МИКРОСХЕМЫ ИНТЕГРАЛЬНЫЕ

1874ВЕ86Т, 1874ВЕ16Т

Руководство пользователя

2006

Содержание

Введение

В настоящем руководстве КФДЛ.431295.021 приведено описание архитектуры, функционального построения, системы команд и особенностей применения ИМС 1874ВЕ86Т, 1874ВЕ16Т, которые представляют собой СБИС однокристального 16-разрядного микроконтроллера с тактовой частотой 16 МГц, ОЗУ 488×8 бит, 13-канальным 8/10-разрядным АЦП, с функцией управления двигателями, в том числе модификация с внутренней программной памятью (ЭППЗУ типа OTPROM) объемом 16К×8 (функциональные аналоги 87С196МС, 80С196МС).

Разработанные микросхемы будут служить основой для создания перспективных цифровых систем управления.

В настоящее время одним из основных факторов обеспечения конкурентоспособности отечественной радиоэлектронной аппаратуры и ее живучести является применение при ее разработке и производстве импортонезависимой элементной базы. Импортозамещение электронных компонентов наиболее эффективно в случае использования полных функциональных аналогов изделий микроэлектроники.

Появление отечественного варианта высокопроизводительного 16-разрядного микроконтроллера для систем управления и призвано обеспечить импортонезависимость проектируемых на его основе устройств.

Настоящее руководство может служить практическим пособием по применению микроконтроллеров для разработчиков систем на основе ИМС 1874ВЕ86Т и 1874ВЕ16Т.

1 Назначение и область применения

Архитектура ИМС ориентирована для создания цифровых управляющих систем, функционирующих в режиме реального времени с возможностью адаптации и модификации под конкретные приложения. Изделия могут служить элементной базой для цифровых систем управления различной аппаратурой, силовой электроникой, бытовой техникой, теле-, аудио- и видеоаппаратурой, в компьютерной технике, автомобильной технике 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 высокопроизводительный 16-разрядный микроконтроллер, выполненный в двух вариантах:

- 1874ВЕ86Т – со встроенной памятью типа ЭППЗУ 16К×8 типа OTPROM (аналог 87С196МС фирмы Intel);
- 1874ВЕ16Т – без встроенной памяти программ (функциональный аналог 80С196МС фирмы Intel) и предназначенный для применения во встроенных цифровых системах управления.

2 Краткое техническое описание ИМС 1874ВЕ86Т, 1874ВЕ16Т

2.1 Функциональные возможности:

- разрядность данных, бит	16;
- тактовая частота, МГц	от 8 до 16;
- динамически конфигурируемая шина данных, бит	8 или 16;
- встроенная память программ, бит	16К×8;
- регистровое ОЗУ, бит	488×8;
- число источников прерывания	14;
- число параллельных 8-разрядных портов ввода/вывода	5;
- разрядность сторожевого таймера	16;
- число таймеров/счетчиков	2;
- разрядность таймеров/счетчиков	16;
- число каналов аналого-цифрового преобразователя	13;
- число разрядов аналого-цифрового преобразователя	8, 10;
- процессор событий (EPA)	1;
- блок ШИМ сигналов	2;
- периферийный сервер (PTS)	1;
- 3-х фазный генератор сигналов (функция управления двигателем)	1.

Микросхемы разработаны в металлокерамическом планарном корпусе с четырехсторонним расположением выводов 4235.88-1.

Номинальное значение напряжения питания микросхем плюс 5 В. Допустимое отклонение напряжения питания от номинального ±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.

Структурная схема ИМС приведена на рисунке 2.1.

Условное графическое обозначение микросхем приведено на рисунке 2.2.

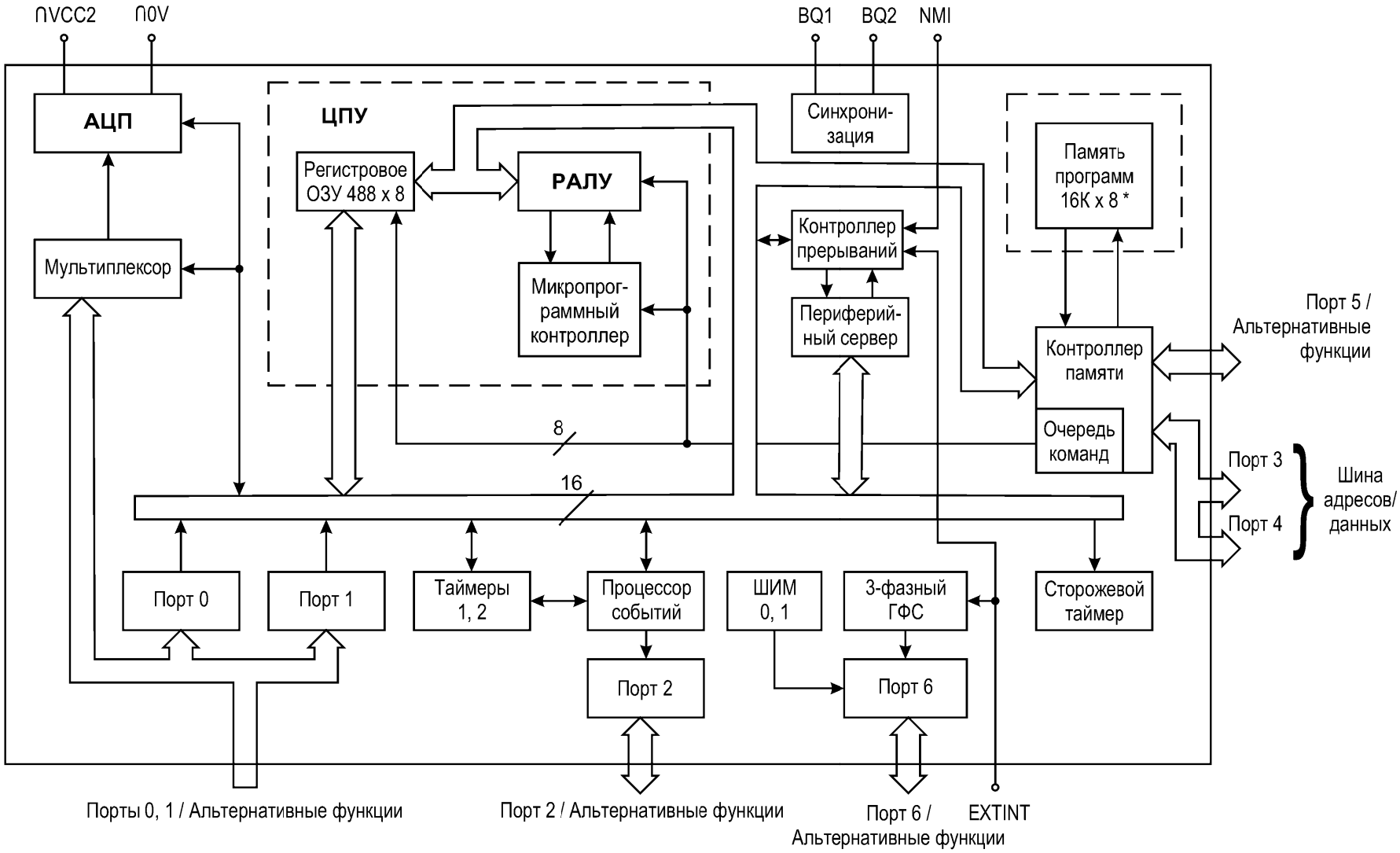


Рисунок 2.1 – Структурная схема микроконтроллеров

* Для варианта со встроенной памятью программ.

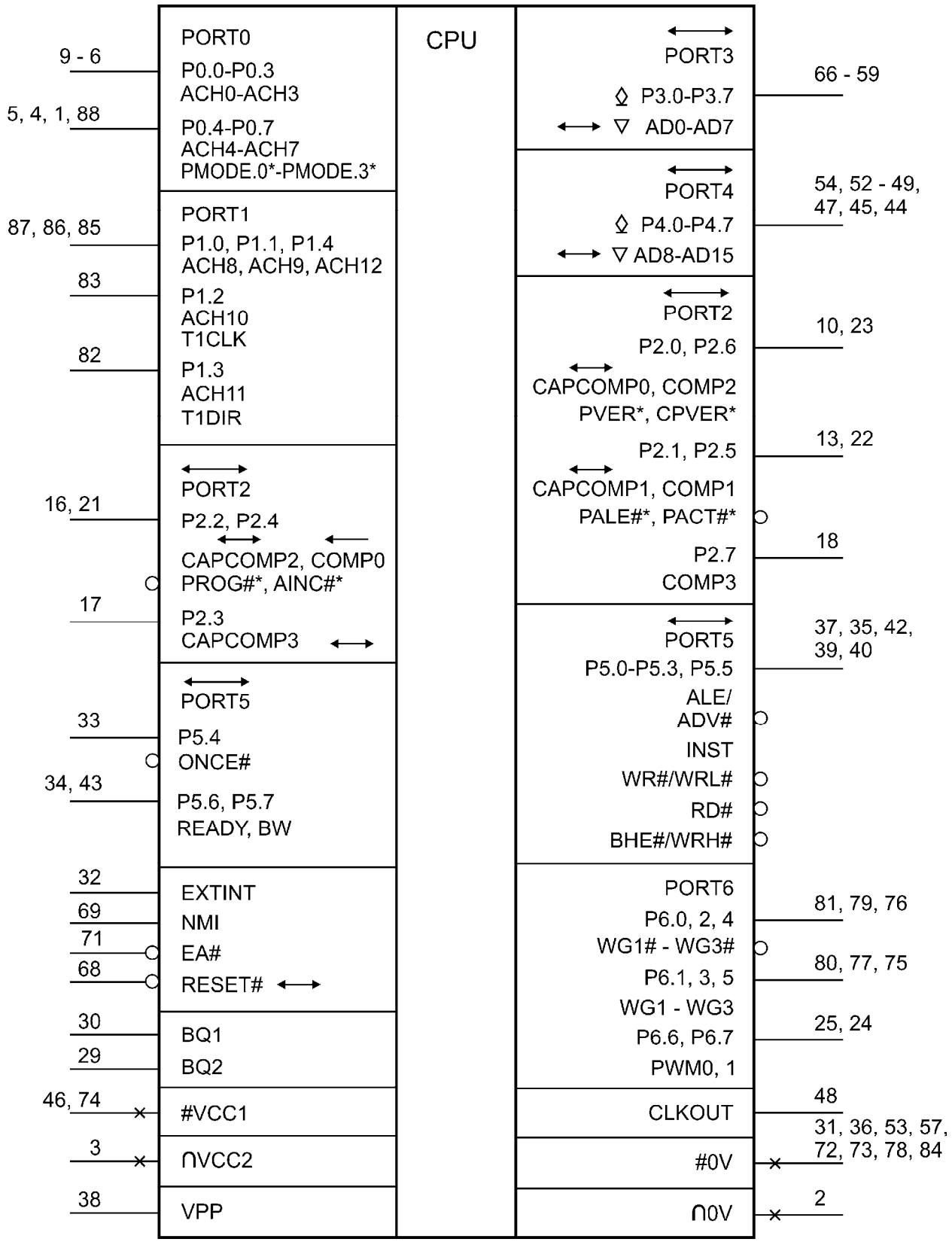


Рисунок 2.2 – Условно-графическое обозначение ИМС 1874ВЕ86Т, 1874ВЕ06Т

* Только для МК 1874ВЕ86Т.

