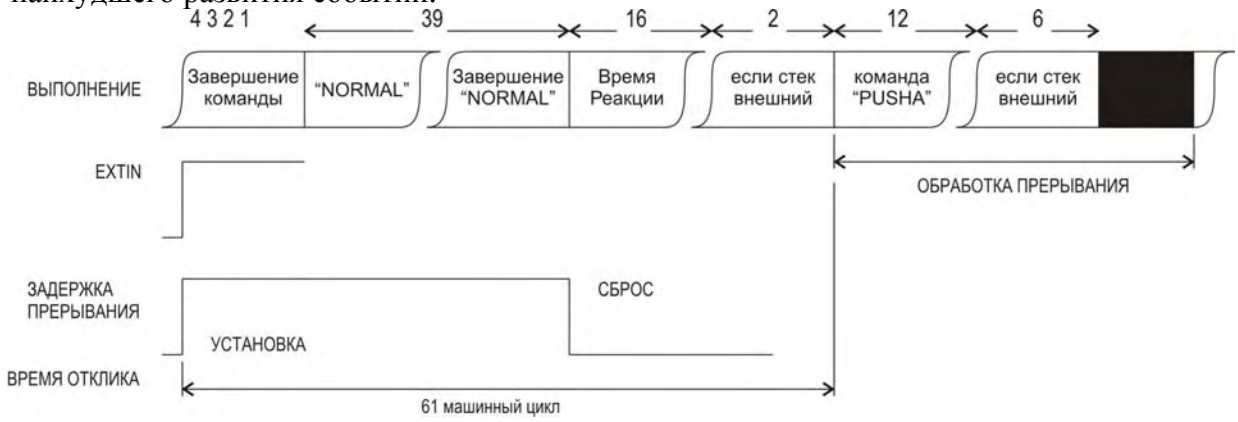


Рисунок 6.3 – Время реагирования на стандартное прерывание

6.3.2.2 Время Ожидания Прерывания PTS

Максимальная задержка для PTS прерывания составляет 43 такта (4 + 39), что иллюстрирует рисунок 6.4. Эта задержка не включает дополнительную задержку, когда PTS запрещён (PSW.2 очищен) или когда обслуживается прерывание PTS большего приоритета. В таблице A.11 приложения A приведено время выполнения PTS цикла.

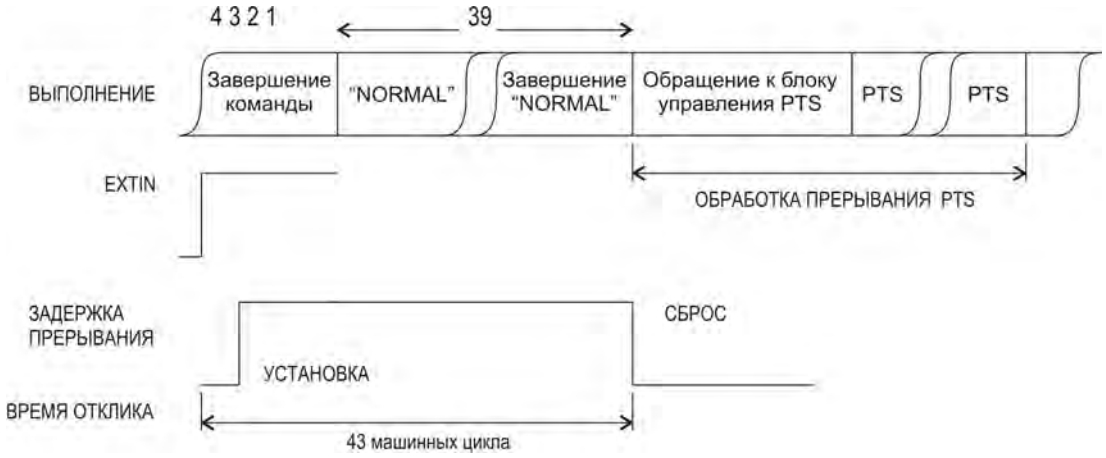


Рисунок 6.4 – Время реагирования на прерывание PTS

6.4 Специальные прерывания

МК поддерживают три специальных прерывания: по неподдерживаемым кодам, пошагового исполнения и NMI. Эти прерывания не используют бит разрешения прерывания (I) в PSW (PSW.1), и они не могут быть замаскированы. Все эти прерывания обслуживаются контроллером прерывания; они не могут быть назначены на PTS. Из них только NMI проходят через детектор передачи и декодировщик приоритетов. Остальные два специальных прерывания проходят напрямую в контроллер прерывания на обработку. Следует быть осторожными, так как данные прерывания часто назначаются на специальные функции в современных устройствах отладки.

6.4.1 Неподдерживаемые коды

Если ЦПУ попытается выполнить неподдерживаемый код, то произойдёт переход по вектору 2012h. Это предотвращает их случайное программное исполнение в случае аппаратного или программного сбоя. Вектор прерывания должен обязательно содержать стартовый адрес программы обработки ошибки, чтобы не усугубить и без того ошибочную ситуацию. Прерывание по неподдерживаемым кодам запрещает подтверждение остальных прерываний, пока не выполнится следующая команда.

6.4.2 Программное прерывание

Команда TRAP (код 0F7h) вызывает прерывание по вектору, расположенному в ячейке 2010h. Команда TRAP позволяет выработать прерывание после каждой команды. Это полезно при отладке программы или генерации программных прерываний. Команда TRAP запрещает подтверждение остальных прерываний на время выполнения следующей команды.

6.4.3 Немаскируемое прерывание (NMI)

Внешний вывод NMI генерирует немаскируемое прерывание для выполнения экстренных программ прерывания. NMI имеет высший приоритет над всеми прерываниями с приоритетом. Оно передается напрямую от детектора передачи к декодировщику приоритетов и вызывается косвенно через ячейку 203Eh. Прерывание NMI опознается во время Phase2 (высокий уровень CLKOUT) и запоминается. Если на этом входе поддерживается высокий уровень, то другие прерывания блокируются. Если ваша система не использует NMI - прерывание, необходимо заземлить вывод NMI для предотвращения неожиданных прерываний.

По аналогии с регистром INT_PEND1 в регистре INT_MASK1 существует бит маски NMI. Однако этот бит не используется; NMI разрешён как при установленном, так и при сброшенном NMI_MASK. Для обеспечения совместимости с будущими версиями контроллера всегда следует записывать ноль в бит маски NMI. В 8096BH программа

обработки NMI располагается по адресу 0000h. Для совместимости программ, использующих NMI, с программами для 8096BH необходимо загрузить 0000h в ячейку 203Eh.

Таблица 6.4 – Прерывание, управление PTS и регистры статуса

Мнемоника Регистра	Имя Регистра	Описание
INT_MASK INT_MASK1	Регистры маски прерывания	Данные регистры разрешают/запрещают все маскируемые прерывания (то есть все прерывания, кроме прерываний по неподдерживаемым кодам, пошагового исполнения и NMI).
INT_PEND INT_PEND1	Регистры ожидания прерывания	Биты этого регистра устанавливаются аппаратно для индикации ожидания прерываний.
IOC1	Регистр управления вводом-выводом	Данный регистр указывает источник для прерываний INT00, INT02 и INT07.
IOS1	Регистр статуса ввода-вывода	Этот регистр содержит флаги, индицирующие какие события вызвали прерывания.
PSW	Слово состояния программы	Этот регистр содержит один бит, который глобально разрешает или запрещает обработку всех маскируемых прерываний, а также бит, который разрешает или запрещает PTS.
PTSSEL	Регистр выбора PTS	Это регистр выбирает либо PTS цикл, либо программу стандартной обработки прерываний для каждого из 15-ти маскируемых запросов прерываний.
PTSSRV	Регистр обработки PTS	Биты в этом регистре устанавливаются аппаратно по запросу.

6.5 Программирование прерываний

В таблице 6.4 показаны программируемые регистры, которые воздействуют на работу и функции контроллера прерываний и PTS.

6.5.1 Выбор способа обработки прерываний: через PTS или стандартной процедуры обработки прерываний

Через регистр выбора PTS (PTSSEL) выбирается либо цикл PTS, либо программа стандартной обработки прерываний для каждого из 15-ти маскируемых запросов прерывания. Установкой бита выбирается цикл PTS; обнулением бита выбирается программа стандартной обработки прерываний.

6.5.2 Разрешение PTS прерываний

Если прерывания назначены на подпрограммы стандартной программной обработки, нужно разрешить как обработку прерываний, так и каждое прерывание в отдельности. Бит глобального разрешения прерываний (I) в PSW (PSW.1) глобально разрешает или запрещает обработку всех маскируемых прерываний. Команда EI устанавливает этот бит, который разрешает обработку прерываний. Команда DI очищает бит, чем запрещает обработку прерываний. Биты в INT_MASK и INT_MASK1 индивидуально разрешают или запрещают прерывания (смотрите таблицу 6.5). Прерывания, которые имеют место, когда обработка прерываний глобально запрещена (PSW.1 очищен) сохраняются в регистрах ожидания прерывания.

Таблица 6.5 – Источники стандартных прерываний, векторы и биты регистра

Источник прерывания	Вектор прерывания	Номер	Бит разрешения прерывания[1)]	Условия выбора источника[2)]
Завершение A/D преобразования	A/D Convertion Complete	INT01	INT_MASK.1	–
Запуск A/D преобразования	Software Timer	INT05	INT_MASK.5	–
Четвёртая запись в FIFO HSI	HSI FIFO 4	INT10	INT_MASK1.2	–
Заполнение FIFO HSI	HSI FIFO Full	INT14	INT_MASK1.6	–
Заполнение FIFO HSI	HSI Data Available	INT02	INT_MASK.2	IOC1.7 = 1
Регистр удержания HSI загружен	HSI Data Available	INT02	INT_MASK.2	IOC1.7 = 0
HSI.0	Вывод HSI.0	INT04	INT_MASK.4	–
HSO.0 - HSO.5	HSO	INT03	INT_MASK.3	–
NMI	NMI	INT15	–	–
P0.7	EXTINT	INT07	INT_MASK.7	IOC1.1=1
P2.2	EXTINT	INT07	INT_MASK.7	IOC1.1=0
P2.2	EXTINT1	INT13	INT_MASK1.5	–
RI	Receive	INT09	INT_MASK1.1	–
RI	Serial Port	INT06	INT_MASK.6	–
Программные таймеры 0 – 3	Software Timer	INT05	INT_MASK.5	–
TI	Transmit	INT08	INT_MASK1.0	–
TI	Serial Port	INT06	INT_MASK.6	–
Переполнение таймера 1	Timer Overflow	INT00	INT_MASK.0	IOC1.2=1
Захват значения таймера 2	Timer 2 Capture	INT11	INT_MASK1.3	–
Переполнение таймера 2	Timer Overflow	INT00	INT_MASK.0	IOC1.3=1
Переполнение таймера 2	Timer 2 Overflow	INT12	INT_MASK1.4	–
Сброс таймера 2	Software Timer	INT05	INT_MASK.5	–

[1)] В этой колонке перечисляются для каждого из источников прерываний биты маски, которые необходимо установить для разрешения прерываний. Пять источников прерываний могут генерировать два отдельных прерывания - прерывания, совместимые с 8096BH, и новые, специальные прерывания (HSI FIFO Full, EXTINT1, Receive, Transmit, Timer 2 Overflow). Во всех случаях только одно прерывание может быть разрешено для каждого источника. (Для этого бит маски должен быть установлен только для одного из двух возможных прерываний.)

[2)] Три из совместимых с 8096BH прерывания (HSI Data Available, EXTINT и Timer Overflow) могут генерироваться любым из двух источников. В колонке приведены биты регистра IOC1 и их значения, которые выбирают соответствующий источник.

6.5.3 Выбор источников прерываний

Пять источников прерываний могут генерировать два отдельных прерывания - прерывания, совместимые с 8096BH, и новые, специальные прерывания (HSI FIFO Full, EXTINT1, Receive, Transmit, Timer 2 Overflow). Во всех случаях только одно прерывание может быть разрешено для каждого источника (для этого бит маски должен быть установлен только для одного из двух возможных прерываний). Рисунок 6.5 показывает источники прерывания для каждого вектора прерывания. В таблице 6.5 приведены биты регистра IOC1 и их значения, которые выбирают соответствующий источник.

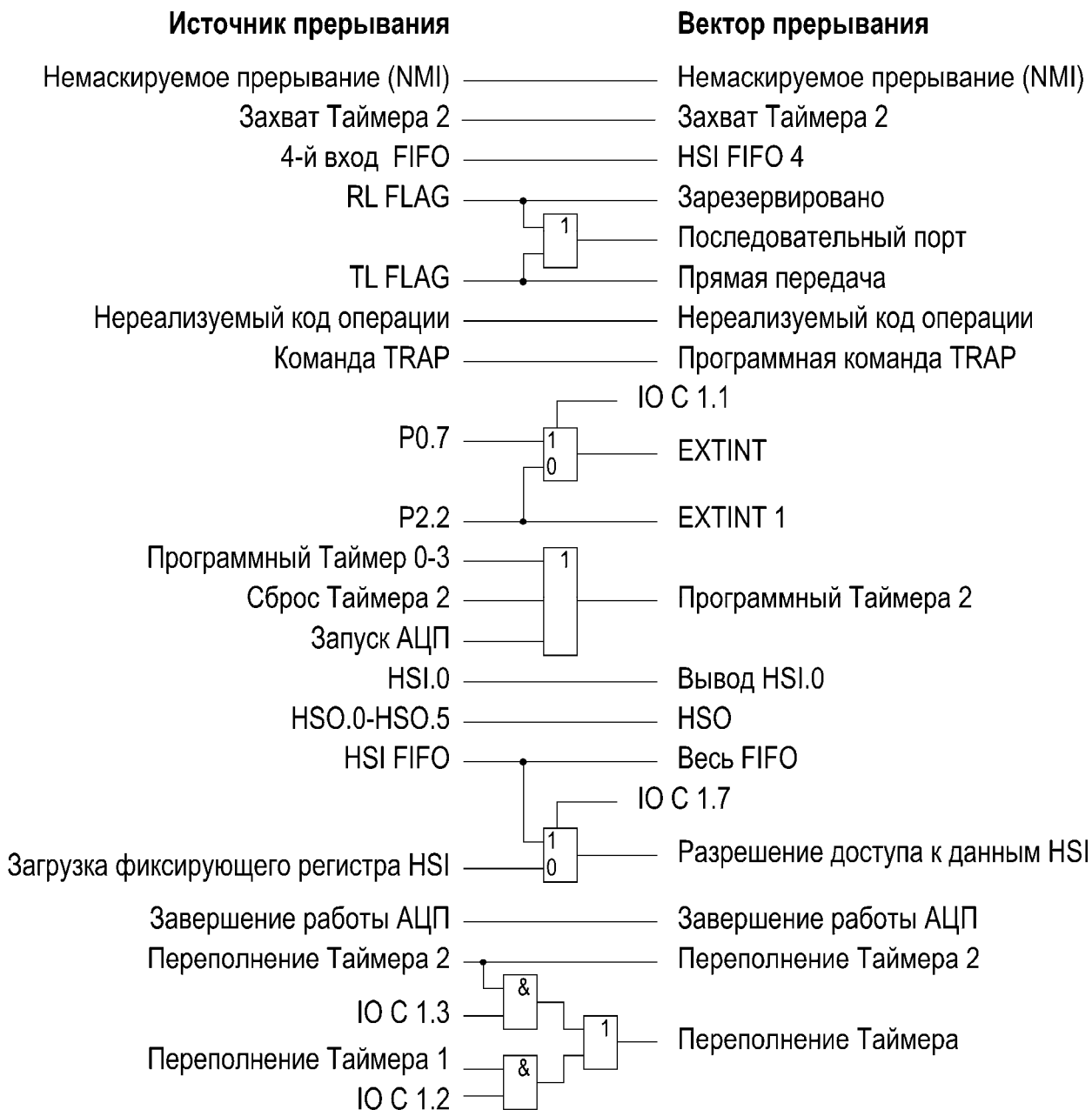


Рисунок 6.5 – Источники прерываний

6.5.4 Регистры масок прерываний

Регистры масок прерываний INT_MASK и INT_MASK1 разрешают или запрещают (маскируют) каждое из прерываний. За исключением бита Немаскируемого Прерывания (NMI) (INT_MASK1.7), установка бита разрешает соответствующий источник прерываний; очистка бита запрещает источник. Если устройство сбрасывается, то регистры масок прерываний очищаются (прерывания запрещены).

Для достижения соответствия с регистром INT_PEND1 в регистре INT_MASK1 также содержится бит маски NMI. Однако этот бит маски не используется; NMI разрешено как при установленном, так и при сброшенном INT_MASK. Для обеспечения совместимости с будущими версиями контроллера всегда следует записывать ноль в бит маски NMI.

6.5.5 Регистры ожидания прерываний

Когда Детектор Запросов обнаруживает прерывание, он устанавливает соответствующий бит в регистре INT_PEND или INT_PEND1. Этот бит устанавливается даже для запрещённых (замаскированных) прерываний. Бит ожидания прерывания

очищается, когда программа переходит на подпрограмму обработки прерывания (стандартные прерывания) или на PTSCB (PTS прерывания). INT_PEND и INT_PEND1 могут быть считаны для определения ожидающих прерываний. Они также могут быть модифицированы (записаны) либо для снятия ожидания прерываний, либо для генерации прерываний программным обеспечением.

Необходимо быть внимательным при написании части программы, которая модифицирует данные регистры. Например, последовательность команд, которая очищает бит ожидания, может привести к тому, что прерывание будет подтверждено после начала выполнения этой последовательности, но до того как бит действительно будет очищен. В этом случае имеют место пять тактов, разделяющих цикл прерывания. То есть подготовка прерывания начнётся, но не будет перехода на программу обработки прерывания. Данную задержку можно избежать, делая эту часть программы неделимой, в том смысле, что прерывание не может быть подтверждено, пока она выполняется. Простейшим случаем является использование логических команд двух- или трёхоперандного формата, например:

```
ANDB INT_PEND, #01111111B ; Clears the EXTINT interrupt
ORB INT_PEND, #10000000B  ; Sets the EXTINT interrupt
```

МК не подтверждает прерывания, пока выполняются команды “считывание-модификация-запись”.

Регистр PTSSEL содержит запросы end-of-PTS прерываний. End-of-PTS прерывания показывают, что PTS требует обслуживания. Программа обработки end-of-PTS прерывания обновляет PTSCB и устанавливает соответствующий бит, разрешая обработку PTS прерываний.

6.6 Блоки управления PTS

Каждое PTS-прерывание запрашивает блок данных, называемых Блоком Управления PTS (PTSCB). PTSCB определяет режим PTS, число циклов PTS и адреса источника и приёмника передаваемых данных. Необходимо установить PTSCB для каждого источника прерывания, прежде чем будут разрешены прерывания PTS. Каждый PTSCB требует восемь байт данных в регистровом ОЗУ. Адрес первого (младшего) байта хранится в таблице векторов PTS в специально отведённой памяти. Первый байт записывается по адресу, кратному восьми (граница по четвертому слову). На рисунке 6.6 представлены PTSCB для каждого режима PTS. Неиспользуемые байты PTSCB могут использоваться как дополнительное ОЗУ.

Номер байта	Одиночная передача	Блочная передача	Режим АЦП-сканирования	Режим HSO	Режим HSI
8	Не используется	Не используется	Не используется	Не используется	Не используется
7	Не используется	PTSBLOCK	Не используется	PTSBLOCK	PTSBLOCK
6	PTSDST (HI)	PTSDST (HI)	PTSREG (HI)	Не используется	Не используется
5	PTSDST (LO)	PTSDST (LO)	PTSREG (LO)	Не используется	Не используется
4	PTSSRC (HI)	PTSSRC (HI)	PTS_S/D (HI)	PTSSRC (HI)	PTSDST (HI)
3	PTSSRC (LO)	PTSSRC (LO)	PTS_S/D (LO)	PTSSRC (LO)	PTSDST (LO)
2	PTSCON	PTSCON	PTSCON	PTSCON	PTSCON
1	PTSCOUNT	PTSCOUNT	PTSCOUNT	PTSCOUNT	PTSCOUNT

Рисунок 6.6 – Блок управления PTS

6.6.1 Регистр PTSCOUNT

Первым байтом каждого PTSCB всегда является регистр PTSCOUNT. PTSCOUNT определяет число циклов PTS, выполняемых без участия программного обеспечения. Так как PTSCOUNT имеет восьмибитное значение, максимальное количество циклов - 256. PTSCOUNT уменьшается на единицу в конце каждого цикла PTS. Когда PTSCOUNT становится равным нулю, аппаратно очищается соответствующий бит PTSSEL и устанавливается бит PTSSRV, который запрашивает end-of-PTS прерывание.

6.6.1.1 End-of-PTS прерывание

End-of-PTS прерывание является стандартным. Контроллер прерываний обслуживает его с помощью программы обработки прерывания, находящейся в памяти, расположение которой указано стандартным вектором прерывания. Например, PTS обрабатывает прерывание Transmit, если установлен PTSSEL.8. Вектор PTS 2050h, а соответствующий вектор end-of-PTS прерывания 2030h – стандартный вектор прерывания Transmit. Когда передано управление программе обработки прерывания end-of-PTS, аппаратно очищается бит PTSSRV. Программа обработки прерывания должна установить бит PTSSEL, разрешающий обработку PTS прерывания.

6.6.2 Регистр PTSCON

Вторым байтом каждого PTSCB всегда является регистр PTSCON. Три бита регистра PTSCON определяют режим PTS: Одиночная Передача, Блочная Передача, Сканирование АЦП, HSO или HSI (смотрите таблицу 6.6). Режим PTS определяет функцию остальных битов; смотрите таблицу 6.7 для режимов Одиночной и Блочной Передачи. PTSCON имеет одну конфигурацию для режимов Одиночной и Блочной Передачи (смотрите рисунок 6.7) и другую для режимов Сканирование АЦП, HSO или HSI (смотрите рисунок 6.8). Таблица A.11 приложения A содержит время выполнения цикла для каждого режима PTS.

Таблица 6.6 – Выбор режимов PTS

Бит 7	Бит 6	Бит 5	Выбираемый режим
0	0	0	Одиночная передача
0	0	1	HSI
0	1	0	Не используется
0	1	1	HSO
1	0	0	Блочная передача
1	0	1	Не используется
1	1	0	Сканирование АЦП
1	1	1	Не используется

Таблица 6.7 – Биты 0-4 PTSCON (режимы одиночной и блочной передачи)

Номер бита	Мнемоника бита	Имя бита	Описание
0	DI	Автоинкремент PTSDST	При установке этого бита PTS регистр назначения инкрементируется в конце каждого PTS цикла.
1	SI	Автоинкремент PTSSRC	При установке этого бита PTS регистр источника инкрементируется в конце каждого PTS цикла.
2	DU	Обновление PTSDST	При установке этого бита PTSDST сохраняет конечное значение в конце PTS цикла. При отсутствии установки регистр возвращается к значению, которое было в начале PTS цикла.
3	SU	Обновление PTSSRC	При установке этого бита PTSSRC сохраняет конечное значение в конце PTS цикла. При отсутствии установки регистр возвращается к значению, которое было в начале PTS цикла.
4	BW	Передача байт/ слово	Установка этого бита указывает на передачу байта, отсутствие - на передачу слова.

Таблица 6.8 – Бит 3 PTSCON (Режимы сканирования АЦП, HSI и HSO)

Номер бита	Мнемоника бита	Имя бита	Описание
3	UPDT	Обновление регистра	При установке этого бита соответствующий регистр будет загружен значением, которое будет в конце PTS цикла. При очистке этого бита регистр будет загружен значением, которое было в начале PTS цикла. Режим Регистр A/D PTS_S/D HSI PTSDST HSO PTSSRC

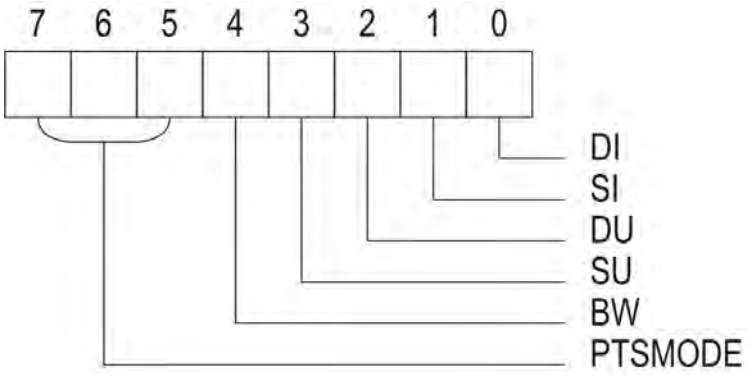


Рисунок 6.7 – PTSCON (Режимы одиночной и блочной передачи)

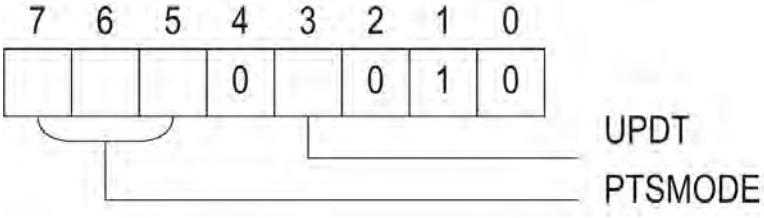


Рисунок 6.8 – PTSCON (Режимы сканирования АЦП, HSI и HSO)

6.7 Режим одиночной передачи

В режиме Одиночной передачи каждый PTS цикл передает один байт или слово (установка BW бита в PTSCON) из одной ячейки памяти в другую. Этот режим обычно используется с прерываниями по последовательному порту ввода-вывода. PTSCOUNT регистр указывает количество передач (каждая передача – один PTS цикл). PTS пересылает байт или слово из ячейки, указанной регистром источником (PTSSRC), в ячейку, указанную регистром назначения (PTSDST).

PTSSRC и PTSDST могу указывать любую ячейку памяти; тем не менее, они должны указывать четный адрес, если выбрана передача слова. При установке битов автоинкремента и обновления PTS увеличивает адрес источника (если установлены биты SI и SU) и/ или адрес назначения (если установлены биты DI и DU) в конце каждого PTS цикла. Адреса увеличиваются на 1, если установлен режим передачи байта, и на 2, если установлен режим передачи слова. В режиме Одиночной передачи каждая пара битов автоинкремента и обновления должна быть либо установлена, либо очищена. Программирование битов автоинкремента и обновления (0, 1) или (1, 0) – ошибка. PTSSRC и PTSDST могут быть увеличены (и обновлены) независимо друг от друга.

6.7.1 Пример режима одиночной передачи

Следующий PTSCB определяет девять PTS циклов (смотрите таблицу 6.9). Каждый цикл пересылает одно слово с адреса 20h во внешнюю память. PTS передает первое слово по адресу 6000h, затем он увеличивает и обновляет адрес назначения и декрементирует PTSCOUNT регистр; он не увеличивает адрес источника. В начале второго цикла PTS пересылает второе слово с адреса 20h в адрес 6002h. Когда PTSCOUNT равен нулю, PTS заполнит область 6000h - 600Fh и сформирует прерывание end-of-PTS.

Таблица 6.9 – PTSCB в режиме одиночной передачи

не используется
не используется
PTSDST(HI)=60h
PTSDST(LO)=00h
PTSSRC(HI)=00h
PTSSRC(LO)=20h
PTSCON=15h (DI, DU, & BW=1)
PTSCOUNT=09h

6.8 Режим Блочной передачи

В режиме Блочной передачи PTS пересылает блок данных из одних ячеек памяти в другие. PTSBLOCK регистр указывает количество байтов или слов в каждом блоке (n=1÷32). PTS пересылает блок байтов или слов из ячейки, указанной регистром источника (PTSSRC), в ячейку, указанную регистром назначения (PTSDST).

PTSSRC и PTSDST могут указывать на любую ячейку в памяти, тем не менее они должны указывать на четный адрес, если выбрана передача слова. При установке битов автоинкремента в регистре PTSCON PTS увеличивает адрес источника (установлен SI) и/или адрес назначения (установлен DI) в конце каждой PTS передачи. Если бит обновления также установлен, увеличенный адрес записывается в PTSSRC (установлен SU) или в PTSDST (установлен DU) после каждого PTS цикла. Установка обоих битов инкремента и обновления означает, что адрес источника и/или адрес назначения будут инкрементироваться после каждого PTS цикла. Регистры увеличиваются на 1, если выбрана передача байта, или на 2, если выбрана передача слова. Увеличение и обновление могут быть выбраны независимо (не так, как в режиме одиночной передачи).

В этом режиме важно различать PTS передачу и PTS цикл. PTS передача – это пересылка одного байта или слова от источника к месту назначения. PTS цикл состоит из передачи полного блока байтов или слов. Так как PTS цикл непрерываемый, режим Блочной передачи может вызвать большую задержку реакции на прерывание. В худшем случае длительность может быть 500 тактов. Эта худшая длительность допускает передачу блока из 32 слов из одной области внешней памяти в другую, используя 8-битовую шину без режима ожидания. Смотри таблицу A.11 в приложении A для определения времени выполнения PTS цикла.

6.8.1 Пример режима блочной передачи

PTSCB в таблице 6.10 определяет три PTS цикла, каждый из которых передает байты из области памяти 20h - 24h в один из следующих блоков: 6000h - 6004h, 6005h - 6009h или 600Ah - 600Eh. Каждый PTS цикл требует появления пяти пересылок. Источник и место назначения инкрементируются после каждой передачи, но только место назначения обновляется после каждого цикла. Первый байт в каждом цикле всегда читается из ячейки 20h.

Таблица 6.10 – Режим PTSCB Block Transfer

Не используется
PTSBLOCK = 05h
PTSDST(HI) = 60h
PTSDST(LO) = 00h
PTSSRC(HI) = 00h
PTSSRC(LO) = 20h
PTSCON = 97h (DI, SI, DU, BW = 1)
PTSCOUNT = 03h

6.9 Режим сканирования АЦП

Данный подраздел относится только к изделиям 1874BE76T, 1874BE06T. В микросхеме 1874BE05T АЦП отсутствует. В режиме сканирования АЦП, PTS заставляет АЦП совершать многократные преобразования одного или более каналов и затем запоминает результат. Для использования режима Сканирования АЦП необходимо в первую очередь установить в памяти таблицу команда/данные (смотрите таблицу 6.11). Таблица команда/данные содержит команды для АЦП, которые чередуются с пустыми ячейками памяти. PTS запоминает результаты преобразований в этих пустых ячейках.

Для инициализации режима Сканирования АЦП необходимо: разрешить прерывание по законченному А/Ц преобразованию и назначить его на PTS, затем программно начать первое преобразование. Когда АЦП закончит первое преобразование и выработает прерывание по законченному А/Ц преобразованию, начнется PTS цикл.

Во время каждого PTS цикла PTS запоминает результат предыдущего преобразования и затем выполняет команду следующего преобразования. Так как результат преобразования не запоминается до следующего PTS цикла, последняя командная ячейка должна содержать все нули для предотвращения начала последнего преобразования. Обычно команды для АЦП загружаются в таблицу из внешней ПЗУ. Только количество доступной памяти ограничивает размер таблицы; она может находиться во внутренней или внешней ОЗУ.

Таблица 6.11 – Таблица команда/данные режима сканирования АЦП

XXX+0Ah	Результат A/D преобразования 2	
XXX+8h	не используется	Команда A/D 3
XXX+6h	Результат A/D преобразования 1	
XXX+4h	не используется	Команда A/D 2
XXX+2h	Результат A/D преобразования 0 *	
XXX	не используется	Команда A/D 1

_____
* Результат A/D преобразования, которое начинает PTS цикл.

В режиме сканирования АЦП PTSCOUNT указывает общее количество циклов А/Ц преобразования. PTS_S/D регистр указывает на таблицу команд и результатов преобразования. При установке бита UPDT в регистре PTSCON (PTSCON.3) регистр PTS_S/D сохраняет свое последнее значение в конце PTS цикла. При очистке бита регистр возвращается к первоначальному значению PTS цикла. PTS_REG указывает на адрес 02h в HWindow 0. При чтении эта ячейка содержит AD_RESULT регистр; при записи она содержит AD_COMMAND регистр. Режим Сканирования АЦП также использует два временных регистра, которые недоступны пользователю.

6.9.1 PTS циклы в режиме сканирования АЦП

Первое A/D преобразование должно начинаться программно. Прерывание по завершенному A/D преобразованию начинает PTS цикл. После начала PTS цикла выполняются следующие действия:

1 PTS считывает первую команду, запоминает ее в рабочей ячейке и затем дважды инкрементирует PTS_S/D регистр. Теперь PTS_S/D указывает на первую пустую ячейку в таблице команда/данные (адрес XXX+2).

2 PTS считывает AD_RESULT регистр, записывает результат первого преобразования в ячейку XXX+2 таблицы команда/данные и инкрементирует PTS_S/D регистр дважды. Теперь PTS_S/D указывает на ячейку XXX+4.

3 PTS загружает команду из рабочей ячейки в AD_COMMAND регистр. Начинается следующий цикл А/Ц преобразования.

4 Если UPDT (PTSCON.3) очищен, PTS_S/D регистр переустанавливается в его первоначальное значение. Следующий цикл использует ту же команду и перезапишет предыдущие данные. Если UPDT установлен, PTS запишет новое содержимое в PTS_S/D, и он будет указывать на следующую команду.

5 PTSCOUNT декрементируется, и ЦПУ вернется к обычному выполнению программы. Когда PTSCOUNT достигнет нуля, аппаратно очистится соответствующий PTSSEL бит, установится PTSSRV бит, который запросит прерывание end-of-PTS.

Если преобразование начинается по завершению PTS цикла, и АЦП выработает прерывание по завершенному A/D преобразованию, то начнется новый PTS цикл. Повторяются шаги с 1 по 5.

Так как младшие шесть битов AD_RESULT регистра содержат информацию состояния, программа обслуживания прерывания end-of-PTS должна сдвигать результирующие данные шесть раз вправо, оставляя в ячейке памяти только результат преобразования.

6.9.2 Режим сканирования АЦП. Пример 1

Таблица 6.12 (команда/данные) инициализирует серии А/Ц преобразований, начинающихся с канала 7 и заканчивающихся каналом 0. Каждый вход в таблицу - слово (два байта). Таблица 6.13 соответствует PTSCB.

Программа начинает преобразование с канала 7. Прерывание по окончанию А/Ц преобразования инициализирует первый PTS цикл. Первым шагом запоминается

команда 6-го канала в рабочую ячейку и инкрементирует PTS_S/D до 102h. Вторым шагом запоминается результат преобразования канала 7 в ячейку 102h и инкрементируется PTS_S/D до 104h. Третьим шагом загружается команда 6-го канала из рабочей ячейки в AD_COMMAND регистр для начала следующего преобразования. Четвертым шагом обновляется PTS_S/D (PTS_S/D указывает на 104h), и пятым шагом декрементируется PTSCOUNT до 7. Следующий цикл начинается записью команды 5-го канала в рабочую ячейку. Во время восьмого цикла (PTSCOUNT=1) холостая команда загружается в AD_COMMAND регистр, и преобразование не выполняется. PTSCOUNT декрементируется до нуля и запрашивается прерывание end-of-PTS.

Таблица 6.12 – Таблица команда/данные (пример 1)

Адрес	Содержимое
11Eh	AD_RESULT для ACH0
11Ch	0000h (Холостая команда)
11Ah	AD_RESULT для ACH1
118h	AD_COMMAND для ACH0
116h	AD_RESULT для ACH2
114h	AD_COMMAND для ACH1
112h	AD_RESULT для ACH3
110h	AD_COMMAND для ACH2
10Eh	AD_RESULT для ACH4
10Ch	AD_COMMAND для ACH3
10Ah	AD_RESULT для ACH5
108h	AD_COMMAND для ACH4
106h	AD_RESULT для ACH6
104h	AD_COMMAND для ACH5
102h	AD_RESULT для ACH7
100h	AD_COMMAND для ACH6

Таблица 6.13 – PTSCB в режиме сканирования АЦП (пример 1)

Не используется
Не используется
PTS_REG (HI) = 00h
PTS_REG (LO) = 02h
PTS_S/D (HI) = 01h
PTS_S/D (LO) = 00h
PTSCON = CAh (UPDT = 1)
PTSCOUNT = 0Ah

6.9.3 Режим сканирования АЦП. Пример 2

В таблице 6.14 приведены установки для серии из десяти PTS циклов, каждый из которых считывает один А/Ц канал и записывает результат в единственную ячейку (102h). UPDT установлен в исходное состояние так, что первоначальное содержимое PTS_S/D восстанавливается после цикла. Таблица команда/данные показана в таблице 6.15.

Таблица 6.14 – PTSCB в режиме сканирования АЦП (пример 2)

Не используется
Не используется
PTS_REG (HI) = 00h
PTS_REG (LO) = 02h
PTS_S/D (HI) = 01h
PTS_S/D (LO) = 00h
PTSCON = C2h (UPDT = 0)
PTSCOUNT = 0Ah

Таблица 6.15 – Таблица команда/данные (пример 2)

Адрес	Содержимое
102h	AD_RESULT для ACH*x*
100h	AD_COMMAND для ACH*x*

Программа начинает преобразование в канале x. Первый PTS цикл начинается, когда преобразование закончено, и сгенерировано прерывание A/D Conversion Complete. PTS записывает результат преобразования в ячейку 102h и затем копирует команду преобразования из ячейки 100h в AD_COMMAND регистр. ЦПУ может обрабатывать или пересылать из таблицы результат преобразования данных перед завершением следующего преобразования и началом нового PTS цикла. Когда начинается следующий цикл, PTS_S/D снова указывает на 100h. Результат преобразования записывается в ячейку 102h, и команда из ячейки 100h выполняется заново.

6.10 Режим HSI

В режиме HSI PTS сбрасывает содержимое HSI FIFO в таблицу, расположенную либо во внутренней либо во внешней памяти. PTSDST регистр содержит адрес таблицы.

Любое HSI прерывание может быть использовано для запуска PTS цикла. PTSBLOCK регистр указывает сколько HSI FIFO блоков (n=1÷7) будет передано в таблицу памяти в течение каждого PTS цикла. Необходимо ввести значение, соответствующее прерыванию, которое генерирует PTS цикл. Например, четвертый вход в FIFO может заставить HSI сгенерировать HSI FIFO 4 прерывание (INT10). Если это прерывание было использовано для запуска PTS цикла, то следует записать 4 в PTSBLOCK регистр, таким образом четыре входа в FIFO будут переписаны в таблицу.

Первым из буфера FIFO читается HSI_STATUS регистр, а затем HSI_TIME регистр. HSI_STATUS регистр содержит биты состояния результата и текущее состояние HSI выводов. HSI_TIME регистр содержит значение таймера 1 в момент HSI события. Смотри раздел 9 для детального знакомства с HSI модулем.

Каждая PTS передача пересылает HSI_STATUS и HSI_TIME регистры в последовательные слова в памяти (смотрите таблицу 6.16). При установке UPDT бита (PTSCON.3) PTSDST регистр сохраняет свое конечное значение в конце PTS цикла.

Таблица 6.16 – Таблица PTS для режима HSI

XXX+0Ah	HSI_TIME_2	
XXX+8h	0FFh	HSI_STATUS_2
XXX+6h	HSI_TIME_1	
XXX+4h	0FFh	HSI_STATUS_1
XXX+2h	HSI_TIME_0	
XXX	0FFh	HSI_STATUS_0

6.10.1 Пример режима HSI

PTSCB в таблице 6.17 определяет десять PTS циклов, каждый из которых передает семь блоков данных HSI_STATUS/HSI_TIME из HSI FIFO в таблицу, начинающуюся с ячейки памяти 100h. Адрес назначения инкрементируется после каждой передачи и обновляется с новым значением после каждого цикла.

Таблица 6.17 – PTSCB в режиме HSI

не используется
PTSBLOCK=07h
не используется
не используется
PTSDST(HI)=01h
PTSDST(LO)=00h
PTSCON=2Ah (UPDT=1)
PTSCOUNT=0Ah

6.11 Режим HSO

В режиме HSO PTS загружает HSO CAM из таблицы, расположенной либо во внутренней, либо во внешней памяти (смотрите таблицу 6.18). Режим HSO аналогичен режиму HSI, за исключением того, что PTS пересылает данные из таблицы в HSO CAM, а не наоборот. PTSSRC регистр содержит адрес таблицы.

Таблица 6.18 – PTS для режима HSO

XXX+0Ah	HSI_TIME_2	
XXX+8h	0FFh	HSI_STATUS_2
XXX+6h	HSI_TIME_1	
XXX+4h	0FFh	HSI_STATUS_1
XXX+2h	HSI_TIME_0	
XXX	0FFh	HSI_STATUS_0

Длина каждого CAM регистра 24 бита. Шестнадцать битов содержат время наступления события по таймеру 1 или таймеру 2. Оставшиеся 8 битов определяют тип события. Загрузка этой информации в таблицу памяти производится в формате, приведенном в таблице 6.18. Каждый PTS-цикл передает данные в регистры HSO_COMMAND и HSO_TIME. Данные передаются из регистров в буферный регистр HSO. Команды задерживаются в этом регистре до тех пор, пока CAM-регистр содержит информацию, в определенное время команды вводятся в CAM.

Любое HSO прерывание может быть использовано для запуска PTS цикла. PTSBLOCK регистр указывает, сколько HSO входов (n=1÷8) будут переданы из таблицы памяти в течение каждого PTS цикла. При установке UPDT бита (PTSCON.3) PTSSRC регистр сохраняет свое конечное значение в конце PTS цикла.

6.11.1 Пример режима HSO

PTSCB в таблице 6.19 определяет десять PTS циклов, каждый из которых передает восемь блоков данных HSO_COMMAND/HSO_TIME в HSO из таблицы, начинающейся с ячейки памяти 100h. Адрес источника инкрементируется после каждой передачи и обновляется после каждого цикла.

Таблица 6.19 – PTSCB в режиме HSO

не используется
PTSBLOCK=08h
не используется
не используется
PTSSRC(HI)=01h
PTSSRC(LO)=00h
PTSCON=6Ah (UPDT=1)
PTSCOUNT=0Ah

7 Порты ввода-вывода

Порты I/O служат для передачи информации между главным устройством и окружающими его системами. Они способны считывать состояние системы, управлять системой, вывести статус устройства, выбрать параметры конфигурации системы, формировать управляющие сигналы, обеспечивать последовательную связь и т. д. Их использование в проекте ограничено только количеством доступных выводов I/O и воображением инженера.

7.1 Обзор функциональных возможностей

Каждый микроконтроллер комплекта имеет пять 8-битных портов ввода-вывода (I/O). Каждый порт имеет внешние выводы, которые выполняют функции входа (ввод информации), выхода (вывод информации), квазидвунаправленного или двунаправленного ввода-вывода с открытым стоком или комбинацию этих функций. Почти все выводы портов I/O имеют дополнительные функции. Например, один из выводов может стать PWM выходом, тогда как другой может использоваться как аналоговый вход.

Каждый вывод порта I/O устанавливается в определенный режим по умолчанию (например, в режим выхода). Однако режим работы вывода может измениться при выборе его дополнительной функции. Например, квазидвунаправленный вывод станет выходом, если выбрать дополнительную функцию. В любом случае, вывод имеет основной набор характеристик, которые связаны с данной структурой. В таблице 7.1 показаны 5 портов I/O, их дополнительные функции и функции по умолчанию. Пятая колонка таблицы показывает, какой SFR определяет, будет ли вывод работать как порт или выполнять альтернативную функцию. Некоторые выводы портов выполняют как функцию порта, так и альтернативную функцию одновременно (например, порт 0 и аналоговые входы). В этом случае пятая колонка таблицы 7.1 пустая.

Дополнительно к стандартным портам I/O, модули HSI и HSO обеспечивают функции ввода-вывода, если не нужны зависящие от времени характеристики этих выводов.

Таблица 7.1 – Функции выводов портов

Выводы Портов	Тип Вывода Порта*	Дополнительные Функции	Тип вывода порта при выбранной дополнительной функции *	Управляющий регистр SFR
P0	INPUT	ACH0−ACH7	INPUT	
P1.0, P1.1, P1.2	QBD	Нет		
P1.3, P1.4	QBD	PWM1, PWM2	OUTPUT	IOC3
P1.5, P1.6	QBD	REQ#, HLDA#	OUTPUT	WSR
P1.7	QBD	HOLD	INPUT	WSR
P2.0	OUTPUT	TXD	OUTPUT	IOC1
P2.1	INPUT	RXD	INPUT OR OUT	SPCON
P2.2	INPUT	EXTINT	INPUT	IOC1
P2.3	INPUT	T2CLK	INPUT	IOC0
P2.4	INPUT	T2RST	INPUT	IOC0
P2.5	OUTPUT	PWM0	OUTPUT	IOC1
P2.6	QBD	T2UP−DN	INPUT	IOC2
P2.7	QBD	T2CAP	QBD	
P3, P4	ODBD	AD0−AD15	ODBD	
* Обозначения типов выводов расшифровываются в следующих разделах				

7.1.1 Входной вывод порта

Любой вывод порта, определенный как вход (INPUT), может быть только считан. Любая операция записи игнорируется (или недопустима). Схема ввода состоит из входного фиксирующего триггера и буфера считывания (смотрите рисунок 7.1). Состояние на входе фиксируется за 1 такт перед разрешением чтения буфера ввода.

Рисунок 7.1 – Входной вывод порта

Захват происходит в течение Фазы 1 (пока CLKOUT имеет низкий уровень), и значение (высокое или низкое) передается на внутреннюю шину. Если состояние на выводе изменяется во время фиксации, то новое значение можно получить, а можно и не получить при считывании. Входные порты не имеют выходных драйверов. Ток утечки на входе при этом очень мал, так же как и емкостная нагрузка.

7.1.2 Выходной вывод порта

Выходной порт (OUTPUT) состоит из защелки порта и логики управления выходом (смотрите рисунок 7.2) Новое значение на выходе появляется во время Фазы 1 (пока CLKOUT имеет низкий уровень). Это значение остается стабильным, пока другая операция записи не изменит его (или не произойдет сброса устройства).

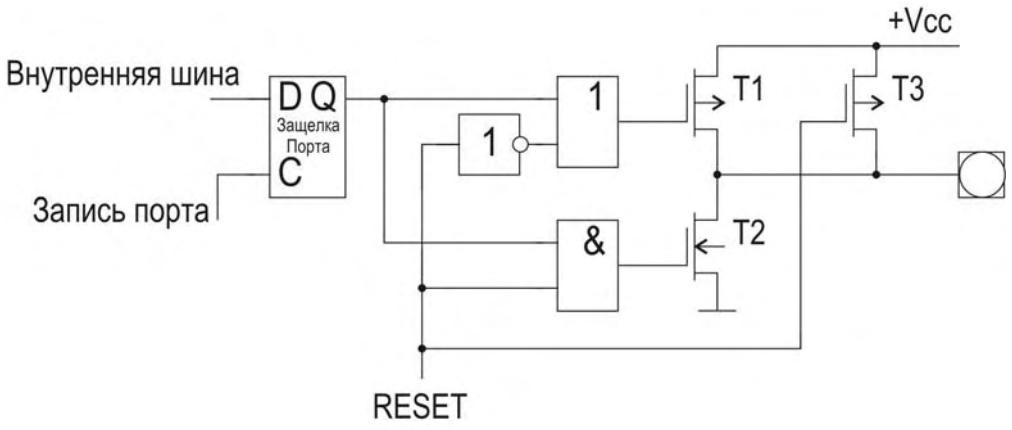


Рисунок 7.2 – Выходной вывод порта

Выходной порт не имеет схемы фиксации и буфера для чтения. При операции считывания этого порта его значение неопределенно и должно быть замаскировано.

Не существует механизма считывания значения с вывода порта, возможна только установка или очистка этого вывода. Если необходимо модифицировать значение на выходе, то это значение должно быть сохранено в памяти. Значение модифицируется в памяти и затем используется для установки или очистки выхода.

7.1.3 Квазидвунаправленный вывод порта

Выводы квазидвунаправленного (QBD) порта могут быть использованы в качестве выводов для ввода и/или вывода (схема управления направлением не нужна). Эти выводы имеют активные низкие и пассивные высокие значения. При высоком уровне на выводе поддерживается функция ввода.

Рисунок 7.3 показывает общую схему вывода QBD порта. Так же, как и схема входного порта, она имеет фиксирующий триггер (защелку) и буфер считывания. Как схема выходного порта, она имеет защелку данных. Структура выходного драйвера делает QBD вывод уникальным.

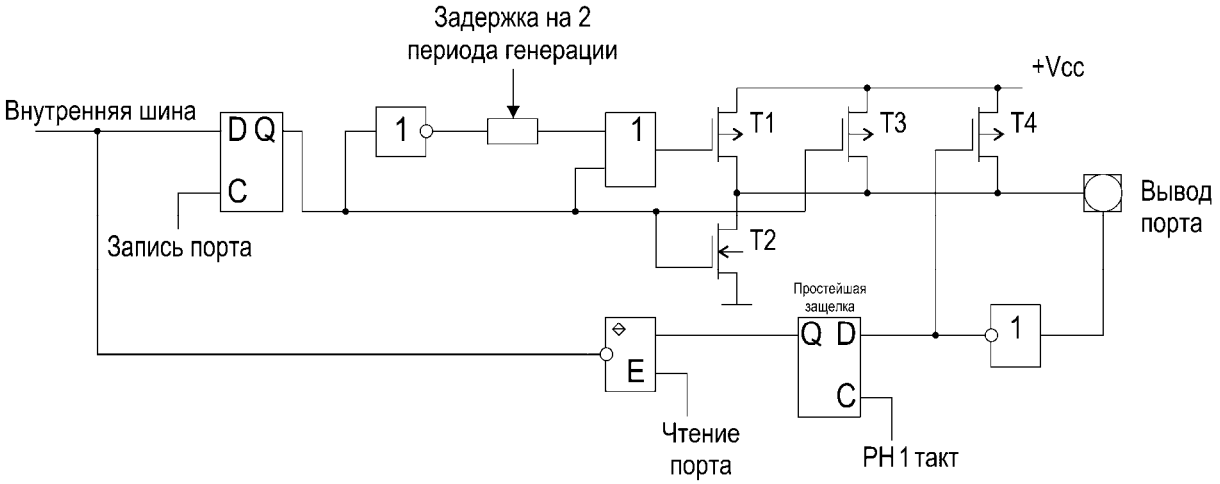


Рисунок 7.3 – Квазидвунаправленный вывод порта

Выход схемы имеет два возможных состояния: низкий уровень, обеспечивающий достаточно большой ток, и слаботочный высокий уровень (если выход переключается с низкого уровня на высокий, то переход осуществляется за более короткое время). Пока на выходе низкий уровень, переключение невозможно (т. е. невозможно обеспечить высокий уровень на выходе внешними средствами без нарушения спецификации устройства). Запись 1 в порт закрывает транзистор Т2 и открывает Т3. Чтобы переключение транзисторов происходило быстрее, на время одного такта сильноточный

транзистор T1 открывается, а затем закрывается, оставляя активным T3. T3 удерживает на выводе уровень логической единицы до тех пор, пока нагрузка на выводе мала.

Вывод QBD порта имеет схему считывания, которая дает возможность считывать выходное значение вывода непосредственно (в отличие от выходного порта). Однако, если выход порта подключается внешними средствами к уровню логического нуля, то чтение этого порта ошибочно покажет, что в него был записан ноль. Квазидвунаправленный вывод порта может быть входом, благодаря возможности отключения выходного транзистора T4. Уменьшение суммарного тока, когда вывод подключен к низкому внешнему уровню, отключает транзистор T4 при достижении значения нуля (низкий уровень, который фиксируется при напряжении около 2 вольт).

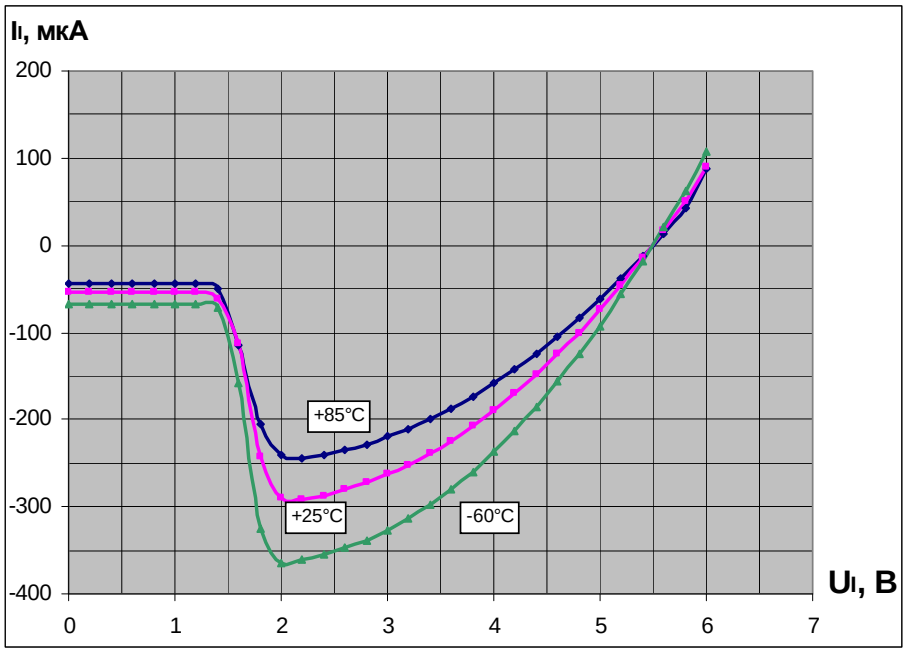


Рисунок 7.4 – Передаточная характеристика квазидвунаправленного выхода

На рисунке 7.4 приведена переходная характеристика порта, когда транзисторы T3 и T4 открыты. Напряжение на выходе уменьшается от U_{CC} до тех пор, пока ток на выходе не достигнет порогового значения I_{TL}. В этой точке T4 закрывается, и остается открытым только T3.

Выходной двунаправленный порт включает в себя транзистор T3, который открывается, когда в защелку порта записывается 1. Этот транзистор T3 поддерживает очень маленький суммарный ток, который создает на неподключенном выводе высокий логический уровень.

Если некоторые выводы порта используются как входные, а некоторые как выходные, то программист должен быть осторожен при записи в порт. Особое внимание нужно уделить при использовании команды XOR или любой другой команды “чтение-модификация-запись”. В двунаправленном порте возможно считать вместо «1» − «0», если внешнее устройство удерживает на входе уровень меньший U_{IH}.

Временные характеристики QBD порта аналогичны соответствующим характеристикам входного и выходного портов. При считывании порта его значение фиксируется во время Фазы 1 (CLKOUT имеет низкий уровень) за один такт до разрешения считывания из буфера чтения. Запись в защелку порта изменяет значение на выходе во время Фазы 1.

7.1.4 Двунаправленный вывод порта с открытым стоком

Работа двунаправленного порта (ODBD) с открытым стоком аналогична работе QBD порта. Отличие в том, что здесь отсутствуют транзисторы T1 и T4 (смотрите

рисунок 7.3). На рисунке 7.5 приведена блок-схема вывода двунаправленного порта с открытым стоком.

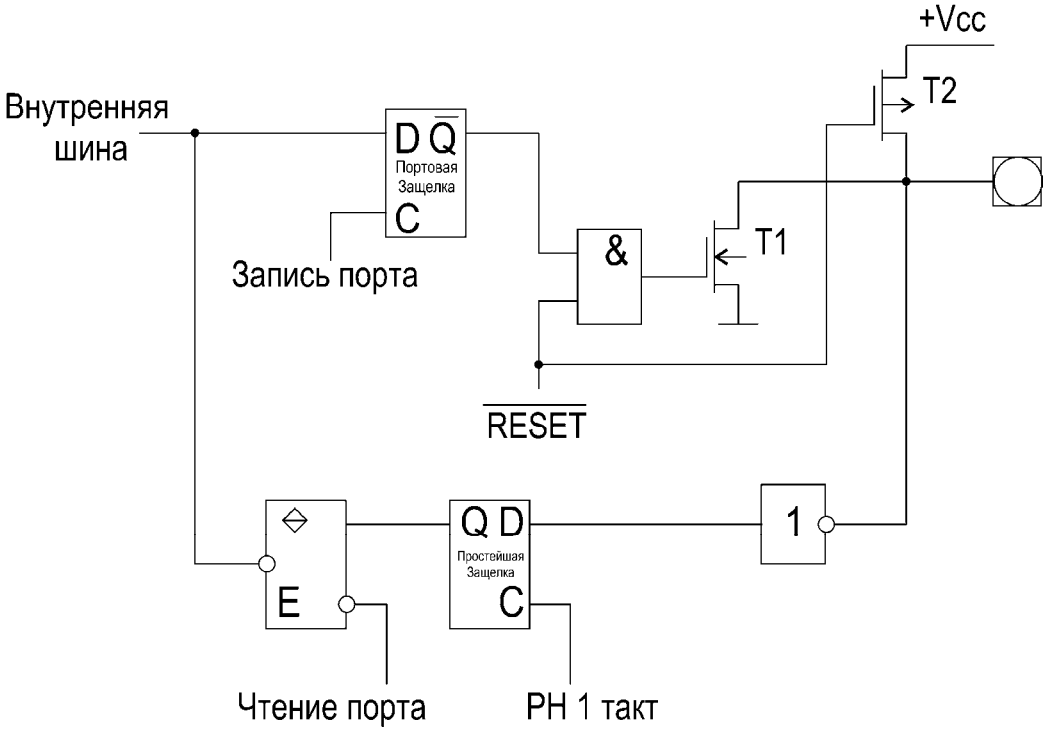


Рисунок 7.5 – Контакт двунаправленного порта с открытым стоком

Запись 0 в защелку порта открывает T1. Пока T1 открыт, его невозможно перевести в состояние высокого логического уровня внешней схемой без нарушения спецификации на устройство. Единица, записанная в защелку порта, закрывает T1 и оставляет вывод неподключенным. Внешним резистором, подсоединенным к U_{CC}, или логической схемой уровень на неподключенном выводе порта можно поднять до уровня логической единицы.

7.2 Программирование портов I/O

Каждый из 5 портов I/O связан с SFR для записи или чтения выводов порта. Некоторые порты I/O имеют дополнительный SFR, который реконфигурирует вывод порта для поддержки альтернативных функций. Не все регистры SFR, связанные с портами, доступны для чтения и записи (Порт 0 - только для чтения). Кроме того, не все SFR находятся в одном и том же горизонтальном окне. (Например, управляющие регистры SFR записываются в HWindow0, а считываются обратно в HWindow15).

IOPORT0, IOPORT1 и IOPORT2 - являются регистрами, используемыми для чтения Порта 0, чтения/записи Порта 1 и чтения/записи Порта 2 соответственно. Эти регистры имеют одинаковый формат, показанный на рисунке 7.6. Таблица 7.2 обобщает операции чтения/записи и условия, которые могут привести к непредвиденному результату.

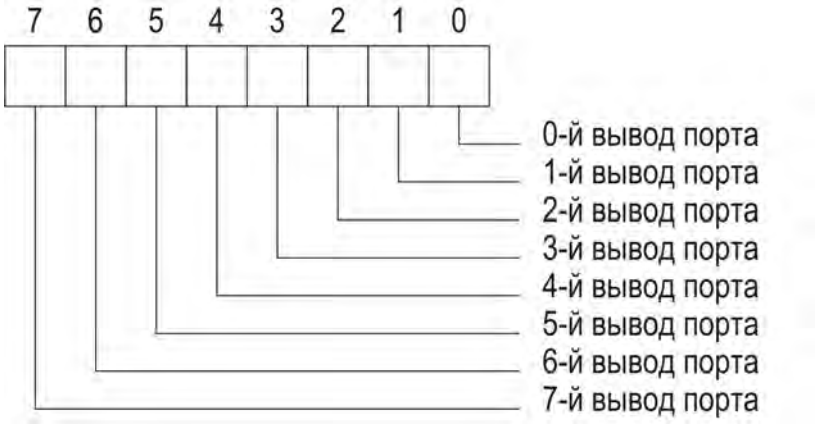


Рисунок 7.6 – Регистр SFR порта ввода-вывода

Таблица 7.2 – Операции чтения /записи порта

Порт	Доступ к Порту	Операция Чтения	Операция Записи
0	IOPORT0	Чтение текущего значения выводов порта. Порт не может считывать, пока идет A/D преобразование.	Невозможна. Запись в IOPORT0 не действует на порт 0, но действует на скорость обращения к контроллеру.
1	IOPORT1	Чтение текущего значения выводов порта. В порт была записана 1, но обратно считывается 0, так как нуль поддерживается внешней схемой.	Запись изменяет состояние защелки порта. Изменение состояния защелки не действует на вывод порта, если выбрана его дополнительная функция.
2	IOPORT2	Чтение текущего значения выводов порта, исключение составляют выходные выводы. Эти выводы возвращают неопределенное значение, и оно должно быть замаскировано программно.	Запись изменяет состояние защелки порта, исключение составляют входные выводы. Они не имеют защелки, поэтому записанное значение игнорируется.
3,4	1FFEh	Чтение текущего значения выводов порта. В порт была записана 1, обратно считывается 0, так как нуль задается извне. Порт 3 и 4 всегда доступны вместе как слово в операции чтения.	Изменение состояния защелки. Любое внешнее обращение контроллера к шине игнорирует значение защелки, при выполнении команды шины. Порты 3 и 4 всегда доступны вместе как слово операции записи.

После сброса все порты I/O, исключая порты 3 и 4, выполняют функции портов I/O. Когда EA# имеет низкий уровень, во время переднего фронта RESET# порты 3 и 4 автоматически переключаются на поддержку системной шины адрес/данные и не могут быть использованы как порты I/O, пока они не будут реконфигурированы. Все или часть выводов портов 0, 1 и 2 обычно конфигурируются на выполнение альтернативных функций.

Использование SFR для конфигурации портов в зависимости от того, какой вывод порта I/O реконфигурируется в этом разделе не рассматривается. Вместо этого, раздел описывает операции, которые могут выполняться портами I/O, и содержит информацию, необходимую для реконфигурации портов. Смотрите таблицу 7.1, чтобы определить, какие SFR управляют дополнительными функциями портов I/O.

7.2.1 Входной порт

При чтении порта I/O сначала фиксируется, а затем запоминается внутри устройства значение каждого вывода порта. Поэтому при каждом считывании порта фиксируется новое значение, и невозможно получать то же значение порта, считывая его каждый раз. Если значение порта необходимо многократно использовать, то применяют рабочие ячейки памяти для хранения истинного значения. Например, следующая операция считывает текущее значение Порта 1 и запоминает его в ячейке памяти, обозначенной P1TEMP:

LDB P1TEMP, IOPORT1 ; чтение P1 и сохранение в P1TEMP

Определенные выводы Порта 2 сконфигурированы как выходы и не содержат схем ввода. После считывания порта в любую ячейку биты SFR, содержащиеся в этой ячейке и являющиеся только выходами, должны быть замаскированы до того, как будет использовано входное значение.

Чтобы использовать вывод QBD порта как входной, необходимо убедиться в том, что транзистор, обеспечивающий мощный низкий уровень на выходе, закрыт. Запись 1 в порт закрывает этот транзистор. Например, команда, приведенная ниже, конфигурирует младшую половину Порта 1 для использования в качестве входов:

LDB IOPORT1, #00001111B; Запись 1 в порт для использования этих разрядов как входов.

7.2.1.1 Описание Порта 0

Особенность Порта 0 в том, что его выводы в одно и то же время могут использоваться как цифровые, так и аналоговые входы (для АЦ-преобразования). Вывод Порта 0, использующийся как цифровой вход, имеет высокий импеданс. Однако выводам Порта 0, использующимся как аналоговые входы, требуется на короткое время обеспечить достаточный ток, чтобы зарядить внутреннюю эталонную емкость. Это значит, что входные характеристики вывода моментально изменятся, если на этом выводе произошло A/D-преобразование. В любом случае, если Порт 0 используется как аналоговый и как цифровой I/O, необходимо подать питание на этот порт через выводы ∩VCC2 и ∩0V.

АЦ-преобразователь чувствителен к различного рода помехам. Настоятельно рекомендуем не считывать состояния порта, пока осуществляется АЦ-преобразование. Порт 0 был спроектирован так, чтобы фиксирующий триггер не стробировался, пока не окончено считывание.

7.2.2 Порт вывода

Запись в порт I/O устанавливает или очищает связанную с ним защелку порта. Выходное значение защелки порта затем используется выходным драйвером. Операция записи относится только к выходному или к двунаправленному (квазидвунаправленному или с открытым стоком) выводам. Запись в порт, работающий только на ввод, не выполняет никаких действий (т. е. защелка не устанавливается и не очищается).

Существует множество способов записи в порт I/O. Порт может быть записан непосредственно с использованием следующих команд:

```
LDB IOPORT1, #10000001B      ; установка 0 и 7 битов, очистка 1-6
LDB intrega, #81h            ; запись значения
STB intrega, IOPORT1         ; вывод значения в порт
```

Обычно непосредственно в порт записывают только для инициализации, или когда все выводы порта необходимо изменить одновременно. Чаще только один вывод необходимо установить, очистить или модифицировать. Чтобы выполнить однобитовые команды, используются команды “чтение-модификация-запись” с текущим значением порта, и выглядит это следующим образом:

```
LDB AL, IOPORT1          ; чтение текущего значения порта
ORB AL, #10000001B       ; установка 0, 7 битов без действующих 1-6 битов
LDB IOPORT1, AL          ; запись нового значения порта
```

Эта операция может быть выполнена одной командой:

```
ORB IOPORT1, #10000001B  ; чтение IOPORT1, его модификация и обратная запись в ioport1
```

Следующая операция может быть использована, чтобы инвертировать значение вывода порта:

```
XORB IOPORT1, #1000000B ; дополнение P1.7
```

Во всех приведенных выше операциях значение порта считывается, модифицируется и записывается обратно. Однако есть случаи, в которых приведенные выше операции не дают желаемого результата. Например, Порт 2 имеет 2 вывода только на выход. Выходные выводы не имеют считывающего буфера, и поэтому текущее значение порта нельзя считать. Ясно, что операция “чтение-модификация-запись” ничего не изменит.

Другой пример – Порт 1, имеющий как входы, так и выходы. Все выводы Порта 1 квазидвунаправленные, любой вывод, использованный как входной, требуют записи единицы. Это становится настоящей проблемой, когда значение порта считывается, модифицируется и записывается обратно. Если на любой вход подается нулевое значение с внешнего устройства, то выполнение операции “чтение-модификация-запись” считает и запишет 0 обратно на входной вывод. Запись 0 на квазидвунаправленный вывод открывает драйвер с мощным низким уровнем запуска, и произойдет ”короткое замыкание”, если единица введется от внешнего устройства.

Решение этой проблемы – отображение значения порта в ячейке памяти. Операция (установка, очистка, переключение) выполняется со значением порта, сохраненным в памяти, которое затем записывается непосредственно в порт.

```
XORB P1IMAGE, #10000000B  ; инверсия отображения P1.7
LDB IOPORT1, P1IMAGE      ; запись нового значения в Порт 1
```

Есть ещё один путь решения проблемы – использование операции OR с единицами для тех двунаправленных выводов, которые являются входами. Важно помнить, что одновременное использование входов и выходов одного и того же порта требует особого внимания при модификации выходного значения.

7.2.3 Доступ к портам 3 и 4

Доступ к этим портам легко осуществляется с помощью внутреннего регистра Порта 3 и Порта 4 (IOPORT34) с адресом 1FFEh. Заметьте, что Порты 3 и 4 доступны только как слово. Невозможно считать или записать каждый порт отдельно. Чтобы использовать Порты 3 и 4 как порты I/O, устройство должно быть установлено на работу с внутренним ПЗУ (смотрите подраздел 7.2.3.1).

LD и ST команды требуют использования внутренних вспомогательных регистров, это необходимо для переноса информации порта в Нижний Регистровый Файл перед использованием данных. Если данные уже во внутренней памяти, то в команде LD нет необходимости. Пример записи значения слова в Порты 3 и 4:

```
LD   intreg, portdata    ; регистр← данные
ST   intreg, 1FFEh       ; регистр → Порт3 и 4
```

Ввод из Портов 3 и 4 требует, чтобы сначала записью в регистры портов была установлена конфигурация портов на ввод. Это переводит порты в высокоимпедансное состояние. Следует заметить, что порты при сбросе находятся в режиме входов. Если в порт были записаны нули, то на выводы, которые должны быть входами, необходимо записать единицы. Чтение Портов 3 и 4, если в них предварительно были записаны нули, идет в следующем порядке:

```
LD      intrega, #0FFFFh      ; установка порта изменяет режим схемы
ST      intrega, 1FFEh        ; регистр → Порты 3 и 4
                              ; ST и LD не
                              ; нужны, если предварительно были
                              ; записаны единицы
LD      intregb, 1FFEh        ; регистр ← Порты 3 и 4
```

Форматы команд LD и ST аналогичны, только изменяется направление передачи и приема.

7.2.3.1 Описание портов 3 и 4

Выводы Портов 3 и 4 имеют 2 функции I/O. Они выполняют функции либо двунаправленного порта с открытым стоком, либо двунаправленной системной шины адрес/данные, либо их комбинации. То, какие функции выполняют Порты 3 и 4, зависит от уровня на выводе EA# во время последнего сброса и от необходимости использования внешней памяти программ и данных.

Вывод EA# не только определяет функцию Портов 3 и 4, но также определяет, доступны ли порты. Если EA# имеет низкий уровень во время системного сброса, то Порты 3 и 4 используются только для поддержки системной шины адрес/данные. Чтение или запись в адрес Портов 3 и 4 (1FFEh) интерпретируется как доступ к внешней шине, а не к портам. Если необходимо, Порты 3 и 4 можно “перестроить” в соответствии со схемой на рисунке 7.7. Эта схема использует стандартные защелки и драйверы, чтобы сымитировать функцию I/O с открытым стоком. Доступ к шине для чтения/записи по адресу 1FFEh должен быть декодирован, чтобы разрешить функционирование защелок.

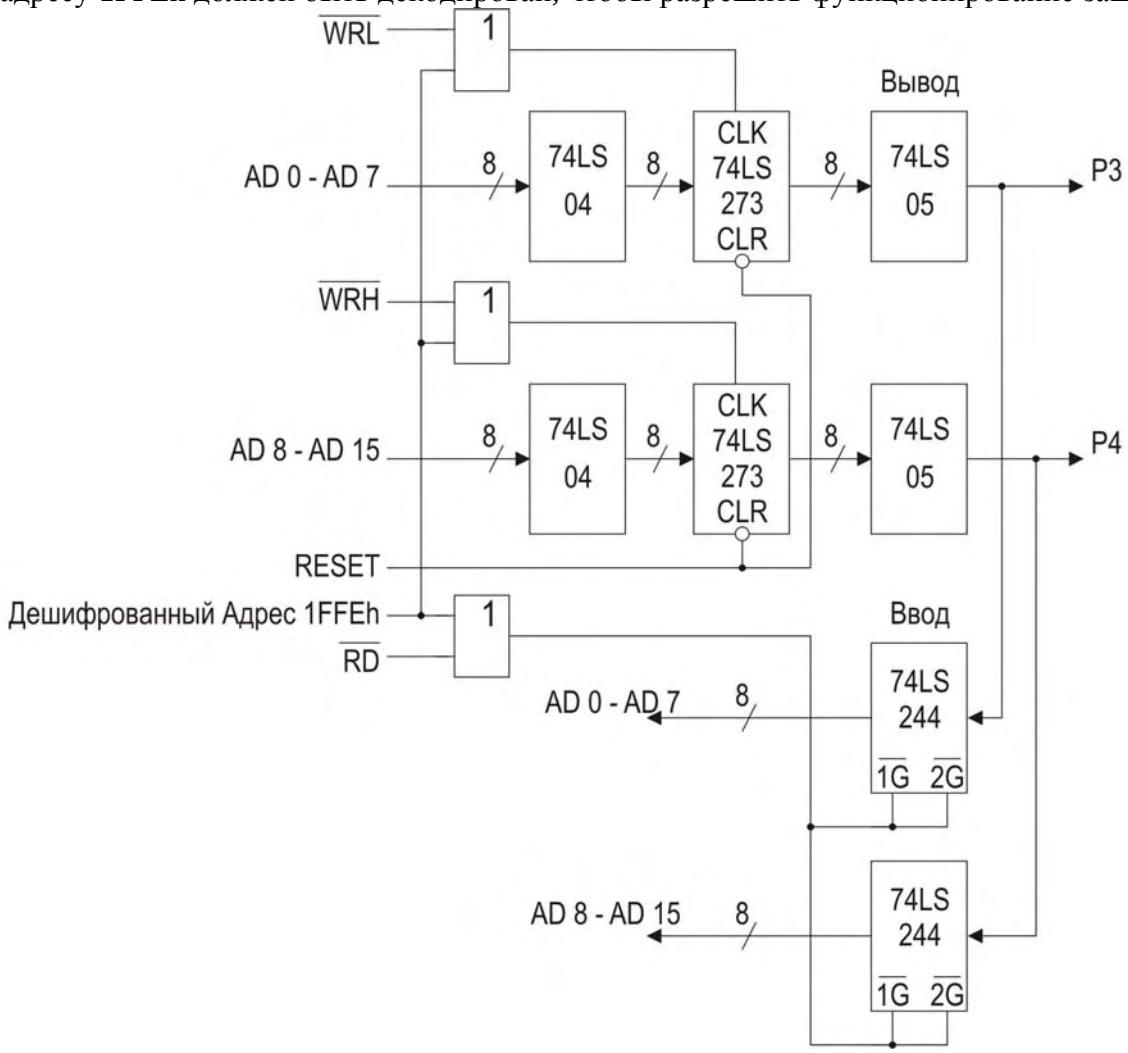


Рисунок 7.7 – Конфигурация Портов 3 и 4

Когда EA# имеет высокий уровень во время сброса, Порты 3 и 4 имеют 2 функции: I/O и системную шину адресов/данных. Обычно порты функционируют как I/O с открытым стоком. Ноль, записанный в защелку порта с адресом 1FFEh, открывает транзистор с мощным низким выходным уровнем, тогда как 1, записанная в защелку порта, закрывает выходной транзистор и переводит вывод в “плавающее” состояние. Запись “1” на выводы портов разрешает использовать их как входы. Чтение или запись в порт интерпретируется как внутренний доступ, если не выполняется цикл внешней шины. Порты 3 и 4 остаются портами I/O с открытым стоком все время, пока выполняется цикл внешней шины.

Любой доступ к адресу, не интерпретируемому как внутренний адрес, вызывает выполнение цикла доступа к внешней шине. Чтобы выполнить цикл шины, Порты 3 и 4 временно переключаются на функцию поддержки системной шины адрес/данные. Это значит, что выводы портов имеют мощные высокие и низкие уровни во время цикла шины записи адреса и данных и имеют “плавающие” состояния во время цикла шины считывания данных. После завершения цикла выводы портов переключаются обратно на функцию портов I/O. Состояние вывода порта после переключения зависит от последнего значения, записанного в защелку порта.

В проектах, где используется системная шина адрес/данные, значение, запрограммированное в защелке порта, имеет несколько назначений. Значение защелки порта сохраняется на шине во время простоя (когда Порты 3 и 4 не являются системной шиной). Запись нуля в Порты 3 и 4 означает, что на шине во время простоя сохраняется низкий уровень, тогда как запись единицы означает, что шина находится в “плавающем” состоянии во время простоя. “Плавающая” шина может привести к увеличению потребления тока, но его легко ограничить подключением выводов шины через резисторы к напряжению питания. Использование шины с низким уровнем снимает необходимость в резисторах, но необходимо быть уверенным, что никакое другое устройство не сможет подключиться к шине в то же самое время (возможно короткое замыкание).

Когда Порты 3 и 4 используются исключительно как системная шина адреса/данные, имеющая нагрузочные резисторы, то на шине устанавливается значение 0FFFFh, если команда считывается из несуществующей ячейки памяти. Выборка операнда 0FFh приводит к сбросу устройства. Следует предположить, что выборка команды из несуществующей памяти говорит об аппаратной или программной ошибке, поэтому сброс устройства полезен и предотвращает дальнейшую работу устройства.

7.3 Аппаратное подключение к квазидвунаправленным портам

При использовании QBD портов как входных с возможностью переключения, могут понадобиться резисторы, подключенные последовательно, если производится запись в порты после того, как они инициализированы. Например, запись “0” в порт, который подключен к VCC1 через внешнюю переключательную схему, приведет к короткому замыканию (большой ток потребления). Суммарный ток короткого замыкания для каждого вывода может превысить 20 мА. Этот ток может выйти за допустимые пределы для вывода или порта.

Эту потенциальную проблему можно решить и программно, и аппаратно. При программировании никогда не записывайте “0“ на вывод, используемый как вход. Аппаратно подключайте резистор номиналом 1 кОм, соединённый последовательно с каждым выводом, для ограничения тока до допустимого значения. Проблема не так существенна при подключении входов к электронным устройствам (TTL и CMOS) вместо переключательных схем, так как электронные устройства обычно не вызывают чрезмерного суммарного тока.

Запись в QBD порт с присоединенными к его выводам электронными устройствами требует особого внимания. Рассмотрим попытку переключения P1.1 в состояние выхода:

```
XORB IOPORT1, #00000010B     ; переключение P1.1
```

Проблема может возникнуть при выполнении команды. Даже если P1.1 в высоком уровне от МК, он может удерживаться в низком уровне. Это обычно случается, если вывод порта подключен к базе n-p-n транзистора, который управляет внешними устройствами. Переключение транзистора происходит при напряжении база-эмиттер чуть выше нуля, обычно около 0,7 В. МК примет это значение за “0”, если в порт была записана “1”. Поэтому команда XORB запишет “1” в SFR вывода порта и вывод не переключится.

Задачу можно решить аппаратно, используя внешний драйвер. Использование резисторов между выводом порта и базой транзистора часто помогает решить эту проблему, увеличивая напряжение на выводе порта. Программным решением является сохранение байта, который отображает данные, выводимые в порт, в ОЗУ. Всякий раз, когда программе необходимо изменить данные, выводимые в порт, можно модифицировать этот байт и скопировать его в порт.

Если переключатель использует длинную линию для подсоединения к выводу QBD порта, рекомендуется подсоединять его через резистор к VCC1, чтобы уменьшить возможность возникновения помех и время задержки в линии. На очень длинных линиях с низкоскоростным обменом для уменьшения помех в дополнение к резистору полезен и конденсатор.

8 Последовательный порт ввода-вывода

Последовательный порт ввода-вывода МК является синхронно/асинхронным портом, который включает в себя универсальный асинхронный приемник и передатчик (UART). UART имеет синхронный режим (Mode 0) и три асинхронных режима (Mode 1, Mode 2, Mode 3) как для приема, так и для передачи. Этот раздел дает обзор каждого режима и описывает как программировать последовательный порт.

Чтобы помочь пониманию каждого режима, следующие абзацы кратко описывают регистры, связанные с последовательным портом ввода-вывода.

SP_CON регистр выбирает режим обмена, разрешает и запрещает прием, проверку четности и передачу девятого бита данных (смотрите подраздел 8.3).

SP_STAT регистр содержит флаги “прерывание по приему” (RI) и “прерывание при передаче” (TI) и три других статусных бита (смотрите подраздел 8.4). Следует отметить, что чтение регистра SP_STAT очищает все флаги, исключая TXE. По этой причине рекомендуется записывать содержимое SP_STAT в рабочий регистр и выполнять команды, тестирующие биты, такие как JBC и JBS над рабочим регистром. Иначе, если будет выполнено более одной команды, тестирующей биты, возможен возврат неверного значения.

Последовательный порт принимает данные в SBUF(RX) регистр, а передает данные из порта через SBUF(TX) регистр.

Доступ к регистрам приема и передачи для чтения и записи осуществляется в режиме Горизонтальных окон (смотрите подраздел 5.5).

Приемник и передатчик буферированы, чтобы поддерживать процесс передачи и позволить принять второй байт до того, как первый байт был считан.

Прерывания последовательного порта разрешаются и запрещаются через регистры масок (INT_MASK или INT_MASK1); ожидаемые прерывания индицируются через регистр ожидания прерывания (INT_PEND или INT_PEND1). (Смотрите подраздел 8.3.3.).

Независимый генератор скорости управляет скоростью обмена последовательного порта. Сигналы синхронизации либо XTAL1, либо T2CLK. BAUD_RATE регистр выбирает скорость обмена и источник синхронизации (смотрите подраздел 8.3.4.). Следует обращаться к приложению Б за информацией о сигналах, обсуждаемых в этом разделе.

8.1 Режим 0 (MODE 0)

Наиболее частым использованием Режима 0, синхронного режима, является расширение функций ввода-вывода МК через сдвиговые регистры. Схема сдвигового регистра для режима 0 приведена на рисунке 8.1. В этом режиме на вывод TXD выводится последовательность из восьми тактовых импульсов, пока вывод RXD принимает или передает данные. Передаются 8-битные данные, младший значащий бит является первым во времени. На рисунке 8.2 приведена диаграмма распределения этих сигналов во времени. Следует заметить, что только Режим 0 использует RXD как выход с открытым стоком. В Режиме 0 RXD должно быть запрещено для начала передачи и разрешено для начала приема (смотрите подраздел 8.3.2). Когда RXD запрещено, записанное в SBUF (TX) начинает передаваться. Когда RXD разрешен, начинается прием либо по возрастающему фронту на RXD выводе, либо при обнулении флага RI.