Таблица 2.1 – Назначение выводов микросхемы

Обозначение вывода	Номер вывода	Функциональное назначение	Тип вывода	Обозначение альтернативной функции вывода
1	2	3	4	5
P0.0	9	Вход "порт 0, 0 разряд" Вход АЦП, канал 0	I I	 ACH0
P0.1	8	Вход "порт 0, 1 разряд" Вход АЦП, канал 1	I I	 ACH1
P0.2	7	Вход "порт 0, 2 разряд" Вход АЦП, канал 2	I I	 ACH2
P0.3	6	Вход "порт 0, 3 разряд" Вход АЦП, канал 3	I I	 ACH3
P0.4	5	Вход "порт 0, 4 разряд" Вход АЦП, канал 4 Вход "режим программирования, 0 разряд"	I I I	 ACH4, PMODE.0 *
P0.5	4	Вход "порт 0, 5 разряд" Вход АЦП, канал 5 Вход "режим программирования, 1 разряд"	I I I	 ACH5, PMODE.1 *
P0.6	1	Вход "порт 0, 6 разряд" Вход АЦП, канал 6 Вход "режим программирования, 2 разряд"	I I I	 ACH6, PMODE.2 *
P0.7	88	Вход "порт 0, 7 разряд" Вход АЦП, канал 7 Вход "режим программирования, 3 разряд"	I I I	 ACH7, PMODE.3 *
P1.0	87	Вход "порт 1, 0 разряд" Вход АЦП, канал 8	I I	 ACH8
P1.1	86	Вход "порт 1, 1 разряд" Вход АЦП, канал 9	I I	 ACH9
P1.2	83	Вход "порт 1, 2 разряд" Вход АЦП, канал 10 Вход "синхронизация таймера 1"	I I I	 ACH10 T1CLK
P1.3	82	Вход "порт 1, 3 разряд" Вход АЦП, канал 11 Вход "режим таймера 1"	I I I	 ACH11 T1DIR
P1.4	85	Вход "порт 1, 4 разряд" Вход АЦП, канал 12	I I	 ACH12
P2.0	10	Вход-выход "порт 2, 0 разряд" Вход-выход "выборка- сравнение, 0 канал" Выход "верификация"	I/O I/O O	 CAPCOMP0 PVER *
P2.1	13	Вход-выход "порт 2, 1 разряд" Вход-выход "выборка- сравнение, 1 канал" Выход "строб записи "	I/O I/O O	 CAPCOMP1 PALE# *
P2.2	16	Вход-выход "порт 2, 2 разряд" Вход-выход "выборка- сравнение, 2 канал" Вход "программирование"	I/O I/O I	 CAPCOMP2, PROG# *
P2.3	17	Вход-выход "порт 2, 3 разряд" Вход-выход "выборка-сравнение, 3 канал"	I/O I/O	 CAPCOMP3

Продолжение таблицы 2.1

1	2	3	4	5
P2.4	21	Вход-выход "порт 2, 4 разряд" Выход "сравнение, 0 канал" Вход "автоинкремент"	I/O O I	 COMP0 AINC# *
P2.5	22	Вход-выход "порт 2, 5 разряд" Выход "сравнение, 1 канал" Выход "подтверждение программирования"	I/O O O	 COMP1 PACT# *
P2.6	23	Вход-выход "порт 2, 6 разряд" Выход "сравнение, 2 канал" Выход "ошибка программирования"	I/O O O	 COMP2 CPVER *
P2.7	18	Вход-выход "порт 2, 7 разряд" Выход "сравнение, 3 канал"	I/O O	 COMP3
P3.0	66	Вход-выход "порт 3, 0 разряд" Вход-выход "адрес-данные, 0 разряд"	I/O/2 I/O/Z	 AD0
P3.1	65	Вход-выход "порт 3, 1 разряд" Вход-выход "адрес-данные, 1 разряд"	I/O/2 I/O/Z	 AD1
P3.2	64	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	60	Вход-выход "порт 3, 6 разряд" Вход-выход "адрес-данные, 6 разряд"	I/O/2 I/O/Z	 AD6
P3.7	59	Вход-выход "порт 3, 7 разряд" Вход-выход "адрес-данные, 7 разряд"	I/O/2 I/O/Z	 AD7
P4.0	54	Вход-выход "порт 4, 0 разряд" Вход-выход "адрес-данные, 8 разряд"	I/O/2 I/O/Z	 AD8
P4.1	52	Вход-выход "порт 4, 1 разряд" Вход-выход "адрес-данные, 9 разряд"	I/O/2 I/O/Z	 AD9
P4.2	51	Вход-выход "порт 4, 2 разряд" Вход-выход "адрес-данные, 10 разряд"	I/O/2 I/O/Z	 AD10
P4.3	50	Вход-выход "порт 4, 3 разряд" Вход-выход "адрес-данные, 11 разряд"	I/O/2 I/O/Z	 AD11
P4.4	49	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	45	Вход-выход "порт 4, 6 разряд" Вход-выход "адрес-данные, 14 разряд"	I/O/2 I/O/Z	 AD14
P4.7	44	Вход-выход "порт 4, 7 разряд" Вход-выход "адрес-данные, 15 разряд"	I/O/2 I/O/Z	 AD15
P5.0	37	Вход-выход "порт 5, 0 разряд" Выход "разрешение записи адреса" Выход "адрес действителен"	I/O O O	 ALE ADV#
P5.1	35	Вход-выход "порт 5, 1 разряд" Выход "чтение команды"	I/O O	 INST

Продолжение таблицы 2.1

1	2	3	4	5
P5.2	42	Вход-выход "порт 5, 2 разряд" Выход "запись" Выход "запись младшего байта"	I/O O O	 WR# WRL#
P5.3	39	Вход-выход "порт 5, 3 разряд" Выход "чтение"	I/O O	 RD#
P5.4	33	Вход-выход "порт 5, 4 разряд" Вход "внутрисхемная эмуляция"	I/O I	 ONCE#
P5.5	40	Вход-выход "порт 5, 5 разряд" Выход "выбор старшего байта" Выход "запись старшего байта"	I/O O O	 BHE# WRH#
P5.6	34	Вход-выход "порт 5, 6 разряд" Вход "готовность"	I/O I	 READY
P5.7	43	Вход-выход "порт 5, 7 разряд" Вход "разрядность шины данных"	I/O I	 BW
P6.0	81	Выход "порт 6, 0 разряд" Выход "фаза 1 инверсная"	O O	 WG1#
P6.1	80	Выход "порт 6, 1 разряд" Выход "фаза 1"	O O	 WG1
P6.2	79	Выход "порт 6, 2 разряд" Выход "фаза 2 инверсная"	O O	 WG2#
P6.3	77	Выход "порт 6, 3 разряд" Выход "фаза 2"	O O	 WG2
P6.4	76	Выход "порт 6, 4 разряд" Выход "фаза 3 инверсная"	O O	 WG3#
P6.5	75	Выход "порт 6, 5 разряд" Выход "фаза 3"	O O	 WG3
P6.6	25	Выход "порт 6, 6 разряд" Выход "ШИМ 0"	O O	 PWM0
P6.7	24	Выход "порт 6, 7 разряд" Выход "ШИМ 1"	O O	 PWM1
EXTINT	32	Вход "внешнее прерывание"	I	
NMI	69	Вход "немаскируемое прерывание"	I	
EA#	71	Вход "обращение к внешней памяти"	I	
CLKOUT	48	Выход "системный тактовый сигнал"	O	
RESET#	68	Вход-выход "сброс"	I/O	
BQ1	30	Вывод для подключения кварцевого резонатора / вход тактового сигнала	–	
BQ2	29	Вывод для подключения кварцевого резонатора	–	
VPP	38	Вывод "напряжение программирования" * Вход "возврат из режима пониженного потребления"	– I	
#VCC1	46,74	Выводы питания цифровой части микросхемы	–	
#0V	31, 36, 53, 57, 72, 73, 78, 84	Общие выводы цифровой части микросхемы	–	

Продолжение таблицы 2.1

1	2	3	4	5
∩VCC2	3	Вывод питания аналоговой части микросхемы	–	
∩0V	2	Общий вывод аналоговой части микросхемы	–	

Примечания

1 Выводы 11, 12, 14, 15, 19, 20, 26, 27, 28, 41, 55, 56, 58, 67, 70 не задействованы, могут быть подключены к общим выводам цифровой части микросхемы.

2 Выводы 44, 45, 47, 49-52, 54, 59-66 при использовании их в качестве входов/выходов портов 3 и 4 функционируют или как комплементарные, или как выводы с открытым стоком. При использовании в качестве шины адреса/данных (AD0-AD15) они функционируют как входы-выходы с тремя состояниями.

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

_____

* Функция только для микросхем 1874ВЕ86Т.

В таблице 2.2 приведены основные функциональные параметры, характеризующие ИМС с точки зрения применения в аппаратуре.

Таблица 2.2 – Основные функциональные параметры ИМС

Наименование параметра, единица измерения	Значение параметра
Объем встроенной программной памяти	16К байт (вариант с OTPROM) 0 (вариант без ПЗУ)
Тактовая частота, МГц	от 8 до 16
Регистровое ОЗУ, бит	488 × 8
Источники прерываний/векторы вызова подпрограмм	14/27
Умножение 16×16 бит Деление 32/16 бит	1,75 мкс 3,0 мкс
Режимы энергосбережения	2
Порты ввода/вывода	5 × 8
16-разрядный сторожевой таймер	1
Реконфигурируемая 8/16 разрядная шина данных	да
Процессор событий (EPA) высокоскоростные модули захвата/сравнения высокоскоростные модули сравнения	 4 4
Сервер периферийных транзакций (PTS)	да
3-х фазный генератор сигналов	да
16-разрядные таймеры	2
Блоки ШИМ	2
АЦП 10- или 8-разрядный/13 каналов	да
Количество команд	112

Особенности архитектуры:

- быстродействующая архитектура типа "регистр-регистр";
- регистровое ОЗУ емкостью до 488 байт;
- однократно программируемое ПЗУ емкостью 16К байт (или вариант без встроенного ПЗУ);
- динамически конфигурируемая разрядность шины данных - 8 или 16 бит;
- протокол захваты шины HOLD/HLDA;
- 8-канальная матрица процессоров событий (EPA);
- сервер периферийных транзакций (PTS);

- два 16-разрядных таймера/счетчика с предделителями и режимами квадратурного счета;
- 16-разрядный сторожевой таймер;
- до 13 каналов аналого-цифрового преобразователя с разрешением в 8 или 10 бит;
- 3-фазный генератор сигналов с широтно-импульсной модуляцией;
- 2-канальный блок ШИМ;
- пять 8-разрядных портов ввода/вывода, один 5-разрядный и один 8-разрядный порты ввода;
- режимы холостого хода (IDLE) и пониженного энергопотребления (POWERDOWN).

Центральное процессорное устройство (ЦПУ)

Микроконтроллер построен по Фон Неймановской архитектуре. Он имеет внешнюю системную магистраль для обмена данными с внешней памятью и дополнительными периферийными устройствами. Система команд микроконтроллера поддерживает широкий набор методов адресации, в т.ч. битовую адресацию.

Архитектура микроконтроллеров, называемая архитектурой типа "регистр-регистр", обеспечивает достижение высокой производительности и упрощает работу с периферией. Главным отличительным признаком архитектуры микроконтроллеров 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. ЦПУ микроконтроллера реализует инструкции с байтами, словами, несколько инструкций с 32-разрядными операндами, а также команды перехода по битам.

Микроконтроллер имеет регистровую организацию. Память данных обменивается данными с процессорным ядром по внутренней магистрали микроконтроллера. Поэтому обмен данными с внутренней памятью данных осуществляется быстрее, чем с внешней.

ЦПУ осуществляет обмен информацией с внутренней и (или) внешней памятью программ и данных с помощью встроенного контроллера памяти, а с периферийными устройствами – через регистры специальных функций и имеет общее адресное пространство размером 64К байт.

Система команд обеспечивает обработку данных длиной байт или 16-разрядное слово. Часть команд поддерживает инструкции над двойными словами. В том числе, с такими же числами осуществляются и инструкции умножения и деления.

Встроенный контроллер памяти работает с 8- и 16-разрядной внешней шиной. Обеспечена возможность работы с внешней медленной памятью, с автоматическим формированием циклов ожидания.

Все внутренние регистры микроконтроллера могут выполнять функции «аккумулятора», что обеспечивает более эффективное программирование и повышение реальной производительности.

Несколько определенных ячеек памяти программ используются для хранения байта конфигурации кристалла (векторов подпрограмм обработки прерываний), а также содержат сигнатурный код доступа встроенного ПЗУ для считывания, что обеспечивает защиту программных продуктов от несанкционированного доступа.

Структура внешней шины программ/данных обеспечивает режим непосредственного доступа к памяти и многопроцессорный режим, а четырехбайтная очередь инструкций – оптимальный режим выборки команд без потери производительности.

Слово состояния программы содержит информацию о текущем состоянии работы. В нем имеется два байта: старший байт текущего слова состояния и младший байт масок прерывания.

Реализована развитая система прерываний. Система прерываний микроконтроллера поддерживается двумя устройствами: программируемым контроллером прерываний (IC) и сервером периферийных транзакций (PTS), обслуживает 14 источников прерываний (14 уровней приоритета для IC и 12 уровней для PTS) и формирует 18 адрес - векторов подпрограмм обслуживания через IC и 15 через PTS. Это существенно повышает эффективность применения микроконтроллера для управления реальными объектами.

Кроме того, микроконтроллер содержит встроенные периферийные блоки – многоканальный аналого-цифровой преобразователь, широтно-импульсные модуляторы, таймеры и т. п., которые могут функционировать автономно, практически не загружая центральный процессор. Характерные особенности контроллера шины – программное задание числа тактов ожидания, конфигурируемая шина данных (8-ми или 16-разрядная), а также протокол HOLD/HLDA для многопроцессорных систем.

На частоте 16 МГц ЦПУ выполняет 2 млн. операций/с при выполнении элементарных инструкций над знаковыми/беззнаковыми данными длиной 1 или 2 байт. Для этих чисел имеются также и инструкции умножения и деления (быстродействие: 580 тысяч умножений/с, 330 тысяч делений/с).

Память и внешняя шина ЦПУ имеет одно адресное пространство размером 64К байт, в котором находятся регистры общего назначения (488 байт), регистры специального назначения, встроенная программная память (1874ВЕ86Т), внешняя память для программы и данных. Вариант со встроенным ПЗУ имеет объем памяти 16К байт и программную защиту от несанкционированного доступа.

Интерфейс и встроенные функциональные устройства

Микроконтроллер позволяет реализовать эффективное управление различного рода устройствами, в том числе и электродвигателями. В него встроены периферийные устройства, оптимизированные для управления трехфазными индукционными двигателями переменного тока, бесколлекторными трехфазными двигателями постоянного тока, четырёхфазными шаговыми двигателями.

Периферийные устройства управляются и опрашиваются через специальные функциональные регистры SFR, которые могут быть доступны косвенно или окнами и как CPU-аккумуляторы.

Генератор формы сигналов

В микроконтроллере имеется уникальное периферийное устройство, называемое генератором формы сигналов (Waveform Generator – WFG) (ГФС), которое используется для формирования трехфазных широтно-модулированных сигналов.