Запрещение RXD останавливает процесс приема и запрещает дальнейшие приемы. Чтобы избежать частичного или нежелательного завершения приема, следует запрещать RXD только перед очисткой RI флага. Это можно осуществить в обработке прерывания использованием программных флагов или в коде основной программы использованием регистра ожидания прерывания для определения окончания приема.

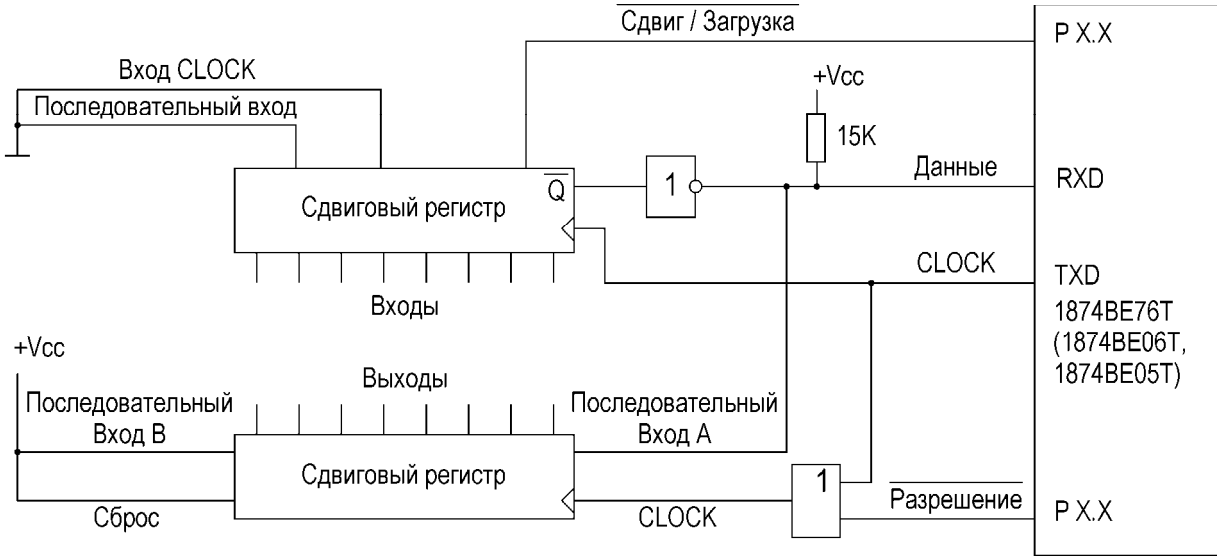


Рисунок 8.1 – Схема сдвигового регистра для режима 0

Во время приема RI флаг устанавливается за время одного бита, после того как принят последний бит. Сигнал прерывания по приему генерируется непосредственно перед тем, как устанавливается флаг RI. Во время передачи TI флаг устанавливается сразу после конца передачи последнего (восьмого) бита данных. Сигнал прерывания при передаче генерируется, если TI флаг установлен.

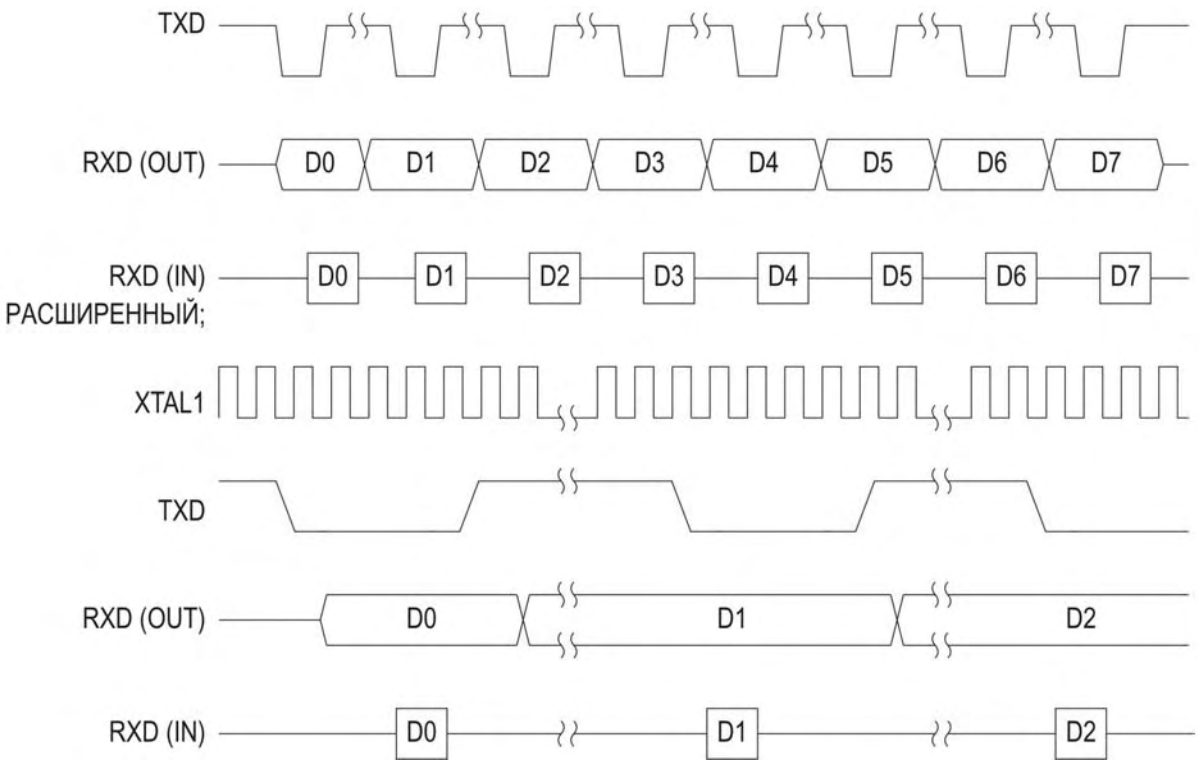


Рисунок 8.2 – Временные диаграммы для режима 0

8.2 Асинхронные режимы

Режимы 1, 2, 3 являются полнодуплексными режимами последовательной передачи/приема, это означает, что прием и передача в них могут происходить одновременно. Режим 1 является стандартным 8-битным асинхронным режимом, который используется в обычных последовательных коммуникациях. Режимы 2 и 3 – девятибитные асинхронные режимы, обычно используемые для межпроцессорных коммуникаций (смотрите подраздел 8.2.5). В режиме 2 последовательный порт генерирует прерывание, только если 9 бит равен единице. В режиме 3 последовательный порт генерирует прерывание всегда по окончанию передачи или приема данных.

Если последовательный порт сконфигурирован в Режимы 1, 2 или 3, то запись в SBUF (TX) вызывает начало передачи данных. Новые данные, размещенные в SBUF (TX), не передаются до тех пор, пока не будет послан стоп-бит предыдущих данных. Спадающий фронт сигнала на входе RXD вынуждает последовательный порт начать прием данных, если RXD разрешена. Запрещение RXD останавливает процесс приема и запрещает дальнейшие приемы (смотрите подраздел 8.3.2).

8.2.1 Режим 1

Режим 1 является стандартным режимом асинхронной связи. Блок данных, используемых в этом режиме (смотрите рисунок 8.3), содержит 10 бит: стартовый бит “0”, восемь битов данных (младший значащий (LSB) первый), стоповый бит “1”. Если четность разрешена, то бит дополнения до четности (even parity) посылается взамен восьмого бита данных, а контроль четности проверяется при приеме.

Передача и прием управляются отдельными сигналами синхронизации сдвига. Синхросигнал сдвига передачи запускается, если инициализирован генератор скорости передачи. Сигнал сдвига для приема возобновляется, если зафиксирован перепад из 1 в 0 (стартовый бит). Таким образом, синхронизации передачи и приема может и не быть, хотя оба процесса идут на одной частоте.

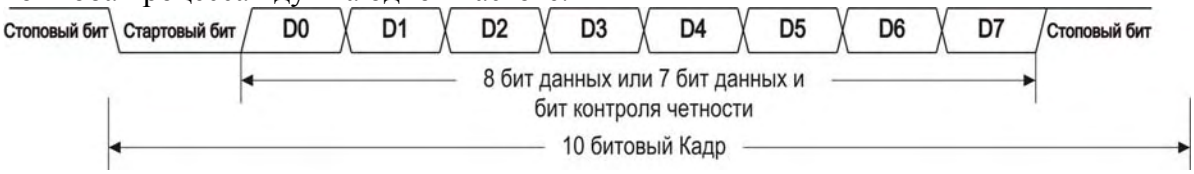


Рисунок 8.3 – Кадр последовательного порта в режиме 1

Флаги прерывания при передаче (TI) и приеме (RI) устанавливаются для индикации окончания операций. Во время приема флаг RI устанавливается как раз перед окончанием стопового бита. Сигнал “прерывание по приему” генерируется, если RI флаг установлен. Во время передачи TI флаг устанавливается в начале стопового бита. Сигнал “прерывание для передачи” генерируется, если TI флаг установлен. Следующий байт не может быть послан, пока не будет послан стоповый бит.

Будьте внимательны, если соединяются более чем два устройства по последовательному каналу в полудуплексе (когда передача и прием осуществляются по одному проводу). Принимающий процессор перед началом передачи должен задержать обмен на время одного бита после установки RI флага. Иначе передача может исказить стоповый бит, вызывая проблемы в другом устройстве, подсоединенном как приёмник.

8.2.2 Режим 2

Режим 2 – асинхронный режим, распознающий 9 бит. Этот режим обычно используется как и режим 3 для мультипроцессорных связей. На рисунке 8.4 показан формат посылки этого режима. Он содержит стартовый бит “0”, девять бит данных (LSB первый) и стоповый бит “1”. При передаче в качестве девятого передаваемого бита устанавливается бит TB8 в регистре SP_CON перед записью регистра SBUF (TX). Бит TB8 очищается аппаратно после каждой передачи, поэтому он должен быть установлен

(если необходимо) перед каждой записью в SBUF (TX). Во время приема флаг RI установлен, а прерывание генерируется, только если бит TB8 установлен. Это обеспечивает простой способ селектирования приема на линии данных (смотрите подраздел 8.2.5.). Четность в этом режиме не может быть разрешена.

Рисунок 8.4 – Кадр последовательного порта в режимах 2 и 3

8.2.3 Режим 3

Режим 3 – асинхронный 9-битный режим. Формат посылки в этом режиме идентичен режиму 2. Режим 3 отличается от режима 2 тем, что во время передачи может быть разрешена четность, и в этом случае девятый бит становится битом дополнения до четности.

Когда четность запрещена, биты 0-7 данных записываются в буфер передачи последовательного порта, а девятый бит данных в бит 4 (TB8) в регистре SP_CON. В режиме 3 прием всегда вызывает прерывание, независимо от состояния девятого бита. Если четность запрещена, девятый бит может быть считан в бит 7 (RB8) регистра SP_STAT. Если четность разрешена, то RB8 становится флагом ошибки по четности при приеме (RPE).

8.2.4 Временные соотношения в Режимах 2 и 3

Работа в Режимах 2 и 3 сходна с работой в Режиме 1. Единственное отличие это то, что данные содержат девятый бит, а при передаче и приеме пакет содержит 11 бит. Во время приема RI флаг устанавливается сразу после конца стопового бита. Сигнал прерывания генерируется, когда флаг RI установлен. Во время передачи TI флаг устанавливается в начале стоп-бита. Сигнал прерывания для передачи генерируется, когда флаг TI установлен. Девятый бит может быть использован для передачи четности или мультипроцессорных связей.

8.2.5 Мультипроцессорные связи

Режимы 2 и 3 обеспечивают мультипроцессорные связи. В режиме 2 последовательный порт генерирует прерывание по приему (RI), только если установлен девятый бит. В режиме 3 последовательный порт генерирует прерывание по приему (RI) независимо от значения девятого бита. Девятый бит всегда устанавливают при передаче адреса и очищают при посылке данных. Один из вариантов использования этих режимов для мультипроцессорных связей – установить главный процессор в режим 3, а ведомый – в режим 2. Когда главный процессор хочет передать блок данных в один из нескольких ведомых, он посылает адресный блок, который идентифицирует нужный ведомый. Адресный блок будет прерывать все ведомые, так как девятый бит установлен. Каждый ведомый может проверить байт адреса и понять, его ли это адрес. Адресованный ведомый переключается в Режим 3 для получения блока данных, а неадресованные ведомые остаются в Режиме 2 и не прерываются.

8.3 Программирование последовательного порта

В таблице 8.1 показаны регистры, которые воздействуют на работу и функции последовательного порта.

Таблица 8.1 – Управление последовательным портом и регистры состояния

Обозначение регистров	Название регистров	Описание
BAUD_RATE	Регистр скорости обмена	Данным регистром выбирается скорость обмена последовательного порта и исходная синхронизация. Регистр должен заполняться двумя байтами, первым - младший значащий байт. Старший значащий байт определяет источник синхронизацию. Младшие 15 бит содержат BAUD_VALUE, беззнаковое целое число, определяющее скорость обмена.
IOC1	Регистр 1 управления вводом-выводом	Установка бита 5 в этом регистре разрешает функцию TXD порта P2.0
SBUF (RX)	Буфер приёма последовательного порта	Регистр содержит данные, полученные из последовательного порта.
SBUF (TX)	Буфер передачи последовательного порта	Регистр содержит данные, готовые для передачи. В режимах 1, 2 и 3 запись в SBUF (TX) начинает передачу. В режиме 0 запись в SBUF (TX) начинает передачу только если прием запрещён (SP_CON.3 = 0)
SP_CON	Регистр управления последовательным портом	Регистр выбирает режим связи, разрешает и запрещает прием, а так же проверку дополнения до четности и передачу девятого бита. Бит TB8 очищается после каждой передачи.
SP_STAT	Регистр состояния последовательного порта	Регистр содержит биты состояния последовательного порта. В нем есть биты состояния для ошибки переполнения при приеме (OE), опустошения буфера передачи (TXE), ошибки кадра (FE), прерываний по передаче (TI), прерываний по приему (RI), а также ошибки четности при приеме (RPE) или приёма 8 бита (RB8). Чтение из SP_STAT обнуляет биты OE, FE, TI, RI и RPE/RB8. Запись байта в SBUF(TX) очищает бит TXE.
Примечание – Всегда следует записывать нули в зарезервированные биты этих регистров.		

8.3.1 Выбор режима связи и разрешение проверки чётности

Биты 0 и 1 в регистре SP_CON задают режим последовательного порта (смотрите рисунок 8.5 и таблицу 8.2). Выбор нового режима сбрасывает последовательный порт и прерывает процесс передачи (приема) в канале. Установка бита 2 разрешает проверку четности (parity). Установка бита 3 разрешает функцию RXD на выводе P2.1. Бит 4 является девятым битом данных, который передается в режимах 2 и 3, или, если разрешена проверки четности, это дополнение до четности. Заметим, что бит 4 (TB8) очищается после каждой передачи.

Таблица 8.2 – Выбор режима

Бит 1	Бит 2	Выбранный Режим
0	0	Режим 0 (Mode 0)
0	1	Режим 1 (Mode 1)
1	0	Режим 2 (Mode 2)
1	1	Режим 3 (Mode 3)

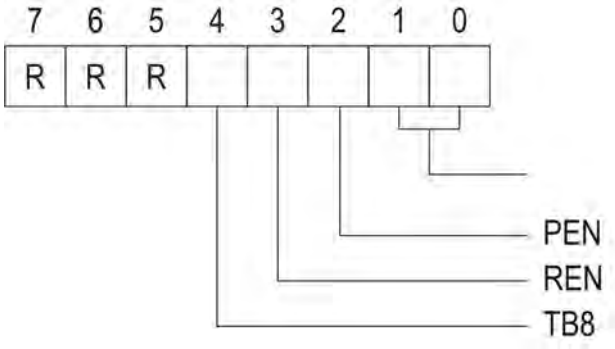


Рисунок 8.5 – Регистр SP_CON

8.3.2 Конфигурирование TXD и RXD

Таблица 8.3 показывает выводы, связанные с последовательным портом. Чтобы разрешить TXD, следует установить бит 5 регистра IOC1; чтобы запретить TXD следует обнулить этот бит. Разрешение RXD определяется установкой бита 3 регистра SP_CON; для запрещения RXD обнулите этот бит. Смотрите подраздел 8.3.4. для разрешения T2CLK.

Таблица 8.3 – Сигналы последовательного порта

Имя функции	Дополнительная функция	Выбор	Тип вывода	Описание
RXD	P2.1/PALE#	SP_CON.3 = 1	I/O	Прием последовательных данных. В Режимах 1, 2 и 3 RXD - вход для приема данных. В Режиме 0 это как вход, так и выход с открытым стоком для данных.
T2CLK	P2.3	BAUD_RATE MSB=0	I	Синхровход таймера 2 и один из двух возможных источников синхросигналов для входа генератора скорости обмена.
TXD	P2.0/PVER#	IOC1.5=1	O	Передача последовательных данных. В Режимах 1, 2 и 3 TXD – выход данных, передаваемых последовательным портом. В режиме 0 – это выход синхросигнала.

8.3.3 Разрешение работы по прерываниям для последовательного порта

Последовательный порт может быть сконфигурирован на генерацию либо единственного прерывания Serial Port, либо двух: прерывания для передачи (TI) и прерывания по приему (RI). Если прерывание Serial Port замаскировано, а прерывание для передачи и приема разрешены, генерируются раздельные прерывания по RI и TI флагам. Если совместимость с 8096BH не нужна, эта конфигурация предпочтительней.

Когда требуется совместимость с 8096BH, запретите прерывание по передаче и приему и разрешите Serial Port прерывание. Оба флага RI и TI генерируют это прерывание. Когда TI флаг генерирует прерывание, устанавливается бит 5 регистра SP_STAT, когда RI флаг генерирует прерывание, устанавливается бит 6 того же регистра.

Для разрешения отдельных прерываний установите биты TI_MASK и RI_MASK в регистре INT_MASK1. Для разрешения прерываний, совместимых с 8096BH, установите бит SER_MASK в регистре INT_MASK (смотрите раздел 6 для получения большей информации о прерываниях).

Таблица 8.4 – Регистры прерываний последовательного порта

Обозначение регистра	Имя регистра	16-ричный адрес	Описание
INT_MASK	Регистр маски прерывания	08h	Установка бита 6 этого регистра разрешает прерывание Serial Port последовательного порта (INT06, 200Ch), которое совместимо с конфигурацией 8096BH. Очистка бита 6 запрещает (маскирует) прерывание.
INT_MASK1	Регистр маски прерывания 1	13h	Установка бита 0 этого регистра разрешает прерывание для передачи (INT08, 2030h), очистка бита 0 запрещает (маскирует) прерывание. Установка бита 1 этого регистра разрешает прерывание по приему (INT09, 2032h), очистка бита 1 запрещает (маскирует) прерывание.
INT_PEND	Регистр ожидания прерывания	09h	Установка бита 6 этого регистра индицирует ожидание прерывания Serial Port (INT06) и обнуляется при переходе по вектору прерывания 200Ch.
INT_PEND1	Регистр ожидания прерывания 1	12h	Установка бита 0 индицирует ожидание передачи (INT09) и обнуляется по вектору прерывания 2030Ch. Установка бита 1 индицирует ожидание прерывания по приему (INT09) и обнуляется по вектору прерывания 2032h.

8.3.4 Программирование скорости обмена и источника синхронизации

Регистр BAUD_RATE выбирает вход синхронизации для генератора и определяет скорость обмена для всех режимов ввода-вывода. Этот двухбайтный регистр должен загружаться побайтно. Первым всегда загружается младший значащий байт.

Старший значащий бит выбирает один из двух возможных источников синхронизации для генератора скорости. Чтобы выбрать входной сигнал от XTAL1 (Fosc) как источник синхронизации, установите бит 15; для выбора внешнего сигнала на вывод T2CLK очистите бит 15. Максимальная частота на входе T2CLK - Fosc/4.

Младшие 15 бит представляют переменную BAUD_VALUE как беззнаковое целое, которое соответствует скорости обмена. Максимальное значение BAUD_VALUE 7FFFh (32767 десятичное). Минимальное значение в Режимах 1, 2 и 3 - это 0000h, если XTAL1 является генератором синхронизации, иначе оно 0001h. Минимальное значение в режиме 0 всегда 0001h.

Для задания скорости обмена через определения BAUD_VALUE необходимо использовать следующие соотношения:

Синхронный режим 0:

BAUD_VALUE = (Fosc /BAUD_RATE * 2) – 1 или T2CLK / BAUD_RATE

Асинхронные режимы 1, 2 и 3:

BAUD_VALUE = (Fosc / BAUD_RATE * 16) – 1 или T2CLK /BAUD_RATE * 8

В таблице 8.5 приведены стандартные значения BAUD_RATE (скоростей обмена), когда используется 16 МГц синхронизация по входу XTAL1. Из-за округления формула для вычисления BAUD_VALUE неточна, и полученное значение скорости обмена слегка отличается от требуемой. В таблице 8.5 приведены проценты ошибок, получаемые при использовании эталонного значения BAUD_RATE. В большинстве случаев последовательная связь будет работать при разнице до 5 % в скоростях приема и передачи.

Таблица 8.5 – Значение BAUD_RATE, если используется XTAL1 в 16 МГц

Скорость обмена	BAUD_RATE*		% ошибки	
	Режим 0	Режимы 1, 2, 3	Режим 0	Режимы 1, 2, 3
9600	8340h	8067h	0,04	0,16
4800	8682h	80CFh	0,02	0,16
2400	8D04h	81A0h	0,01	0,08
1200	9A0Ah	8340h	0	0,04
300	0E82Bh	8D04h	0	0,01
* бит 15 всегда установлен, если XTAL1 выбран как источник синхронизации генератора скорости обмена.				

8.4 Состояние последовательного порта

Состояние последовательного порта отражается в регистре SP_STAT (смотрите рисунок 8.6). Следует заметить, что чтение регистра SP_STAT очищает все биты кроме TXE. По этим причинам рекомендуется скопировать содержимое регистра SP_STAT в рабочий регистр, а затем выполнять команды, тестирующие биты (такие как JBC и JBS), с рабочим регистром. Иначе флаги будут очищены при выполнении команд, тестирующих биты.

Приемник МК проверяет наличие стопового бита. Если стопового бита нет в требуемое время, устанавливается ошибка кадра (FE). Если стоповый бит обнаружен, данные загружаются из сдвигового регистра приемника в SBUF(RX), и устанавливается флаг прерывания по приему (RI). Если это произошло прежде, чем считан предыдущий байт из SBUF(RX), устанавливается бит ошибки переполнения (OE). SBUF(RX) всегда содержит последний принятый байт и никогда комбинацию из двух байт.

Флаг прерывания по приёму (RI) показывает, получен ли принимаемый байт данных. Флаг прерывания для передачи (TI) показывает, кончилась ли передача байта данных. Таким образом, эти флаги запускают прерывания и устанавливают соответствующие биты в регистре ожидания прерывания (смотрите подраздел 8.3.3).

Заметим, что в то время, как событие приема/передачи устанавливает бит RI/TI в SP_STAT и соответствующие биты ожидания прерывания (INT_PEND, INT_PEND1), программная запись RI/TI в SP_STAT не влияет на биты Ожидания Прерывания и не вызывает прерывания. Аналогично, чтение SP_STAT очищает биты RI и TI, но не очищает соответствующий бит в регистре Ожидания Прерывания.

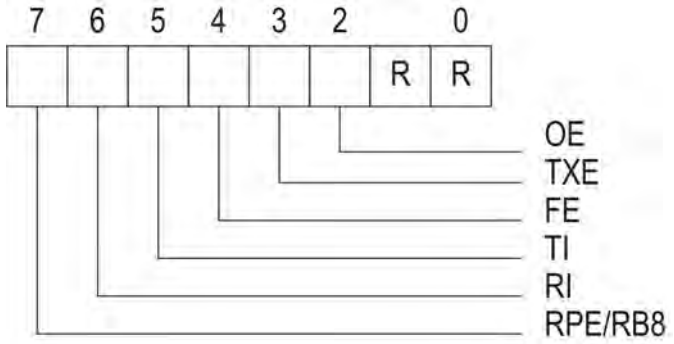


Рисунок 8.6 – Регистр SP_STAT

Бит Передатчик Пуст (TXE) устанавливается, если SBUF(TX) и его буфер пусты и готовы принять два байта. TXE очищается, как только байт записан в SBUF(TX). Один байт может быть записан, если только TI установлен. По определению, если TXE уже установлен, то передача окончена и TI также установлен.

Флаг Ошибка Четности при Передаче (RPE) или Флаг Приема Бита 8 (RB8) определяются в зависимости от разрешения или запрещения проверки четности соответственно. Когда проверка четности разрешена, то RPE устанавливается, если обнаружена ошибка четности. Если проверка четности не разрешена, в RB8 помещается девятый бит данных, посылаемый в Режимах 2 и 3.

9 Устройство высокоскоростного ввода-вывода

МК содержат четыре устройства, которые вместе формируют гибкое, базируемое на таймере - счетчике устройство скоростного ввода-вывода (HSIO). Устройства, входящие в HSIO: таймер 1, таймер 2, модуль высокоскоростного ввода и модуль высокоскоростного вывода (смотрите рисунок 9.1). HSIO может измерять ширину импульсов, вырабатывать импульсную последовательность и вызывать периодические прерывания с незначительным использованием ЦПУ. Этот раздел описывает каждый модуль внутри HSIO устройства.

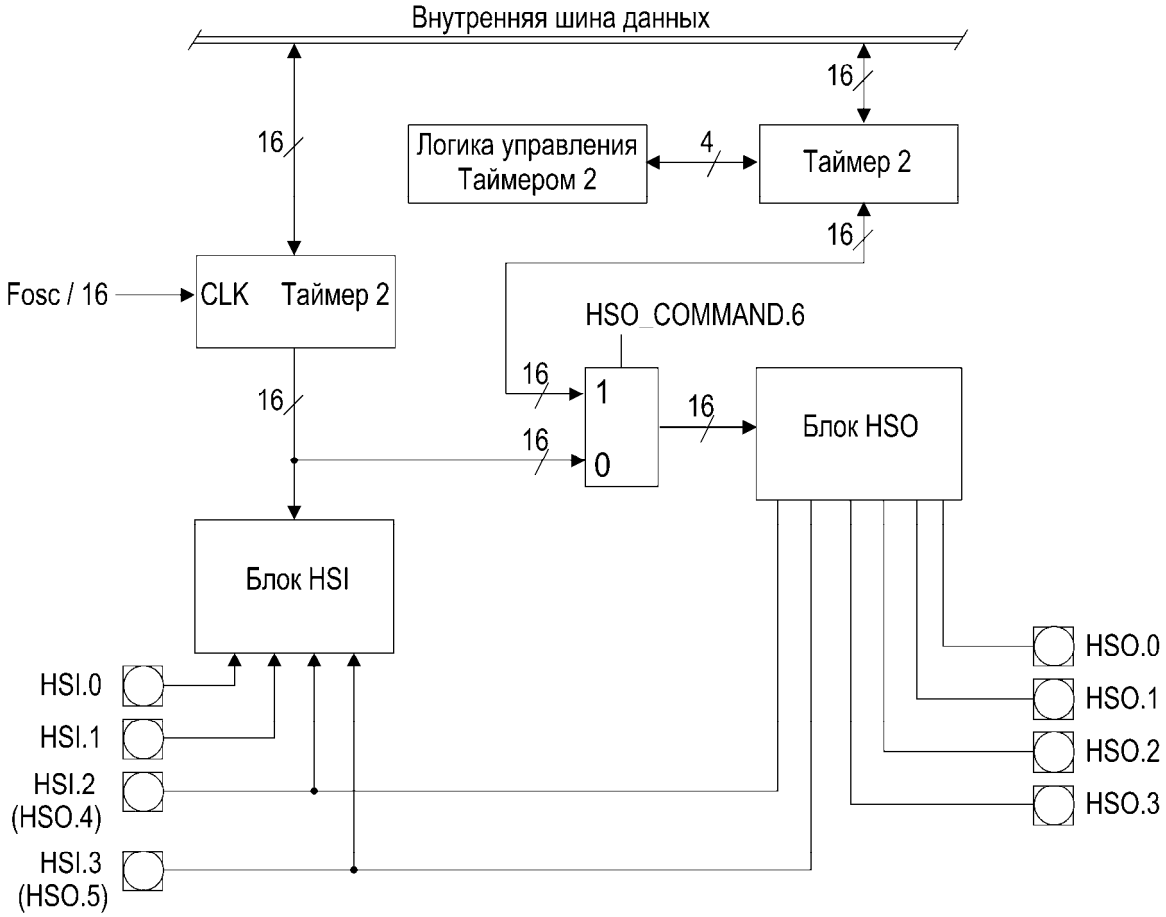


Рисунок 9.1 – Блок схема модуля HSIO (HSI + HSO)

9.1 Таймеры

МК имеет два 16-битных таймера: таймер 1 и таймер 2. HSI-модуль всегда использует таймер 1 как базовый. HSO-модуль использует в качестве базового таймер 1 или таймер 2.

9.1.1 Функции таймера 1

Таймер 1 является стандартным 16-битным независимым таймером, который увеличивается на каждый восьмой такт процессора. Таймер 1 всегда является базовым для HSI-модуля. Он также может быть выбран как базовый для HSO-модуля. Два байта регистра TIMER1 содержат его значение. Вы можете инициализировать таймер 1, записав отличное от нуля значение в регистр TIMER1 через HWindow15. Если Вы изменили значение TIMER1 после инициализации HSI-модуля, Вы можете исказить временное соотношение между HSI-событиями. Изменение значения TIMER1 после инициализации HSO-модуля, если таймер 1 является базовым, может вызвать потерю HSO команд.

9.1.2 Функции таймера 2

Таймер 2 – реверсивный программируемый 16-битный счетчик, который может являться базовым для HSO-модуля или использоваться как высокоскоростной счетчик. Он может также фиксировать внешние события. Таймер 2 может тактироваться внешним или внутренним источником. При внешнем источнике в качестве источников тактов Вы можете выбрать выводы T2CLK или HSI.1. Максимальная скорость счета для таймера 2: каждый такт ЦПУ (быстрый режим) или каждый восьмой такт (нормальный режим).

Два байта регистра TIMER2 содержат его значение. Вы можете инициализировать таймер 2, записав отличное от нуля значение в регистр TIMER2 через HWindow0. Когда используется таймер 2 как базовый для HSO, изменение его значения после инициализации модуля HSO может вызвать пропуск HSO команд. Дополнительную информацию смотрите в подразделе 9.1.6.

9.1.3. Программирование модуля таймера 2

В таблице 9.1 показаны регистры, влияющие на работу и функции модуля таймера 2.

Таблица 9.1 – Регистры управления и статуса таймера 2

Мнемоника	Имя	Описание
TIMER 2	Timer 2	Содержит значения таймера 2
T2CAPTURE	Timer 2 Capture	Нарастающий фронт на P2.7 вызывает захват значения таймера 2 в этом регистре и производит прерывание Timer 2 Capture (INT11)
INT_MASK INT_MASK1	Interrupt Mask InterruptMask1	Разрешение или запрещение прерываний таймера 2
IOC0	Input/Output Control Register 0	Выбирает внешний источник синхронизации и сброса для таймера 2
IOC1	Input/Output Control Register 1	Выбирает источники прерывания для прерывания Timer Overflow (INT07)
IOC2	Input/Output Control Register 2	Разрешает или запрещает быстрый режим и работу счетчика на увеличение или уменьшение, выбирает границу переполнения для прерывания Timer Overflow (INT12)
IOC3	Input/Output Control Register 3	Выбирает внутренний или внешний источник синхронизации для таймера 2

Таблица 9.2 показывает отдельные биты и выводы порта, которые управляют режимами таймера 2. Колонка 2 показывает результат установки бита или вывода порта. Колонка 3 показывает результат очистки бита или вывода порта.

Таблица 9.2 – Выводы и биты для управления таймером 2

Биты регистра выводы порта	Бит = 1	Бит = 0
1	2	3
IOC 0.1	Сброс таймера 2 при каждой записи	Не действует
IOC 0.3	Разрешение внешнего сброса	Запрещение внешнего сброса
IOC 0.5	HSI.0 является источником внешнего сброса	T2RST (P2.4) является внешним источником сброса
IOC 0.7	HSI.1 является источником внешних	T2CLK (P2.3) является источником
IOC 1.3	Разрешение таймера 2 как источника прерывания Timer Overflow (INT00)	Запрещение таймера 2 как источника прерывания Timer Overflow (INT00)
IOC 2.0	Разрешение режима быстрого увеличения	Запрещение режима быстрого увеличения
IOC 2.1	Разрешение счета на уменьшение	Счет только на увеличение
IOC 2.5	Прерывание на границе 7FFFh/8000h	Прерывание на границе 0FFFFh/0000h
T2UP-DN (P2.6)	Счет на уменьшение, если IOC2.1=1	Счет на увеличение, если IOC2.1=1
T2CAP (P2.7)	Захват данных из таймера 2 в T2CAPTURE имеет место, когда происходит положительный перепад и логический уровень сохраняется стабильным в течение одного машинного цикла.	

9.1.3.1. Выбор источника тактового сигнала

Таймер 2 считает оба перепада: положительный и отрицательный. Он может тактироваться или внутренне, или внешне в зависимости от состояния бита T2_ENA в регистре IOC3 (IOC3.0). Установка IOC3.0 выбирает внутренний источник синхронизации; сброс этого бита выбирает внешний источник (смотрите рисунок 9.2).

Когда выбран внешний источник синхронизации, бит T2CLK_SRC в регистре IOC0 (IOC0.7) управляет выбором одного из двух внешних источников. Установка IOC0.7 выбирает HSI.1 вывод как источник внешней синхронизации, очистка - выбирает вывод T2CLK (P2.3) как внешний источник синхронизации.

Когда таймер 2 синхронизируется внешне, то бит FAST_T2_ENA в регистре IOC2 (IOC2.0) управляет максимальной входной частотой на внешнем выводе. Когда установлен FAST_T2_ENA, максимальная входная скорость ввода – один раз в такт (быстрый режим). Когда FAST_T2_ENA очищен, максимальная скорость ввода - один раз в восемь тактов (нормальный режим).

Когда таймер 2 тактируется внутренне, бит FAST_T2_ENA определяет значение входной тактовой частоты. Когда установлен FAST_T2_ENA, тактовая частота равна Fosc/4, и таймер 2 увеличивается каждый такт (быстрый режим). Когда FAST_T2_ENA очищен, тактовая частота равна Fosc/32, и таймер 2 увеличивается каждый восьмой такт.

Если таймер 2 является базовым для HSO-модуля, используется нормальный режим. HSO-модуль требует восемь тактов для выборки всех CAM вводов. HSO событие

может быть пропущено, если таймер 2 использует быстрый режим (смотрите подраздел 9.1.6).

9.1.3.2. Установка Направления Счёта

Бит T2UP_ENA в регистре IOC2 (IOC2.1) управляет таймером 2. Таймер считает либо только на увеличение, либо на увеличение или уменьшение в зависимости от значения вывода T2UP_DN (смотрите рисунок 9.2). Если IOC2.1 очищен, таймер 2 всегда считает на увеличение. Если IOC2.1 установлен и на T2UP_DN низкий уровень, таймер 2 считает на увеличение. Если IOC2.1 установлен и на T2UP_DN высокий уровень, таймер 2 считает на уменьшение.

Сигнал T2UP_DN должен быть стабилен до изменения сигнала T2CLK или во время его изменения, чтобы быть уверенным, что таймер изменит направление счета в следующем такте.

Когда таймер 2 является базовым для HSO-модуля, не изменяйте направление счета, иначе события могут произойти в неправильном порядке (смотрите раздел 9.1.6.).

9.1.3.3 Выбор сброса таймера 2

Таймер 2 может быть сброшен аппаратно, программно или модулем HSO (смотрите рисунок 9.2). Установка бита T2RST_ENA (IOC0.3) разрешает источник внешнего сброса. Установка бита T2RST_SRC (IOC0.5) выбирает вывод HSI.0, а очистка этого бита выбирает T2RST (P2.4) как внешний сигнал сброса таймера 2. Нарастающий фронт выбранного сигнала сбрасывает таймер 2.

Программа может сбросить этот таймер установкой SW_T2RST бита (IOC0.1). HSO может сбросить этот таймер выполнением команды RESET Timer2 (CMD_TAG=0Eh). Этот вариант сброса обычно применяется, когда HSO-модуль используется как широтно-импульсный модулятор (PWM) (смотрите раздел 9.3.3).

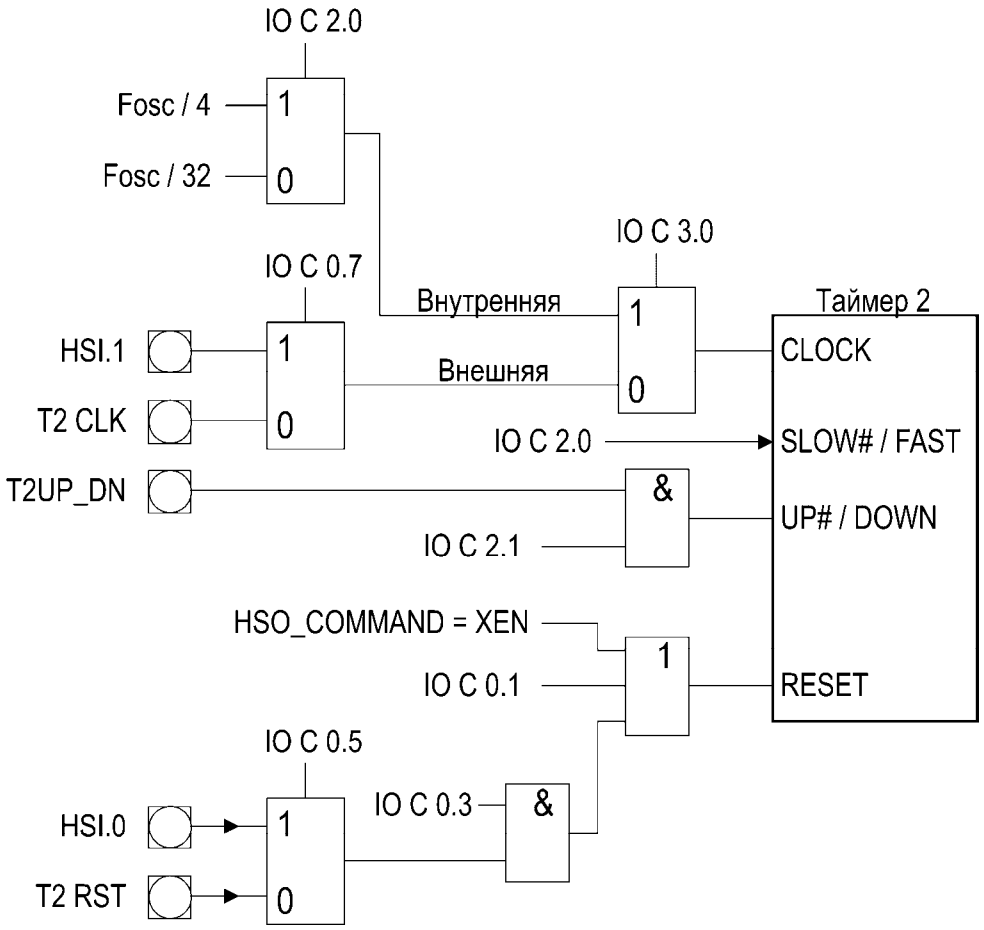


Рисунок 9.2 – Логика управления таймером 2

9.1.4 Использование внешних входов таймера 2

Значения на выводах T2UP_DN, T2CLK, T2RST и T2CAP фиксируются, когда CLKOUT имеет низкий уровень. Эти входы должны быть стабильными, по крайней мере в течение одного такта, либо должны удерживать свое значение не менее 45 нс до нарастающего фронта CLKOUT, чтобы гарантировать правильное распознавание сигналов.

9.1.4.1 Синхронизация

T2RST всегда синхронизируется от внутреннего счетчика по модулю 8. Общее время до сброса таймера зависит от того, когда активизируется T2RST. Сброс таймера может осуществиться с 1 по 9 такт после установки T2RST. Это действует как для нормального, так и для быстрого режима счета.

Во время режима нормального счета T2CLK и T2CAP, также синхронизируются от внутреннего счетчика по модулю 8. Во время быстрого режима эти сигналы подаются прямо на таймер 2. В случае, когда оба сигнала устанавливаются одновременно, внутренняя схема таймера 2 гарантирует, что захват осуществится перед увеличением счетчика.

9.1.4.2 Одновременная установка T2RST, T2CLK и T2CAP

Когда T2RST, T2CLK и T2CAP устанавливаются одновременно, поведение таймера 2 можно предсказать. Поведение таймера 2 определяется временем, когда сигнал устанавливается по отношению к внутреннему счетчику по модулю 8. Внутренний счетчик по модулю 8 имеет 8 фиксированных состояний, пронумерованных от 1 до 8. При увеличении на единицу счетчик переходит из одного состояния в другое. После достижения состояния 8 он возвращается в состояние 1 и продолжает изменяться бесконечно по циклу. Сигналы T2CAP и T2CLK воздействуют на таймер 2 во время состояния 1; T2RST сигнал транслируется на таймер 2 во время состояния 2. Так как нет возможности синхронизировать события с внутренним счетчиком, Вы должны решить эту проблему программно или аппаратно сами.

9.1.4.2.1 Режим нормального счета

Когда T2RST, T2CLK и T2CAP одновременно устанавливаются во время состояния 1, сброс осуществляется первым.

Последовательность событий	Содержимое TIMER2	Содержимое T2CAPTURE1
1 Сброс TIMER2	0000h	????h
2 Захват внешнего события	0000h	0000h
3 Инкремент TIMER2	0001h	0000h

Когда сигналы одновременно устанавливаются во время состояний 2-8, захват и инкремент осуществляются перед сбросом.

Последовательность событий	Содержимое TIMER2	Содержимое T2CAPTURE2
1 Захват внешнего события	5A56h	5A56h
2 Инкремент TIMER2	5A57h	5A56h
3 Сброс TIMER2	0000h	5A56h

9.1.4.2.2. Режим быстрого счета (Fast Increment)

Когда T2RST, T2CLK и T2CAP одновременно устанавливаются во время состояний 1, 3, 4, 5, 6, 7 или 8, захват и инкремент осуществляются перед сбросом.

Последовательность событий	Содержимое TIMER2	Содержимое T2CAPTURE1
1 Захват внешнего события	5A56h	5A56h
2 Инкремент TIMER2	5A57h	5A56h
3 Сброс TIMER2	0000h	5A56h

Когда они одновременно устанавливаются во время состояния 2, осуществляется захват, затем сброс и, наконец, инкремент.

Последовательность событий	Содержимое TIMER2	Содержимое T2CAPTURE2
1 Захват внешнего события	5A56h	5A56h
2 Сброс TIMER2	0000h	5A56h
3 Инкремент TIMER2	0001h	5A56h

9.1.5 Прерывания таймера

С таймером 1 и таймером 2 связаны три вектора прерывания:

- прерывание по переполнению таймера (Timer Overflow)
- прерывание по переполнению таймера 2 (Timer2 Overflow)
- прерывания по фиксации таймера 2 (Timer2 Capture).

Регистр IOS1 содержит флаги, которые показывают, какое событие вызвало прерывание. Чтение регистра IOS1 очищает биты 0–5. Поэтому рекомендуется скопировать содержимое регистра IOS1 в рабочий регистр и затем выполнить команды проверки битов, такие как JBC или JBS, с рабочим регистром.

9.1.5.1 Прерывание по переполнению таймера

Таймер 1 и таймер 2 могут вызвать прерывание по переполнению таймера Timer Overflow (INT00). Установка INT_MASK.0 разрешает это прерывание. Установка IOC1.2 (Таймер 1) или IOC1.3 (Таймер 2) выбирает источник прерывания. Когда происходит переполнение, флаг состояния устанавливается в IOS1 регистре. Переполнение таймера 1 устанавливает IOS1.5, а переполнение таймера 2 устанавливает IOS1.4.

9.1.5.2 Прерывание по тереполнению таймера 2

Таймер 2 может вырабатывать прерывание по переполнению таймера 2 (Timer 2 Overflow - INT12) взамен стандартному прерыванию по переполнению таймера (Timer Overflow). Это прерывание разрешается установкой INT_MASK1.4. Переполнение Timer2 устанавливает IOS1.4.

Таймер 2 может вырабатывать прерывание по переполнению таймера 2 при переходе границы 0FFFFh/0000h или 7FFFh/8000h. Переполнение может осуществляться в любом направлении. IOC2.5 выбирает границу переполнения. Когда установлен IOC2.5, таймер 2 вырабатывает прерывание на границе 7FFFh/8000h. В противном случае прерывание происходит на границе 0FFFFh/0000h.

9.1.5.3 Прерывание по захвату значения таймера 2 (Timer2 Capture)

Положительный перепад на выводе T2CAP вызывает загрузку значения таймера 2 в регистр T2CAPTURE. Это событие вырабатывает прерывание Timer2 Capture (INT11), если INT_MASK1.3 установлен и T2CAP удерживается более чем два такта.

9.1.6 Предосторожности при работе с таймером

Когда таймеры используются как базовые в HSI или HSO модулях, соблюдение этих рекомендаций поможет избежать возможных проблем. В этом подразделе приведены перечень возможных проблем и рекомендаций.

- Следует быть осторожным при записи в регистры таймеров TIMER1 и TIMER2.
- Необходимо конфигурировать таймер 2 для работы в нормальном режиме (не в режиме быстрого счёта).
- Следует конфигурировать таймер 2 для счета в одном направлении.
- Необходимо быть осторожным при сбросе таймера 2.

Когда таймер 2 сконфигурирован на сброс от внешнего вывода, программные события могут не осуществиться при значении таймера 2, равном нулю.

Изменение значения таймера 1 после инициализации модуля HSI может исказить временное соотношение между HSI событиями. Также изменение опорного значения таймера (1 или 2) после инициализации модуля HSO может вызвать потерю или неправильный порядок запрограммированных HSO событий.

Так как HSO требует 8-и тактов для полного поиска в CAM, таймер 2, при использовании его как базового для HSO, должен работать в режиме нормального счёта (а не в режиме быстрого счёта). Очистка бита FAST_T2_ENA (IOC2.0) устанавливает режим нормальной работы.

Не следует сбрасывать таймер 2 до достижения им максимального значения времени, запрограммированного в CAM. CAM удерживает ожидаемое событие до тех пор, пока не осуществится временное соответствие. Если значение таймера 2 не достигается, событие будет оставаться ожидаемым до тех пор, пока устройство не сбросится или CAM не очистится.

Когда таймер 2 сконфигурирован на сброс внешним выводом (установлен IOC0.3), не следует программировать события, осуществляющиеся при значении таймера 2, равном 0. Событие может не осуществиться, если HSI.0 или T2RST (P2.3) сбрасывают таймер 2. Очистка таймера 2 внешними выводами асинхронна, и таймер 2 может увеличиться на 1 до того, как HSO сможет сравнить и вызвать событие из CAM. Программирование событий при значении таймера 2, равном 1, гарантирует наличие достаточного количества времени для опознавания модулем HSO входа в CAM.

9.2 Модуль высокоскоростного ввода HSI

HSI модуль работает от 4 внешних выводов (HSI.0-HSI.3). Когда предопределенные события осуществляются на одном или более выводах, HSI модуль записывает текущее значение счетчика таймера 1 вместе с битом состояния для каждого входа. HSI модуль сохраняет данные максимум для семи событий в FIFO очереди размером (7×20) бит и для одного дополнительного события в фиксирующем регистре. Данные переносятся из FIFO в регистр захвата и могут быть считаны только после загрузки их в этот регистр.

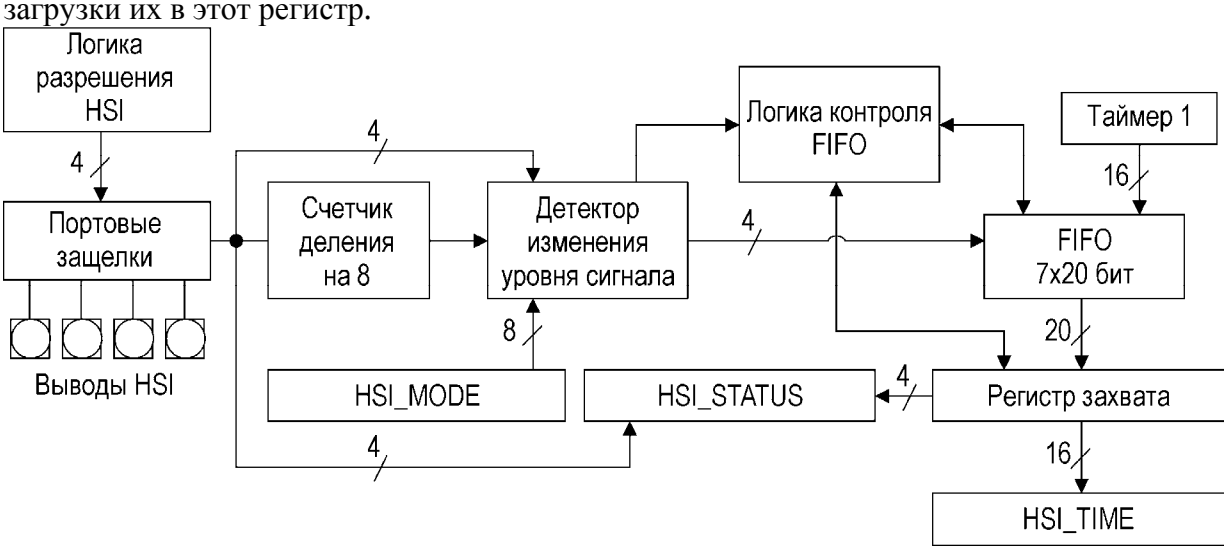


Рисунок 9.3 – Блок-схема модуля HSI

HSI (смотрите блок-схему модуля HSI на рисунке 9.3) может быть запрограммирован на захват каждого положительного перепада и отрицательного перепада, любого перепада или на захват последовательности из 8 положительных перепадов. Эта гибкость позволяет использовать HSI модуль для измерения различных входных параметров (длительность импульсов, период, скважность, сравнение фаз и т.д.). Запись последовательности из 8 положительных перепадов позволяет считать и измерять короткие импульсы.

9.2.1 Временные соотношения для событий в HSI

События загружаются в HSI FIFO с максимальной скоростью – одно событие в каждые девять тактов. Всегда необходимо иметь 9 тактов между последовательностью событий на каждом выводе, иначе HSI может потерять событие. Если вывод сконфигурирован в режиме восьми переключений, максимальная скорость для положительных фронтов – 8 раз в 18 тактов.

При постоянном повторении событий, которые осуществляются один раз в 9 тактов, несоответствие между девятитактным разрешением HSI и восьмитактным таймером может вызывать пропуск одного значения метки времени на каждом 9 цикле таймера. Если событие осуществляется сразу же перед увеличением значения таймера на 1, и следующее событие произойдет после девяти периодов, таймер увеличится на единицу дважды между двумя событиями (смотрите рисунок 9.4). В результате в FIFO сохранится значение таймера на единицу больше, чем ожидается при каждом девятом счете.

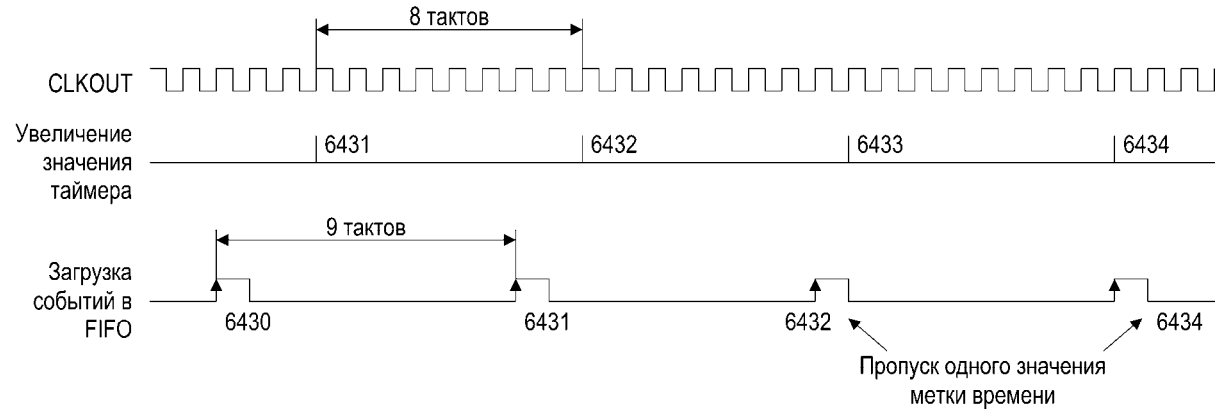


Рисунок 9.4 – Потеря значения таймера

Первое событие в пустом FIFO будет передаваться в регистр захвата после восьми тактов. Если второе событие осуществится после перемещения первого в регистр захвата, с ним обращаются как с первым событием в пустом FIFO. Дополнительные события не записываются, если и FIFO, и регистр захвата заполнены.

Если первые два события в пустом FIFO (исключая регистр захвата) осуществляются во время одной внутренней фазы, оба они записываются с одинаковым временным значением. Однако, если второе событие осуществляется в пределах 9 тактов после первого, его временное значение на 1 больше, чем временное значение для первого. Если временное значение пропущено, то временная метка второго события на 2 больше, чем у первого.

Логика обнаружения изменения фиксирует значение на HSI выводах, когда CLKOUT имеет низкий уровень. Чтобы гарантировать, что событие HSI запишется с правильным значением таймера 1, HSI вход должен быть стабильным, по крайней мере, один машинный такт или должен устанавливаться не менее, чем за 45 нс перед нарастающим фронтом CLKOUT. Если эти условия не выполнились, значение HSI входа может быть зафиксировано в следующем CLKOUT. Следовательно, если необходимо синхронизироваться с HSI модулем, то следует фиксировать информацию по нарастающему фронту CLKOUT.

Внутренний счетчик-делитель на 8 в HSI модуле постоянно считает перепады. Когда программа на входе HSI обнаруживает каждый восьмой положительный перепад, счетчик не сбрасывается (он сохраняет накопленное значение и продолжает считать). Если накопленное значение равно 6, потребуется только два дополнительных положительных перепада для фиксации HSI события. По этой причине лучше пропустить первое HSI событие, когда используется режим восьми перепадов.

9.2.2 Чтение информации о событии

Информация о событии может быть считана только после загрузки ее в регистр захвата. Необходимо 8 тактов для перемещения данных через FIFO в регистр захвата. Когда событие перемещается из FIFO в регистр захвата, устанавливается бит HSI_RDY в регистре IOS1 (IOS1.7). Если прерывание разрешено, и событие выбрано в качестве источника прерывания, генерируется прерывание HSI Data Available (INT02) . Более подробно смотрите подраздел 8.2.3.2.

Для чтения фиксирующего регистра первым необходимо считать регистр HSI_STATUS. Это заставляет регистр захвата загрузить биты состояния события для HSI.0 – HSI.3 в четные биты регистра HSI_STATUS. Установленные биты состояния показывают, что событие произошло на соответствующем HSI выводе. Нечетные биты всегда содержат текущее состояние выводов HSI.0 – HSI.3.

Следующим читается регистр HSI_TIME. Регистр захвата загружает значение счетчика TIMER1 в регистр HSI_TIME. Чтение регистра HSI_TIME заставляет FIFO загрузить следующее событие в регистр захвата. Если вы читаете регистр HSI_TIME до первого чтения регистра HSI_STATUS, то биты состояния события будут перезаписаны.

9.2.3 Программирование HSI модуля

Таблица 9.3 показывает регистры, которые влияют на работу и на функции модуля HSI.

Таблица 9.3 – Регистры состояния и управления HSI

Мнемоника регистра	Имя регистра	Описание
HSI_MODE	Режим HSI	Описывает тип перепада, который вызывает событие на каждом выводе HSI
HSI_STATUS	Состояние HSI	Содержит биты обнаружения события и входные текущие уровни для каждого HSI вывода
HSI_TIME	Время HSI	Содержит 16-битное значение таймера 1 после того, как событие произошло
INT_MASK INT_MASK1	Маска прерывания Маска прерывания 1	Разрешает и запрещает прерывания от HSI
IOC0	Регистр 0 управления входом/выходом	Разрешает и запрещает каждый вывод HSI. Управляет альтернативными функциями для HSI.0 и HSI.1
IOC1	Регистр 1 управления входом/выходом	Управляет источником для HSI Data Available Interrput (INT02) прерывания
IOS1	Регистр 1 состояния входа/выхода	Показывает состояние HSI FIFO

9.2.3.1 Описание HSI События

Регистр HSI_MODE указывает для каждого вывода HSI, какой из 4 типов событий вызовет захват в HSIFIFO: каждый положительный перепад, каждый отрицательный перепад, любой перепад (и положительный, и отрицательный), последовательность из 8 положительных перепадов (смотрите рисунок 9.5).

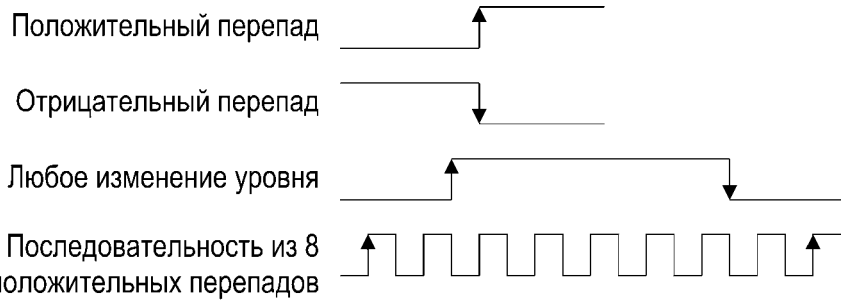


Рисунок 9.5 – Возможные события для HSI

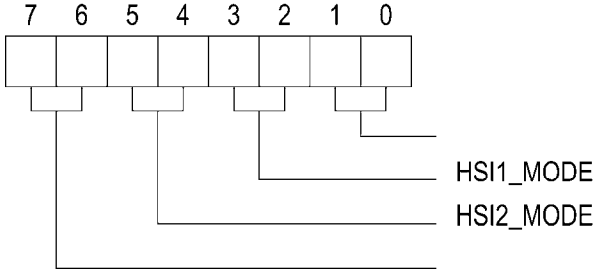


Рисунок 9.6 – Регистр HSI_MODE

Каждое 2-битное поле в регистре HSI_MODE программируется для того, чтобы определить режим перепада на соответствующем входе HSI (смотрите рисунок 9.6 и таблицу 9.4).

Таблица 9.4 – Расшифровка режимов перепада

	Описание
00	Восемь положительных перепадов вызывают захват в HSI FIFO
01	Каждый положительный перепад вызывает захват в HSI FIFO
10	Каждый отрицательный перепад вызывает захват в HSI FIFO
11	Каждый перепад (и положительный, и отрицательный) вызывает захват в HSI FIFO

9.2.3.2 Разрешение Прерываний от HSI

HSI может вызывать прерывание, когда происходит одно из следующих событий:

- Четвертое значение загружается в FIFO.
- Шестое значение загружается в FIFO.
- Значение из FIFO перемещается в регистр захвата.

Кроме того, вывод HSI.0 может быть сконфигурирован для работы как независимое внешнее прерывание. Необходимо запретить его как HSI вход, чтобы он функционировал как внешнее прерывание.

Регистры INT_MASK и INT_MASK1 содержат биты, которые разрешают и запрещают каждое прерывание. Таблица 9.5 кратко описывает условия, вызывающие каждое HSI прерывание, порядок прерываний и приоритеты.

Таблица 9.5 - Прерывания HSI

Источники прерываний	Когда вырабатывается	Имя прерывания	Приоритет *
HSI FIFO Full **	INT_MASK1.6 = 1 и шестое значение загружено в FIFO. Содержимое регистра хранения не изменяется	HSI FIFO Full (INT14)	14
	IOC1.7 = 1, INT_MASK.2 = 1, INT_MASK1.6 = 0 и шестое значение загружено в FIFO. Содержимое регистра хранения не изменяется	HSI Data Available (INT02)	2

Окончание таблицы 9.5

Источники прерываний	Когда вырабатывается	Имя прерывания	Приоритет *
HSI FIFO 4	INT_MASK1.2 = 1 и четвёртое значение загружено в FIFO. Содержимое регистра хранения не изменяется	HSI FIFO 4 (INT10)	10
HSI.0 External	INT_MASK1.4 = 1 и сигнал HSI.0 переключается из низкого уровня в высокий и сохраняет его в течение как минимум одного периода. Это прерывание может быть сконфигурировано, чтобы действовать как независимое внешнее прерывание, для этого HSI.0 не должно быть разрешено как вход HSI	HSI.0 Pin (INT04)	4
HSI Data Available	IOC1.7 = 0, INT_MASK.2 = 1 и данные из FIFO переносятся в регистр хранения. Это прерывание показывает, что по крайней мере одно HSI событие произошло и готово для обработки	HSI Data Available (INT02)	2
* 15 приоритет является высшим, 0 – низшим. ** Шестое значение в FIFO может вызывать одно из прерываний – или HSI Data Available (INT02), или HSI FIFO Full (INT14), но нельзя его сконфигурировать для обоих прерываний одновременно.			

9.2.3.3. Разрешение и Запрещение Выводов HSI

Можно индивидуально разрешить или запретить функции HSI для каждого вывода путем установки или очистки соответствующего бита в IOC0 регистре (смотри рисунок 9.7).

Установка соответствующего бита в регистре IOC0 связывает определитель перепада на HSI со счетчиком-делителем на 8. Очистка бита отсоединяет вывод от HSI логики, но не от регистра HSI_STATUS (смотрите рисунок 9.7). Запрещение действия входов HSI не запрещает и не препятствует использование каждого вывода для реализации дополнительных функций.

9.2.3.4. Назначение Дополнительных функций для HSI выводов

Можно запрограммировать HSI выводы для обеспечения альтернативных функций. Таблица 9.6 показывает дополнительные функции для каждого вывода HSI. Колонка 3 описывает как выбрать дополнительную функцию.

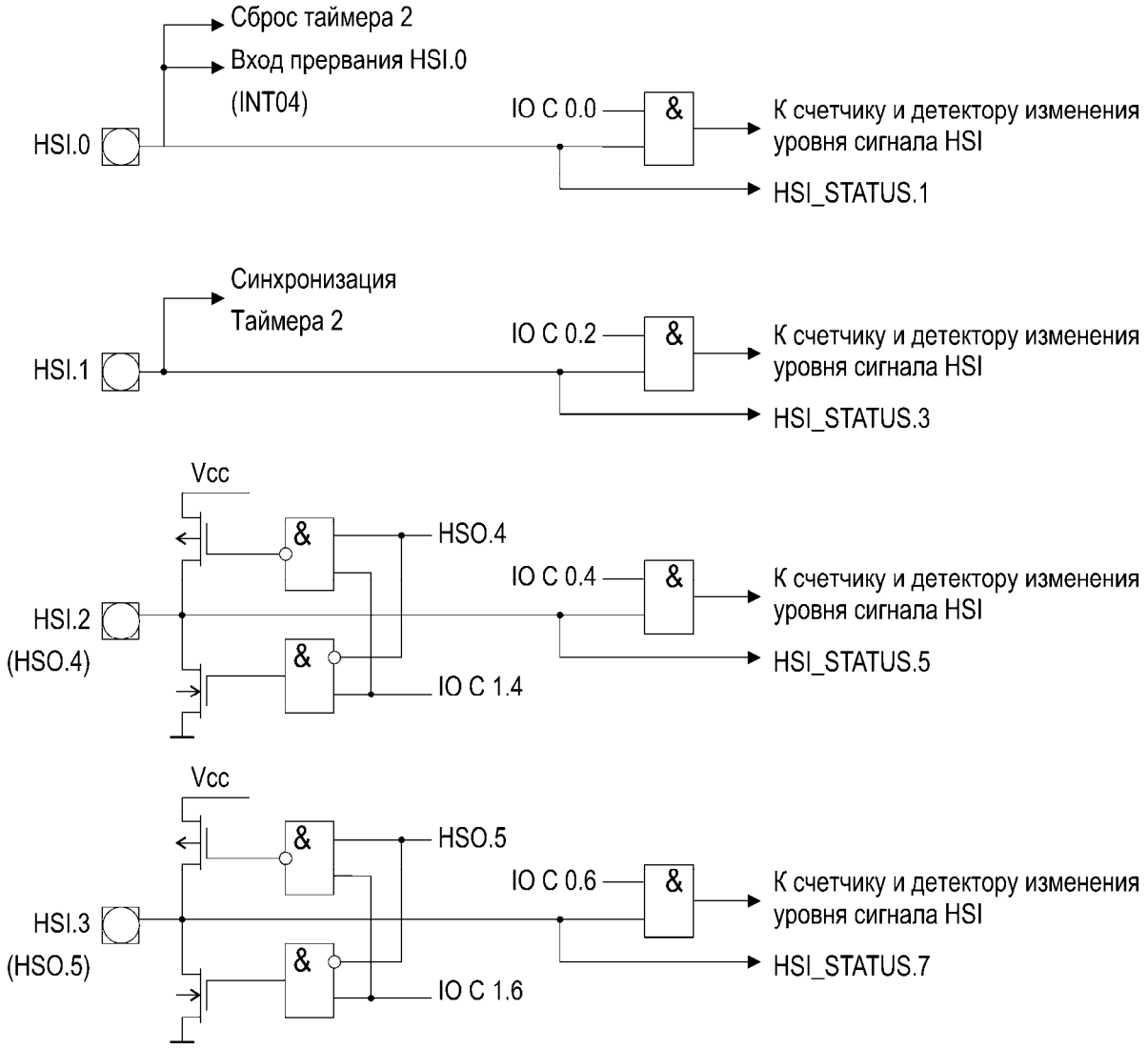


Рисунок 9.7 – Возможные входы HSI

Таблица 9.6 – Дополнительные функции для HSI выводов

Функциональное имя HSI	Дополнительные функции	Выбирается
1	2	3
HSI.0	Прерывание HSI.0 Pin (INT04)	Установка INT_MASK1.4 разрешает прерывание HSI.0 Pin (INT04). Рекомендуется удерживать HSI.0 в течение более чем 2 тактов, чтобы гарантировать захват сигнала
	Источник сброса таймера 2	Установка IOC0.5 выбирает HSI.0 как источник сброса таймера 2. Когда разрешено, нарастающий фронт на HSI.0 сбрасывает таймер 2
HSI.1	Источник тактов таймера 2	Установка IOC0.7 и IOC3.0 выбирает вывод HSI.1 как источник тактового сигнала таймера 2
HSI.2	HSO.4	Установка IOC1.4 разрешает действие выхода HSO.4. Примечание – HSI и HSO действия могут быть активны в одно и то же время
HSI.3	HSO.5	Установка IOC1.6 разрешает действие выхода HSO.5. Примечательно то, что HSI и HSO действия могут быть активны в одно и то же время

9.3 Модуль высокоскоростного вывода HSO

Модуль высокоскоростного вывода (HSO) вызывает события в определенное время с использованием таймера 1 или таймера 2 как базовых. Эти программируемые события включают: запуск АЦП, перезапуск таймера 2, генерацию до 4-х программных временных задержек и установку или очистку одного или более HSO выходов (HSO.0 – HSO.5). HSO модуль сохраняет до 8 ожидаемых событий и соответствующее им время в блоке памяти CONTENT_ADDRESSABLE MEMORY (CAM). На рисунке 9.8 показана блок-схема модуля HSO.

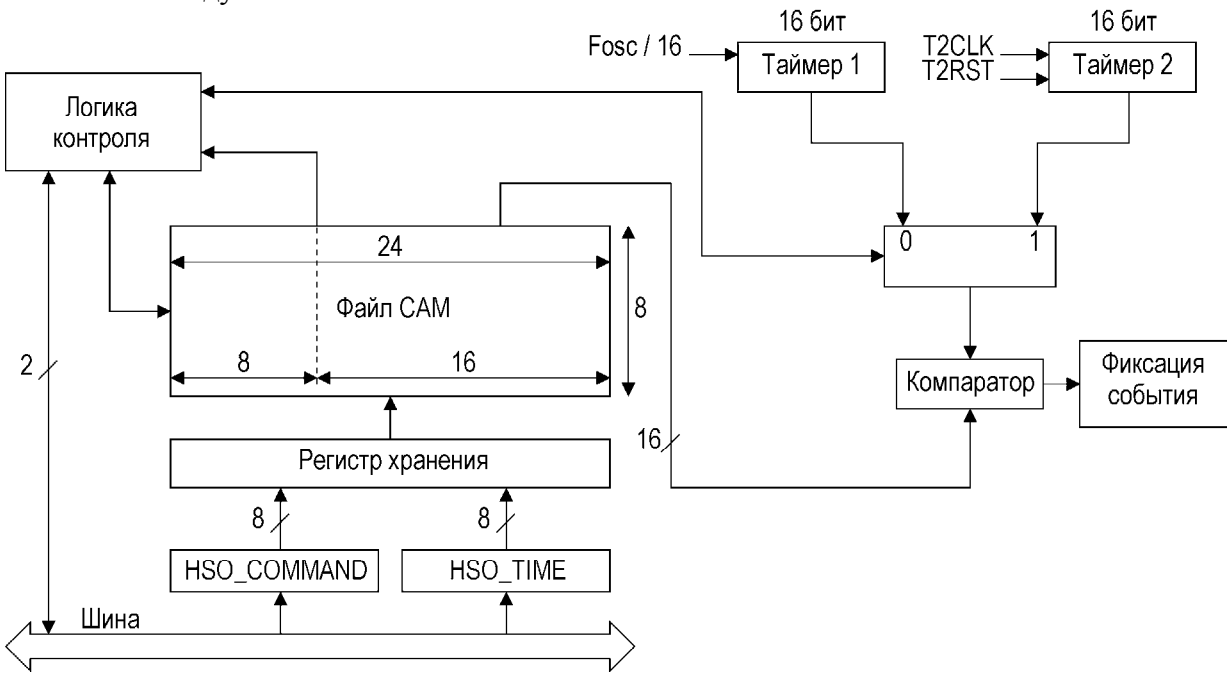


Рисунок 9.8 – Блок-схема модуля HSO

9.3.1 Функционирование HSO

CAM – основной компонент модуля HSO. Эта область памяти сохраняет до 8 команд. Каждая запись в CAM файле содержит 24 бита. 16 бит загружаются из HSO_Time регистра, чтобы определить время действия, и 8 бит загружаются из HSO_COMMAND регистра, чтобы определить тип действия, требуется ли прерывание и какой из таймеров, 1 или 2, является базовым.

Примечание – Когда таймер 2 используется в модуле HSO как базовый, он должен работать в нормальном режиме (а не в режиме fast increment) и считать только в одном направлении. Более подробно смотрите раздел 9.1.6.

HSO сравнивает все восемь записей в CAM файле со значением таймера до фиксации события. Необходимо затратить один такт, чтобы сравнить каждую запись, таким образом 8 тактов необходимо для полной проверки CAM файла. HSO фиксирует событие, когда значение базового таймера соответствует запрограммированному значению времени.

HSO может вызвать 15 различных событий, которые классифицируются как внешние или внутренние. Внешние – события, которые проявляются на одном или более выходах HSO; внутренние – события, которые устанавливают программные таймеры, сбрасывают таймер 2 или запускают АЦП. Все эти события устанавливают флаги, которые могут вызывать определенные прерывания. Внешние события вырабатывают прерывание High Speed Output (INT03); внутренние события вырабатывают прерывание Software Timer (INT05).

Четыре программных таймера имеют возможность вырабатывать прерывания в заранее определенное время; в общем случае они используются для вызова программ

обслуживания прерывания, которые должны повторяться с определенными интервалами. В определенное время HSO устанавливает флаг HSO в IOS1 регистре; если прерывания разрешены, это также вызовет прерывание Software Timer. Если больше чем один программный таймер срабатывает в пределах одного временного фрагмента, устанавливается несколько битов состояний. Подпрограмма обслуживания прерывания может проверить регистр IOS1, чтобы определить, какой программный таймер (или таймеры) вызвали прерывание.

9.3.2. Программирование HSO Модуля

В таблице 9.7 приведены регистры, которые влияют на работу и функции модуля HSO.

Таблица 9.7 – Регистры управления и состояния модуля HSO

Мнемоника регистра	Имя регистра	Описание
HSO_COMMAND	Регистр команд HSO	Определяет, что за событие или события произойдут в HSO в определенное время
HSO_TIME	Время HSO	Определяет время, в которое HSO команда будет выполнена
INT_MASK	Маска прерывания	Разрешает или запрещает HSO прерывания
IOC1	Регистр 1 управления входом/выходом	Разрешает или запрещает HSO.4 и HSO.5 как выходы
IOC2	Регистр 2 управления входом/выходом	Разрешает и запрещает команды просмотра HSO области CAM, может также очистить HSO CAM
IOS0	Регистр 0 состояния входа/выхода	Показывает текущее состояние HSO выводов, фиксирующего регистра и CAM файла; запись в биты может устанавливать или очищать соответствующие HSO выводы
IOS1	Регистр 1 состояния входа/выхода	Содержит флаги, которые показывают какие внутренние события вызваны прерыванием
IOS2	Регистр 2 состояния входа/выхода	Содержит флаги, которые показывают какие внешние HSO события имели место

9.3.2.1. Программирование HSO Событий

Содержимое регистров HSO_TIME и HSO_COMMAND вместе описывают каждое HSO событие. HSO_TIME регистр определяет время выполнения команды HSO. Биты 0-3 регистра HSO_COMMAND содержат команду, которая определяет, какое событие или события HSO будут вызваны в определенное время (смотрите рисунок 9.9). Таблица 9.8 описывает коды HSO команд. HSO_COMMAND регистр также определяет, будет ли событие вызывать прерывание, выполняет ли команда установку или очистку соответствующего вывода (выводов), выбирает базовый таймер для HSO-команд и управляет сохранением команд в CAM или удалением после выполнения (смотрите таблицу 9.9).

Таблица 9.8 – Кодировка CMD_TAG

В шестнадцатиричной системе	Мнемоника команд	Описание
00	HSO0	Включить High-Speed Output 0
01	HSO1	Включить High-Speed Output 1
02	HSO2	Включить High-Speed Output 2
03	HSO3	Включить High-Speed Output 3
04	HSO4	Включить High-Speed Output 4
05	HSO5	Включить High-Speed Output 5
06	HSO01*	Включить High-Speed Outputs 0 и 1
07	HSO23*	Включить High-Speed Outputs 2 и 3
08	SWT0	Программирование программного таймера 0
09	SWT1	Программирование программного таймера 1
0A	SWT2	Программирование программного таймера 2
0B	SWT3	Программирование программного таймера 3
0C	HSOALL*	Включение High-Speed Outputs 0, 1, 2, 3, 4, 5
0D		Резерв; не используется
0E	T2RST	Сброс Timer2
0F	A-D	Запуск АЦП
* В этой конфигурации два или более выводов устанавливаются или очищаются одновременно.		

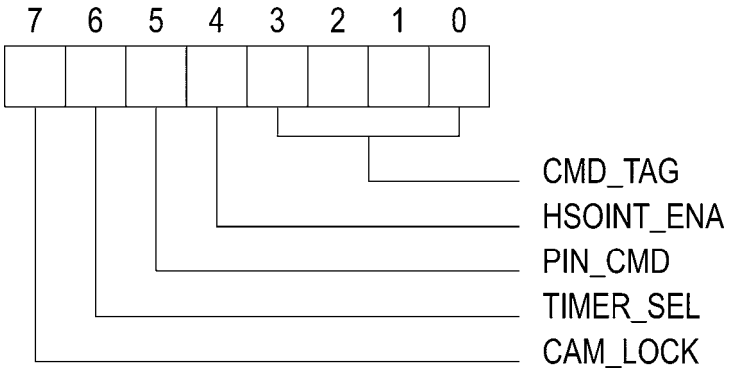


Рисунок 9.9 – Регистр HSO_COMMAND

Таблица 9.9 – Описание битов регистра HSO_COMMAND

Номера битов	Мнемоника битов	Имя бита	Описание
0-3	CMD_TAG	команды HSO	определяют, какие события или событие будут (смотрите таблицу 9.8)
4	HSOINT_ENA	разрешение/ запрещение HSO прерывания	определяет, вырабатывает ли HSO событие прерывание: 1 – вырабатывает прерывание, 0 – не вырабатывает прерывание. Когда этот бит установлен, запрограммированное на вывод HSO событие вырабатывает High-Speed Output прерывание (INT03, 2006h), а внутреннее событие вырабатывает Software Timer прерывание (INT05, 200Ah). Когда прерывание произошло, биты состояния HSO события в IOS1 и IOS2 регистрах должны быть проанализированы, чтобы определить, какое событие вызвало прерывание
5	PIN_CMD	установка/ очистка выбранного вывода HSO	устанавливается или сбрасывается определенный вывод или выводы: 1 – устанавливает вывод(ы), 0 – сбрасывает вывод(ы)
6	TIMER_SEL	выбирает Timer1 / Timer2	выбирает опорный таймер для HSO команд: 1 – выбирается таймер 2, 0 – выбирается таймер 1
7	CAM_LOCK	блокировка команды в CAM	когда IOS2.6 установлен (фиксация команды разрешена), этот бит управляет, должна ли HSO-команда зафиксироваться в CAM или очиститься после выполнения, 1 – фиксация команды в CAM, 0 – стереть команду из CAM после выполнения. Запись “1” в IOS2.7 очищает всё содержимое (фиксированное и нет) из CAM, как это делает сброс кристалла

Примечание – Перед программированием HSO_COMMAND и HSO_TIME регистров, необходимо прочитать или IOS0.7, или IOS0.6, чтобы убедиться, что регистр захвата HSO пустой. Если IOS0.7 очищен, то регистр захвата пуст. Если IOS0.6 очищен, то регистр захвата пуст и, по крайней мере, один CAM регистр пуст. Запись в HSO_TIME, когда регистр захвата не пустой, вызывает потерю предыдущего значения фиксирующего регистра.

Для того, чтобы ввести команду в CAM-файл, первым записывается HSO_COMMAND регистр, чтобы описать событие, и затем записывается HSO_TIME регистр, чтобы определить относительное время, в которое событие произойдет. Запись в HSO_TIME регистры загружают HSO регистр захвата. Команда сохраняется в фиксирующем регистре до тех пор, пока невозможна запись в CAM файл, при освобождении команда переписывается в CAM-файл. Это может потребовать 8 тактов для перемещения команды из фиксирующего регистра в CAM файл, таким образом всегда необходимо иметь по крайней мере 8 тактов между последовательными записями в регистры HSO_COMMAND и HSO_TIME. В противном случае команда в фиксирующем регистре может быть потеряна.

Примечание – Команда в фиксирующем регистре не будет выполняться, даже если ее время подошло. Команды для их выполнения должны находиться в CAM-файле.

Значение в HSO_TIME может быть или определенным значением таймера, или значением смещения относительно текущего значения таймера. Необходимо использовать стандартные команды загрузки, чтобы запрограммировать определенное время, иначе используется трех-операндная команда ADD, чтобы определить значение, которое является смещением относительно текущего значения таймера.

```
LDB     HSO_COMMAND, #what_to_do          ; Loads command
ADD     HSO_TIME,TIMER1, #when_to_do_it   ; ADD_3op
```

Для обеспечения правильной синхронизации минимальное время, которое может быть загружено в таймер, должно быть равно current_time_value+2. Меньшее значение может быть причиной того, что событие осуществится на 65536 тактов позже, чем требуется. Это ограничение касается обоих таймеров: таймера 1 и таймера 2.

Следует внимательно записывать в регистры HSO_COMMAND и HSO_TIME. Прерывание может осуществиться после записи команд в HSO_COMMAND и перед записью значения времени HSO_TIME. Если программа обработки прерывания также записывает в эти регистры, команда основной программы в CAM-файле перекрывается, поэтому команда основной программы никогда не выполнится. Запрещение прерываний перед записью в HSO-модуль решает эту проблему (смотрите раздел 6).

9.3.2.2 Разрешение Прерываний от HSO

Установка бита HSOINT_ENA (HSO_COMMAND.4) позволяет HSO событию генерировать прерывание. HSO модуль может генерировать два различных прерывания: прерывание High-Speed Output (INT03) и прерывание Software Timer (INT05).

Внешние события вырабатывают прерывание High-Speed Output, если оно разрешено (INT_MASK.3 установлен). Биты состояния IOS2.0 – IOS2.5 связаны с прерыванием High-Speed Output. Биты состояния IOS1.0 – IOS1.3 используются с прерыванием Software Timer для команд программного таймера, а IOS2.6 – IOS2.7 для сброса таймера 2 и старта A/D преобразования, соответственно.

Когда прерывание произошло, необходимо считать статусные биты в IOS1 и IOS2 регистрах, чтобы определить, какое событие вызвало прерывание. Чтение IOS1 регистра очищает биты 0 – 5; чтение IOS2 регистра очищает все биты. По этой причине мы рекомендуем копировать содержимое регистров в теневые регистры и затем выполнять команды тестирования битов, такие как JBC или JBS в теневых регистрах.

9.3.2.3 Сохранение Записей в CAM

LOCK_ENA бит в IOC2 регистре (IOC2.6) разрешает или запрещает фиксацию записи с командой. Когда этот бит установлен, CAM_LOCK бит (HSO_COMMAND.7) управляет либо сохранением команд в CAM, либо их удалением после выполнения. Когда CAM_LOCK установлен, команда остается в CAM после выполнения. Это дает возможность просто генерировать периодические события или сигналы. Обнуленный CAM_LOCK вызывает удаление HSO команды из CAM после выполнения.

Фиксация команд обеспечивает возможность программирования периодических или повторяющихся событий. Одним из наиболее полезных применений является использование сохранения команд для генерации повторяющихся импульсов с задаваемой длительностью (PWM) с минимумом программных затрат (смотрите подраздел 9.3.3). Сохранение команды A/D преобразования вызывает многократное выполнение A/D преобразования. Когда таймер 2 используется как базовый в HSO, зафиксированная команда T2RST может вырабатывать периодические события; события со значением HSO_TIME меньшим, чем значение сброса таймера 2, повторяются с каждым новым сбросом таймера 2. Сохранение команд программного таймера может организовать повторяющийся вызов программных задач; программы обслуживания

прерывания могут выполнять задачи без использования других команд программного таймера HSO.

9.3.2.4 Удаление Команды из CAM

Три события могут удалить команду из CAM:

- если значение базового таймера соответствует запрограммированному времени и не запрограммировано сохранение события;
- сброс устройства HSO;
- установлен CAM_CLR бит (IOC2.7).

Чтобы удалить сохраненные события, необходимо установить CAM_CLR бит (IOC2.7) или сбросить устройство. Любое из этих действий очистит весь в CAM-файл.

9.3.2.5 Отмена События

Можно отменить внешнее событие, записав противоположное событие с тем же временем в CAM. Например, если команда устанавливает HSO.1, когда TIMER1=1234, следует записать команду, которая очищает HSO.1, когда TIMER1=1234, тогда HSO.1 не изменится. Однако обе команды остаются в CAM до тех пор, пока или TIMER1 не будет равен 1234 (если сохранение незапрограммировано), или устройство не будет сброшено, или не будет установлен CAM_CLR бит (IOC2.7).

Можно отменить внутреннее событие только установкой CAM_CLR бита (IOC2.7) или сбросом устройства.

9.3.2.6 Разрешение Выводов HSO.4 и HSO.5

HSO.4 и HSO.5 выходы мультиплексируются с HSI.2 и HSI.3 входами соответственно. Эти выводы могут быть разрешены для той и другой функций (смотрите рисунок 9.7). Установка HSO4_ENA бита (IOC1.4) разрешает HSO.4 как выход; установка HSO5_ENA бита (IOC1.6) разрешает HSO.5.

9.3.2.7 Использование Выводов HSO.0-HSO.5 как выходов

Выходы HSO могут дать 6 дополнительных выходных контактов, если не используются функции HSO. Чтобы использовать выходы HSO как стандартные выходы, необходимо разрешить их использование как выходов, и затем установить или очистить их записью в IOS0 регистр через HWindow 15.

9.3.3 Использование модуля HSO как широтно-импульсного модулятора (PWM)

И HSO модуль, и PWM модуль могут генерировать последовательность прямоугольных импульсов, в которой изменяются период и скважность импульсов. Используя соответствующие внешние компоненты, можно реализовать высокоточный восьмиразрядный цифро-аналоговый преобразователь с использованием либо HSO, либо PWM выходов (смотрите раздел 11).

HSO модуль может генерировать либо одиночную PWM последовательность, либо повторяющуюся PWM последовательность. Одиночная последовательность требует программирования периода и времени сброса (clear time) сигнала и двух HSO команд: установить активное состояние вывода и сбросить его. Для управления может быть использован таймер 1 или таймер 2.

Генерация повторяющейся последовательности производится аналогичным способом, для управления используется таймер 2. Для генерации повторяющейся последовательности требуется запрограммировать период для таймера 2, время установки (set time) (PWMx_ST) и время сброса (PWMx_CT) для каждого выходного контакта и нескольких HSO команд: одну для сброса таймера 2, несколько для установки выводов и несколько для их сброса.

В приведенном ниже примере показан способ генерации нескольких PWM сигналов (ШИМ последовательностей) с использованием HSO.0, HSO.1 и HSO.2 как PWM выходов и таймера 2 как базового. Рисунок 9.10 показывает три результирующие PWM последовательности.

```
SET_0:
    LDB     HSO_COMMAND, #11100000B     ; set HSO.0 when ;
    LD      HSO_TIME, #PWM0_ST R0 R0    timer2 = pwm0_st ;
    SKIP                                wait 4 state times ;
    SKIP                                wait 4 state times
                HSO_COMMAND, #11100001B
SET_1:          HSO_TIME, #PWM1_ST R0 R0
    LDB                                 ; set HSO.1 when ;
    LD                                  timer2 = pwm1_st ;
    SKIP        HSO_COMMAND, #11100010B wait 4 state times ;
    SKIP        HSO_TIME, #PWM2_ST ZERO_REG
                ZERO_REG                wait 4 state times
SET_2:
    LDB                                 ; set HSO.2 when ;
    LD          HSO_COMMAND, #11000000B timer2 = pwm2_st ;
    SKIP        HSO_TIME, #PWM0_ST ZERO_REG
    SKIP        ZERO_REG                wait 4 state times ;
                                        wait 4 state times
CLEAR_0:
    LDB         HSO_COMMAND, #11000001B ; clear HSO.0 when ;
    LD          HSO_TIME, #PWM1_ST ZERO_REG
    SKIP        ZERO_REG                timer2 = pwm0_ct ;
    SKIP                                wait 4 state times ;
                                        wait 4 state times
CLEAR_1:
    LDB                                 ; clear HSO.1 when ;
    LD                                  timer2 = pwm1_ct ;
    SKIP                                wait 4 state times ;
    SKIP                                wait 4 state times
```

9.3.4 Синхронизация Выхода HSO

HSO выводы синхронизируются базовым таймером. Все внешние выводы HSO изменяются при достижении таймером заданного значения сразу после того, как таймер увеличился на единицу. При внутренней синхронизации таймер изменяет значение каждый восьмой такт во время фазы 1. При внешней HSO вывод будет изменяться сразу после спада импульса CLKOUT и будет стабилен при его нарастающем фронте. Информация от HSO может быть зафиксирована при нарастающем фронте CLKOUT. Внутреннее HSO событие происходит, когда инкрементируется базовый таймер.

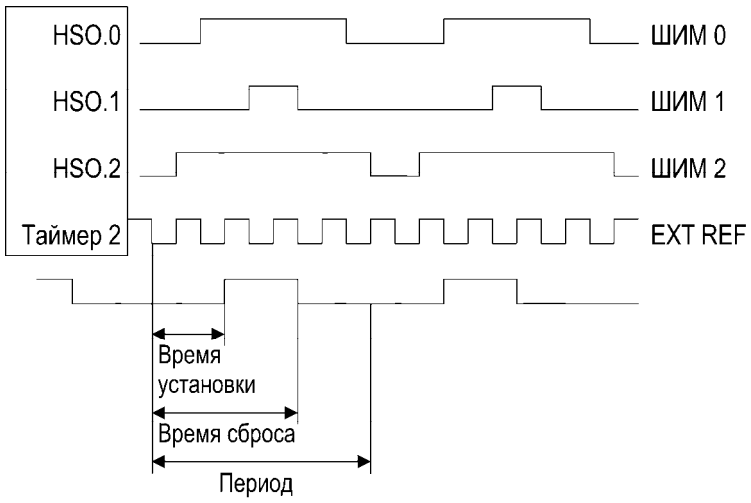


Рисунок 9.10 – Пример PWM последовательностей

10 Аналого-цифровой преобразователь

Микроконтроллер 1874BE05T не содержит АЦП.

В этом разделе приведен обзор процесса аналого-цифрового (А/Ц) преобразования и описывается, как программировать А/Ц преобразователь, считывать результаты преобразования и взаимодействовать с внешними схемами. Здесь также описаны передаточные функции и источники ошибок.

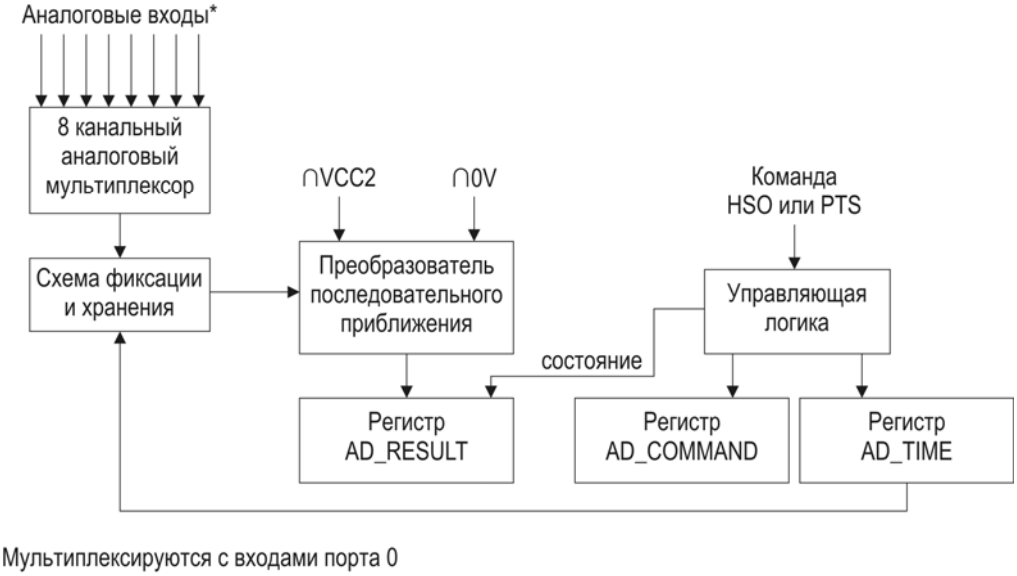


Рисунок 10.1 – Блок-схема аналого-цифрового преобразователя

10.1 Обзор функций АЦП

Модуль АЦП (показанный на рисунке 10.1) преобразует значение сигнала на аналоговом входе в 8-и или 10-ти битное цифровое значение. Основными частями модуля АЦП являются:

- восемь аналоговых входов (ACH0-ACH7);
- 8-канальный мультиплексор (8-Channel Analog Mux), выбирающий один из 8 входных каналов;
- схема фиксации и хранения (Sample and Hold);
- 10-битный преобразователь последовательного приближения (Successive Approximation Converter);
- регистр AD_COMMAND, который управляет операциями преобразования через управляющую логику (Control Logic);
- двухбайтный регистр AD_RESULT, в котором содержится результат преобразования и информация о состоянии;
- регистр AD_TIME, который содержит время фиксации и время преобразования.

После получения АЦП команды START_AD пройдет один машинный цикл до начала сравнения с эталоном. Во время этой задержки регистр последовательного приближения сбрасывается, и через мультиплексор выбирается нужный канал. После заданной задержки выход мультиплексора соединяется с эталонной емкостью и остается подсоединенным заранее определенное время. После окончания времени запоминания вход эталонной емкости отсоединяется от мультиплексора, так что изменения на выводе не влияют на запомненную величину во время преобразования. Затем компаратор автоподстраивается на нуль, и преобразование начинается.

АЦП использует алгоритм последовательного приближения. Аппаратура преобразователя содержит 256-резисторную матрицу, компаратор, накапливающие конденсаторы и 10-битный регистр последовательного приближения (SAR) с логикой управления процессом. Резистивная матрица обеспечивает шаг в 20 мВ (U_{CC2} = 5,12 В), в то время как цепочка емкостей создает шаг в 5 мВ внутри 20 мВ напряжения матрицы.

Таким образом, имеется 1024 внутренних уровней напряжения для сравнения с аналоговым входом для генерации 10-битного конечного результата.

Преобразование последовательным приближением осуществляет сравнение заданных напряжений матрицы с величиной на аналоговом входе методом половинного деления, чтобы обеспечить лучшее приближение. Эталонное напряжение, равное 1/2 полной шкалы, проверяется первым. Это соответствует 10-битному результату, в котором старший значащий бит нуль, а остальные – единицы (0111.1111.11b). Если на аналоговом входе значение меньше, чем значение тестирующего напряжения, бит 10 в SAR останется нулевым, и вход будет сравниваться с новым тестовым напряжением – 1/4 полной шкалы (0011.1111.11b). Если значение на входе больше тестового напряжения, устанавливается 9 бит SAR. 8 бит затем очищается для следующего теста (0101.1111.11b). Такой бинарный поиск продолжается, пока 10 (или 8) испытаний не будут произведены, к этому времени результат преобразования будет располагаться в SAR, откуда он может быть считан программно. Результат эквивалентен отношению входного напряжения к напряжению аналогового источника. Если отношение равно 1,00, то результат равен напряжению этого источника.

Примечание – Для достижения максимальной точности результата рекомендуется опорное напряжение U_{CC2} = 5,12 В. Не рекомендуется превышать это значение U_{CC2}.

10.2 Программирование АЦП

Программируются следующие параметры АЦП:

- 8- или 10-битное преобразование;
- выбор входного канала;
- время запоминания и преобразования;
- прерывание по завершению преобразования;
- режим совместимости с 1874ВЕ36;
- непосредственное преобразование или запуск преобразования модулем HSO или PTS.

Таблица 10.1 – Регистры управления и состояния А/Ц преобразователя

Мнемоника	Название	Описание
AD_COMMAND	Регистр A/D команд	Регистр выбирает номер канала для A/D преобразования, определяет, начинается ли A/D преобразование немедленно или по HSO команде и выбирает 8- или 10-битный режим преобразования.
AD_RESULT	Регистр результата АЦП	Регистр содержит 2 байта. Старший байт содержит 8 старших бит преобразования. Младший байт показывает номер канала АЦП, подлежащего преобразованию, индицирует, идет ли преобразование и содержит 2 младших бита при 10-битном режиме.
AD_TIME	Регистр времени A/D преобразования	Регистр определяет время запоминания и преобразования, если это разрешено IOC2 регистром.
IOC2	Регистр 2 управления	Регистр определяет, чем управляется время A/D преобразования: либо регистром AD_TIME, либо эмуляцией режимов быстрого и нормального преобразований 1874ВЕ36.
INT_MASK	Маскирование прерываний	Регистр разрешает/запрещает прерывание по окончанию A/D преобразования (INT01, 2002h). Установка бита 1 разрешает это прерывание; очистка бита 1 запрещает (маскирует) прерывание.
INT_PEND	Ожидание прерываний	Регистр показывает, что произошло окончание A/D преобразования, если бит 1 установлен. Бит 1 очищается при переходе по вектору 2002h.

10.2.1 Выбор времени запоминания и преобразования

Два параметра: время запоминания и время преобразования - управляют общим временем А/Ц преобразования. Время запоминания – это время, в тёчение которого аналоговый вход подключен к запоминающей емкости. Если это время слишком короткое, то запоминающая емкость еще полностью не зарядится. Если это время слишком велико, напряжение на входе может изменяться, что вызовет ошибку преобразования. Время преобразования – это время, требуемое для преобразования величины напряжения, сохраненной на запоминающей емкости в цифровое значение. Время преобразования должно быть достаточно длинным, чтобы компаратор и цепи измерения смогли сброситься и замерить напряжение. Слишком длинное время преобразования позволяет запоминающей емкости разрядиться и уменьшает точность.

Время запоминания и преобразования определяется либо по умолчанию, в заранее заданном режиме совместимости с 1874BE36, либо определяется AD_TIME регистром. Очистка бита AD_TIME_ENA (IOC2.3) разрешает преобразование в режиме совместимости с 1874BE36. 1874BE36 – в совместимом режиме AD_FAST бит (IOC2.4) управляет временем запоминания и преобразования. Медленный режим 1874BE36 использует 15 тактов для запоминания и 158 тактов для преобразования входного значения. Быстрый режим использует 8 тактов для запоминания и 91 такт для преобразования. Когда используется быстрый или медленный режим, преобразование этих тактов в микросекунды производится по значениям времени Tsam и Tconv в спецификации на устройство, которые зависят от частоты работы процессора.

Установка бита AD_TIME_ENA (IOC2.3) разрешает AD_TIME регистр (показана на рисунке 10.2). AD_TIME регистр программирует скорость А/Ц преобразования. Разрешающая способность и тактовая частота влияют на скорость преобразования. Используя спецификацию Tsam и Tconv, определяют соответствующую величину для времени запоминания (SAM) и преобразования (CONV), иначе результат преобразования может быть ошибочным.

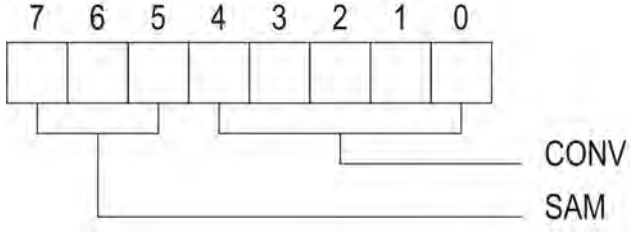


Рисунок 10.2 - Регистр AD_TIME

Необходимо использовать следующие формулы для определения оптимальных значений SAM и CONV:

SAM = (Tsam × Fosc - 2) / 8,
CONV = (Tconv × Fosc - 3) / (2 * B) - 1,

где SAM = от 1 до 7, CONV = от 2 до 31,
Tsam - время запоминания в мкс из спецификации,
Tconv - время преобразования в мкс из спецификации,
Fosc - частота XTAL1 в МГц,
B - число бит при преобразовании (8 или 10).

Когда SAM и CONV вычислены, их записывают в AD_TIME регистр. Не записывайте в этот регистр в процессе преобразования: результат будет непредсказуем.

10.2.2 Программирование IOC2 Регистра

Регистр IOC2 (смотрите рисунок 10.3) выбирает, устанавливается ли время А/Ц преобразования регистром AD_TIME или битом AD_FAST (IOC2.4). Установка бита

AD_TIME_ENA выбирает режим, совместимый с 1874BE36. В этом режиме бит AD_FAST разрешает или запрещает предварительное деление частоты для полной совместимости с быстрым или медленным режимами 1874BE36. Очистка AD_FAST разрешает медленный режим (15 тактов для запоминания, 158 тактов для полного преобразования), установка – быстрый режим (8 тактов для запоминания, 91 такт для преобразования).

Примечание – Изменение режима во время преобразования может вызвать непредсказуемый результат.

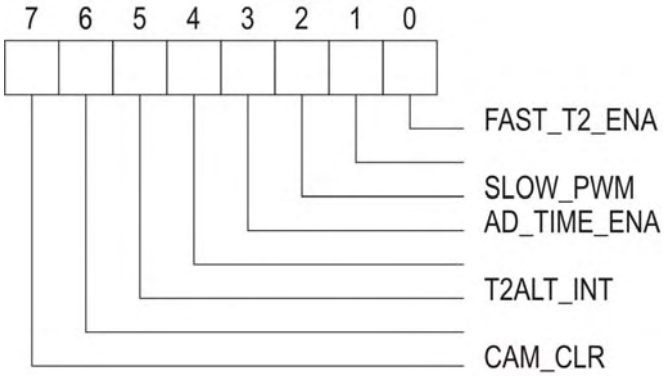


Рисунок 10.3 - Регистр IOC2

10.2.3 Программирование регистра ADCOMMAND

Регистр AD_COMMAND, показанный на рисунке 10.4, выбирает канал преобразования, 8 или 10 битное преобразование и определяет либо начало преобразования, либо использование HSO модуля для запуска. Номер канала записан в AD_CHAN_SEL (биты 0-2). Очистка бита AD_MODE (бит 4) определяет выбор 10-битного преобразования, а установка бита AD_MODE - выбор 8-битного преобразования. Резервные биты 5, 6, 7 должны быть очищены.

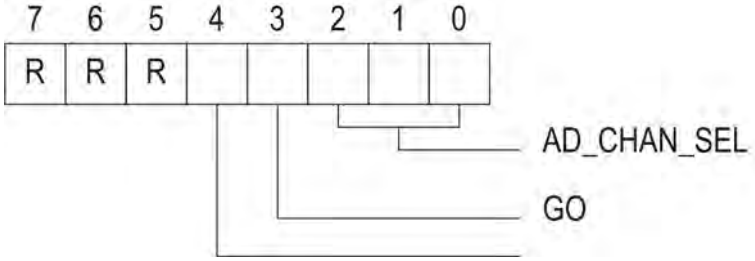


Рисунок 10.4 – Регистр AD_COMMAND

Установка бита GO (бит 3) вызывает немедленное начало преобразования. Преобразование, инициированное установкой бита GO, начнется в пределах трех тактов после исполнения команды.

Очистка бита GO позволяет модулю HSO инициировать преобразование в будущем. Модуль HSO включает преобразование при выполнении команды CMD_TAG = 0Fh. Процесс преобразования начнется, когда таймер 1 достигнет запрограммированного значения. Это помогает зафиксировать более точную форму сигнала, поскольку последовательное запоминание, осуществляемое через задержки, определяемые таймером 1, происходит с разбросом ±50 нс (тактовый сигнал на XTAL1 считается стабильным).

Следующий пример показывает, как использовать HSO модуль для запуска преобразования:

```
LDB AD_COMMAND, #XXXX1XXXB   – установка бита GO
LDB HSO_COMMAND, #START_AD   – загрузка команды начала
                               преобразования в
                               HSO_COMMAND ADD
HSO_TIME, TIMER1, #TIME_DELAY – сложение 16-битной константы
                               с текущим значением TIMER1
                               результат в HSO_TIME
```

HSO регистр допускает загрузку в CAM множества команд START_AD. Однако следует быть осторожным при записи команд в HSO. Если таймер 1 увеличится и запустит новое преобразование до завершения предыдущего, текущее преобразование будет прервано и начнется новое.

PTS может начать новое преобразование в ответ на прерывание A/D Conversion Complete.

10.3 Чтение результатов преобразования

Регистр AD_RESULT содержит 2 байта (смотрите рисунок 10.5). Старший байт (03h) содержит 8 старших значащих битов А/Ц преобразования. Младший байт (02h) показывает номер А/Ц канала, который используется для преобразования, производится ли в данный момент преобразование, а также содержит 2 младших значащих бита 10-битного А/Ц преобразования. Регистр AD_RESULT очищается с началом нового преобразования; поэтому для предотвращения потери информации оба байта необходимо считать перед началом нового преобразования.

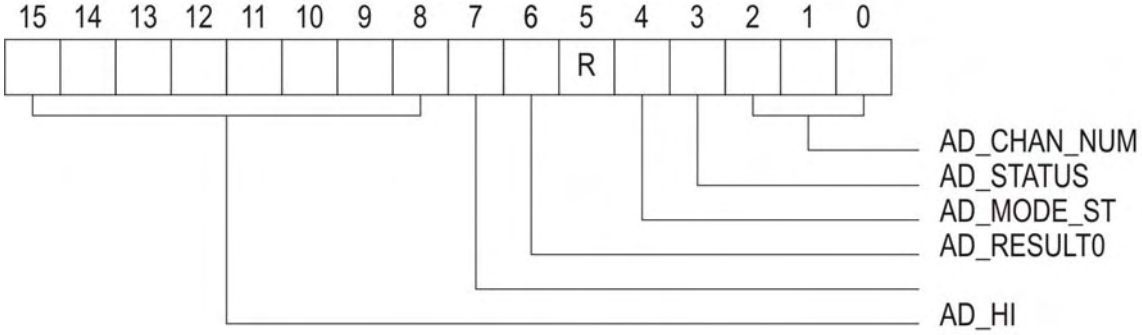


Рисунок 10.5 – Регистр AD_RESULT

АЦП генерирует прерывание ADC Conversion Complete (INT01), когда он завершает преобразование. Адрес вектора прерывания - 2002h, прерывание разрешается установкой бита AD_MASK (INT_MASK.1). Бит AD_PEND (INT_PEND.1) устанавливается, когда преобразование закончено, даже если прерывание замаскировано. Этот бит остается установленным, пока не произойдет прерывание по вектору 2002h или регистр не очистится.

Другой способ определения статуса преобразования - это опрос AD_RESULT регистра. Бит AD_STATUS (AD_RESULT.3) устанавливается во время процесса преобразования. С момента начала преобразования может пройти 8 тактов до установки бита, поэтому не начинайте опрос в первые 8 тактов после начала преобразования.

10.4 Интерфейс с АЦ-преобразователем

Внешняя схема согласования с аналоговым входом определяется конкретным применением и влияет на характеристики преобразователя. При разработке внешней схемы факторы, такие как входной ток утечки, эталонная емкость, сопротивление последовательных переключателей с заданной емкостью для входных выводов должны быть правильно выбраны. На рисунке 10.6 приведена упрощенная внешняя схема. Внешняя схема на входе должна быть способна заряжать эталонную емкость (Cs) через последовательное входное сопротивление (R1) до точного напряжения, задаваемого постоянным током утечки (I_{IL2}). Обычно Cs около 3 пФ, R1 примерно 1 кОм, I_{IL2} приблизительно 3 мкА.

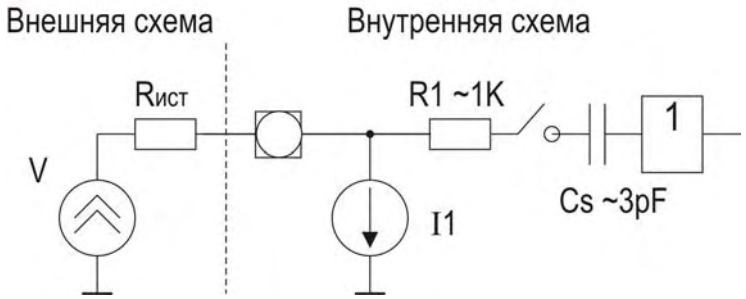


Рисунок 10.6 – Упрощенная схема фиксации напряжения в АЦП

Внешняя схема с сопротивлением источника (Rsource) не более 1 кОм может поддерживать входное напряжение с точностью ±0,6 точности младшего значащего разряда LSB (1,0 кОм × 3,0 мкА = 3,0 мВ), задаваемое постоянным входным током утечки (I_{IL2}). Сопротивление источника около 2 кОм может вызвать ошибку, по крайней мере, в один значащий разряд, обусловленную понижением напряжения, вызванным током утечки 3 мкА.

Ток утечки I_{IL2} обычно меньше максимального значения. Типовое значение тока утечки может быть превышено изменением сопротивления источника до появления ошибки в один LSB. Кроме того, источник с высоким сопротивлением может мешать внутренней эталонной емкости полностью заряжаться в течение заданного промежутка времени. Эта ошибка может быть вычислена с помощью следующей формулы:

ERROR(LSBs) = [exp(–Tsam/RC)] × 1024,

где Tsam – заданное время, мкс;

R = Rsource + R1, Ом;

C = Cs, мкФ.

Воздействие этой ошибки может быть уменьшено подключением внешней емкости (Cext) к выводу ∩0V. Внешний сигнал будет заряжать Cext до напряжения источника. При фиксации значения сигнала в канале малая часть заряда, хранимого на Cext, будет передаваться на внутреннюю эталонную емкость. Перераспределение заряда между Cs и Cext вызывает понижение точности. Если величина Cext не менее 0,005 мкФ, максимальная ошибка будет 0,6 LSB.

Размещение внешнего конденсатора на каждом аналоговом входе также будет уменьшать чувствительность к шуму, так как комбинация конденсатора с последовательным резистором во внешней цепи образует низкочастотный фильтр (смотрите рисунок 10.7). На практике следует включать небольшой последовательный резистор перед внешней емкостью на аналоговом выводе и выбрать значение емкости конденсатора больше, чем значение, определяемое частотой сигнала, который необходимо преобразовать. Это обеспечивает низкочастотный фильтр на входе и, кроме того, резистор будет ограничивать входной ток при условии перегрузки по напряжению.

Пример входной цепи АЦП, показанный на рисунке 10.7, обеспечивает ограничительную защиту от перегрузки по напряжению на аналоговом входе. Если входное напряжение принимает отрицательное значение, диод D2 будет способствовать ограничению на уровне 0,8 В по постоянному току (dcV). Вывод имеет максимально допустимый нижний уровень напряжения минус 0,5 × dcV; поэтому падение напряжения на резисторе 270 Ом должно быть около 0,3 × dcV.

Примечание – При подключении любого аналогового входа к напряжению более, чем на 0,5 В превышающего ∩0V или ∩VCC2, включается входное защитное устройство. Появляющийся при этом ток во внутренней цепи источника опорного напряжения существенно уменьшает точность АЦП на всех каналах.

Необходим тщательный анализ возможности применения схемы, приведенной на рисунке 10.7, для использования ее в конкретной разработке.

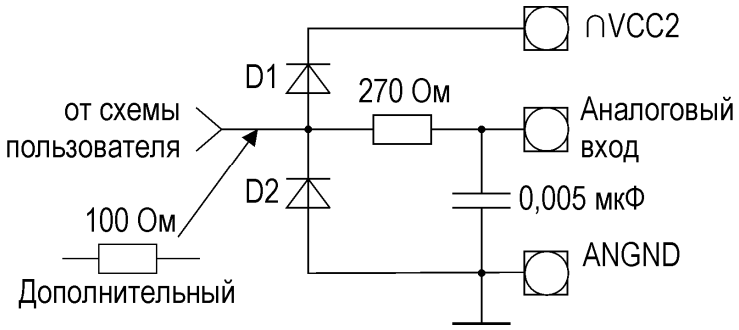


Рисунок 10.7 – Рекомендуемая входная схема для АЦП

10.4.1 Аналоговая земля и источник опорного напряжения

Пульсации опорного напряжения сильно влияют на абсолютную точность преобразования. По этой причине рекомендуется соединять вывод ∩0V с выводом #0V как можно ближе к устройству, используя минимальную длину связей. При внешних помехах настоятельно рекомендуется использовать отдельную шину ∩0V, которая соединяется с ∩0V в единственной точке как можно ближе к устройству. Следует также использовать фильтрующий конденсатор между ∩VCC2 и ∩0V. Уровень напряжения на ∩0V может быть в пределах 0,1 В относительно #0V. ∩VCC2 должен быть стабильным и использоваться только для АЦП.

Напряжение питание U_{CC2} возможно в пределах от 4,0 В до 5,5 В и источник питания должен обеспечивать ток 10 мА. Большие отрицательные пики токов на выводе ∩0V относительно #0V могут вызвать насыщение в аналоговой цепи. Это является дополнительной причиной, из-за которой следует внимательно заземлять устройство.

Аналоговое опорное напряжение является положительным напряжением, с помощью которого происходит сравнение при A/D преобразовании. Это также питание и порта 0, если АЦП не будет использоваться. Если высокая точность не требуется, ∩VCC2 может быть соединен с #VCC1. Если точность важна, ∩VCC2 должен быть очень стабильным. Один из путей выполнения этого – использование прецизионных источников напряжения или отдельного стабилизатора напряжения. Эти устройства должны быть связаны с ∩0V, а не с #0V, чтобы гарантировать, что ∩VCC2 имеет отдельную землю ∩0V, а не соединяется с #0V.

10.4.2 Совместное использование аналоговых и цифровых входов

Порт 0 может быть использован для аналоговых и цифровых сигналов в одно время. Однако, когда порт 0 читается (чтение по адресу 0Eh), небольшой шум может быть наведен в аналоговую цепь. По этой причине необходимо гарантировать, что не идет процесс A/D преобразования, когда требуется выполнить чтение выводов порта 0 .

10.5 Передаточная функция и источник ошибок АЦП

Результат преобразования – это 8-ми или 10-ти битное представление отношения входного напряжения к опорному. Для вычисления 10-ти битного результата используется следующая формула:

RESULT (10-бит.) = $1023 \times (U_I - U_{\cap 0V}) / (U_{CC2} - U_{\cap 0V})$

Получается ступенчатая передаточная функция, когда выходной код отображает значение входного напряжения. Полученный цифровой код может являться информацией о простом соотношении наряжений, обеспечивать информацию об абсолютном значении напряжения на входе или его относительном изменении.

Чем больше требований предъявляется к преобразователю, тем более важно полное понимание операций преобразования. Для простого применения достаточно знания абсолютной погрешности преобразователя. Однако чтобы реализовать следящую обратную связь с аналоговыми входами, требуется полное понимание операций преобразования и ошибок АЦП.

Во многих разработках целесообразнее работать с абсолютным значением на входе, чем определять, что произошло изменение. Такое происходит, когда преобразовывается монотонный и непрерывный сигнал.

Если преобразователь является 10-разрядным, то для взаимнооднозначного соответствия диапазон изменения входного сигнала должен быть разбит на 1024 уровня. Каждая величина интервала такого разбиения представляет собой значение аналоговой величины, на которое отличаются уровни входного сигнала, представляемые двумя соседними кодовыми комбинациями. Этот интервал называют также величиной младшего значащего разряда (LSB). Все аналоговые величины внутри заданного интервала разбиения представляются одним и тем же цифровым кодом, которому обычно ставят в соответствие значение аналоговой переменной в средней точке интервала, называемое пороговым уровнем. Тот факт, что входной сигнал может отличаться от порогового уровня на величину, достигающую ±0,5 LSB, не отличаясь при этом по кодовому представлению, означает, что любому процессу A/D преобразования присуща неопределенность (ошибка дискретизации, равная ±0,5 LSB). Ее влияние можно уменьшить увеличением числа разрядов в выходном коде преобразователя.

Передаточная функция идеального 3-х битного АЦП приведена на рисунке 10.8.

Заметим, что идеальная характеристика имеет следующие особенности:

- появление только младшего разряда кода происходит при входном напряжении 0,5 LSB;

- максимальное значение кода достигается, когда входное напряжение равно ∩VCC2 – 1,5 LSB;

- величина интервала (кванта) равна 1 LSB.

Эта характеристика не учитывает смещение нуля, ошибок усиления и нелинейностей. Другими словами, это идеальное преобразование.

Внутренние ошибки А/Ц преобразования следующие: ошибка дискретизации, ошибка смещения нуля, ошибка усиления, дифференциальная и интегральная нелинейности. Все эти ошибки являются ошибками передаточной функции АЦП.

Кроме того, должны учитываться температурные коэффициенты, флюктуация #VCC1, утечка в эталонной емкости, пробой мультиплексора, взаимное влияние каналов и случайный шум. Обычно абсолютная ошибка включает в себя общую сумму ошибок и все расхождения между реальным процессом преобразования и идеальным. Однако отдельные составляющие ошибки важны в конкретных разработках.

Реальная характеристика 3-битного преобразователя является не идеальной. Если идеальную характеристику сопоставить с реальной, то реальная характеристика, как показано на рисунке 10.9, имеет как ошибки в расположении минимального и максимального кодового значения напряжения, так и в величине интервала (кванта). Смещение минимального значения кода от идеального называется ошибкой нулевого смещения, смещение максимального кодового значения называется ошибкой на полном диапазоне (full-scale error). Отклонение величины интервала (кванта) от идеальной характеристики вызвано двумя типами ошибок: дифференциальной и интегральной нелинейностью.

Интегральная нелинейность определяется максимальным отклонением передаточной характеристики от идеальной прямолинейной характеристики при нулевых значениях смещения нуля и ошибки на полной шкале.

Дифференциальная нелинейность - это отклонение величины одного из квантов от его идеального значения. Заметим, что если дифференциальная нелинейность превышает 1 LSB, то в выходном сигнале может отсутствовать одна из кодовых комбинаций (выпадающий код). В десятибитном преобразователе идеальный шаг квантования 5 мВ (5,12 × ∩VCC2 / 1024). Если такой преобразователь будет иметь максимальную дифференциальную нелинейность 2 LSB (10 мВ), тогда максимальная величина кванта будет не более чем на 10 мВ больше идеальной или 15 мВ.

Рисунок 10.8 – Идеальная передаточная характеристика АЦП

Реальная величина кванта в этом преобразователе обычно изменяется от 2,5 мВ до 7,5 мВ. Ошибка дифференциальной нелинейности в ширине одного кванта компенсируется шириной другого кванта, так что сохраняется 1024 шага квантования.

Интегральная нелинейность вызывает в худшем случае отклонение реального кода от соответствующего кода идеальной характеристики. Интегральная нелинейность характеризует, насколько накопление дифференциальных нелинейностей по отдельным квантам может увеличить общий уход от линейной характеристики. Если ошибки дифференциальной нелинейности слишком велики, возможно выпадение кода или нарушение монотонности. Ни то, ни другое нежелательно в контурах управления. Преобразователь не имеет выпадающих кодов, если для каждого выходного кода существует входной диапазон напряжения, который характеризуется только этим кодом. Преобразователь является монотонным, если каждое последующее изменение кода вызвано изменением входного напряжения в том же направлении.

Дифференциальная нелинейность и интегральная нелинейность определяются при измерении ошибок общей (Terminal-Based) линейности. Базовую выходную (Terminal-Based) характеристику получают из реальной характеристики, преобразуя и масштабируя ее для уничтожения ошибок нулевого смещения и ошибки на полном диапазоне как показано на рисунке 10.10. Terminal-Based характеристика подобна реальной, в которой смещенит нуля и ошибка на полном диапазоне компенсированы подстройкой. На практике это достигается использованием входных цепей, которые обеспечивают подстройку усиления и компенсацию смещения нуля. К тому же, ∩VCC2 в МК может точно регулироваться в заданном диапазоне для уменьшения ошибки полного диапазона.

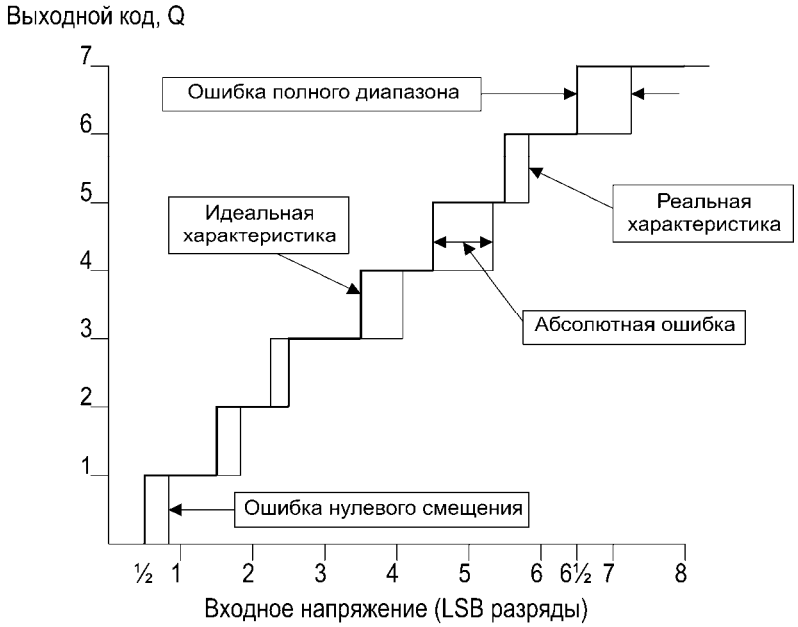


Рисунок 10.9 – Реальная и идеальная передаточные характеристики АЦП

Другие факторы, которые воздействуют на реальную АЦП систему, включают: температурный дрейф, импульсные помехи, взаимное влияние каналов мультиплексора и случайный шум. Обычно эти влияния незначительны.

Температурный дрейф - это темп, с которым изменяются характеристики с изменением температуры. Эти изменения отражаются в температурных коэффициентах.

Основные источники паразитных сигналов – это помехи по №VCC1, изменение входного сигнала в канале во время преобразования (после заряда эталонной емкости) и сигналы на каналах, не выбранных мультиплексором.

И наконец, входное сопротивление мультиплексора слегка отличается на разных каналах, что приводит к появлению при одинаковом входном сигнале различных значений на разных каналах и на том же канале при повторных преобразованиях. Различия в токах утечки в разных каналах и случайный шум вызывают ошибки повторного измерения.

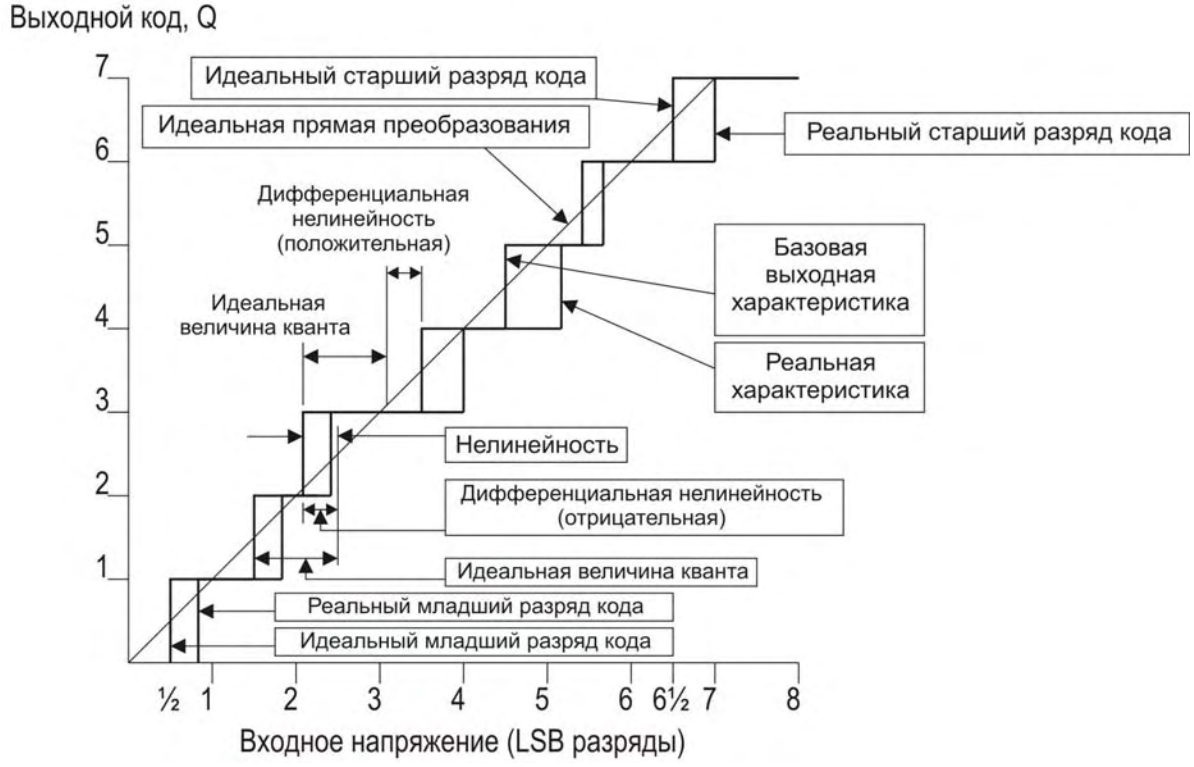


Рисунок 10.10 – Выходная передаточная характеристика АЦП

11 Широтно-импульсный модулятор (PWM)

МК имеют 3 модуля PWM (смотрите рисунок 11.1). Каждый выход обеспечивает выдачу импульса переменной длительности с фиксированной частотой. Эти выходы могут использоваться для управления двигателями, оптимально работающими с PWM импульсами без фильтрации, или же эти импульсы могут быть отфильтрованы для получения гладких аналоговых сигналов.

В этом разделе приведен функциональный обзор модулей PWM, описывается их программирование и дан пример схемы преобразования выходов PWM в аналоговый сигнал.

Смотрите приложение Б для получения информации о выходах PWM.

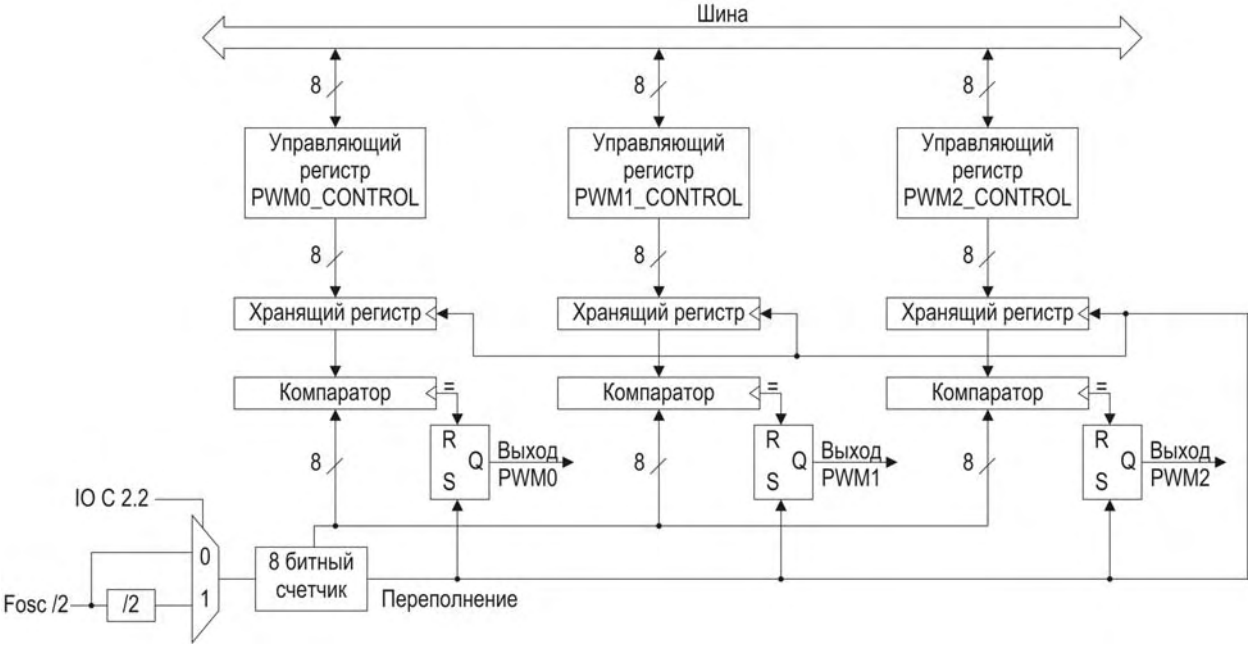


Рисунок 11.1 – Блок-схема модуля PWM

11.1 Функциональный обзор PWM

Модули PWM имеют следующие основные компоненты:

- 8-битный счетчик;
- предделитель на 2;
- три компаратора;
- три управляющих регистра (PWMx_CONTROL, где x равен 0, 1 или 2);
- три хранящих регистра (Holding Registers);
- три RS-триггера.

SLOW_PWM бит (IOC2.2) управляет периодом выходных импульсов, разрешая или запрещая работу предделителя на 2. Разрешение работы предделителя заставляет 8-битный счетчик увеличиваться каждые 2 такта; запрещение вызывает увеличение счетчика каждый такт.

Каждый управляющий регистр содержит 8-битное значение, загружаемое в регистр хранения, когда 8-битный счетчик переполняется.

Компараторы сравнивают содержимое регистров хранения со значением счетчика. Когда значение счетчика равно 0, выход PWMx находится в высоком уровне. Он остается в этом состоянии, пока значение счетчика не будет равно значению, находящемуся в регистре хранения, после сравнения выход переходит в низкий уровень. Загрузка в регистры PWMx_CONTROL значения 00h обеспечивает низкий уровень на выходе. При переполнении счетчика на выходе вновь высокий уровень. Рисунок 11.2 показывает типичную выходную диаграмму на выходе PWM.

Рисунок 11.2 – Форма выходного сигнала PWM

11.2 Программирование рабочего цикла PWM

Регистр PWMx_CONTROL, вместе с битом SLOW_PWM (IOC2.2), определяет длительность высокого уровня импульса на выходе PWMx, управляя рабочим цилом. Значение, записанное в регистр PWMx_CONTROL, может обеспечивать длительность от 0 до 255 тактов (то есть от 0 % до 99,6 % рабочего цикла). Установка IOC2.2 разрешает предварительный делитель PWM; полная длительность импульса становится 512 тактов, а значение регистра PWMx_CONTROL умножается на 4. Очистка IOC2.2 запрещает работу предделителя; полная длительность импульса – 256 тактов, а значение регистра PWMx_CONTROL умножается на 2. В таблице 11.1 приведены выходные частоты PWM.

Таблица 11.1 – Выходная частота PWM

IOC2.2	Частота на выводе XTAL1 (f_{OSC})			
	8 МГц	10 МГц	16 МГц	20 МГц
0	15,6 кГц	19,6 кГц	31,25 кГц	39,1 кГц
1	7,8 кГц	9,8 кГц	15,63 кГц	19,5 кГц

Следует использовать следующие формулы для вычисления периода PWM и времени, в течение которого выход PWM остается в высоком уровне. Заметим, что значение PWMx_CONTROL – 8-битное значение и f_{OSC} – частота XTAL1 в МГц.

	Предделитель запрещён (IOC2.2 = 0)	Предделитель разрешён (IOC2.2 = 1)
Период PWM (мкс) =	$\frac{512}{f_{OSC}}$	$\frac{1024}{f_{OSC}}$
Высокий PWM*x* (мкс) =	$\frac{PWMx\_CONTROL \times 2}{f_{OSC}}$	$\frac{PWMx\_CONTROL \times 4}{f_{OSC}}$

Например, если f_{OSC} равна 16 МГц, то период на выходе PWM – 32 мкс. Если PWM0_CONTROL равен 8Ah (десятичное 138) и IOC2.2 очищен, PWM0 сохраняет высокий уровень 17,25 мкс (и низкий – 14,8 мкс) из общих 32 мкс, в результате скважность приблизительно 54 %. Когда IOC2.2 установлен, то же самое значение в регистре PWM0_CONTROL будет формировать период в 64 мкс, и PWM0 будет сохранять высокий уровень 34,5 мкс (и низкий – 29,5 мкс), скважность при этом также около 54 %.

11.3 Разрешение выходов PWM

Каждый выход PWM мультиплексирован с выводом порта. В таблице 11.2 приведены альтернативные функции порта вместе с установками регистров, которые выбирают выходы PWM взамен функции порта.

Выбор выходных функций PWM1 или PWM2 делает невозможным выполнение функций квазидвунаправленного Порта 1 и разрешает выход с повышенной нагрузочной способностью.

Таблица 11.2 – Альтернативные функции выхода PWM

Выход PWM	Альтернативная функция порта	Выход PWM разрешён, если:
PWM0	P2.5	IOC1.0=1
PWM1	P1.3	IOC3.2=1
PWM2	P1.4	IOC3.3=1

11.4 Формирование аналоговых выходов

Модули PWM и модуль HSO вместе могут генерировать последовательность прямоугольных импульсов с изменяемыми скважностью и периодом. Фильтрация этого выхода обеспечивает плавный аналоговый сигнал. Чтобы сделать периодический сигнал в требуемых аналоговых пределах, необходимо сначала его буферизировать и затем сгладить. На рисунке 11.3 приведена типовая блок – схема, позволяющая получить плавный аналоговый сигнал. Следует использовать либо простую RC-цепь, либо активный фильтр.

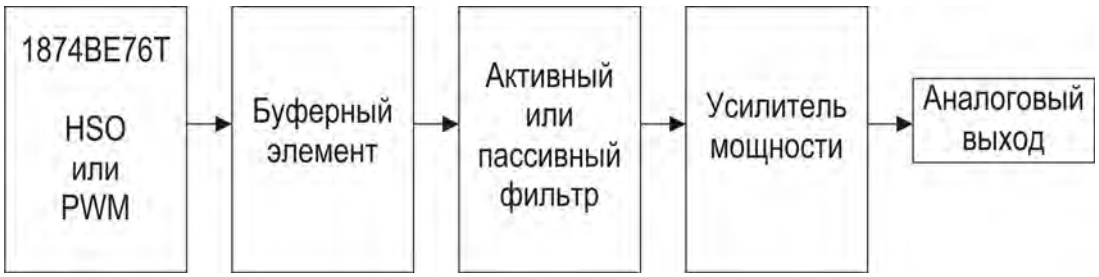


Рисунок 11.3 – Блок-схема буферизации цифроаналогового преобразователя

На рисунке 11.4 приведен пример схемы, используемой для слаботочного выхода (меньше, чем 100 мкА). Неоходимо учитывать температурный диапазон и уход питающего напряжения при выборе компонентов внешней цифро-аналоговой схемы. При соответствующем выборе компонентов может быть реализован 8-битный Ц/А преобразователь с использованием либо PWM, либо HSO выходов. Использование HSO может теоретически увеличить точность до 16 бит, однако проблемы, связанные с температурой и шумами, трудноразрешимы.

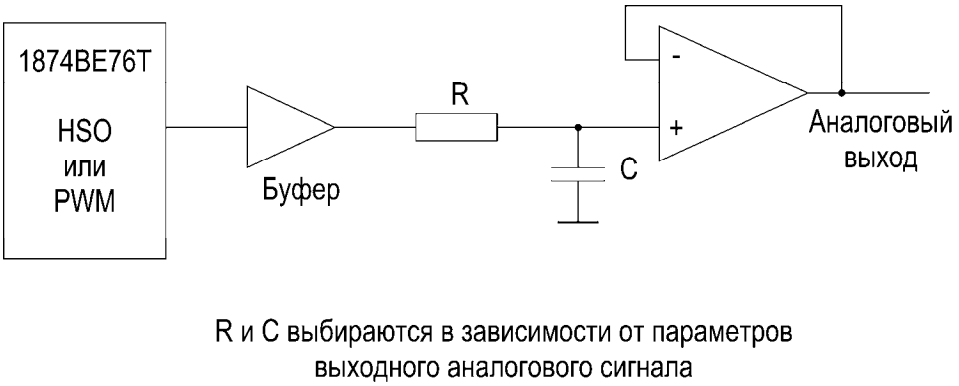


Рисунок 11.4 – Схема преобразования импульсов PWM в аналоговый сигнал

12 Специальные режимы работы

МК поддерживают два специальных рабочих режима: IDLE (холостой ход), POWERDOWN (низкое энергопотребление). Режимы IDLE и POWERDOWN снижают потребление энергии. Данный раздел описывает каждый режим, вход и выход из режима. Смотрите команды в приложении А, информацию о сигналах в приложении Б.

12.1 Режим IDLE

В режиме IDLE тактирование ЦПУ заморожено в состоянии логического нуля, но периферия тактируется и сигнал CLKOUT остается активным. ЦПУ останавливает выполнение команд, но внутреннее ОЗУ, таймеры/счетчики, последовательный порт и система прерываний продолжают функционировать. Потребление энергии уменьшается на 40 % по сравнению с нормальным режимом работы.

Выводы управления шиной (ALE, RD#, WR#, INST и BHE#) переведены в неактивное состояние. Если Порты 3 и 4 использовались как порты ввода-вывода, они будут сохранять значения, записанные в их защелках. Если эти порты использовались как шина адрес/данные, то выводы будут "плавающими".

Если таймер WDT разрешен, то он продолжает работать в IDLE режиме. Устройство должно выходить из IDLE режима каждые 64К такта, чтобы очистить регистр WATCHDOG (0Ah), иначе таймер WDT сбросит устройство.

12.1.1 Вход в IDLE Режим

Чтобы войти в IDLE режим, следует выполнить команду IDLPD #1.

12.1.2 Выход из режима IDLE

Прерывания или аппаратный сброс выведут устройство из IDLE режима. Из-за того, что вся периферия остается активной в IDLE режиме, любой разрешенный источник прерывания может выработать прерывание. Когда произойдет прерывание, ЦПУ выходит из IDLE режима и начинает выполнять соответствующую подпрограмму обслуживания прерывания. После завершения подпрограммы обслуживания прерывания ЦПУ выбирает и затем выполняет команду, которая следует за командой IDLPD #1.

12.2 Режим POWERDOWN

Режим POWERDOWN переводит МК в состояние с очень низким потреблением энергии. Если значение U_{CC} сохраняется, то регистры специальных функций (SFRs) и регистры ОЗУ сохраняют данные. Со всех периферийных устройств питание снимается, все сигналы внутреннего тактирования замораживаются в состоянии логического нуля и генератор отключается. Энергопотребление падает до микроваттного уровня. Значение I_{CC} снижается до тока утечки устройства.

Во время режима POWERDOWN, внутренний генератор и схема тактирования запрещены. Выводы управления шиной (ALE, RD#, WR#, INST и BHE#) переведены в неактивное состояние. Все выходы сохраняют значение, находящееся в их защелках. Если Порты 3 и 4 использовались как порты ввода-вывода (EA# = 0), их выводы будут удерживать последнее выведенное значение. Если Порты 3 и 4 использовались как шина адрес/данные (EA# = 1), их выводы будут "плавающими".

На рисунке 12.1 приведена упрощенная схема снижения энергопотребления в режиме POWERDOWN. Таблица 12.1 описывает внутренние сигналы, показанные на рисунке 12.1. Команда IDLPD #2 устанавливает Q1, выключая при этом внутреннее фазированное тактирование, и сбрасывает Q2, выключая при этом внутренний генератор.

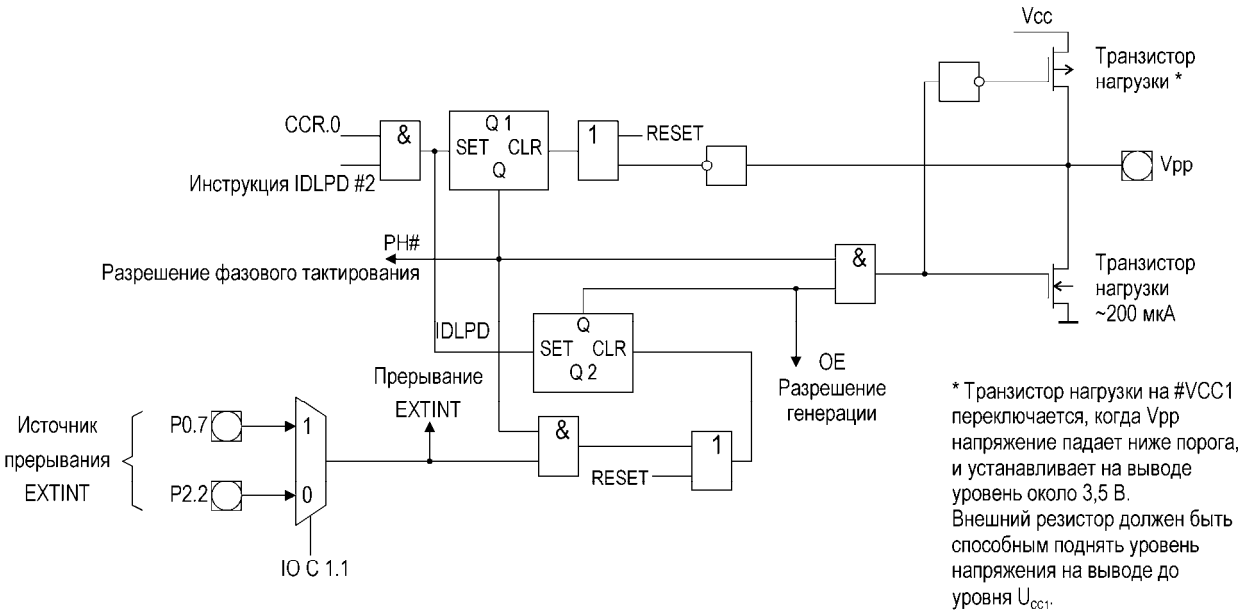


Рисунок 12.1 – Упрощенная схема для режима POWERDOWN

Таблица 12.1 – Внутренние сигналы для режима POWERDOWN

Имя сигнала	Описание
IDLPD	IDLE/POWERDOWN. Этот сигнал переводит устройство в режим POWERDOWN. Он блокирует сигналы: разрешения фазового тактирования (PH#) и разрешения генерации (OE). Команда IDLPD #2 вырабатывает этот сигнал.
OE	Разрешение генерации. Этот сигнал разрешает генерацию для режима нормальной работы. Во время режима POWERDOWN, он блокирует внутренний генератор.
PH#	Разрешение фазированного тактирования. Этот сигнал разрешает внутреннее фазированное тактирование для режима нормальной работы. Во время режима POWERDOWN он блокирует внутреннее фазированное тактирование.
RESET	Внутренний сброс. Этот сигнал является инвертированным внешним сигналом RESET#.

12.2.1 Запрещение режима POWERDOWN

Режим POWERDOWN запрещается, когда бит PD (CCR.0) очищен. В CCR загружается байт конфигурации кристалла (CCB ячейка 2018h) при сбросе устройства.

12.2.2 Переход в Режим POWERDOWN

После перехода в режим POWERDOWN завершаются следующие задачи:

- Завершается обмен через последовательный порт. Когда устройство выходит из режима POWERDOWN, последовательный порт активизируется и возобновит обмен, если была потеря данных, то возможна передача или прием неправильных данных.
- Завершаются все аналоговые преобразования. Если переход в режим POWERDOWN произошел во время преобразования, то результат может быть неправильным;
- Если таймер WDT был разрешен, то необходимо очистить регистр WATCHDOG перед выполнением команды перехода в режим POWERDOWN. Это гарантирует, что устройство сможет выйти из режима POWERDOWN "чисто", иначе WDT может сбросить МК до стабилизации тактового генератора. (WDT не может

сбросить устройство во время режима POWERDOWN, потому что генератор остановлен).

После завершения этих задач необходимо выполнить команду IDLPD #2 для перехода в режим POWERDOWN. Команда IDLPD #2 устанавливает Q1, выключая внутреннее фазированное тактирование, и очищает Q2, блокируя внутренний генератор (смотрите рисунок 12.1).

Примечание – Вывод EXTINT должен удерживаться на низком уровне, пока устройство находится в режиме POWERDOWN.

12.2.3 Выход из Режима POWERDOWN

Выход из режима POWERDOWN происходит путем подачи на вывод V_{PP} низкого уровня при установленном сигнале RESET# или EXTINT.

12.2.3.1 Подключение V_{PP} к низкому уровню

Наиболее быстрый способ для выхода из режима POWERDOWN – подать низкий уровень на V_{PP} как минимум на 50 нс. Это очистит Q1 (смотрите рисунок 12.1), который разрешает внутреннее фазированное тактирование, но не установит Q2. Внутренний генератор остается заблокированным, таким образом, этот способ можно использовать, если тактовый сигнал является внешним.

12.2.3.2 Установка RESET#

Другой путь для выхода из режима POWERDOWN – установить сигнал RESET#. RESET# должен удерживаться в низком уровне до тех пор, пока генератор не стабилизируется. Схема генератора должна характеризоваться определенным временем наступления стабильной генерации. Когда используется внешнее тактирование, RESET# должен удерживаться в низком уровне, по крайней мере, один такт. Когда RESET# устанавливается, внутренний сигнал сброса очищает Q1 и устанавливает Q2, одновременно разрешая и внутреннее фазированное тактирование, и внутренний генератор (смотрите рисунок 12.1).

12.2.3.3 Установка EXTINT

Последний способ для выхода из режима POWERDOWN – установка сигнала EXTINT не менее чем на 50 нс. EXTINT обычно вход, фиксирующий прерывание. Однако в режиме POWERDOWN он используется как вход, чувствительный к уровню. EXTINT прерывание необходимо запретить для вывода устройства из режима POWERDOWN. Бит EXTINT_SRC (ЮС1.1) определяет источник EXTINT. Установка ЮС1.1 выбирает P0.7 как источник; очистка ЮС1.1 выбирает P2.2 как источник. Рисунок 12.2 показывает последовательность входа и выхода из режима POWERDOWN, когда используется EXTINT для выхода из режима.

Если сигнал EXTINT будет использоваться для выхода из режима POWERDOWN, рекомендуется подключить внешнюю RC цепь, показанную на рисунке 12.3 к выводу V_{PP}. Разрядка конденсатора дает задержку, которая позволяет генератору стабилизироваться до разрешения внутреннего фазированного тактирования.

Когда сигнал EXTINT принимается, Q2 устанавливается и подключает высокоомную нагрузку на корпус. Эта высокоомная нагрузка заставляет внешний конденсатор (C1) начать разряжаться с током разряда в диапазоне от 100 мкА до 200 мкА. Когда напряжение V_{PP} упадет ниже порогового (около 2,5 В), триггер Шмидта очистит Q1, который разрешит фазированное тактирование (PH#=0), и устройство возобновит выполнение программы.

Транзистор нагрузки на U_{CC1} переключается, когда напряжение на V_{PP} падает ниже порога, и быстро устанавливает на выводе уровень около 3,5 В. Нагрузка на U_{CC1} становится неэффективной и внешний резистор (R1) поднимает уровень напряжения на выводе до уровня U_{CC1} (смотрите время восстановления на рисунке 12.4). Постоянная

времени дает экспоненциально изменяющуюся кривую: если C1 = 1 мкФ и R1 = 1 МОм, время восстановления будет порядка нескольких секунд.

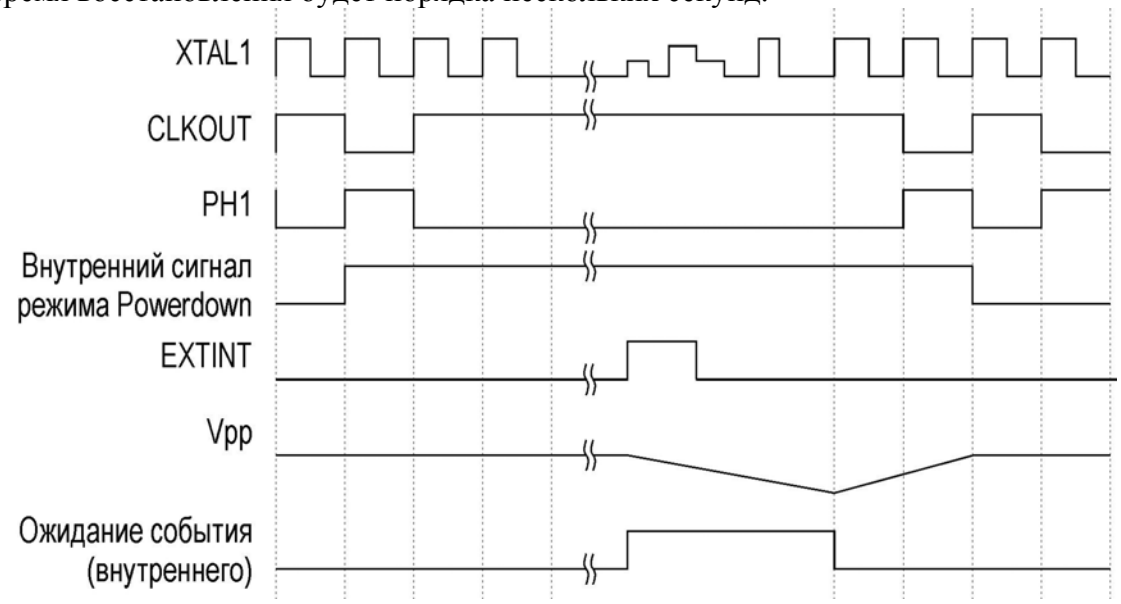


Рисунок 12.2 – Последовательность входа и выхода из режима POWERDOWN

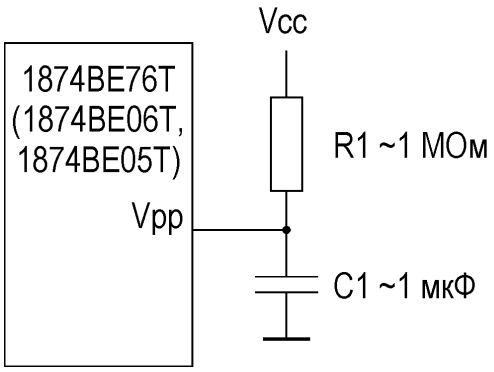


Рисунок 12.3 – Внешняя RC цепь на V_{PP}

12.2.3.3.1 Выбор R1 и C1

Выбор элементов необходим для обеспечения определенного времени разряда, чтобы дать возможность схеме внутреннего генератора стабилизироваться. Из-за того, что много факторов могут влиять на требуемое время разряда, Вы должны всегда ориентироваться на худшие условия работы Вашего устройства при выполнении этой операции. Выбирайте резистор, который не будет влиять на ток разряда. Значение R1 в пределах от 200 кОм до 1 МОм удовлетворяет этим условиям.

Когда выбираете емкость, определите наихудшее время разряда, необходимое для стабилизации генератора, затем используйте формулу для вычисления приближенного значения C1:

Cl=(Tdis*I)/dV,

где C1 - значение емкости в фарадах,
Tdis - наихудшее время разряда в секундах,
I - ток разряда в амперах,
dV - пороговое напряжение в вольтах.

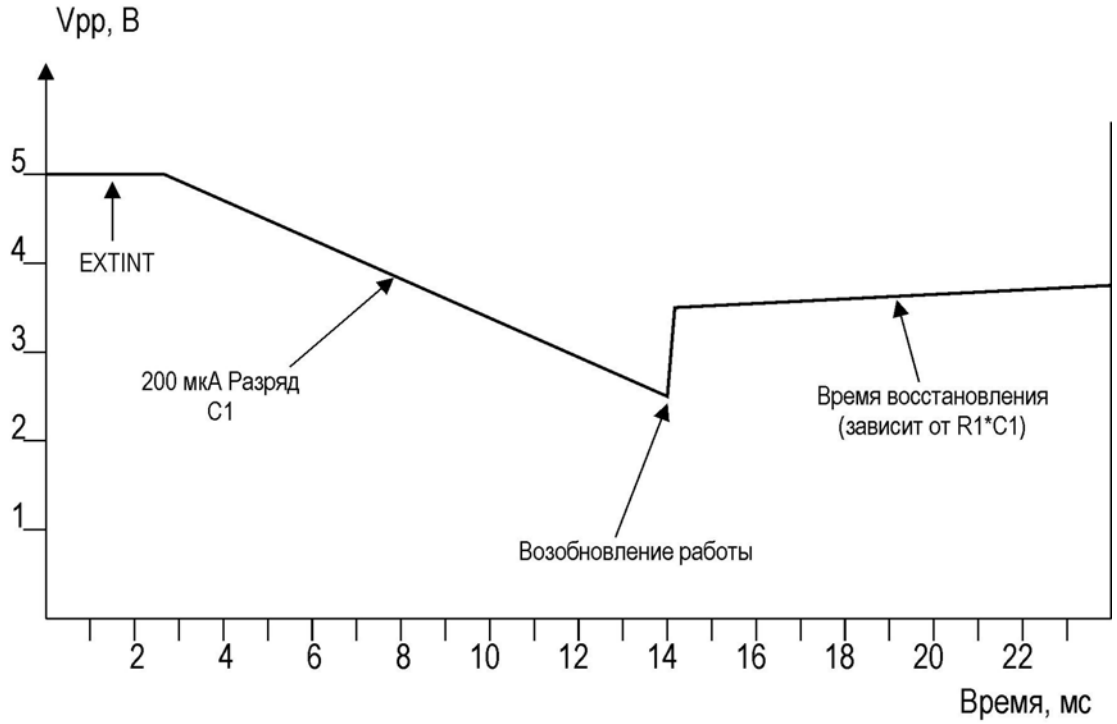


Рисунок 12.4 – Напряжения на выводе V_{PP} при выходе из режима POWERDOWN

Примечание – Если происходит повторный вход и выход из режима POWERDOWN до того, как C1 зарядится до U_{CC1}, то это потребует меньше времени для снижения напряжения до порогового напряжения (dV будет меньше). Устройству потребуется меньше времени для выхода из режима POWERDOWN.

Например, примем, что генератору необходимо минимум 12,5 мс для стабилизации (Tdis=12,5 мс), dV=2,5 В и ток разряда 200 мкА.

Минимальная емкость C1 = 1 мкФ:

C1=(0,0125*0,0002)/2,5=1 мкФ

Когда используется внешний генератор, значение C1 может быть гораздо меньше, и это позволяет быстро восстанавливаться после режима POWERDOWN. Например, конденсатор в 100 пФ разряжается за 1,25 мкс:

Tdis=(Cl*dV)/I=(1,0e–10*2,5)/0,0002=1,25 мкс.

12.3 Вход в режимы, зарезервированные для тестирования

Когда на время сброса вывод PWM0 (P2.5) удерживается в высоком уровне, (в комбинации с другими выводами), то вызываются режимы, зарезервированные для тестирования. Чтобы избежать входа в эти режимы, не следует удерживать PWM0 в состоянии высокого уровня во время сброса.

13 Интерфейс внешней памяти

МК может взаимодействовать с различными устройствами памяти. Он поддерживает или фиксированную шину разрядностью 8 бит, или динамическую шину разрядностью 8/16 бит. Существует внутренний контроль готовности для медленных внешних устройств памяти READY и несколько режимов управления шиной. Эта особенность обеспечивает большую гибкость, когда идет взаимодействие с устройствами внешней памяти. Данный раздел описывает сигналы внешней памяти и регистры, которые управляют интерфейсом с внешней памятью, а также рассказывает о различных режимах и особенностях внешней шины.

13.1 Сигналы интерфейса внешней памяти

Таблица 13.1 описывает сигналы интерфейса внешней памяти. Многие из этих сигналов имеют альтернативные функции. Графа “Выбор” показывает биты регистра и значения, которые выбирают функцию из первой колонки.

Таблица 13.1 – Сигналы интерфейса внешней памяти

Имя	Альтернативная функция	Чем выбирается	Тип	Описание
1	2	3	4	5
AD0-AD7 AD8-AD15	P3.0-P3.7 P4.0-P4.7	Шина доступа к внешней памяти	I/O	Системная шина: адрес/данные. Эти выводы обеспечивают мультиплексированную шину адрес/данные. В течение адресной фазы шинного цикла адресные биты 0-15 выводятся на шину и могут быть зафиксированы с помощью ALE или ADV#. 8- или 16-битные данные передаются в течение определенной фазы. Когда доступ к шине прекращается, эти выводы возвращаются к первоначальным функциям порта ввода-вывода.
ADV#	ALE	CCR.3=0	O	Адрес верен. Выходной сигнал становится активным только в течение доступа к внешней памяти (активный уровень низкий). ADV# показывает, что адрес зафиксирован на системной шине адрес/данные. Этот сигнал сохраняет низкий уровень до тех пор, пока шинный цикл продолжается, и возвращается в высокий уровень, как только шинный цикл завершен. Внешняя защелка может использовать этот сигнал для выделения адреса на шине адрес/данные. Дешифратор может использовать этот сигнал для выработки сигнала “выбор кристалла”, используемого внешней памятью.

Продолжение таблицы 13.1

1	2	3	4	5
ALE	ADV#	CCR.3=1	O	Разрешение фиксации адреса. Выходной сигнал становится активным только в течение доступа к внешней памяти (активный уровень – высокий). ALE показывает, что значение адресной информации этого сигнала действительно находится на системной шине адрес/данные. ALE отличается от ADV# тем, что не остается активным в течение всего шинного цикла. Внешняя защелка может использовать этот сигнал для выделения адреса с шины адрес-данные.
BHE#	WRH#	CCR.2=1	O	Разрешение записи старшего байта. Это выходной сигнал с активным низким уровнем, устанавливающийся только при записи слова или старшего байта во внешнюю память. Чтобы выбрать записываемый байт памяти, старший байт используется вместе с A0 = 1.
BREQ#	P1.5	WSR.7= 1	O	Запрос шины. Это выходной сигнал с активным низким уровнем, устанавливающийся во время HOLD-цикла, когда контроллер шины ожидает цикл внешней памяти. Контроллер шины может установить BREQ# в любое время, если установлен HLDA#, при этом он остается активным до тех пор, пока HOLD# не сбросится. Если функция активна, вывод действует как стандартный выход (т. е. не квазидвунаправленный).
BW	–	CCR.1=1	I	Разрядность внешней шины. Если CCR.1=1, этот сигнал управляет разрядностью шины во время внешнего доступа. Когда BW высокий, выбирается 16-битная шина, когда низкий – 8-битная. Если CCR.1=0, разрядность шины всегда 8 бит, при этом данный сигнал игнорируется.
EA#	Выбор режима программиро-вания EA#=VEA	–	I	Доступ к внешней памяти. Этот сигнал с активным низким уровнем разрешает доступ к памяти вне кристалла. Если уровень высокий это выбирает внутрикристальная OTPROM.
HLDA#	P1.6	WSR.7=1	O	Подтверждение запроса шины. Выход с актив-ным низким уровнем показывает, что контроллер не управляет шиной. Это происхо-дит в ответ на установку внешним устройством сигнала HOLD#.
HOLD#	P1.7	WSR.7=1	I	Запрос шины. Это сигнал с активным низким уровнем, показывающий, что внешнее устройство запрашивает управление шиной. Если функция активна, вывод действует как стандартный вход (не квазидвунаправленный).

Окончание таблицы 13.1

1	2	3	4	5
INST	–	–	O	Выборка команды. Этот сигнал действителен только в цикле чтения внешней памяти. Если уровень сигнала высокий, INST показывает, что выбирается команда. Если сигнал низкого уровня – считываются данные. INST может использоваться в устройствах, которые требуют разделения памяти для байтов данных и команд. Примечание – CCB-байт и вектор прерывания рассматриваются в качестве данных.
RD#	–	–	O	Внешнее чтение. Это выходной сигнал с активным низким уровнем, показывающий чтение внешней памяти.
READY	–	–	I	Готовность. Это входной сигнал с активным высоким уровнем, показывающий, что внешняя память готова передавать или принимать данные.
WR#	WRL#	CCR.2=1	O	Внешняя запись. Это выходной сигнал с активным низким уровнем, показывающий, что идет запись во внешнюю память.
WRH#	BHE#	CCR.2=0	O	Запись старшего байта. При 16-ти битной разрядности шины WRH# устанавливается при записи старшего байта и слова. При 8-битной разрядности (CCR.1=0) - при записи старшего и младшего байтов, а также слова.
WRL#	WR#	CCR.2=0	O	Запись младшего байта. При 16-ти битной разрядности WRL# устанавливается при записи младшего байта или слова. При 8-битной разрядности (CCR.1=1) – при записи старшего и младшего байтов, а также слова.

13.2 Регистр конфигурации кристалла

Регистр конфигурации кристалла (CCR) Chip Configuration Register управляет разрядностью шины, формированием сигнала строб записи и сигнала “адрес действителен” с количеством циклов ожидания, которые могут быть вставлены, пока вывод READY удерживается в низком уровне. Это первый байт, считываемый из памяти, после сброса устройства. Рисунок 13.1 показывает формат CCR.

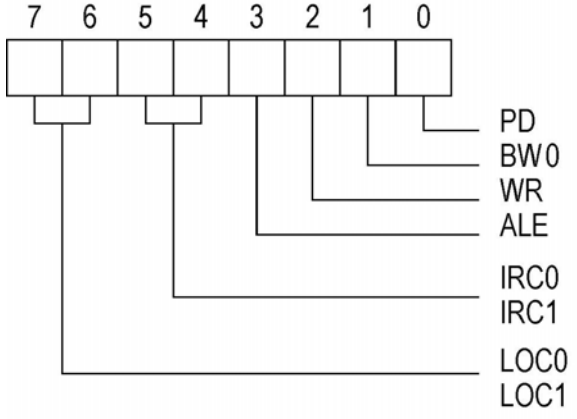


Рисунок 13.1 – Регистр конфигурации кристалла

Когда устройство сбрасывается, контроллер шины выбирает Chip Confuration Byte (CCB) из ячейки 2018h и загружает его в CCR. Во время выборки CCB адрес выставляется так же, как и при нормальных операциях. CCB выбирается из внутренней OTPROM, если сигнал EA# высокий, и из внешней памяти, если сигнал низкий. CCR загружается только в течение “последовательности” сброса. Однажды загруженный, CCR не может быть изменен до следующего сброса устройства. Обычно CCB программируется однократно, когда пользовательская программа откомпилирована и не переопределяется во время обычных операций.

Отсутствие установок в CCR позволяет сигналам READY и BASWIDTH управлять временем и разрядностью шины во время выборки CCB. Если CCB хранится в медленной внешней памяти, необходимо установить сигнал READY в низкий уровень. Это приводит к тому, что контроллер шины автоматически вставляет до трех циклов ожидания при выборе CCB (CCR.4 и CCR.5 определяют количество циклов ожидания). Если CCB расположен в 16-ти битной внешней памяти, управляющий сигнал BW высокий, если в 8-ми битной внешней памяти – низкий. Легче управлять, если вывод BW в низком уровне (8-ми битная шина), пока происходит выборка CCB (даже когда в устройстве используется 16-ти битная внешняя шина). Чтобы обеспечить совместимость, рекомендуется загрузить в ячейку 2019h константу 20h. При этих условиях МК считывает 20h в Порт 4 во время полной выборки CCB. Если внешняя память выводит 20h в старший байт, не будет конфликтов на шине.

После загрузки CCB в CCR разрядность шины конфигурируется как фиксированная 8-ми битная или как динамически управляемая с помощью сигнала BW, как определено в CCR.1.