Блок ГФС генерирует три не перекрывающиеся во времени комплементарные последовательности импульсов с управляемой длительностью (PWM сигналы) с разрешением в 125 нс (при запуске по фронту) или 250 нс (для «центрированной» оценки). Особенностями WFG-генератора являются программируемые частота, время рабочего цикла и время блокировки. Для каждой фазы блок ГФС имеет по два выхода с повышенной нагрузочной способностью, при этом полярность выходных сигналов может программироваться. Схема защиты позволяет отключать все выходы генератора WFG одновременно в ответ на внешние события.

Каждый из выходов программируется независимо.

Блок ГФС позволяет обеспечить высокоэффективное управление 3-фазными двигателями переменного тока, а также бесколлекторными двигателями постоянного тока.

Блок широтно-импульсных модуляторов (ШИМ)

Изделие содержит два аппаратно реализованных широтно-импульсных модулятора, для которых используется общий программируемый источник опорной частоты, но длительность рабочего цикла для каждого выхода программируется независимо.

Встроенный ШИМ предназначен для генерации широтно-модулированного сигнала на выходе микросхемы без участия процессора.

На таймер подается системный синхросигнал (возможно через предделитель). В регистр периода процессор записывает число, соответствующее периоду ШИМ. При совпадении содержимого этого регистра и таймера, последний сбрасывается в нулевое состояние и на соответствующем выходе микросхемы формируется положительный перепад. В регистре длительности процессор записывает число, соответствующее длительности импульса. При совпадении его содержимого с содержимым таймера на выходе микросхемы формируется отрицательный перепад.

Система ввода/вывода

Микроконтроллер имеет 53 линии ввода/вывода, которые могут быть использованы встроенными периферийными устройствами.

Микроконтроллер имеет шесть 8-разрядных портов ввода/вывода и один 5-разрядный, часть из них имеет программно управляемые альтернативные функции для работы с встроенными устройствами.

Процессор событий (EPA) (event processor array)

Блок аналогичен ранее применявшимся устройствам HSIO. EPA имеет более простую архитектуру, чем HSIO, обладая при этом лучшей разрешающей способностью. В HSIO все входные каналы имеют общую память (7-уровневое FIFO), в которой запоминаются временные отметки, соответствующие событиям на входах. То же касается выходных линий HSIO – все они имеют общую память (8 ячеек), в которую процессор записывает команды для всех выходных каналов HSIO. Поэтому за один такт HSIO может обработать только один входной и один выходной канал. В EPA каждый канал имеет свой собственный буфер, а выдача и прием сигналов производятся одновременно по всем каналам. Поэтому разрешающая способность EPA выше, чем у HSIO, в 4 раза. Кроме того, EPA – более гибкий функциональный блок: каждый его канал может служить и входом, и выходом, тогда как HSIO имеет 4 выходных, 2 входных и 2 двунаправленных линии. EPA выполняет функции ввода/вывода, связанные с таймерами 1, 2.

EPA используется для работы с событиями. Матрица процессора событий состоит из 4 модулей захвата/сравнения и 4 модулей сравнения. Блок EPA имеет разрешение в 125 нс. Эти модули предназначены для быстрой генерации события на выходе микросхемы либо быстрой реакции на событие на её входе без участия процессора.

Таймер модуля считает импульсы, поступающие с постоянной частотой, соответствующей сигналу синхронизации (возможно деленной на предделителе). В регистр времени события предварительно записывается по команде определенное число. Компаратор сравнивает каждый момент времени содержимое таймера и содержимое регистра времени события. В момент, когда их содержимое становится равным друг другу, происходит заданное событие на соответствующем выходе микросхемы, и выставляется запрос на прерывание. Таким образом, можно запрограммировать определённое выходное событие в заданное время и осуществить его без участия процессора. Этот режим используется в частности для генерации ШИМ.

Периферийный сервер (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.

Блок PTS предназначен для обеспечения механизма обработки 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 ввода-вывода) 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 поддерживает обработку запросов на прерывание на микропрограммном уровне, не требуя вмешательства процессора. Специальный режим работы PTS обеспечивает поддержку функций последовательного ввода/вывода (SIO).

Таймеры-счетчики

Микроконтроллер имеет два независимых 16-разрядных многофункциональных таймера-счетчика. В одном из режимов 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.

Сторожевой таймер (Watchdog timer – WDT)

Сторожевой таймер 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.

Аналого-цифровой преобразователь (АЦП)

В составе микроконтроллера – встроенный 8-, 10-разрядный 13-входовый АЦП с диапазоном входного напряжения от (#0V – 0,5) В до (∩VCC2 + 0,5) В. Модуль АЦП предназначен для преобразования входной аналоговой информации в цифровую и передачи ее в процессор для дальнейшей обработки.

Время преобразования (измерения) составляет:

- 10,0 мкс – в режиме 10-разрядного преобразования;
- 7,0 мкс – в режиме 8-разрядного преобразования.

Схема выборки/хранения и отдельные входы опорного напряжения и аналоговой земли обеспечивают повышение точности преобразования.

Преобразование информации на том или ином канале инициализируется соответствующей командой.

Преобразование начинается выбором требуемого канала, который осуществляется записью номера канала в специальный служебный регистр. Аналоговый мультиплексор пропускает на выход сигнал выбранного канала. После этого подается сигнал фиксации на устройство выборки-хранения. Оно фиксирует уровень выбранного аналогового сигнала на весь период преобразования. Затем подается сигнал начала преобразования. АЦП представляет собой АЦП последовательных приближений. После окончания преобразования АЦП выдает сигнал готовности, по которому его выходная информация записывается в буферный регистр, с которого она затем может быть считана процессором по соответствующей команде. При этом вырабатывается запрос на прерывание.

Энергосберегающие режимы работы

Реализованы режимы работы с пониженным энергопотреблением:

- режим IDLE (HALT), когда работают только встроенные функциональные устройства, а микроконтроллер находится в режиме ожидания разрешенного прерывания от внешнего или внутреннего устройства;
- POWERDOWN (STOP), когда все устройства выключены до появления сигнала внешнего прерывания. В этом случае ток потребления – всего несколько мкА.

Система команд

Система команд включает в себя полный набор арифметических, логических команд, а также инструкций управления и перехода над 8-битовыми операндами и над 16-битными словами с различными способами адресации. Типы данных двойное слово (DOUBLE-WORD) и длинное целое (LONG) (32 битные) поддерживаются для результатов 16×16 умножения и для делимого при делении 32 на 16, а также для сдвиговых инструкций и 32-разрядном сравнении. Остальные инструкции с 32-разрядными операндами могут быть выполнены комбинацией инструкций по 16 бит. Общее число команд –112.

Обеспечивается защита против невыполнимых кодов операций.

2.2 Электрические параметры микросхем

Электрические параметры микросхем при приёмке и поставке и предельно-допустимые значения параметров приведены в таблицах 2.3 и 2.4 соответственно.

Номинальное значение напряжения питания микросхем плюс 5,0 В. Допустимое отклонение напряжения питания ±10 %. Амплитуда пульсаций напряжения питания не более 50 мВ.

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, плюс 85 ºС;

повышенной предельной температуры, плюс 150 ºС.

Таблица 2.3 – Электрические параметры при приёмке и поставке в диапазоне рабочих температур окружающей среды от минус 60 ºС до плюс 85 ºС

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, ºС
		не менее	не более	
1	2	3	4	5
1 Выходное напряжение высокого уровня по выводам P2 - P6, В I_{OH} = –0,2 мА I_{OH} = –3,2 мА I_{OH} = –7,0 мА	U_{OH}	 U_{CC1} – 0,3 U_{CC1} – 0,7 U_{CC1} – 1,5	-	–60 ± 3 25 ± 10 85 ± 3
2 Выходное напряжение низкого уровня по выводам P2, P5, P6.6, P6.7, CLKOUT, В I_{OL} = 0,2 мА I_{OL} = 3,2 мА I_{OL} = 7,0 мА	U_{OL1}	-	 0,3 0,45 1,5	
3 Выходное напряжение низкого уровня по выводам P3, P4, В I_{OL} = 15,0 мА	U_{OL2}	-	1,0	
4 Выходное напряжение низкого уровня по выводам P6.0 - P6.5, В I_{OL} = 10,0 мА	U_{OL3}	-	0,45	

Продолжение таблицы 2.3

1		2	3	4	5
5 Токи утечки по входам P2.0 - P2.7, P3.0 - P3.7, P4.0 - P4.7, P5.0 - P5.7, мкА $0 < U_I < (U_{CC1} - 0,3)$ В		I_{LI}	–1	1	25 ± 10 85 ± 3
6 Токи утечки на входах P0.0 - P0.7, P1.0 - P1.4, мкА $0 < U_I < U_{CC2}$		I_{LI1}	–1	1	
7 Динамический ток потребления от источника U_{CC1}, мА $U_{PP} = U_{CC1} = U_{CC2} = 5,5$ В, $f_{CI} = 16$ МГц		I_{OCC1}	-	75	
8 Динамический ток потребления от источника U_{CC2}, мА $U_{PP} = U_{CC1} = U_{CC2} = 5,5$ В, $f_{CI} = 16$ МГц		I_{OCC2}	-	2,0	
9 Ток потребления в режиме хранения, мА $U_{PP} = U_{CC1} = U_{CC2} = 5,5$ В, $f_{CI} = 16$ МГц		I_{CCS}	-	30	–60 ± 3 25 ± 10 85 ± 3
10 Функциональный контроль $U_{CC1} = (4,5; 5,5)$ В, $U_{CC2} = (4,0; 5,5)$ В, $f_{CI} = 8; 16$ МГц		ФК	-	-	
11 Время переключения сигнала на выходе CLKOUT, нс	время нарастания	t_r	-	22	
	время спада	t_f			

Примечания

1 Нормы статических параметров приведены при $U_{CC1} = (4,5; 5,5)$ В, $U_{CC2} = (4,0; 5,0)$ В.

2 Функциональный контроль (ФК) осуществляется в соответствии с системой команд микроконтроллера по тестам, приведенным в «Программе и методике функционального контроля» КФДЛ.431295.021ПМ.

3 Проверку динамических параметров не проводят, так как функциональный контроль проводят на максимальной рабочей частоте 16 МГц.

4 Параметр по пункту 7 проверяется при низком уровне сигнала на входе RESET#.

Таблица 2.4 – Значения предельно-допустимых электрических режимов эксплуатации в диапазоне рабочих температур

Наименование параметра режима, единица измерения	Буквенное обозначение	Предельно-допустимый режим		Предельный режим	
		не менее	не более	не менее	не более
1	2	3	4	5	6
1 Напряжение питания цифровой части ИМС, В	U_{CC1}	4,5	5,5	–0,3	7,0
2 Напряжение питания аналоговой части ИМС, В	U_{CC2}	4,0	5,5	–0,3	7,0
3 Входное напряжение низкого уровня, В	U_{IL}	–0,3	$0,3U_{CC1}$	–0,5	-
4 Входное напряжение высокого уровня, В	U_{IH}	$0,7U_{CC1}$	$U_{CC1}+0,3$	-	$U_{CC1}+0,5$
5 Напряжение программирования внутреннего ЭППЗУ, В *	U_{PP}	12,25	12,75	–0,3	13,0
6 Емкость нагрузки, пФ	C_L	-	100	-	200
7 Частота следования импульсов тактового сигнала, МГц	f_{CI}	8	16	-	-
8 Длительность фронтов тактового сигнала, нс	t_{LH}, t_{HL}	-	5	-	-
* Для МК 1874ВЕ86Т.					

3 Архитектура изделий

16-разрядные микроконтроллеры 1874ВЕ86Т, 1874ВЕ16Т 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 1874ВЕ86Т, 1874ВЕ16Т разработаны для использования в быстродействующих системах управления в режиме реального времени.

3.1 Особенности архитектуры

Структурная схема ИМС приведена на рисунке 2.1 раздела 2.

Микроконтроллеры 1874ВЕх6Т являются полными функциональными аналогами изделий 8хС196МС, 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.

По общепринятому среди разработчиков аппаратуры мнению, однокристальные микроконтроллеры семейства MCS-196 фирмы Intel являются своего рода индустриальным стандартом для 16-разрядных встроенных систем управления. 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 микроконтроллера идеально ориентирована для создания модификаций систем для реализации функций управления и вычисления в реальном режиме времени под конкретные приложения. Этому способствует высокая производительность микроконтроллера, развитая система встроенных функциональных блоков, внутренняя программная память объемом 8К и мощная система команд. Может служить элементной базой для систем управления различной аппаратурой, в том числе может применяться в спецаппаратуре, бытовой технике, теле-, аудио- и видеоаппаратуре, телефонной и радиосвязи, компьютерной технике, автомобильной и силовой электронике, особенно в части эффективного управления моторами и многих других отраслях.

Ключевые особенности микроконтроллеров семейства:

- 3-фазный генератор сигналов широтно-импульсной модуляции;
- функционирование на частотах до 16 МГц;
- быстродействующая архитектура типа "регистр-регистр";
- регистровое ОЗУ емкостью до 488 байт;
- однократно программируемое ПЗУ емкостью 16К байт;
- динамически конфигурируемая разрядность шины данных – 8 или 16 бит;
- протокол захвата шины HOLD/HLDA;
- 8-канальная матрица процессоров событий;
- два 16-разрядных таймера/счетчика с предделителями и режимами квадратурного счета;
- 16-разрядный сторожевой таймер;
- до 13 каналов аналого-цифрового преобразователя с разрешением в 8 или 10 бит;
- шесть 8-разрядных портов ввода/вывода;
- режимы холостого хода (IDLE) и пониженного энергопотребления (POWERDOWN);
- сервер периферийных транзакций (PTS);
- двухканальный модуль сигналов ШИМ.