13.3 Разрядность шины и конфигурация памяти

Внешняя шина МК может работать как 8-битная или как 16-битная мультиплексированная шина адрес/данные (смотрите рисунок 13.2). Значение CCR.1 определяет разрядность шины как фиксированную 8-битную или как динамическую 16-/8-битную шину, управляемую сигналом BW. Если CCR.1 очищен, контроллер шины видит ее в 8-битном режиме. Когда выполняется код с 8-битной шины, происходит некоторое понижение производительности. Слово считывается и записывается во внешнюю память с использованием дополнительного шинного цикла и предварительная выборка в очередь не используется полностью.

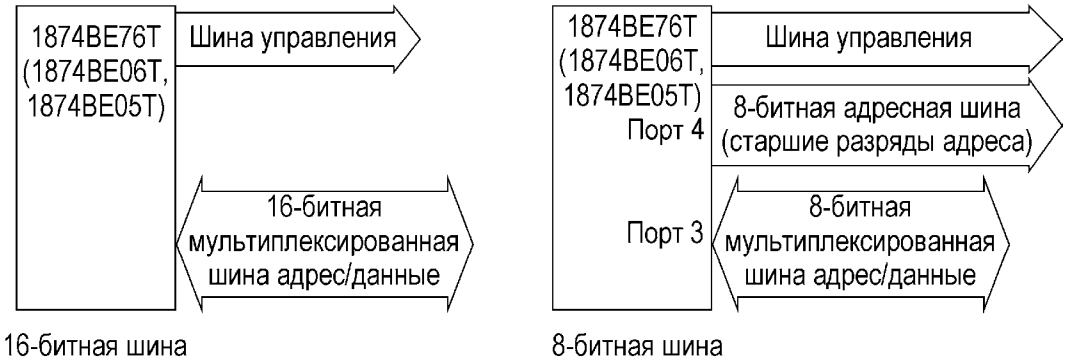


Рисунок 13.2 – Возможные варианты шины адрес/данные

Если установлен CCR.1, то сигнал BW управляет разрядностью шины. Когда BW высокий, шина имеет разрядность 16 бит, и разрядность 8 бит, когда BW имеет низкий уровень. Сигнал BW является действительным после появления адреса на шине (смотрите рисунок 13.3).

Сигнал BW может использоваться в различных применениях. Например, система может хранить в 16-битных устройствах памяти код программы, а в 8-битных устройствах памяти – данные. Сигнал BW может быть связан с входом выборки

кристалла у 8-битных устройств памяти как показано на рисунке 13.11). Когда BW имеет низкий уровень, разрешается 8-битный режим шины, и выбирается 8-битное устройство памяти. Когда BW высокий, то разрешается 16-битный режим шины и не выбирается 8-битное устройство памяти.

13.3.1 Временные требования для сигнала BW

При использовании сигнала BW для динамического переключения разрядности между 8-битной и 16-битной шинами нужно согласовать для правильной работы время установки и удержания (смотрите рисунок 13.3). Так как декодированное значение адреса используется для выработки сигнала BW, то время установки определяется от момента установки истинного адреса. Это определяется величиной Tavgv, которая показывает имеющийся интервал времени для декодирования значения адреса и выработки сигнала BW.

BW должен удерживать свое значение в течение минимального времени Tclgx. Обычно время удержания минимум 0 нс после перехода сигнала CLKOUT в низкий уровень.

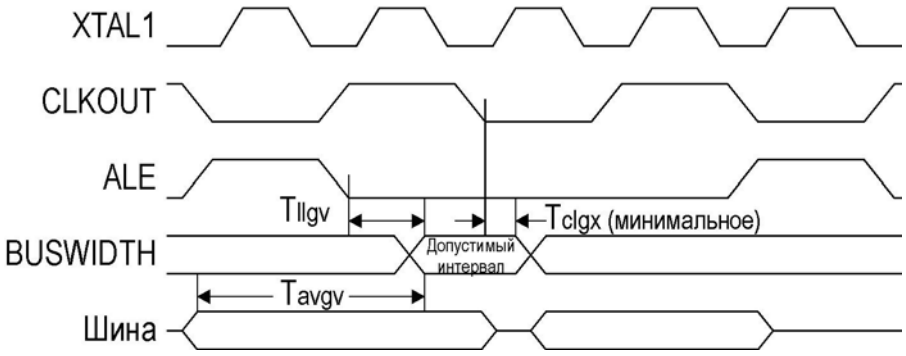


Рисунок 13.3 – Временные диаграммы для сигнала BW

Примечание – Ранние HMOS устройства использовали время установки BW, которое определялось задним фронтом сигнала ALE (Tllgv). Это значение является некритичным для CMOS устройств, которые используют внутренний двухфазный тактовый сигнал. Этот параметр приведен только для сравнения.

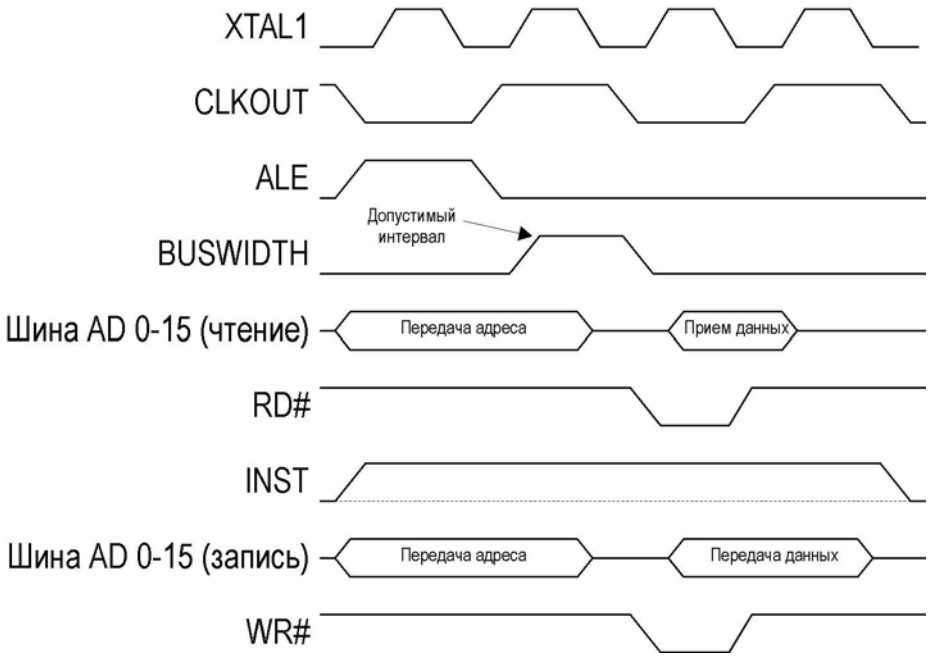


Рисунок 13.4 – Упрощенные временные диаграммы обмена по 16-битной внешней шине

13.3.2 Шина разрядностью 16 бит

Когда контроллер сконфигурирован для работы в режиме16-битной шины (т. е. CCR.1 установлен, и сигнал BW имеет высокий уровень), линии AD0-AD15 формируют 16-битную мультиплексированную шину адрес/данные, шина адрес/данные разделяет выводы Портов 3 и 4 (смотрите таблицу 13.1). Рисунок 13.4 показывает упрощенные временные диаграммы для внешних циклов записи и чтения. Следует учитывать характеристики деталей, которые приводятся в спецификациях устройств внешней памяти.

Address Latch Enable (ALE) выделяет адрес на шине, стробируя регистр-защелку (такую, как КР1533ИР22). Адрес (‘AddressOut’) будет верным на шине после сброса ALE.

13.3.2.1 16-битный Цикл Чтения

Контроллер шины переключает шину на ввод и затем вырабатывает сигнал RD# (переводит в низкий уровень), таким образом он может принимать данные. Внешняя память должна поместить данные на шину после спадающего фронта сигнала RD#. Установка сигнала INST показывает, что идет выборка команды.

13.3.2.2 16-битный Цикл Записи

Контроллер шины переводит сигнал WR# в низкий уровень, затем помещает данные на шину. Спадающий фронт сигнала показывает, что данные на шине стабильны. В это время внешняя система должна зафиксировать данные.

13.3.3 Шина Разрядностью 8 Бит

Когда МК сконфигурирован для работы в режиме 8-битной шины, линии AD0-AD7 формируют мультиплексированную шину, младший адрес и данные. Линии AD8-AD15 не мультиплексируются. Старшие адреса защелкиваются и сохраняют свое значение в течение всего шинного цикла. Рисунок 13.5 показывает упрощенную временную диаграмму для циклов внешнего чтения и записи.

Сигнал ALE используется для выделения младшего адреса, стробируя регистр-защелку (КР1533ИР22).

13.3.3.1 Цикл Чтения на 8-битной шине

После сброса сигнала ALE контроллер шины переключает шину на ввод и переводит сигнал RD# в низкий уровень. Затем внешняя память должна поместить данные на шину. Эти данные должны быть стабильными после нарастающего фронта сигнала RD#. Чтобы считать слово данных, контроллер шины осуществляет два последовательных чтения, читая первым младший байт, а затем старший байт.

13.3.3.2 Цикл Записи на 8-битной шине

После сброса сигнала ALE контроллер шины выдает данные на AD0 – AD7 и затем устанавливает сигнал WR# в низкий уровень. Внешняя память должна зафиксировать данные до момента перехода сигнала WR# в высокий уровень. Эти данные должны удерживаться на шине после перехода сигнала WR# в высокий уровень. Чтобы записать слово данных, контроллер шины осуществляет два последовательных цикла записи, записывая первым младший байт, а затем старший байт.

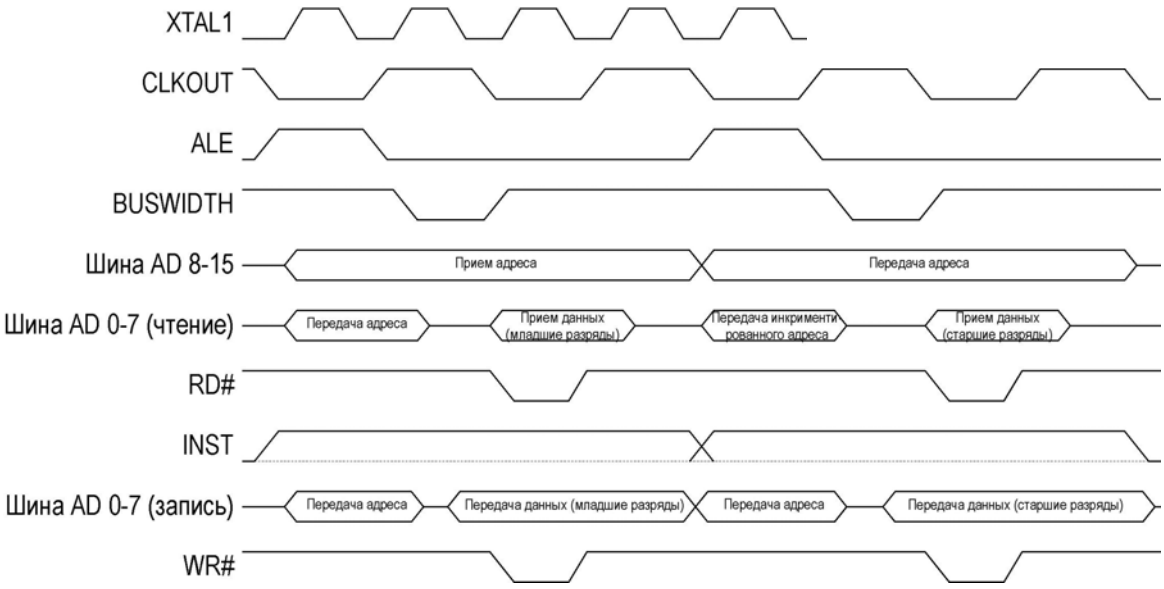


Рисунок 13.5 – Упрощенные временные диаграммы обмена по 8 битной внешней шине

13.4 Управление сигналом READY

Внутренняя схема управления готовностью позволяет для низкоскоростных устройств внешней памяти увеличивать длину шинного цикла записи и чтения у МК. Если устройство внешней памяти не готово для доступа, то необходимо установить сигнал READY в низкий уровень и удерживать его так до тех пор, пока устройство памяти не будет готово завершить операцию. До тех пор, пока сигнал READY низкий, контроллер шины вносит состояния ожидания в шинный цикл. CCR биты 4 и 5 (IRC0 и IRC1) определяют максимальное количество состояний ожиданий, которые могут быть внесены (смотрите таблицу 13.2). Когда установлены оба бита, контроллер шины вносит состояния ожидания до того времени, пока внешняя память удерживает сигнал READY низким. Однако контроллер шины находится в состоянии ожидания до того момента, пока устройство внешней памяти удерживает сигнал READY, или количество состояний ожидания ограничено числом циклов (1, 2 или 3), определяемым CCR.4 и CCR.5.

Таблица 13.2 – Управление периодами ожидания

CCR.5	CCR.4	Максимальное число тактов ожидания
0	0	Ограничено одним
0	1	Ограничено двумя
1	0	Ограничено тремя
1	1	Не определено (управляется сигналом)

При использовании сигнала READY для управления состояниями ожидания необходимо добавить внешнее аппаратное обеспечение для счета состояний ожидания и задержки сигнала READY в течение определенного времени (смотрите в спецификации характеристики для сигнала READY). Следует учитывать, что неисправные устройства памяти могут занимать шину адрес/данные неопределённо долго.

Примечание – Управление готовностью действует только для внешней памяти. Нет возможности добавить состояния ожидания, когда идет доступ к внутренней ОЗУ или OTPROM области.

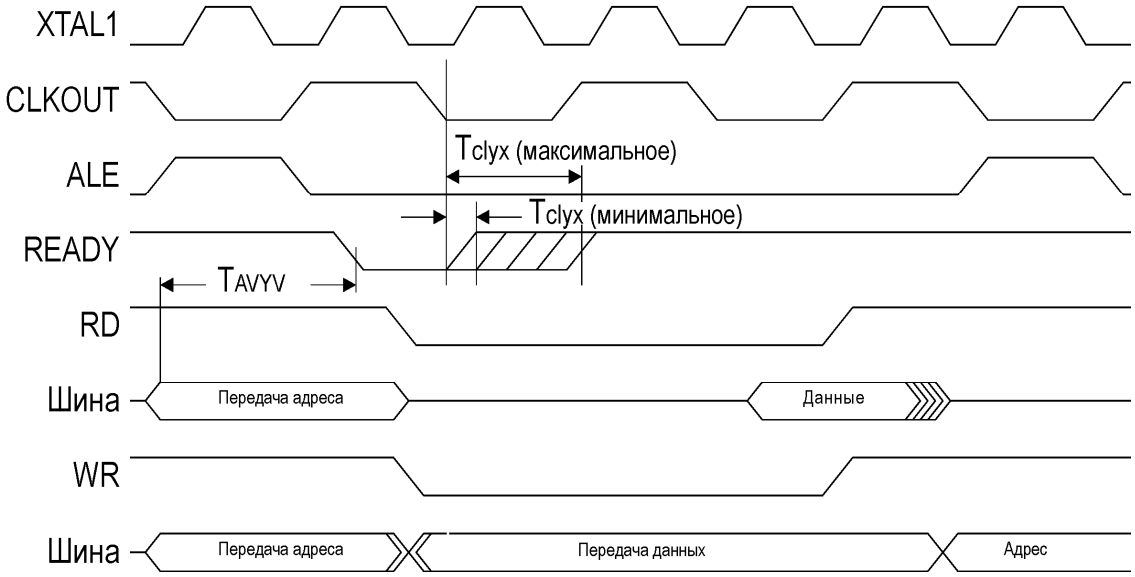


Рисунок 13.6 – Временные диаграммы для сигнала READY

13.4.1 Временные требования для сигнала READY

При использовании сигнала READY для ввода состояний ожиданий в цикл шины время установки и удержания должны быть определенными (смотрите рисунок 13.6). Так как используется дешифрированное значение адреса для выработки сигнала READY, время установки определяется от момента установки адреса на шине. Значение Tavyv определяет количество времени, имеющееся для декодирования и формирования READY от момента, когда адрес уже установлен. Сигнал READY должен удерживать свое значение до тех пор, пока не кончится время Tclyx. Обычно это минимум – 0 нс с момента перехода сигнала CLKOUT в низкий уровень. Нельзя превышать максимальное время Tclyx, иначе появляются дополнительные (нежелательные) циклы ожидания.

Когда для ограничения циклов ожидания (до 1, 2 или 3 максимум) программируется CCR, можно связать прямо сигнал READY с разрешением кристалла в устройстве внешней памяти. Это простейший путь для добавления 1-3 циклов ожидания при выборе устройства.

Примечание - Ранние HMOS устройства использовали время установки сигнала READY, которое определялось спадающим фронтом сигнала ALE (Tllyv). Этот параметр не критичен для CMOS устройств, которые используют внутренний двухфазный тактовый сигнал. Он приводится только для сравнения.

13.5 Протокол захвата шины

МК поддерживает протокол захвата шины, который позволяет внешним устройствам получать управление шиной адрес/данные. Протокол использует три сигнала: HOLD# (Hold Reguest - запрос захвата), HLDA# (Hold Acknowledge - подтверждение захвата) и BREQ# (Bus Reguest - запрос шины), которые являются альтернативными функциями Порта 1. Когда устройство хочет использовать МК шину, оно активизирует сигнал HOLD#. HOLD# является недействующим, когда CLKOUT имеет низкий уровень. МК отвечает освобождением шины, активизируя сигнал HLDA#. Шина адрес/данные отключена, и сигналы ALE, RD#, WR# и BHE# выключены и удерживаются в неактивном состоянии в течение всего времени захвата. Рисунок 13.7 показывает временную диаграмму для протокола захвата шины. Смотрите спецификацию для определения временных требований.

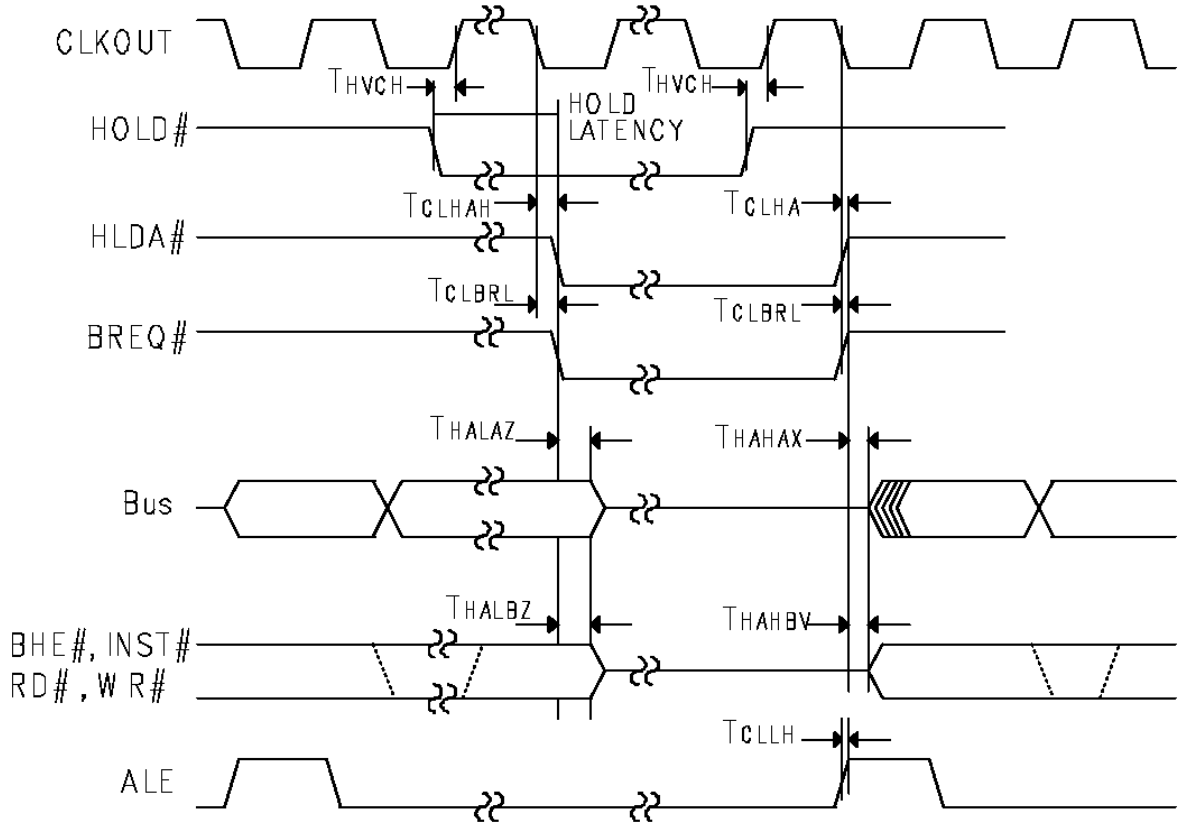


Рисунок 13.7 – Временные диаграммы для сигналов HOLD#, HLDA#

После того как внешнее устройство заканчивает работать с шиной, оно сбрасывает сигнал HOLD# в высокий уровень. В ответ МК переводит сигнал HLDA# в высокий уровень и принимает управление шиной. Пока МК отключен от шины, он может запросить управление шиной, активизируя сигнал BREQ#. Далее внешнее устройство устанавливает сигнал HOLD# в высокий уровень, МК захватывает управление шиной и затем деактивизирует сигналы BREQ# и HLDA#.

13.5.1 Разрешение протокола захвата шины

Бит HLDEN в выбранном окне регистров (WSR.7) разрешает протокол захвата шины. Установка этого бита разрешает протокол и выбирает альтернативные функции выводов порта P1.7 (HOLD#), P1.6 (HLDA#) и P1.5 (BREQ#). Если альтернативные функции разрешены, то выводы действуют как стандартные (не квазидвунаправленные) входы или выходы. Если протокол захвата шины был уже выбран, то функции порта не могут быть переназначены без сброса устройства. Однако функции захвата могут динамически разрешаться и запрещаться как описано ниже.

13.5.2 Запрещение протокола захвата шины

Для запрещения запроса захвата необходимо очистить WSR.7. МК не берет немедленно управление шиной после очистки HLDEN. Он ожидает окончание текущего HOLD# запроса и затем запрещает будущие захваты шины, игнорируя любые новые запросы до тех пор, пока бит снова не установится. Иногда важно, чтобы другие устройства не мешали управлять шиной, пока выполняется блок кода. Один путь защиты блока кода - это очистка WSR.7, после чего выполняется команда JBC для проверки статуса сигнала HLDA#. Команда JBC препятствует началу выполнения в РАЛУ защищенного блока до тех пор, пока обслуживается текущий HOLD# запрос, и будущие захваты запрещены.

```
        DI                          ; Запрещение прерываний
        ANDB WSR,#7Fh               ;Запрещение запроса захвата
WAIT:
        JBC IOPORT1,6 WAIT          ;Проверка сигнала HLDA#
        ***                         ;Выполнение запрещенного блока команд
        ORB WSR,#80h                ;Разрешения запроса захвата
        EI                          ;Разрешение прерываний
```

13.5.3 Скрытое состояние HOLD (Hold Latency)

При установке внешним устройством HOLD# МК заканчивает текущий цикл шины и затем устанавливает HLDA#. Время, необходимое МК для установки HLDA# после установки внешним устройством HOLD#, называется скрытым состоянием HOLD (HoldLatency) (смотрите рисунок 13.7). Таблица 13.3 показывает максимальное время (Hold Latency) для каждого типа шинного цикла.

Таблица 13.3 - Максимальное время реагирования на сигнал Hold

Тип шинного цикла	Максимальное время скрытого состояния HOLD
Внутреннее выполнение или Idle-режим	1,5 машинных цикла
16-битное внешнее выполнение	2,5 машинных цикла
8-битное внешнее выполнение	4,5 машинных цикла

13.5.4 Возврат управления шины

Пока установлен HOLD# МК может выполнять код из внутренней памяти. Когда необходим доступ к внешней памяти, он устанавливает сигнал BREQ# и ожидает сброс сигнала HOLD# внешним устройством. После установки BREQ# контроллер не может ответить на любое требование прерывания, включая NMI, до тех пор, пока внешнее устройство не сбросит HOLD#. Через один временной цикл после перехода сигнала HOLD# в высокий уровень МК сбрасывает HLDA# и вновь принимает управление шиной.

Если МК сбрасывается, когда шина захвачена внешним устройством, то может случиться конфликт на шине. Например, ЦПУ может сделать выборку CCB из внешней памяти после перехода сигнала RESET# в высокий уровень. Может произойти конфликт на шине, так как и внешнее устройство, и МК пытаются обратиться к памяти. Одно из решений этой проблемы – использовать сигнал RESET# как системный сброс; при этом все устройства, управляющие шиной (включая МК), сбрасываются одновременно. В разделе 11 приведен пример схемы системного сброса.

13.6 Режимы управления шиной

CCR.2 и CCR.3 определяют, какие сигналы управления шиной будут генерироваться во время внешнего цикла чтения или записи. Таблица 13.4 показывает 4 режима управления шиной и установки CCR.3 и CCR.4 для каждого из них.

Таблица 13.4 – Режимы управления шиной

Режим управления шиной	Сигналы управления шиной	CCR.3	CCR.2
Adress Valid with Write Strobe	ADV#, RD#, WRL#, WRH#	0	0
Adress Valid Strobe	ADV#, RD#, WR#, BHE#	0	1
Write Strobe	ALE, RD#, WRL#, WRH#	1	0
Стандартный режим управления	ALE, RD#, WR#, BHE#	1	1

13.6.1 Стандартный режим управления шиной (Standart Bus Control)

В стандартном режиме управления шиной МК вырабатывает стандартные сигналы управления шиной: ALE, WR#, RD# и BHE# (смотрите рисунок 13.8). ALE устанавливается при установке адреса и может быть использован для фиксации адреса снаружи. Низкий BHE# выбирает банк памяти, который адресуется старшим байтом на шине данных. RD# устанавливается для чтения внешней памяти, и WR# устанавливается для записи во внешнюю память.

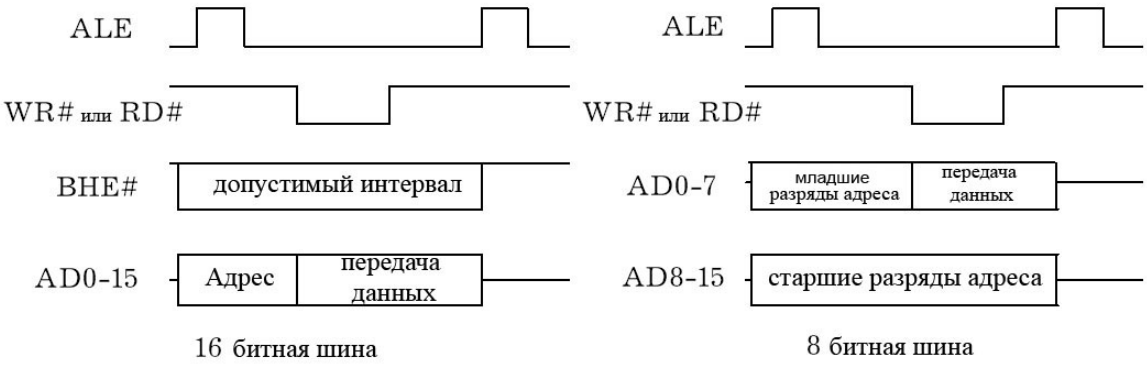


Рисунок 13.8 – Стандартное управление шиной

Когда сконфигурировано устройство для использования 16-битной шины, сигналы, разделяющие запись младшего и старшего байтов, должны быть сгенерированы для записи единственного байта. На рисунке 13.9 показана простейшая цепь, которая комбинирует BHE# и A0 для обеспечения этих сигналов (WRL# и WRH#). Схожая пара сигналов для чтения необязательна. Для чтения одного байта с 16-битной шины оба байта выставляются на шину данных, и процессор отбрасывает ненужный байт.

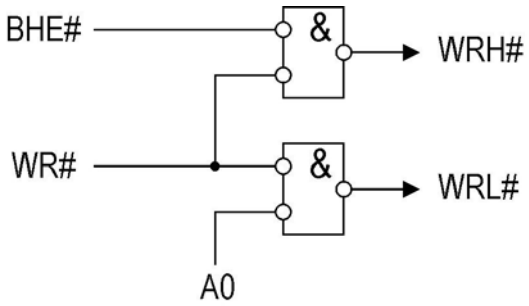


Рисунок 13.9 – Дешифрация сигналов WRL# и WRH#

На рисунке 13.10 показана 8-битная система с EPROM и RAM. EPROM расположен в нижней части памяти, а RAM находится в верхней части. Эта система конфигурируется с использованием старшего значащего бита адреса (A15) как сигнал выбора кристалла и ALE - как сигнал защелки адреса.

На рисунке 13.11 показана система, которая использует динамическую разрядность шины (установлен CCR.1). Код исполняется из двух EPROM, и данные сохраняются в RAM разрядностью в один байт. RAM расположена в старшей памяти и выбирается, если A15 имеет высокий уровень; RAM также выбирается в режиме 8-битной шины под управлением сигнала BW, который имеет низкий уровень.

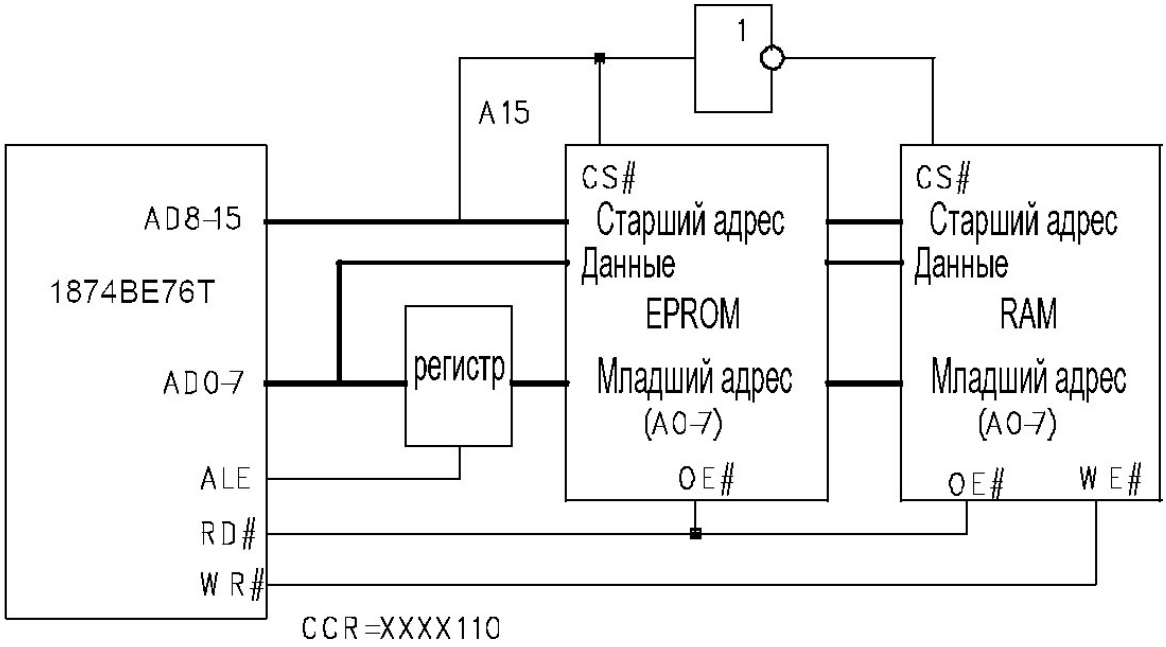


Рисунок 13.10 – Пример 8-битной системы с EPROM и RAM

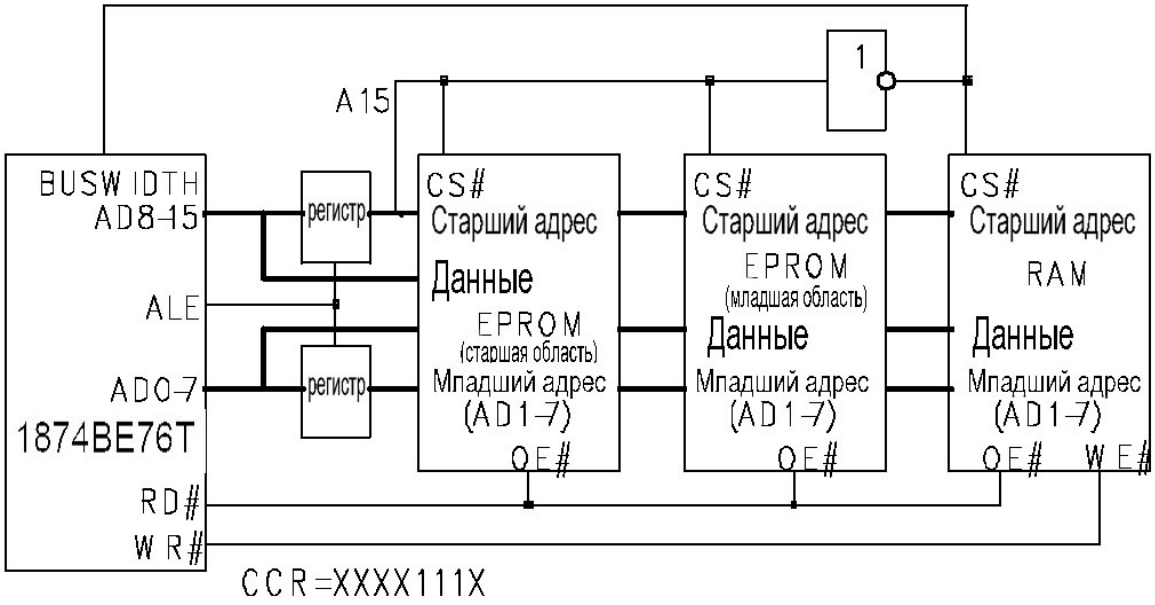


Рисунок 13.11 – Пример 16-битной системы с изменяемой разрядностью шины

13.6.2 Режим Write Strobe

Режим Write Strobe устраняет необходимость во внешней дешифрации при записи старшего и младшего байтов во внешнюю 16-битную RAM в 16-битном режиме. При выборе режима Write Strobe вырабатывается WRL# и WRH# взамен WR# и BHE#. WRL# устанавливается для всех записей младших байтов (четные адреса) и всех записей слов. WRH# устанавливается при записи старшего байта (нечетный адрес) и записи слова. В 8-битном режиме WRH# и WRL# устанавливаются для четных и нечетных адресов. Временные диаграммы режима Write Strobe показаны на рисунке 13.12.

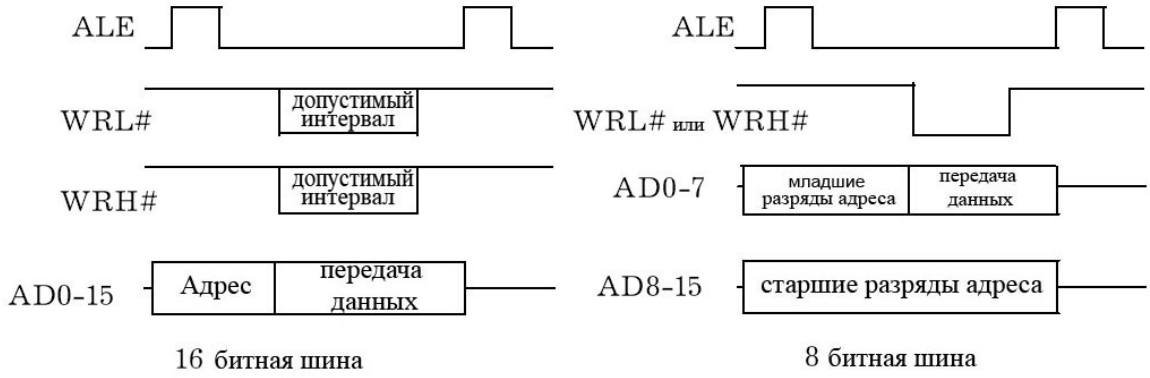


Рисунок 13.12 – Режим Write Strobe

На рисунке 13.13 показана 16-битная система, которая использует два EPROM и два RAM. Это пример конфигурации для использования режима Write Strobe. ALE фиксирует адрес, и A15 служит для выбора кристалла для EPROM и RAM. WRL# устанавливается в момент записи младшего байта и слова. WRH# устанавливается в момент записи старшего байта и слова. При этом устройство RAM не использует A0, WRL# и WRH#, определяющие выбор младшего (A0=0) или старшего (A0=1) байтов.

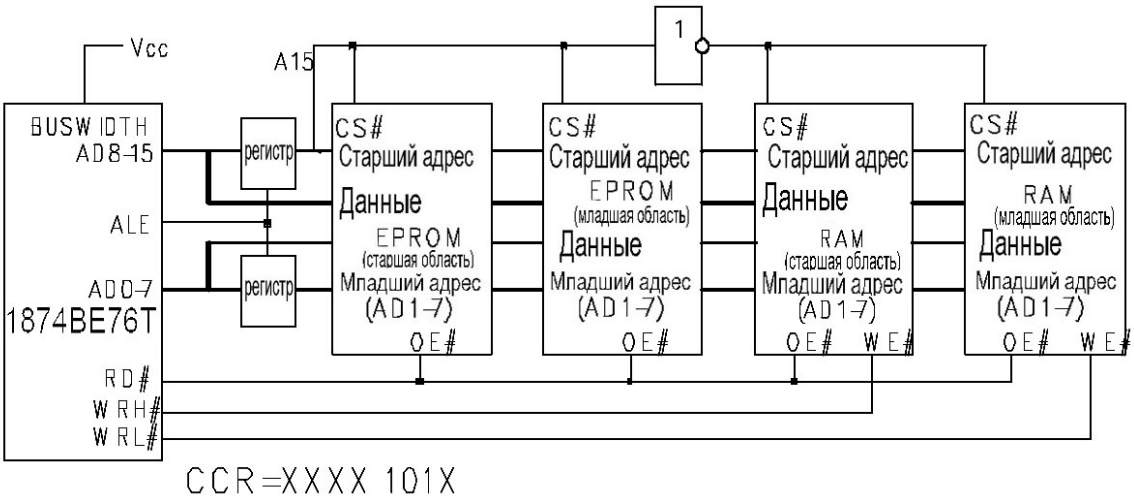


Рисунок 13.13 – Пример 16-битной системы с однобайтной записью в RAM

13.6.3 Режим Address Valid Strobe

Когда выбран Address Valid Strobe режим, МК вырабатывает сигнал Address Valid (ADV#) взамен сигнала Address Latch Enable (ALE). ADV# устанавливается после того, как установился адрес. Этот сигнал можно использовать для фиксации значения адреса и одновременно для разрешения устройства внешней памяти.

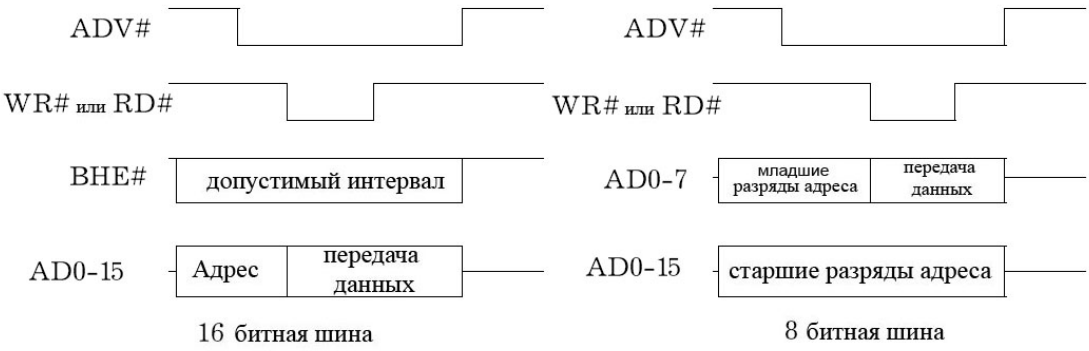


Рисунок 13.14 – Режим Address Valid Strobe

Различие между ALE и ADV# заключается в том, что ADV# устанавливается для полного цикла шины, а не для точной фиксации адреса. Рисунок 13.15 показывает различие между ALE и ADV# для одного цикла чтения или записи. Примечательно, что когда циклы обмена с памятью следуют друг за другом (back-to-back), действие ADV# выглядит идентично действию ALE. Различие становится явным, только когда шина не занята. Из-за того, что ADV# имеет высокий уровень в течение этих периодов, внешняя память будет запрещена, таким образом сохраняется и экономится энергия.

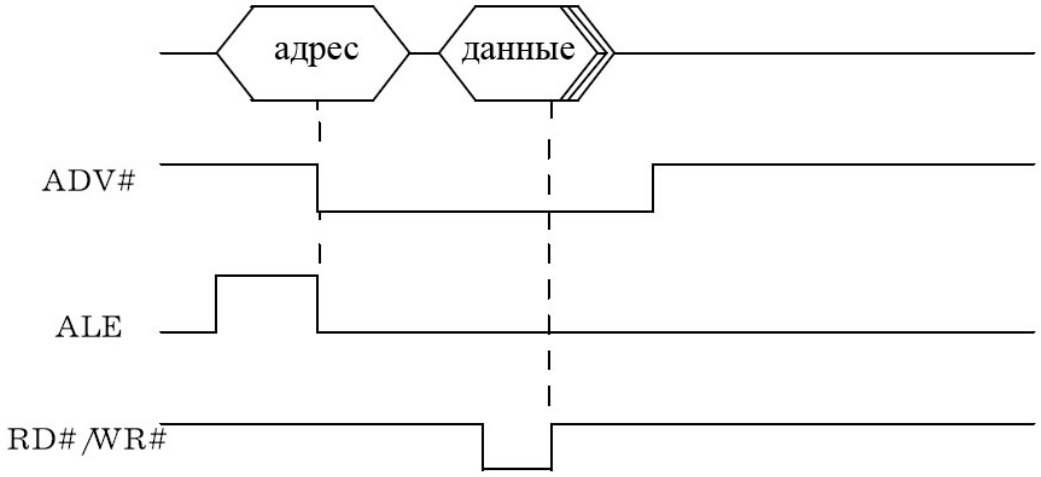


Рисунок 13.15 – Сравнение шинных циклов ALE и ADV#

На рисунках 13.16 и 13.17 показана простейшая схема, которая использует режим Address Valid Strobe. На рисунке 13.16 показана простая 8-битная система с одной EPROM для режима Address Valid Strobe. Такая конфигурация системы использует сигнал ADV# дважды: как сигнал выбора кристалла EPROM и сигнал защелки адреса.

На рисунке 13.17 показана 16-битная система с двумя EPROM. Эта конфигурация системы использует сигнал ADV# дважды как сигнал выбора кристалла EPROM и сигнал защелки адреса.

13.6.4 Режим Address Valid with Write Strobe

Когда выбран режим Address Valid with Write Strobe, МК вырабатывает сигналы управления шиной ADV#, WRL#, WRH#. Этот режим применим в простой системе, использующей внешнюю 16-битную RAM. На рисунке 13.18 показаны временные соотношения. Сигнал RD#, не описанный выше, аналогичен WRL#, WRH# и WR#. Пример системы, использующей режим Address Valid with Write Strobe, приведен на рисунке13.19.

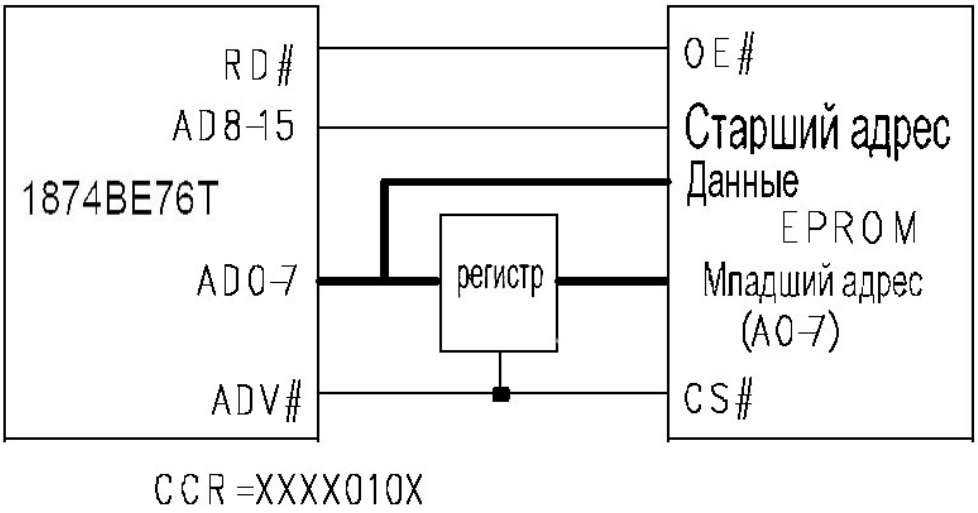


Рисунок 13.16 – 8 битная система с EPROM

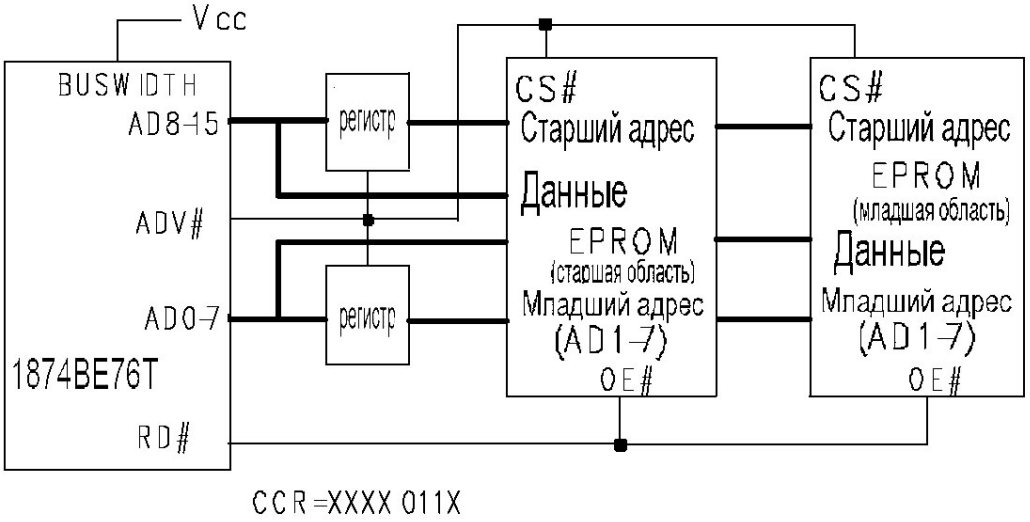


Рисунок 13.17 – 16 битная система с EPROM

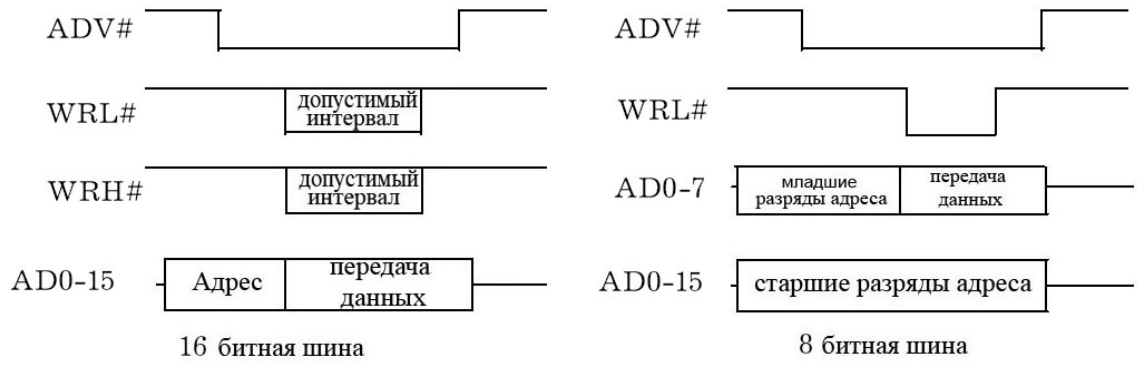


Рисунок 13.18 – Временные соотношения для режима Address Valid with Write Strobe

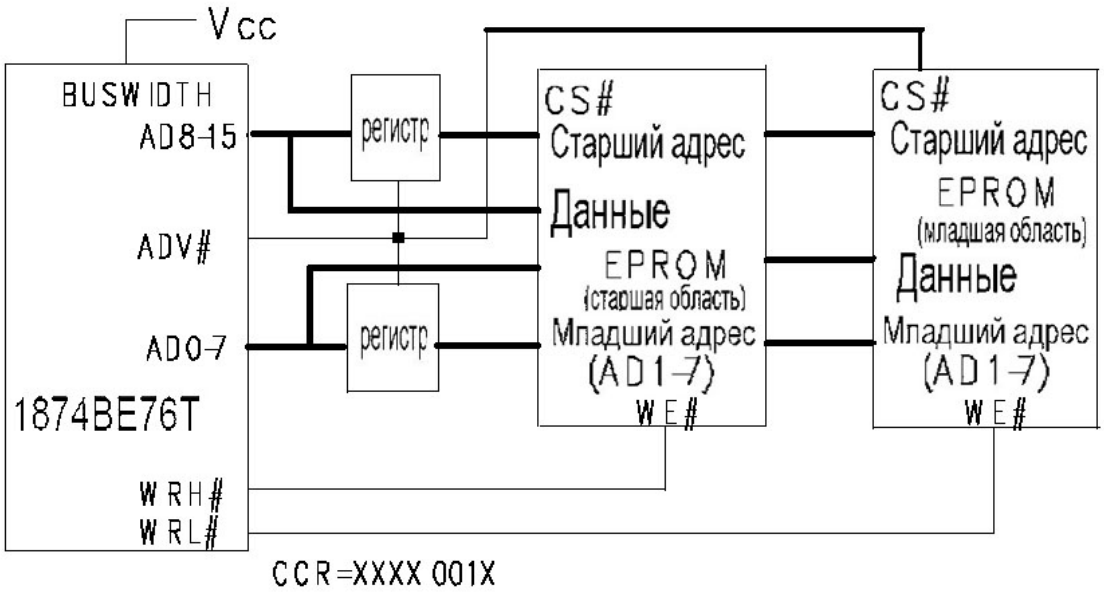


Рисунок 13.19 – 16-битная система с RAM

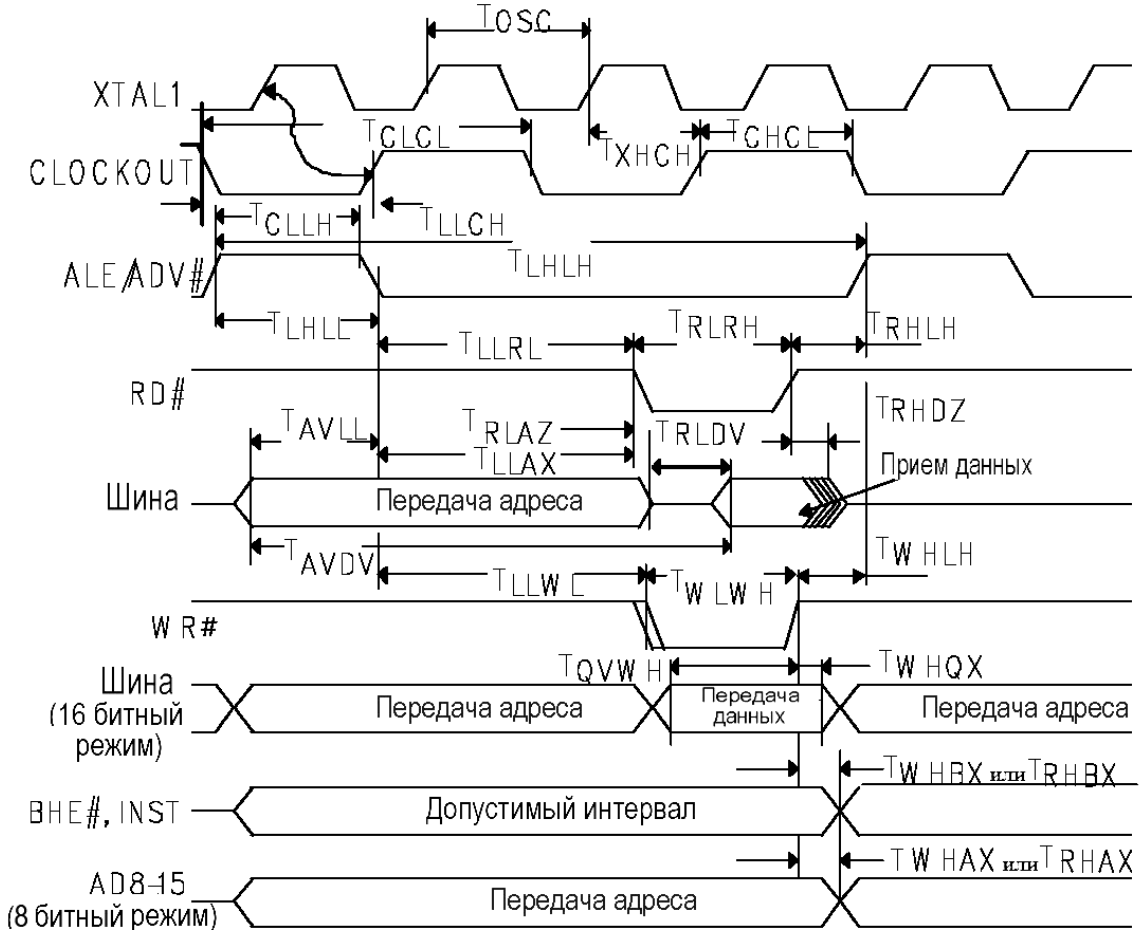


Рисунок 13.20 – Временная диаграмма системной шины

13.7 Временные соотношения системной шины

Большинство системных шин имеют временные соотношения, показанные на рисунке 13.20 и описанные в таблице 13.5.

13.7.1 Перечень Принятых Сокращений

Каждый символ имеет два буквенных префикса у “T” (для времени). Характеристика показывает сигнал и условия его появления. Символы отображают время между двумя точками сигнал/условие.

Условия:			Сигналы:	
H - высокий уровень	A -	адрес	HA -	HLDA#
L - низкий уровень	B -	BHE#	L -	ALE/ADV#
V - значение	BR -	BREQ#	W -	WR#/WRH#/WRL#
X - неопределённое	C -	CLKOUT	Q -	выходные данные
Z - неподключенный	D -	данные	R -	RD#
	G -	BW	X -	XTALI
	H -	HOLD#	Y -	READY

Таблица 13.5 – Описание временных соотношений

Параметр	Описание
1	2
Tavyv	Время от установления Address до установления Ready: максимальное время для системы памяти, устанавливающей сигнал Ready после вывода адреса из МК и гарантирующей один цикл ожидания.
Tllyh *	Установление Ready после перехода ALE в низкий уровень: максимальное время для системы памяти, устанавливающей сигнал READY после сброса ALE, гарантирует введение по крайней мере одного цикла ожидания
Tclyx	Сохранение Ready после сброса CLKOUT: минимальное время сохранения всегда 0 нс.
Tllyx	Сохранение Ready после сброса ALE: минимальное время удержания сигнала Ready после сброса ALE. Если максимальное значение превышается, дополнительно вводятся циклы ожидания.
Tavgv	Address действителен и BW активен. Максимальное время между установлением адреса и появлением BW. Если это требование превышается, МК может не ответить соответствующим шинным циклом.
Tllgv	Сброс ALE и BW активен: максимальное время между сбросом ALE/ADV и появлением BW. Если это требование превышается, МК может не ответить соответствующим шинным циклом.
Tclgx	BW Hold после сброса CLKOUT: минимальное время, необходимое для удержания BW, после сброса CLKOUT.
Tavdv	Адрес и входные данные действительны: максимальное время, которое имеют устройства памяти для выдачи данных после установления адреса.
Trldv	Сброс RD# и входные данные действительны: максимальное время для выдачи данных системой памяти после установления сигнала RD#.
Tcldv	Сброс CLKOUT и входные данные действительны: максимальное время удержания выходных данных системы памяти после сброса CLKOUT.
Trhdz	RD# High и “плавающие” входные данные: максимальное время перехода RD# в неактивное состояние, пока система памяти освобождает шину. Если это время не учитывается, возможен конфликт на шине.
Trhdx	Удержание данных после сброса RD#: время после сброса RD# в неактивное состояние, в течение которого система памяти должна удерживать данные на шине.
Fxtal	Частота на XTAL: частота входного сигнала на входе XTAL1. Скорость работы внутренней шины в устройстве равна 1/2 Fxtal.
Tosc	1/Fxtal: все временные соотношения относятся и к Tosc.
Txhch	XTAL1 высокий для CLKOUT (высокого или низкого).
Tclcl	CLKOUT временной цикл: нормально 2Tosc.
Tchcl	Период высокого CLKOUT: необходим в системе, использующей CLKOUT как тактовый сигнал для внешних устройств.
Tcllh	Время от сброса CLKOUT до установки ALE/ADV#. Используется для установления других временных соотношений.
Tllch	Время от спадающего фронта ALE/ADV# до нарастающего фронта CLKOUT, используется для определения других временных соотношений.
Tlhlh	Время ALE цикла: минимальное время между импульсами ALE.
Tlhll	Длительность высокого уровня ALE/ADV#. Этот параметр используется при разработке внешней защелки адреса.
Tavll	Установка адреса относительно заднего фронта сигнала ALE/ADV#. Время, необходимое для фиксации адреса до заднего фронта сигнала ALE/ADV#. Этот параметр используется при разработке внешней защелки адреса.

Окончание таблицы 13.5

1	2
Tllax	Удержание адреса после заднего фронта сигнала ALE/ADV#. Время удержания адреса после заднего фронта сигнала ALE/ADV#. Этот параметр используется при разработке внешней защелки адреса.
Tllrl	Задержка перехода сигнала RD# в низкий уровень относительно заднего фронта сигнала ALE/ADV#. Время между спадающим фронтом сигнала ALE/ADV# и установкой сигнала RD#. Необходимо для уверенной дешифрации устройствами памяти текущего адреса и активизации микросхем.
Trlcl	Переход в низкий уровень сигнала RD# относительно заднего фронта сигнала CLKOUT. Время между задним фронтом сигнала CLKOUT и установкой сигнала RD#.
Trlrh1	Длительность импульса сигнала RD#.
Trhlh	Время с момента перехода сигнала RD# в неактивное состояние до появления следующего сигнала ALE/ADV#. Используется для вычисления времени с момента снятия адреса до появления следующего установившегося адреса.
Trlaz	Время между освобождением шины адреса и установлением сигнала RD#. Используется для определения момента, когда МК освобождает шину адрес/ данные.
Tllwl	Задержка перехода сигнала WR# в низкий уровень относительно заднего фронта сигнала ALE/ADV#. Время между спадающим фронтом сигнала ALE/ADV# и установкой сигнала WR#. Необходимо для уверенной дешифрации устройствами памяти текущего адреса и активизации микросхем.
Tclwl	Переход в низкий уровень сигнала WR# относительно заднего фронта сигнала CLKOUT. Время между задним фронтом сигнала CLKOUT и установкой сигнала WR#.
Tqvwh	Время с момента фиксации данных на шине до перехода сигнала WR# в неактивное состояние. Устройства памяти должны соответствовать этому параметру.
Tchwh	Время с момента перехода сигнала CLKOUT в состояние с высоким уровнем до перехода сигнала WR# в неактивное состояние.
Twlwh	Длительность сигнала WR#.
Twhqx	Время удержания данных на шине после перехода сигнала WR# в высокий уровень. Устройства памяти должны соответствовать этому параметру.
Twhlh	Время с момента перехода WR# в неактивное состояние до появления следующего сигнала ALE/ADV#. Также используется для вычисления времени между переходом сигнала WR# в неактивное состояние и фиксацией нового адреса на шине.
Twhbx	Минимальное время с момента перехода сигнала WR# в высокий уровень, в течение которого сигналы BHE#, INST, HOLD должны сохранять свои значения.
Trhbx	Минимальное время с момента перехода сигнала RD# в высокий уровень, в течение которого сигналы BHE#, INST, HOLD должны сохранять свои значения.
Twhax	Минимальное время с момента перехода сигнала WR# в высокий уровень, в течение которого старший байт адреса (AD8 - AD15) в 8 битном режиме будет сохранять свое значение.
Trhax	Минимальное время с момента перехода сигнала RD# в высокий уровень, в течение которого старший байт адреса (AD8 - AD15) в 8- битном режиме будет сохранять свое значение.
* Включён только для совместимости с временными соотношениями HMOS устройств.	

14 Программирование постоянной памяти

Входящая в состав описываемого комплекта микросхема 1874BE76T содержит однократное программируемое запоминающее устройство (типа OTPROM), версию электрически перепрограммируемого запоминающего устройства (EPROM). В МК под ПЗУ отводится 16К байт адресного пространства с 2000h по 5FFFh, можно программировать OTPROM самим или использовать готовый продукт с запрограммированным ПЗУ (ROM).

Данный раздел предназначен для помощи потребителю в программировании устройства 1874BE76T. Есть несколько способов. Могут использоваться автоматический режим (AUTO), режим с использованием специального программатора (SLAVE) или режим программирования во время исполнения программы (RUN-TIME), которые помогут программировать OTPROM. Также можно выбрать режим вывода содержимого памяти (ROM-DUMP) для верификации содержимого OTPROM или режим UPROM, или PCCB для установки защиты ПЗУ. Раздел описывает режимы программирования, их выбор и работу в них.

14.1 Общее описание

Когда выбирается режим программирования (за исключением режима RUN-TIME), устройство отзывается на него выполнением внутреннего алгоритма, который находится в области тестового ПЗУ (test ROM). В общем случае, внутренний алгоритм выполняет функции программирования под руководством внешних источников информации.

Выбор потребителем режима программирования определяет, какой алгоритм выбирается из тестового ПЗУ. Чтобы выполнить программирование правильно, устройство должно иметь следующую минимальную схему включения: XTAL1 подключен, неиспользованные входы связаны с логической единицей или нулем, как этого требует режим, питание и земля подключены. Условия работы, заданные в описании устройства, должны быть выполнены.

В данном разделе рассматриваются требования для каждого режима программирования.

14.2 Режимы программирования

МК поддерживает три способа программирования OTPROM:

- Режим автоматического программирования AUTO дает возможность самопрограммирования 1874BE76T из внешнего EPROM, без использования специального программатора. Этот режим позволяет потребителю использовать прототип схемы проектируемого устройства для программирования.
- Режим программирования SLAVE требует специального программатора. При использовании этого режима потребитель может программировать или верифицировать отдельные слова или наборы слов в OTPROM.
- Режим RUN-TIME позволяет программировать отдельные ячейки OTPROM во время обычного выполнения программы полностью под управлением программного обеспечения. В этом случае потребителю не нужно специально устанавливать режим программирования.
- Режимы ROM-DUMP, UPROM и PCCB обеспечивают другие аспекты программирования ПЗУ:
- режим ROM-DUMP позволяет верифицировать содержимое OTPROM путем записи массива из OTPROM во внешнюю память;
- режим UPROM дает возможность запрограммировать два нестираемых бита, которые запрещают контроллеру шины доступ к внешним командам (DEI бит) или внешним данным (DED бит);

- режим PCCB позволяет программировать байт состояния схемы PCCB. Это не входящая в адресное пространство ячейка тестового ПЗУ, которая устанавливает аппаратную защиту во время режимов программирования.

14.2.1 Выбор режима программирования

Для выбора режима программирования используются сигналы PMODE (PMODE.0 – PMODE.3). Таблица 14.1 описывает функции PMODE и указывает значения PMODE и режимы программирования.

Следует ввести МК в режим программирования установкой питания Vpp (обычно плюс 12,5 В) на EA# во время нарастания сигнала RESET#. Более детально это рассмотрено в подразделе 14.5.

PMODE выбирает один из 5 режимов (не включая режим RUN-TIME). Выводы P0.4 – P0.7 задают 16-ричное значение PMODE. 5 режимов – это SLAVE, ROM-DUMP, UPROM, AUTO и PCCB. PMODE фиксируется при нарастании сигнала на RESET#. Необходимо сбросить устройство, чтобы переключить режимы.

Таблица 14.1 – Описание функций PMODE и его значений

PMODE значения	Режим программирования
0h – 4h	Зарезервированы
5h	SLAVE режим
6h	ROM-DUMP
7h – 8h	Зарезервированы
9h	Режим UPROM
0Ah – 0Bh	Зарезервированы
0Ch	Автопрограммирование
0Dh	PCCB
0Eh – 0Fh	Зарезервировано

14.3 Функции выводов в режиме программирования

Чтобы обслуживать различные режимы программирования, некоторые выводы приобретают новые функции. Таблица 14.2 описывает эти новые функции

Таблица 14.2 – Описание функций выводов в режиме программирования

Имя функции	Тип	Режим	Описание
AINC# P2.4	Вход	SLAVE	Автоинкремент. Активный низкий уровень на входе разрешает автоинкремент. Автоинкремент позволяет считывать и записывать в ячейки OTPROM последовательно, без передачи адреса при каждом обмене.
CPVER P2.6	Выход	SLAVE	Полная верификация программирования. Высокий уровень на этом выводе в режиме SLAVE показывает, что все ячейки запрограммированы верно.

Окончание таблицы 14.2

Имя функции	Тип	Режим	Описание
EA# Выбор режима программирования	Вход	Все режимы программирования, исключая RUN-TIME	Внешний доступ. Если на EA# подано Vpp во время нарастания сигнала RESET#, устройство входит в режим программирования. EA# фиксируется только при нарастании сигнала RESET#. Изменение уровня на входе EA# после сброса не дает эффекта.
PACT# P2.7	Выход	AUTO	Программирование активно. В режиме программирования AUTO низкий уровень на выходе показывает, что идет процесс программирования, а высокий уровень индицирует окончание программирования.
PALE# P2.1	Вход	SLAVE	В режиме SLAVE по заднему фронту PALE считывается адрес/команда из портов 3, 4.
PBUS PORTS 3, 4	Вход/ Выход	Все режимы программирования, исключая RUN-TIME	Шина адрес/команда/ данные. Используется как системная шина для доступа к внешней памяти в режимах ROM-DUMP и автопрограммирования. В режиме SLAVE выводы PBUS требуется подсоединять к U_{CC1} через резистор.
PMODE P0.4 - P0.7	Вход	Все режимы программирования, исключая RUN-TIME	Выбор режима. Определяет, какой алгоритм выбран для программирования OTPROM. Сигналы PMODE фиксируются после сброса устройства и должны сохранять свое значение во время операции. В таблице 14.1 приведены значения PMODE и режимы программирования, им соответствующие.
PROG# P2.2	Вход	SLAVE	Программирование. В режиме SLAVE по спаду сигнала фиксируются данные на PBUS и начинается программирование, по фронту программирование заканчивается.
PVER P2.0	Выход	AUTO SLAVE	Верификация программирования. В режимах SLAVE и AUTO высокий уровень сигнала PVER при успешном программировании ячейки, низкий при обнаружении ошибки.
Vpp	Вход	Все режимы	Напряжение программирования, обычно плюс 12,5 В. В спецификации на каждое устройство определено точное значение напряжения. Превышение этого значения U_{VPP} может вывести устройство из строя.

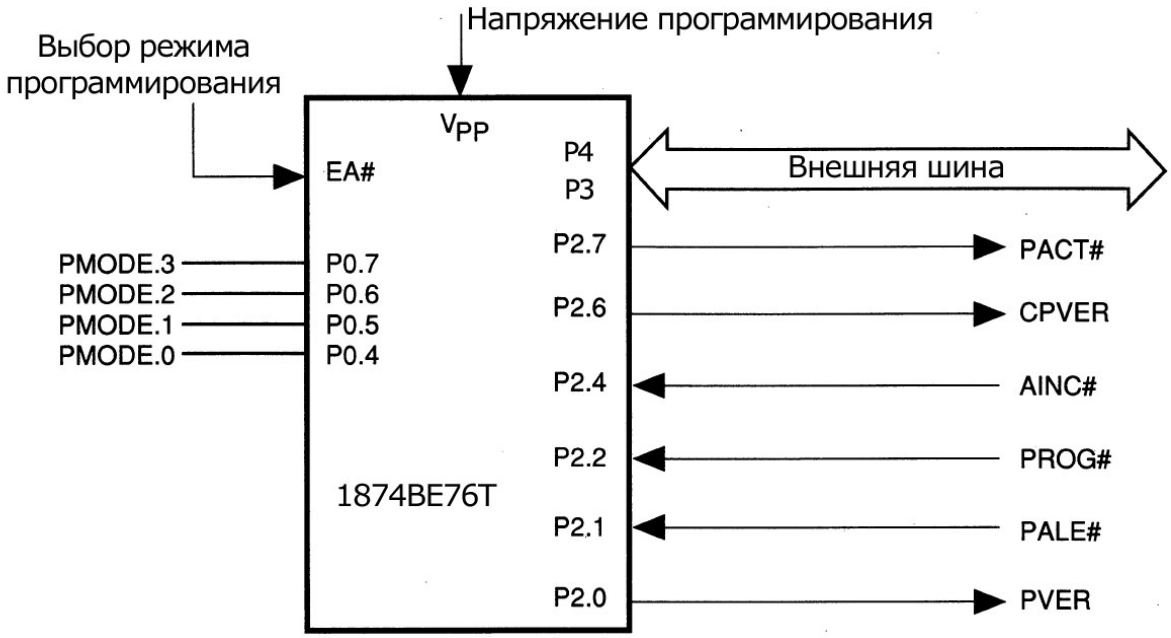


Рисунок 14.1 – Назначение выводов в режиме программирования

14.4 Распределение памяти

Содержимое памяти программ, находящейся по адресам 2080h – 5FFFh и памяти специального назначения (2000h – 207Fh) определяется пользователем. Таблица 14.3 описывает начальные и конечные адреса памяти специального назначения МК шестнадцатиричной и десятичной системах исчисления.

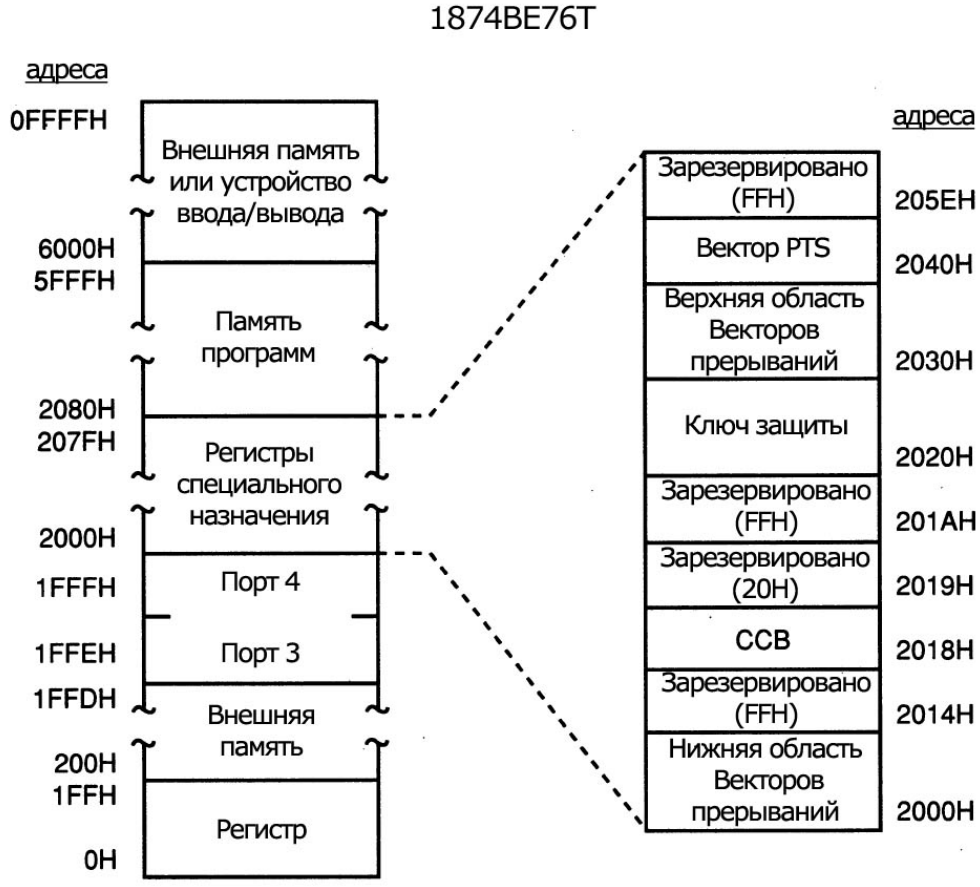


Рисунок 14.2 – Карта памяти специального назначения МК

Таблица 14.3 – Адреса памяти специального назначения МК

Описание	Диапазон адресов	
	16-ричный	10-тичный
Зарезервированы (должны содержать 0FFh)	205Eh – 207Fh	8286 – 8319
PTS вектора	2040h – 205Dh	8256 – 8285
Старшие векторы прерывания	2030h – 203Fh	8240 – 8255
Ключ безопасности	2020h – 202Fh	8224 – 8239
Зарезервированы (должны содержать 0FFh)	201Ah – 201Fh	8218 – 8223
Зарезервированы (должны содержать 20h)	2019h	8217
CCB	2018h	8216
Зарезервированы (должны содержать 0FFh)	2014h – 2017h	8212 – 8215
Младшие векторы прерывания	2000h – 2013h	8192 – 8211

14.5 Включение и выключение питания

Перед началом программирования необходимо выполнить последующие инструкции во избежание выхода из строя устройства:

- Не подавать напряжение на вывод VPP, пока U_{CC1} низкое.
- Не выходить за верхний предел U_{VPP}.
- Напряжение питания на выводах #VCC1, VPP, EA# и RESET# должно быть стабильным, без шума и импульсных помех. На выводы #VCC1 необходимо подать напряжение U_{CC1} = 5,5 В.
- Все выводы #0V должны быть заземлены.

14.5.1 Последовательность включения питания

1 Подать напряжение низкого уровня на вывод RESET#, пока U_{CC1} не стабилизируется. В это время выводы VPP и EA# могут быть не подключены.

2 После стабилизации генератора и U_{CC1}, оставить устройство в состоянии сброса и подать напряжение U_{VPP} на вывод EA#. После стабилизации U_{VPP} на выводе EA#, подать напряжение U_{VPP} на вывод VPP.

3 Установить значение PMODE для выбора алгоритма программирования. Значения для каждого режима приведены в таблице 14.1.

4 Подать высокий уровень на вывод RESET#.

5 Закончить работу с выбранным алгоритмом программирования.

14.5.2 Последовательность выключения питания

1 Подать сигнал RESET# (RESET# = 0). Уровень напряжения на выводе RESET# должен быть низким в течение всей процедуры выключения питания.

2 Снять напряжение U_{VPP} с вывода VPP и оставить вывод неподключенным.

3 Снять напряжение U_{VPP} с вывода EA# и оставить вывод неподключенным.

4 Снять питание с выводов #VCC1 и подождать пока установится 0 В.

14.6 Защита памяти

Аппаратная и программная защита – это отдельные части в схеме защиты памяти. Аппаратная защита предохраняет от внешней записи и считывания следуюшим образом. В регистре состояния схемы (CCR) очищаются биты кода безопасности, CCR.6 (LOC0) и CCR.7 (LOC1), запрещая чтение или запись в OTPROM. Попытка записи вызовет

периодическое повторение циклов записи контроллера шины, однако записи не произойдет. При попытке чтения внутренняя память недоступна. Установка битов регистра специальных функций USFR, USFR.2 (DED) и USFR.3 (DEI) запрещает контроллеру шины доступ к внешним командам или данным. Если контроллер шины попытается выбрать внешние команды или данные, произойдет сброс.

Программная защита предотвращает неавторскую запись или верификацию OTPROM следующим образом. Механизм защиты по ключу ограничивает доступ во внутреннюю память. Если очищен CCB.6 или CCB.7 и устройство введено в режим программирования, то алгоритм режима требует проверки ключа перед началом работы. Если проверка не пройдена, устройство зацикливается, пока не произойдет сброс

14.6.1 Биты кода защиты в CCR

Регистр состояния схемы (CCR) устанавливает структуру шин и другие параметры устройства во время программирования. Рисунок 14.3 иллюстрирует его структуру, включая биты защиты. Если МК находится в нормальном режиме выполнения программ, процедура сброса загружает CCR из ячейки с адресом 2018h, где расположен байт конфигурации устройства (CCB). Это обеспечивает защиту в течение нормального выполнения программы. Очистка CCB.6 или CCB.7 воздействует на режим RUN-TIME и на любые попытки доступа к внутренней памяти от внешних устройств.

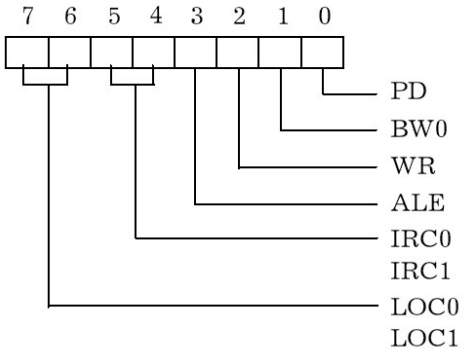


Рисунок 14.3 – Регистр конфигурации кристалла

Если устройство находится в режиме программирования, то процедура сброса загружает CCR из специальной ячейки тестового ПЗУ (доступной только в режиме программирования), называемой байт состояния устройства при программировании (PCCB). Это обеспечивает аппаратную безопасность во время режима программирования. Очистка этих битов воздействует на доступ к OTPROM во всех режимах программирования. PCCB по умолчанию имеет значение 0FFh, которое запрещает аппаратную защиту в режиме программирования.

Рисунок 14.4 иллюстрирует взаимодействие CCB и PCCB с регистром конфигурации кристалла.

Следует очистить CCR.6, чтобы сделать невозможной запись и предотвратить дальнейшее программирование области ПЗУ 2000h–5FFFh. Попытка записи вызывает периодическое повторение цикла записи контроллера шины, однако записи не произойдет.

Следует очистить CCR.7, чтобы сделать невозможным чтение и запретить программе во внешней памяти доступ к содержимому OTPROM. Внутренняя память недоступна для считывания.

Чтение данных может происходить, если вторичный программный счетчик (slave PC) находится в области 2000h–5FFFh. Адрес в Slave PC может быть на 4 байта больше, чем адрес текущей команды. По этой причине команда, находящаяся после 5FFAh, может быть недоступна при защищенной памяти. Векторы прерывания и CCB не являются защищенными от чтения, так как прерывание может происходить, даже если выполнение программы идет из внешней памяти.

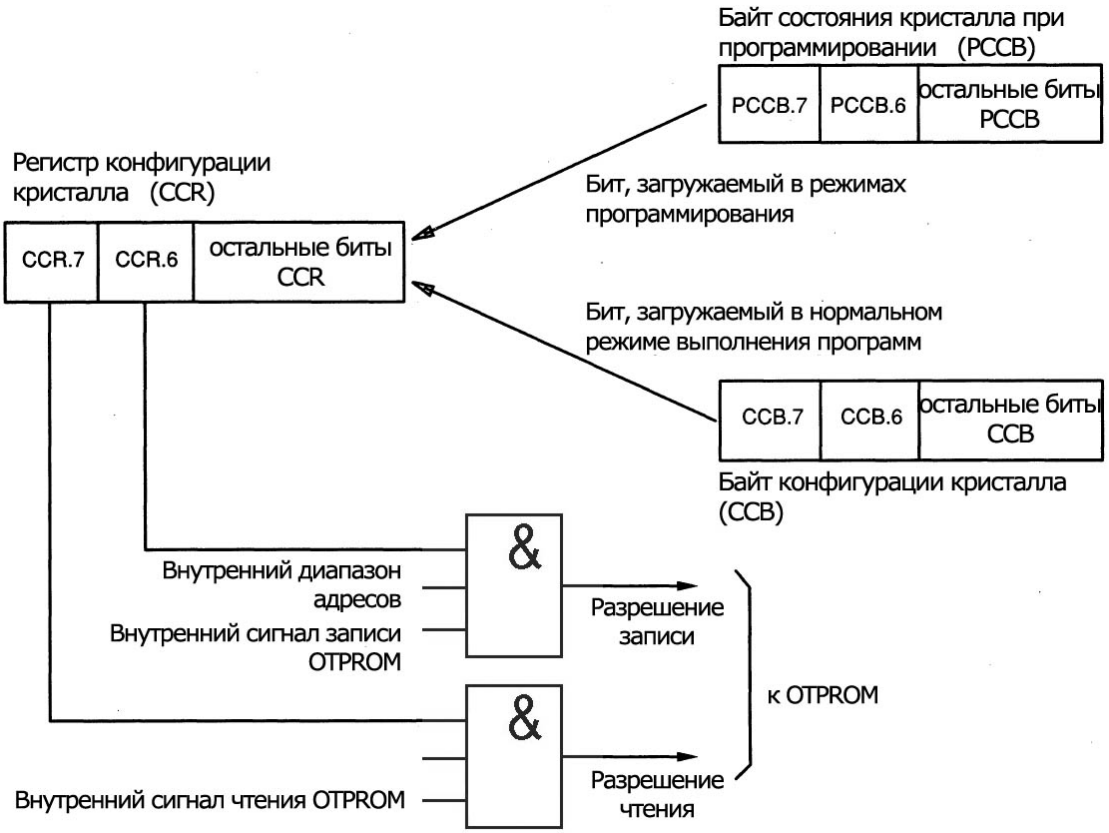


Рисунок 14.4 – Байт конфигурации кристалла

Если вводится код защиты (биты CCR.6 или CCR.7), некоторые режимы программирования потребуют проверки ключа защиты до начала выполнения и не будут работать. Для получения большей информации о битах защиты в каждом режиме следует обратиться к таблице 14.4. В ней приведены все варианты защиты.