3.2 Описание функционирования микроконтроллера

Центральный процессор управляется контроллером микрокода, который выдает команды RALU на исполнение команд, используя байты, слова или двойные слова из 256 байтов младшего файла регистров или через «окно», которое непосредственно имеет доступ к старшему файлу регистров. Команды центрального процессора сдвигаются из 4-байтовой очереди команд. Контроллер микрокода расшифровывает инструкции и затем выдает последовательность микрокоманд, управляющих выполнением заданной инструкции.

Файл регистров

Файл регистров разделен на старший и младший файл. В младшем файле регистров младшие 24 байта отведены регистрам специальных функций центрального процессора (SFRs) и указателю стека, в то время как остальные доступны как регистры общего назначения (РОН). Старший файл регистров содержит только регистры общего назначения. К РОН можно обратиться как к байтам, словам или двойным словам. РАЛУ доступны старшие и младшие файлы регистров отдельно. Младший файл регистров всегда доступен с прямой адресацией. Старший файл регистров доступен с прямой адресацией только при использовании режима «окна».

Регистровое арифметико-логическое устройство (РАЛУ)

РАЛУ содержит контроллер микрокода, арифметико-логическое устройство (АЛУ), главный программный счетчик (PC), слово состояния процессора (PSW) и служебные регистры. Регистры РАЛУ: регистр инструкций, регистр констант, регистр выбора бит, счетчик циклов и три временных регистра (старшее слово, младшее слово и регистр второго операнда).

PSW содержит один бит (PSW.1), который глобально разрешает или запрещает обслуживание всех маскируемых прерываний, один бит (PSW.2), который разрешает или запрещает периферийный сервер (PTS), и шесть булевых флагов, которые отражают состояние программы пользователя.

Все регистры, кроме 3-разрядного регистра выбора бита и 6-разрядного счетчика циклов, являются 16- или 17-разрядными (16 бит плюс расширение знака). Некоторые из этих регистров могут уменьшить загруженность АЛУ, выполняя простые действия.

РАЛУ использует старшие и младшие регистры слов совместно для 32-разрядных инструкций и как временные регистры для большинства инструкций. Эти регистры имеют собственную логику сдвига и используются для операций, которые требуют логических сдвигов, включая нормализацию, умножение и деление. 6-разрядный счетчик циклов считает повторяющиеся сдвиги. Регистр второго операнда хранит второй операнд для инструкций с двумя операндами, включая множитель в течение умножения, и делитель в течение действия деления. При вычитании выходы этого регистра дополняются перед пересылкой в АЛУ.

РАЛУ ускоряет вычисления, храня константы (например, 0, 1 и 2) в регистре констант так, чтобы они были доступны при дополнении, инкременте или декременте байт или слов. Кроме того, регистр констант генерирует одноразрядные маски на базе регистра битового выбора для инструкций проверки бит.

Выполнение инструкций

РАЛУ исполняет большинство вычислений автономно, без использования аккумулятора. Вместо этого РАЛУ работает непосредственно с младшим файлом регистров, которые фактически являются 256 «аккумуляторами». Поскольку данные не проходят через единственный аккумулятор, команды микроконтроллера выполняются быстрее и более эффективно.

Формат команд

Микроконтроллеры сочетают большой набор регистров общего назначения с форматом трехоперандных инструкций. Этот формат позволяет в одной команде определить два регистра источника и отдельно регистр назначения. Например, следующая команда умножает две 16-битные переменные и сохраняет 32-битный результат в третьей переменной.

```
MULRESULT, FACTOR_1, FACTOR_2; умножает FACTOR_1 и FACTOR_2
                              ; и помещает в RESULT
                              ; (RESULT) ← (FACTOR_1 × FACTOR_2)
```

Интерфейс памяти

РАЛУ связывается со всем объемом памяти, кроме регистрового файла и периферийных SFRs, через контроллер памяти. (РАЛУ оперирует со старшим файлом регистров через контроллер памяти кроме случаев, когда используются «окна»; см. раздел 5 “Распределение памяти”). Контроллер памяти содержит очередь команд, подчинённый программный счётчик, регистры адресов и данных и контроллер шины.

Контроллер шины управляет шиной памяти, которая состоит из внутренней шины памяти и внешней шины адреса/данных. Контроллер шины получает запросы доступа к

памяти или от РАЛУ, или от очереди команд; запросы очереди всегда имеют приоритет. Эта очередь прозрачна к РАЛУ и программному обеспечению.

При использовании логического анализатора для отладки программы необходимо помнить, что инструкции предзагружены в очередь команд и не обязательно выполняются немедленно после того, как они загружены.

Обслуживание прерываний

Гибкая система обработки прерываний микроконтроллера имеет два главных компонента: программируемый контроллер прерываний (КП) и периферийный сервер (PTS). Программируемый контроллер прерываний имеет аппаратную схему приоритетов PTS, которая может быть изменена программным обеспечением. Прерывания, которые выполняются контроллером прерывания, обслуживаются сервисными программами пользователя. Периферийный сервер транзакций (PTS) – микропрограммный аппаратный процессор, обеспечивающий быстрое обслуживание прерываний. Большинство прерываний (кроме NMI, TRAP и неисполняемого кода) можно запрограммировать на обслуживание их PTS вместо контроллера прерываний. PTS может передавать байты или слова индивидуально либо блоками между любыми ячейками памяти, управлять многократными аналого-цифровыми (A/D) преобразованиями и генерировать ШИМ сигналы. Микроконтроллеры 1874ВЕ86Т, 1874ВЕ16Т имеют дополнительные режимы, которые обеспечивают асинхронную или синхронную последовательную связь. PTS прерывания имеют более высокий приоритет, чем стандартные прерывания и могут временно приостанавливать программы обслуживания прерываний.

3.3 Синхронизация

На схему синхроимпульсов (рисунок 3.1) поступает входной тактовый сигнал по выводу BQ1, 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.

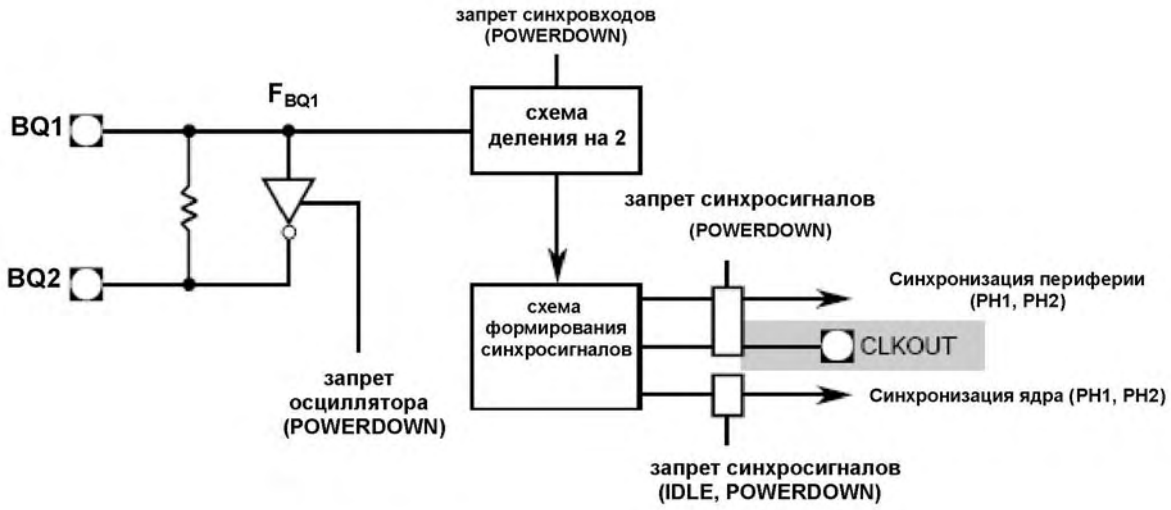


Рисунок 3.1 – Схема формирователя синхроимпульсов

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.

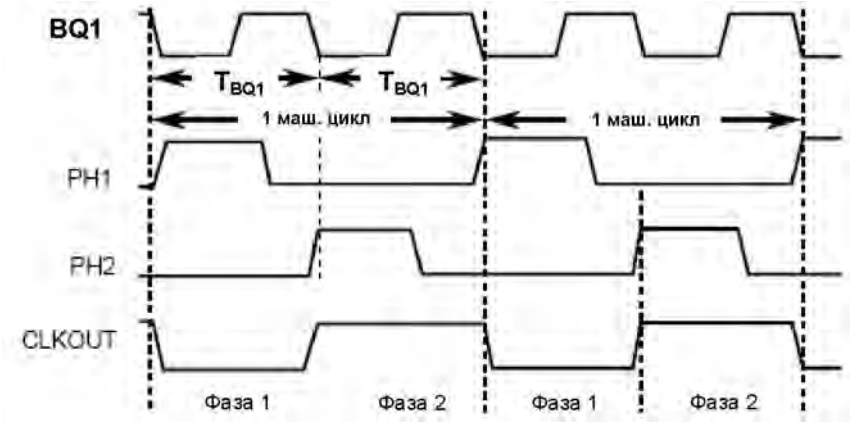


Рисунок 3.2 – Внутренние фазы синхроимпульсов

Комбинация периодов фазы 1 и фазы 2 внутреннего CLKOUT сигнала определяет основную единицу времени микроконтроллера, называемую «машинным циклом». В таблице 3.1 приведены параметры временной диаграммы (см. рисунок 3.2) на различных частотах.

Таблица 3.1 – Длительность машинного цикла при различной тактовой частоте

F_{BQ1}	Длительность машинного цикла
8 МГц	250 нс
12 МГц	167 нс
16 МГц	125 нс

Следующие формулы позволяют вычислять частоты PH1 и PH2 – формула (1), длительность машинного цикла - формула (2), период синхроимпульсов (T_{BQ1}) - (3).

$$PH1 \text{ (в МГц)} = F_{BQ1}/2 = PH2 \quad (1),$$

$$\text{длительность машинного цикла (в мкс)} = 2/ F_{BQ1} \quad (2),$$

$$T_{BQ1} = 1/F_{BQ1} \quad (3)$$

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

3.4 Встроенные периферийные устройства

Внутренние периферийные модули обеспечивают дополнительные функции для разнообразных специализированных применений. В данном подразделе приведен краткий обзор функциональности периферийных устройств, а в последующих разделах дается более подробное описание каждого устройства.

Порты ввода-вывода

Микроконтроллеры 1874ВЕ86Т, 1874ВЕ16Т имеют семь портов ввода-вывода (порты 0–6). Выводы портов мультиплексированы, чтобы служить стандартным вводом-выводом или обеспечивать специальные функции, связанные с периферийным устройством или компонентом системы. Если определённый сигнал специальной функции не используется, вывод может быть индивидуально сконфигурирован, чтобы служить стандартным для ввода-вывода. Порты 3 и 4 – исключения; они управляются на уровне порта, не на уровне вывода. Когда контроллер шины должен использовать шину адреса/данных, он осуществляет управление портами. Когда шина адреса/данных не используется, можно использовать порты для ввода-вывода.

Порт 0 – порт только для ввода, который является также аналоговым входом для АЦП.

Порт 1 – порт только для ввода, который обслуживает аналоговые входы для АЦП.

Порт 2 – стандартный двунаправленный порт ввода-вывода, который обслуживает выходы EPA и таймеров.

Порты 3, 4 и 5 – распределенные в памяти, двунаправленные порты ввода-вывода. Порты 3 и 4 служат внешней шиной адреса/данных, в то время как порт 5 обеспечивает сигналы шинного управления.

Порт 6 – стандартный порт только для вывода, который обслуживает выходы блока ШИМ и генератора формы сигналов. В разделе 7 “Порты ввода-вывода” описываются порты ввода-вывода более подробно.

Процессор событий (EPA) и таймеры/счетчики

Процессор событий выполняет быстродействующий ввод-вывод сигналов на базе таймеров/счетчиков. Во входном режиме EPA контролирует переключение входных сигналов. Когда происходит событие, EPA делает запись соответствующего значения из таймера. Это захват события. В режиме вывода EPA контролирует таймер до совпадения его значения с хранимым. При совпадении EPA запускает выходное событие, это может быть установка, очистка или переключение выходного сигнала. Это сравнение события. И захват, и сравнение событий могут вызвать прерывания, которые обслуживаются или контроллером прерываний, или периферийным сервером.

Таймер 1 и таймер 2 – это 16-разрядные реверсивные таймеры/счетчики, которые могут тактироваться внутренним или внешним сигналом. Каждый таймер/счетчик – называют таймером, если такт внутренний, и счетчиком, если такт внешний. Для дополнительной информации – см. раздел 10 «Процессор событий (EPA)».

Блок ШИМ

Сигналы каждого ШИМ канала – импульсы с переменной скважностью. Некоторые типы двигателей требуют ШИМ сигналы для эффективной работы. После фильтра ШИМ сигнал формируется постоянным уровнем, длительность которого может меняться с дискретным шагом (256 в рабочем цикле). В ШИМ период также программируем (8 бит). См. раздел 9 «Широтно-импульсный модулятор» для дополнительной информации.

Генератор формы сигнала (ГФС)

Генератор формы сигнала решает задачу формирования и синхронизации ШИМ сигналов. ГФС оптимизирован для применений в управлении 3-х фазными двигателями постоянного тока, 3-х фазными безколлекторными двигателями, 4-х фазными шаговыми двигателями. ГФС может формировать три независимых пары ШИМ сигналов, которые имеют общий период, так называемое «мёртвое» время и операционный режим. После программирования и инициализации генератор формы сигнала работает без вмешательства центрального процессора до тех пор, пока не программируется новая скважность. См. раздел 8 «Генератор формы сигнала» для дополнительной информации.

Аналого-цифровой преобразователь

Аналого-цифровой преобразователь (АЦП) преобразует аналоговое входное напряжение в цифровой эквивалент. Точность преобразования (8 или 10 битов), время выборки, время преобразования программируются. Преобразования могут быть выполнены с аналоговой землей и опорным напряжением, и результаты могут использоваться, чтобы вычислить ошибки нулевого смещения и усиления. Внутренняя схема компенсации нулевого смещения позволяет автоматически регулировать нулевое

смещение. АЦП также имеет режим порогового детектирования, который может использоваться, чтобы вызвать прерывания, когда программируемое пороговое напряжение превышено в любом направлении. Режим АЦП сканирования блоком PTS облегчает автоматизацию АЦП преобразования и сохранения его результатов. См. раздел 11 «Аналого-цифровой преобразователь (АЦП)» для дополнительной информации.

Сторожевой таймер

Сторожевой таймер – внутренний 16-разрядный таймер, который сбрасывает микроконтроллер, если программное обеспечение не в состоянии работать должным образом из-за сбоя или «зависания».

3.5 Специальные режимы работы

В дополнение к нормальному режиму включения, микроконтроллер работает в нескольких режимах специального назначения. Режимы холостого хода понижают мощность, когда микроконтроллер не активен. Режим внутрисхемной эмуляции (ONCE) электрически изолирует микроконтроллер от системы, а некоторые другие режимы обеспечивают варианты программирования для энергонезависимого запоминающего устройства.

Сокращение потребления мощности

В режиме холостого хода (IDLE) центральный процессор прекращает выполнять инструкции, но периферийные устройства остаются активными. Потребление мощности понижается приблизительно до 40 % нормального потребления. Аппаратный сброс или любое разрешенное прерывание выводят из режима холостого хода.

В режиме пониженного энергопотребления (POWERDOWN) все внутренние фазы установлены в логическом нуле и внутренний генератор отключен. Файл регистров и большинство периферийных устройств сохраняют их данные, если поддерживается #VCC1. Потребление мощности понижается до мкА диапазона.

Тестирование печатной платы микроконтроллера

Режим внутрисхемной эмуляции (ONCE) электрически изолирует микроконтроллер от системы. Устанавливая режим ONCE, можно проверить схемы печатной платы, когда микроконтроллер впаян в плату.

Программирование EPROM

Микроконтроллер, который имеет внутреннее OTPROM (1874BE86T), поддерживает несколько вариантов программирования:

- Программирование в подчиненном режиме позволяет программатору стираемой программируемой постоянной памяти программировать и проверять один или более микроконтроллеров.
- Автопрограммирование позволяет микроконтроллеру программировать себя кодом и данными, расположенными во внешнем запоминающем устройстве.
- Программирование в ходе выполнения программы позволяет программировать отдельные ячейки EPROM в течение нормального выполнения кода под управлением программного обеспечения.
- Режим выгрузки позволяет при необходимости сосчитать содержание энергонезависимого запоминающего устройства микроконтроллера.

4 Обзор программирования

В этом разделе приведено краткое описание набора инструкций 1874ВЕ86Т, 1874ВЕ16Т микроконтроллеров и рекомендации по разработке программ.

4.1 Краткий обзор набора команд

Система команд поддерживает различные типы операндов необходимые для разработки программ в системах управления (см. таблицу 4.1).

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 – 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 операнд 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 «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 (2^8–1). 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 ($–2^7$) до плюс 127 ($2^{7–1}$). 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”.

Преобразование операндов

Система команд поддерживает преобразования между разными типами операндов. LDBZE (load byte, zero extended), конвертирует BYTE в WORD. CLR (очистка) конвертирует WORD в DOUBLE WORD, очищая (записывая ноль) в старшее слово в DOUBLE WORD. LDBSE (load byte, sign extended) конвертирует SHORT INTEGER в INTEGER. EXT (расширение знака) конвертирует INTEGER в LONG INTEGER.

Условные переходы

Команды для сложения, вычитания и сравнения не различают операнды без знака (BYTE, WORD) и со знаком (SHORT INTEGER, INTEGER). Однако условные команды перехода позволяют рассматривать результаты этих действий как числа со знаком или без знака. Например, команда CMP (сравнение) используется, чтобы сравнить 16-битные числа со знаком или без знака. После инструкции сравнения можно использовать JH (переход, если больше) команду для операндов без знака или JGT (переход, если больше чем) команда для операндов со знаком.

Инструкции с плавающей точкой

Аппаратные средства непосредственно не поддерживают действия с переменными REAL (плавающей точкой). Эти действия поддерживаются только с использованием библиотеки с плавающей точкой. Производительность этих инструкций значительно повышена при использовании NORML инструкции и бита ST в слове состояния процессора (PSW). Инструкция NORML нормализует 32-битную переменную. Флаг ST может использоваться в комбинации с флагом переноса (C), чтобы достигнуть более высокой точности при округлении.

4.2 Режимы адресации

Система команд использует четыре основных режима адресации:

- прямой;
- непосредственный;
- косвенный (с или без автоинкремента);
- индексный (короткий, длинный или с нулевым индексом).

Указатель стека может использоваться с косвенной адресацией, чтобы получить доступ к вершине стека, и может также использоваться с коротко индексированной адресацией к данным в пределах стека. Нулевой регистр может использоваться с длинной индексной адресацией для доступа к любой ячейке запоминающего устройства.

Инструкция может содержать только одну косвенную или индексную ссылку; остальные операнды должны быть прямо адресованы.

Таблица 4.3 – Описание временных регистров

Рабочий регистр	Описание
AX	выровненный словом 16 битный регистр; AH - старший байт AX и AL - младший байт
BX	выровненный словом 16 битный регистр; BH - старший байт BX и BL - младший байт
CX	выровненный словом 16 битный регистр; CH - старший байт CX и CL - младший байт
DX	выровненный словом 16 битный регистр; DH - старший байт DX и DL - младший байт

Прямая адресация

Прямая адресация непосредственно обеспечивает доступ к ячейкам (256 байтов) в младшем файле регистров, не загружая контроллер памяти. «Окна» позволяют перераспределять другие секции запоминающего устройства, в младший файл регистров для прямого доступа. Нужно определять регистры как операнды в пределах инструкции. Адрес регистра должен соответствовать правилам выравнивания для типа операнда. В вычислении может принять участие до трех регистров в зависимости от инструкции. Следующие инструкции используют прямую адресацию:

```
ADD     AX, BX, CX  ; AX ← BX + CX
ADDB    AL, BL, CL  ; AL ← BL + CL
MULB    AX, BL      ; AX ← AX × BL
INCB    CL          ; CL ← CL + 1
```

Непосредственная адресация

Непосредственный режим адресации определяет одну непосредственную величину как операнд в инструкции. Непосредственное значение помечается символом (#). Инструкция может содержать только одну непосредственную величину. Остальные операнды должны быть прямо адресованы. Следующие инструкции используют непосредственную адресацию:

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

Косвенная адресация

Косвенный режим адресации обеспечивает доступ к операнду, получая его адрес из регистра WORD в младшем файле регистров. Регистр, содержащий косвенный адрес, помечается квадратными скобки ([]). Косвенный адрес может обратиться к любой ячейке в пределах памяти, включая файл регистров. Регистр, который содержит косвенный адрес, должен быть выровнен словом, и косвенный адрес должен соответствовать правилам для типа операнда. Инструкция может содержать только одну косвенную ссылку, остальные операнды должны быть прямо адресованы. Следующие инструкции используют косвенную адресацию:

```
LD      AX, [BX]    ; AX ← MEM_WORD(BX)
```

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

Косвенная адресация с автоинкрементом

Можно автоматически увеличивать косвенный адрес после текущего доступа к памяти. При определении автоинкремента в конце косвенной ссылки добавляется символ плюс (+). В этом случае инструкция автоматически увеличивает косвенный адрес (на 1, если регистр результата 8-битный или на два, если это 16-битный регистр). Следующие инструкции используют косвенную адресацию с автоинкрементом:

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

Косвенная адресация с указателем стека

Вы можете также использовать косвенную адресацию, чтобы получить доступ к вершине стека, используя указатель стека как регистр WORD в косвенной ссылке. Следующая инструкция использует косвенную адресацию с указателем стека:

```
PUSH [SP]    ; duplicate top of stack; SP ← SP + 2
```

Индексная адресация

Индексная адресация вычисляет адрес, добавляя поле смещения к основному адресу. Есть три разновидности индексной адресации: короткая индексная, длинная индексная и адресация с нулевым индексом. И короткая, и длинная индексные адресации используются, чтобы получить доступ к определенному элементу в пределах структуры. Короткая индексная адресация может получить доступ к 255 байтам, длинная индексная адресация может получить доступ к 65 535 байтам, адресация с нулевым индексом может получить доступ к одному байту. Инструкция может содержать только одну индексную ссылку, остальные операнды должны быть прямо адресованы.

Короткая индексная адресация

В короткой индексной инструкции смещение определяется как 8-битная константа и основной адрес как косвенный адрес регистра (WORD). Следующие инструкции используют короткую индексную адресацию.

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

Инструкция LD AX, 12H [BX] загружает AX содержимым ячейки по адресу BX+12H. То есть инструкция добавляет BX+12H к содержимому [BX] , затем загружает AX содержимым. Например, если [BX] содержит 1000H, то AX загружается содержимым ячейки 1012H. Короткая индексная адресация обычно используется для доступа, когда [BX] содержит базовый адрес и константа (12H в этом примере) – смещение определенного элемента в структуре.

Можно также использовать указатель стека в короткой индексной инструкции и получить доступ к ячейке в пределах стека как показано в следующей инструкции.

```
LD AX, 2 [SP]
```

Длинная индексная адресация

В длинной индексной инструкции основной адрес определяется как 16-битная переменная и смещение как косвенный адрес регистра (WORD). Следующие инструкции используют длинную индексную адресацию.

```
LD AX, TABLE [BX]; AX ← MEM_WORD (TABLE + BX)
AND AX, BX, TABLE [CX]; AX ← BX AND MEM_WORD (TABLE + CX)
```

ST AX, TABLE [BX]; MEM_WORD (TABLE + BX) ← AX

ADDB AL, LOOKUP [CX]; AL ← BL + MEM_BYTE (LOOKUP + CX)

Инструкция LD AX, TABLE [BX] загружает AX содержимым ячейки памяти по адресу TABLE+BX. То есть инструкция добавляет содержание BX к константе TABLE (основной адрес), затем загружает AX содержанием результирующего адреса. Например, если TABLE равняется 4000H и BX содержит 12H, то AX загружается содержимым ячейки 4012H. Длинная индексная адресация обычно используется для обращения к элементам в таблице, где константа TABLE является основным адресом структуры, и BX – масштабированное смещение (n × размер элемента, в байтах) в структуре.

Адресация с нулевым адресом

В команде с нулевым индексом определяется адрес как 16-битная переменная, смещение – ноль, и можно записать это одним из трех способов: [0], [ZERO_REG] или без указания. Каждая из следующих инструкций загружает AX содержимым переменной THISVAR.

LD AX, THISVAR [0]

LD AX, THISVAR [ZERO_REG]

LD AX, THISVAR

Следующие инструкции также используют адресацию с нулевым индексом:

ADD AX, 1234H [ZERO_REG]; AX ← AX + MEM_WORD (1234H)

POP 5678H [ZERO_REG]; MEM_WORD (5678H) ← MEM_WORD (SP); SP ← SP + 2

4.3 Выборы режима адресации ассемблера

Ассемблер упрощает выбор режимов адресации. Рекомендуется использовать это везде, где возможно.

Прямая адресация

Ассемблер выбирает между прямой адресацией и адресацией с нулевым индексом в зависимости от местоположения операнда в памяти. Необходимо обратиться к операнду 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.

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 «Си», и они обеспечивают совместимость между этими средами.

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).

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.

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

Ниже описывается адресное пространство, его основные разделы, методику работы с окнами, а также порядок обращения к старшему файлу регистров и периферийным SFR с использованием команд с прямым методом адресации.

5.1 Карта памяти микроконтроллеров

В таблице 5.1 приведена карта памяти МК 1874ВЕх6Т.

Внешние устройства (память или устройства ввода-вывода)

Несколько областей памяти отведены для внешних устройств (см. таблицу 5.1). Данные могут быть сохранены в любой части этой памяти. Эти области могут также использоваться для связи с внешней периферией, подключенной к шине адреса/данных.