Таблица 14.4 – Варианты защиты

PCCB.6	CCB.6	Варианты защиты записи
1	1	Нет защиты.
1	0	Режим RUN-TIME не допускается, остальные режимы допускаются после проверки ключа защиты.
0	1	Допускается только RUN-TIME.
0	0	Программирование невозможно.
PCCB.7	CCB.7	Варианты защиты чтения
1	1	Нет защиты.
1	0	Программирование возможно после проверки ключа защиты. Чтение внутренней OTPROM невозможно, если выполнение программы идет извне.
0	1	Программирование доступно.
0	0	Все режимы программирования доступны после проверки ключа. Чтение внутренней памяти невозможно, если выполнение программы идет извне.

14.6.1.1 Режим программирования PCCB

Можно программировать CCB точно так же, как и любые другие ячейки OTPROM, используя режимы AUTO или RUN-TIME. Вы можете программировать PCCB во время SLAVE режима или PCCB режимов. Если необходимо задать только PCCB, то можно работать в PCCB режиме.

Следует задать низкий PALE# для начала программирования PCCB. Алгоритм записывает данные из порта 3 в PCCB. Когда запись закончена, происходит проверка.

Устройство выставляет высокий PVER, если все прошло правильно и низкий – в ином случае. Можно повторить программирование. Таблица 14.5 показывает функции сигналов в данном режиме.

Таблица 14.5 – Функции выводов в режимах програмирования PCCB и UPROM

Имя функции	Тип	Описание
PALE# (P2.1)	Вход	Программный ALE. Необходимо очистить PALE#, чтобы начать запись из порта 3 в CCB и PCCB.
PVER (P2.0)	Выход	Проверка правильности программирования. Изменяется после каждого программирующего импульса. Высокий уровень свидетельствует об успешном программировании, а низкий указывает на обнаруженную ошибку.

Войдите в режим PCCB посредством использования стандартной последовательности включения питания (Раздел 14.5.1) после установки следующих значений:

PMODE=0Dh;

Port4=0FFh;

Port3= значение PCCB.

Алгоритм считывает значения порта 3 и 4, а затем программирует значение PCCB запуском 25 отдельных 500 мкс циклов программирования, в соответствующую внутреннюю ячейку. Закончите процедуру выполнением последовательности выключения питания (подраздел 14.5.2).

14.6.2 Биты защиты UPROM

МК имеет два непереустанавливаемых бита UPROM, используемые для дополнительной защиты. Рисунок 14.5 показывает местонахождение битов защиты UPROM USFR.2 и USFR.3, запрещающих выборку внешних данных (DED) и внешних команд (DEI) в регистре специальных функций UPROM (USFR). Следует программировать эти биты записью в соответствующую ячейку в режиме SLAVE или UPROM (таблица 14.6).

Примечание – Программирование этих битов делает динамический анализ ошибок невозможным.

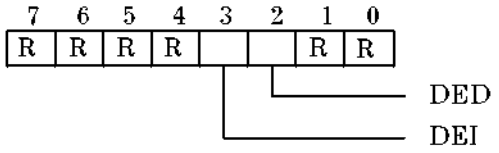


Рисунок 14.5 – Регистр специальных функций UPROM (USFR)

Таблица 14.6 – Карта памяти режима UPROM

Установить этот бит	В этом режиме	Записать это значение	В эту область
DEI	SLAVE	08h	0718h
DEI	UPROM	08h	Порт 3
DED	SLAVE	04h	0758h
DED	UPROM	04h	Порт 3
DEI и DED	UPROM	0Ch	Порт 3

Установка USFR.2 (DEI) запрещает контроллеру шины выборку внешних команд для исполнения. Если установливается этот бит, то предварительная выборка в очередь команд, осуществляемая контроллером шины, запрещает выполнение команд из последних четырех байтов внутренней памяти. Любая попытка загрузить вторичный

программный счетчик внешним адресом приведет к сбросу устройства. Автосброс, кроме того, дополнительно защищает от ухода с программы.

Установка USFR.3 (DED) запрещает контроллеру шины выполнение чтения и записи внешних данных. Любая попытка выборки внешних данных через контроллер шины приводит к автоматическому сбросу.

Можно проверить UPROM биты, чтобы быть уверенными, что они запрограммированы. Но нельзя их стереть.

14.6.2.1 Режим Программирования UPROM

Можно программировать UPROM в режиме SLAVE или режиме UPROM. Если необходимо программировать только UPROM, используется режим UPROM.

Следует задать низкий PALE# для начала режима UPROM. Алгоритм передает данные из порта 3 в UPROM. Когда программирование закончено, происходит проверка. Устройство выставляет высокий PVER, если все прошло правильно, и низкий в ином случае. Можно повторить программирование, подав импульс на PALE#. Таблица 14.6 показывает функции сигналов в данном режиме. Следует войти в UPROM режим при включении питания (подраздел 14.5.1) после установки следующих величин:

PMODE = 9h;

Port 4 = 0FFh;

Port 3 = данные для UPROM.

Алгоритм считывает значения портов 3 и 4, а затем программирует значения UPROM запуском 25 отдельных 500 мкс циклов программирования соответствующей внутренней ячейки.

Следует закончить процедуру выполнением стандартной последовательности выключения питания (подраздел 14.5.2).

14.6.3 Ключ защиты

Ключ – это 128-битная область во внутренней памяти с адреса 2020h до 202Fh. Запишите пароль – ваш ключ защиты – в эти ячейки. Разделы о режимах AUTO, SLAVE, ROM-DUMP содержат информацию о проверке ключа в каждом режиме. Если проверка ключа не пройдена, устройство входит в бесконечный цикл и должно быть сброшено для выхода из него.

14.6.3.1 Программирование ключа защиты

Если во время записи ключа произошел скачок напряжения или сброс, может случайно записаться неизвестный ключ, делающий устройство недоступным для последующего программирования. Чтобы избежать этого в режиме SLAVE, программируют оставшийся массив OTPROM до программирования битов кода защиты CCB (CCB.6 и CCB.7).

Чтобы предотвратить запись случайного ключа в режиме AUTO, необходимо использовать последовательность действий, приведенную ниже. Алгоритм этого режима пропускает все ячейки со значением 0FFh. Поэтому первая последовательность программирует все области, кроме CCB, а вторая – только CCB.

1 Следует выполнить все подпункты, чтобы запрограммировать массив ячеек OTPROM, кроме CCB. Эта последовательность программирует ключ, но не разрешает защиту:

а) записать значение 0FFh во внешнюю ячейку CCB (2018h = 0FFh);

б) установить длительность задающего импульса (PPW) во внешних ячейках 2014h – 2015h;

с) установить код пользователя в соответствующие ячейки внешней EPROM (смотрите таблицу 14.8);

д) запустить алгоритм режима AUTO, рисунок 14.4. Алгоритм пропускает CCB и программирует остальное OTPROM.

2 Выполнить все подпункты, чтобы запрограммировать только CCB (ячейка 2018h). Эта последовательность дает возможность доступа к ключу:

а) установить присваиваемое CCB значение во внешнюю ячейку 2018h;

б) установить длительность задающего импульса (PPW) во внешнюю область 2014h-2015h;

с) записать 0FFh во все остальные ячейки внешней EPROM;

д) запустить алгоритм автоматического режима, рисунок 14.4. За это время алгоритм программирует CCB и пропускает остальные области OTPROM

Примечание - Если быть абсолютно уверенными, что скачка напряжения не произойдет, можно программировать биты кода защиты (CCB.6 и CCB.7) в то же самое время, что и ключ защиты.

14.7 Модифицированный быстрый (QUICK_PULSE) алгоритм

Программа, которая выполняет модифицированный алгоритм, программирует каждую ячейку OTPROM, отправляя 25 отдельных 500 мкс циклов по каждому адресу. После 5-го (25-го) импульса программа верификации сравнивает содержимое ячейки с входными данными.

Алгоритмы режимов AUTO, PCCB и UPROM работают с 5 программирующими импульсами.

Во время SLAVE режима количество импульсов для каждой ячейки задается внешним программирующим устройством. Следует разработать внешнюю программу для выполнения модифицированного QUICK_PULSE алгоритма, как показано на рисунке 14.6.

14.8 Слово-сигнатура

МК 1874ВЕ76Т содержит слово-сигнатуру в ячейке с адресом 0070h. Сигнатура определяет тип прибора. Это слово может быть доступно в SLAVE режиме путем выполнения команды считывания слова (Dump-Word command) из ячейки 0070h. Напряжение программирования определяется чтением ячеек с адресами 0072h и 0073h в SLAVE режиме. Значения слова-сигнатуры и уровни напряжения приведены в таблице 14.7. Значение напряжения программирования вычисляется по следующей формуле:

U = 20 / 256 * (значение в ПЗУ)

1 Начало выдачи таймером программирующих импульсов (PWM).

2 Запись информационных слов в OTPROM. Включение внутреннего программирующего напряжения и начало записи в ячейки.

3 Разрешение прерываний и вход в режим Idle.

4 Ожидание PPW прерывания по окончании программирующего импульса.

5 Чтение счетчика циклов. Если не равно 25, возврат к началу алгоритма. Если равно 25 – продолжение. В режиме программирования SLAVE счетчик цикла = 1.

6 Проверка правильности программирования. Если программирование идет успешно, возврат в основную программу. Если программирование не идет, очистить PVER (низкий уровень P2.0), затем возврат в основную программу.

Рисунок 14.6 – Модифицированный QUICK_PULSE алгоритм

Таблица 14.7 – Слово-сигнатура и уровни напряжения

Описание	Адрес *	Значения
Слово-сигнатура	0070h	879Ch
#VCC1	0072h	40h
V_{PP}	0073h	0A0h
* Ячейка доступна при считывании слов во время режима SLAVE.		

14.9 Режим программирования AUTO

Этот режим является альтернативным режимом, обеспечивающим программирование с минимальными затратами. Можно программировать МК без использования программирующего устройства. Прибор программирует себя сам, выбирая данные из внешнего EPROM ячейки 4000h–7FFFh. Этот режим использует значение длительности задающего импульса (PPW) и модифицированный Quick-Pulse алгоритм (см. подраздел 14.7).

PCCB загружается в регистр состояния схемы (CCR). МК получает данные через внешнюю шину; поэтому PCCB должен соответствовать системе памяти, установленной в режиме программирования, которая не обязательно соответствует системе памяти в разрабатываемом проекте. PCCB рассматривается в подразделе 14.6.

Если биты CCB кода защиты (CCB.6 и CCB.7) активны, проверка ключа происходит в начале работы алгоритма режима AUTO. Если проверка не прошла, устройство входит в бесконечный цикл и должно быть возвращено в исходное состояние через сброс. Программирование ключа защиты описывается в подразделе 14.6.3.1.

Таблица 14.8 – Карта памяти автоматического режима

Адреса внешнего EPROM	Адреса внутреннего OTPROM	Описание
2014h	N/A	Младший PPW бит
2015h	N/A	Старший PPW бит
4000h–7FFFh	2000h–5FFFh	Зарезервированные для данных и команд ячейки
E020h–E02Fh	2020h–202Fh	Ключ во время проверки

Таблица 14.9 – Функции выводов в режиме AUTO

Имя	Тип	Описание
PACT# (P2.7)	Выход	Активность программирования. Нуль указывает на продолжающееся программирование.
PBUS (Порты 3 и 4)	Вход / Выход	Шина адреса/управления/данных. Используется для доступа к внешней памяти.
PVER (P2.0)	Выход	Проверка. Сигнал корректируется после каждого такта. Высокий уровень сигнала указывает на успешное программирование, низкий – на ошибку.
P1.0 – P1.2	0	Линии выбора блока. Формирует старший адрес для внешней памяти (EPROM).

Если при программировании устройства очищается CCB.6 или CCB.7, чтобы разрешить защиту, то алгоритм режима проверяет ключ до начала работы. Сравниваются внешняя область E020h–E02Fh с внутренней 2020h–202Fh. Внешняя и внутренняя области должны совпадать, иначе устройство входит в бесконечный цикл, предотвращающий программирование.

Чтобы войти в режим AUTO, необходимо установить PMODE в 0Ch и подать питание в последовательности, приведенной в разделе 14.5.1. Рисунок 14.7 описывает программу в тестовом ПЗУ.

Для вычисления длительности программирующего импульса (PPW) необходимо воспользоваться алгоритмом, приведенным ниже.

Регистр PPW доступен только в течение автоматического режима; он загружается с внешних адресов 2014h-2015h. PPW должен быть равен по крайней мере 500 мкс для того, чтобы программирование выполнялось корректно. Необходимо использовать следующую формулу для вычисления значения PPW (PPW_VALUE), приведя полученное значение к наибольшему целому.

PPW_VALUE = $(0{,}6944 * f_{OSC}) - 1$,

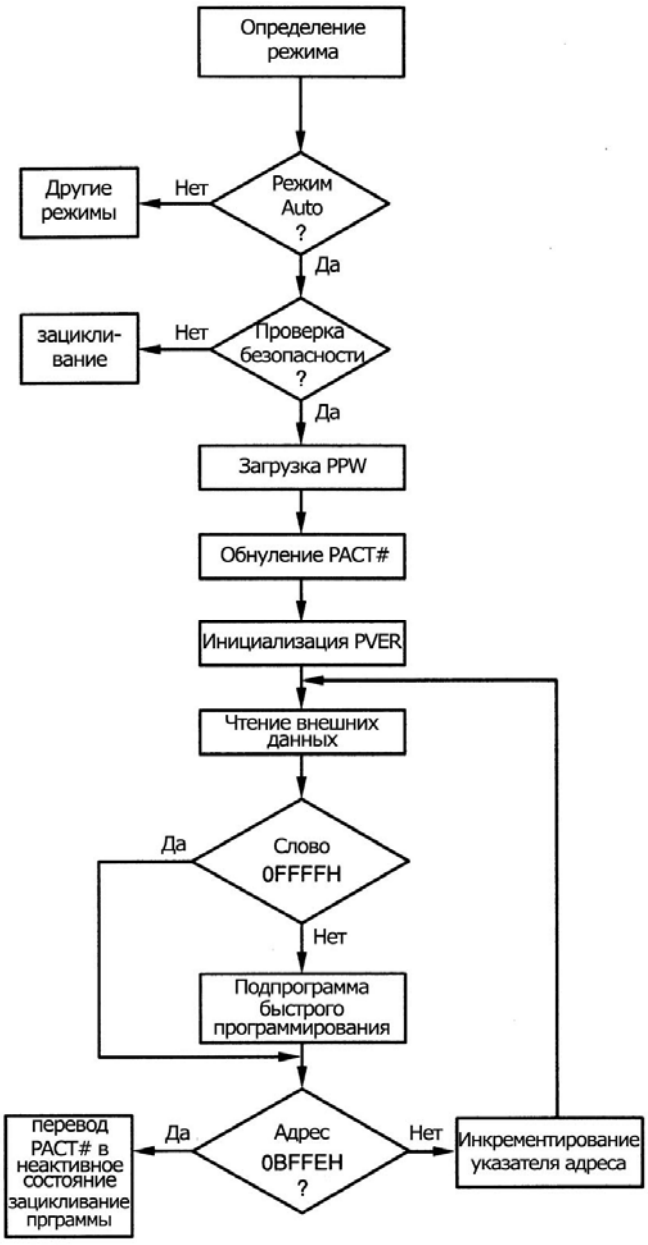
где PPW_VALUE – 15 битное слово,

f_{OSC} – частота XTAL1, в МГц.

Например, XTAL1 – 8 МГц:

PPW_VALUE = (0,6944*8) – 1 = 5,5552 – 114 – 16 = 4,5552 = 5

Ячейка внешней памяти с адресом 2014h загружается младшим байтом PPW, и ячейка 2015h – старшим байтом. В 2015h обычно записывается 80h, однако ячейка может иметь и другое значение. В любом случае старший значащий бит должен быть равен 1. Рисунок 14.8 иллюстрирует пример, приведенный выше.



1 Определение режима программирования (Чтение PMODE).

2 Если выбран режим программирования Auto, продолжение алгоритма. (PMODE = 0Ch).

3 Если ячейки 2020h-202Fh запрограммированы, сравнить ключ защиты с данными во внешней памяти по адресам E020h-E02Fh.

4 Загрузка PPW. Внешняя память по адресам 2014h и 2015h.

5 Обнуление PACT# (низкий уровень P2.7).

6 Инициализация PVER (высокий уровень P2.0).

7 Чтение внешних данных, начинающихся с 4000h.

8 Если слово не 0FFFFh, вызывается подпрограмма быстрого программирования (Modified Quick-Pulse).

9 Смотрите рисунок 14.3.

10 Если адрес = 0BFFEh, то PACT# переводится в неактивное состояние (высокий уровень P2.7) и программа зацикливается. Если адрес не = 0BFFEh, то инкрементируется указатель адреса и процесс программирования продолжается.

Рисунок 14.7 – Алгоритм режима AUTO

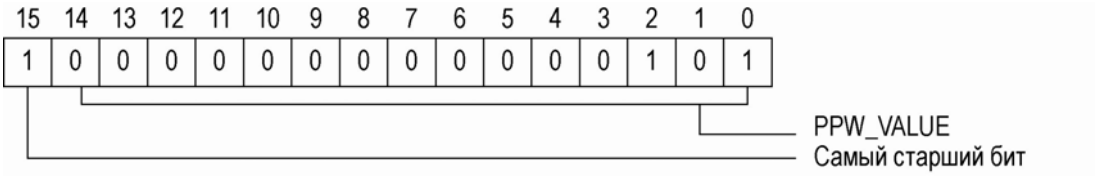


Рисунок 14.8 – Регистр длительности программирующего импульса

14.10 Режим программирования SLAVE

В течение этого режима каждая внутренняя ячейка OTPROM доступна для записи и проверки. Можно использовать этот режим для записи битов защиты UPROM и PCCB.

Существует подрежим, в котором адрес инкрементируется автоматически, и возможно чтение или запись в ячейки ПЗУ, расположенные последовательно, без необходимости передачи адреса по шине для каждого Чтения или Записи. Скорость

процесса увеличивается, так как устраняется необходимость формировать и вводить каждый последующий адрес.

Алгоритм режима SLAVE проверяет биты кода защиты в CCB до выполнения процедур Программирования Слова (Program Word) или Считывания Слова (Dump Word). Если установлена защита записи или чтения, алгоритм проверяет ключ защиты. Если ключ неверен, запись или считывание содержимого любой ячейки запрещены.

14.10.1 Режим SLAVE

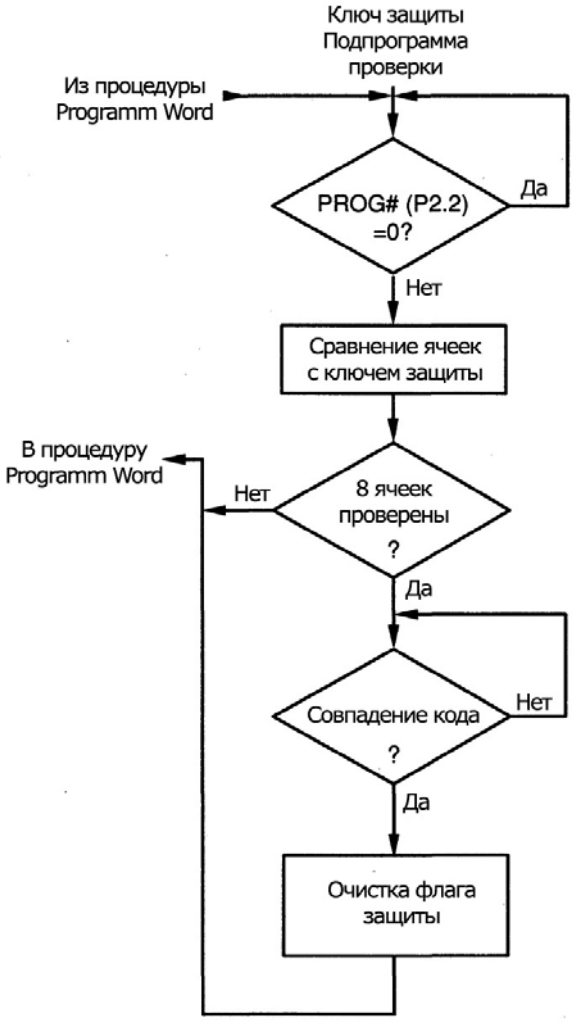
МК в режиме SLAVE реагирует на команду Program Word или Dump Word. Порты (PBUS) служат для передачи адреса, данных и выборки команд. Некоторые из выводов порта 2 обеспечивают квитирование установления связи. Наименьший значащий бит порта 3 выбирает команду Program Word (P3.0 =1) или Dump Word (P3.0 = 0). Старшие 15 битов содержат адрес программируемого или считываемого слова. 16 битов вместе представляют собой комбинацию адреса/команды. Адрес размещается с 2000h по 9FFFh. Таблица 14.10 описывает функции сигналов в режиме SLAVE.

Таблица 14.10 – Функции выводов в режиме программирования SLAVE

Имя	Тип	Описание
AINC# (P2.4)	Вход	Автоматический инкремент. Низкий активный уровень разрешает режим автоматического инкремента. Он позволяет осуществлять чтение или запись последовательных ячеек без передачи каждый раз адреса по шине. В режиме Program Word: AINC# проверяется после записи в каждую ячейку. Если AINC# установлен, адрес инкрементируется и вводится следующее слово данных. В режиме Dump Word: AINC# проверяется после каждого слова, которое было записано в Порты 3 и 4. Если AINC# установлен, адрес инкрементируется и устройство готово к выводу следующего слова данных.
CPVER (P2.6)	Выход	Заключительная проверка программы. Когда работа закончена, высокий уровень показывает, что все ячейки записаны успешно, а низкий уровень указывает на ошибку, случившуюся во время одной из операций.
PALE# (P2.1)	Вход	Используется как разрешающий сигнал для чтения адресов и команд в устройство. PALE# обычно фиксируется на высоком уровне пользователем. PALE# разрешает чтение адресов и команд с 3-го и 4-го портов.
PBUS (Порты 3 и 4)	Вход/ Выход	Шина адреса/управления/данных. Используется для чтения и записи команд, адресов и данных. Порты 3 и 4 используются в режиме ввода-вывода с открытым стоком (не как системная шина). Добавив внешние нагрузочные резисторы, можно считывать данные из устройства во время процедуры Dump Word.

Окончание таблицы 14.10

Имя	Тип	Описание
PROG# (P2.2)	Вход	Начало программирования. PROG# используется как разрешающий сигнал для чтения данных в течение процедуры Program Word. PROG# обычно фиксируется на высоком уровне пользователем. PROG# разрешает чтение данных из Портов 3/4 и программирование внутреннего ПЗУ (OTPROM). Если PROG# остается установленным, в ячейку записываются те же данные в течение двадцати пяти 500 мкс импульсов. По этой причине данные в портах должны оставаться неизменными, пока PROG# установлен. Когда PROG# станет высоким, работа продолжается. PROG# используется как разрешение установления связи для вывода данных через порт в процедуре Dump Word. PROG# устанавливается в высокий уровень пользователем. PROG# разрешает вывод содержимого ячейки ПЗУ (OTPROM) через порт. Когда PROG# станет высоким, работа продолжается.
PVER (P2.0)	Выход	Проверка правильности программирования. Сигнал корректируется после каждого программирующего импульса. Высокий уровень указывает на успешное программирование, низкий - на обнаруживаемую ошибку.



1 Ожидание сброса PROG#

2 Сравнение ячеек с ключом защиты (внутренних с внешними).

3 Повторение цикла, пока не проверены все восемь ячеек с ключами защиты.

4 Если ячейка с ключом защиты имеет не тот код, то устройство входит в цикл, продолжающийся до сброса.

5 Если все ключи защиты верны, флаг защиты очистить и осуществить возврат в основную программу.

Рисунок 14.9 – Подпрограмма проверки ключа защиты

14.10.2 Проверка ключа защиты в режиме SLAVE

Алгоритм программирования в режиме SLAVE проверяет биты кода защиты в CCB (ячейка 2018h) до выполнения процедур Program Word или Dump Word. Если

обнаружено, что вы очистили CCB.6 или CCB.7 (запись или чтение запрещены), следующим шагом будет программирование ячеек ключа защиты. Задайте восемь последовательных слов с 2020h по 202Fh ячейки (когда программирование не требуется, квитирование установления связи происходит аналогично квитированию в процедуре Program Word). Алгоритм сравнивает записываемые величины с содержимым ячеек 2020h–202Fh внутреннего ПЗУ. Если проверка прошла, то предоставляется доступ к устройству, иначе оно зависает и должно быть возвращено в исходное состояние через сброс. Устройство запрещает запись или вывод содержимого любой ячейки, если проверка не прошла.

14.10.3 Алгоритм режима SLAVE

Последовательность включения питания (подраздел 14.5.1) и таблица значений PMODE (таблица 14.1) содержат информацию, необходимую для входа в режим SLAVE. Алгоритм включает три функциональных блока: декодер адрес/команда, процедуры программирования слова (Program Word) и считывания слова (Dump Word).

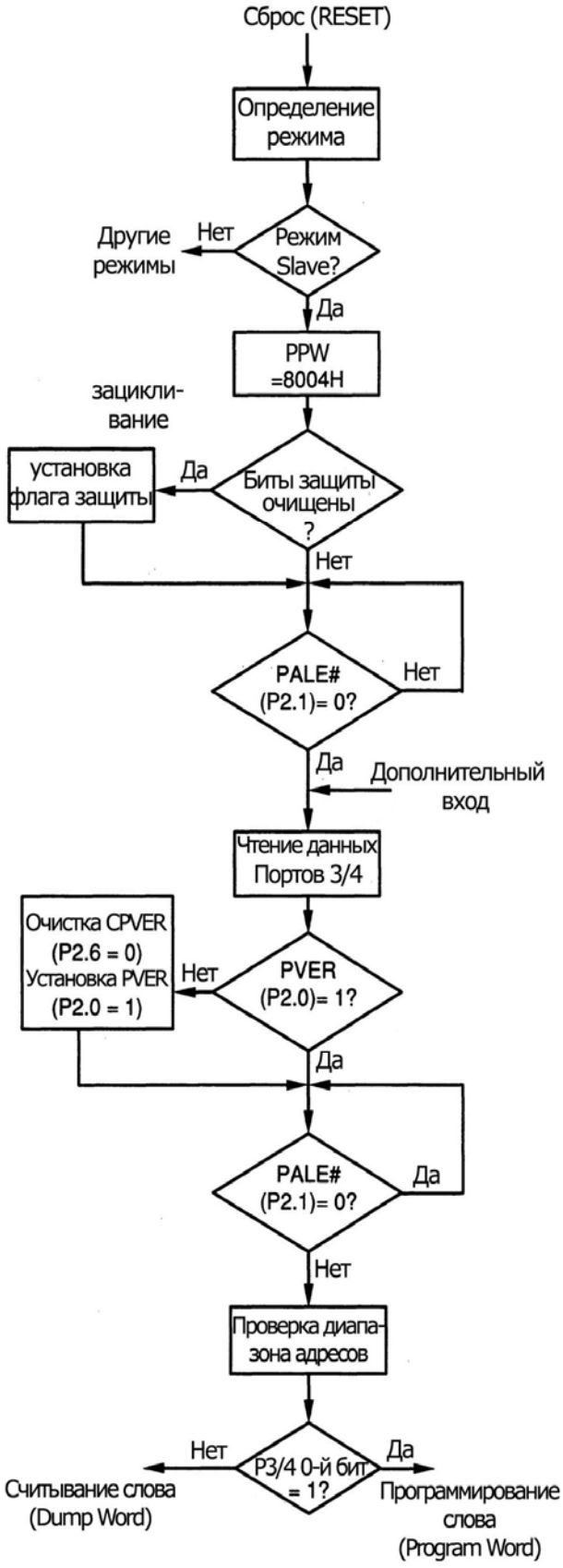
На рисунке 14.10 показана программа дешифрации адреса/данных, на рисунке 14.11 – алгоритм Program Word, на рисунке 14.12 – алгоритм Dump Word.

14.10.4 Команды Program Word и Dump Word

Таблица 14.11 определяет обозначения временных интервалов во временных диаграммах команд Program Word и Dump Word. На рисунках 14.13 и 14.14 приведены временные характеристики этих сигналов.

Таблица 14.11 – Обозначения временных характеристик на временных диаграммах команд Program Word и Dump Word

Обозначение	Описание
Tshll	Сброс высокого уровня до первого низкого PALE#.
Tlllh	Длительность PALE# импульса.
Tavll	Время установки адреса.
Tllax	Время удержания адреса.
Tpldv	Время от низкого PROG# до действительного слова вывода.
Tphdx	Удержание данных слова вывода.
Tdvpl	Время установки данных.
Tpldx	Время удержания данных.
Tplph	Длительность импульса PROG#.
Tphll	Время от высокого PROG# до следующего низкого. PALE#.
Tlhpl	Время от высокого PALE# до низкого PROG#.
Tphpl	Время от высокого PROG# до следующего низкого PROG#.
Tphil	Время от высокого PROG# до низкого AINC#.
Tilih	Длительность импульса AINC#.
Tilvh	PVER удерживается после низкого AINC#.
Tilpl	Время от низкого AINC# до низкого PROG#.
Tphvl	Время от высокого PROG# до действительного PVER.



1 Определение режима программирования (Чтение PMODE).

2 Если выбран режим SLAVE (PMODE=5h), продолжение алгоритма.

3 Установка PPW в 8004h.

4 Если очищен CCB.6 или CCB.7, перед шагом 5 выставляется флаг безопасности.

5 Если PALE# активен, переход к следующему шагу, иначе – повторная итерация.

6 Чтение данных с портов 3/4.

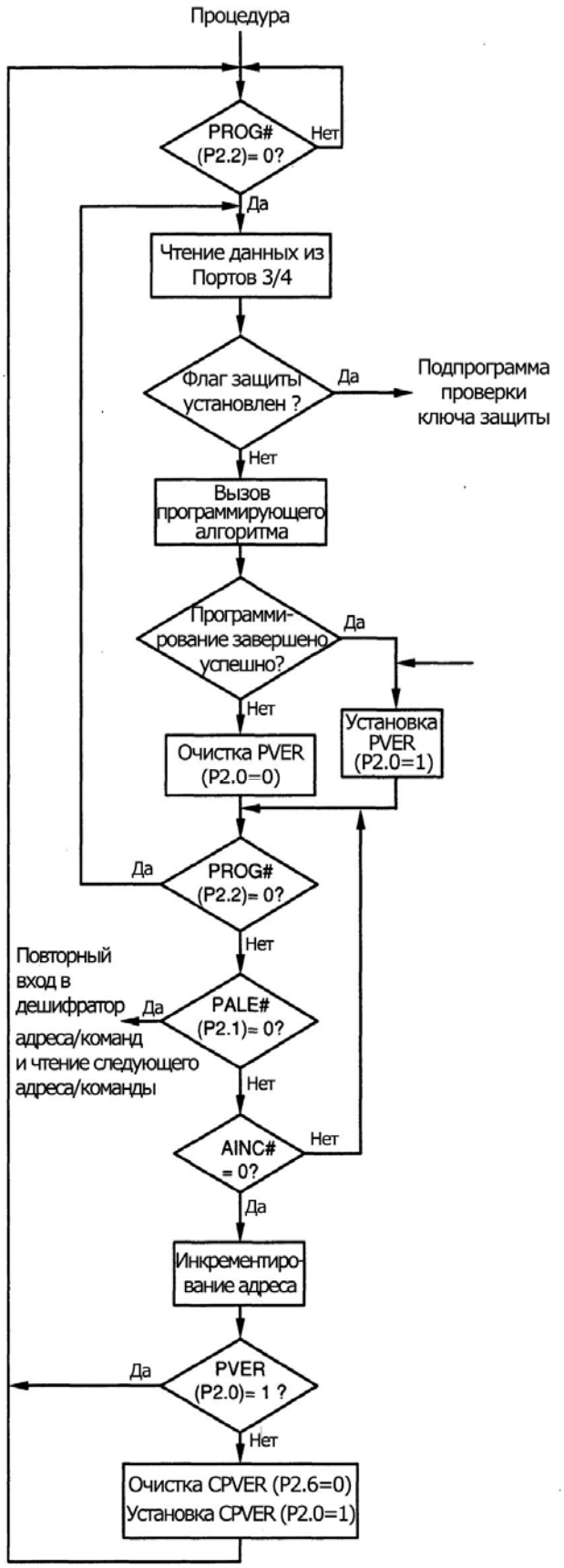
7 Если PVER установлен (ошибки нет), переход к шагу 8. Если PVER не установлен, очистить CPVER, установить PVER, затем переход к шагу 8.

8 Ожидание сброса PALE#.

9 Проверка диапазона адресов портов 3 и 4.

10 Если P3.0=1, переход к процедуре Program Word (режим команд). Если P3.0=0, переход к процедуре Dump Word (режим адреса). Используется слово в портах 3 и 4 как адрес программирования или считывания.

Рисунок 14.10 – Алгоритм дешифратора адреса/данных



1 Ожидание установки PROG# в активный уровень.

2 При установлении PROG# – чтение данных из портов 3 и 4.

3 Если флаг безопасности установлен (см. рисунок 14.10), переход к процедуре верификации ключа безопасности. Если флаг очищен, переход к шагу 4.

4 Вызов алгоритма программирования (см. рисунок 14.6).

5 Если верификация пройдена, установка PVER, иначе – очистка PVER. Этот шаг отрабатывается для каждого из двадцати пяти 500-микросекундных импульсов программирования слова.

6 Удержание PROG# в активном уровне пока не завершатся двадцать пять циклов, затем сброс PROG#. Альтернативный способ – двадцать пять импульсов PROG# при PALE# = 1 и AINC# = 1.

7 Если PALE# активен, повторный вход в процедуру дешифрации адреса/команд и чтение следующего адреса/команды.

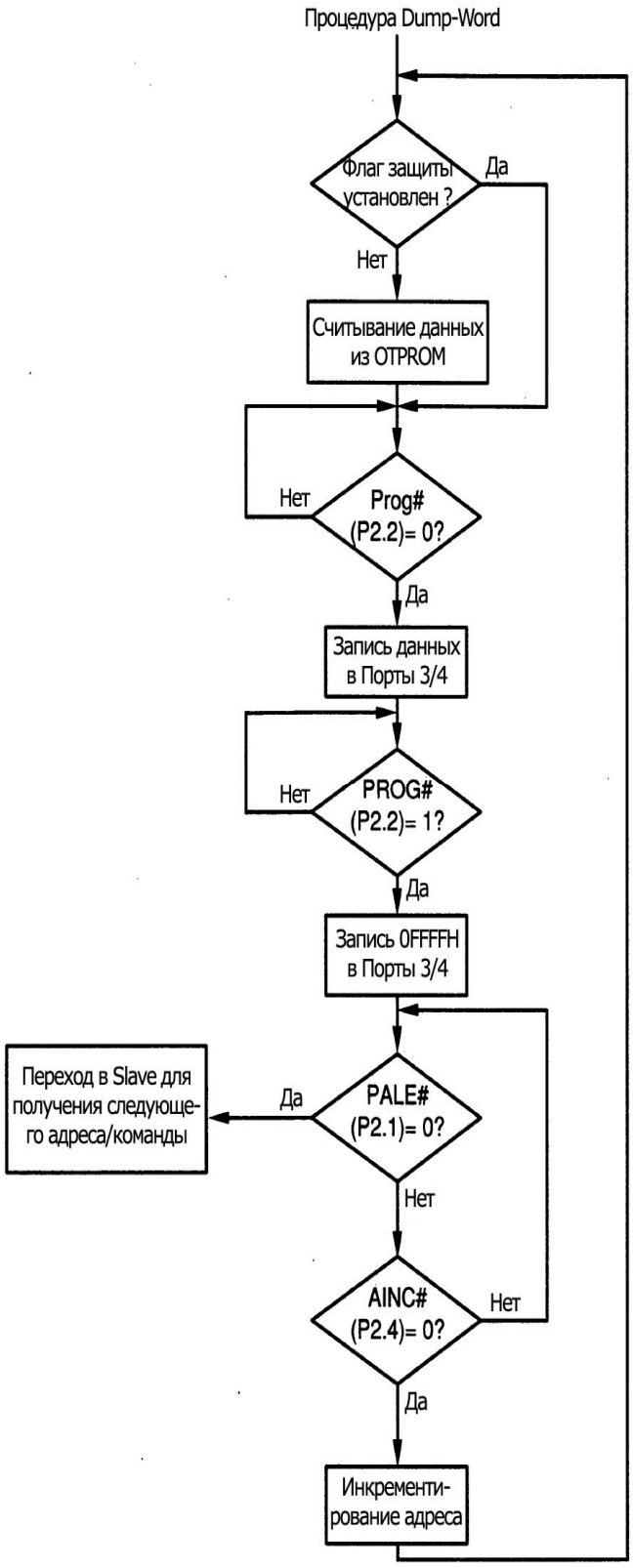
8 Если PALE# сброшен, проверка AINC#.

9 Если AINC# активен, адрес инкрементируется на два, затем верификация.

10 Если PVER установлен (нет ошибки), повторный вход в процедуру Program Word.

11 Если PVER сброшен (ошибка верификации), очистка CPVER, установка PVER, затем повторный вход в процедуру Program Word.

Рисунок 14.11 – Процедура Program Word



1 Если флаг безопасности установлен (см. рисунок 14.10), переход к шагу 3.

2 Чтение данных из OTPROM.

3 Ожидание установки PROG# в активный уровень.

4 При установлении PROG# – передача данных через порты 3 и 4.

5 Ожидание сброса PROG# .

6 Запись всех единиц в порты 3 и 4. (Это позволяет перевести выводы портов в третье состояние, избежав конфликта с процедурами Program Word и дешифрации адреса/команд, использующими порты 3 и 4 как входы.)

7 Если PALE# установлен, возврат в процедуру дешифрации адреса/команд. Если PALE# сброшен, чтение AINC#.

8 Если AINC# не установлен, возврат к шагу 7.

9 Если AINC# установлен, адрес инкрементируется на два, затем повторный вход в эту процедуру.

Рисунок 14.12 – Процедура Dump Word

Рисунок 14.13 показывает временные диаграммы команды Program Word с повторением импульсов программирования и автоинкрементом. Единица на P3.0 выбирает команду Program Word. Командой 3501h программируется ячейка слова, расположенная по адресу 3500h. Устройство получает входной сигнал PALE#, что показывает активность команды; установка PALE# фиксирует команду и адрес в портах 3 и 4 (PBUS). Установка PROG# фиксирует данные в Портах 3 и 4 для начала

программирования; устройство считает или выводит слово. Длительность импульса PROG# определяет длительность программирующего импульса (режим Slave не использует PPW). По нарастающему фронту сигнала PROG# идет автоматическая проверка содержимого только что запрограммированной ячейки. Установка сигнала PVER# показывает успешное программирование. AINC# служит для автоматического инкремента адреса..

Рисунок 14.14 иллюстрирует команду Dump Word. Ноль на P3.0 выбирает команду Dump Word. Посылка команды “2100h” помещает слово из адреса внутренней памяти 2100h в Порты 3 и 4. PROG# управляет подключением устройства к шине. Временные диаграммы команды Dump Word аналогичны приведенным на рисунке 14.7. В режиме Dump Word сигнал на выводе AINC# может оставаться активным или переключаться из одного состояния в другое. Вывод PROG# автоматически инкрементирует адрес.

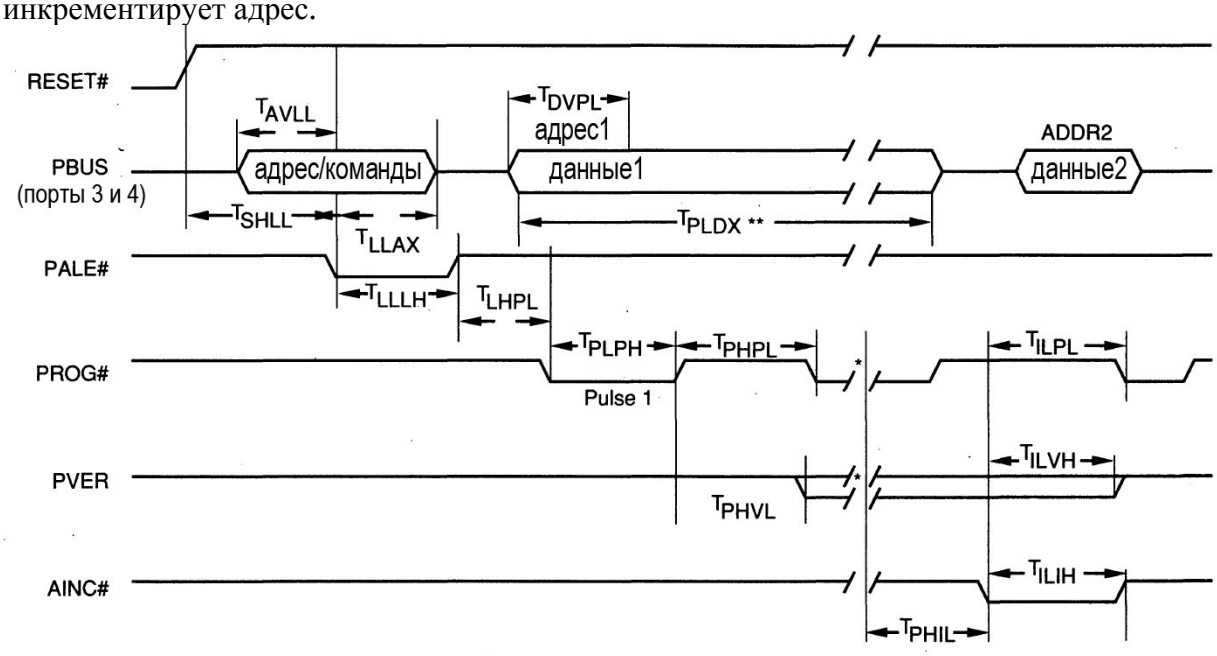


Рисунок 14.13 – Временные диаграммы в режиме Program Word

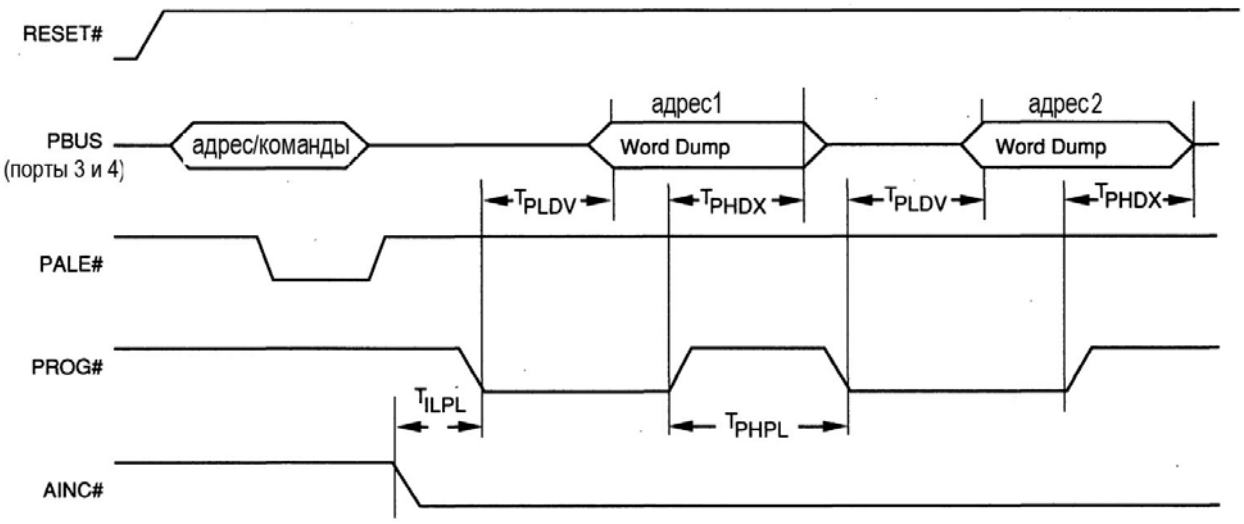


Рисунок 14.14 – Временные диаграммы режима Dump Word

14.11 Режим ввода содержимого памяти (ROM-DUMP)

Этот режим легко позволяет проверить содержимое OTPROM. Последовательность включения питания (раздел 14.5.1) и таблица значений PMODE (таблица 14.1) помогут войти в режим ROM-DUMP.

После проверки ключа защиты режим записывает всю область внутренней памяти во внешнюю память. Ключ защиты во внутренней памяти должен соответствовать при проверке ключу защиты в соответствующей области внешней памяти. Если ключ не совпадает, устройство входит в бесконечный цикл и может быть выведено из него только сбросом. Если USFR.2 (бит запрета выборки внешних данных DED) запрограммирован, переход в режим ROM-DUMP запрещен. Подраздел 14.6 дает более полную информацию о ключе защиты и программировании UPROM).

Таблица 14.12 – Карта памяти режима ROM-DUMP

Устройство	Адреса внутреннего OTPROM	Адреса внешней памяти
МК	2000h – 5FFFh	2000h – 5FFFh
МК Ключ защиты	2020h – 202Fh	2020h – 202Fh

14.12 Режим RUN-TIME

Можно программировать ячейки OTPROM во время обычного выполнения программы. Чтобы сделать OTPROM доступной, необходимо установить на EA# высокий уровень во время сброса устройства. Затем просто записать информацию в ячейку, которую необходимо запрограммировать. Подать напряжение программирования на вывод Vpp на время всего процесса программирования.

Непосредственно после записи в OTPROM устройство должно либо войти в режим IDLE, либо выполнить команду из внешней памяти; доступ к OTPROM прекратит текущий цикл программирования. Каждый цикл программирования начинается, когда слово записывается в OTPROM и заканчивается, когда происходит следующее обращение к OTPROM. Каждое слово требует двадцать пять программных циклов. Длительность каждого цикла должна быть не менее 500 мкс.

```
Program:     POP ADDRESS_TEMP                        ;load program data
             POP DATA_TEMP                           ;and address
             PUSHF
             LD COUNT, #5t                           ;program using
                                                     ;Modified Quick-
                                                     ;Pulse
LOOP:        LDB INT_MASK, #ENABLE_SWT               ;program SWT for
             LDB HSO_COMMAND, #SWT0_OVF              ;program pulse width
             ADD HSO_TIME, TIMER1, #PROGRAM_PULSE
             EI
             ST DATA_TEMP, [ADDR_TEMP]               ;enter idle mode until
             IDLPD 1                                 ;SWT expires
             DJNZ COUNT, LOOP                        ;loop 5 times
             POPF
             RET

SWT_EXPIRED:                                         ;service SWT
             RET                                     ;and return
```

15 Рекомендации по отладочным средствам ИМС

15.1 Программаторы для программируемого варианта ИМС

Могут использоваться программаторы любой фирмы, обеспечивающие программирование аналогов разработанной ИМС 1874ВЕ76Т (TN87C196KC-20 фирмы Intel), например, типа PicProg+, ChipProg, ChipProg+ ООО «Фирма Фитон», г. Москва.

15.2 Описание инструментальных средств для ИМС

Для проектирования и отладки систем на основе микроконтроллеров могут использоваться инструментальные средства, поставляемые ООО «Фирма Фитон».

Ниже приводится краткая информация о средствах отладки, поставляемых ООО «Фирма Фитон», г. Москва, обеспечивающих отладку систем на базе разработанных микроконтроллеров.

Реквизиты ООО «Фирма ФИТОН»:

- Адрес: Россия, Москва, 127015, ул. Новодмитровская, д. 5а, 11 этаж, оф. 1103
- Тел/факс: (495) 730-75-84 (многоканальный)
- E-mail: PHYTON@phyton.ru
- Интернет-телефон **(**Skype**):** phytonmsk

15.3 Интегрированный пакет разработки и отладки систем на базе микроконтроллеров. (Project-96)

Пакет Project-96 – набор программно-аппаратных средств, предназначенный для разработки и отладки систем на базе микроконтроллеров семейства MCS-196 фирмы Intel, являющихся функциональными аналогами изделий.

Концепция Project-96 – объединение внутрисхемного эмулятора, программного отладчика-симулятора, компиляторов, текстового редактора, менеджера проектов и программатора в рамках единой интеллектуальной среды разработки.

При наличии одного из программаторов PicProg+, ChipProg, ChipProg+ пакет поддерживает работу и с программатором. Программный интерфейс пакета унифицирован и поддерживает все этапы разработки программного обеспечения – от написания исходного текста программы до ее компиляции и отладки.

Пакет Project-96 ориентирован на отладку программ на языке высокого уровня по исходному тексту. Встроенные многооконный редактор, менеджер проектов и большое количество сервисных возможностей существенно облегчают труд разработчика, избавляя его от рутинных инструкций.

Редактор предназначен для написания исходных текстов программ и поддерживает редакцию с блоками текста, поиск/замену, цветовое выделение синтаксических конструкций языка Си и ассемблера.

Встроенный менеджер проектов поддерживает автоматическую компиляцию программ, написанных для компилятора Си и ассемблера. Переход от редактирования исходного текста к отладке и обратно происходит прозрачно, т. е. менеджер проектов автоматически запускает компиляцию изменившихся исходных текстов, активизирует отладчик, осуществляет загрузку программ.

Полная конфигурация пакета называется Project-96/ESCA и включает в себя:

- Менеджер проектов.
- Кросс-компилятор языка ассемблер МСА-96.
- Кросс-компилятор языка «Си» МСС-96.
- Отладчик-симулятор PDS-96.

15.3.1 Внутрисхемный эмулятор PICE-196

PICE-196 - эмулятор нового поколения, созданный с применением новых технологий разработки аппаратуры и программного обеспечения. Применение программируемых матриц большой емкости позволило значительно сократить размеры

эмулятора без какого-либо ущерба его функциональным возможностям, свести к минимуму отличия в электрических и частотных характеристиках эмулятора от характеристик эмулируемого процессора и, тем самым, добиться максимальной точности эмуляции на частотах до 20 МГц при напряжениях питания от 4,5 В до 5,5 В.

Программная поддержка PICE-196 работает в среде Windows-95/98/ME/NT/2000/XP и предоставляет пользователю обширный сервис как по разработке программ, так и по их отладке.

Эмулятор состоит из основной платы размером (85×82) мм, сменного пода под определенную группу процессоров и сменного адаптера под конкретный тип корпуса. На основной плате реализованы: трассировщик, процессор точек останова. Плата сменного пода содержит эмулирующий процессор под конкретный тип микроконтроллера. Сменные адаптеры обеспечивают установку эмулятора в колодки PLCC на плате пользователя. Питание эмулятора осуществляется от блока питания 5,0 В, 0,5 А или непосредственно от отлаживаемого устройства. Связь с компьютером – по гальванически развязанному каналу RS-232C на скорости 115К бод.

Характеристики аппаратуры

- Точная эмуляция – отсутствие каких-либо ограничений на использование программой пользователя ресурсов микроконтроллера.
- До 1М байт при использовании основной платы М-196/Х (до 128К байт при использовании основной платы М-196) эмулируемой памяти программ и данных. Поддержка банкированной модели памяти. Распределение памяти между эмулятором и устройством пользователя с точностью до 1-го байта.
- Аппаратная поддержка для отладки программ на языках высокого уровня.
- Трассировка 8 произвольных внешних сигналов.
- Выходы синхронизации аппаратуры пользователя.
- Трассировщик реального времени с буфером объемом 16К фреймов по 64 бита с доступом "на лету". Трассировка адреса, данных, сигналов управления, таймера реального времени и 8-ми внешних сигналов пользователя.
- Программируемый фильтр трассировки.
- Аппаратный процессор точек останова с возможностью задания сложного условия останова эмуляции по комбинации сигналов адреса, данных, управления, 8-ми внешних сигналов, таймера реального времени, счетчиков событий и таймера задержки.
- Четыре комплексных точки останова, которые могут быть использованы независимо или в комбинациях по условиям AND/OR/IF-THEN.
- 48-разрядный таймер реального времени.
- Прозрачная эмуляция – доступ "на лету" к эмулируемой памяти, точкам останова, процессору точек останова, буферу трассировки, таймеру реального времени.
- Управляемый генератор тактовой частоты для эмулируемого процессора. Возможность плавного изменения тактовой частоты от 500 кГц до 20 МГц.
- Гальванически развязанный от компьютера канал связи RS-232C со скоростью обмена 115К бод.
- Встроенная система самодиагностики аппаратуры эмулятора.

Характеристики программного обеспечения

- Программное обеспечение работает в среде Windows-95/98/ME/NT/2000/XP.
- Поддерживается разработка программ на уровне ведения проектов для макроассемблера МСА-96 и Си-компилятора МСС-96 «Фирмы Фитон», а также для пакетов кросс-средств языка «Си» и ассемблера фирм Intel и Tasking Software. Помимо указанных пакетов, поддерживается полнофункциональная символьная отладка программ, созданных с помощью компилятора фирмы IAR Systems.
- Автоматическое сохранение и загрузка файлов конфигурации аппаратуры, интерфейса и опций отладки. Обеспечивается совместимость файлов конфигурации с симулятором PDS-96. Обеспечена переносимость проектов между эмулятором PICE-196 и симулятором PDS-96.

- Возможность настройки цветов, шрифтов и других параметров для всех окон одновременно и для каждого окна в отдельности.
- Обновление версий PICE-196 осуществляется обновлением его программного обеспечения.

Наименования компонентов эмулятора PICE-196

Для эмулятора PICE-196 существует два варианта основной платы, различающихся по скорости, объему памяти и, соответственно, по цене. Каждый вариант имеет свое наименование: M-196 или M-196-X. Минимальные параметры и цену обеспечивает базовая для PICE-196 плата M-196.

Название ПОДа состоит из следующих символов (слева направо): "POD" - указывает, что это ПОД; "196" - обозначает семейство микроконтроллеров.

Название адаптера состоит из следующих символов (слева направо): "ADP" - указывает, что это адаптер; "196" - обозначает семейство микроконтроллеров; "LCC" - характеризуют тип корпуса эмулируемого микроконтроллера ("LCC" - обозначает PLCC); "52", "68" и "84" - указывают число выводов корпуса.

Комплект поставки эмулятора PICE-196

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

15.4 Отладчик-симулятор микроконтроллеров семейства MCS-196 PDS-96

PDS-96 – это интегрированный комплекс профессиональных средств для разработки систем на базе семейства микроконтроллеров MCS-196 фирмы Intel, включающий среду разработки, макроассемблер, отладчик-симулятор, примеры программ и проектов, мощную систему контекстной помощи, электронные гипертекстные руководства по всем компонентам пакета, а также краткое руководство пользователя в печатном виде. PDS-96 работает в среде Windows-95/98/ME/NT/2000/XP.

С помощью PDS-96 можно эффективно разрабатывать и отлаживать программы, используя не только входящий в комплект макроассемблер MCA-96, но и Си-компилятор MCC-96 «Фирмы Фитон», а также кросс-средства фирм Intel и Tasking Software, для которых также предоставляется возможность разработки программ на уровне ведения проектов. Помимо указанных пакетов, PDS-96 обеспечивает полнофункциональную символьную отладку программ, созданных с помощью пакета кросс-средств фирмы IAR Systems. Пользователю предоставляется обширный сервис по выполнению отлаживаемой программы в различных режимах, манипуляции различными типами точек останова, просмотру и модификации состояния ресурсов микроконтроллера. Поддерживается отладка программ по исходному тексту, а также просмотр и изменение значений сложных объектов языка высокого уровня – массивов, структур, указателей.

Среда разработки программ PDS-96 интегрирует в себе средства, используемые при разработке программ для микроконтроллеров MCS-196. Обеспечивается интерактивная поддержка всех этапов разработки от написания исходного текста до зашивки готовой программы в ПЗУ микроконтроллера, а именно:

- написание исходных текстов программ с помощью встроенного многооконного редактора;

- настройка опций кросс-средств, используемых для компиляции программы (ассемблера, компилятора Си, линкера, библиотекаря). Настройка производится с помощью диалогов, снабженных контекстной справочной информацией;
- компиляция и линковка программы. Если компилятор обнаруживает ошибки в исходном тексте программы, то строка с ошибкой в окне редактора подсвечивается, и ошибки можно сразу же исправить;
- отладка программы;
- "зашивка" программы в ПЗУ микроконтроллера.

"Интегрированность" среды PDS-96 проявляется в том, что перечисленные этапы разработки связываются в одно целое. Самые трудоемкие этапы, а именно компиляция/линковка с диагностикой и исправлением ошибок, максимально упрощены. PDS-96 самостоятельно следит за изменениями, которые вносятся в исходные тексты программ. Например, исправив ошибку в исходном тексте, можно нажатием одной кнопки выполнить программу "до курсора", заставить PDS-96 перетранслировать изменившиеся модули, загрузить полученную программу в память отладчика и запустить ее до указанной строки. Переход от отладки к редактированию происходит так же прозрачно и быстро.

Отладочные возможности PDS-96

Симулятор PDS-96 представляет собой программно-логическую модель микроконтроллера, имитирующую (симулирующую) работу всех его узлов – памяти, АЛУ, системы команд, регистров и т. д. Возможности PDS-96:

- отслеживание выполнения программы по ее исходному тексту;
- просмотр и изменение значений любых переменных;
- встроенный анализатор эффективности программного кода;
- точки останова по сложному условию;
- неограниченное количество точек останова по доступу к ячейкам памяти;
- просмотр стека вызовов подпрограмм и функций;
- встроенный строчный ассемблер;
- возможность выполнения программы "назад" на большое количество шагов, а также в непрерывном режиме. При этом состояние модели микроконтроллера полностью восстанавливается;
- точный подсчет интервалов времени и многое другое.

Основные достоинства программно-логической модели микроконтроллера, реализованной в PDS-96 – точная симуляция узлов микроконтроллера и возможность моделировать устройства, подключенные к микроконтроллеру "снаружи" (т. н. моделирование внешней среды), например, внешнюю логику, датчики, клавиатуру, исполнительные устройства (дисплеи), задавать периодические и непериодические воздействия и т. п.

15.5 Кросс-макроассемблер MCA-96

Кросс-макроассемблер MCA-96 предназначен для трансляции исходных текстов программ для процессоров семейства MCS-196 фирмы Intel:

- поддерживает все микроконтроллеры MCS-196 фирмы Intel;
- генерирует HEX-файл и подробный листинг;
- поддерживает широкий набор директив условной трансляции;
- предоставляет удобные средства работы с макросами;
- генерирует подробную символьную информацию для отладчиков;
- допускает использование русских букв в именах;
- поддерживает 32-битные арифметические и логические выражения;
- выполняет проверку перекрытия кода;
- выполняет проверку размещения данных в запрещенных областях;
- включает полный набор include-файлов;

- поставляется как в составе пакета Project-96, так и отдельно.

Макроассемблер МСА-96 поддерживает процессоры семейства Intel MCS-196, включая кристаллы с 24-битной адресацией. Имеется возможность расширять номенклатуру поддерживаемых процессоров без обновления версии ассемблера. В комплект входит набор включаемых файлов, содержащий определения регистров спецназначения для всех ОЭВМ MCS-196 фирмы Intel.

Использование русских букв в именах позволяет создавать исходные тексты программ, обладающие превосходной читаемостью. Генерируется подробный листинг, включающий не только текст программы и адреса инструкций, но также и таблицы символов, макросов, констант и т. п. с указанием имен, к которым не было ссылок в программе.

15.6 Кросс-компилятор языка Си МСС-96

Кросс-компилятор языка Си МСС-96 предназначен для трансляции исходных текстов программ для процессоров семейства MCS-196:

- поддерживаются все кристаллы MCS-196;
- соответствует стандарту ANSI/ISO 9899-1990;
- генерирует быстрый код;
- три модели памяти;
- встроенный ассемблер;
- поддержка всей специфики MCS-196 из «Си»;
- быстрая библиотека функций с плавающей точкой;
- большая библиотека стандартных функций (более 120);
- функции для работы с потоками ввода-вывода;
- поддержка для динамически распределяемой памяти;
- поставляется в составе Project-96.

«Си» – исключительно гибкий язык, реализующий концепцию структурного программирования и обладающий богатым набором инструкций. В «Си» удачно совмещены как высокоуровневые абстракции – модульность, процедурность, читабельность исходного текста, так и низкоуровневые средства – работа с абсолютными адресами, встроенный ассемблер, работа с битами. Кроме этого, «Си» позволяет получить эффективно работающий код. Именно эти особенности делают «Си» идеальным для встроенных приложений, где требуется доступ ко всем ресурсам процессора при наличии высокоуровневого синтаксиса. Выполнен в соответствии со стандартом ANSI, поэтому можно в полной мере пользоваться свойством переносимости «Си»-программ, используя уже готовые и отлаженные алгоритмы. МСС-96 поддерживает все ОЭВМ Intel MCS-196, включая кристаллы с 24-битной адресацией. Для полного использования всех возможностей MCS-196 в язык введены необходимые расширения. Встроенный ассемблер дает возможность написания макросов с параметрами на ассемблере и их использования в качестве inline-функций. Все особенности архитектуры MCS-196 поддерживаются непосредственно из «Си». Например, подпрограммы обслуживания прерываний можно писать на «Си». Вся поддержка такого рода реализована при помощи стандартных директив #pragma, поэтому получающийся исходный текст хорошо переносим на другие типы процессоров.

Библиотека компилятора оптимизирована для исполнения на ОЭВМ MCS-196, и содержит более 120 функций, включая инструкции с потоками, форматированный ввод-вывод и поддержку для динамически распределяемой памяти (HEAP). Благодаря оптимизированным алгоритмам, инструкции над числами с плавающей точкой производятся в 3-4 раза быстрее, чем у компиляторов фирм Intel и Tasking, причем без потери точности. В комплект компилятора входит набор включаемых файлов, содержащий определения регистров специальных функций для всех ОЭВМ семейства MCS-196, в том числе и для ИМС 1874ВЕ06Т, 1874ВЕ76Т, 1874ВЕ05Т.

16 Заключение

В настоящем руководстве КФДЛ.431295.019 приведено подробное описание архитектуры, функционального построения, системы команд и особенностей применения ИМС 1874ВЕ76Т, 1874ВЕ06Т и 1874ВЕ05Т, которые представляют собой СБИС однокристальных 16-разрядных микроконтроллеров с тактовой частотой 20 МГц, ОЗУ (488×8) бит, 8-канальным 10-разрядным АЦП, 3-канальным ШИМ и портом ввода-вывода, в том числе с внутренней памятью программ (типа OTP EPROM для ИМС 1874ВЕ76Т) объемом (16К×8) бит (функциональные аналоги TN87C196KC-20, TN80C196KC-20 фирмы Intel), а также стойкий к специальным видам внешних воздействующих факторов 16-разрядный микроконтроллер (1874ВЕ05Т) с системой команд микроконтроллера 1874ВЕ36.

Микроконтроллеры предназначены для применения в системах управления, для управления робототехническими комплексами, в системах автоматизации технологических процессов, в системах автоматизированного управления электроприводом, телекоммуникационной и другой технике.

Все значения электрических параметров ИМС приведены в АЕЯР.431280.346 на изделия 1874ВЕ76Т, 1874ВЕ06Т и в АЕЯР.431280.575 на изделие 1874ВЕ05Т; значения параметров, приведенные в настоящем руководстве, являются справочными.

КФДЛ.431295.019 может служить практическим пособием по применению микроконтроллеров для разработчиков систем на основе ИМС 1874ВЕ76Т, 1874ВЕ06Т, 1874ВЕ05Т и программистов.

Приложение А

(обязательное)

Система команд МК 1874ВЕ76Т (1874ВЕ06Т, 1874ВЕ05Т)

Это приложение даёт справочную информацию о системе команд МК. В нём описана каждая команда, показаны соотношения между командами и флагами PSW, показаны шестнадцатиричные коды команд, длины команд и время их выполнения.

Таблица А.1 определяет переменные, используемые в таблице А.2 для замены операндов команд.

В таблице А.2 приведён список команд в алфавитном порядке и описание каждой из них.

Таблицы А.3 и А.4 определяют аббревиатуру и символы, используемые в таблицах А.5 и А.6.

Таблица А.5 показывает влияние каждой команды на флаги PSW, а таблица А.6 показывает влияние флагов PSW или соответствующих битов регистра на команды условного перехода.

В таблице А.7 приводится список шестнадцатиричных кодов команд по порядку, вместе с соответствующими обозначениями команд. Таблица А.8 – карта кодов команд МК.

В таблице А.9 приводится список длин и кодов для каждого используемого режима адресации.

В таблице А.10 приводится время выполнения команд, измеряемых в машинных тактах.

В таблице А.11 приводится время выполнения PTS-циклов, измеряемых в машинных тактах.

Таблица А.1 – Переменные, используемые в операндах

Переменные	Описание
aa	Двухбитное поле внутри кода команды, определяющее основной используемый режим адресации. Это поле имеется только в кодах, в которых проводится выбор режима адресации. Поле декодируется следующим образом: 0 0 Прямая регистровая 0 1 Непосредственная 1 0 Косвенная 1 1 Индексная
baop	Однобайтный операнд, адресуются к которому любым способом.
bbb	Трёхбитное поле внутри кода команды, определяющее выбор специфичного бита внутри регистра.
bitno	Трёхбитное поле внутри кода команды, определяющее выбор одного из 8-и битов внутри байта.
breg	Однобайтный регистр во внутреннем Регистровом Файле. Когда не ясно, ссылается ли эта переменная на источник или приёмник, переменная помечается соответственно S или D.
cadd	Адрес в программном коде.
Dbreg *	Однобайтный регистр во внутреннем Регистровом Файле, использующийся как операнд-приемник в команде.
disp	Смещение. Расстояние между концом команды и целевой меткой в программе.
Dwreg *	Регистр слова (двухбайтный) во внутреннем Регистровом Файле, использующийся как операнд-приемник в команде. Должен выравниваться по адресу, делящемуся нацело на 2.
Ireg	32-битный регистр во внутреннем Регистровом Файле. Должен выравниваться по адресу, делящемуся нацело на 4.
Sbreg *	Однобайтный регистр во внутреннем Регистровом Файле, обслуживающий операнд-источник в команде.
Swreg *	Регистр слова (двухбайтовый) во внутреннем Регистровом Файле, обслуживающий операнд-источник в команде. Должен выравниваться по адресу, кратному 2.
waop	Двухбайтный операнд, к которому адресуются любым способом.
wreg	Двухбайтный регистр во внутреннем Регистровом Файле. Когда не ясно, ссылается ли эта переменная на источник или приёмник, переменная помечается соответственно S или D. Должен выравниваться по адресу, кратному 2.
XXX	Три старших бита смещения.
* Когда не ясно, ссылается ли эта переменная на источник или приёмник, переменная помечается соответственно префиксом S или D.	

Таблица А.2 – Система команд

Обозначение	Операция	Формат команды
ADD (2 operands)	ADD WORDS (Сложение слов). Сложение слов источника и приёмника, и сохранение суммы в приёмнике. (DEST) ← (DEST)+(SRC)	DEST, SRC ADD wreg, waop (011001aa)(waop)(wreg)
ADD (3 operands)	ADD WORDS (Сложение слов). Сложение слов двух источников, и сохранение суммы в приёмнике. (DEST) ← (SRC1)+(SRC2)	DEST, SRC1, SRC2 ADD Dwreg, swreg, waop (010001aa)(waop)(Swreg)(Dwreg)
ADDB (2 operands)	ADD BYTES (Сложение байтов). Сложение байтов источника и приёмника и сохранение суммы в приёмнике (левый операнд). (DEST) ← (DEST)+(SRC)	DEST, SRC ADD breg, baop (011101aa)(baop)(breg)
ADDB (3 operands)	ADD BYTES (Сложение байтов). Сложение байтов двух источников и сохранение суммы в приёмнике. (DEST) ← (SRC1)+(SRC2)	DEST, SRC1, SRC2 ADDB Dbreg, Sbreg, baop (010101aa)(baop)(breg)
ADDC	ADD WORDS WITH CARRY (Сложение слов с переносом). Сложение слов источника и приёмника, сохранение суммы в приемнике и установка флага переноса (0 или 1). (DEST) ← (DEST)+(SRC)+C	DEST, SRC ADDC wreg, waop (101001aa)(waop)(wreg)
ADDCB	ADD BYTES WITH CARRY (Сложение байтов с переносом). Сложение байтов источника и приёмника, сохранение суммы в приемнике и установка флага переноса (0 или 1) (DEST) ← (DEST)+(SRC)+C	DEST,SRC ADDCB breg, baop (101101aa)(baop)(breg)
AND (2 operands)	LOGICAL AND WORDS (Логическое И слов). Операция И над словами источника и приёмника и сохранение результата в приёмнике. Результат - 1 в соответствующем бите, если оба операнда в этом бите имеют 1; 0 – в остальных случаях. (DEST) ← (DEST)AND(SRC)	DEST, SRC AND wreg, waop (011000aa)(waop)(wreg)
AND (3 operands)	LOGICAL AND WORDS (Логическое И слов). Операция И над словами 2-х источников и сохранение результата в приёмнике. Результат: 1 в соответствующем бите, если оба операнда в этом бите имеют 1; 0 - в остальных случаях. (DEST) ← (SRC1)AND(SRC2)	DEST, SRC1, SRC2 AND Dwreg, Swreg, waop (010000aa)(waop)(Swreg)(Dwreg)
ANDB (2 operands)	LOGICAL AND BYTES (Логическое И байтов). Операция И над байтами источника и приёмника и сохранение результата в приёмнике. Результат: 1 в соответствующем бите, если оба операнда в этом бите имеют 1; 0 – в остальных случаях. (DEST) ← (DEST)AND(SRC)	DEST, SRC AND breg, baop (011100aa)(baop)(breg)

Продолжение таблицы А.2

Обозначение	Операция	Формат команды
ANDB (3 operands)	LOGICAL AND BYTES (Логическое И байтов). Операция И над байтами 2-х источников и сохранение результата в приёмнике. Результат: 1 в соответствующем бите, если оба операнда в этом бите имеют 1; 0 – в остальных случаях. (DEST) ← (SRC1)AND(SRC2)	DEST, SRC1, SRC2 AND Dbreg, Sbreg, baop (010100aa)(baop)(Sbreg)(Dbreg)
BMOV	BLOCK MOVE (Перемещение блоков). Перемещение блоков данных типа word из одной зоны памяти в другую. Адреса источника и приёмника вычисляются при помощи режима косвенной адресации с автоинкрементом. Длинный регистр (Ireg) содержит к указатели источника и приёмника, которые хранятся в соседних регистрах слов. Регистр слова Wreg (CNTREG) определяет количество перемещений. Блоки данных могут находиться в любом месте памяти, но не должны перекрываться.	DEST, SRC BMOV Ireg, wreg (11000001)(wreg)(Ireg) Примечание – CNTREG (Wreg) не декрементируется в процессе операции. Легко неумышленно создать длительную непрерываемую операцию с командой BMOV. Для создания прерываемых операций используется команда BMOVI.
BMOVI	INTERRUPTABLE BLOCK MOVE (Прерываемое перемещение блоков). Перемещение блоков данных типа word из одной зоны памяти в другую. Команда идентична BMOV, исключая то, что BMOVI - прерываема. Адреса источника и приёмника вычисляются при помощи косвенного режима адресации с автоинкрементом. Длинный регистр (Ireg) адресуется к указателям источника и приёмника, которые хранятся в соседних регистрах слов. Регистр слов Wreg (CNTREG) определяет количество перемещений. Блоки данных могут находиться в любом месте памяти, но не должны перекрываться. COUNT ← (CNTREG) LOOP: SRCPTR ← (PTRS) DSTPTR ← (PTRS+2) (DSTPTR) ← (SRCPTR) (PTRS) ← SRCPTR+2 (PTSR+2) ← DSTPTR+2 COUNT ← COUNT – 1 if COUNT <> 0 then go to LOOP	DEST,SRC BMOVI Ireg,wreg (11001101)(wreg)(Ireg) Примечание – CNTREG (Wreg) не декрементируется, если выполнение команды не было прервано. Если BMOVI прервана, то в CNTREG сохраняется значение, бывшее в нём во время прерывания. По этой причине необходимо перезагружать CNTREG перед началом выполнения операции BMOVI.
BR	BRANCH INDIRECT (Косвенное разветвление). Продолжает выполнение с адреса, определяемого операндом регистра слова. PC ← (DEST)	DEST BR [wreg] (11100011)(wreg)
CLR	CLEAR WORD (Очистка слова). Обнуляет значение операнда. (DEST) ← 0	DEST CLR wreg (00000001)(wreg)

Продолжение таблицы А.2

Обозначение	Операция	Формат команды
CLRB	CLEAR BYTE (Очистка байта). Обнуляет значение операнда. (DEST) ← 0	DEST CLRB breg (00010001)(breg)
CLRC	CLEAR CARRY FLAG (Очистка флага переноса). Обнуляет флаг переноса. C ← 0	CLRC (11111000)
CLRVT	CLEAR OVERFLOW-TRAP FLAG (Очистка дополнительного флага переполнения). Обнуляет флаг VT VT ← 0	CLRVT (11111100)
CMP	COMPARE WORDS (Сравнение слов). Вычитает слово источника из слова приёмника. Флаги устанавливаются, но операнды остаются прежними. Если имеется заём, флаг переноса равен 0, иначе - 1. (DEST) – (SRC)	DEST, SRC CMP wreg, waop (100010aa)(waop)(wreg)
CMPB	COMPARE BYTES (Сравнение байтов). Вычитает байт источника из байта приёмника. Флаги устанавливаются, но операнды остаются прежними. Если имеется заём, флаг переноса равен 0, иначе - 1. (DEST) – (SRC)	DEST, SRC CMPB breg, baop (100110aa)(baop)(breg)
CMPL	COMPARE LONG (Сравнение чисел типа LONG). Сравнение величин двух операндов типа double-word (long). Операнды определяются с использованием режима прямой адресации. Флаги устанавливаются, но операнды остаются прежними. Если имеется заём, флаг переноса равен 0, иначе 1. (DEST) – (SRC)	DEST, SRC CMPL Ireg, Ireg (11000101)(src Ireg) (destIreg)
DEC	DECREMENT WORD (Декремент слова). Декрементирует величину операнда на 1. (DEST) ← (DEST) – 1	DEST DEC wreg (00000101)(wreg)
DECB	DECREMENT BYTE (Декремент байта). Декрементирует величину операнда на 1. (DEST) ← (DEST) – 1	DEST DECB breg (00010101)(breg)
DI	DISABLE INTERRUPTS (Запрещение прерываний). Запрещает прерывания. Запросы на прерывания не удовлетворяются после этой команды. Interrupt Enable (PSW.1) ← 0	DI (11111010)
CLRB	CLEAR BYTE (Очистка байта). Обнуляет значение операнда. (DEST) ← 0	DEST CLRB breg (00010001)(breg)
CLRC	CLEAR CARRY FLAG (Очистка флага переноса). Обнуляет флаг переноса. C ← 0	CLRC (11111000)
CLRVT	CLEAR OVERFLOW-TRAP FLAG (Очистка дополнительного флага переполнения). Обнуляет флаг VT VT ← 0	CLRVT (11111100)

Продолжение таблицы А.2

Обозначение	Операция	Формат команды
CMP	COMPARE WORDS (Сравнение слов). Вычитает слово источника из слова приёмника. Флаги устанавливаются, но операнды остаются прежними. Если имеется заём, флаг переноса равен 0, иначе 1. (DEST) – (SRC)	DEST, SRC CMP wreg, waop (100010aa)(waop)(wreg)
CMPB	COMPARE BYTES (Сравнение байтов). Вычитает байт источника из байта приёмника. Флаги устанавливаются, но операнды остаются прежними. Если имеется заём, флаг переноса равен 0, иначе - 1. (DEST) – (SRC)	DEST, SRC CMPB breg, baop (100110aa)(baop)(breg)
CMPL	COMPARE LONG (Сравнение чисел типа LONG). Сравнение величин двух операндов типа double-word (long). Операнды определяются с использованием режима прямой адресации. Флаги устанавливаются, но операнды остаются прежними. Если имеется заём, флаг переноса равен 0, иначе - 1. (DEST) – (SRC)	DEST, SRC CMPL Ireg, Ireg (11000101)(src Ireg) (destIreg)
DEC	DECREMENT WORD (Декремент слова). Декрементирует величину операнда на 1. (DEST) ← (DEST) – 1	DEST DEC wreg (00000101)(wreg)
DECB	DECREMENT BYTE (Декремент байта). Декрементирует величину операнда на 1. (DEST) ← (DEST) – 1	DEST DECB breg (00010101)(breg)
DI	DISABLE INTERRUPTS (Запрещение прерываний). Запрещает прерывания. Запросы на прерывания не удовлетворяются после этой команды. Interrupt Enable (PSW.1) ← 0	DI (11111010)
DIV	DIVIDE INTEGER (Деление чисел типа INTEGER). Делит содержимое приёмника - операнд типа LONG-INTEGER на содержимое источника - операнд типа INTEGER, используя знаковую арифметику. Частное сохраняет в младшем слове (т. е. в слове с меньшим адресом) приёмника, а остаток в старшем слове. (low word DEST) ← (DEST)/(SRC) (high word DEST)← (DEST) MOD (SRC)	DEST, SRC DIV Ireg, waop (11111110)(100011aa) (waop)(Ireg)
DIVB	DIVIDE SHORT-INTEGER (Деление чисел типа SHORT INTEGER). Делит содержимое приёмника - операнд типа INTEGER на содержимое источника - операнд типа SHORT-INTEGER, используя знаковую арифметику. Частное сохраняет в младшем байте (т. е. в байте с меньшим адресом) приёмника, а остаток в старшем байте. (low word DEST) ← (DEST)/(SRC) (high word DEST) ← (DEST)MOD(SRC)	DEST, SRC DIVB wreg, baop (11111110)(100111aa) (baop)(wreg)

Продолжение таблицы А.2

Обозначение	Операция	Формат команды
DIVU	DIVIDE WORDS, UNSIGNED (Деление чисел типа WORD, незнаковых). Делит содержимое приёмника - операнд типа DOUBLE-WORD на содержимое источника -операнд слова типа WORD, используя буззнаковую арифметику. Частное сохраняет в младшем слове (т. е. в слове с меньшим адресом) приёмника, а остаток - в старшем слове. Следующие две операции проводятся одновременно. (low word DEST) ← (DEST)/(SRC) (high word DEST)← (DEST)MOD(SRC)	DEST, SRC DIVU Ireg, waop (100011aa)(waop)(Ireg)
DIVUB	DIVIDE BYTES, UNSIGNED (Деление чисел типа WORD, незнаковых). Делит содержимое приёмника - операнд типа WORD на содержимое источника - операнд типа BYTE, используя беззнаковую арифметику. Частное сохраняет в младшем байте (т. е. в байте с меньшим адресом) приёмника, а остаток - в старшем байте. Следующие две операции проводятся одновременно. (low byte DEST) ← (DEST)/(SRC) (high byte DEST)← (DEST)MOD(SRC)	DEST, SRC DIVUB wreg, baop (100111aa)(baop)(wreg)

DJNZ	DECREMENT AND JUMP IF NOT ZERO
	(Декремент и переход при отсутствии нуля). Декрементирует величину операнда типа BYTE на 1. Если результат равен 0, управление передаётся следующей по порядку команде. Если результат не равен 0, команда добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Смещение может быть в диапазоне от минус 128 до плюс 127.
	(COUNT) ← (COUNT) - 1
	if (COUNT)<>0 then
	PC ← PC + disp (Примечание 1)
	endif
		COUNT,ADDR
		DJNZ breg, cadd
		(11100000)(breg)(disp)

DJNZW	DECREMENT AND JUMP IF NOT ZERO WORD
	(Декремент и переход при отсутствии нуля). Декрементирует величину операнда типа WORD на 1. Если результат равен 0, управление передаётся следующей по порядку команде. Если результат не равен 0, команда добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Смещение может быть в диапозоне от минус 128 до плюс 127.
	(COUNT) ← (COUNT) - 1
	if (COUNT)<>0 then
	PC ← PC + disp (Примечание 1)
	endif
		COUNT, ADDR
		DJNZW wreg, cadd
		(11100001)(wreg)(disp)

DPTS	DISABLE PERIPHERAL TRANSACTION SERVER (PTS) (Блокирование Периферийного Сервера Обмена - PTS). Блокирует PTS. PTS Disable (PSW.2) ← 0	DPTS (11101100)

Продолжение таблицы А.2

Обозначение	Операция	Формат команды
EI	ENABLE INTERRUPTS (Разрешение прерываний). Разрешает прерывания, запрашиваемые после выполнения следующего состояния. Запросы на прерывания не могут удовлетворяться немедленно после выполнения этой команды. Interrupt Enable (PSW.1) ← 1	EI (11111011)
EPTS	ENABLE PERIPHERAL TRANSACTION SERVER (PTS) (Разблокировка Периферийного Сервера Обмена - PTS). Разблокирует PTS. PTS Enable (PSW.2) ← 1	EPTS (11101101)