Таблица 5.1 – Карта памяти МК 1874ВЕх6Т

Шестнадцати-ричный диапазон адресов	Описание	Способы адресации
FFFF 6000	Внешнее устройство (память или устройство ввода-вывода), подключенное к шине адреса/данных	Косвенный или индексный
5FFF 2080	Память программ (внутренняя энергонезависимая или внешняя память), см. примечание 1.	Косвенный или индексный
207F 2000	Память специального назначения (внутренняя энергонезависимая или внешняя память)	Косвенный или индексный
1FFF 1FE0	Картированные в памяти SFRs	Косвенный или индексный
1FDF 1F00	Периферийные SFRs	Косвенный, индексный или прямой через окно
1EFF 0200	Внешние устройства (память или ввод - вывод), подключенные к шине адреса/данных (будущее SFR расширение; см. примечание 2)	Косвенный или индексный
01FF 0100	Старший файл регистров (оперативная память регистров общего назначения)	Косвенный, индексный или прямой через окно
00FF 0000	Младший регистровый файл (оперативная память регистров общего назначения, указатель стека и центрального процессора SFRs)	Прямой, косвенный или индексный
Примечания 1 После сброса МК 1874ВЕх6Т выбирает первую команду из ячейки 2080Н. 2 Содержимое или функции этих ячеек могут измениться в будущих версиях микроконтроллера.		

Программная память и память специального назначения

Внутренняя энергонезависимая память – дополнительный компонент МК 1874ВЕ86Т (память OTPROM). Энергонезависимая память разделяется на память специального назначения и память программ (ячейки 2000Н и выше; см. таблицу 5.1). Сигнал EA# управляет доступом к этой области памяти. Доступ осуществляется к внутренней памяти, если сигнал EA# поддерживается в высоком уровне, и к внешней памяти, если сигнал EA# поддерживается в низком уровне. Для устройств без

внутренней энергонезависимой (МК 1874ВЕ16Т) памяти сигнал EA# должен быть установлен в низкий уровень. Захват сигнала EA# осуществляется при сбросе микроконтроллера.

Память программ

Память программ расположена в области, начинающейся с адреса 2080H. Эта область памяти доступна для хранения выполняемых кодов и данных. Сигнал EA# служит для разрешения доступа к памяти программ. Доступ разрешен к внутренней памяти, если сигнал EA# установлен в высокий уровень, и к внешней памяти, если сигнал EA# установлен в низкий уровень. Для устройств без внутренней энергонезависимой памяти сигнал EA# должен быть установлен в низкий уровень. Сигнал EA# захватывается при сбросе МК 1874ВЕ86Т.

Примечание – Рекомендуется запись кода FFH (код инструкции для команды RST) в используемые ячейки памяти программ. Это вызовет сброс устройства, если программа начинает выполнять коды из неиспользованной памяти.

Память специального назначения

Память специального назначения находится по адресам 2000–207FH (таблица 5.2). Она содержит несколько зарезервированных ячеек памяти, байты конфигурации (CCBs) и векторы для периферийного сервера (PTS) и стандартных прерываний. Эта область адресов доступна во внутренней памяти МК 1874ВЕ86Т, если сигнал EA# установлен в высокий уровень, и во внешней памяти, если сигнал EA# установлен в низкий уровень. Для устройств без внутренней энергонезависимой памяти сигнал EA# должен быть установлен в низкий уровень. Сигнал EA# захватывается МК 1874ВЕ86Т при сбросе.

Таблица 5.2 – Адреса памяти специального назначения

Адрес	Описание
207F-205E	Зарезервированы (каждый байт должен содержать FFH)
205D-2040	Векторы PTS
203F-2030	Старшие векторы прерывания
202F-2020	Ключ защиты
201F	Зарезервирован (должен содержать 20H)
201E	Зарезервирован (должен содержать FFH)
201D	Зарезервирован (должен содержать 20H)
201C	Зарезервирован (должен содержать FFH)
201B	Зарезервирован (должен содержать 20H)
201A	CCB1 (байт конфигурации 1)
2019	Зарезервирован (должен содержать 20H)
2018	CCB0 (байт конфигурации 0)
2017-2014	Зарезервированы (каждый байт должен содержать FFH)
2013-2000	Младшие векторы прерывания

Резервные ячейки памяти

Несколько ячеек памяти зарезервировано. Не рекомендуется считывать или записывать эти ячейки, за исключением инициализации. Всегда инициализируйте резервные ячейки, устанавливая значения, приведенные в таблице 5.2.

Прерывания и векторы PTS

Старшие и младшие векторы прерываний содержат адреса подпрограмм обслуживания прерываний. Векторы периферийного сервера (PTS) содержат адреса блоков управления PTS.

Ключ защиты

Ключ защиты предотвращает несанкционированное программирование энергонезависимой памяти. См. раздел 13 “Программирование постоянной памяти”.

Байты конфигурации (CCBs)

Байты конфигурации (CCBs) определяют режим работы МК 1874ВЕ86Т. Они задают ширину шины, режим управления шиной и состояния ожидания. Они также управляют режимом пониженного энергопотребления, сторожевым таймером и защитой энергонезависимой памяти.

CCBs – первые байты, записываемые из памяти, когда устройство выходит из состояния сброса. После сброса последовательность CCBs загружается в регистры конфигурации (CCRs). После записи содержимое CCRs не может быть изменено до следующего сброса устройства. Как правило, CCBs программируются один раз, когда пользовательская программа компилируется, и не переопределяются в течение нормального режима работы.

Для устройств с программируемой пользователем энергонезависимой памятью CCBs загружены для нормальных операций, но PCCB загружаются в CCR, если МК 1874ВЕ86Т входит в режимы программирования, см. раздел 13 “Программирование постоянной памяти”.

Специально-функциональные регистры (SFRs)

Эти регистры или картирование в памяти SFR или периферийные SFR. К картированным в памяти SFR необходимо обращаться, используя режимы косвенной или индексной адресации, и они не могут выбираться в режиме окон. Периферийные SFR физически расположены на кристалле МК 1874ВЕх6Т, и они могут выбираться в режиме «окон» (см. подраздел 5.2 "Работа с окнами"). Не рекомендуется использовать резервные SFR, необходимо записать в них нули или оставить их в заданном по умолчанию состоянии. При чтении резервные биты и SFRs принимают неопределенные значения.

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 SFR. Также, потому что некоторые SFR очищаются при чтении, считается возможным использование SFR в инструкциях "чтение-модификация-запись" (например, XORB).

Картированные в памяти SFR

Ячейки 1FE0–1FFFH содержат картированные в памяти SFR (см. таблицу 5.3). Ячейки в этом диапазоне, которые пропущены в таблице 5.3, резервные. К SFR необходимо обращаться в режимах косвенной или индексной адресации, и они не могут быть выбраны в режимах «окон». К картированным SFRs доступ осуществляется через контроллер памяти, т.к. команды, которые работают с этим SFRs, выполняются так, как если бы они были из внешней памяти с нулевым временем ожидания.

Таблица 5.3 – Картированные в памяти SFRs

Порты 3, 4, 5, UPROM SFRs		
Адрес (шестнадцатиричный)	Старший (нечетный) байт	Младший (четный) байт
1FFE	P4_PIN	P3_PIN
1FFC	P4_REG	P3_REG
…	…	…
1FF6	P5_PIN	USFR
1FF4	P5_REG	Зарезервировано
1FF2	P5_DIR	Зарезервировано
1FF0	P5_MODE	Зарезервировано

Периферийные SFRs

Ячейки 1F00–1FDFH обеспечивают доступ к периферийным SFRs. Таблица 5.4 содержит список периферийных SFRs микроконтроллера. Ячейки, которые пропущены в таблице, резервные. Периферийные SFRs – регистры управления ввода-вывода; они физически расположены на кристаллах МК 1874ВЕх6Т. К периферийным SFRs возможен доступ в режиме окон и к ним можно обратиться как к слову или байту (кроме случаев, отдельно оговорённых в примечаниях к таблицам данного раздела).

К периферийным SFRs обращение непосредственное, без использования контроллера памяти, и команды, которые обращаются к этим SFRs, выполняются так, как если бы они оперировали с регистровым файлом.

Таблица 5.4 – Периферийные SFRs

SFR порта 2		
Адрес (шестнадцатиричный)	Старший (нечетный) байт	Младший (четный) байт
1	2	3
1FDEH	Зарезервирован	Зарезервирован
…	…	…
1FD6H	Зарезервирован	P2_PIN
1FD4H	Зарезервирован	P2_REG
1FD2H	Зарезервирован	P2_DIR
1FD0H	Зарезервирован	P2_MODE
SFR генератора формы волны		
Адрес (шестнадцатиричный)	Старший (нечетный) байт	Младший (четный) байт
1FCEH	Зарезервирован	WG_PROTECT
1FCCH	WG_CONTROL (H)	WG_CONTROL (L)
1FCAH	WG_COUNTER (H)	WG_COUNTER (L)
1FC8H	WG_RELOAD (H)	WG_RELOAD (L)
1FC6H	WG_COMP3 (H)	WG_COMP3 (L)
1FC4H	WG_COMP2 (H)	WG_COMP2 (L)
1FC2H	WG_COMP1 (H)	WG_COMP1 (L)
1FC0H	WG_OUTPUT (H)	WG_OUTPUT (L)
SFR периферийного прерывания и PWM		
Адрес (шестнадцатиричный)	Старший (нечетный) байт	Младший (четный) байт
1FBEH	Зарезервирован	PI_PEND
1FBCH	Зарезервирован	PI_MASK
…	…	…
1FB6H	Зарезервирован	PWM_COUNT
1FB4H	Зарезервирован	PWM_PERIOD
1FB2H	Зарезервирован	PWM1_CONTROL
1FB0H	Зарезервирован	PWM0_CONTROL
SFR АЦП		
Адрес (шестнадцатиричный)	Старший (нечетный) байт	Младший (четный) байт
1FAEH	AD_TIME	AD_TEST
1FACH	Зарезервирован	AD_COMMAND
1FAAH	AD_RESULT (H)	AD_RESULT (L)
1FA8H	P1_PIN	P0_PIN
1FA6H	Зарезервирован	Зарезервирован
…	…	…
1F80H	Зарезервирован	Зарезервирован

Продолжение таблицы 5.4

Адрес (шестнадцатиричный)	Старший (нечетный) байт	Младший (четный) байт
1	2	3
SFR EPA и таймеров		
†1F7EH	TIMER2 (H)	TIMER2 (L)
1F7CH	Зарезервирован	T2CONTROL
†1F7AH	TIMER1 (H)	TIMER1 (L)
1F78H	Зарезервирован	T1CONTROL
1F76H	Зарезервирован	Зарезервирован
1F74H	Зарезервирован	Зарезервирован
1F72H	T1RELOAD (H)	T1RELOAD (L)
1F70H	Зарезервирован	Зарезервирован
…	…	…
†1F66H	COMP3_TIME (H)	COMP3_TIME (L)
1F64H	Зарезервирован	COMP3_CON
†1F62H	COMP2_TIME (H)	COMP2_TIME (L)
1F60H	Зарезервирован	COMP2_CON
†1F5EH	COMP1_TIME (H)	COMP1_TIME (L)
1F5CH	Зарезервирован	COMP1_CON
†1F5AH	COMP0_TIME (H)	COMP0_TIME (L)
1F58H	Зарезервирован	COMP0_CON
1F56H	Зарезервирован	Зарезервирован
…	…	…
†1F4EH	EPA3_TIME (H)	EPA3_TIME (L)
1F4CH	Зарезервирован	EPA3_CON
†1F4AH	EPA2_TIME (H)	EPA2_TIME (L)
1F48H	Зарезервирован	EPA2_CON
†1F46H	EPA1_TIME (H)	EPA1_TIME (L)
1F44H	Зарезервирован	EPA1_CON
†1F42H	EPA0_TIME (H)	EPA0_TIME (L)
1F40H	Зарезервирован	EPA0_CON
† Адресуется как слово.		

Регистровый файл

Регистровый файл (см. рисунок 5.1) разделен на старший файл регистров и младший файл регистров. Старший файл регистров состоит из регистров общего назначения оперативной памяти. Младший файл регистров содержит регистры общего назначения оперативной памяти наряду с указателем стека (SP) и регистрами специальных функций центрального процессора (SFR).

Таблица 5.5 перечисляет адреса памяти регистрового файла. РАЛУ обращается к младшему файлу регистров непосредственно, без использования контроллера памяти. Он также доступен в режиме окон прямой адресацией (см. подраздел 5.2 "Работа с окнами"). Старший регистровый файл и периферийные SFR могут быть доступны в режиме окон. К регистрам в младшем файле регистров и регистрам в режиме окон можно обратиться прямой адресацией.

Примечание – Регистровый файл не должен содержать коды команд. Попытка выполнять команду из ячейки в файле регистров заставляет контроллер памяти принять команду из внешней памяти.