EXT	SIGN-EXTEND INTEGER INTO LONG-INTEGER
	(Знаковое расширение INTEGER в LONG-INTEGER).
	Расширяет со знаком младшее слово операнда до двойного слова,
	if (low word DEST) < 8000h then
	(high word DEST) ← 0
	else
	(high word DEST) ← 0FFFFh
	endif
		EXT Ireg
		(00000110)(Ireg)

EXTB	SIGN-EXTEND SHORT-INSGER INTO INTEGER
	(Знаковое расширение SHORT-INTEGER в INTEGER).
	Расширяет со знаком младший байт операнда до слова.
	if (low byte DEST) < 80h then
	(high byte DEST) ← 0
	else
	(high byte DEST) ←0FFh
	endif
		EXTB wreg
		(00010110)(wreg)

IDLPD	IDLE/POWERDOWN
	В зависимости от 8-битной величины операнда KEY эта команда выбирает:
	- вход в режим IDLE (KEY=1);
	- вход в режим POWERDOWN (KEY=2);
	- выполнить последовательность сброса
	(любое другое значение KEY, не равное 1 или 2).
	Контроллер шины завершает цикл упреждающей выборки перед остановкой ЦПУ или сбросом.
	if KEY=1 then
	enter Idle
	else
	if KEY=2 then
	enter POWERDOWN
	else
	execute reset
		IDLPD #key
		(11110110)(key)

INC	INCREMENT WORD (Инкремент слова) Увеличение значение слова операнда на 1. (DEST) ← (DEST)+1	INC wreg (00000111)(wreg)
INCB	INCREMENT BYTE (Инкремент байта) Инкрементирует байт операнда на 1. (DEST) ← (DEST)+1	INCB breg (00010111)(breg)

Продолжение таблицы А.2

Обозначение	Операция	Формат команды
JBC	JUMP IF BIT IS CLEAR (Переход при обнуленном бите). Тестирует определённый бит. Если бит установлен, управление передаётся следующей по порядку команде. Если бит обнулен, то команда добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Смещение может быть в диапазоне от минус 128 до плюс 127. if (specified bit)=0 then PC ← PC + disp (Примечание 1)	JBC breg, bitno, cadd (00110bbb)(breg)(disp)
JBS	JUMP IF BIT IS SET (Переход при установленном бите). Тестирует определённый бит. Если бит обнулен, управление передаётся следующей по порядку команде. Если бит установлен, то команда добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Смещение может быть в диапазоне от минус 128 до плюс 127. if (specified bit)=1 then PC ← PC + disp (Примечание 1)	JBS breg, bitno, cadd (00111bbb)(breg)(disp)
JC	JUMP IF CARRY FLAG IS SET (Переход при установленном флаге переноса). Тестирует C - флаг переноса. Если флаг переноса обнулен, управление передаётся следующей по порядку команде.Если флаг переноса установлен, то команда добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Смещение может быть в диапазоне от минус 128 до плюс 127. if C=1 then PC ← PC + disp (Примечание 1)	JC cadd (11011011)(disp)
JE	JUMP IF EQUAL (Переход при равенстве). Тестирует Z - флаг нуля. Если флаг обнулен, управление передаётся следующей по порядку команде. Если флаг нуля установлен, то команда добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Смещение может быть в диапазоне от минус 128 до плюс 127. if Z=1 then PC ← PC + disp (Примечание 1)	JE cadd (11011111)(disp)
JGE	JUMP IF SIGNED GREATER THAN OR EQUAL (Переход, если знаковое больше или равно). Тестирует N-флаг отрицательного результата. Если флаг установлен, управление передаётся следующей по порядку команде. Если флаг N обнулен, то команда добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Смещение может быть в диапазоне от минус 128 до плюс 127. if N=0 then PC ← PC + disp (Примечание 1)	JGE cadd (11010110)(disp)

Продолжение таблицы А.2

Обозначение	Операция	Формат команды
JGT	JUMP IF SIGNED GREATER THAN (Переход, если знаковое больше). Тестирует Z-флаг нуля и N-флаг отрицательного результата. Если один из флагов установлен, управление передаётся следующей по порядку команде. Если оба флага очищены, то команда добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Смещение может быть в диапазоне от минус 128 до плюс 127. if N=0 AND Z=0 then PC ← PC + disp (Примечание 1)	JGT cadd (11010010)(disp)
JH	JUMP IF HIGHER (UNSIGNED) (Переход, если больше (беззнаковое).) Тестируются флаги нуля и переноса. Если флаг переноса обнулен или флаг нуля установлен, управление передаётся следующей по порядку команде. Если флаг переноса установлен и флаг нуля очищен, то команда добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Смещение может быть в диапазоне от минус 128 до плюс 127. if C=1 AND Z=0 then PC ← PC + disp (Примечание 1)	JH cadd (11011001)(disp)
JLE	JUMP IF SIGNED LESS THAN OR EQUAL (Переход, если знаковое меньше или равно). Тестирует negative flag- N и carry flag-C. Если оба флага очищены, управление передаётся следующей по порядку команде.Если один флаг установлен, то команда добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Смещение может быть в диапазоне от минус 128 до плюс 127. if N=1 OR Z=1 then PC ← PC + disp (Примечание 1)	JLE cadd (11011010)(disp)
JLT	JUMP IF SIGNED LESS THAN (Переход, если знаковое меньше). Тестирует negative flag - N. Если флаг очищен, управление передаётся следующей по порядку команде. Если флаг установлен, то команда добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Смещение может быть в диапазоне от минус 128 до плюс 127. if N=1 then PC ← PC + disp (Примечание 1)	JLT cadd (11011110)(disp)

Продолжение таблицы А.2

Обозначение	Операция	Формат команды
JNC	JUMP IF CARRY FLAG IS CLEAR (Переход, если флаг переноса чист). Тестирует carry flag - C. Если флаг установлен, управление передаётся следующей по порядку команде. Если флаг очищен, то команда добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Смещение может быть в диапазоне от минус 128 до плюс 127. if C=0 then PC ← PC + disp (Примечание 1)	JNC cadd (11010011)(disp)
JNE	JUMP IF NOT EQUAL (Переход при неравенстве). Тестирует флаг нуля. Если флаг установлен, управление передаётся следующей по порядку команде. Если флаг очищен, то команда добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Смещение может быть в диапазоне от минус 128 до плюс 127. if Z=0 then PC ← PC + disp (Примечание 1)	JNE cadd (11010111)(disp)
JNH	JUMP IF NOT HIGHER (UNSIGNED) (Переход если не больше (беззнаковое)). Тестирует флаг нуля и флаг переноса. Если флаг переноса установлен и флаг нуля очищен, управление передаётся следующей по порядку команде. Если флаг переноса установлен или флаг нуля установлен, то команда добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Смещение может быть в диапазоне от минус 128 до плюс 127. if C=0 OR Z=1 then PC ← PC + disp (Примечание 1)	JNE cadd (11010001)(disp)
JNST	JUMP IF STICKY BIT FLAG IS CLEAR (Переход, если флаг ST чист). Тестирует флаг ST (sticky bit). Если флаг установлен, управление передаётся следующей по порядку команде. Если флаг очищен, то команда добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Смещение может быть в диапазоне от минус 128 до плюс 127. if ST=0 then PC ← PC + disp (Примечание 1)	JNST cadd (11010000)(disp)

Продолжение таблицы А.2

Обозначение	Операция	Формат команды
JNV	JUMP IF OVERFLOW FLAG IS CLEAR (Переход, если флаг переполнения чист). Тестирует overflow flag - V. Если флаг установлен, управление передаётся следующей по порядку команде. Если флаг очищен, то команда добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Смещение может быть в диапазоне от минус 128 до плюс 127. if V=0 then PC ← PC + disp (Примечание 1)	JNV cadd (11010101)(disp)
JNVT	JUMP IF OVERFLOW-TRAP FLAG IS CLEAR (Переход, если дополнительный флаг переполнения очищен). Тестирует overflow-trap flag- VT. Если флаг установлен, эта команда очищает флаг, и управление передаётся следующей по порядку команде. Если флаг очищен, то команда добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Смещение может быть в диапазоне от минус 128 до плюс 127. if VT=0 then PC ← PC + disp (Примечание 1)	JNVT cadd (11010100)(disp)
JST	JUMP IF STICKY BIT FLAG IS SET (Переход, если флаг ST установлен). Тестируется флаг ST. Если флаг очищен, управление передаётся следующей по порядку команде. Если флаг ST установлен, то команда добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Смещение может быть в диапазоне от минус 128 до плюс 127. if ST=1 then PC ← PC + disp (Примечание 1)	JST cadd (11011000)(disp)
JV	JUMP IF OVERFLOW FLAG IS SET (Переход, если установлен флаг переполнения). Тестируется overflow flag (V). Если флаг переполнения очищен, управление передаётся следующей по порядку команде. Если флаг переполнения установлен, то команда добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Смещение может быть в диапазоне от минус 128 до плюс 127. if V=1 then PC ← PC + disp (Примечание 1)	JV cadd (11011101)(disp)

Продолжение таблицы А.2

Обозначение	Операция	Формат команды
JVT	JUMP IF OVERFLOW-TRAP FLAG IS SET (Переход, если дополнительный флаг переполнения установлен). Тестируется флаг VT. Если флаг очищен, управление передаётся следующей по порядку команде. Если флаг ловушки переполнения установлен, то команда добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Смещение может быть в диапазоне от минус 128 до плюс 127. if VT=1 then PC ← PC + disp (Примечание 1)	JST cadd (11011100)(disp)
LCALL	LONG CALL (Длинный вызов). Загружает содержимое программного счётчика (адрес возврата) в стек, затем добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Операнд может иметь любой адрес во всём адресном пространстве. SP ← SP - 2 (SP) ← PC PC ← PC + disp	LCALL cadd (11101111)(disp-low) (disp-high)
LD	LOAD WORD (Загрузка слова). Загружает значение слова источника в приёмник. (DEST) ← (SRC)	DEST, SRC LD wreg, waop (101000aa)(waop) (wreg)
LDB	LOAD BYTE (Загрузка байта). Загружает значение байта источника в приёмник. (DEST) ← (SRC)	DEST, SRC LDB breg, baop (101100aa)(baop)(breg)

LDBSE	LOAD BYTE SIGN-EXTENDED
	(Загрузка байта со знаковым расширением).
	Расширяет знаковое значение операнда источника типа SHORT- INTEGER и загружает его в приёмник типа INTEGER.
	(low byte DEST) ← (SRC)
	if (SRC) < 80h then
	(high byte DEST) ← 0
	else
	(high byte DEST) ← 0FFh
		DEST, SRC
		LDBSE wreg, baop
		(101111aa)(baop)
		(wreg)

LDBZE	LOAD BYTE ZERO-EXTENDED (Загрузка байта, дополненного нулём). Значение операнда источника типа BYTE дополняется нулями и загружается в приёмник типа WORD, (low byte DEST) ← (SRC) (high byte DEST) ← 0	DEST, SRC LDBZE wreg, baop (101011aa)(baop) (wreg)

Продолжение таблицы А.2

Обозначение	Операция	Формат команды
LJMP	LONG JUMP (Длинный переход). Добавляет к программному счётчику смещение между концом этой команды и меткой перехода. Операнд может иметь любой адрес во всём адресном пространстве. PC ← PC + disp	LJMP cadd (11100111) (disp-low)(disp-high)
MUL (2 операнда)	MULTIPLY INTEGERS (Умножение чисел типа INTEGER). Перемножает операнды источника и приёмника типа INTEGER, используя знаковую арифметику, и сохраняет 32-битный результат в приёмнике типа LONG- INTEGER. После выполнения этой команды флаг ST не определен. (DEST) ← (DEST) * (SRC)	DEST, SRC MUL Ireg, waop (11111110)(010111aa) (waop)(Ireg)
MUL 3 операнда	MULTIPLY INTEGERS (Умножение чисел типа INTEGER). Перемножает операнды двух источников типа INTEGER, используя знаковую арифметику, и сохраняет 32-битный результат в приёмнике типа LONG-INTEGER. После выполнения этой команды флаг ST не определен. (DEST) ← (SRC1) * (SRC2)	DEST, SRC1, SRC2 MUL Ireg, wreg, waop (11111110)(010011aa) (waop)(wreg) (Ireg)
MULB 2 операнда	MULTIPLY SHORT-INTEGERS (Умножение чисел типа SHORT-INTEGER). Перемножает операнды источника и приёмника типа SHORT- INTEGER, используя знаковую арифметику, и сохраняет 16-битный результат в приёмнике типа INTEGER. После выполнения этой команды флаг ST не определен. (DEST) ← (DEST) * (SRC)	DEST, SRC MULB wreg, baop (11111110)(011111aa) (baop)(wreg)
MULB 3 операнда	MULTIPLY SHORT-INTEGERS (Умножение чисел типа SHORT-INTEGER). Перемножает операнды двух источников типа SHORT-INTEGER, используя знаковую арифметику и сохраняет 16-битный результат в приёмнике типа INTEGER. После выполнения этой команды флаг ST не определен. (DEST) ← (SRC1) * (SRC2)	DEST, SRC1, SRC2 MULB wreg, breg, baop (11111110)(010111aa) (baop)(breg) (wreg)
MULU 2 операнда	MULTIPLY WORDS, UNSIGNED (Умножение чисел типа WORD, незнаковое). Перемножает операнды источника и приёмника типа WORD, используя безнаковую арифметику, и сохраняет 32-битный результат в приёмнике типа DOUBLE-WORD. После выполнения этой команды флаг ST не определен. (DEST) ← (DEST) * (SRC)	DEST, SRC MULU Ireg, waop (011011aa)(waop)(Ireg)

Продолжение таблицы А.2

Обозначение	Операция	Формат команды
MULU 3 операнда	MULTIPLY WORDS, UNSIGNED (Умножение чисел типа WORD, незнаковое). Перемножает операнды двух источников типа WORD, используя безнаковую арифметику, и сохраняет32-битный результат в приёмнике типа DOUBLE- WORD. После выполнения этой команды флаг ST не определен. (DEST) ← (SRC1) * (SRC2)	DEST, SRC1, SRC2 MULU Ireg, wreg, waop (010011aa)(waop)(wreg) (Ireg)
MULUB 2 операнда	MULTIPLY BYTES, UNSIGNED (Умножение чисел типа BYTE, безнаковое). Перемножает операнды источника и приёмника типа BYTE, используя беззнаковую арифметику, и сохраняет результат типа WORD в приёмнике. После выполнения этой команды флаг ST не определен. (DEST) ← (DEST) * (SRC)	DEST, SRC MULUB wreg, baop (011111aa)(baop)(wreg)
MULUB 3 операнда	MULTIPLY BYTES, UNSIGNED (Умножение чисел типа BYTE, незнаковое). Перемножает операнды двух источников типа BYTE, используя незнаковую арифметику, и сохраняет результат типа WORD в приёмнике. После выполнения этой команды флаг ST не определен. (DEST) ← (SRC1) * (SRC2)	DEST, SRC1, SRC2 MULUB wreg, breg, baop (010111aa)(baop)(breg) (wreg)
NEG	NEGATE INTEGER (Изменение знака числа типа INTEGER). Изменяет знак операнда типа INTEGER. (DEST) ← – (DEST)	NEG wreg (00000011)(wreg)
NEGB	NEGATE SHORT-INTEGER (Изменение знака числа типа SHORT-INTEGER). Изменяет знак операнда типа SHORT-INTEGER. (DEST) ← – (DEST)	NEGB breg (00010011)(breg)
NOP	NO OPERATION (Нет операции). Ничего не делает. Управление переходит к следующей по порядку команде.	NOP (11111101)
NORML	NORMALIZE LONG-INTEGER (Нормализация числа типа LONG-INTEGER). Нормализует операнд источника типа LONG-INTEGER (левый операнд). Эта команда сдвигает операнд влево до тех пор, пока его старший значащий бит<>"1" или пока не будет совершен 31 сдвиг. Если в старшем значащем бите остался "0" после 31 сдвига, команда останавливает процесс и устанавливает флаг нуля. Команда сохраняет число совершённых сдвигов в приёмнике (правый операнд) (COUNT) ← 0 do while(MSB(SRC) ← 0)AND (COUNT)<31) (SRC) ← (SRC) * 2 (COUNT) ← (COUNT) + 1	SRC, DEST NORML Ireg, breg (00001111)(breg)(Ireg)

Продолжение таблицы А.2

Обозначение	Операция	Формат команды
NOT	COMPLEMENT WORD (Инверсия числа типа WORD). Инвертирует значение операнда типа word (заменяет каждую "1" на"0" и каждый "0" на "1") (DEST) ← NOT(DEST)	NOT wreg (00000010)(wreg)
NOTB	COMPLEMENT BYTE (Инверсия числа типа BYTE). Инвертирует значение операнда типа byte (заменяет каждую "1" на"0" и каждый "0" на "1") (DEST) ← NOT(DEST)	NOTB breg (00010010)(breg)
OR	LOGICAL OR WORDS (Логическое ИЛИ слов). Проводит операцию ИЛИ над операндом источника типа word и операндом приёмника типа word и заменяет первоначальное значение приёмника на результат. Результат = 1 в каждом разряде, когда хотя бы один из операндов имеет 1. (DEST) ← (DEST)OR(SRC)	DEST, SRC OR breg, waop (100000aa)(waop)(breg)
ORB	LOGICAL OR BYTES (Логическое ИЛИ байтов). Проводит операцию ИЛИ над операндом источника типа byte и операндом приёмника типа byte и заменяет первоначальное значение приёмника на результат. Результат = 1 в каждом разряде, когда хотя бы один из операндов имеет 1. (DEST) ← (DEST)OR(SRC)	DEST, SRC ORB breg, baop (100100aa)(baop)(breg)
POP	POP WORD (Чтение слова из стека). Читает слово из вершины стека и помещает его в приёмник. (DEST) ← (SP) SP ← SP + 2	POP waop (110011aa)(waop)
POPA	POP ALL (Читать всё из стека). Эта команда используется взамен POPF, для поддержки восьми добавочных прерываний. Она читает два слова из стека и помещает первое слово в регистр INT_MASK1/WSR, а второе слово в PSW/INT_MASK сдвоенный регистр. Эта команда инкрементирует указатель стека SP на 4. Запросы на прерывания не могут идти сразу после этой команды. INT_MASK1/WSR ← (SP) SP ← SP + 2 PSW/INT_MASK ← (SP) SP ← SP + 2	POPA (11110101)
POPF	POP FLAGS (Чтение флагов из стека). Читает слово из вершины стека и помещает его в PSW. Запросы на прерывания не могут идти сразу после этой команды. (PSW) ← (SP) SP ← SP + 2	POPF (11110011)

Продолжение таблицы А.2

Обозначение	Операция	Формат команды
PUSH	PUSH WORD (Загрузка слова в стек). Загружает операнд типа word в стек. SP ← SP - 2 (SP) ← (DEST)	PUSH waop (110010aa)(waop)
PUSHA	PUSH ALL (Загрузить всё в стек). Эта команда используется взамен PUSHF, для поддержки восьми добавочных прерываний. Она загружает два слова в стек – PSW/INT_MASK и слово, формируемое сдвоенным регистром INT_MASK1/WSR Эта команда очищает регистры PSW, INT_MASK и INT_MASK1 и декрементирует SP (указатель стека) на 4. Запросы на прерывания не могут идти сразу после этой команды. SP ← SP 2 (SP) ← PSW/INT_MASK PSW/INT_MASK ← 0 SP ← SP 2 (SP) ← INT_MASK1/WSR INT_MASK1 ← 0	PUSHA (11110100)
PUSHF	PUSH FLAGS (Загрузка флагов в стек). Загружает PSW в вершину стека, затем устанавливает его в нулевое состояние. Это подразумевает то, что все прерывания запрещены. Запросы на прерывания не могут идти сразу после этой команды. SP ← SP – 2 (SP) ← PSW/INT_MASK PSW/INT_MASK ← 0	PUSHF (11110010)
RET	RETURN FROM SUBROUTINE (Возврат из подпрограммы). Считывает значение программного счётчика PC из вершины стека. PC ← (SP) SP ← SP + 2	RET (11110000)
RST	RESET SYSTEM (Сброс системы). Инициализирует PSW в 0, PC в 2080h, а SFR в их начальные значения. Выполнение этой команды ведёт к тому, что контакт RESET# находится в низком состоянии 16 машинных тактов. SFR Reset Status Pin Reset Status PSW ← 0 PC ← 2080h	RST (11111111)
SCALL	SHORT CALL (Короткий вызов). Загружает содержимое PC (адрес возврата) в стек, затем добавляет к PC смещение между концом этой команды и меткой. Смещение должно быть в пределах от минус 1024 до плюс 1023 включительно. SP ← SP – 2 (SP) ← PC PC ← PC + disp (Примечание 1)	SCALL cadd (00101xxx)(disp-low)

Продолжение таблицы А.2

Обозначение	Операция	Формат команды
SETC	SET CARRY FLAG (Установка флага переноса). Устанавливает флаг переноса. C ← 1	SETC (11111001)
SHL	SHIFT WORD LEFT (Сдвиг слова влево). Сдвигает операнд типа word приёмника влево столько раз, сколько установлено специальным операндом - счётчиком. Значение счётчика может быть задано непосредственно, в пределах от 0 до 15 (0Fh) включительно, или как содержимое какого-либо регистра со значением в пределах от 0 до 31(1Fh) включительно. Правые биты результата заполняются нулями. Смещение производится через флаг переноса Temp ← (COUNT) do while Temp <> 0 C ← High order bit of (DEST) (DEST) ← (DEST) * 2 Temp ← Temp - 1	SHL wreg, #count (00001001)(count) (wreg) или SHL wreg, breg (00001001)(breg)(wreg)
SHLB	SHIFT BYTE LEFT (Сдвиг байта влево). Сдвигает операнд типа byte приёмника влево столько раз, сколько установлено специальным операндом - счётчиком. Значение счётчика может быть задано непосредственно, в пределах от 0 до 15 (0Fh) включительно, или как содержимое какого-либо регистра со значением в пределах от 0 до 31 (1Fh) включительно. Правые биты результата заполняются нулями. Смещение производится через флаг переноса Temp ← (COUNT) do while Temp <> 0 C ← High order bit of (DEST) (DEST) ← (DEST) * 2 Temp ← Temp - 1	SHLB breg, #count (00011001)(count) (breg) или SHLB breg, breg (00011001)(breg)(breg)
SHLL	SHIFT DOUBLE-WORD LEFT (Сдвиг числа типа DOUBLE- WORD влево). Сдвигает операнд типа double-word приёмника влево столько раз, сколько установлено специальным операндом-счётчиком. Значение счётчика может быть задано непосредственно, в пределах от 0 до 15 (0Fh) включительно, или как содержимое какого-либо регистра со значением в пределах от 0 до 31 (1Fh) включительно. Правые биты результата заполняются нулями. Смещение производится через флаг переноса Temp ← (COUNT) do while Temp <> 0 C ← High order bit of (DEST) (DEST) ← (DEST) * 2 Temp ← Temp - 1	SHLL Ireg, #count (00001101)(count) (breg) или SHLL Ireg, breg (00001101)(breg)(Ireg)

Продолжение таблицы А.2

Обозначение	Операция	Формат команды
SHR	LOGICAL RIGHT SHIFT WORD (Логический сдвиг слова вправо). Сдвигает операнд типа word приёмника вправо столько раз, сколько установлено специальным операндом - счётчиком. Значение счётчика может быть задано непосредственно, в пределах от 0 до 15 (0Fh) включительно, или как содержимое какого-либо регистра со значением в пределах от 0 до 31 (1Fh) включительно. Левые биты результата заполняются нулями. Смещение производится через флаг переноса. Эта команда очищает флаг ST в начале команды. Если в какой-то момент времени в течение сдвига "1" сдвигается во флаг переноса и осуществляется другой цикл сдвига, команда устанавливает флаг ST. Temp ← (COUNT) do while Temp <> 0 C ← Low order bit of (DEST) (DEST) ← (DEST)/ 2 (Примечание 2) Temp ← Temp - 1	SHR wreg, #count (00001000)(count) (wreg) или SHR wreg, breg (00001000)(breg)(wreg)
SHRA	ARITHMETIC RIGHT SHIFT WORD (Арифметический сдвиг слова вправо). Сдвигает операнд типа word приёмника вправо столько раз, сколько установлено специальным операндом - счётчиком. Значение счётчика может быть задано непосредственно, в пределах от 0 до 15 (0Fh) включительно, или как содержимое какого-либо регистра со значением в пределах от 0 до 31 (1Fh), включительно. Если в оригинале величина старшего бита была "0", сдвигаются нули. Если величина была "1", сдвигаются единицы. Смещение производится через флаг переноса. Эта команда очищает флаг ST в начале команды. Если в какой-то момент времени в течение сдвига "1" сдвигается во флаг переноса и осуществляется другой цикл сдвига, команда устанавливает флаг ST. Temp ← (COUNT) do while Temp <> 0 C ← Low order bit of (DEST) (DEST) ← (DEST)/ 2 (Примечание 3) Temp ← Temp – 1	SHRA wreg, #count (00001010)(count) (wreg) или SHRA wreg, breg (00001010)(breg)(wreg)

Продолжение таблицы А.2

Обозначение	Операция	Формат команды
SHR	LOGICAL RIGHT SHIFT WORD (Логический сдвиг слова вправо). Сдвигает операнд типа word приёмника вправо столько раз, сколько установлено специальным операндом - счётчиком. Значение счётчика может быть задано непосредственно, в пределах от 0 до 15 (0Fh) включительно, или как содержимое какого-либо регистра со значением в пределах от 0 до 31 (1Fh) включительно. Левые биты результата заполняются нулями. Смещение производится через флаг переноса. Эта команда очищает флаг ST в начале команды. Если в какой-то момент времени в течение сдвига "1" сдвигается во флаг переноса и осуществляется другой цикл сдвига, команда устанавливает флаг ST. Temp ← (COUNT) do while Temp <> 0 C ← Low order bit of (DEST) (DEST) ← (DEST)/ 2 (Примечание 2) Temp ← Temp – 1	SHR wreg, #count (00001000)(count)(wreg) или SHR wreg, breg (00001000)(breg)(wreg)
SHRA	ARITHMETIC RIGHT SHIFT WORD (Арифметический сдвиг слова вправо). Сдвигает операнд типа word приёмника вправо столько раз, сколько установлено специальным операндом - счётчиком. Значение счётчика может быть задано непосредственно, в пределах от 0 до 15 (0Fh) включительно, или как содержимое какого-либо регистра со значением в пределах от 0 до 31 (1Fh), включительно. Если в оригинале величина старшего бита была "0", сдвигаются нули. Если величина была "1", сдвигаются единицы.Смещение производится через флаг переноса. Эта команда очищает флаг ST в начале команды. Если в какой-то момент времени в течение сдвига "1" сдвигается во флаг переноса и осуществляется другой цикл сдвига, команда устанавливает флаг ST. Temp ← (COUNT) do while Temp <> 0 C ← Low order bit of (DEST) (DEST) ← (DEST)/ 2 (Примечание 3) Temp ← Temp - 1	SHRA wreg, #count (00001010)(count)(wreg) или SHRA wreg, breg (00001010)(breg)(wreg)

Продолжение таблицы А.2

Обозначение	Операция	Формат команды
SHRB	LOGICAL RIGHT SHIFT BYTE (Логический сдвиг байта вправо). Сдвигает операнд типа byte приёмника вправо столько раз, сколько установлено специальным операндом - счётчиком. Значение счётчика может быть задано непосредственно, в пределах от 0 до 15 (0Fh) включительно, или как содержимое какого-либо регистра со значением в пределах от 0 до 31 (1Fh) включительно. Левые биты результата заполняются нулями. Смещение производится через флаг переноса. Эта команда очищает флаг ST в начале команды. Если в какой-то момент времени в течение сдвига "1" сдвигается во флаг переноса и осуществляется другой цикл сдвига, команда устанавливает флаг ST. Temp ← (COUNT) do while Temp <> 0 C ← Low order bit of (DEST) (DEST) ← (DEST)/ 2 (Примечание 2) Temp ← Temp - 1	SHRB breg, #count (00011000)(count)(breg) или SHRB breg, breg (00011000)(breg)(breg)
SHRL	LOGICAL RIGHT SHIFT DOUBLE-WORD (Логический сдвиг вправо числа типа DOUBLOE-WORD). Сдвигает операнд типа double-word приёмника вправо столько раз, сколько установлено специальным операндом-счётчиком. Значение счётчика может быть задано непосредственно, в пределах от 0 до 15 (0Fh) включительно, или как содержимое какого-либо регистра со значением в пределах от 0 до 31 (1Fh) включительно. Левые биты результата заполняются нулями. Смещение производится через флаг переноса. Эта команда очищает флаг ST в начале команды. Если в какой-то момент времени в течение сдвига "1" сдвигается во флаг переноса и осуществляется другой цикл сдвига, команда устанавливает флаг ST. Temp ← (COUNT) do while Temp <> 0 C ← Low order bit of (DEST) (DEST) ← (DEST)/ 2 (Примечание 2) Temp ← Temp - 1	SHRL Ireg, #count (00001100)(count)(Ireg) или SHRL Ireg, breg (00001100)(breg)(Ireg)
SJMP	SHORT JUMP (Короткий переход). Добавляет к PC смещение между концом этой команды и меткой перехода. Смещение может быть в пределах от минус 1024 до плюс 1023 включительно. PC ← PC + disp (Примечание 1)	SJMP cadd (00100xxx)(disp-low)

Продолжение таблицы А.2

Обозначение	Операция	Формат команды
SKIP	TWO BYTE NO-OPERATION (Двухбайтная пустая команда). Ничего не делает. Управление передаётся следующей по порядку команде. Это двухбайтная NOP, где второй байт - любая величина, которая просто игнорируется.	SKIP breg (00000000)(breg)
ST	STORE WORD (Сохранение слова). Сохраняет величину операнда источника типа word (левый операнд) в приёмнике (правый операнд). (DEST) ← (SRC)	SRC, DEST ST wreg,waop (110000aa)(waop)(wreg)
STB	STORE BYTE (Сохранение байта). Сохраняет величину операнда источника типа byte (левый операнд) в приёмнике (правый операнд). (DEST) ← (SRC)	SRC, DEST STB breg, baop (110001aa)(baop)(breg)
SUB (2 операнда)	SUBTRACT WORDS (Вычитание слов). Вычитает операнд источника типа word из операнда приёмника типа word, сохраняет результат в приёмнике и устанавливает флаг переноса при заёме. (DEST) ← (DEST) – (SRC)	DEST, SRC SUB wreg, waop (011010aa)(waop)(wreg)
SUB (3 операнда)	SUBTRACT WORDS (Вычитание слов). Вычитает операнд первого источника типа word из операнда второго источника, сохраняет результат в приёмнике и устанавливает флаг переноса при заёме. (DEST) ← (SRC1) – (SRC2)	DEST, SRC1, SRC2 SUB Dwreg, Swreg, waop (010010aa)(waop)(Swreg)(Dwreg)
SUBB (2 операнда)	SUBTRACT BYTES (Вычитание байтов). Вычитает операнд источника типа byte из операнда приёмника типа byte, сохраняет результат в приёмнике и устанавливает флаг переноса при заёме. (DEST) ← (DEST) – (SRC)	DEST, SRC SUBB breg, baop (010110aa)(baop)(breg)
SUBB (3 операнда)	SUBTRACT BYTES (Вычитание байтов). Вычитает операнд первого источника типа byte из операнда второго источника, сохраняет результат в приёмнике и устанавливает флаг переноса при заёме. (DEST) ← (SRC1) – (SRC2)	DEST, SRC1, SRC2 SUBB Dbreg, Sbreg, baop (010110aa)(baop)(Sbreg)(Dbreg)
SUBC	SUBTRACT WORDS WITH BORROW (Вычитание слов с заёмом). Вычитает операнд источника типа word из операнда приёмника типа word. Если флаг переноса установлен, SUBC вычитает 1 из результата. Сохраняет результат в приёмнике и устанавливает флаг переноса при заёме. (DEST) ← (DEST) – (SRC) – (C)	DEST, SRC SUBC wreg, waop (101010aa)(waop)(wreg)

Продолжение таблицы А.2

Обозначение	Операция	Формат команды
SUBCB	SUBTRACT BYTES WITH BORROW (Вычитание байтов с заёмом). Вычитает операнд источника типа byte из операнда приёмника типа byte. Если флаг переноса установлен, SUBC вычитает 1 из результата. Сохраняет результат в приёмнике и устанавливает флаг переноса при заёме. (DEST) ← (DEST) – (SRC) – (C)	DEST, SRC SUBCB breg, baop (101110aa)(baop)(breg)
TIJMP	TABLE INDIRECT JUMP (Табличный косвенный переход). Выполнение продолжается по адресу, выбранному из таблицы адресов. Первый регистр типа word, TBAZE, содержит 16-битный адрес начала таблицы. Второй регистр типа word, INDEX, содержит 16-битный адрес байта, который содержит индекс в таблице. Величина INDEX должна быть в пределах от 0 до 128. Операнд #byte, INDEX_MASK, - 8-битные непосредственно получаемые данные по маскирования INDEX. INDEX_MASK суммируется (AND) с INDEX для определения смещения (OFFSET). OFFSET умножается на 2, затем добавляется к базовому адресу (TBASE) для определения адреса приёмника (DEST X). [INDEX] AND INDEX_MASK =OFFSET (2*OFFSET)+ TBASE=DEST X PC ← (DEST X)	TIJMP wreg, [wreg], #byte (11100010)(wreg)(#byte)(wreg)
TRAP	SOFTWARE TRAP (Программное пошаговое выполнение). Эта команда ведёт к запросу прерывания, вектор которого расположен в 2010h. Эта операция не влияет на состояние флага разрешения прерываний в PSW(1). Запросы прерываний не могут следовать сразу после этой команды. SP ← SP - 2 (SP) ← PC PC ← (2010H)	TRAP (11110111) Примечание – Этой команды нет в версии 1.2 языка ассемблера 8096. Команда TRAP предназначена для использования средствами Intel. Эти средства не поддерживают применение пользователем этой команды.
XCH	EXCHANGE WORD (Обмен слов). Меняет значение операнда источника типа word со значением операнда приёмника типа word. (DEST) ← (SRC); (SRC) ← (DEST)	DEST, SRC XCH wreg, waop (00000100)(waop)(wreg) (00001011)(waop)(wreg)
XCHB	EXCHANGE BYTE (Обмен байтов). Меняет значение операнда источника типа byte со значением операнда приёмника типа byte. (DEST) ← (SRC); (SRC) ← (DEST)	DEST, SRC XCHB breg, baop (00010100)(baop)(breg) (00011011)(baop)(breg)

Окончание таблицы А.2

Обозначение	Операция	Формат команды
XOR	LOGICAL EXCLUSIVE-OR WORDS (Логическое "исключающее ИЛИ" слов). Логически складывает операнды типа word источника и приёмника и сохраняет результат в приёмнике. Результат имеет "1" в битовой позиции, если один из операндов (но не оба) имеет "1" в этой позиции, и нули при всех других значениях битовых позиций. (DEST) ← (DEST)XOR(SRC)	DEST, SRC XOR wreg, waop (100001aa)(waop)(wreg)
XORB	LOGICAL EXCLUSIVE-OR BYTES (Логическое "исключающее ИЛИ" байтов). Логически складывает операнды типа byte источника и приёмника и сохраняет результат в приёмнике. Результат имеет "1" в битовой позиции, если один из операндов (но не оба) имеет "1" в этой позиции, и нули при всех других значениях битовых позиций. (DEST) ← (DEST)XOR(SRC)	DEST, SRC XORB breg, baop (100101aa)(baop)(breg)
Примечания 1 Сдвиг (disp) расширяется до 16 бит со знаком. 2 В этой операции DEST/2 заменяет деление без знака. 3 В этой операции DEST/2 заменяет деление со знаком.		

Таблицы А.3 и А.4 определяют аббревиатуры и символы, используемые в таблицах А.5 и А.6. Таблица А.5 показывает влияние каждой команды на Флаги Слова Состояния Программы (PSW), а таблица А.6 показывает влияние флагов PSW или специального регистрового бита на команды условного перехода.

Таблица А.3 – Обозначение флагов PSW

Обозначение	Название флага PSW
C	Carry Flag - Флаг Переноса
N	Negative Flag - Флаг Отрицательного Результата
ST	Sticky Bit Flag - Флаг "Дополнительного Бита"
V	Overflow Flag - Флаг Переполнения
VT	Overflow-Trap Flag - Дополнительный Флаг Переполнения
Z	Zero Flag - Флаг Нуля

Таблица А.4 - Символы установки флагов PSW

Символ	Описание
+	Команда устанавливает или сбрасывает флаг, то есть модифицирует, когда требуется.
-	Команда не модифицирует флаг.
↓	Команда сбрасывает флаг, если требуется, но не устанавливает.
↑	Команда устанавливает флаг, если требуется, но не сбрасывает.
1	Команда устанавливает флаг.
0	Команда сбрасывает флаг.
?	Команда оставляет флаг в неопределённом состоянии.

Таблица А.5 – Влияние команд на установку флагов PSW

Обозначение	Установка флага PSW					
	Z	N	C	V	VT	ST
ADD, ADDB	Z	+	+	+	↑	-
ADDC, ADDCB	+	+	+	+	↑	-
AND, ANDB	↓	+	0	0	-	-
BMOV, BMOVI	+	-	-	-	-	-
BR (Косвенный)	-	-	-	-	-	-
CLR, CLRB	-	0	0	0	-	-
CLRC	1	-	0	-	-	-
CLRVT	-	-	-	-	0	-
CMP, CMPB	-	+	+	+	↑	-
CMPL	+	+	+	+	+	-
DEC, DECB	+	+	+	+	↑	-
DI	+	-	-	-	-	-
DIV, DIVB, DIVU, DIVUB	-	-	-	+	↑	-
DJNZ, DJNZW	-	-	-	-	-	-
DPTS	-	-	-	-	-	-
EI	-	-	-	-	-	-
EPTS	-	-	-	-	-	-
EXT, EXTB	-	+	0	0	-	-
IDLPD Верный параметр	+	-	-	-	-	-
Неверный параметр	-	0	0	0	0	0
INC	0	+	+	+	↑	0
INCB	+	+	+	+	↑	-
JBC, JBS, JC, JE, JGE, JGT, JH, JLE, JLT, JNC, JNE, JNH, JNST, JNV	+	-	-	-	-	-
JNVT	-	-	-	-	0	-
JST, JV	-	-	-	-	-	-
JVT	-	-	-	-	0	-
LCALL, LD, LDB, LDBSE, LDBZE	-	-	-	-	-	-
LJMP	-	-	-	-	-	?
MUL, MULB, MULU, MULUB	-	-	-	-	-	?
NEG, NEGB	-	+	+	+	↑	-
NOP	+	-	-	-	-	-
NORML	-	?	0	-	-	-
NOT, NOTB	+	+	0	0	-	-
OR, ORB	+	+	0	0	-	-
POP	+	-	-	-	-	-
POPA, POPF	-	+	+	+	+	+
PUSH	+	-	-	-	-	-
PUSHA, PUSHF	-	0	0	0	0	0
RET	0	-	-	-	-	-
RST	-	0	0	0	0	0
SCALL	0	-	-	-	-	-
SETC	-	-	1	-	-	-
SHL, SHLB, SHLL	-	?	+	+	↑	-
SHR	+	0	+	0	-	+
SHRA, SHRAB, SHRAL	+	+	+	0	-	+
SHRB, SHRL	+	0	+	0	-	+
SJMP	+	-	-	-	-	-
SKIP	-	-	-	-	-	-
ST, STB	-	-	-	-	-	-
SUB, SUBB	-	+	+	+	↑	-
SUBC, SUBCB	+	+	+	+	↑	-

Окончание таблицы А.5

Обозначение	Установка флага PSW					
	Z	N	C	V	VT	ST
TIJMP	↓	-	-	-	-	-
TRAP	-	-	-	-	-	-
XCH, XCHB	-	-	-	-	-	-
XOR, XORB	-	+	0	0	-	-
	+					

Таблица А.6 – Влияние флагов PSW или Тестируемых Битов на Команды Условного Перехода

Команда	Переход осуществляется, если	Продолжает, если
DJNZ	Декрементированный байт <> 0	Декрементированный байт=0
DJNZW	Декрементированное слово <> 0	Декрементированное слово=0
JBC	Выбранный регистровый бит=0	Выбранный регистровый бит=1
JBS	Выбранный регистровый бит=1	Выбранный регистровый бит=0
JNC	C = 0	C = 1
JNH	C = 0 OR Z = 1	C = 1 AND Z = 0
JC	C = 1	C = 0
JH	C = 1 AND Z = 0	C = 0 OR Z = 1
JGE	N = 0	N = 1
JGT	N = 0 AND Z = 0	N = 1 OR Z = 1
JLT	N = 1	N = 0
JLE	N = 1 OR Z = 1	N = 0 AND Z = 0
JNST	ST = 0	ST = 1
JST	ST = 1	ST = 0
JNV	V = 0	V = 1
JV	V = 1	V = 0
JNVT	VT = 0	VT = 1 (сбрасывает VT)
JVT	VT = 1 (сбрасывает VT)	VT = 0
JNE	Z = 0	Z = 1
JE	Z = 1	Z = 0

В таблице А.7 приводится список кодов команд по возрастанию, с соответствующими обозначениями команд.

Таблица А.7 – Коды команд МК

16-ричный код	Обозначение команды	16-ричный код	Обозначение команды
00	SKIP	89	CMP Immediate
01	CLR	8A	CMP Indirect
02	NOT	8B	CMP Indexed
03	NEG	8C	DIVU Direct
04	XCH	8D	DIVU Immediate
05	DEC	8E	DIVU Indirect
06	EXT	8F	DIVU Indexed
07	INC	90	ORB Direct
08	SHR	91	ORB Immediate
09	SHL	92	ORB Indirect
0A	SHRA	93	ORB Indexed
0B	XCH	94	XORB Direct
0C	SHRL	95	XORB Immediate

Продолжение таблицы А.7

16-ричный код	Обозначение команды	16-ричный код	Обозначение команды
0D	SHLL	96	XORB Indirect
0E	SHRAL	97	XORB Indexed
0F	NORML	98	CMPB Direct
10	Reserved	99	CMPB Immediate
11	CLRB	9A	CMPB Indirect
12	NOTB	9B	CMPB Indexed
13	NEGB	9C	DIVUB Direct
14	XCHB	9D	DIVUB Immediate
15	DECB	9E	DIVUB Indirect
16	EXTB	9F	DIVUB Indexed
17	INCB	A0	LD Direct
18	SHRB	A1	LD Immediate
19	SHLB	A2	LD Indirect
1A	SHRAB	A3	LD Indexed
1B	XCHB	A4	ADDC Direct
1C-1F	Reserved	A5	ADDC Immediate
20-27	SJMP	A6	ADDC Indirect
28-2F	SCALL	A7	ADDC Indexed
30-37	JBC	A8	SUBC Direct
38-3F	JBS	A9	SUBC Immediate
40	AND Direct (3 ops)	AA	SUBC Indirect
41	AND Immediate(3 ops)	AB	SUBC Indexed
42	AND Indirect (3 ops)	AC	LDBZE Direct
43	AND Indexed (3 ops)	AD	LDBZE Immediate
44	ADD Direct (3 ops)	AE	LDBZE Indirect
45	ADD Immediate(3 ops)	AF	LDBZE Indexed
46	ADD Indirect (3 ops)	B0	LDB Direct
47	ADD Indexed (3 ops)	B1	LDB Immediate
48	SUB Direct (3 ops)	B2	LDB Indirect
49	SUB Immediate(3 ops)	B3	LDB Indexed
4A	SUB Indirect (3 ops)	B4	ADDCB Direct
4B	SUB Indexed (3 ops)	B5	ADDCB Immediate
4C	MULU Direct (3 ops)	B6	ADDCB Indirect
4D	MULU Immediate(3 ops)	B7	ADDCB Indexed
4E	MULU Indirect (3 ops)	B8	SUBCB Direct
4F	MULU Indexed (3 ops)	B9	SUBCB Immediate
50	ANDB Direct (3 ops)	BA	SUBCB Indirect
51	ANDB Immediate(3 ops)	BB	SUBCB Indexed
52	ANDB Indirect (3 ops)	BC	LDBSE Direct
53	ANDB Indexed (3 ops)	BD	LDBSE Immediate
54	ADDB Direct (3 ops)	BE	LDBSE Indirect
55	ADDB Immediate(3 ops)	BF	LDBSE Indexed
56	ADDB Indirect (3 ops)	C0	ST Direct
57	ADDB Indexed (3 ops)	C1	BMOV
58	SUBB Direct (3 ops)	C2	ST Indirect
59	SUBB Immediate(3 ops)	C3	ST Indexed
5A	SUBB Indirect (3 ops)	C4	STB Direct
5B	SUBB Indexed(3ops)	C5	CMPL
5C	MULUB Direct (3 ops)	C6	STB Indirect
5D	MULUB Immediate(3 ops)	C7	STB Indexed
5E	MULUB Indirect (3 ops)	C8	PUSH Direct
5F	MULUB Indexed (3 ops)	C9	PUSH Immediate

Окончание таблицы А.7

16-ричный код	Обозначение команды	16-ричный код	Обозначение команды
60	AND Direct (3 ops)	CA	PUSH Indirect
61	AND Immediate(3 ops)	CB	PUSH Indexed
62	AND Indirect (3 ops)	CC	POP Direct
63	AND Indexed (3 ops)	CD	BMOVI
64	ADD Direct (3 ops)	CE	POP Indirect
65	ADD Immediate(3 ops)	CF	POP Indexed
66	ADD Indirect (3 ops)	D0	JNST
67	ADD Indexed (3 ops)	D1	JNH
68	SUB Direct (3 ops)	D3	JNC
69	SUB Immediate(3 ops)	D4	JNVT
6A	SUB Indirect (3 ops)	D5	JNV
6B	SUB Indexed (3 ops)	D6	JGE
6C	MULU Direct (3 ops)	D7	JNE
6D	MULU Immediate(3 ops)	D8	JST
6E	MULU Indirect (3 ops)	D9	JH
6F	MULU Indexed (3 ops)	DA	JLE
70	ANDB Direct (3 ops)		
71	ANDB Immediate(3 ops)		
72	ANDB Indirect (3 ops)		
73	ANDB Indexed (3 ops)		
74	ADDB Direct (3 ops)		
75	ADDB Immediate(3 ops)		
76	ADDB Indirect (3 ops)		
77	ADDB Indexed (3 ops)		
78	SUBB Direct (3 ops)		
79	SUBB Immediate(3 ops)		
7A	SUBB Indirect (3 ops)		
7B	SUBB Indexed (3 ops)		
7C	MULUB Direct (3 ops)		
7D	MULUB Immediate (3 ops)		
7E	MULUB Indirect (3 ops)		
7F	MULUB Indexed (3 ops)		
80	OR Direct		
81	OR Immediate		
82	OR Indirect		
83	OR Indexed		
84	XOR Direct		
85	XOR Immediate		
86	XOR Indirect		
87	XOR Indexed		
88	CMP Direct		

* Код EE зарезервирован; однако он не генерирует прерывания невыполнимого кода.

** Знаковое умножение и деление, 2-байтные команды. Для любой знаковой команды первый байт - "FE", второй - код соответствующей буззнаковой команды. Например: код MULU (3 operands) direct "4C", поэтому код MUL (3 operands) direct - "FE4C".

Таблица А.8 – это карта кодов команд МК. Первая цифра кода команды по вертикали, вторая – по горизонтали. Соответствующее обозначение команды показано на пересечении двух чисел

Таблица А.8 – Схема кодов МК

Код	x0	x1	x2	x3	x4	x5	x6	x7
0x	SKIP	CLR	NOT	NEG	XCH di	DEC	EXT	INC
1x	***	CLRB	NOTB	NEGB	XCHB di	DECB	EXTB	INCB
2x	SJMP							
3x	JBC							
	bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7
4x	AND 3op				ADD 3op			
	di	im	in	ix	di	im	in	ix
5x	ANDB 3op				ADDB 3op			
	di	im	in	ix	di	im	in	ix
6x	AND 2op				ADD 2op			
	di	im	in	ix	di	im	in	ix
7x	ANDB 2op				ADDB 2op			
	di	im	in	ix	di	im	in	ix
8x	OR				XOR			
	di	im	in	ix	di	im	in	ix
9x	ORB				XORB			
	di	im	in	ix	di	im	in	ix
Ax	LD				ADDC			
	di	im	in	ix	di	im	in	ix
Bx	LDB				ADDCB			
	di	im	in	ix	di	im	in	ix
Cx	ST	BMOV	ST		STB	CMPL	STB	
	di	-	in	ix	di	-	in	ix
Dx	JNST	JNH	JGT	JNC	JNVT	JNV	JGE	JNE
Ex	DJNZ	DJNZW	TIJMP	BR in	***	***	***	LJMP
Fx	RET	***	PUSHF	POPF	PUSHA	POPA	IDLPD	TRAP

Окончание таблицы А.8

Код	x8	x9	xA	xB	xC	xD	xE	xF
0x	SHR	SHL	SHRA	XCH in	SHRL	SHLL	SHRAL	NORML
1x	SHRB	SHLB	SHRAB	XCHB in	***	***	***	***
2x	SCALL							
3x	JBS							
	bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7
4x	SUB 3op				MULU 3op *			
	di	im	in	ix	di	im	in	ix
5x	SUBB 3op				MULUB 3op *			
	di	im	in	ix	di	im	in	ix
6x	SUB 2op				MULU 2op *			
	di	im	in	ix	di	im	in	ix
7x	SUBB 2op				MULUB 2op *			
	di	im	in	ix	di	im	in	ix
8x	CMP				DIVU *			
	di	im	in	ix	di	im	in	ix
9x	CMPB				DIVUB *			
	di	im	in	ix	di	im	in	ix
Ax	SUBC				LDBZE			
	di	im	in	ix	di	im	in	ix
Bx	SUBCB				LDBSE			
	di	im	in	ix	di	im	in	ix
Cx	PUSH				POP	BMOVI	POP	
	di	im	in	ix	di	-	in	ix
Dx	JST	JH	JLE	JC	JVT	JV	JLT	JE
Ex	***	***	***	***	DPTS	EPTS	**	LCALL
Fx	CLRC	SETC	DI	EI	CLRVT	NOP	*	RST

* Знаковое умножение и деление- 2-байтные команды. Для любой знаковой команды первый байт - "FE", второй - код соответствующей незнаковой команды.

** Код EE зарезервирован; однако он не генерирует прерывания невыполнимого кода.

*** Резервный код команды.

В таблице А.9 приведён список команд с их длинами и кодами для каждого режима адресации. Каждый режим адресации занимает два столбца. В первом столбце приведены длины команд, а во втором - 16-ричные коды. Для индексных команд в первом столбце длины команд приведены в виде S/L, где S - длина коротко-индексной команды и L -длина длинно-индексной команды. Прочерк (-) в любом столбце показывает, что данный способ адресации для данной команды не применим.

Таблица А.9 – Длина команды и 16-ричный код

Арифметические команды (Группа 1)								
Обозначение	Прямая		Непосредственная		Косвенная (1)		Индексная S/L (1)	
ADD (2 ops)	3	64	4	65	3	66	4/5	67
ADD (3 ops)	4	44	5	45	4	46	5/6	47
ADDB (2 ops)	3	74	3	75	3	76	4/5	77
ADDB (3 ops)	4	54	4	55	4	56	5/6	57
ADDC	3	A4	4	A5	3	A6	4/5	A7
ADDCB	3	B4	3	B5	3	B6	4/5	B7

Продолжение таблицы A.9

Обозначение	Прямая		Непосредственная		Косвенная (1)		Индексная S/L (1)	
CMP	3	88	4	89	3	8A	4/5	8B
CMPB	3	98	3	99	3	9A	4/5	9B
SUB (2 ops)	3	68	4	69	3	6A	4/5	6B
SUB (3 ops)	4	48	5	49	4	4A	5/6	4B
SUBB (2 ops)	3	78	3	79	3	7A	4/5	7B
SUBB (3 ops)	4	58	4	59	4	5A	5/6	5B
SUBC	3	A8	4	A9	3	AA	4/5	AB
SUBCB	4	B8	3	B9	3	BA	4/5	BB
Арифметические команды (Группа 2)								
Обозначение	Прямая		Непосредственная		Косвенная (1)		Индексная S/L (1)	
DIV	4	FE 8C	5	FE 8D	4	FE 8E	5/6	FE 8F
DIVB	4	FE 9C	4	FE 9D	4	FE 9E	5/6	FE 9F
DIVU	3	8C	4	8D	3	8E	4/5	8F
DIVUB	3	9C	3	9D	3	9E	4/5	9F
MUL (2 ops)	4	FE 6C	5	FE 6D	4	FE 6E	5/6	FE 6F
MUL (3 ops)	5	FE 4C	6	FE 4D	5	FE 4E	6/7	FE 4F
MULB (2 ops)	4	FE 7C	4	FE 7D	4	FE 7E	5/6	FE 7F
MULB (3 ops)	5	FE 5C	5	FE 5D	5	FE 5E	6/7	FE 5F
MULU (2 ops)	3	6C	4	6D	3	6E	4/5	6F
MULU (3 ops)	4	4C	5	4D	4	4E	5/6	4F
MULUB(2 ops)	3	7C	3	7D	3	7E	4/5	7F
MULUB(3 ops)	4	5C	4	5D	4	5E	5/6	5F

Продолжение таблицы А.9

Логические команды								
Обозначение	Прямая		Непосредственная		Косвенная (1)		Индексная S/L (1)	
AND (2 ops)	3	60	4	61	3	62	4/5	63
AND (3 ops)	4	40	5	41	4	42	5/6	43
ANDB (2 ops)	3	70	3	71	3	72	4/4	73
ANDB (3 ops)	4	50	4	51	4	52	5/5	53
OR	3	80	4	81	3	82	4/5	83
ORB	3	90	3	91	3	92	4/5	93
XOR	3	84	4	85	3	86	4/5	87
XORB	3	94	3	95	3	96	4/5	97
Стековые команды								
Обозначение	Прямая		Непосредственная		Косвенная (1)		Индексная S/L (1)	
POP	2	CC	-	-	2	CE	s	CF
POPA	-	-	-	-	1	F5	-	-
POPF	-	-	-	-	1	F3	-	-
PUSH	2	C8	3	C9	2	CA	S	св
PUSHA	-	-	-	-	1	F4	-	-
PUSHF	-	-	-	-	1	F2	-	-
Команды работы с данными								
Обозначение	Прямая		Непосредственная		Косвенная (1)		Индексная S/L (1)	
LD	3	A0	4	A1	3	A2	4/5	A3
LDB	3	B0	3	B1	3	B2	4/5	B3
LDBSE	3	BC	3	BD	3	BE	4/5	BF
LDBZE	3	AC	3	AD	3	AE	4/5	AF
ST	3	C0	-	-	3	C2	4/5	C3
STB	3	C4	-	-	3	C6	4/5	C7
XCH	3	04	-	-	-	-	4/5	0B
XCHB	3	14	-	-	-	-	4/5	1B
Команды перехода								
Обозначение	Прямая		Непосредственная		Косвенная (1)		Индексная S/L (1)	
BR (Indirect)	-	-	-	-	2	E3	2	E3
LJMP	-	-	-	-	-	-	-/2	E7
SJMP	-	-	-	-	-	-	2/-	20-27 (2)
TIJMP	4	E2	4	E2	-	-	-/4	E2
Команды вызова								
Обозначение	Прямая		Непосредственная		Косвенная (1)		Индексная S/L (1)	
LCALL	-	-	-	-	-	-	-/3	EF
SCALL	-	-	-	-	-	-	2/-	28-2F (2)
RET	-	-	-	-	1	F0	-	-
TRAP	1	F7	-	-	-	-	-	-

Продолжение таблицы А.9

Команды условного перехода								
Обозначение	Прямая		Непосредственная		Косвенная (1)		Индексная S/L (1)	
DJNZ	-	-	-	-	-	-	3/-	E0
DJNZW	-	-	-	-	-	-	3/-	E1
JBC	-	-	-	-	-	-	3/-	30-37
JBS	-	-	-	-	-	-	3/-	38-3F
JC	-	-	-	-	-	-	1/-	DB
JE	-	-	-	-	-	-	1/-	DF
JGE	-	-	-	-	-	-	1/-	D6
JGT	-	-	-	-	-	-	1/-	D2
JH	-	-	-	-	-	-	1/-	D9
JLE	-	-	-	-	-	-	1/-	DA
JLT	-	-	-	-	-	-	1/-	DE
JNC	-	-	-	-	-	-	1/-	D3
JNE	-	-	-	-	-	-	1/-	D7
JNH	-	-	-	-	-	-	1/-	D1
JNST	-	-	-	-	-	-	1/-	D0
JNV	-	-	-	-	-	-	1/-	D5
JNVT	-	-	-	-	-	-	1/-	D4
JST	-	-	-	-	-	-	1/-	D8
JV	-	-	-	-	-	-	1/-	DD
JVT	-	-	-	-	-	-	1/-	DC
Команды сдвига								
Обозначение	Прямая		Непосредственная		Косвенная (1)		Индексная S/L (1)	
NORML	3	0F	-	-	-	-	-	-
SHL	3	09	-	-	-	-	-	-
SHLB	3	19	-	-	-	-	-	-
SHLL	3	0D	-	-	-	-	-	-
SHR	3	08	-	-	-	-	-	-
SHRA	3	0A	-	-	-	-	-	-
SHRAB	3	1A	-	-	-	-	-	-
SHRAL	3	0E	-	-	-	-	-	-
SHRB	3	18	-	-	-	-	-	-
SHRL	3	0C	-	-	-	-	-	-
Блочные команды								
Обозначение	Прямая		Непосредственная		Косвенная (1)		Индексная S/L (1)	
BMOV	-	-	-	-	-	-	-/3	C1
BMOVI	-	-	-	-	-	-	-/3	CD
Специальные команды								
Обозначение	Прямая		Непосредственная		Косвенная (1)		Индексная S/L (1)	
CLRC	1	F8	-	-	-	-	-	-
CLRVT	1	FC	-	-	-	-	-	-
DI	1	FA	-	-	-	-	-	-
EI	1	FB	-	-	-	-	-	-
IDLPD	-	-	1	F6	-	-	-	-
NOP	1	FD	-	-	-	-	-	-
RST	1	FE	-	-	-	-	-	-
SETC	1	F9	-	-	-	-	-	-
SKIP	2	00	-	-	-	-	-	-

Окончаниение таблицы А.9

Команды управления PTS								
Обозначение	Прямая		Непосредственная		Косвенная (1)		Индексная S/L (1)	
DPTS	1	EC	-	-	-	-	-	-
EPTS	1	ED	-	-	-	-	-	-
Остальные команды								
Обозначение	Прямая		Непосредственная		Косвенная (1)		Индексная S/L (1)	
CLR	-	01	-	-	-	-	-	-
CLRB	-	11	-	-	-	-	-	-
CMPL	-	C5	-	-	-	-	-	-
DEC	-	05	-	-	-	-	-	-
DECB	-	15	-	-	-	-	-	-
EXT	-	06	-	-	-	-	-	-
EXTB	-	16	-	-	-	-	-	-
INC	-	07	-	-	-	-	-	-
INCB	-	17	-	-	-	-	-	-
NEG	-	03	-	-	-	-	-	-
NEGB	-	13	-	-	-	-	-	-
NOT	-	02	-	-	-	-	-	-
NOTB	-	12	-	-	-	-	-	-

Примечания

1 Косвенная обычная и косвенная автоинкрементная адресация имеют одинаковые коды команд, также как и коротко- и длинно-индексная адресация. Для использования косвенного обычного или коротко-индексного режима второй байт команды должен быть чётным. Для использования косвенного авто-инкрементного или длинно-индексного режима второй байт должен быть нечётным.

2 Для этих команд (SCALL, SJMP), 3 младших значащих бита кода команды соединяются с 8 битами для формирования 11-битного дополнительного кода смещения.

В таблице А.10 приведён список команд в алфавитном порядке, по группам, с их временем выполнения, измеренным в State Times – машинных тактах. В таблице А.10 время выполнения для косвенного и индексного режимов адресации обозначены как R/M, где R – время выполнения с использованием SFRs и внутреннего RAM (0H – 1FFh), а M – время выполнения с использованием контроллера памяти (200h – 0FFFFh).

Таблица А.10 – Время выполнения команд МК (в машинных тактах)

Арифметические команды (Группа 1)						
Обозначение	Прямая	Непосред-ственная	Косвенная		Индексная	
			Обычная	Автоинкрементная	Короткая	Длинная
ADD (2 ops)	4	5	6/8	7/9	6/8	7/9
ADD (3 ops)	5	6	7/10	8/11	7/10	8/11
ADDB (2 ops)	4	5	6/8	7/9	6/8	7/9
ADDB (3 ops)	5	6	7/10	8/11	7/10	8/11
ADDC	4	5	6/8	7/9	6/8	7/9
ADDCB	4	5	6/8	7/9	6/8	7/9
CMP	4	5	6/8	7/9	6/8	7/9
CMPB	4	5	6/8	7/9	6/8	7/9
SUB (2 ops)	4	5	6/8	7/9	6/8	7/9
SUB (3 ops)	5	6	7/10	8/11	7/10	8/11
SUBB (2 ops)	4	5	6/8	7/9	6/8	7/9
SUBB (3 ops)	5	6	7/10	8/11	7/10	8/11
SUBC	4	5	6/8	7/9	6/8	7/9
SUBCB	4	5	6/8	7/9	6/8	7/9
Арифметические команды (Группа 2)						
Обозначение	Прямая	Непосред-ственная	Косвенная		Индексная	
			Обычная	Автоинкрементная	Короткая	Длинная
DIV	26	27	28/31	29/32	29/32	30/33
DIVB	18	18	20/23	21/24	21/24	22/25
DIVU	24	25	26/29	27/30	27/30	28/31
DIVUB	16	16	18/21	19/22	19/22	20/23
MUL (2 ops)	16	17	18/21	19/22	19/22	20/23
MUL (3 ops)	16	17	18/21	19/22	19/22	20/23
MULB (2 ops)	12	12	14/17	15/18	15/18	16/19
MULB (3 ops)	12	12	14/17	15/18	15/18	16/19
MULU (2 ops)	14	15	16/19	17/19	17/20	18/21
MULU (3 ops)	14	15	16/19	17/19	17/20	18/21
MULUB(2 ops)	10	10	12/15	13/15	12/16	14/17
MULUB(3 ops)	10	10	15/15	12/16	12/16	14/17
Логические команды						
Обозначение	Прямая	Непосред-ственная	Косвенная		Индексная	
			Обычная	Автоинкрементная	Короткая	Длинная
AND (2ops)	4	5	6/8	7/9	6/8	7/9
AND (3 ops)	5	6	7/10	8/11	7/10	8/11
ANDB (2 ops)	4	5	6/8	7/9	6/8	7/9
ANDB (3 ops)	5	6	7/10	8/11	7/10	8/11
OR	4	5	6/8	7/9	6/8	7/9
ORB	4	5	6/8	7/9	6/8	7/9
XOR	4	5	6/8	7/9	6/8	7/9
XORB	4	5	6/8	7/9	6/8	7/9

Продолжение таблицы А.10

Стековые команды (Внутренний Стек)						
Обозначение	Прямая	Непосред-ственная	Косвенная		Индексная	
			Обычная	Автоинкрементная	Короткая	Длинная
POP	8	-	10/12	11/13	11/13	12/14
POPA	12	-	-	-	-	-
POPF	7	-	-	-	-	-
PUSH	6	7	9/12	10/13	10/13	11/14
PUSHA	12	-	-	-	-	-
PUSHF	6	-	-	-	-	-
Стековые команды (Внешний Стек)						
Обозначение	Прямая	Непосред-ственная	Косвенная		Индексная	
			Обычная	Автоинкрементная	Короткая	Длинная
POP	11	-	13/15	14/16	14/16	15/17
POPA	18	-	-	-	-	-
POPF	10	-	-	-	-	-
PUSH	8	9	11/14	12/15	12/15	13/16
PUSHA	18	-	-	-	-	-
PUSHF	8	-	-	-	-	-
Команды работы с данными						
Обозначение	Прямая	Непосред-ственная	Косвенная		Индексная	
			Обычная	Автоинкрементная	Короткая	Длинная
LD	4	5	5/8	6/8	6/9	7/10
LDB	4	5	5/8	6/8	6/9	7/10
LDBSE	4	4	5/8	6/8	6/9	7/10
LDBZE	4	4	5/8	6/8	6/9	7/10
ST	4	-	5/8	6/9	6/9	7/10
STB	4	-	5/8	6/9	6/9	7/10
XCH	5	-	-	-	8/13	9/14
XCHB	5	-	-	-	8/13	9/14
Команды перехода						
Обозначение	Прямая	Непосред-ственная	Косвенная		Индексная	
			Обычная	Автоинкремент-ная	Короткая	Длинная
BR (Indirect)	-	-	7	7	7	7
LJMP	-	-	-	-	-	7
SJMP	-	-	-	-	7	-
TIJMP internal/ internal external/ internal external/ external	15 18 21	15 18 21				
Команды вызова (Внутренний Стек)						
Обозначение	Прямая	Непосред-ственная	Косвенная		Индексная	
			Обычная	Автоинкремент-ная	Короткая	Длинная
LCALL	-	-	-	-	-	11
RET	-	-	11	-	-	-
SCALL	-	-	-	-	11	-
TRAP	16	-	-	-	-	-

Продолжение таблицы А.10

Команды вызова (Внешний Стек)						
Обозначение	Прямая	Непосред-ственная	Косвенная		Индексная	
			Обычная	Автоинкрементная	Короткая	Длинная
LCALL	-	-	-	-	-	13
RET	-	-	14	-	-	-
SCALL	-	-	-	-	13	-
TRAP	18	-	-	-	-	-
Команды условного перехода						
Обозначение	Коротко-индексная адресация					
DJNZ	5 (перехода нет), 9 (переход есть)					
DJNZW	7 (перехода нет), 11 (переход есть) 6 (перехода нет), 10 (переход есть)					
JBC	5 (перехода нет), 9 (переход есть)					
JBS	5 (перехода нет), 9 (переход есть)					
JC	4 (перехода нет), 8 (переход есть)					
JE	4 (перехода нет), 8 (переход есть)					
JGE	4 (перехода нет), 8 (переход есть)					
JGT	4 (перехода нет), 8 (переход есть)					
JH	4 (перехода нет), 8 (переход есть)					
JLE	4 (перехода нет), 8 (переход есть)					
JLT	4 (перехода нет), 8 (переход есть)					
JNC	4 (перехода нет), 8 (переход есть)					
JNE	4 (перехода нет), 8 (переход есть)					
JNH	4 (перехода нет), 8 (переход есть)					
JNST	4 (перехода нет), 8 (переход есть)					
JNV	4 (перехода нет), 8 (переход есть)					
JNVT	4 (перехода нет), 8 (переход есть)					
JST	4 (перехода нет), 8 (переход есть)					
JV	4 (перехода нет), 8 (переход есть)					
JVT	4 (перехода нет), 8 (переход есть)					
Команды сдвига						
Обозначение	Прямая адресация					
NORML	8 + 1 на сдвиг (9 - сдвига нет)					
SHL	6 + 1 на сдвиг (7 - сдвига нет)					
SHLB	6 + 1 на сдвиг (7 - сдвига нет)					
SHR	6 + 1 на сдвиг (7 - сдвига нет)					
SHRA	6 + 1 на сдвиг (7 - сдвига нет)					
SHRAB	6 + 1 на сдвиг (7 - сдвига нет)					
SHRAL	7 + 1 на сдвиг (8 - сдвига нет)					
SHRB	6 + 1 на сдвиг (7 - сдвига нет)					
SHRL	7 + 1 на сдвиг (8 - сдвига нет)					
Блочные команды						
Обозначение	Длинно-индексная адресация					
BMOV	internal/internal 6 + 8 на слово external/internal 6 + 11 на слово external/external 6 + 14 на слово					
BMOVI	internal/internal 7 + 8 на слово + 14 на прерывание external/internal 7 + 11 на слово + 14 на прерывание external/external 7 + 14 на слово + 14 на прерывание					

Окончание таблицы А.10

Специальные команды						
Обозначение	Прямая	Непосред-ственная	Косвенная		Индексная	
			Обычная	Автоинкрементная	Короткая	Длинная
CLRC	2	–	–	–	–	–
CLRVT	2	–	–	–	–	–
DI	2	–	–	–	–	–
EI	2	–	–	–	–	–
IDLPD неверный параметр правильный параметр	– –	8 25	– –	– –	– –	– –
NOP	2	–	–	–	–	–
RST (включая выборку байта конфигурации)	20	–	–	–	–	–
SETC	2	–	–	–	–	–
SKIP	3	–	–	–	–	–
Команды управления PTS						
Обозначение	Прямая	Непосред-ственная	Косвенная		Индексная	
			Обычная	Автоинкрементная	Короткая	Длинная
DPTS	2	–	–	–	–	–
EPTS	2	–	–	–	–	–
Остальные команды						
Обозначение	Прямая	Непосред-ственная	Косвенная		Индексная	
			Обычная	Автоинкрементная	Короткая	Длинная
CLR	3	–	–	–	–	–
CLRB	3	–	–	–	–	–
CMPL	7	–	–	–	–	–
DEC	3	–	–	–	–	–
DECB	3	–	–	–	–	–
EXT	4	–	–	–	–	–
EXTB	4	–	–	–	–	–
INC	3	–	–	–	–	–
INCB	3	–	–	–	–	–
NEG	3	–	–	–	–	–
NEGB	3	–	–	–	–	–
NOT	3	–	–	–	–	–
NOTB	3	–	–	–	–	–

Таблица А.11 – Время выполнения PTS циклов МК

PTS циклы	
Режим PTS	Время выполнения в машинных циклах
Режим одиночного обмена: internal/internal external/internal external/external	 18 21 24
Режим блочного обмена: internal/internal external/internal external/external	 13 + 7 на обмен (минимум 1) 16 + 7 на обмен (минимум 1) 19 + 7 на обмен (минимум 1)
Режим A/D - Scan (многократное аналого-цифровое преобразование): SFR/Internal RAM Контроллер Памяти	 21 25
Режим HSI (высокоскоростного ввода): SFR/Internal RAM Контроллер Памяти	 12 + 10 на обмен (минимум 1) 16 + 10 на обмен (минимум 1)
Режим HSO (высокоскоростного вывода): SFR/Internal RAM Контроллер Памяти	 11 + 10 на обмен (минимум 1) 15 + 10 на обмен (минимум 1)

Приложение Б

(обязательное)

Описание сигналов

Это приложение содержит информацию о функциях выводов МК.

Таблица Б.1 характеризует колонки, используемые в таблице Б.2, которая описывает функции выводов.

Таблица Б.3 показывает состояние выводов управления и ввода-вывода после сброса, таблица Б.4 описывает терминологию состояния вывода.

Таблицы Б.5 и Б.6 показывают номера выводов корпуса вместе с соответствующими функциями.

Таблица Б.1 – Описание колонок таблицы Б.2

Название колонки	Описание
Имя	Описывает функции вывода, расположенные по алфавиту. Многие выводы имеют более чем одну функцию. Каждая функция вывода занесена в эту колонку; если он имеет дополнительные функции, то они описаны в колонке "Дополнительная функция ".
Дополнительная функция	Описывает все другие функции, обеспечиваемые этим выводом.
Выбор	Описывает установки регистра, состояние вывода или другие условия, которые заставляют вывод функционировать, как описано в колонке "Имя". Тире (-) показывает "нет" или "не применяется".
Тип	Определяет функцию вывода, описанную в колонке "Имя", как вход (I), выход (O), двунаправленный (I/O), квазидвунаправленный (QBD), питание (PWR) или земля (GND). Примечательно, что все входы, за исключением RESET#, являются фиксирующими входами. RESET# является уровнечувствительным входом. Во время режима POWERDOWN, схема POWERDOWN использует вход EXTINT как уровнечувствительный вход.
Описание	Краткое описание функции вывода приведено в колонке "Имя".

Таблица Б.2 – Описание сигналов

Имя	Дополнительная функция	Выбор	Тип	Описание
ACH0 ACH1 ACH2 ACH3 ACH4 ACH5 ACH6 ACH7	P0.0 P0.1 P0.2 P0.3 P0.4/PMODE.0 P0.5/ PMODE.1 P0.6/PMODE.2 P0.7/PMODE.3	—	I	Аналоговые каналы с 0 до 7. ACH0 – ACH7 являются аналоговыми входами АЦП. Эти выводы могут использоваться отдельно как аналоговый вход (ACHx) или цифровой вход (P0.x). Хоть и возможно одновременное функционирование этих входов как аналоговых и цифровых, но это не применяется, так как чтение Port0 во время преобразования может привести к неправильному результату преобразования. ∩0V и ∩VCC2 выводы должны быть подключены для функционирования АЦП и порта 0.
AD0-AD7 AD8-AD15	P3.0-P3.7 P4.0-P4.7	шина доступа к внешним адресам	I/O	Системная шина адрес-данные. Эти выводы обеспечивают мультиплексированную шину адрес-данные. В течение адресной фазы шинного цикла адресные биты 0-15 выводятся на шину и могут быть зафиксированы с помощью ALE или ADV#. 8 или 16-битные данные передаются в течение фазы передачи данных.
ADV#	ALE	CCR.3=0	O	Значение адреса. Выходной сигнал становится активным только во время доступа к внешней памяти. ADV# показывает, что значение данного адреса действительно находится на системной шине адрес-данные. Этот сигнал сохраняет низкий уровень до тех пор, пока шинный цикл продолжается, и возвращается в высокий уровень, как только шинный цикл завершен. Внешняя защелка может использовать этот сигнал для выделения адреса с шины адрес/данные. Он также может использоваться с шифратором для выработки сигнала выборки кристалла внешней памяти.

Продолжение таблицы Б.2

Имя	Дополнительная функция	Выбор	Тип	Описание
AINC#	P2.4/T2RST	EA#=Vea (режим программирования)	I	Автоувеличение на 1. В режиме управляемого программирования (SLAVE Programming) этот входной сигнал с активным низким уровнем разрешает автоувеличения на 1 (auto-increment). Автоувеличение на 1 позволяет читать или записывать последовательные ячейки EPROM без передачи адреса по шине программирования, для каждого цикла чтения или записи.
ALE	ADV#	CCR.3=1	O	Разрешение функции адреса. Этот выходной сигнал становится активным только во время доступа к внешней памяти (активный уровень - высокий). ALE показывает,что значение адресной информации действительно находится на системной шине адрес/данные, и сигнализирует о начале цикла шины. ALE отличается от ADV# тем, что не остается активным в течение всего шинного цикла. Внешняя защелка может использовать этот сигнал для выделения адреса с шины адрес/данные
∩0V	—	—	GND	Опорная и логическая земля для АЦП, используется для чтения порта 0. ∩0V должна быть подключена для А/Ц преобразователя и порта 0.
BHE#	WRH#	CCR.2=1	O	Разрешение старшего байта. Это выходной сигнал с активным низким уровнем, устанавливающимся только при записи во внешнюю память слова или старшего байта. BHE# показывает, что данные передаются (записываются) через старшую половину системной шины адрес/данные. Чтобы выбрать записываемый байт памяти, BHE# используется вместе с A0: BHE# A0 Записываемый байт 0 0 оба байта 0 1 только старший байт 1 0 только младший байт

Продолжение таблицы Б.2

Имя	Дополнительная функция	Выбор	Тип	Описание
BREQ#	P1.5	WSR.7=1	O	Запрос шины. Это выходной сигнал с активным низким уровнем, устанавливающийся во время HOLD-цикла, когда контроллер шины ожидает цикл внешней памяти. Контроллер шины может установить BREQ# в любое время, если установлен HLDA#, при этом он остается активным до тех пор, пока HOLD# не сбросится. Если эта функция активна, выходной контакт действует как стандартный вывод (т. е. неквазидвунаправленный). Однажды разрешенная дополнительная функция функционального вывода BREQ# действует, пока устройство не сбросят.
BW	—	CCR.1=1	I	Ширина шины. Если CCR.1=1, этот сигнал выбирает ширину шины во время внешнего доступа. Когда BW высокий, выбирается 16-битная разрядность шины; когда низкий - 8-битная. Если CCR.1=0, ширина шины всегда8 бит, при этом BW- сигнал игнорируется.
CLKOUT	—	IOC3.1	O	Тактовый выход. Выход внутреннего тактового генератора. Частота CLKOUT равна 1/2 частоты на входе (XTAL1) и скважность сигнала CLKOUT равна 2. Очистка IOC3.1 разрешает CLKOUT; установка IOC3.1 запрещает сигнал.
CPVER	P2.6/T2UP-DN	EA#=Vea (режим программирования)	O	Общая верификация программы. Это выходной сигнал с высоким активным уровнем позволяет обнаружить любую ошибку сравнения, которая имела место в режиме программирования. CPVER остается высоким до тех пор, пока не появится ошибка сравнения, в это время он становится низким. Если существует хотя бы одна ошибка, CPVER остается в низком уровне, пока устройство не выйдет из режима программирования. Когда сигнал CPVER имеет высокий уровень, это показывает, что все ячейки запрограммированы верно во время режима программирования.

Продолжение таблицы Б.2

Имя	Дополнительная функция	Выбор	Тип	Описание
EA#	выбор режима программирования (EA#=Vea)	—	I	Доступ к внешней памяти. Этот сигнал с активным низким уровнем, разрешает доступ к памяти вне кристалла. Если уровень высокий, это выбирает внутрикристальный OTPROM. EA# фиксируется только по нарастающему фронту сигнал RESET#. Если EA#=Vea в момент нарастающего фронта сигнала RESET#, то устройство входит в режим программирования, выбираемый PMODE.0-PMODE.3. Для устройств без внутреннего ПЗУ, EA# должен быть всегда в низком уровне.
EXTINT	P2.2/PROG#	IOC1.1= 0	I	Внешнее прерывание. IOC1.1 назначает этот вход или на P2.2 или на P0.7. EXTINT должен удерживаться более чем два машинных такта, чтобы гарантировать его захват. EXTINT-динамический вход (sampled input), однако схема POWERDOWN использует его как чувствительный к уровню вход в режиме POWERDOWN. P2.2 всегда вырабатывает EXTINT1 прерывание (INT13, 203Ah). Когда IOC1.1=0, нарастающий фронт на выводе P2.2 также вырабатывает прерывание EXTINT (INT07, 200Eh). EXTINT и EXTINT1 должны удерживаться более чем два машинных такта, чтобы гарантировать, что они зафиксированы.
EXTINT	P0.7/PMODE. 3 /ACH7	IOC1.1=1	I	Когда IOC1.1=1, нарастающий фронт на входе P0.7 вырабатывает внешнее прерывание EXTINT(INT07, 200Eh). EXTINT должен удерживаться более чем два машинных такта, чтобы гарантировать его захват.

Продолжение таблицы Б.2

Имя	Дополнительная функция	Выбор	Тип	Описание
HLDA#	P1.6	WSR.7=1	O	Подтверждение запроса шины. Выход с активным низким уровнем показывает, что контроллер не управляет шиной. Это происходит в ответ на установку внешним устройством сигнала HOLD#. Когда эта функция активна, вход действует как стандартный выход (неквазидвунаправленный). Однажды разрешенная функция HLDA# для вывода HLDA# действует до тех пор, пока устройство не сбросят.
HOLD#	P1.7	WSR.7=1	I	Запрос шины. Это сигнал с активным низким уровнем, показывающий, что внешнее устройство запрашивает управление шиной. Если функция активна, вывод действует как стандартный вход (неквазидвунаправленный). Однажды разрешенная дополнительная функция для вывода действует как HOLD# до тех пор, пока устройство не сбросят.
HSI.0	INT04 прерывание/ источник сброса T2	IOC0.0=1	I	Вход для модуля высокоскоростного ввода. HSI.0 вывод может также использоваться для выработки прерывания. Нарастающий фронт на HSI.0 вырабатывает HSI.0 прерывание (INT04, 2008h). HSI.0 должен удерживаться более чем два машинных такта, чтобы гарантировать его захват. Дополнительно: когда IOC0.5=1, нарастающий фронт HSI.0 сбрасывает таймер 2.
HSI.1	Источник тактового сигнала для T2	IOC0.2=1	I	Вход для модуля высокоскоростного ввода. Когда IOC0.7=0 и IOC3.0=0, вывод HSI.1 действует как источник тактов T2.
HSI.2	HSO.4	IOC0.4=1	I	Вход для модуля высокоскоростного ввода. Примечание: функции HSI и HSO могут быть активными в одно время, в этом случае вывод действует как выход, управляемый HSO.

Продолжение таблицы Б.2

Имя	Дополнительная функция	Выбор	Тип	Описание
HSI.3	HSO.5	IOCO.6=1	I	Вход для модуля высокоскоростного ввода. Примечание: функции HSI и HSO могут быть активными в одно время, в этом случае вывод действует как выход, управляемый HSO.
HSO.0 HSO.1 HSO.2	— — —	— — —	O	Выход для модуля высокоскоростного вывода.
HSO.3	—	—		
HSO.4	HSI.2	IOC1.4=1	O	Выход для модуля высокоскоростного вывода. Примечание: функции HSI и HSO могут быть активными в одно время, в этом случае вывод действует как выход, управляемый HSO.
HSO.5	HSI.3	IOC1.6=1	O	Выход для модуля высокоскоростного вывода. Примечание: функции HSI и HSO могут быть активными в одно время, в этом случае вывод действует как выход, управляемый HSI.
INST	—	—	O	Выборка команды. Этот сигнал действителен только в цикле чтения внешней памяти. Если уровень сигнала высокий, INST показывает,что выбирается команда. Если сигнал низкого уровня - считываются данные. INST может использоваться в устройствах, которые требуют разделения байтов памяти для данных и команд. Примечание - CCB-байт и вектор прерывания рассматриваются в качестве данных.
NMI	—	—	I	Немаскируемое прерывание. Положительный перепад вызывает немаскируемое прерывание через вектор, размещенный в ячейке 203Eh. NMI должен удерживаться более чем один машинный такт, чтобы гарантировать его захват. NMI используется системами отладки фирмы Intel, это может привести к конфликтам с программным обеспечением пользователя. Когда NMI не используется, он должен быть подключен к низкому уровню.

Продолжение таблицы Б.2

Имя	Дополнительная функция	Выбор	Тип	Описание
P0.0 P0.1 P0.2 P0.3 P0.4 P0.5 P0.6 P0.7	ACH0 ACH1 ACH2 ACH3 ACH4/PMODE.0 ACH5/PMODE.1 ACH6/PMODE.2 ACH7/PMODE.3 /EXTINT	—	I	Порт 0. Этот порт является 8-битным, с высоким сопротивлением, порт только на ввод. Порт0 только читается через ячейку 0Eh в HWindow0. P0.0 - P0.7 являются цифровыми входами. Эти входы могут индивидуально использоваться на аналоговые входы (ACHx) или цифровые входы (P0.x). Возможно функционирование этих выводов одновременно как аналоговых, так и цифровых входов, но это не рекомендуется из-за того, что чтение порта0 во время А/Ц преобразования может привести к непредсказуемому результату преобразования. ∩0V и ∩VCC2 должны быть подключены для функционирования порта0 и А/Ц преобразователя.
P1.0 P1.1 P1.2 P1.3 P1.4 P1.5 P1.6 P1.7	— — — PWM1 PWM2 BREQ# HLDA# HOLD#	— — — IOC3.2=0 IOC3.3=0 WSR.7=0 WSR.7=0 WSR.7=0	QBD	Порт1. Это 8-битный квазидвунаправленный порт ввода-вывода. Порт1 читается и записывается через ячейку 0Fh в Hwindow 0. Дополнительные функции действуют как выводы стандартного ввода/вывода (неквазидвунаправленные).
P2.0 P2.1 P2.2 P2.3 P2.4 P2.5 P2.6 P2.7	TXD/PVER# RXD/PALE# EXTINT/ PROG# T2CLK T2RST/AINC# PWM0 T2UP-DN/CPVER T2CAP/PACT	IOC1.5=0 SPCON.3=0 — — — IOC1.0=0 — —	O I I I I O QBD QBD	Порт 2. Это многофункциональный 8-битный порт. Порт 2 читается и записывается через ячейки 10h в Hwindow 0.
P3.0 - P3.7 P4.0 - P4.7	AD0-AD7 AD8-AD15	EA#=1	I/O	Порт 3 и 4. Это 8-битные двунаправленные порты ввода/вывода с выходами с открытым стоком. Эти выводы образуют мультиплексированную шину адрес/данные, и имеют внутреннюю низкоомную нагрузку на U_{CC1}. Порты 3 и 4 могут быть прочитаны и записаны только как слово; в ячейке 1FFEh. Во время режима программирования эти порты действуют как PBUS.

Продолжение таблицы Б.2

Имя	Дополнительная функция	Выбор	Тип	Описание
PACT#	P2.7/ T2CAP	EA#=Vea (режим программи-рования)	O	Активное программирование. В режиме автоматического программирования низкий уровень сигнала PACT# показывает, что осуществляется программирование.
PALE#	P2.1/RXD	EA#=Vea (режим программи-рования)	I	Программирующий ALE. Когда PALE# установлен, команды и данные с портов 3 и 4 читаются в устройство.
PMODE.0 PMODE.1 PMODE.2 PMODE.3	P0.4/ACH4 P0.5/ACH5 P0.6/ACH6 P0.7/ACH7	EA#=Vea (режим программи-рования)	I	Выбор режима программирования. Определяют алгоритм программирования EPROM, который будет выполняться. PMODE фиксируется после сброса устройства, когда EA#=Vea; PMODE должны быть стабильными, пока устройство работает.
PROG#	P2.2 /EXTINT	EA#=Vea (режим программи-рования)	I	Начало программирования. Вход с активным низким уровнем имеет значение только во время подчиненного программирования (Slave Programming Mode). Когда установлен PROG#, это вызывает программирование данных с шины программирования в EPROM. Когда PROG# снимается, программирующие.
PVER	P2.0/TXD	EA#=Vea (режим программирования)	O	Сравнение программы. В режиме программирования этот выходной сигнал с активным высоким уровнем показывает, что слово запрограммировано верно.
PWM0	P2.5	IOC1.0=1	O	Выход широтно-импульсного модулятора 0 (PWM). Если PWM0 специально удерживается в высоком уровне в момент нарастающего фронта RESET#, то устройство входит в режим тестирования.
PWM1	P1.3	IOC3.2=1	O	Выход PWM1.
PWM2	P1.4	IOC3.3=1	O	Выход PWM2.
RD#	—	—	O	Внешнее чтение. Это выходной сигнал с активным низким уровнем, показывающий чтение внешней памяти.

Продолжение таблицы Б.2

Имя	Дополнительная функция	Выбор	Тип	Описание
READY	—	—	I	Вход готовности. Этот сигнал используется для удлинения цикла внешней памяти, выработавшей "состояния ожидания" для согласования с медленной памятью. Когда READY высокий, ЦПУ продолжает работать в нормальном режиме. Если READY принимает низкий уровень перед спадающим фронтом сигнала CLKOUT, контроллер памяти вводит циклы ожидания, пока в момент CLKOUT не будет высокого уровня на READY, или до тех пор, пока количество циклов ожиданий не будет равно количеству, запрограммированному в CCR.4 и CCR.5. READY игнорируется для всей внутренней памяти. READY является активным во время выборки CCR.
RESET#	—	—	I/O	Вход сброса и выход с открытым стоком из кристалла. Спадающий фронт сигнала RESET# инициирует процесс сброса. Когда RESET# устанавливается впервые, кристалл открывает транзистор с нагрузкой на Vss, соединенный с выводом RESET, на 16 машинных тактов. Эта функция может также быть активизирована переполнением таймера "сторож цикла" (Watchdog) или выполнением команды RST. В режиме POWERDOWN процесс сброса вызывает переход кристалла в нормальный режим работы.
RXD	P2.1/PALE#	SPCON.3=1	I/O	Последовательный вход данных. В режимах 1, 2 и 3 RXD используется для приема данных с последовательного порта. В режиме 0 он действует как вход или как выход.
T2CAP	P2.7/PACT#	—	QBD	Нарастающий фронт на P2.7 фиксирует значение таймера 2 в регистр T2CAPTURE и взводит прерывание фиксации таймера 2 (Timer Capture) (INT11, 2036h). T2CAP должен удерживаться более чем два машинных такта, чтобы гарантировать захват.

Продолжение таблицы Б.2

Имя	Дополнительная функция	Выбор	Тип	Описание
T2CLK	P2.3	IOC0.7=0 и IOC3.0=0 RATE. 15=0	I	Вход тактирования таймера 2 и вход генератора, задающего скорость передачи для последовательного порта.
T2RST	P2.4/AINC#	IOC0.3=1 и IOC0.5=0	I	Сброс таймера 2. Нарастающий фронт на T2RST сбрасывает таймер 2.
T2UP-DN	P2.6/CPVER	IOC2.1=1	I	Управление направлением счета таймера 2. Это вход с активным высоким уровнем управляет направлением счета таймера 2. Когда T2UP-DN имеет высокий уровень, таймер 2 считает на уменьшение ; когда T2UP-DN имеет низкий уровень, таймер 2 считает на увеличение.
TXD	P2.0/PVER	IOC1.5=1	O	Выход последовательных данных. В режимах 1, 2 и 3 TXD используется для передачи данных последовательного порта. В режиме 0 он используется как выход тактовых импульсов.
#U1	—	—	PWR	Напряжение питания цифровой части устройства (плюс 5 вольт)
Vpp	—	—	PWR	Напряжение программирования, а также времязадающий вывод для схемы "возврат из режима POWERDOWN".
∩U2	—	—	PWR	Опорное напряжение для А/Ц преобразователя, ∩VCC2 также является напряжением питания для аналоговой части А/Ц преобразователя и логики, используемой для чтения порта0. ∩VCC2 должен быть подключен для нормального функционирования А/Ц преобразователя и порта 0.
#0V	—	—	GND	Цифровая схемная земля (0 В). Имеется несколько выводов Vss, все они должны быть соединены.
WR#	WRL#	CCR.2=1	O	Внешняя запись. Это выходной сигнал с активным низким уровнем, устанавливающийся при записи во внешнюю память.

Окончание таблицы Б.2

Имя	Дополнительная функция	Выбор	Тип	Описание
WRH#	BHE#	CCR.2=0	O	Запись старшего байта.В 16-ти битном шинном режиме WRH# устанавливается при записи старшего байтаслова, в 8-битном шинном режиме (CCR.1=0) - при записи старшего и младшего байтов, а также слова.
WRL#	WR#	CCR.2=0	O	Запись младшего байта. В 16-ти битном шинном режиме WRL# устанавливается при записи младшего байта или слова, в 8-битном шинном режиме (CCR.1=1) - при записи старшего и младшего байтов, а также слова.
XTAL1	—	—	I	Вход инвертора внутрикристального генератора и внутреннего генератора тактов. Если используется внешний генератор, XTAL1 также служит как вход тактов МК.
XTAL2	—	—	O	Выход инвертора внутрикристального генератора

Таблица Б.3 описывает функции МК (по умолчанию), значения выходных управляющих контактов и портов ввода-вывода во время и после сброса.

Таблица Б.3 – Состояние вывода после сброса

Название вывода	Мультиплек-сированные выводы портов	Состояние вывода во время сброса	Состояние вывода после сброса
ACH0-ACH	P0.0-P0.7	Входы [1)]	Входы [1)]
PORT1	P1.0-P1.7	Высокоомная нагрузка на #U1	Высокоомная нагрузка на #U1
TXD	P2.0	Низкоомная нагрузка на #U1	Выход
RXD	P2.1	Вход [3)]	Вход [3)]
EXTINT	P2.2	Вход [3)]	Вход [3)]
T2CLK	P2.3	Вход [3)]	Вход [3)]
T2RST	P2.4	Вход [3)]	Вход [3)]
PWM0	P2.5	Среднеомная нагрузка на #U1	Выход
—	P2.6-P2.7	Высокоомная нагрузка на #U1	Высокоомная нагрузка на #U1
AD0-AD15	P3.0-P4.7	Высокоомная нагрузка на #U1	Шина адрес/данные или порт ввода-вывода [2)]
HSI.0, HSI.1	—	Вход [3)]	Вход [3)]
HSI.2/HSO.4	—	Вход [3)]	Вход [3)]
HSI.3/HSO.5	—	Вход [3)]	Вход [3)]

Окончание таблицы Б.3

Название вывода	Мультиплек-сированные выводы портов	Состояние вывода во время сброса	Состояние вывода после сброса
HSO.0-HSO.3	—	Высокоомная нагрузка на #0V	Высокоомная нагрузка на #0V
ALE	—	Высокоомная нагрузка на #U1	Выход
BHE#	—	Высокоомная нагрузка на #U1	Выход
BW	—	Вход [3)]	Вход [3)]
CLKOUT	—	CLKOUT (Третье состояние)	CLKOUT (Третье состояние)
EA#	—	Вход [3)]	Вход [3)]
INST	—	Среднеомная нагрузка на #U1	Третье состояние
NMI	—	Высокоомная нагрузка на #0V	Высокоомная нагрузка на #0V
RD#	—	Высокоомная нагрузка на #U1	Выход
READY	—	Вход [3)]	Вход [3)]
RESET#	—	Среднеомная нагрузка на #U1	Среднеомная нагрузка на #U1
WR#	—	Высокоомная нагрузка на #U1	Выход

[1)] Эти выводы могут быть неподключенными. Однако рекомендуется подключить их к высокому или низкому логическому уровню.

[2)] Состояние этих выводов зависит от конфигурации устройства. Если шина адрес/данные активна, выводы действуют как шинные формирователи, иначе они действуют как порт ввода/вывода с открытым стоком и являются неподключенными (плавающими).

[3)] Эти выводы должны быть подключены и не оставаться плавающими. Входное напряжение не должно превышать уровень U_{CC} в режиме нормальной работы.

Таблица Б.4 – Описания состояния вывода

Состояние вывода	Приближенное значение
Высокоомная нагрузка на #U1	70 мкА
Среднеомная нагрузка на #U1	1 мА
Низкоомная нагрузка на #U1	12 мА
Высокоомная нагрузка на #0V	200 мкА
Среднеомная нагрузка на #0V	1 мА
Высокий уровень	U_{OH}
Низкий уровень	U_{OL}

Примечание – В таблице приведены приблизительные максимальные значения; они необходимы для оценки и не гарантируются.

Приложение В

(рекомендуемое)

Регистры

Данное приложение даёт справочную информацию о регистрах МК.

Таблица В.1 представляет список модулей и основных элементов МК и связанных с ними конфигурационных регистров и регистров статуса.

Таблица В.2 представляет список горизонтальных окон (HWindow) со специальными функциональными регистрами (SFR) и функциями, имеющимися в них.

Таблица В.3 представляет список SFR, расположенных в алфавитном порядке, мнемоник с названиями, адресами, состояниями после сброса.

Таблица В.4 представляет список прерываний МК, расположение векторов для нормальных и PTS прерываний, приоритеты прерываний.

Таблица В.5 представляет список источников прерывания с их векторами прерывания, битами маски, разрешающими прерывания, и биты выбора источника.

Описание регистров, как и в таблицах, приводится в алфавитном порядке.

Таблица В.1 – Модули и связанные с ними регистры

A/D	HSI	HSO	PTS	PWM
AD_COMMAND AD_RESULT AD_TIME IOC2	HSI_TIME HSI_STATUS HSI_TIME IOC0 IOC1	HSO_COMMAND HSO_TIME IOC1 IOC2 IOS0 IOS1 IOS2	PTSBLOCK PTSCON PTSCOUNT PTSDST PTS_REG PTS_S/D PTSSEL PTSSRC PTSSRV	PWM0_CONTROL PWM1_CONTROL PWM2_CONTROL IOC2 IOC3
INTERRUPTS	I/O PORTS	SERIAL PORT	TIMER 1	TIMER 2
INT_MASK INT_MASK1 INT_PEND INT_PEND1	IOPORT0 IOPORT1 IOPORT2 IOPORT34	BAUD_RATE SBUF(RX) SBUF(TX) SP_CON SP_STAT	TIMER1 IOC1 IOS1	TIMER2 T2CAPTURE IOC0 IOC1 IOC2 IOC3
Chip Config	Stack	Watchdog	Window Select	Zero
CCR	SP	WATCHDOG	WSR	ZERO_REG

Таблица В.2 – Регистры специальных функций (SFR) горизонтальных окон (HWindows)

16-ричный код	Чтение/ Запись	HWindow 0	HWindow 1	HWindow 15
1	2	3	4	5
00	Чтение Запись	ZERO_REG (LO) ZERO_REG (LO)	ZERO_REG (LO) ZERO_REG (LO)	ZERO_REG (LO) ZERO_REG (LO)
01	Чтение Запись	ZERO_REG (HI) ZERO_REG (HI)	ZERO_REG (HI) ZERO_REG (HI)	ZERO_REG (HI) ZERO_REG (HI)
02	Чтение Запись	AD_RESULT (LO) AD_COMMAND	Зарезервирован	AD_COMMAND AD_RESULT (LO)
03	Чтение Запись	AD_RESULT (HI) HSI_MODE	AD_TIME AD_TIME	HSI_MODE AD_RESULT (HI)
04	Чтение Запись	HSI_TIME (LO) HSO_TIME (LO)	PTSSEL (LO) PTSSEL (LO)	HSO_TIME (LO) HSI_TIME (LO)

Окончание таблицы В.2

1	2	3	4	5
05	Чтение Запись	HSI_TIME (HI) HSO_TIME (HI)	PTSSEL (HI) PTSSEL (HI)	HSO_TIME (HI) HSI_TIME (HI)
06	Чтение Запись	HSI_STATUS HSO_COMMAND	PTSSRV (LO) PTSSRV (LO)	HSO_COMMAND HSI_STATUS, биты 0, 2, 4, 6
07	Чтение Запись	SBUF (RX) SBUF (TX)	PTSSRV (HI) PTSSRV (HI)	SBUF (TX) SBUF (RX)
08	Чтение Запись	INT_MASK INT_MASK	INT_MASK INT_MASK	INT_MASK INT_MASK
09	Чтение Запись	INT_PEND INT_PEND	INT_PEND INT_PEND	INT_PEND INT_PEND
0A	Чтение Запись	TIMER1 (LO) WATCHDOG	Зарезервирован	WATCHDOG TIMER1 (LO)
0B	Чтение Запись	TIMER1 (HI) IOC2	Зарезервирован	IOC2, кроме бита 7 [1)] TIMER1 (HI)
0C	Чтение Запись	TIMER2 (LO) TIMER2 (LO)	IOC3 IOC3	T2CAPTURE (LO) T2CAPTURE (LO)
0D	Чтение Запись	TIMER2 (HI) TIMER2 (HI)	Зарезервирован	T2CAPTURE (HI) T2CAPTURE (HI)
0E	Чтение Запись	IOPORT0 BAUD_RATE	Зарезервирован	Зарезервирован
0F	Чтение Запись	IOPORT1 IOPORT1	Зарезервирован	Зарезервирован
10	Чтение Запись	IOPORT2 IOPORT2	Зарезервирован	Зарезервирован
11	Чтение Запись	SP_STAT SP_CON	Зарезервирован	INT_PEND1 INT_PEND1
12	Чтение Запись	INT_PEND1 INT_PEND1	INT_PEND1 INT_PEND1	INT_PEND1 INT_PEND1
13	Чтение Запись	INT_MASK1 INT_MASK1	INT_MASK1 INT_MASK1	INT_MASK1 INT_MASK1
14	Чтение Запись	WSR WSR	WSR WSR	WSR WSR
15	Чтение Запись	IOS0 IOC0	Зарезервирован	IOC0, кроме бита 1[1)] IOS0, кроме битов 6 и 7
16	Чтение Запись	IOS1 IOS1	PWM2_CONTROL PWM2_CONTROL	IOC1 IOS1, кроме битов 6 и 7 [2)]
17	Чтение Запись	IOS2 PWM0_CONTROL	PWM1_CONTROL PWM1_CONTROL	PWM0_CONTROL IOS2 [2)]
18	Чтение Запись	SP (LO) SP (LO)	SP (LO) SP (LO)	SP (LO) SP (LO)
19	Чтение Запись	SP (HI) SP (HI)	SP (HI) SP (HI)	SP (HI) SP (HI)

[1)] IOC2.7 (7-ой бит регистра IOC2) и IOC0.1 не снабжены защёлкой и будут читаться как "1".

[2)] Запись в регистры SP_STAT, IOS1 и IOS2 в горизонтальном окне 15 устанавливает биты статуса, но не вызывает прерывания.

Таблица В.3 – Имена SFRs, адреса SFRs и величины SFRs после сброса

Обозначение регистра	Название регистра	16-ричный код	Значение в двоичном коде после сброса
1	2	3	4
AD_COMMAND	Регистр команд АЦП	02	XXXX XXXX
AD_RESULT	Регистр результата АЦП	03, 02	0111 1111 1100 0000
AD_TIME	Регистр времени АЦП	03	1111 1111
BAUD_RATE	Регистр скорости передачи	0E	0000 00X0
HSI_MODE	Регистр режима высокоскоростного ввода	03	1111 1111
HSI_STATUS	Регистр статуса высокоскоростного ввода	06	X0X0 X0X0
HSI_TIME	Регистр времени высокоскоростного ввода	05, 04	XXXX XXXX XXXX XXXX
HSO_COMMAND	Регистр управления высокоскоростным выводом	06	XXXX XXXX
HSO_TIME	Регистр времени высокоскоростного вывода	05, 04	XXXX XXXX XXXX XXXX
INT_MASK	Регистр маски прерывания	08	0000 0000
INT_MASK1	Регистр маски прерывания 1	13	0000 0000
INT_PEND	Регистр хранения прерывания	09	0000 0000
INT_PEND1	Регистр хранения прерывания 1	12	0000 0000
IOC0	0 регистр управления вводом/выводом	15	0000 00X0
IOC1	1 регистр управления вводом/выводом	16	0010 0001
IOC2	2 регистр управления вводом/выводом	0B	X000 0000
IOC3	3 регистр управления вводом/выводом	0C	1111 0000
IOPORT0	Регистр порта 0	0E	XXXX XXXX
IOPORT1	Регистр порта 1	0F	1111 1111
IOPORT2	Регистр порта 2	10	1100 0001
IOPORT34	Регистр портов 3, 4	1FFE	1111 1111 1111 1111
IOS0	Регистр статуса ввода-вывода 0	15	0000 0000
IOS1	Регистр статуса ввода-вывода 1	16	0000 0000
IOS2	Регистр статуса ввода-вывода 2	17	0000 0000
PTSSEL	Регистр выбора PTS	05, 04	0000 0000 0000 0000
PTSSRV	Регистр обслуживания PTS	07, 06	0000 0000 0000 0000
PWM0_CONTROL	Регистр управления PWM0	17	0000 0000
PWM1_CONTROL	Регистр управления PWM1	16	0000 0000
PWM2_CONTROL	Регистр управления PWM2	17	0000 0000
SBUF(RX/TX)	Буфер последовательного порта	07	0000 0000
SP	Указатель стека	19, 18	0001 1101 0001 1000
SP_CON	Регистр управления последовательным портом	11	0000 1011
SP_STAT	Регистр статуса последовательного порта	11	0000 1011
T2CAPTURE	Регистр захвата таймера 2	0D, 0C	XXXX XXXX XXXX XXX
TIMER1	Регистр значения таймера 1	0B, 0A	0000 0000 0000 0000
TIMER2	Регистр значения таймера 2	0D, 0C	0000 0000 0000 0000
WATCHDOG	Регистр сторожевого таймера (WDT)	0A	XXXX XXXX
WSR	Регистр выбора окна	14	XXXX 0000
ZERO_REG	Регистр нуля	01, 00	0000 0000 0000 0000

Таблица В.4 – Источники векторов прерывания, расположение в памяти и приоритеты у МК комплекта

Номер	Название прерывания	Источник(и)	Вектор прерывания	Вектор для PTS	Приоритет [1]
Специальный	Unimplemented Opcode	Неисполняемый код	2012h	–	–
Специальный	Software Trap	Команда TRAP	2010h	–	–
INT15	NMI (2)	NMI	203Eh	–	15
Примечание - Любое из следующих маскируемых прерываний может быть определено с помощью PTS. Любое PTS-прерывание имеет приоритет над всеми другими маскируемыми прерываниями.					
INT14	HSI FIFO Full	Очередь HSI заполнена	203Ch	205Ch	14
INT13	EXTINT1 [2]	P2. 2	203Ah	205Ah	13
INT12	Timer 2 Overflow	Переполнение таймера 2	2038h	2058h	12
INT11	Timer 2 Capture [2]	Захват таймера 2	2036h	2056h	11
INT10	HSI FIFO 4	4-ый вход HSI FIFO	2034h	2054h	10
INT09	Receive	Флаг RI [3]	2032h	2052h	9
INT08	Transmit	Флаг TI [3]	2030h	2050h	8
INT07	EXTINT [2]	P2.2 или P0.7	200Eh	204Eh	7
INT06	Serial Port	Флаг RI или флаг TI [4]	200Ch	204Ch	6
INT05	Software Timer	Программные таймеры 0 – 3 Сброс таймера 2 Начало АЦП	200Ah	204Ah	5
INT04	HSI.0 Pin [2]	HSI.0	2008h	2048h	4
INT03	HSO	HSO.0 – HSO.5	2006h	2046h	3
INT02	HSI Data Available	Очередь HSI заполнена или регистр захвата HSI загружен	2004h	2044h	2
INT01	A/D Conversion Complete	А/Ц преобразование завершено	2002h	2042h	1
INT00	Timer Overflow	Переполнение таймера 1 или таймера 2	2000h	2040h	0

[1] Неисполняемый код и запрос на прерывание от программного обеспечения не имеют приоритета. Они сразу поступают в контроллер прерываний для обслуживания. NMI имеет наивысший приоритет из всех приоритизированных прерываний. Любое PTS-прерывание имеет приоритет перед всеми остальными маскируемыми прерываниями.

[2] Эти прерывания могут считаться независимыми внешними прерываниями.

[3] Если прерывание от Serial Port замаскировано, а прерывания "Receive" и "Transmit" возможны, флаги RI и TI запрашивают отдельно прерывания "Receive" и "Transmit". Если нет совместимости с 8096BH, то эта конфигурация предпочтительнее.

[4] Если прерывания "Receive" и "Transmit" замаскированы и возможно прерывание Serial Port, оба флага RI и TI запрашивают прерывание Serial Port. Эта конфигурация даёт совместимость с 8096BH.

Таблица В.5 – Источники прерывания, векторы и биты регистров

Источник прерывания	Вектор прерывания	Номер	Прерывание, возможное при установке [1)]	Источник выбран путём... [2)]
АЦП завершено	A/D Conversion Complete	INT01	INT_MASK.1	–
Начало АЦП	Software Timer	INT05	INT_MASK.5	–
4-ый вход HSI FIFO	HSI FIFO 4	INT10	INT_MASK1.2	–
HSI FIFO Full	HSI FIFO Full	INT14	INT_MASK1.6	–
	HSI Data Available	INT02	INT_MASK.2	IOC1.7=1
Регистр захвата HSI загружен	HSI Data Available	INT02	INT_MASK.2	IOC1.7=0
Контакт HSI.0	HSI. 0	INT04	INT_MASK.4	–
HSI.0-HSI.5	High-Speed Output	INT03	INT_MASK.3	–
NMI	NMI	INT15	–	–
P0.7	EXTINT	INT07	INT_MASK.7	IOC1.1 = 1
P2.2	EXTINT	INT07	INT_MASK.7	IOC1.1 = 0
	EXTINT1	INT13	INT_MASK1.5	–
Флаг RI	Receive	INT09	INT_MASK1.1	–
	Serial Port	INT06	INT_MASK.6	–
Программные таймеры 0 – 3	Software Timer	INT05	INT_MASK.5	–
Флаг TI	Transmit	INT08	INT_MASK1.0	–
	Serial Port	INT06	INT_MASK.6	–
Переполнение таймера 1	Timer Overflow	INT00	INT_MASK.0	IOC1.2 = 1
Захват таймера 2	Timer 2 Capture	INT11	INT_MASK1.4	–
Переполнение таймера 2	Timer Overflow	INT00	INT_MASK.0	IOC1.3 = 1
	Timer 2 Overflow	INT12	INT_MASK1.4	–
Сброс таймера 2	Software Timer	INT05	INT_MASK.5	–
Команда TRAP	Software TRAP	Специ-альный	–	–
Неисполняемый код	Unimplemented Opcode	Специ-альный	–	–

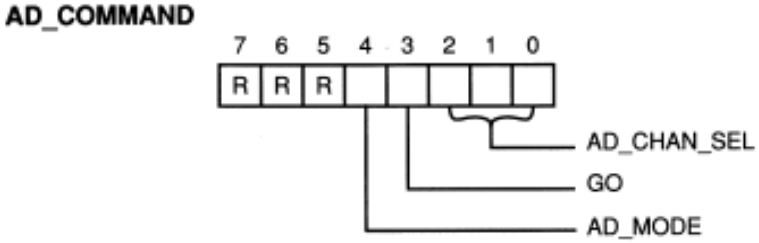
[1)] В этой колонке для каждого источника прерываний приводится список битов маски, которые должны быть установлены, чтобы это прерывание было возможным. Пять источников прерываний могут запрашивать по два различных прерывания – 8096BH-совместимое прерывание и новое, отдельное прерывание (HSI FIFO Full, EXTINT1, Receive, Transmit, Timer 2 Overflow). В любом случае для одного источника возможно только одно прерывание. Биты маски могут устанавливаться только для одного из двух возможных прерываний).

[2)] Три из 8096BH-совместимых прерываний (HSI Data Available, EXTINT и Timer Overflow) могут запрашиваться двумя источниками. Эта колонка показывает значения битов регистра IOC1 для выбора источника прерывания.

Регистр Управления АЦП

AD_COMMAND
02h
Горизонтальное окно 0 (Запись), горизонтальное окно 15 (Чтение)

Регистр управления АЦП выбирает номер канала АЦП, управляет началом АЦП (немедленно или по команде HSO) и выбирает 8-битный или 10-битный режим преобразования.

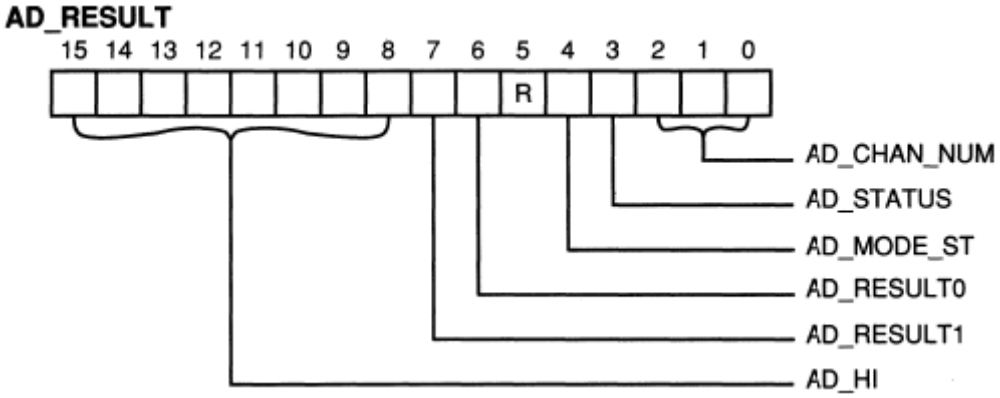


Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0 – 2	AD_CHAN_SEL	Выбор канала ЦАП	XXX	Три бита выбирают номер канала для преобразования: бит 2 1 0 канал 0 0 0 ACH0 0 0 1 ACH1 0 1 0 ACH2 0 1 1 ACH3 1 0 0 ACH4 1 0 1 ACH5 1 1 0 ACH6 1 1 1 ACH7
3	GO	Источник A/D преобразования	X	Этот бит определяет когда начинается A/D преобразование: 1 = немедленно, 0 = по команде HSO
4	AD_MODE	Режим A/D преобразования	X	Этот бит определяет 8-битный или 10-битный режим преобразования: 1 = 8-битный режим, 0 = 10-битный режим
5-7	–	–	XXX	Зарезервированы: всегда записываются нули

Регистр Результата АЦП

AD_RESULT
03/02h
Горизонтальное окно 0 (Чтение), горизонтальное окно 15 (Запись)

Регистр AD_RESULT состоит из двух байтов. Старший байт содержит 8 старших значащих разрядов АЦП. Младший байт показывает номер канала АЦП, который используется в преобразовании, состояние АЦП и 2 младших значащих бита из 10-битного АЦП. Регистр обнуляется в начале нового преобразования, следовательно, чтобы избежать потери данных, эти два байта необходимо считать перед началом нового преобразования.



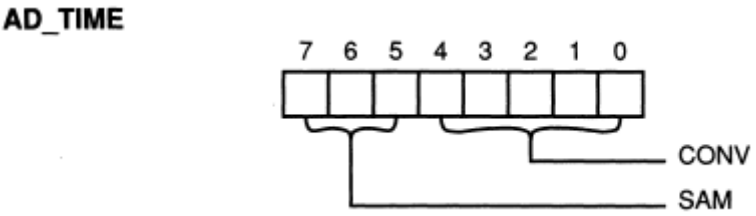
Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-2	AD_CHAN_SUM	Номер канала АЦП	0 0 0	Три бита выбирают номер канала для преобразования
3	AD_STATUS	Состояние АЦП	0	Показывает состояние АЦП. Требуется 8 машинных тактов, чтобы установить этот бит после команды начала преобразования. Для проверки этого бита требуется подождать, по крайней мере, ещё 8 машинных тактов. 1 = АЦП в процессе преобразования, 0 = A/D преобразования нет
4	AD_MODE_ST	Режим АЦП	0	Этот бит определяет 8-битный или 10-битный режим преобразования: 1 = 8-битный режим, 0 = 10-битный режим
5	–	–	0	Зарезервировано: всегда пишутся нули.
6	AD_RESULT0	0 при 10-битном АЦП	1	Бит 0 (LSB) при 10-битном преобразовании
7	AD_RESULT1	0 при 10-битном АЦП	1	Бит 1 при 10-битном преобразовании
8-15	AD_HI	Старшие биты результата АЦП	1111 1110	8 старших значащих бита результата из ЦАП

Регистр времени АЦП

AD_TIME
03h
Горизонтальное окно 1 (Чтение/Запись)

Регистр AD_TIME программирует время фиксации и время преобразования для АЦП. Эти величины используются, когда IOC2.3 установлен. Когда IOC2.3 чист (80C196KB – совместимый режим), IOC2.4 управляет временем фиксации и временем преобразования.

Время фиксации (SAM) – промежуток времени, в течение которого аналоговый входной сигнал связывается с эталонной ёмкостью. Время фиксации должно быть достаточно велико, чтобы эталонная ёмкость полностью зарядилась, но не настолько, чтобы входное напряжение изменилось, что приведёт к ошибке. Время преобразования (CONV) определяет промежуток времени, необходимый для преобразования аналогового напряжения на эталонной ёмкости в цифровую величину. Время преобразования должно быть достаточно велико, чтобы позволить компаратору отследить и оцифровать напряжение, но не настолько, чтобы эталонная емкость разрядилась и точность потерялась.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-4	CONV	Время преобразования АЦП	11111	Эти биты определяют время преобразования. CONV может быть от 2 до 31 включительно
5-7	SAM	Время фиксации АЦП	111	Эти биты определяют время фиксации. SAM может быть от 1 до 7 включительно.

Примечания

1 Регистр программирует скорость, при которой АЦП может работать, а не скорость, при которой он преобразует правильно.

2 Инициализируются А/Ц регистры в следующем порядке: AD_TIME, IOC2, AD_COMMAND.

3 Не следует начинать преобразование с использованием регистра AD_TIME (IOC2. 3=1), если идёт 80C196KB-совместимое преобразование (IOC2. 3=1) и наоборот.

Для вычисления времени фиксации и времени преобразования используют следующие формулы:

$$SAM = ((Tsam * f_{OSC} – 2) / 4) / 2,$$

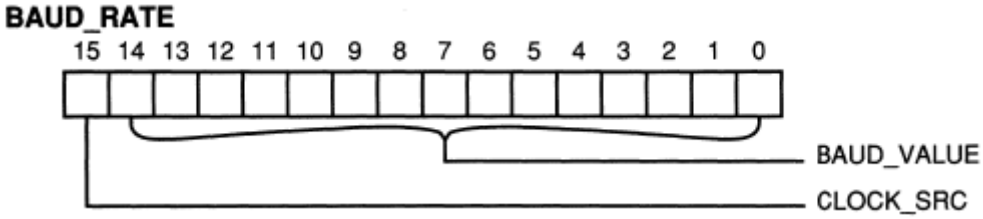
$$CONV = (((Tconv * f_{OSC} – 3) / B) / 2) – 1,$$

где Tsam – время, в мкс
Tconv – время преобразования, в мкс
f_{OSC} – частота XTAL1, в МГц,
B – количество преобразуемых битов (8 или 10).

Регистр скорости передачи

BAUD_RATE
0Eh
Горизонтальное окно 0 (Запись)

Регистр скорости передачи выбирает скорость передачи (в бодах) последовательного порта и источник синхронизации. Он должен быть записан как два байта, сначала младший байт. Старший значащий бит выбирает источник синхронизации (clock source). Младшие 15 битов BAUD_VALUE – незнаковое целое, которое определяет скорость передачи в бодах. BAUD_ VALUE имеет максимальное значение 32767 и может быть равен нулю только при использовании XTAL1 в асинхронных режимах 1, 2 и 3.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-14	BAUD_VALUE	Скорость передачи в бодах	0000 00X0 0000 00X	Эти биты определяют скорость передачи в бодах. Первым загружается младший байт.
15	CLOCK_SRC	Источник синхронизации последовательного порта	0	Этот бит определяет, от какого источника (внутреннего или внешнего) синхронизируется последовательный порт. 1 = XTAL1 (внутренний) 0 = T2CLK (внешний)

Для определения скорости передачи (в бодах) используются следующие формулы.

Синхронный режим 0:
BAUD_VALUE = (F_{OSC} / (Baud Rate * 8)) – 1 или T2CLK / Baud Rate
Асинхронные режимы 1, 2 и 3:
BAUD_VALUE = (F_{OSC} / (Baud Rate * 16)) – 1 или T2CLK/ (Baud Rate * 8),
где F_{OSC} – частота XTAL1, в МГц.

Значения скорости передачи в бодах при использовании XTAL1 на частоте 16 МГц показаны ниже.

Скорость передачи (бод)	Режим 0	Режимы 1, 2, 3
9600	8340h	8067h
4800	8682h	80CFh
2400	8D04h	81A0h
1200	9A0Ah	8340h
300	E82Bh	8D04h

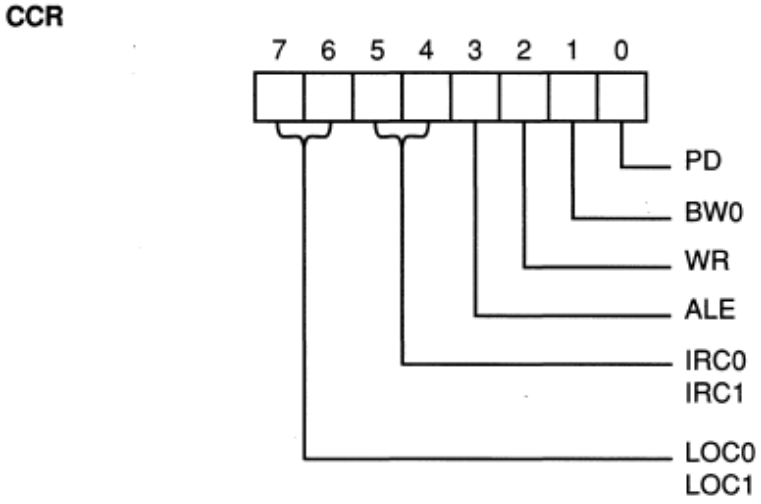
Регистр конфигурации кристалла

CCR

Регистр конфигурации кристалла (CCR) управляет режимами пониженного энергопотребления, разрядностью шин, управляющими сигналами шин, режимом READY и защитой внутренней памяти.

В нормальном рабочем режиме CCR загружается из байта конфигурации кристалла (CCB), расположенного в ячейке 2018h во внутренней или внешней памяти, в зависимости от входа EA#. (Низкий уровень EA# – внешняя память; высокий уровень EA# – внутренняя память.) В режиме программирования CCR загружается из Программирующего Байта Конфигурации Кристалла (PCCB). CCB или PCCB – первый байт, выбираемый из памяти после сброса устройства. CCR загружается только однажды при выполнении последовательности сброса; однажды загруженный CCR не может быть изменён до следующего сброса устройства.

Если контакт READY имеет низкий уровень во время выборки CCR, контроллер шин автоматически выставляет максимум три состояния ожидания в шинном цикле CCR. Это позволяет производить выборку CCR из медленной памяти. CCR.4 и CCR.5 управляют числом состояний ожидания в шинном цикле.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	PD	Разрешение режима POWERDOWN	1	Определяет, ведёт ли команда IDLPD#2 к переходу устройства в режим POWERDOWN. 1 = режим возможен, 0 = режим невозможен
1	BW0	Управление разрядностью шины	1	Выбор: изменяемая или 8-битная разрядность шины. 1 = изменяемая разрядность шины; управляется выводом BW BW = 1, 16-битная шина BW = 0, 8-битная шина 0 = 8-битная разрядность шины; вывод BW игнорируется.

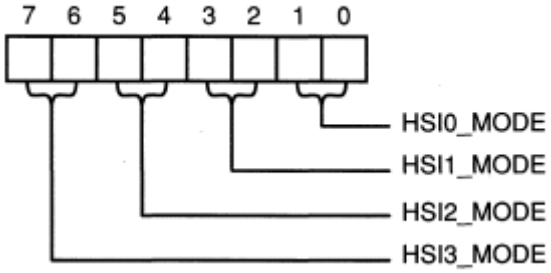
2	WR	Выбор режима Write Strobe	1	Выбор генерируемых стробирующих сигналов записи для 16-битных циклов: 1 = WR# и BHE# генерируются в режимах Standard Bus и Address Valid Strobe. 0 = WRL# и WRH# генерируются в режимах Write Strobe и Address Valid with Strobe Write.
3	ALE	Выбор режима Address Valid Strobe	1	Выбор сигналов, извещающих о фиксации адреса: 1 = ALE фиксирует правильный адрес в режимах Standard Bus и Write Strobe. 0 = ADV# фиксирует адрес вместо ALE и может быть использован в качестве сигнала выбора кристалла внешней памяти.
4-5	IRC0–IRC1	Внутреннее управление готовностью внешних устройств	10	Определяют число состояний ожидания, которые могут быть введены, пока контакт READY поддерживается в низком состоянии. Состояние ожидания поддерживается до тех пор, пока либо сигнал READY не станет высоким, либо не будет достигнуто заданное число состояний ожидания. IRC1 IRC0 Максимальное количество состояний ожидания 0 0 1 0 1 2 1 0 3 1 1 управляется только сигналом READY
6-7	LOC0–LOC1	Биты блокировки	00	Определяют вид программной защиты внутренней памяти. LOC1 LOC0 Вид защиты 0 0 Защита чтения и записи 0 1 Защита только чтения 1 0 Защита только записи 1 1 Нет защиты

Регистр режима HSI

HSI_MODE
03h
Горизонтальное окно 0 (Запись), Горизонтальное окно 15 (Чтение)

Режим высокоскоростного ввода определяет для каждого канала HSI, какое из 4-х типов событий будет фиксироваться в буфере HSI FIFO: каждый передний фронт, каждый задний фронт, любой фронт (передний или задний) или серия из восьми передних фронтов. Каналы должны индивидуально получить разрешающие сигналы через регистр IOC0.

HSI_MODE



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-1	HSI0_MODE	Режим HSI.0	11	Режим работы нулевого канала HSI.0
2-3	HSI1_MODE	Режим HSI.1	11	Режим работы первого канала HSI.1
4-5	HSI2_MODE	Режим HSI.2	11	Режим работы второго канала HSI.2
6-7	HSI3_MODE	Режим HSI.3	11	Режим работы третьего канала HSI.3

Каждое двубитное поле определяет режим работы для соответствующего канала:

Кодирование режимов работы для модуля HSI

Бит 1	Бит 0	Описание
0	0	В буфере HSI FIFO фиксируется каждый восьмой передний фронт
0	1	В буфере HSI FIFO фиксируется каждый передний фронт
1	0	В буфере HSI FIFO фиксируется каждый задний фронт
1	1	В буфере HSI FIFO фиксируется любой фронт (передний и задний)

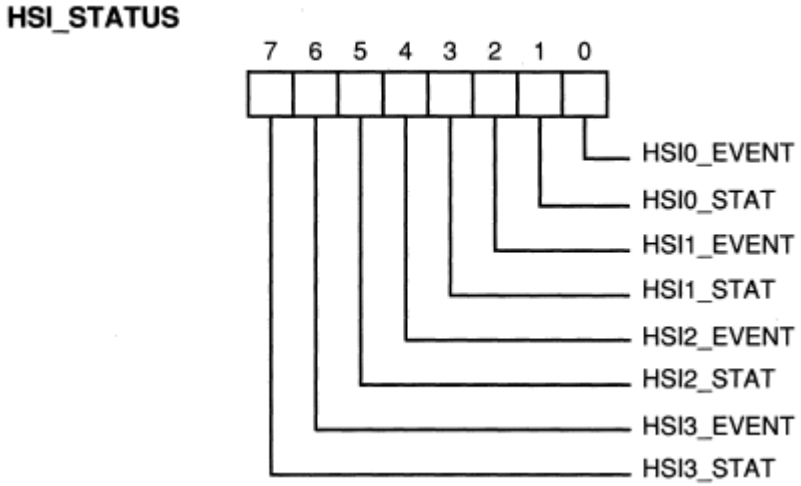
Регистр состояния HSI

HSI_STATUS
06h
Горизонтальное окно 0 (Чтение), Горизонтальное окно 15 (Запись битов 0, 2, 4, 6)

Регистр HSI_STATUS показывает наличие событий в HSI и текущие состояния каналов. Регистр HSI_TIME содержит временную метку. Чтение регистра HSI_ TIME перезагружает данный регистр. Если HSI_TIME считывается перед HSI_STATUS, то информация о состоянии, ассоциируемая с временной меткой регистра HSI_TIME, теряется.

Если данный HSI регистр не содержит событий, то биты состояния событий регистра не определены, однако биты состояния каналов могут быть считаны в любое время. Это позволяет читать каналы HSI как входы даже тогда, когда они не используются в HSI-устройстве. Запись в HSI_STATUS в горизонтальном окне 15 устанавливает биты состояния событий, но не влияет на состояние выводов. Заметим, что текущее состояние вывода не обязательно соответствует статусному биту события, так как другие события могут произойти со времени последнего записанного состояния каналов.

Регистр IOC0 управляет альтернативными функциями HSI.0 – HSI.3, а регистр HSI_ MODE управляет событиями каналов, которые будут захватываться системой HSI в очередь FIFO.



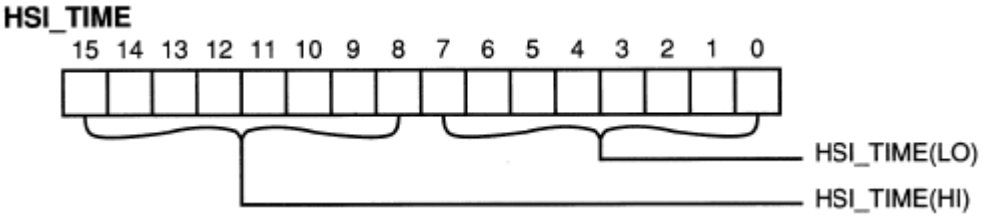
Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	HSI0_EVENT	Событие канала HSI.0	0	1 = событие в канале HSI.0 имеет место, 0 = событий нет
1	HSI0_STAT	Состояние канала HSI.0	X	Текущее состояние канала HSI.0
2	HSI1_EVENT	Событие канала HSI.1	0	1 = событие в канале HSI.1 имеет место, 0 = событий нет
3	HSI1_STAT	Состояние канала HSI.1	X	Текущее состояние канала HSI.1
4	HSI2_EVENT	Событие канала HSI.2	0	1 = событие в канале HSI.2 имеет место, 0 = событий нет
5	HSI2_STAT	Состояние канала HSI.2	X	Текущее состояние канала HSI.2
6	HSI3_EVENT	Событие канала HSI.3	0	1 = событие в канале HSI.3 имеет место, 0 = событий нет
7	HSI3_STAT	Состояние канала HSI.3	X	Текущее состояние канала HSI.3

Регистр времени HSI

HSI_TIME
04, 05h
Горизонтальное окно 0 (Чтение), Горизонтальное окно 15 (Запись)

Регистр HSI-TIME содержит время, в которое происходит HSI-событие. Когда событие в канале HSI имеет место, регистр HSI_TIME загружается текущим 16-битным значением таймера 1. Время хранится в буфере HSI FIFO вместе с четырьмя битами состояния событий из регистра HSI_STATUS.

Чтение регистра HSI_TIME перегружает регистр HSI_STATUS. Если вы считываете HSI_TIME перед HSI_STATUS, информация, связанная с временной меткой в регистре HSI_TIME, теряется. Если регистр HSI_STATUS не содержит событий, HSI_TIME не определён. Запись в HSI_TIME в горизонтальном окне 15 перезагружает регистр хранения, перезаписывая другие данные.

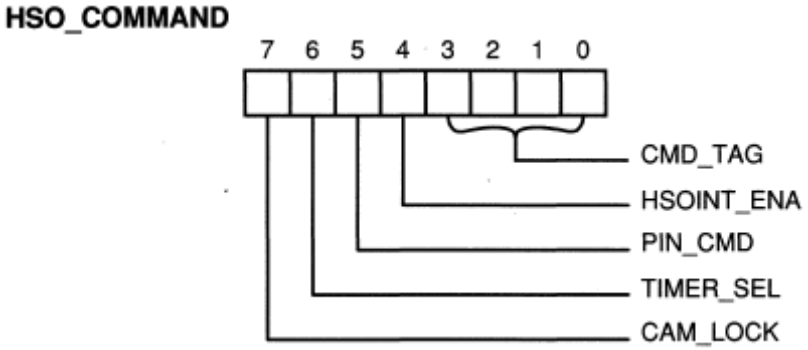


Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-7	HSI_TIME (LO)	Время событий HSI	XXXX XXXX	Младшие 8 битов времени событий высокоскоростного ввода.
8-15	HSI_TIME (HI)	Время событий HSI	XXXX XXXX	Старшие 8 битов времени событий высокоскоростного ввода.

Регистр управления HSO

HSO_COMMAND
06h
Горизонтальное окно 0 (Запись), Горизонтальное окно 15 (Чтение)

Модуль высокоскоростного вывода (HSO) может генерировать события в определённое время с минимальными затратами CPU. Регистр управления HSO определяет, какое событие (события) будет иметь в модуле HSO время, определённое регистром HSO_TIME.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-3	CMD_TAG	Команды HSO	XXXX	Определяет, какое событие (события) будет (будут) происходить во время, заданное в HSO_TIME.
4	HSOINT_ENA	Разрешение/ Запрет HSO прерывания	X	Определяет, будет ли событие HSO генерировать прерывание. 1 = генерация прерывания 0 = нет прерывания Когда этот бит установлен, программируемый вывод генерирует прерывание HSO (INT03, 2006h), а внутренние события генерируют прерывания Software Timer (INT05, 200Ah). После генерации прерывания биты состояния событий HSO в регистрах IOS1 и IOS2 должны быть опрошены для определения, какое событие вызвало прерывание.

5	PIN_CMD	Установка/ Сброс выбранного вывода HSO	X	Определяет, будет ли команда сбрасывать определённый вывод или выводы. 1 = установка вывода (выводов) 0 = сброс вывода (выводов)
6	TIMER_SEL	Выбор таймера 1 / таймера 2	X	Выбирает таймер для команд HSO. 1 = выбор таймера 1 0 = выбор таймера 2
7	CAM_LOCK	Захват команды в CAM	X	Когда IOC2.6 установлен (возможна захват команд), этот бит определяет, будет ли команда HSO оставаться в CAM или сбрасываться после выполнения. 1 = захват команды в CAM 0 = сброс команды из CAM после выполнения Запись "1" в IOC2.7 сбрасывает все команды из CAM, как и сброс кристалла

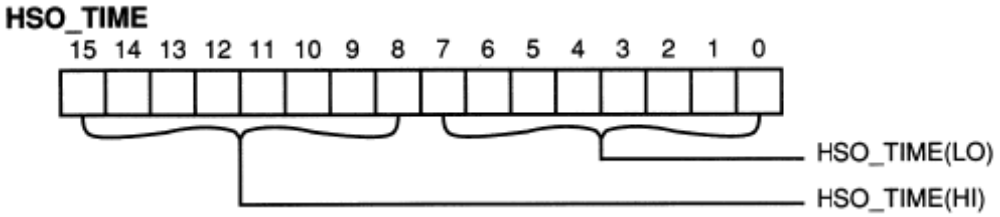
Кодирование CMD_TAG

Бит3	Бит 2	Бит 1	Бит 0	Обозначение команды	Определение
0	0	0	0	HSO0	Переключение на HSO.0
0	0	0	1	HSO1	Переключение на HSO.1
0	0	1	0	HSO2	Переключение на HSO.2
0	0	1	1	HSO3	Переключение на HSO.3
0	1	0	0	HSO4	Переключение на HSO.4
0	1	0	1	HSO5	Переключение на HSO.5
0	1	1	0	HSO01 *	Переключение на HSO.0 и HSO.1
0	1	1	1	HSO23 *	Переключение на HSO.2 и HSO.3
1	0	0	0	SWT0	Программирование таймера 0
1	0	0	1	SWT1	Программирование таймера 1
1	0	1	0	SWT2	Программирование таймера 2
1	0	1	1	SWT3	Программирование таймера 3
1	1	0	0	HSOALL *	Переключение на HSO.0 – HSO.5
1	1	0	1	–	Зарезервировано; не используется
1	1	1	0	T2RST	Сброс таймера 2
1	1	1	1	A_D	Начало аналого-цифрового преобразования
* В этих комбинациях битов два или более выводов устанавливаются или сбрасываются одновременно					

Регистр времени HSO

HSO_TIME
05/04h
Горизонтальное окно 0 (Запись), горизонтальное окно 15 (Чтение)

Регистр времени HSO определяет время выполнения команды HSO. Команды определяются битами HSO_COMMAND. 0 – HSO_COMMAND. 3, а таймер выбирается битом HSO_COMMAND. 6. Когда события загружаются в CAM, сначала записывайте в HSO_COMMAND, затем в HSO_TIME.

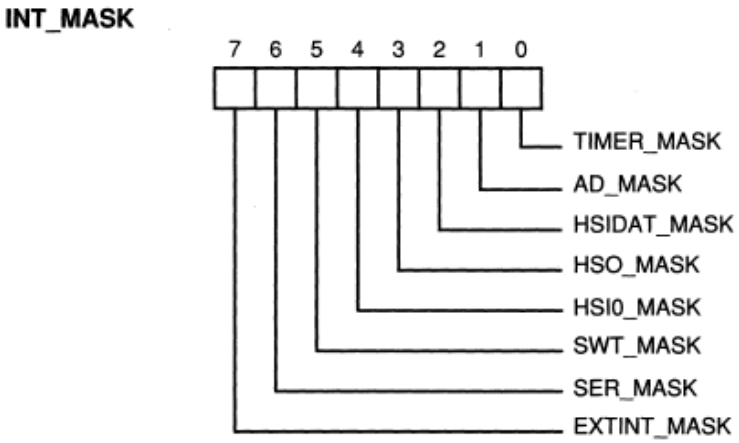


Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-7	HSO_TIME (LO)	Время выполнения команд HSO, младший байт	XXXX	Это младший байт регистра время выполнения команд HSO.
8-15	HSO_TIME (HI)	Время выполнения команд HSO, старший байт	XXXX	Это старший байт регистра время выполнения команд HSO.

Регистр масок прерываний

INT_MASK
08h
Все горизонтальные окна (Чтение/Запись)

Регистр INT_MASK разрешает или запрещает (маскирует) отдельные прерывания. (PSW.2 глобально разрешает или запрещает обслуживание всех маскируемых прерываний.) Из INT_MASK можно читать и записывать как в байтовый регистр во всех горизонтальных окнах. INT_MASK – младший байт PSW (Слова Состояния Программы); следовательно, PUSHF или PUSHA записывают содержимое этого регистра в стек, а POPF или POPA восстанавливают его.



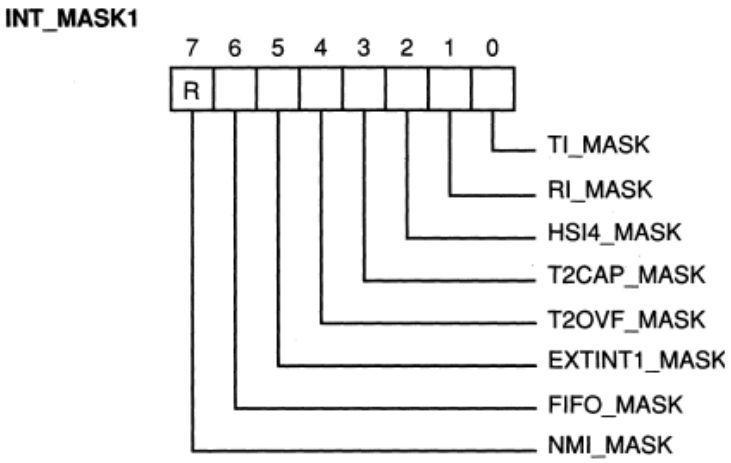
Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	TIMER_MASK	Timer 1 или Timer 2 Overflow	0	Установка этого бита разрешает прерывание Timer Overflow (INT00, 2000h). Оба таймер 1 и таймер 2 могут генерировать прерывание INT00. Установка IOC1.2 выбирает в качестве источника таймер 1; а установка IOC1.3 – таймер 2. Таймер 2 может генерировать или INT00 или INT12, но не оба одновременно.
1	AD_MASK	A/D Conversion Complete	0	Установка этого бита разрешает прерывание по завершению АЦП (INT01, 2002h).

2	HSIDAT_MASK	HSI Data Available / FIFO Full	0	Установка этого бита разрешает прерывание HSI Data Available (INT02, 2004h). IOC1.7 выбирает источник прерывания.
3	HSO_MASK	HSO Output Event	0	Установка этого бита разрешает прерывание HSO (INT03, 2004h).
4	HSI0_MASK	HSI. 0 внешнее прерывание	0	Установка этого бита разрешает прерывание от внешнего контакта HSI.0 (INT04, 2008h).
5	SWT_MASK	Software Timer	0	Установка этого бита разрешает прерывание Software Timer (INT05, 200Ah).
6	SER_MASK	Serial Port	0	Установка этого бита разрешает прерывание Serial Port (INT06, 200Ch), которое совместимо с 8096BH. Если этот бит установлен, INTMASK1.0 и INTMASK1.1 должны быть очищены для запрета прерываний Receive и Transmit.
7	EXTINT_MASK	Прерывание от EXTINT или P0.7	0	Установка этого бита разрешает прерывание EXTINT (INT07, 200Eh). IOC1.1 выбирает источник прерывания (P0.7, P2.2).

Регистр масок прерываний 1

INT_MASK1
13h
Все горизонтальные окна (Чтение/Запись)

INT_MASK1 разрешает или запрещает (маскирует) индивидуальные прерывания (PSW.2 глобально разрешает или запрещает обслуживание всех маскируемых прерываний). INT_MASK1 можно читать и записывать во всех горизонтальных окнах. PUSHA записывает содержимое этого регистра в стек, а POPA – восстанавливает его.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	TI_MASK	Прерывание Transmit	0	Установка этого бита разрешает прерывание Transmit (INT08, 2030h). Если этот бит установлен, INT_MASK1.1 также должен быть установлен, а INT_MASK.6 должен быть очищен.
1	RI_MASK	Прерывание Receive	0	Установка этого бита разрешает прерывание Receive (INT09, 2032h). Если этот бит установлен, INT_MASK1.0 также должен быть установлен, а INT_MASK.6 должен быть очищен.

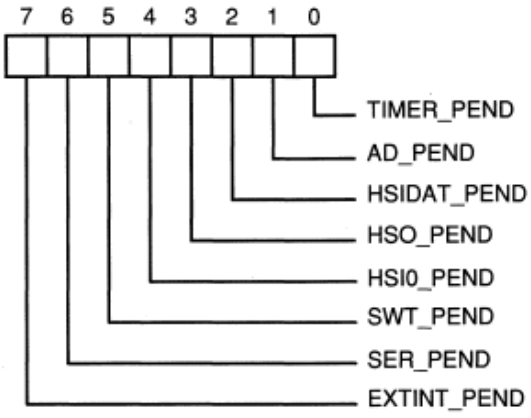
2	HSI4_MASK	Прерывание HSO FIFO 4	0	Установка этого бита разрешает прерывание HSI FIFO 4 (INT10, 2034h).
3	T2CAP_MASK	Прерывание Timer 2 Capture	0	Установка этого бита разрешает прерывание Timer 2 Capture (INT11, 2036h).
4	T2OVF_MASK	Прерывание Timer 2 Overflow	0	Установка этого бита разрешает прерывание Timer 2 Overflow (INT12, 2038h). IOC2.5 выбирает предел переполнения для INT12.
5	EXTINT1_MASK	Прерывание от контакта EXTINT1	0	Установка этого бита разрешает прерывание EXTINT1 (INT13, 203Ah). P2. 2 может генерировать или прерывание EXTINT (INT07) или прерывание EXTINT1 (INT13).
6	FIFO_MASK	Прерывание HSI FIFO Full	0	Установка этого бита разрешает прерывание HSI FIFO Full (INT14, 203Ch).
7	NMI_MASK	NMI	0	Этот нефункционирующий бит маски существует условно. Немаскируемое прерывание (NMI) разрешено и при 0, и при 1. В этот бит всегда записывают ноль.

Регистр ожидания прерываний

INT_PEND
09h
Все горизонтальные окна (Чтение/Запись)

Когда аппаратное обеспечение обнаруживает запрос на прерывание, оно устанавливает соответствующий бит в INT_PEND или INT_PEND1. Когда процессор перешел на обработку прерывания по соответствующему вектору, аппаратное обеспечение очищает бит запроса. Регистр INT_PEND можно читать или записывать во всех горизонтальных окнах. Программное обеспечение может генерировать прерывание установкой соответствующего бита запроса прерывания.

INT_PEND



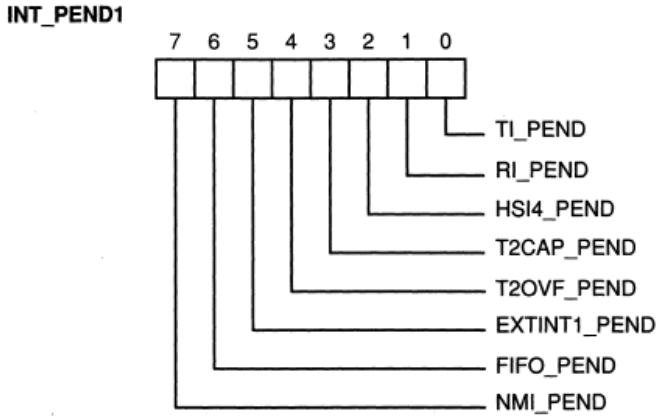
Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	TIMER_PEND	Timer 1 или Timer 2 Overflow	0	Когда этот бит установлен, он показывает на запрос прерывания Timer Overflow (INT00). Он очищается, когда произошел переход по вектору с адресом 2000h.
1	AD_PEND	A/D Conversion Complete	0	Когда этот бит установлен, он показывает на запрос прерывания A/D Conversion Complete (INT01). Он очищается, когда произошел переход по вектору с адресом 2002h.

2	HSIDAT_PEND	HSI Data Available / FIFO Full	0	Когда этот бит установлен, он показывает на запрос прерывания HSI Data Available (INT02). Он очищается, когда произошел переход по вектору с адресом 2004h.
3	HSO_PEND	HSO Output Event	0	Когда этот бит установлен, он показывает на запрос прерывания HSO (INT03). Он очищается, когда произошел переход по вектору с адресом 2006h
4	HSI0_PEND	HSI.0 внешнее прерывание	0	Когда этот бит установлен, он показывает на запрос прерывания HSI.0 (INT04). Он очищается, когда произошел переход по вектору с адресом 2008h.
5	SWT_PEND	Software Timer	0	Когда этот бит установлен, он показывает на запрос прерывания Software Timer (INT05). Он очищается, когда произошел переход по вектору с адресом 200Ah.
6	SER_PEND	Serial Port	0	Когда этот бит установлен, он показывает на запрос прерывания Serial Port (INT06). Он очищается, когда произошел переход по вектору с адресом 200Ch.
7	EXTINT_PEND	Прерывание от EXTINT или P0.7	0	Когда этот бит установлен, он показывает на запрос прерывания EXTINT (INT07). Он очищается, когда произошел переход по вектору с адресом 200Eh.

Регистр ожидания прерываний 1

INT_PEND1
12h
Все горизонтальные окна (Чтение/Запись)

Когда аппаратное обеспечение обнаруживает запрос на прерывание, оно устанавливает соответствующий бит в INT_PEND или INT_PEND1. Когда процессор перешел на обработку прерывания по соответствующему вектору, аппаратное обеспечение очищает бит запроса. Регистр INT_PEND1 можно читать или записывать во всех горизонтальных окнах. Программное обеспечение может генерировать прерывание установкой соответствующего бита запроса прерывания.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	TI_PEND	Прерывание Transmit	0	Когда этот бит установлен, он показывает на запрос прерывания Transmit (INT08). Он очищается, когда произошел переход по вектору с адресом 2030h.
1	RI_PEND	Прерывание Receive	0	Когда этот бит установлен, он показывает на запрос прерывания Receive (INT09). Он очищается, когда произошел переход по вектору с адресом 2032h.

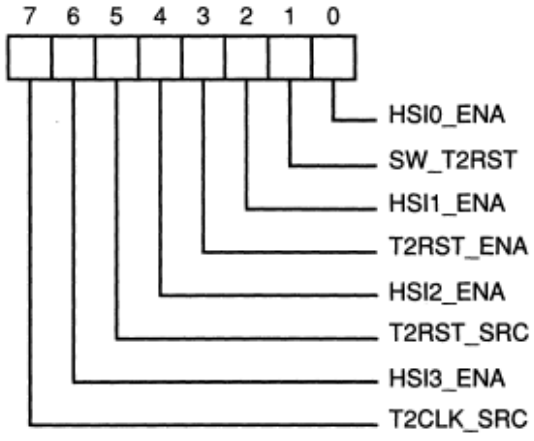
2	HSI4_PEND	Прерывание HSO FIFO 4	0	Когда этот бит установлен, он показывает на запрос прерывания HSO FIFO 4 (INT10). Он очищается, когда произошел переход по вектору с адресом 2034h.
3	T2CAP_PEND	Прерывание Timer 2 Capture	0	Когда этот бит установлен, он показывает на запрос прерывания Timer 2 Capture (INT11). Он очищается, когда произошел переход по вектору с адресом 2036h
4	T2OVF_PEND	Прерывание Timer 2 Overflow	0	Когда этот бит установлен, он показывает на запрос прерывания Timer 2 Overflow (INT12). Он очищается, когда произошел переход по вектору с адресом 2038h.
5	EXTINT1_PEND	Прерывание от контакта EXTINT1	0	Когда этот бит установлен, он показывает на запрос прерывания EXTINT1 (INT13). Он очищается, когда произошел переход по вектору с адресом 203Ah.
6	FIFO_PEND	Прерывание HSI FIFO Full	0	Когда этот бит установлен, он показывает на запрос прерывания HSI FIFO Full (INT14). Он очищается, когда произошел переход по вектору с адресом 203Ch.
7	NMI_PEND	NMI	0	Когда этот бит установлен, он показывает на запрос прерывания NMI (INT15). Он очищается, когда произошел переход по вектору с адресом 203Eh.

Регистр 0 управления вводом/выводом

IOC0
15h
Горизонтальное окно 0 (Запись), горизонтальное окно 15 (Чтение)

Регистр IOC0 выбирает внешний генератор тактовых импульсов и источники перезагрузки таймера 2 и разрешает или запрещает функции HSI для четырех выводов HSI. Когда IOC0 считывается из горизонтального окна 15, IOC.1 всегда читается как "1", потому что его значение не буферизуется.

IOC0



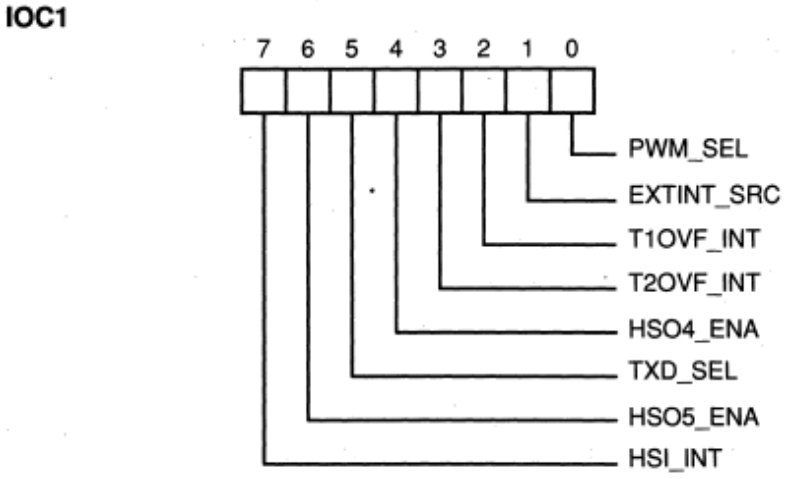
Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	HSI0_ENA	Разрешение HSI.0 как входа HSI	0	Этот бит определяет, будут ли состояния на выводе HSI.0 фиксироваться в буфере HSI FIFO 1 = разрешение HSI; 0 = запрещение HSI

1	SW_T2RST	Программный сброс таймера 2	X	Запись "1" в этот бит сбрасывает таймер 2. Этот бит всегда читается как "1" в горизонтальном окне 15.1 = сброс таймера 2 после каждой записи 0 = нет действия
2	HSI1_ENA	Разрешение HSI.1 как входа HSI	0	Этот бит определяет, будут ли состояния на выводе HSI.1 фиксироваться в буфере HSI FIFO 1 = разрешение HSI; 0 = запрещение HSI
3	T2RST_ENA	Источник внешнего сброса таймера 2	0	Этот бит разрешает внешний сброс таймера 2. Источник внешнего сброса – нарастающий фронт либо T2RST, либо HSI.0, как выбрано в IOC0.5. 1 = разрешение внешнего сброса 0 = запрещение
4	HSI2_ENA	Разрешение HSI.2 как входа HSI	0	Этот бит определяет, будут ли состояния на выводе HSI.2 фиксироваться в буфере HSI FIFO 1 = разрешение HSI; 0 = запрещение HSI
5	T2RST_SRC	Источник сброса таймера 2	0	Этот бит выбирает источник внешнего сброса таймера 2. IOC0.3 должен быть установлен для разрешения внешнего сброса. 1 = нарастающий фронт на выводе HSI.0 0 = нарастающий фронт на выводе (P2.4) T2RST
6	HSI3_ENA	Разрешение HSI.3 как входа HSI	0	Этот бит определяет, будут ли состояния на выводе HSI. 3 фиксироваться в буфере HSI FIFO 1 = разрешение HSI; 0 = запрещение HSI
7	T2CLK_SRC	Входной тактовый сигнал таймера 2	0	Этот бит выбирает внешний источник тактовых импульсов таймера 2. IOC3.0 должен быть очищен для разрешения внешнего источника тактовых импульсов. 1 = вывод HSI.1; 0 = вывод T2CLK (P2.3)

Регистр 1 управления вводом/выводом

IOC1
16h
Горизонтальное окно 0 (Запись), горизонтальное окно 15 (Чтение)

Регистр IOC1 выбирает выходные функции P2.5 и P2.0; разрешает или запрещает вывод с HSO.4 или HSO.5; выбирает источники прерываний для EXTINT (INT07, 200Eh), переполнения таймера (INT00, 2000h) и доступности HSI данных (INT02, 2004h).



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	PWM_SEL	Выбор вывода P2.5/ PWM	1	Этот бит определяет, будет ли P2.5 функционировать как PWM выход или как стандартный вывод порта. 1 = выход PWM 0 = стандартный вывод порта
1	EXTINT_SRC	Выбор источника внешнего прерывания INT07	0	Этот бит выбирает источник внешнего прерывания EXTINT (INT07, 200Eh) Для разрешения быть установлен. 1 = P0.7 0 = P2.2
2	T1OVF_INT	Разрешение прерывания Timer 1 Overflow	0	И таймер 1 и таймер 2 могут генерировать прерывание Timer Overflow (INT00, 2000h). Этот бит определяет, будет ли переполнение таймера 1 генерировать прерывание. Для разрешения прерывания IOS1.5 показывает статус прерывания. 1 = источник прерывания таймер 1 0 = таймер 1 – не является источником
3	T2OVF_INT	Разрешение прерывания Timer 2 Overflow	0	И таймер 1 и таймер 2 могут генерировать прерывание Timer Overflow (INT00, 2000h). Этот бит определяет, будет ли переполнение таймера 2 генерировать прерывание. Для разрешения прерывания IOS1.4 показывает статус прерывания. 1 = источник прерывания таймер 2 0 = таймер 2 – не является источником
4	HSO4_ENA	Разрешение работы выхода HSO.4	0	HSO.4 мультиплексируется с выводом HSI.2. Этот бит разрешает работу HSO.4. 1 = разрешает вывод 0 = запрещает вывод
5	TXD_SEL	Выбор вывода P2.0/ TXD	1	Этот бит определяет, будет ли P2.0 функционировать как выход последовательного порта передатчика или как стандартный вывод порта. 1 = выход последовательного порта TXD 0 = стандартный вывод порта
6	HSO5_ENA	Разрешение работы выхода HSO.5	0	HSO.5 мультиплексируется с выводом HSI.3. Этот бит разрешает работу HSO.5. 1 = разрешает выход 0 = запрещает выход
7	HSI_INT	Выбор источника прерывания HSI	0	Этот бит определяет источник прерывания HSI Data Available (INT02, 2004h). Должен быть установлен для разрешения INT_MASK.2. 1 = очередь HSI FIFO заполнена 0 = регистр захвата HSI загружен

Регистр 2 управления вводом/выводом

IOC2

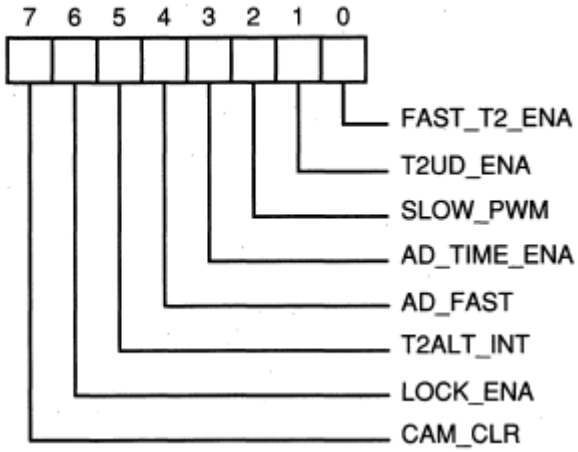
0Bh

Горизонтальное окно 0 (Запись), горизонтальное окно 15 (Чтение)

Регистр IOC2 управляет тремя параметрами таймера 2, предварительным делением частоты для PWM и АЦП и источником времени для аналого-цифрового преобразования.

IOC2 также разрешает или запрещает захват команды в HSO CAM и может очистить всё содержимое HSO CAM.

IOC2



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	FAST_T2_ENA	Разрешение быстрого инкремента таймера 2	0	Этот бит определяет, будет ли таймер 2 работать в нормальном режиме или в режиме быстрого инкремента. 1 = режим быстрого инкремента; считает каждый такт 0 = нормальный режим; считает каждые 8 тактов Когда разрешён режим быстрого инкремента, не используйте таймер 2 в качестве базового при HSO и не сбрасывайте таймер 2.
1	T2UD_ENA	Разрешение режима прямого/ обратного счёта для таймера 2	0	Этот бит определяет, будет ли таймер 2 функционировать как счётчик прямого счёта или как счётчик прямого и обратного счёта. 1 = если P2.6 = 0 прямой счёт; если P2.6 = 1 обратный счёт, 0 = только прямой счёт
2	SLOW_PWM	Разрешение предваритель ного делителя частоты синхронизаци и для управления выходом PWM	0	Этот бит управляет периодом на выходе PWM путём разрешения или запрещения предварительного делителя частоты синхронизации (деление на 2) для PWM.1, PWM.2 и PWM.3. 1 = разрешение, период на выходе PWM = 512 машинных тактов. 0 = запрещение, период на выходе PWM = 256 машинных тактов.
3	AD_TIME_ENA	Разрешение регистра	0	Этот бит выбирает, будет ли время АЦП управляться регистром и нормальным режимами 1874ВЕ36. 0 = 1874ВЕ36-совместимый режим Когда этот бит обнулён, IOC2.4 разрешает или запрещает предварительный делитель частоты аналого-цифровой синхронизации (АЦС) для полной совместимости с 1874ВЕ36.

4	AD_FAST	Разрешение предваритель ного деления частоты АЦС	0	В 1874BE36-совместимом режиме (IOC2.3= 0) этот бит управляет длительностью цикла АЦП путём разрешения или запрещения предварительного деления частоты АЦС на 2. 1 = запрещение; время АЦП = 89,5 машинных тактов, быстрый режим 1874BE36 0 = разрешение; время АЦП = 156,5 машинных тактов, нормальный режим 1874BE36 Если IOC2.3 = 1, то этот бит игнорируется.
5	T2ALT_INT	Выбор границы переполнения таймера2	0	Этот бит выбирает границу переполнения для прерывания по переполнению таймера 2 (INT12, 2038H). равен 1 для разрешения прерывания. 1 = граница 7FFFh/8000h. 0 = граница 0FFFFh/0000h.
6	LOCK_ENA	Разрешение фиксирования команд в CAM файле	0	Этот бит разрешает и запрещает захват команд. Когда этот бит управляет тем, будут ли отдельные команды фиксироваться в CAM файле или стираться после выполнения. 1 = разрешение фиксации команд. 0 = запрещение фиксации команд. Запись 1 в IOC2.7 очищает содержимое CAM файла во всех случаях так же, как и при сбросе кристалла.
7	CAM_CLR	Очистка всего содержимого CAM файла	X	Установка этого бита очищает всё содержимое (даже фиксированные команды) из HSO CAM файла. Этот бит не запоминается; он всегда читается как 1 в горизонтальном окне 15.

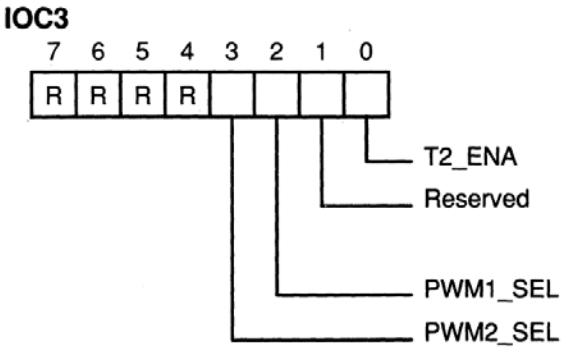
Регистр 3 управления вводом/выводом

IOC3

0Ch

Горизонтальное окно 1 (Чтение/Запись)

Регистр IOC.3 выбирает внутренний или внешний источник синхронизации для таймера 2 и выбирает функции входов P1.2 и P1.3.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	T2_ENA	Разрешение внутреннего источника синхронизаци и таймера 2	0	Этот бит управляет, будет ли таймер 2 синхронизироваться внутренне или внешне. 1 = внутренний источник; 0 = внешний источник; если IOC0.7 = 1, то HSI.1; если IOC0.7 = 0, то T2CLK (P2.3). IOC2.0 определяет, будет ли таймер 2 считать каждый такт (быстрый инкрементный режим) или каждые 8 тактов (нормальный режим).

1	Reserved	Зарезервиро-ван	0	Этот бит зарезервирован, всегда записывают 0.
2	PWM1_SEL	Выбор PWM1	0	Этот бит выбирает функцию P1.3. 1 = вывод PWM1; 0 = вывод квазидвунаправленного порта. Когда этот бит установлен, P1.3 имеет низкоомную нагрузку на VCC или на GND. Функции этого вывода могут переключаться без сброса устройства.
3	PWM2_SEL	Выбор PWM2	0	Этот бит выбирает функцию P1.4. 1 = вывод PWM2 0 = вывод квазидвунаправленного порта. Когда этот бит установлен, P1.4 имеет низкоомную нагрузку на U_{CC1} или на 0V. Функции этого вывода могут переключаться без сброса устройства.
4-7	–	–	1111	Зарезервированы; всегда записывайте нули.