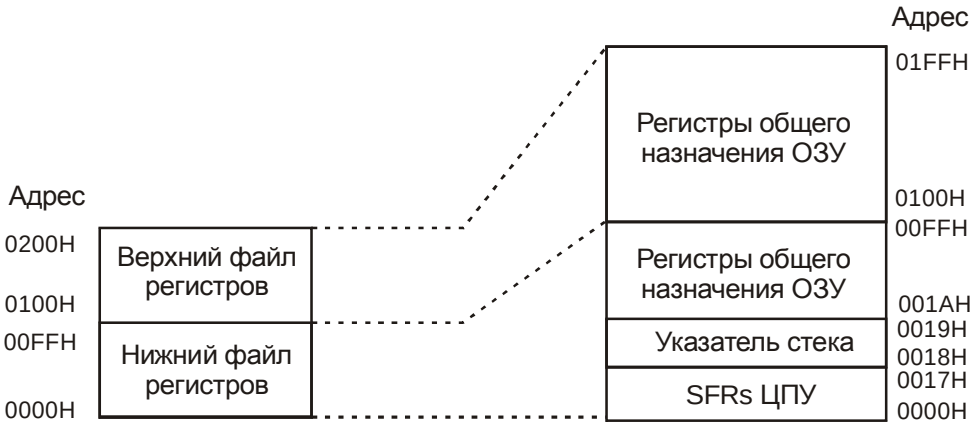


Рисунок 5.1 – Карта памяти файла регистров

Таблица 5.5 – Адреса памяти файла регистров

Диапазон адресов (шестнадцатиричные)	Описание	Способы адресации
01FF 0100	Старший файл регистров (регистры оперативной памяти)	Косвенный, индексный или прямой в режиме окон
00FF 001A	Младший регистровый файл (регистры оперативной памяти)	Прямой, косвенный или индексный
0019 0018	Младший регистровый файл (указатель стека)	Прямой, косвенный или индексный
0017 0000	Младший регистровый файл (SFRs ЦПУ)	Прямой, косвенный или индексный

Регистры общего назначения оперативной памяти

Младший файл регистров содержит оперативную память регистров общего назначения. Ячейки указателя стека могут также использоваться как оперативная память регистров общего назначения, когда операции стека не выполняются. РАЛУ может обратиться к этой памяти непосредственно, используя прямую регистровую адресацию.

Старший файл регистров также содержит оперативную память регистров общего назначения. РАЛУ обычно использует косвенную или индексную адресацию, чтобы обратиться к оперативной памяти в старшем файле регистров. Работа с окнами дает возможность РАЛУ использовать прямую регистровую адресацию, чтобы обратиться к этой памяти.

Работа с окнами может обеспечить быстрое переключение контекста задач прерывания и более быстрого выполнения программы (см. подраздел 5.2 "Работа с окнами"). Блоки PTS и стек наиболее эффективны, когда расположены в старшем файле регистров.

Указатель стека (SP)

Ячейки памяти 0018H и 0019H содержат указатель стека (SP). SP содержит адрес стека. SP должен указывать адрес слова (чётный), который на два байта больше, чем желаемый стартовый адрес. Прежде, чем центральный процессор выполняет вызов подпрограммы или процедуры обслуживания прерывания, он декрементирует SP на два и копирует адрес следующей команды из счетчика программы в стек. ЦП загружает адрес подпрограммы или процедуры обслуживания прерывания в счетчик программы.

Когда выполняется команда (RET) в конце подпрограммы или программы обслуживания прерывания, ЦП загружает (POP) содержимое вершины стека (то есть адрес возврата) в счетчик программы и увеличивает SP на два.

Подпрограммы могут быть вложены. То есть, каждая подпрограмма может вызвать другие подпрограммы. Центральный процессор помещает содержание счетчика программы в стек каждый раз, когда выполняется вызов подпрограммы. Стек становится нисходящим, поскольку увеличивается число записей. Единственное ограничение на глубину вложения – количество доступной памяти. Поскольку центральный процессор возвращается из каждой вложенной подпрограммы, он выталкивает адрес из вершины стека, и следующий адрес возврата перемещается в вершину стека.

Программа должна загрузить выровненный по слову (чётный) адрес в указатель стека. Необходимо выбрать адрес, который на два байта больше, чем желательный стартовый адрес, потому что центральный процессор автоматически уменьшает указатель стека прежде, чем помещает первый байт адреса возврата в стек. Стек становится нисходящим, что позволяет отвести достаточно места для максимального числа обращений к стеку. Стек должен быть расположен или во внутреннем файле регистров или во внешней оперативной памяти. Стек может использоваться наиболее эффективно, когда он расположен в файле регистров.

Следующий пример инициализирует вершину старшего файла регистров микроконтроллера как стек.

LD SP, #200H; загрузить указатель стека

Следующий пример показывает, как позволить устройству ввода позиций компоновщика определять, где стек соответствует карте памяти.

LD SP, #STACK

Регистры специальных функций (SFRs) центрального процессора

Ячейки 0000–0017H в младшем файле – регистры специальных функций центрального процессора (см. таблицу 5.6).

Таблица 5.6 – SFRs центрального процессора

Адрес (шестнадцатиричный)	Старший (нечетный) байт	Младший (четный) байт
0016H	Резервный	Резервный
0014H	Резервный	WSR
0012H	INT_MASK1	INT_PEND1
0010H	Резервный	Резервный
000EH	Резервный	Резервный
000CH	Резервный	Резервный
000AH	Резервный	WATCHDOG
0008H	INT_PEND	INT_MASK
0006H	PTSSRV (H)	PTSSRV (L)
0004H	PTSSEL (H)	PTSSEL (L)
0002H	ONES_REG (H)	ONES_REG (L)
0000H	ZERO_REG (H)	ZERO_REG (L)

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

5.2 Работа с окнами

Работа с окнами расширяет объем памяти, который является доступным с прямой регистровой адресацией. Прямая регистровая адресация может обратиться к младшему файлу регистров с короткими, быстро выполняющими командами. При работе с окнами прямая регистровая адресация может также обратиться к старшему файлу регистров и периферийным SFRs.

Работа с окнами отображает сегмент памяти (верхний файл регистров или периферийные SFRs) в младший файл регистров. Регистр выбора окна (WSR) выбирает 32-, 64- или 128-байтный сегмент старшей памяти для создания окон в вершине младшего файла регистров. На рисунке 5.2 показано 128-байтное окно.

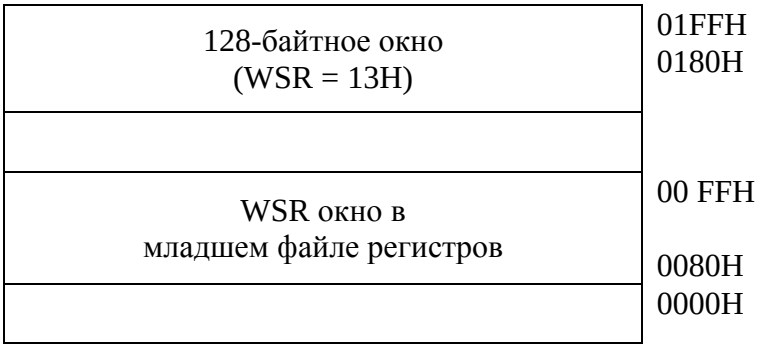


Рисунок 5.2 – Работа с окнами

Примечание – Картированные в памяти SFR могут быть доступны в режимах косвенной или индексной адресации; к ним нельзя обратиться через окно. Чтение картированного в памяти SFR через окно дает FFH (все «1»), а запись в картированный в памяти SFR через окно не имеет никакого эффекта.

Выбор окна

Регистр выбора окна (рисунок 5.3) выбирает окно, которое будет отображено в вершине младшего файла регистров.

Таблица 5.7 содержит справочник значений WSR для работы с окнами периферийных SFR. Таблица 5.8 содержит значения WSR для работы с окнами старшего регистрового файла.

WSR Адрес: 0014H
Состояние сброса: 00H

Регистр выбора окна (WSR) помещает секции оперативной памяти в вершину младшего регистрового файла с 32-, 64- или 128-байтовым приращением. PUSHA сохраняет этот регистр в стеке, а POPA восстанавливает его.

7							0
-	W6	W5	W4	W3	W2	W1	W0

Номер бита	Мнемоника бита	Функция
7	—	Зарезервирован; для совместимости с будущими устройствами записывается «0».
6:0	W6:0	Выбор окна. Эти биты определяют размер окна и их число. См. таблицу 5.7 для периферийных SFR или таблицу 5.8. для старшего файла регистра.

Рисунок 5.3 – Регистр выбора окна (WSR)

Таблица 5.7 – Выбор окна периферийных SFR

Внешние устройства	WSR значение для 32-байтного окна (00E0–00FFH)	WSR значение для 64-байтного окна (00C0–00FFH)	WSR значение для 128-байтного окна (0080–00FFH)
Порт 2 ГФС	7EH	3FH	1FH
Периферийные прерывания ШИМ АЦП	7DH	3EH	
Таймеры 1–2 EPA сравнение 2–3	7BH	3DH	1EH
EPA захват/сравнение 0-3 EPA сравнение 0-1	7AH		

Таблица 5.8 – Выбор окна старшего файла регистров

Оперативная память РОН	WSR значение для 32-байтового окна (00E0–00FFH)	WSR значение для 64-байтового окна (00C0–00FFH)	WSR значение для 128-байтового окна (0080–00FFH)
01E0–01FFH	4FH	27H	13H
01C0–01DFH	4EH		
01A0–01BFH	4DH	26H	
0180–019FH	4CH		
0160–017FH	4BH	25H	12H
0140–015FH	4AH		
0120–013FH	49H	24H	
0100–011FH	48H		

Адресация к ячейкам памяти через окно

После того, как выбрано желательное окно, необходимо знать в окне прямой адрес ячейки памяти (адрес в младшем файле регистра). Вычисляется прямой адрес следующим образом:

Вычесть базовый адрес области, которая будет повторно отображена (см. таблицу 5.9) из адреса выбранной ячейки. Это дает смещение адреса ячейки.

Добавляется смещение к базовому адресу окна (см. таблицу 5.10). Результат – прямой адрес в окне.

Таблица 5.9 – Окна

<table>
<tr><th>Базовый адрес</th><th>WSR значение для 32-байтового окна (00E0 – 00FFH)</th><th>WSR значение для 64-байтового окна (00C0–00FFH)</th><th>WSR значение для 128-байтового окна (0080–00FFH)</th></tr>
<tr><td colspan="4">Периферийные SFRs</td></tr>
<tr><td>1FE0H</td><td>7FH (примечание)</td><td rowspan="2">3FH (примечание)</td><td rowspan="4">1FH (примечание)</td></tr>
<tr><td>1FC0H</td><td>7EH</td></tr>
<tr><td>1FA0H</td><td>7DH</td><td rowspan="2">3EH</td></tr>
<tr><td>1F80H</td><td>7CH</td></tr>
<tr><td>1F60H</td><td>7BH</td><td rowspan="2">3DH</td><td rowspan="4">1EH</td></tr>
<tr><td>1F40H</td><td>7AH</td></tr>
<tr><td>1F20H</td><td>79H</td><td rowspan="2">3CH</td></tr>
<tr><td>1F00H</td><td>78H</td></tr>
<tr><td>02E0H</td><td>57H</td><td rowspan="2">2BH</td><td rowspan="4">15H</td></tr>
<tr><td>02C0H</td><td>56H</td></tr>
<tr><td>02A0H</td><td>55H</td><td rowspan="2">2AH</td></tr>
<tr><td>0280H</td><td>54H</td></tr>
<tr><td>0260H</td><td>53H</td><td rowspan="2">29H</td><td rowspan="4">14H</td></tr>
<tr><td>0240H</td><td>52H</td></tr>
<tr><td>0220H</td><td>51H</td><td rowspan="2">28H</td></tr>
<tr><td>0200H</td><td>50H</td></tr>
<tr><td>01E0H</td><td>4FH</td><td rowspan="2">27H</td><td rowspan="4">13H</td></tr>
<tr><td>01C0H</td><td>4EH</td></tr>
<tr><td>01A0H</td><td>4DH</td><td rowspan="2">26H</td></tr>
<tr><td>0180H</td><td>4CH</td></tr>
<tr><td>0160H</td><td>4BH</td><td rowspan="2">25H</td><td rowspan="4">12H</td></tr>
<tr><td>0140H</td><td>4AH</td></tr>
<tr><td>0120H</td><td>49H</td><td rowspan="2">24H</td></tr>
<tr><td>0100H</td><td>48H</td></tr>
<tr><td colspan="4">Примечание – Ячейки 1FE0–1FFFH содержат картированные в памяти SFR, к которым нельзя обратиться через окно. Чтение по этим адресам через окно устанавливает FFH; запись по этим адресам через окно не имеет никакого эффекта.</td></tr>
</table>

Таблица 5.10 – Базовые адреса окон

Размер окна	WSR базовый адрес (базовый адрес в младшем файле регистров)
32 байта	00E0H
64 байта	00C0H
128 байтов	0080H

Пример работы с 32-байтными окнами

Предположим, необходимо обратиться к ячейке 014BH (в старшем файле регистров, используемом для оперативной памяти регистров общего назначения) с прямой регистровой адресацией через 32-байтовое окно. Таблица 5.9 указывает, что нужно записать 4AH в регистр выбора окна. Базовый адрес 32-байтовой области памяти 0140H. Чтобы определить смещение, необходимо вычесть этот базовый адрес из адреса, к которому будет обращение (014BH – 0140H = 000BH). Добавить смещение к базовому адресу окна в младшем файле регистров (00E0H, см. таблицу 5.10). Прямой адрес – 00EBH (000BH + 00E0H).