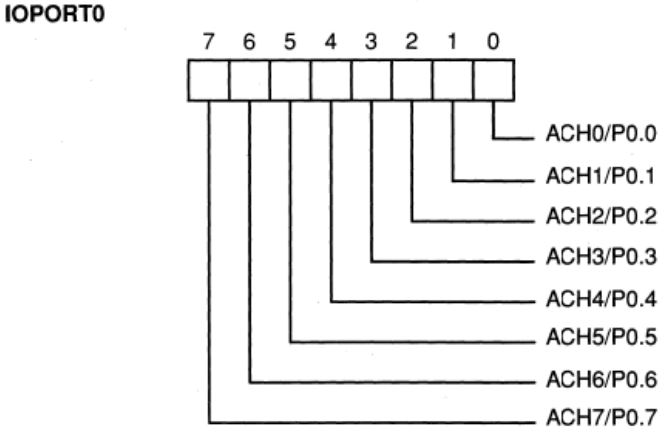
Регистр ввода-вывода порта 0

IOPORT0

0Eh

Горизонтальное окно 0 (Чтение)

Порт 0 – только входной порт. Для уменьшения помех значение на входе порта 0 фиксируется, только когда происходит чтение. Выводы могут использоваться для аналогового ввода в АЦП (ACHx) и цифрового ввода (P0.x) одновременно, но это не рекомендуется. Вывод P0.7 может также использоваться как вход внешнего прерывания.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	ACH0/P0.0	ACH0/ P0.0	X	АЦП канал 0 / вход P0.0
1	ACH1/P0.1	ACH1/ P0.1	X	АЦП канал 1 / вход P0.1
2	ACH2/P0.2	ACH2/ P0.2	X	АЦП канал 2 / вход P0.2
3	ACH3/P0.3	ACH3/ P0.3	X	АЦП канал 3 / вход P0.3
4	ACH4/P0.4	ACH4/ P0.4	X	АЦП канал 4 / вход P0.4
5	ACH5/P0.5	ACH5/ P0.5	X	АЦП канал 5 / вход P0.5
6	ACH6/P0.6	ACH6/ P0.6	X	АЦП канал 6 / вход P0.6
7	ACH7/P0.7	ACH7/ P0.7	X	АЦП канал 7 / вход P0.7

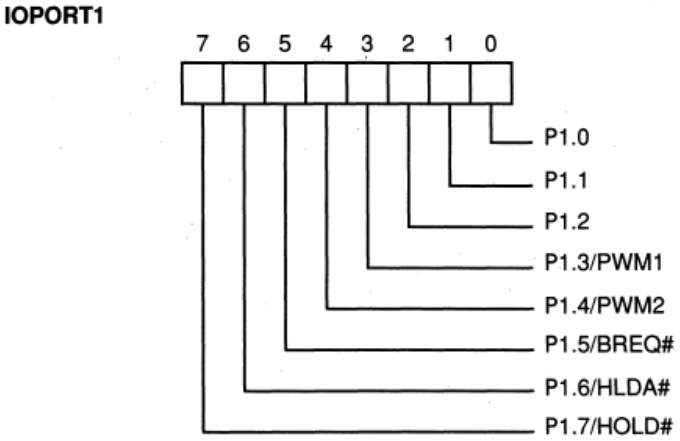
Регистр ввода/вывода порта 1

IOPORT1

0Fh

Горизонтальное окно 0 (Чтение/Запись)

Если не выбраны дополнительные функции, то все выводы порта 1 – квазидвунаправленные. Три вывода порта разделяют функции с сигналами захвата шины; два вывода порта разделяют функции совместно с выводами PWM. Когда выводы порта сконфигурированы на ввод-вывод, то они могут быть считаны или записаны. Когда выводы порта настроены на дополнительные функции, то их можно только считать.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	P1.0	P1.0	1	Контакт ввода-вывода P1.0
1	P1.1	P1.1	1	Контакт ввода-вывода P1.1
2	P1.2	P1.2	1	Контакт ввода-вывода P1.2
3	P1.3/PWM1	P1.3/ PWM1	1	Контакт ввода-вывода P1.3/ выход PWM1 Установка IOC3.2 разрешает использовать P1.3 в качестве выхода PWM1.
4	P1.4/PWM2	P1.4/ PWM2	1	Контакт ввода-вывода P1.4/ выход PWM2. Установка IOC3.3 разрешает использовать P1.4 в качестве выхода PWM2.
5	P1.5/BREQ#	P1.5/ BREQ#	1	Контакт ввода-вывода P1.5/ Запрос шины. Установка WSR.7 разрешает использовать P1.5 как BREQ#. Когда разрешён протокол HOLD, вывод функционирует как BREQ# до сброса устройства. BREQ# становиться активным выходом, когда контроллер шины ожидает цикл обращения к внешней памяти. Когда BREQ# установлен, он остаётся активным до снятия сигнала HOLD#.
6	P1.6/HLDA#	P1.6/ HLDA#	1	Контакт ввода-вывода P1.6/ Подтверждение захвата. Установка WSR.7 разрешает использовать P1.6 как HLDA#. Когда разрешён протокол HOLD, вывод функционирует как HLDA# до сброса устройства. HLDA# становиться активным выходом, когда МК освобождает шину в ответ на установленный другим устройством сигнал HOLD#.
7	P1.7/HOLD#	P1.7/ HOLD#	1	Контакт ввода-вывода P1.6/ Запрос захвата. Установка WSR.7 разрешает использовать P1.7 как HOLD#. Когда разрешён протокол HOLD, вывод функционирует как HOLD# до сброса устройства. HOLD# становится активным входом, когда другое устройство запрашивает управление шиной.

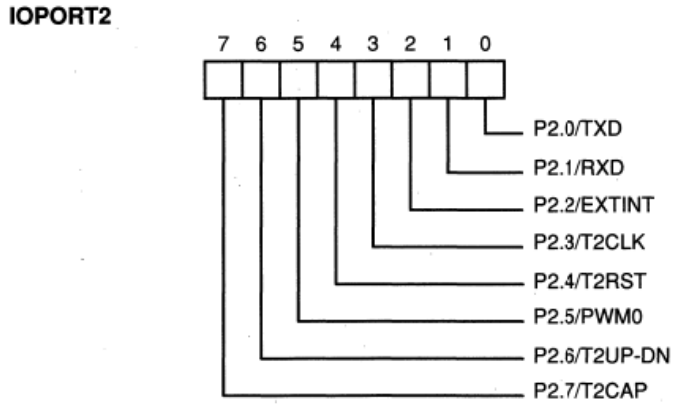
Регистр ввода-вывода порта 2

IOPORT2

10h

Горизонтальное окно 0 (Чтение/Запись)

Порт 2 содержит контакты только для ввода, контакты только для вывода и квазидвунаправленные контакты. Дополнительные функции должны быть разрешены в соответствующих управляющих регистрах.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	P2.0/TXD	P2.0 / TXD	1	Выходной контакт P2.0. Установка IOC1.5 разрешает использовать контакт как TXD, который служит выходом последовательного порта в режимах 1, 2 и 3 и контактом синхронизации для сдвигового регистра в режиме 0.
1	P2.1/RXD	P2.1 / RXD	0	Входной контакт P2.1 пользовать контакт как RXD. В режимах 1, 2 и 3 RXD используется для приёма данных в последовательный порт. В режиме 0 он служит для ввода или вывода данных (во время вывода контакт работает с открытым стоком).
2	P2.2/EXTINT	P2.2 / EXTINT	0	Входной контакт P2.2. Очистка IOC1.1 выбирает P2.2 в качестве источника прерывания EXTINT (INT07, 200Eh). P2.2 может генерировать прерывание или EXTINT (INT07) или EXTINT1 (INT13, 203Ah).
3	P2.3/T2CLK	P2.3 / T2CLK	0	Входной контакт P2.3. Очистка IOC0.7 разрешает использовать контакт в качестве внешней входной синхронизации для таймера 2.
4	P2.4/T2RST	P2.4/ T2RST	0	Входной контакт P2.4. Очистка IOC2.5 разрешает использовать контакт для внешнего сброса таймера 2. IOC0.3 должен быть установлен для разрешения внешнего сброса.
5	P2.5/PWM0	P2.5/ PWM0	0	Выходной контакт P2.5. Установка IOC1.0 разрешает использовать этот контакт в качестве выхода PWM0.

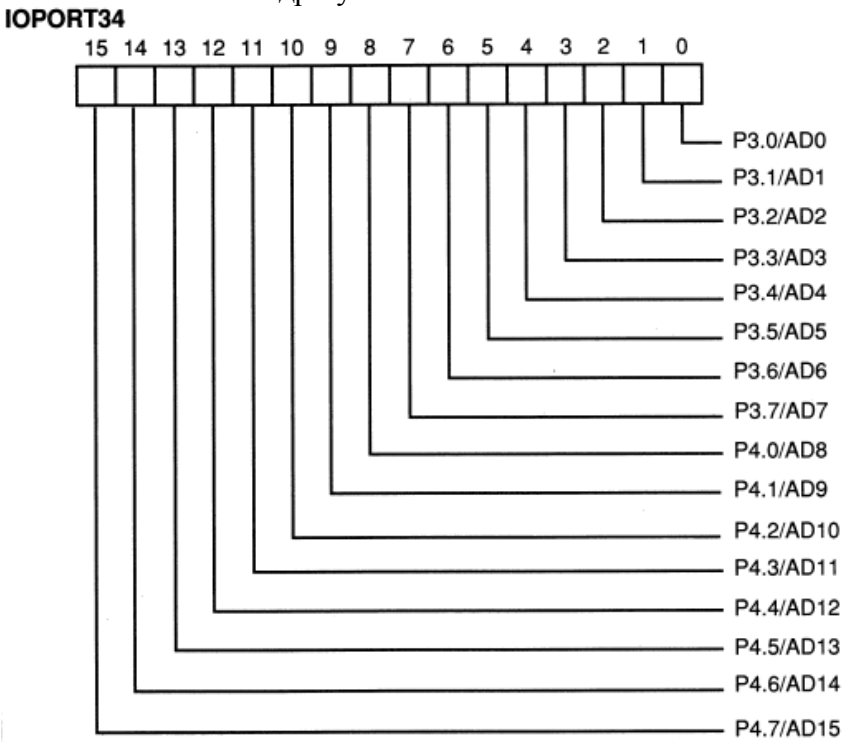
6	P2.6/T2UP-DN	P2.6/ T2UP-DN	1	Квазидвунаправленный контакт P2.6. Установка IOC2.1 разрешает использовать контакт для непосредственного управления направлением счета таймера 2. Таймер 2 считает на увеличение, когда на контакте 0, и на уменьшение, когда – 1.
7	P2.7/T2CAP	P2.7/ T2CAP	1	Квазидвунаправленный контакт P2.7. Нарастающий фронт на контакте P2.7 записывает значение таймера 2 в регистр T2CAPTURE и генерирует прерывание Timer 2 Capture (INT11, 2036h).

Регистр ввода/вывода портов 3 и 4

IOPORT34

1FFEh

Порты 3 и 4 – двунаправленные с открытым стоком. Дополнительная функция данных портов – мультиплексированная шина адрес/данные. Порты автоматически выполняют функции системной шины, когда на EA# низкий уровень, и функционируют как порт с открытым стоком, когда на EA# высокий уровень. Порты 3 и 4 могут быть считаны или записаны только как слово по адресу 1FFEh.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-7	P3.0 – P3.7/ AD0 – AD7	P3.0 – P3.7/ AD0 – AD7	1111 1111	Двунаправленный порт ввода-вывода P3.0 –P3.7, когда EA#=1; в остальное время: AD0 – AD7.
8-15	P4.0 – P4.7/ AD8 – AD15	P4.0 – P4.7/ AD8 – AD15	1111 1111	Двунаправленный порт ввода- вывода P4.0 –P4.7, когда EA#=1; в остальное время: AD8 – AD15.

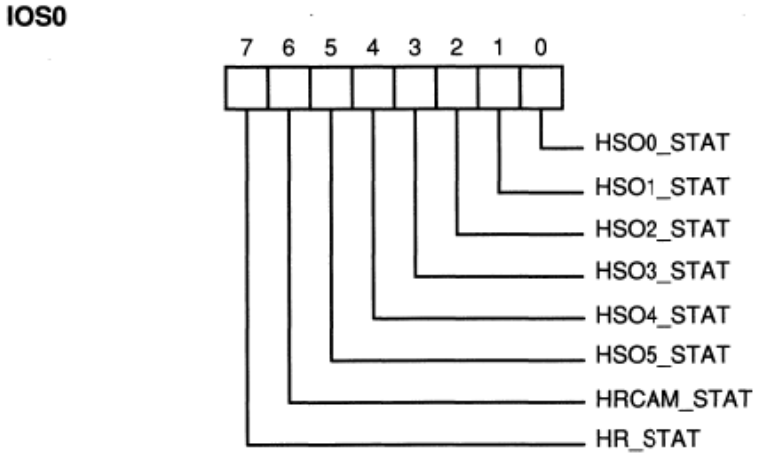
Регистр 0 состояния ввода-вывода

IOS0

15h

Горизонтальное окно 0 (Чтение), горизонтальное окно 15 (Запись)

Регистр IOS0 показывает текущее состояние контактов HSO. Запись в соответствующие биты (IOS0.0 – IOS0.5) в HWindow 15 может устанавливать или сбрасывать контакты HSO. IOS0.6 и IOS0.7 показывают текущее состояние HSO CAM файла и фиксирующего регистра HSO. (IOS0.6 и IOS0.7 нельзя записывать.)



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	HSO.0_STAT	Состояние HSO.0	0	Текущее состояние контакта HSO.0
1	HSO.1_STAT	Состояние HSO.1	0	Текущее состояние контакта HSO.1
2	HSO.2_STAT	Состояние HSO.2	0	Текущее состояние контакта HSO.2
3	HSO.3_STAT	Состояние HSO.3	0	Текущее состояние контакта HSO.3
4	HSO.4_STAT	Состояние HSO.4	0	Текущее состояние контакта HSO.4. HSO.4 – вывод двунаправленного порта, мультиплексированный с контактом HSI.2. Установка IOC1.4 разрешает работу выхода HSO.4
5	HSO.5_STAT	Состояние HSO.5	0	Текущее состояние канала HSO.5. HSO.5 – вывод двунаправленного порта, мультиплексированный с контактом HSI.3. Установка IOC1.6 разрешает работу выхода HSO.5
6	HRCAM_STAT	Состояние HSO CAM и фиксирующего регистра	0	Текущее состояние фиксирующего регистра HSO и CAM. 0 = регистр захвата HSO пуст и по крайней мере одно слово CAM пусто. 1 = регистр захвата HSO загружен. Во избежание перезаписи текущего значения не следует записывать в регистр захвата, пока не сбросится этот бит или бит IOS0.7.
7	HR_STAT	Состояние фиксирующего регистра HSO	0	Текущее состояние фиксирующего регистра HSO. 0 = регистр захвата HSO пуст. 1 = регистр захвата HSO загружен. Во избежание перезаписи текущего значения не записывайте в регистр захвата, пока не сбросится этот бит или бит IOS0. 6.

Регистр 1 состояния ввода-вывода

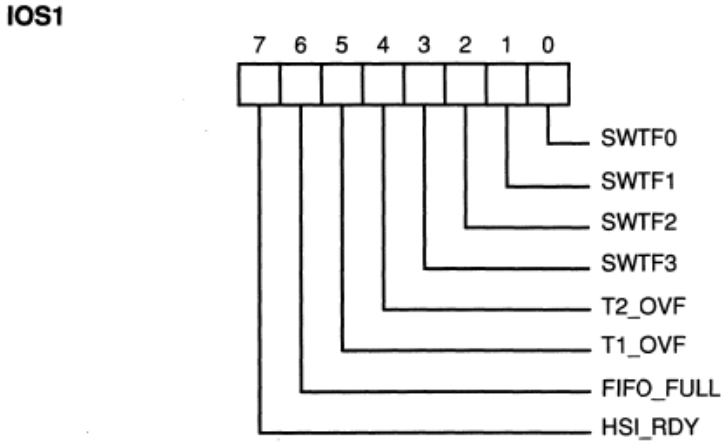
IOS1

16h

Горизонтальное окно 0 (Чтение), горизонтальное окно 15 (Запись)

Регистр IOS1 содержит флаги, которые показывают, какие события установили запрос на прерывание. IOS1.0 – IOS1.5 показывают состояния программных таймеров: таймера 1 и таймера 2. Чтение IOS1 сбрасывает биты IOS1.0 – IOS1.5. Запись в IOS1.0 – IOS1.5 изменяет эти биты, но не вызывает прерывания.

IOS1.6 и IOS1.7 показывают состояние очереди HSI FIFO и буферного регистра HSI. В IOS1.6 и IOS1.7 нельзя записывать.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	SWTF0	Флаг программного таймера 0	0	Этот бит, когда он установлен, показывает, что программный таймер 0 сработал и вызывает прерывание INT05 (200Ah). INT_MASK.5 должен быть установлен для разрешения прерывания.
1	SWTF1	Флаг программного Таймера1	0	Этот бит, когда он установлен, показывает, что программный таймер 1 сработал и вызывает прерывание INT05 (200Ah). INT_MASK.5 должен быть установлен для разрешения прерывания.
2	SWTF2	Флаг программного таймера 2	0	Этот бит, когда он установлен, показывает, что программный таймер 2 сработал и вызывает прерывание INT05 (200Ah). INT_MASK.5 должен быть установлен для разрешения прерывания.
3	SWTF3	Флаг программного таймера 3	0	Этот бит, когда он установлен, показывает, что программный таймер 3 сработал и вызывает прерывание INT05 (200Ah). INT_MASK.5 должен быть установлен для разрешения прерывания.
4	T2_OVF	Флаг переполнения таймера 2	0	И таймер 1, и таймер 2 могут генерировать прерывание Timer Overflow (INT00, 2000h). Этот бит, когда установлен, показывает, что таймер 2 вызвал прерывание. INT_MASK.0 должен быть установлен для разрешения прерывания.

5	T1_OVF	Флаг переполнения таймера 1	0	И таймер 1, и таймер 2 могут генерировать прерывание переполнения таймера (INT00, 2000h). Этот бит, когда установлен, показывает, что таймер 1 вызвал прерывание. INT_MASK.0 должен быть установлен для разрешения прерывания.
6	FIFO_FULL	Флаг шестой записи в FIFO	0	Этот бит, когда установлен, показывает, что буфер HSI FIFO имеет 6 или более записей, без учета буферного регистра. Это событие может генерировать либо прерывание HSI Data Available (INT02, 2004h), либо прерывание HSI FIFO Full (INT14, 203Ch), но не оба сразу.
7	HSI_RDY	Готовность данных в буферном регистре HSI	0	Этот бит, когда установлен, показывает, что буферный регистр HSI загружен. Когда IOC1.7 сброшен, это событие генерирует прерывание HSI Data Available (INT02, 2004h). INT_MASK.2 должен быть установлен для разрешения прерывания.

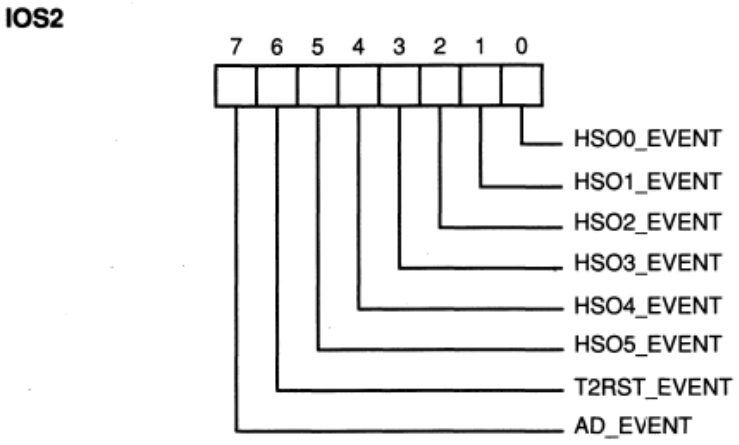
Регистр 2 состояния ввода-вывода

IOS2

17h

Горизонтальное окно 0 (Чтение), горизонтальное окно 15 (Запись)

Регистр IOS2 содержит флаги, которые показывают, какие события HSO имели место. Запись в IOS2 устанавливает или сбрасывает биты состояния, но не вызывает прерывание. Чтение IOS2 очищает все биты.



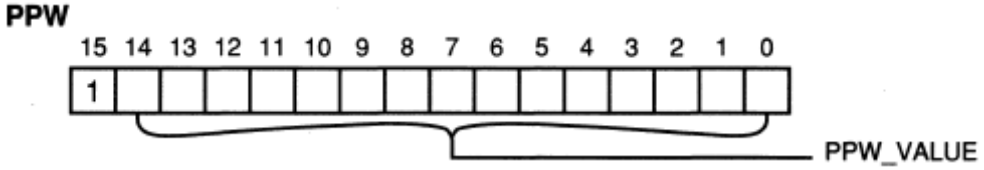
Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	HSO0_EVENT	Событие на выводе HSO.0	0	Этот бит, когда он установлен, показывает, что команда HSO выполнена на контакте HSO.0.
1	HSO1_EVENT	Событие на выводе HSO.1	0	Этот бит, когда он установлен, показывает, что команда HSO выполнена на контакте HSO.1.
2	HSO2_EVENT	Событие на выводе HSO.2	0	Этот бит, когда он установлен, показывает, что команда HSO выполнена на контакте HSO.2.
3	HSO3_EVENT	Событие на выводе HSO.3	0	Этот бит, когда он установлен, показывает, что команда HSO выполнена на контакте HSO.3

4	HSO4_EVENT	Событие на выводе HSO.4	0	Этот бит, когда он установлен, показывает, что команда HSO выполнена на контакте HSO.4.
5	HSO5_EVENT	Событие на выводе HSO.5	0	Этот бит, когда он установлен, показывает, что команда HSO выполнена на контакте HSO.5.
6	T2RST_EVENT	Событие сброс таймера 2	0	Этот бит, когда он установлен, показывает, что команда HSO сбросила таймер 2.
7	AD_EVENT	Событие начала A/D преобразования	0	Этот бит, когда он установлен, показывает, что команда HSO запустила A/D преобразование.

Регистр длительности программирующих импульсов PPW
(нет прямого доступа)

Регистр PPW доступен только в автоматическом режиме программирования. Он загружается из внешнего адреса 4015/4014h.

Значение регистра PPW определяет длительность программирующих импульсов. Длительность программирующих импульсов должна быть не менее 100 мкс для правильного функционирования программ. Старший значащий бит (PPW.15) должен быть всегда установлен в 1. Старший байт PPW обычно содержит 80h, а младший – 05H, но они могут содержать и другие значения.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-14	PPW_VALUE	Длительность программирующего импульса	0000 0101 0000 000	Эти биты обычно равны 05h, для XTAL1 = 8 МГц. Записывают нужное значение PPW_VALUE в эти биты.
15	PPW_MSB	Старший значащий бит PPW	1	Этот бит должен быть всегда равным 1.

Используйте следующую формулу и округление результата до ближайшего целого значения для определения соответствующего значения PPW_VALUE:

$PPW\_VALUE = (0{,}6944 * F_{OSC}) - 1$

где:

PPW_VALUE - 15-битное слово

F_{OSC} - частота XTAL1, в МГц

Например, пусть XTAL1 = 8 МГц:

PPW_VALUE = (0,6944 * 8) – 1
= 5,5552 – 1
= 4,4552
= 5

Слово состояния программы PSW
(нет прямого доступа)

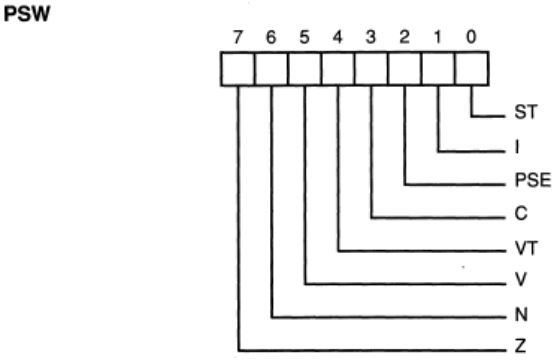
PSW в действительности состоит из двух байтов. Старший байт – слово состояния, которое описано здесь; младший байт – регистр INT_MASK. Слово состояния содержит 1 бит (PSW.1), который разрешает или запрещает обработку всех маскируемых прерываний; один

бит (PSW.2), который разрешает или запрещает Программный Сервер Обмена (PTS) и 6 двоичных флагов, которые отражают состояние программы пользователя.

Слово состояния PSW прямо не доступно. Для доступа к слову состояния следует записать его значение в стек (PUSHF), а затем считать это значение в регистр (POP test_reg). Команды PUSHF и PUSHA сохраняют PSW в системном стеке, POPF и POPA: восстанавливают его.

Команды EI и DI устанавливают и сбрасывают PSW.1; команды EPTS и DPTS устанавливают и сбрасывают PSW.2. Различные команды тестируют, устанавливают и сбрасывают двоичные флаги.

В приложении Б приведена таблица, которая показывает воздействие команд на флаги PSW, и таблица, которая показывает воздействие флагов PSW на команды условного перехода.)



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	ST	Флаг Sticky		Этот флаг устанавливается для того, чтобы показать, что во время сдвига вправо, "1" была сдвинута во флаг переноса и затем сдвинута из него. Этот бит не определён после операции умножения. Флаг ST может использоваться вместе с флагом переноса для более точного округления. (Подробности смотрите в описании флага переноса.)
1	I	Запрещение прерываний (глобальное)		Этот бит разрешает или запрещает обслуживание всех маскируемых прерываний (это все прерывания, кроме NMI, TRAP и неподдерживаемого кода MASK1 индивидуально разрешают или запрещают прерывания. Команда EI устанавливает этот бит; DI- очищает его. 1 = разрешение обработки прерываний, 0 = запрещение обработки прерываний
2	PSE	Разрешение PTS		Этот бит глобально разрешает или запрещает Периферийный Сервер Обмена (PTS). Команда EPTS устанавливает этот бит; DPTS – сбрасывает его. 1 = разрешение PTS, 0 = запрещение PTS

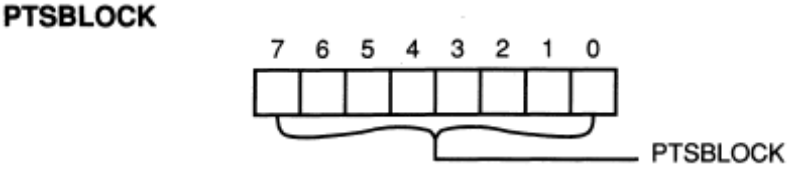
3	C	Флаг переноса		Этот флаг устанавливается, чтобы показывать состояние арифметического переноса из старшего значащего бита ALU или состояние последнего бита, сдвинутого из операнда. Если вычитание содержит заём, флаг переноса устанавливается. C Значение сдвинутых битов 0 < 1/2 LSB 1 >= 1/2 LSB Обычно результат округляется до большего значения при установленном флаге переноса. Флаг Sticky бита позволяет улучшить точность округления. C ST Значение сдвинутых битов 0 0 = 0 0 1 > 0 и < 1/2 LSB 1 0 = 1/2 LSB 1 1 > 1/2 LSB и < 1 LSB
4	VT	Дополнительный флаг переполнения		Этот флаг устанавливается, когда флаг переноса установлен, но он очищается только командами CLRVT, JVT, JNVT. Это позволяет тестировать возможное состояние переполнения в конце последовательности родственных арифметических операций, которое обычно более эффективно, чем тестирование флага переполнения после любой операции.
5	V	Флаг переполнения		Этот флаг устанавливается, если после выполнения операции получен результат, выходящий из диапазона значений типа данных приёмника. При сдвиговых операциях (SHL, SHLB, SHLL) этот флаг устанавливается, если старший значащий бит операнда изменился в процессе сдвига. При операциях деления этот флаг устанавливается в следующих случаях: Команда Результат DIVUB > 155 (0FFh) DIVU > 65535 (0FFFFh) DIVB < -127 (81h) или > +127 (7Fh) DIV < -32767 (8001h) или > +32767 (7FFFh)
6	N	Флаг отрицательного результата		Этот флаг устанавливается, если после выполнения операции получен отрицательный результат. Флаг - правильный, если даже случилось переполнение. При всех сдвиговых операциях и операции NORML этот флаг устанавливается равным старшему значащему биту результата, даже если счётчик сдвигов равен нулю.
7	Z	Флаг нуля		Этот флаг устанавливается, если результат операции равен нулю. При операциях сложения с переносом и вычитания с заёмом этот флаг никогда не устанавливается, но он сбрасывается, если результат не нулевой. Таким образом, флаг нулевого результата показывает наличие нулевого или ненулевого результата при вычислениях с большой точностью.

Регистр блока PTS

PTSBLOCK

Смещение 7 в управляющем блоке PTS

Регистр PTSBLOCK используется во время блочной передачи, в режимах HSI и HSO. PTSBLOCK управляет числом передач.



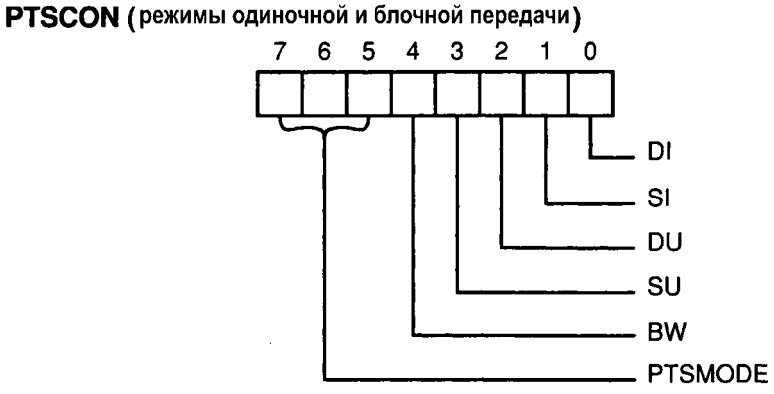
Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-7	PTSBLOCK	Блочный счётчик PTS	XXXX XXXX	Управление числом передач, которые будут иметь место. Значение зависит от режима PTS. Режим Значение Передача блоков 1-32 HSI 1-7 HSO 1-8 Во всех режимах запись нуля в этот регистр ведёт к максимальному числу передач (32 – для режима блочной передачи; 7 – для режима HSI; 8 – для режима HSO).

Регистр управления PTS

PTSCON

Смещение 1 в Управляющем Блоке PTS

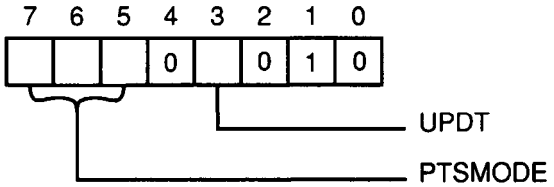
Три бита регистра PTSCON определяют режим PTS: одиночная передача, блочная передача, A/D, HSO или HSI. Режим PTS определяется пятью битами. PTSCON имеет одну конфигурацию для режимов одиночной и блочной передачи и другую для режимов A/D, HSO и HSI. Конфигурации описываются отдельно. (Бит PSW.2, управляемый командами DPTS и EPTS, разрешает или запрещает PTS.)



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	DI	Автоинк-ремент PTSDST	0	Установка этого бита ведёт к инкременту регистра – приёмника обмена в конце каждого цикла обмена.
1	SI	Автоинк-ремент PTSSRC	0	Установка этого бита ведёт к инкременту регистра – источника обмена в конце каждого цикла обмена.

2	DU	Обновле-ние PTSDST	0	Установка этого бита ведёт к сохранению регистром PTSDST его окончательного значения в конце PTS цикла. Его очистка ведёт к возвращению регистру значения, которое существовало в начале PTS цикла.
3	SU	Обновле-ние PTSSRC	0	Установка этого бита ведёт к сохранению регистром PTSSRC его окончательного значения в конце PTS цикла. Его очистка ведёт к возвращению регистру значения, которое существовало в начале PTS цикла.
4	BW	Передача байта/ слова	0	Установкой этого бита выбирается байтовая передача. Очисткой выбирается передача слова.
5-7	PTSMODE	Режим обмена	000	Эти три бита определяют режим обмена: Бит 7 6 5 0 0 0 одиночная передача 1 0 0 блочная передача

PTSCON (режимы A/D, HSO и HSI)



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	–	–	0	Всегда 0.
1	–	–	1	Всегда 1.
2	–	–	0	Всегда 0.
3	UPDT	Обновление регистра	0	Загрузка этого бита ведёт к загрузке соответствующего регистра значением, которое существует в конце обмена. Его очистка ведёт к загрузке этого регистра значением, существовавшим в начале цикла обмена. Режим Регистр A/D PTS_S/D HSI PTSDST HSO PTSSRC
4	–	–	0	Всегда 0.
5-7	PTSMODE	Режим обмена PTS	000	Эти биты определяют режим работы PTS: Бит 7 6 5 1 1 0 A/D 0 1 1 HSO 0 0 1 HSI

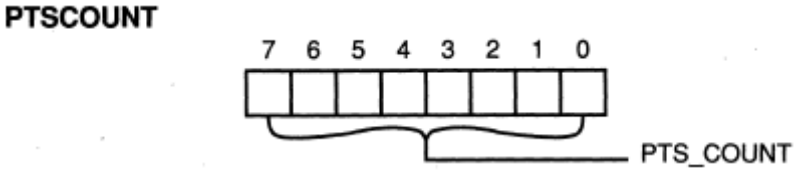
Счётный регистр PTS

PTSCOUNT

Смещение 0 в Управляющем Блоке PTS

Регистр PTSCOUNT используется во всех режимах обмена. PTSCOUNT определяет количество циклов обмена, которые будут последовательно осуществляться без вмешательства CPU. Так как PTSCOUNT – 8-битная величина, то максимальное число циклов – 256. PTSCOUNT декрементируется в конце каждого цикла обмена. Когда PTSCOUNT достигает нуля, аппаратное обеспечение сбрасывает соответствующий бит PTSSEL и устанавливает бит PTSSRV, который запрашивает

прерывание end-of-PTS. Когда это прерывание вызвано, аппаратное обеспечение сбрасывает бит PTSSRV. Бит PTSSEL может устанавливаться вручную для восстановления канала обмена.



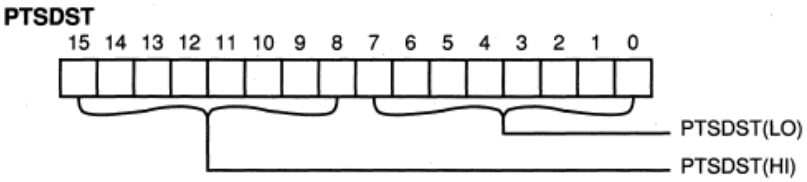
Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-7	PTS_COUNT	Последовательные циклы обмена	XXXX XXXX	Определяют количество циклов обмена, которые будут последовательно осуществляться без вмешательства CPU. Максимальное значение равно 256.

Регистр приёмника PTS

PTSDST

Смещение 4 в Управляющем Блоке PTS

Регистр PTSDST используется в одиночной и блочной передачах и режиме HSI. Регистр PTSDST указывает на ячейки памяти приёмника. PTSDST не обязательно инкрементируется в конце цикла обмена. В режиме одиночной передачи PTSCON.0 и PTSCON.2 определяют, будет ли PTSDST инкрементироваться. В режиме блочной передачи PTSCON.0 определяет, будет ли PTSDST инкрементироваться после любой передачи, а PTSCON.2 определяет, будет ли PTSDST сохранять своё последнее значение или восстанавливать исходное. В режиме HSI PTSCON.3 определяет, будет ли PTSDST обновляться в конце цикла обмена.



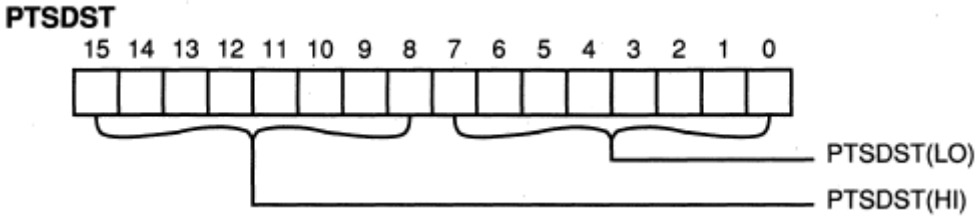
Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-7	PTSDST(LO)	Адрес приёмника PTS, младший байт	XXXX XXXX	Младший байт адреса приёмника PTS.
8-15	PTSDST(HI)	Адрес приёмника PTS, старший байт	XXXX XXXX	Старший байт адреса приёмника PTS.

Регистр источника/приёмника PTS

PTS_S/D

Смещение 3 в Управляющем Блоке PTS

Регистр PTS_S/D используется в АЦП режиме. PTS_S/D указывает на ячейку памяти, которая содержит таблицу команд/ данных для АЦП. В цикле обмена с АЦП слово, на которое указывает PTS_S/D, загружается во временный внутренний регистр, PTS_S/D инкрементируется на 2, а затем регистр AD_RESULT сохраняется по обновлённому адресу PTS_S/D. PTSCON. 3 определяет, будет ли PTS_S/D обновляться в конце PTS цикла обмена для указания на следующее слово в таблице для АЦП.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-7	S/D (LO)	Младший байт адреса источника/ приёмника	XXXX XXXX	Младший байт адреса источника/ приёмника.
8-15	S/D (HI)	Старший байт адреса источника/ приёмника	XXXX XXXX	Старший байт адреса источника/ приёмника.

Регистр выбора PTS (HI/LO)

PTSSEL

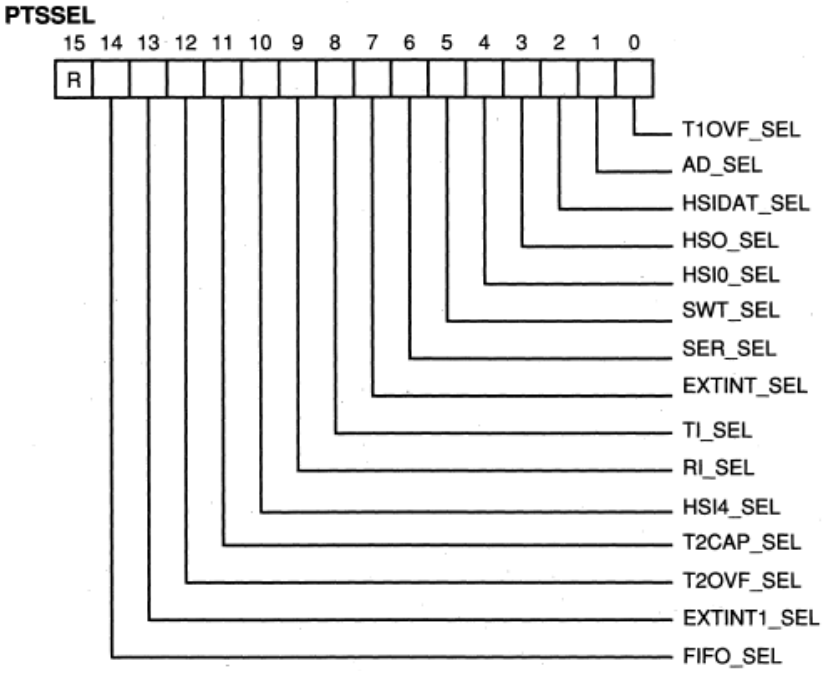
05/04h

Горизонтальное окно 1 (Чтение/Запись)

Регистр PTSSEL состоит из двух байтов. PTSSEL выбирает – цикл PTS или обычная подпрограмма будет обрабатывать прерывание для каждого из пятнадцати запрашиваемых прерываний. Установка бита выбирает цикл PTS; очистка бита выбирает обычную подпрограмму обработки прерываний.

Когда PTSCOUNT достигает нуля, аппаратное обеспечение сбрасывает соответствующий бит PTSSEL и устанавливает бит PTSSRV, который запрашивает прерывание end-of-PTS. Когда это прерывание будет обработано, аппаратное обеспечение сбросит бит PTSSRV. Для восстановления канала обмена бит PTSSEL должен устанавливаться вручную.

Любое прерывание может маскироваться соответствующим битом в регистрах INT_MASK и INT_MASK1. PTS разрешается или запрещается битом PSW.2.

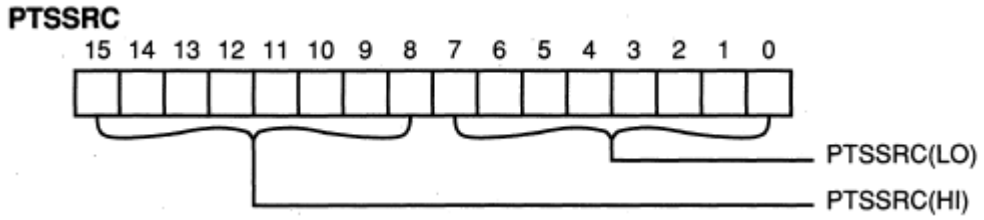


Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	T1OVF_SEL	Выбор прерывания Timer Overflow	0	Установка этого бита ведёт к тому, что прерывание Timer Overflow (INT00) будет обрабатываться циклом PTS.
1	AD_SEL	Выбор прерывания A/D Conversion Complete	0	Установка этого бита ведёт к тому, что прерывание A/D Conversion Complete (INT01) будет обрабатываться циклом PTS.
2	HSIDAT_SEL	Выбор прерывания HSI Data Available/ FIFO Full	0	Установка этого бита ведёт к тому, что прерывание HSI Data Available (INT02) будет обрабатываться циклом PTS.
3	HSO_SEL	Выбор прерывания Event HSO	0	Установка этого бита ведёт к тому, что прерывание Event HSO (INT03) будет обрабатываться циклом PTS.
4	HSI0_SEL	Выбор прерывания внешний HSI. 0	0	Установка этого бита ведёт к тому, что внешнее прерывание HSI.0 (INT04) будет обрабатываться циклом PTS.
5	SWT_SEL	Выбор прерывания Software Timer	0	Установка этого бита ведёт к тому, что прерывание Software Timer (INT05) будет обрабатываться циклом PTS.
6	SER_SEL	Выбор прерывания Serial Port	0	Установка этого бита ведёт к тому, что прерывание Serial Port (INT06) будет обрабатываться циклом PTS.
7	EXTINT_SEL	Выбор прерывания от контакта EXTINT или P0.7	0	Установка этого бита ведёт к тому, что прерывание от контакта EXTINT или P0.7 (INT07) будет обрабатываться циклом PTS.
8	TI_SEL	Выбор прерывания Transmit	0	Установка этого бита ведёт к тому, что прерывание Transmit (INT08) будет обрабатываться циклом PTS.
9	RI_SEL	Выбор прерывания Receive	0	Установка этого бита ведёт к тому, что прерывание Receive (INT09) будет обрабатываться циклом PTS.
10	HSI4_SEL	Выбор прерывания HSI FIFO 4	0	Установка этого бита ведёт к тому, что прерывание HSI FIFO 4 (INT10) будет обрабатываться циклом PTS.
11	T2CAP_SEL	Выбор прерывания Timer 2 Capture	0	Установка этого бита ведёт к тому, что прерывание Timer 2 Capture (INT11) будет обрабатываться циклом PTS.
12	T2OVF_SEL	Выбор прерывания Timer 2 Overflow	0	Установка этого бита ведёт к тому, что прерывание Timer 2 Overflow (INT12) будет обрабатываться циклом PTS.
13	EXTINT1_SEL	Выбор прерывания от контакта EXTINT1	0	Установка этого бита ведёт к тому, что прерывание контакта EXTINT1 (INT13) будет обрабатываться циклом PTS.
14	FIFO_SEL	Выбор прерывания HSI FIFO Full	0	Установка этого бита ведёт к тому, что прерывание HSI FIFO Full (INT14) будет обрабатываться циклом PTS.
15	–	–	0	Зарезервировано; всегда записывают 0.

Регистр источника PTS PTSSRC

Смещение 2 в Управляющем Блоке PTS

Регистр PTSSRC используется в режимах одиночной передачи, блочной передачи и HSO. Регистр PTSSRC указывает на ячейки памяти источника. PTSSRC необязательно инкрементируется в конце цикла обмена. В режиме одиночной передачи PTSCON.1 и PTSCON.3 определяют, будет ли PTSSRC инкрементироваться. В режиме блочной передачи PTSCON.1 определяет, будет ли PTSSRC инкрементироваться после каждой передачи, а PTSCON.3 определяет, будет PTSSRC сохранять своё последнее значение или восстанавливать своё первоначальное значение. В режиме HSO PTSCON.3 определяет, будет ли PTSSRC обновляться в конце цикла обмена.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-7	PTSSRC(LO)	Младший байт адреса источника обмена	XXXX XXXX	Младший байт адреса источника обмена
8-15	PTSSRC(HI)	Старший байт адреса источника обмена	XXXX XXXX	Старший байт адреса источника обмена

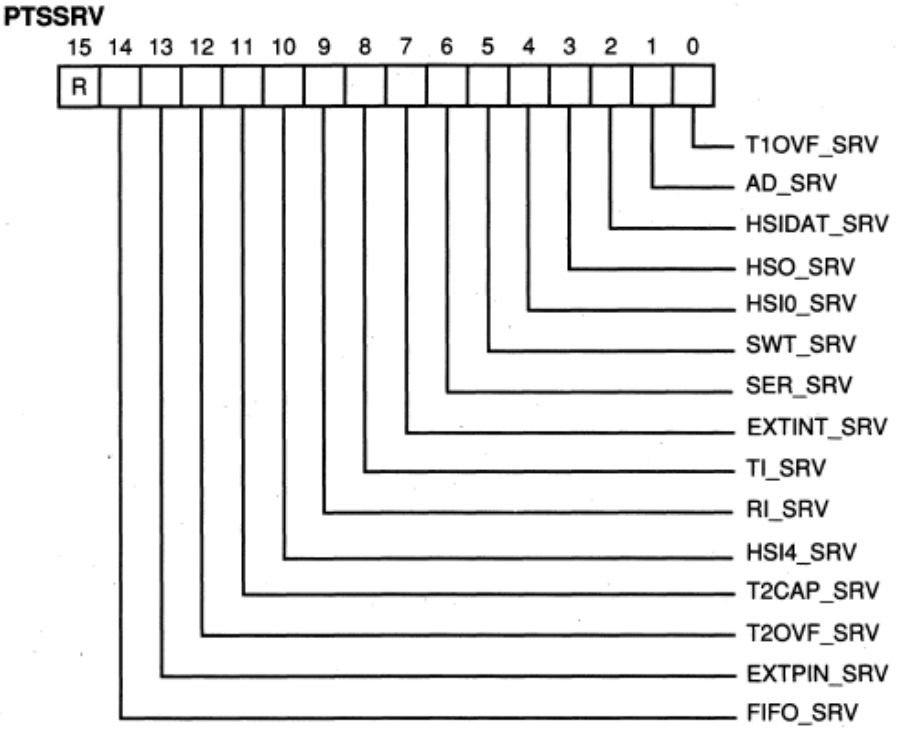
Регистр обслуживания PTS **PTSSRV**

07/06h Горизонтальное окно 1 (Чтение/Запись)

Регистр PTSSRV состоит из двух байт. PTSSRV используется аппаратным обеспечением для указания на то, что последний цикл обмена обслужен. Когда PTSCOUNT достигает нуля, аппаратное обеспечение сбрасывает соответствующий бит PTSSEL и устанавливает бит PTSSRV, который запрашивает прерывание end-of-PTS. Когда это прерывание вызвано, аппаратное обеспечение сбрасывает бит PTSSRV. Для восстановления канала обмена бит PTSSEL должен устанавливаться вручную.

PTS разрешается или запрещается битом PSW.2, который устанавливается командой EPTS и сбрасывается командой DPTS. Прерывания разрешаются или запрещаются (маскируются) соответствующими битами в регистрах INT_MASK или INT_MASK1. Обработка отдельных прерываний циклами обмена разрешается соответствующими битами в регистре PTSSEL.

Векторы прерывания end-of-PTS инициализируются через те же ячейки памяти, что и соответствующие векторы обычных прерываний. Например, если PTSSEL.8 установлен, прерывание передачи (TI) обрабатывается вектором PTS по адресу 2050h, с вектором end-of-PTS по адресу 2030h. (Расположение векторов прерываний смотрите в таблице B.4.)



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0	T1OVF_SRV	Обслуживание прерывания Timer 1 Overflow	0	Этот бит устанавливается, когда PTSCOUNT для цикла PTS Timer 1 Overflow достигает нуля, что вызывает прерывание end-of-PTS с вектором 2000h
1	AD_SRV	Обслуживание прерывания A/D Conversion Complete	0	Этот бит устанавливается, когда PTSCOUNT для цикла PTS A/D Conversion Complete достигает нуля, что вызывает прерывание end-of-PTS с вектором 2002h
2	HSIDAT_SRV	Обслуживание прерывания HSI Data Available/ FIFO Full	0	Этот бит устанавливается, когда PTSCOUNT для цикла PTS HSI Data Available достигает нуля, что вызывает прерывание end-of-PTS с вектором 2004h.
3	HSO_SRV	Обслуживание прерывания Event HSO	0	Этот бит устанавливается, когда PTSCOUNT для цикла PTS HSO достигает нуля, что вызывает прерывание end-of-PTS с вектором 2006h.
4	HSI0_SRV	Обслуживание прерывания HSI.0	0	Этот бит устанавливается, когда PTSCOUNT для цикла PTS HSI.0 достигает нуля, что вызывает прерывание end-of-PTS с вектором 2008h.
5	SWT_SRV	Обслуживание прерывания Software Timer	0	Этот бит устанавливается, когда PTSCOUNT для цикла PTS Software Timer достигает нуля, что вызывает прерывание end-of-PTS с вектором 200Ah.
6	SER_SRV	Обслуживание прерывания Serial Port	0	Этот бит устанавливается, когда PTSCOUNT для цикла PTS Serial Port достигает нуля, что вызывает прерывание end-of-PTS с вектором 200Ch.

7	EXTINT_SRV	Обслуживание прерывания от контакта EXTINT или P0.7	0	Этот бит устанавливается, когда PTSCOUNT для цикла PTS EXTINT достигает нуля, что вызывает прерывание end-of-PTS с вектором 200Eh.
8	TI_SRV	Обслуживание прерывания Transmit	0	Этот бит устанавливается, когда PTSCOUNT для цикла PTS Transmit достигает нуля, что вызывает прерывание end-of-PTS с вектором 2030h.
9	RI_SRV	Обслуживание прерывания Receive	0	Этот бит устанавливается, когда PTSCOUNT для цикла PTS Receive достигает нуля, что вызывает прерывание end-of-PTS с вектором 2032h.
10	HSI4_SRV	Обслуживание прерывания HSI FIFO 4	0	Этот бит устанавливается, когда PTSCOUNT для цикла PTS HSI FIFO 4 достигает нуля, что вызывает прерывание end-of-PTS с вектором 2034h.
11	T2CAP_SRV	Обслуживание прерывания Timer 2 Capture	0	Этот бит устанавливается, когда PTSCOUNT для цикла PTS Timer 2 Capture достигает нуля, что вызывает прерывание end-of-PTS с вектором 2036h.
12	T2OVF_SRV	Обслуживание прерывания Timer 2 Overflow	0	Этот бит устанавливается, когда PTSCOUNT для цикла PTS Timer 2 Overflow достигает нуля, что вызывает прерывание end-of-PTS с вектором 2038h.
13	EXTPIN_SRV	Обслуживание прерывания от вывода EXTINT1	0	Этот бит устанавливается, когда PTSCOUNT для цикла PTS EXTINT1 достигает нуля, что вызывает прерывание end-of-PTS с вектором 203Ah.
14	FIFO_SRV	Обслуживание прерывания HSI FIFO Full	0	Этот бит устанавливается, когда PTSCOUNT для цикла PTS HSI FIFO Full достигает нуля, что вызывает прерывание end-of-PTS с вектором 203Eh.
15	–	–	0	Зарезервировано; всегда записывают 0.

Регистр управления PWM0

PWM0_CONTROL

17h

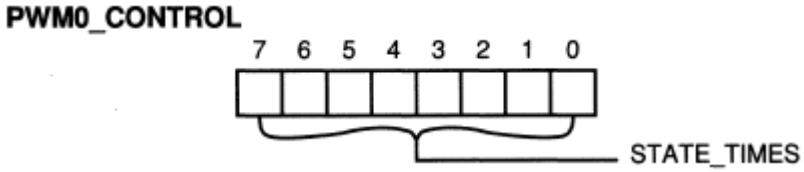
Горизонтальное окно 0 (Запись), горизонтальное окно 15 (Чтение)

PWM0 мультиплексирован с P2.5. Для разрешения функции вывода PWM0 бит IOC1.0 должен быть установлен.

8-битный счётчик PWM инкрементируется каждый машинный такт (с запрещённым предварительным делителем PWM) или каждые два машинных такта (с разрешённым предварительным делителем PWM). Когда счётчик равен 0, выход PWM0 имеет высокий уровень. Он остаётся высоким, пока величина счётчика не достигнет величины регистра PWM0_CONTROL, в момент совпадения на выходе появляется низкий уровень. Когда счётчик переполняется, выход опять переключается на высокий уровень. Когда PWM0_CONTROL равен 0, выход всегда остаётся в низком уровне.

Регистр PWM0_CONTROL вместе с IOC2.2 определяет, как долго вывод PWM0 задерживается на высоком уровне во время импульса, эффективно управляя циклом работы. Значение, записываемое в регистр PWM0_CONTROL, может быть от 0 до 255 машинных тактов (с заполнением от 0 % до 99,6 %).

Установка IOC2.2 разрешает предварительный делитель частоты синхронизации на два для PWM; очистка IOC2.2 запрещает его. Когда предварительный делитель частоты разрешён, общая длительность импульса равна 512 машинных тактов, и значение PWM0_ CONTROL умножается на 4. Когда он запрещён, общая длительность импульса равна 256 машинных тактов, и значение PWM0_CONTROL умножается на 2.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-7	STATE_TIMES	Длительность высокого уровня на выходе PWM0 в машинных тактах	0000 0000	Эти биты определяют количество машинных тактов, в течении которых PWM0 на выходе удерживает высокий уровень. В этот регистр записывается 16-ричная величина (00h-FFh).

Для определения периода PWM и времени, в течение которого выход будет иметь высокий уровень, используются следующие формулы:

	Предварительный делитель частоты синхронизации запрещён (IOC2.2=0)	Предварительный делитель частоты синхронизации разрешён (IOC2. 2=1)
Период PWM (в мкс) =	$\frac{512}{F_{OSC}}$	$\frac{1024}{F_{OSC}}$
Высокий уровень PWMx (в мкс) =	$\frac{PWMx\_CONTROL \times 2}{F_{OSC}}$	$\frac{PWMx\_CONTROL \times 4}{F_{OSC}}$,

где f_{OSC} – частота XTAL1, в МГц;

x = 0, 1, 2 – номер PWM.

Например, если f_{OSC} = 16 МГц, то период вывода PWM равен 32 мкс. Если IOC2.2 = 0 и PWM_CONTROL = 8Ah (138, десятичное), то выход удерживается в высоком уровне 17,25 мкс (в низком 14, 8 мкс) из всех 32 мкс, что составляет примерно 54 % рабочего цикла. Когда IOC2.2 = 1, то эти же значения приведут к периоду в 64 мкс, при этом высокий уровень будет в течение 34,5 мкс (низкий 29,5 мкс), что также составляет около 54 % рабочего цикла.

Регистр управления PWM1

PWM1_CONTROL

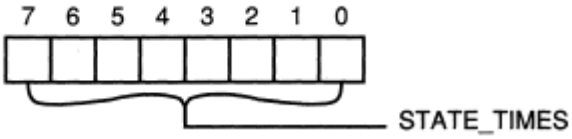
16h

Горизонтальное окно 1 (Чтение/Запись)

PWM1 мультиплексирован с P1.3. Для разрешения функции вывода для PWM1 IOC3.2 должен быть установлен.

Регистр PWM1_CONTROL вместе с IOC2.2 определяет, как долго выход PWM1 будет находиться в высоком уровне. Величина, записываемая в регистр PWM1_CONTROL, может быть от 0 до 255 машинных тактов (0–99,6) % рабочего цикла. Для получения дополнительной информации смотрите описание регистра PWM0_CONTROL.

PWM1_CONTROL



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-7	STATE_TIMES	Длительность высокого уровня на выходе PWM1 в машинных тактах	0000 0000	Эти биты определяют количество машинных тактов, в течение которых PWM1 на выходе удерживает высокий уровень. В этот регистр записывается 16-ричная величина (00h-FFh).

Регистр управления PWM2

PWM2_CONTROL

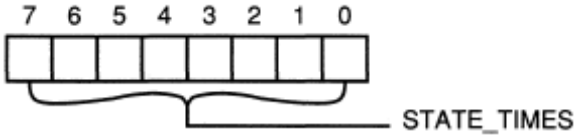
17h

Горизонтальное окно 1 (Чтение/Запись)

PWM2 мультиплексирован с P1.4. Для разрешения функции вывода для PWM2 IOC3.3 должен быть установлен.

Регистр PWM2_CONTROL вместе с IOC2.2 определяет, как долго выход PWM2 будет находиться в высоком уровне. Величина, записываемая в регистр PWM2_CONTROL может быть от 0 до 255 машинных тактов (0–99,6) % рабочего цикла). Для получения дополнительной информации смотрите описание регистра PWM0_CONTROL.

PWM2_CONTROL



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-7	STATE_TIMES	Длительность высокого уровня на выходе PWM2 в машинных тактах	0000 0000	Эти биты определяют количество машинных тактов, в течении которых PWM2 на выходе удерживает высокий уровень. В этот регистр записывается 16-ричная величина (00h-FFh).

Буферный регистр приёма последовательного порта

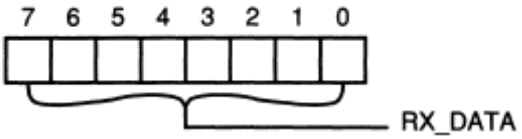
SBUF (RX)

07h

Горизонтальное окно 0 (Чтение), горизонтальное окно 15 (Запись)

Буферный регистр приёма последовательного порта SBUF(RX) содержит данные, принятые из последовательного порта. Приёмник последовательного порта имеет двойную буферизацию и может принимать второй байт данных до того, как первый байт считан. Данные задерживаются в сдвиговом регистре приёмника, пока не получен последний бит данных, а затем байты данных загружаются в SBUF (RX). Если данные в сдвиговом регистре загружаются в SBUF (RX) до того, как предыдущий байт считан, будет установлен бит ошибки переполнения (SP_STAT.2). Данными в SBUF (RX) всегда будет последний принятый байт, но никогда комбинация последних двух байтов.

SBUF (RX)



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-7	RX_DATA	Принятые данные	0000 0000	Эти биты содержат последний байт, принятый из последовательного порта.

Передающий буферный регистр последовательного порта

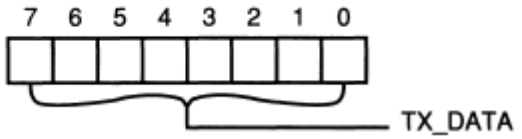
SBUF (TX)

07h

Горизонтальное окно 0 (Запись), горизонтальное окно 15 (Чтение)

Передающий буферный регистр последовательного порта SBUF(TX) содержит данные, которые готовы к передаче. В режимах 1, 2 и 3 запись в SBUF (TX) начинает передачу. В режиме 0 запись в SBUF (TX) начинает передачу только при запрещённом приёме (SP_CON.3=0).

SBUF (TX)



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-7	TX_DATA	Данные для передачи	0000 0000	Эти биты содержат байт данных, предназначенный для передачи последовательным портом.

Указатель стека

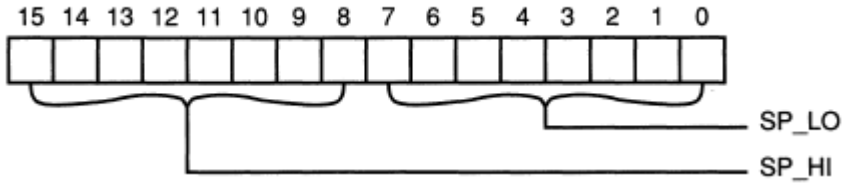
SP

19/18h

Все горизонтальные окна (Чтение/Запись)

Системный указатель стека может указывать на любую ячейку в 64К байтной внутренней или внешней памяти; он должен хранить выравненное слово и должен быть инициализирован перед использованием. Указатель стека декрементируется перед командой PUSH и инкрементируется после команды POP, так что указатель стека нужно инициализировать на 2 байта выше верхней ячейки стека. Если стековые операции не производятся, ячейки 18h и 19h могут использоваться как стандартное ОЗУ (RAM).

SP



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-7	SP_LO	Младший байт указателя стека	XXXX XXXX	Младший байт системного указателя стека.
8-15	SP_HI	Старший байт указателя стека	XXXX XXXX	Старший байт системного указателя стека.

Регистр управления последовательным портом

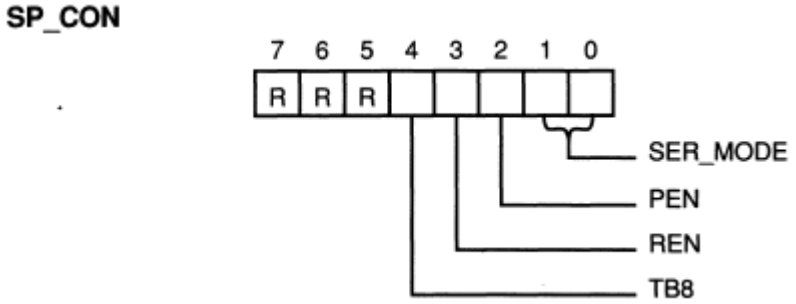
SP_CON

11h

Горизонтальное окно 0 (Запись), горизонтальное окно 15 (Чтение)

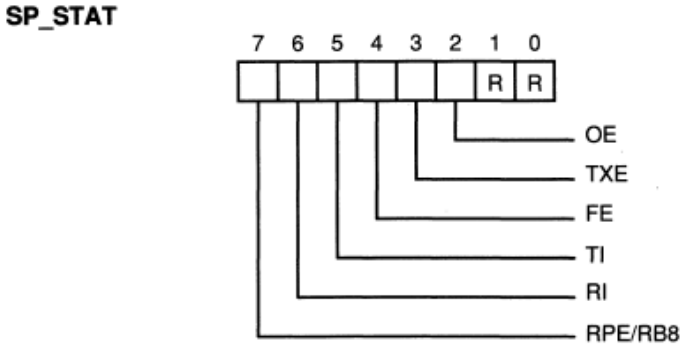
Регистр управления последовательным портом SP_CON выбирает режим связи, разрешает или запрещает приём, а также контролирует чётность и позволяет работать с 9-битной передачей данных.

TXD мультиплексирован с P2.0. Для разрешения функционирования TXD IOC1.5 должен быть установлен. TXD служит как канал передачи для режимов 1, 2 и 3 последовательного порта и как канал синхронизация для сдвигового регистра в режиме 0. RXD мультиплексирован с P2.1. Для разрешения функционирования RXD SP_CON.3 должен быть установлен. RXD получает данные последовательного порта в режимах 1, 2 и 3 и служит для ввода или вывода данных в режиме 0 (при выводе он функционирует как выход с открытым стоком).



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-1	SER_MODE	Выбор режима	1 1	Эти два бита выбирают режим связи. Бит 1 Бит 0 0 0 Режим 0 0 1 Режим 1 1 0 Режим 2 1 1 Режим 3
2	PEN	Разрешение контроля чётности	0	В режимах 1 и 3 установка этого бита разрешает функцию чётности. В режиме 2 этот бит должен быть очищен. Когда этот бит установлен, TB8 содержит дополнение до четности для передачи. С разрешённым контролем чётности при приёме.
3	REN	Разрешение приёма	1	Установка этого бита разрешает функционирование RXD в канале P2.1/ RDX. Когда этот бит установлен, по срезу сигнала начинается приём в режимах 1, 2 и 3. В режиме 0 этот бит должен быть сброшен для начала передачи и установлен для начала приёма. Очистка этого бита останавливает приём и препятствует дальнейшему приёму.
4	TB8	Передача 9-ого бита данных	0	Это 9-ый бит данных, который передаётся в режимах 2 и 3. Этот бит сбрасывается после любой передачи, поэтому он должен быть установлен до записи в SBUF (TX). Когда SPCON.2 установлен, этот бит содержит дополнение до чётности.
5-7	-	-	000	Зарезервированы; всегда записывают 0.

Регистр состояния последовательного порта
SP_STAT
11h
Горизонтальное окно 0 (Чтение), горизонтальное окно 15 (Запись)
Регистр состояния последовательного порта содержит биты, которые показывают состояние последовательного порта.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-1	–	–	11	Зарезервированы; всегда записывают нули.
2	OE	Ошибка переполнения	0	Этот бит устанавливается, если данные из сдвигового регистра приёма загружены в регистр SBUF (RX), до того, как предыдущий байт считан. Чтение из SP_STAT очищает этот бит.
3	TXE	SBUF (TX) пуст	0	Этот бит устанавливается, если буфер передачи пуст и готов к приёму двух символов. Он очищается после записи байта в SBUF (TX).
4	FE	Ошибка кадра	1	Этот бит устанавливается, если стоповый бит не найден в течение определённого периода времени. Чтение из SP_STAT очищает этот бит.
5	TI	Прерывание передачи	0	Этот бит устанавливается в начале передачи стопового бита. Чтение из SP_STAT очищает этот бит.
6	RI	Прерывание при приёме	0	Этот бит устанавливается после выборки последнего бита данных. Чтение из SP_STAT очищает этот бит. Этот бит не обязательно сбрасывать для нормального приёма следующих данных последовательным портом.
7	RPE/RB8	Ошибка приёма по чётности / принятый 8-й бит	0	RPE устанавливается при запрещённой проверке чётности (SP_CON.2=0) и получении 9-ого бита данных, равного 1. RP8 устанавливается при разрешённой проверке чётности (SP_CON.2=1) и возникшей ошибке чётности. Чтение из SP_STAT очищает этот бит.

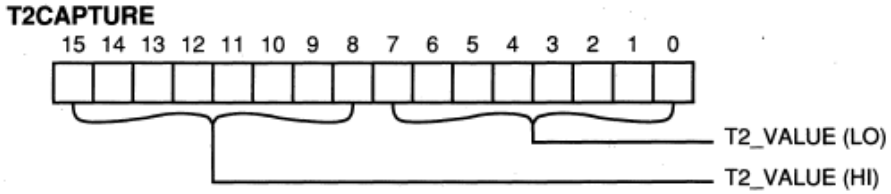
Регистр фиксации значения таймера 2

T2CAPTURE

0D/0Ch

Горизонтальное окно 15 (Чтение/Запись)

Передний фронт сигнала в P2.7 ведёт к запоминанию значения таймера 2 в регистре T2CAPTURE и генерированию прерывания Timer 2 Capture (INT11, 2036h). Для разрешения прерывания необходимо установить бит INT_MASK1.3.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-7	T2_VALUE (LO)	Младший байт таймера 2	XXXX XXXX	Эти биты содержат младший байт захваченного значения таймера 2.
8-15	T2_VALUE (HI)	Старший байт таймера 2	XXXX XXXX	Эти биты содержат старший байт зафиксированного значения таймера 2.

Регистр таймера 1

TIMER1

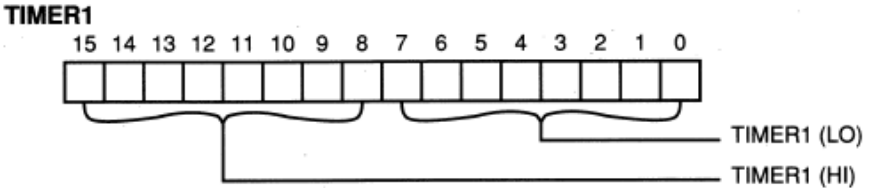
0B/0Ah

Горизонтальное окно 0 (Чтение), горизонтальное окно 15 (Запись)

Два байта регистра TIMER1 содержат значение таймера 1. В этот регистр можно записывать; при инициализации таймера 1 можно записать значение, отличное от нуля.

Таймер 1 – 16-битный, автономно работающий таймер, который инкрементируется каждый 8 машинный такт. Установка IOC1.2 ведёт при переполнении таймера 1 к генерации прерывания Timer Overflow (INT00, 2000h), если в INT_MASK.0 также установлено разрешение прерывания.

Будьте осторожны, если во время записи в таймер 1 работают модули HSI и HSO. Время наступления HSO-события, записанного в CAM, зависит от точного соответствия с таймером 1. Аналогично, изменение таймера 1 между входными событиями в режиме HSI будет искажать временные соотношения между событиями.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-7	TIMER1 (LO)	Значение таймера 1 младший байт	0000 0000	Эти биты составляют младший байт значения таймера 1.
8-15	TIMER1 (HI)	Значение таймера 1 старший байт	0000 0000	Эти биты составляют старший байт значения таймера 1.

Регистр таймера 2

TIMER2

0D/0Ch

Горизонтальное окно 0 (Чтение/Запись)

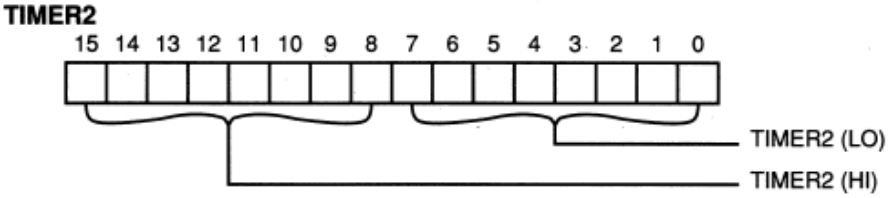
Два байта регистра TIMER2 содержат значение таймера 2. В этот регистр можно записывать; для инициализации таймера 2 разрешается записывать значение, отличное от нуля.

Таймер 2 – 16-битный счётчик, который может быть синхронизирован либо внутренне, либо внешне, в зависимости от состояния IOC3.0. Если выбран внешний источник синхронизации, то IOC0.7 определяет, какой из двух внешних источников будет использоваться. В зависимости от состояния IOC2.0 таймер 2 считает или каждые 8 импульсов (нормальный режим) или каждый импульс (режим быстрого инкрементирования).

IOC2.1 управляет направлением счета таймера 2. Когда IOC2.1 = 0, таймер 2 считает на увеличение; когда IOC2.1 = 1, таймер 2 считает на уменьшение или на увеличение, в зависимости от состояния P2. 6. Будьте осторожны, когда это разрешено, и ведётся работа с HSO. Если таймер 2 не считает от начала до конца все 65536 раз, время события в CAM файле могут никогда не совпасть с текущим значением таймера.

Переполнение таймера 2 может генерировать либо прерывание Timer Overflow (INT00, 2000h), либо прерывание Timer 2 Overflow (INT12, 2038h). Установите или сбросьте IOC1.3, INT_MASK.0 и INT_MASK1.4 для установки желаемого прерывания.

Регистр может обнуляться аппаратно, программно или командой HSO, в зависимости от состояния IOC0.3, IOC0.1 и HSO_COMMAND.0 – HSO_COMMAND.3 соответственно.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-7	TIMER2 (LO)	Значение таймера 2 младший байт	0000 0000	Эти биты составляют младший байт значения таймера 2.
8-15	TIMER2 (HI)	Значение таймера 2 старший байт	0000 0000	Эти биты составляют старший байт значения таймера 2.

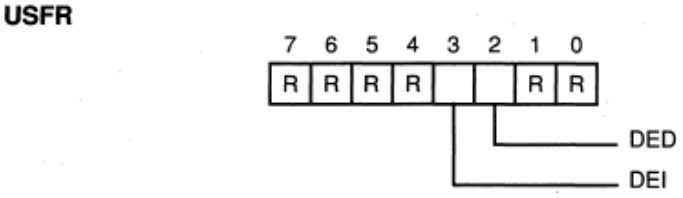
Регистр специальных функций UPROM

USFR

1FF6h (Чтение)

Регистр UPROM SFR содержит два бита, которые запрещают внешнюю выборку данных и команд. Эти биты могут программироваться, но не могут стираться. Остальные биты зарезервированы.

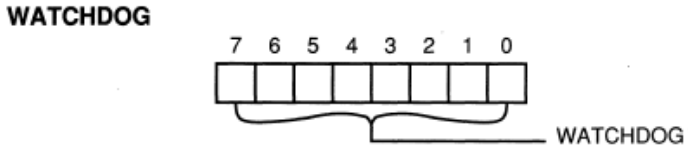
Примечание – Эти биты программируются, но никогда не могут стираться. Программирование этих битов делает невозможным анализ динамических неисправностей. По этой причине устройства с программируемыми битами UPROM не могут быть возвращены производителю для анализа неисправностей.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-1	–	–	00	Зарезервированы; всегда записывают нули.
2	DED	Запрещение внешней выборки данных	–	Установка этого бита препятствует контроллеру шины использовать внешние данные (читать или записывать). Любая попытка доступа через контроллер шины приводит к перезагрузке системы.
3	DEI	Запрещение внешней выборки команд	–	Установка этого бита запрещает контроллеру шины использовать внешние команды (читать или записывать). Любая попытка загрузить внешний адрес приводит к перезагрузке системы.
4-7	–	–	0000	Зарезервированы; всегда записывают нули.

Регистр сторожевого таймера WATCHDOG
0Ah Горизонтальное окно 0 (Сброс/ Разрешение)
Горизонтальное окно 15 (Чтение)

Несброшенный через каждые 65536 машинных тактов сторожевой таймер перезагружает МК. Для очистки сторожевого таймера записывают последовательно, сразу друг за другом, "1Eh" и "E1h" в ячейку 0Ah в горизонтальном окне 0. Очистка этого регистра первый раз разрешает работу сторожевого таймера с начальным значением 0000h, которое инкрементируется каждый машинный такт. После разрешения сторожевой таймер может быть запрещен только через сброс. 8 старших значащих битов значения сторожевого таймера можно прочитать в горизонтальном окне 15.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-7	WATCHDOG	Значение сторожевого таймера	XXXX XXXX	Этот байт содержит 8 старших значащих битов текущего значения сторожевого таймера.

Регистр выбора окна **WSR**
14h Все горизонтальные окна (Чтение/Запись)

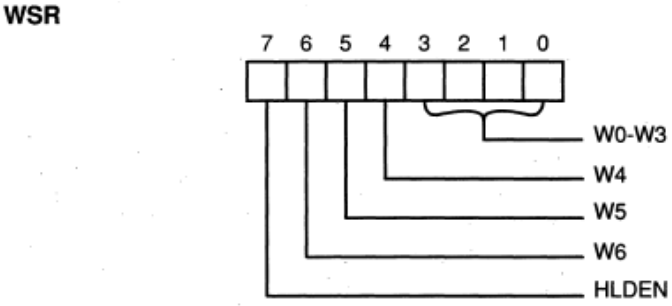
WSR имеет 2 функции. Один бит разрешает и запрещает протокол захвата шины. Остальные биты выбирают горизонтальные (HWindows) и вертикальные окна (VWindows). PUSHA записывает этот регистр в стек; POPA восстанавливает его.

Горизонтальные окна позволяют получить доступ к регистрам специальных функций (SFR) путём преобразования 24-байтного окна в младшие 24 байта Младшего Регистрового Файла. Горизонтальные окна 0, 1 и 15 реализованы в МК. Все другие горизонтальные окна зарезервированы.

Вертикальные окна отображают сегменты ОЗУ (RAM) в старшую часть Младшего Регистрового Файла, в 32-, 64- или 128-байтные сегменты. МК имеет 512 байтов внутреннего RAM (ОЗУ), которые могут быть преобразованы в 16

32-байтных вертикальных окна, в 8 64-байтных вертикальных окна или в 4 128-байтных вертикальных окна.

Горизонтальные и вертикальные окна не могут быть активными одновременно. Для переключения окон следует записывать номер окна в биты 0 – 3 и устанавливать или очищать биты 4 – 6 как надо. Для ясности и получения дополнительной информации смотрите таблицы выбора окон. Таблица B.6 описывает выбор горизонтального окна, таблица B.7 – выбор 128байтных вертикальных окон, таблица B.8 – выбор 64–байтных вертикальных окон, таблица B.9 – выбор 32–байтных вертикальных окон.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-3	W0-W3	0-3 биты выбора окна	0000	В эти биты записывайте номер желаемого горизонтального или вертикального окна. Размер окна выбирается установкой WSR.4, WSR.5 или WSR.6. Действующие горизонтальные окна – 0, 1 и 15. Действующие вертикальные окна – 0-15. Для выбора вертикальных окон 16-31 необходимо установить WSR.4 и WSR.6.
4	W4	4-й бит выбора окна	X	Этот бит устанавливается при выборе 128-байтных вертикальных окон. В других случаях он должен быть равен нулю. Установка этого бита определяет 128-байтные вертикальные окна.
5	W5	5-й бит выбора окна	X	Установка этого бита выбирает 64-байтные вертикальные окна. В других случаях он должен быть равен нулю.
6	W6	6-й бит выбора окна	X	Установка этого бита выбирает 32-байтные вертикальные окна. В других случаях он должен быть равен нулю.
7	HLDEN	Разрешение протокола HOLD#/ HLDA#	0	Этот бит разрешает или запрещает протокол захвата шины. 1 = разрешает 0 = запрещает

Таблица B.6 – Выбор горизонтального окна

Желаемое горизонтальное окно	Биты WSR								16-ричное значение
	7	6	5	4	3	2	1	0	
Hwindow 0	X	0	0	0	0	0	0	0	00h
Hwindow 1	X	0	0	0	0	0	0	1	01h
Hwindow 15	X	0	0	0	1	1	1	1	15h

Таблица В.7 - Выбор 128-байтных вертикальных окон

Желаемое вертикальное окно	Начальный адрес	Биты WSR								16-ричное значение
		7	6	5	4	3	2	1	0	
0	0000h	X	0	0	1	0	0	0	0	10h
1	0080h	X	0	0	1	0	0	0	1	11h
2	0100h	X	0	0	1	0	0	1	0	12h
3	0180h	X	0	0	1	0	0	1	1	13h

Таблица В.8 – Выбор 64-байтных вертикальных окон

Желаемое вертикальное окно	Начальный адрес	Биты WSR								16-ричное значение
		7	6	5	4	3	2	1	0	
0	0000h	X	0	1	0	0	0	0	0	20h
1	0040h	X	0	1	0	0	0	0	1	21h
2	0080h	X	0	1	0	0	0	1	0	22h
3	00C0h	X	0	1	0	0	0	1	1	23h
4	0100h	X	0	1	0	0	1	0	0	24h
5	0140h	X	0	1	0	0	1	0	1	25h
6	0180h	X	0	1	0	0	1	1	0	26h
7	01C0h	X	0	1	0	0	1	1	1	27h

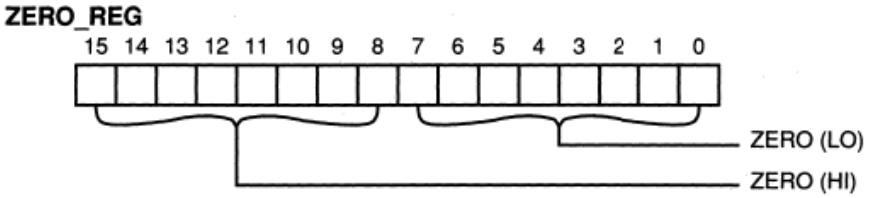
Таблица В.9 – Выбор 32-байтных вертикальных окон

Желаемое вертикальное окно	Начальный адрес	Биты WSR								16-ричное значение
		7	6	5	4	3	2	1	0	
0	0000h	X	1	0	0	0	0	0	0	40h
1	0020h	X	1	0	0	0	0	0	1	41h
2	0040h	X	1	0	0	0	0	1	0	42h
3	0060h	X	1	0	0	0	0	1	1	43h
4	0080h	X	1	0	0	0	1	0	0	44h
5	00A0h	X	1	0	0	0	1	0	1	45h
6	00C0h	X	1	0	0	0	1	1	0	46h
7	00E0h	X	1	0	0	0	1	1	1	47h
8	0100h	X	1	0	0	1	0	0	0	48h
9	0120h	X	1	0	0	1	0	0	1	49h
10	0140h	X	1	0	0	1	0	1	0	4Ah
11	0160h	X	1	0	0	1	0	1	1	4Bh
12	0180h	X	1	0	0	1	1	0	0	4Ch
13	01A0h	X	1	0	0	1	1	0	1	4Dh
14	01C0h	X	1	0	0	1	1	1	0	4Eh
15	01E0h	X	1	0	0	1	1	1	1	4Fh

Регистр нуля **ZERO_REG**

01/00h Все горизонтальные окна (Чтение/ Запись)

Двухбайтный регистр нуля всегда равен нулю. Он используется как фиксированный источник нуля при сравнениях и вычислениях. ZERO_REG может также использоваться как переменная типа WORD в длинно-индексной ссылке. Комбинация выбора регистра и режима адресации делает возможной прямую адресацию к любой ячейке памяти. Команда CMPL (сравнение чисел типа LONG) вместе со средствами регистра ZERO_REG сравнивает с 32-битным нулём.



Номер бита	Мнемоника	Имя бита	Состояние после сброса	Описание
0-7	ZERO(LO)	Регистр нуля, младший байт	0000 0000	Это младший байт регистра нуля
8-15	ZERO(HI)	Регистр нуля, старший байт	0000 0000	Это старший байт регистра нуля

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ докум.	Подп.	Дата
	измененных	замененных	новых	аннулированных				
-	-	-	Все	-	274			23.05.08
1	-	19	-	-	-			31.07.09
2	-	146	-	-	-			01.10.09
3	-	2, 3	29а, 29б, 29в	-	277			07.04.10
4	-	105, 106, 111	-	-	-			27.10.10
5	-	7,8,18-22, 168	-	-	-			20.02.13