Пример работы с 64-байтными окнами

Предположим, необходимо обратиться к регистру WG_CONTROL (ячейка 1FCCH) с прямой адресацией через 64-байтовое окно. Таблица 5.9 указывает, что необходимо записать 3FH в регистр выбора окна. Базовый адрес 64-байтовой области памяти – 1FC0H. Чтобы определить смещение, надо вычесть этот базовый адрес из адреса, к которому будет обращение (1FCCH – 1FC0H = 000CH) и добавить смещение к базовому адресу окна в младшем файле регистров (00C0H, см. таблицу 5.10). Прямой адрес – 00CCH (000CH + 00C0H).

Пример работы со 128-байтными окнами

Предположим, необходимо обратиться к ячейке 1F42H (регистр EPA0_TIME) с прямой регистровой адресацией через 128-байтовое окно. Таблица 5.9 указывает, что необходимо написать 1EH в регистр выбора окна. Базовый адрес 128-байтовой области памяти – 1F00H. Чтобы определить смещение, надо вычесть этот базовый адрес из адреса, к которому будет обращение (1F42H – 1F00H = 0042H) и добавить смещение к базовому адресу окна в младшем файле регистров (0080H, см. таблицу 5.10). Прямой адрес – 00C2H (0042H + 0080H).

Пример работы с окнами в области памяти, неподдерживаемой режимом окон

Предположим, необходимо обратиться к ячейке 1FF1H (регистр P5_MODE, картированный в памяти SFR) с прямой регистровой адресацией через 128-байтовое окно. Эта ячейка находится в диапазоне адресов (1FE0–1FFFH), который не поддерживается режимом окон. Хотя можно выбрать окно, записать 1FH в WSR, чтение этой ячейки через окно установит FFH (все 1), и запись не изменит содержимое. Однако возможно обратиться к периферийным SFRs в диапазоне 1F80–1FDFH по их прямым адресам в окне.

Работа с окнами и способы адресации

При установке режима окна доступ к соответствующим ячейкам осуществляется через окно, используя прямую (8 битов) адресацию и обычную адресацию на 16 битов. Ячейки младшего файла регистров, которые покрыты окном, всегда доступны косвенными или индексными инструкциями. Повторное разрешение прямого доступа ко всему младшему регистровому файлу очищает WSR. Для разрешения прямого доступа к определенной ячейке в младшем файле регистров необходимо выбрать меньшее окно, которое не покрывает эту ячейку.

Когда работа с окнами разрешена:

- прямая регистровая команда, использующая адрес в пределах младшего файла регистров фактически обращается к окну в старшем файле регистров;
- команды косвенные, индексированные или с нулевым регистром, использующие адрес или в пределах младшего файла регистров, или в пределах старшего регистрового файла, обращаются к ячейкам памяти.

Следующая типовая программа иллюстрирует различие между прямой регистровой и индексной адресацией при использовании работы с окнами.

```
PUSHA              ; помещает содержание WSR в стек
LDB WSR, #12H      ; выбор окна 12H, 128-байтовый блок
                   ; следующая команда использует прямую регистровую адресацию
ADD 40H, 80H       ; mem_word (40H) ←mem_word (40H) + mem_word (380H)
                   ; следующие две команды используют косвенную адресацию
ADD 40H, 80H [0]   ; 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               ; перезагружает предыдущее содержание в WSR
```

6 Стандартные и PTS прерывания

В этом разделе описывается схема управления прерываниями, схема приоритетов и синхронизации для стандартных прерываний и прерываний периферийного сервера транзакций (PTS). Описаны три специальных прерывания и семь режимов PTS, четыре из которых используются с EPA, чтобы обеспечить последовательный программный канал ввода-вывода для синхронных и для асинхронных передач и приемов. Описано программирование и управление прерываниями.

6.1 Краткий обзор прерываний

Схема управления прерываниями микроконтроллера разрешает событиям в реальном времени управлять процессом выполнения программы. Когда событие генерирует прерывание, устройство приостанавливает выполнение текущего потока команд и начинает выполнять процедуру обслуживания в ответ на прерывание. Когда обслуживание прерывания завершается, выполнение программы продолжается в точке, где произошло прерывание. Периферийное устройство, внешний сигнал или команда могут произвести запрос прерывания. В самом простом случае устройство получает запрос, исполняет обслуживание и возвращается к задаче, которая была прервана.

Гибкая система обработки прерываний микроконтроллера имеет два главных компонента: программируемый контроллер прерываний и периферийный сервер (PTS). Программируемый контроллер прерываний имеет аппаратную схему приоритета, которая может быть изменена вашим программным обеспечением. Прерывания, которые проходят через контроллер прерываний, обслуживаются программами обслуживания прерываний. Старшие и младшие векторы в памяти специального назначения (см. раздел 5 “Распределение памяти”) содержат адреса программ обслуживания прерываний. Периферийный сервер (PTS) обеспечивает быстродействующую обработку, не изменяя стек или PSW. Большинство прерываний (кроме NMI, TRAP и невыполняемого кода) можно обслуживать в PTS вместо контроллера прерываний.

PTS поддерживает семь специальных микропрограмм, которые позволяют ему выполнять определенные задачи значительно быстрее, чем эквивалентные процедуры обслуживания прерываний. PTS может передать байты или слова (индивидуально или в блоках) между любыми ячейками запоминающего устройства, управлять многократным аналого-цифровым (A/D) преобразованием и передавать и принимать последовательные данные или в асинхронном, или в синхронном режиме. PTS прерывания имеют более высокий приоритет, чем стандартные прерывания, и могут временно приостановить стандартные процедуры обслуживания прерываний.

Блок данных, называемый блоком управления PTS (PTSCB), содержит определенные данные для каждой процедуры PTS. Когда формируется PTS прерывание, кодирующее устройство приоритета выбирает соответствующий вектор и захватывает блок управления PTS (PTSCB).

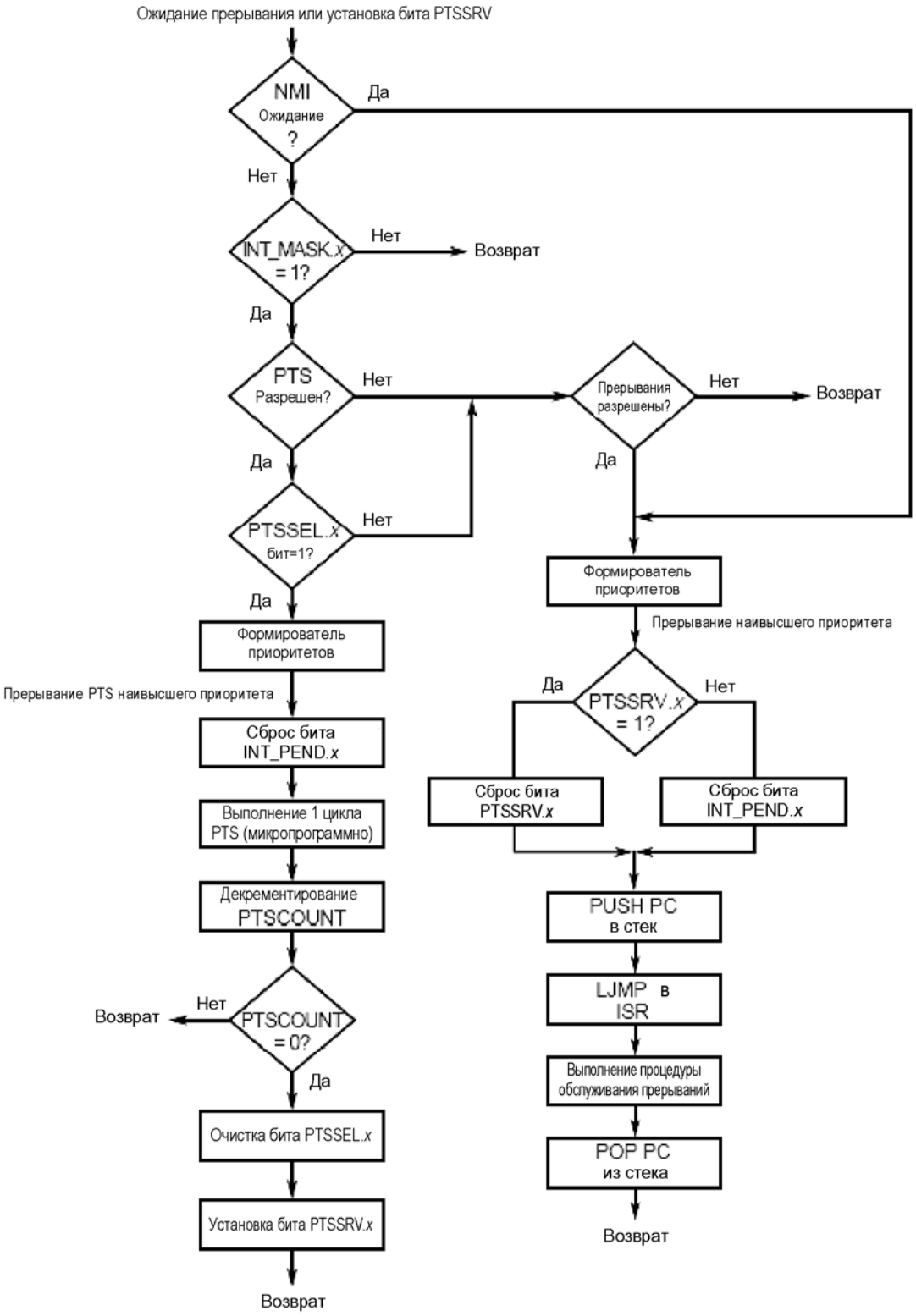


Рисунок 6.1 – Процедура обработки PTS и стандартных прерываний

Рисунок 6.1 иллюстрирует процедуру обработки прерываний, где “INT_MASK” означает и INT_MASK, и INT_MASK1 регистры, а “INT_PEND” означает INT_PEND1 и INT_PEND регистры.

6.2 Сигналы и регистры прерываний

В таблице 6.1 описываются внешние сигналы прерываний, а таблица 6.2 описывает регистры управления и состояния для контроллера прерываний и для PTS.

Таблица 6.1 – Сигналы прерываний

Сигнал прерывания	Тип	Описание
EXTINT	I	Внешнее прерывание. Это программируемое прерывание управляется регистром WG_PROTECT. Регистр определяет, какое событие вызовет прерывание, передний фронт/высокий уровень сигнала или задний фронт/низкий уровень сигнала. В режиме пониженного энергопотребления удержание сигнала EXTINT не менее 50 нс возвращает микроконтроллер в режим нормальной работы. Прерывание при этом не нуждается в разрешении. Если прерывание EXTINT разрешено, процессор вызывает программу обслуживания, иначе центральный процессор выполняет инструкцию, которая немедленно следует за командой, которая вызвала режим пониженного энергопотребления. В режиме холостого хода любое разрешенное прерывание возвращает контроллер в режим нормальной работы.
NMI	I	Немаскируемое прерывание. NMI имеет самый высокий приоритет из всех приоритетных прерываний. Для гарантированного распознания необходимо удерживать длительность сигнала NMI более одного машинного цикла.

Таблица 6.2 – Регистры управления и состояния прерываний и PTS

Мнемоника	Адрес	Описание
INT_MASK INT_MASK1	0008H, 0013H	Регистры маски прерываний. Эти регистры разрешают/запрещают каждое маскируемое прерывание, кроме MNI, программных TRAP и прерываний по некорректному коду инструкций.
INT_PEND INT_PEND1	0009H, 0012H	Регистры задержки прерываний. Биты в этих регистрах, установленные аппаратными средствами, указывают, какие прерывания задержаны в обслуживании.
PI_MASK	1FBCH	Регистр маски периферийных прерываний. Биты в этом регистре разрешают запросы прерываний, переполнение/антипереполнение таймеров 1, 2 и запросы прерываний генератора формы сигнала (WFG).
PI_PEND	1FBEH	Регистр задержки прерываний периферийных устройств. Любой установленный бит указывает на задержку обслуживания соответствующего запроса прерываний.
PSW	Нет прямого доступа	Слово состояния процессора. Этот регистр содержит один бит, который глобально разрешает/запрещает обслуживание всех маскируемых прерываний, и другой, который разрешает/запрещает PTS. Эти биты устанавливают или сбрасывают командами EI, DI, EPTS и DPTS соответственно.
PTSSEL	0004H, 0005H	Регистр выбора PTS. Этот регистр выбирает или процедуру PTS или стандартную процедуру обслуживания прерывания для каждого из маскируемых запросов прерываний.
PTSSRV	0006H, 0007H	Регистр обслуживания PTS. Биты в этом регистре, установленные аппаратными средствами, запрашивают обслуживание окончания прерываний PTS (т.н. end-of-PTS).

6.3 Источники и приоритеты прерывания

Таблица 6.3 содержит перечень источников, их заданные по умолчанию приоритеты (30 – самый высокий и 0 – самый низкий) и их векторы-адреса. Прерывания по некорректному коду инструкции и программные TRAP не имеют приоритетов. Они непосредственно поступают для обслуживания в контроллер прерываний. Кодирующее устройство приоритета определяет приоритет всех остальных задержанных прерываний. NMI имеет самый высокий приоритет. PTS прерывания имеют следующий самый высокий приоритет, а стандартные прерывания – самый низкий. Кодирующее устройство приоритета выбирает самый высокий приоритет удержанного запроса, и контроллер прерываний выбирает соответствующую вектору ячейку в запоминающем устройстве специального назначения. Этот вектор содержит стартовый адрес соответствующего блока управления PTS (PTSCB) или программу обслуживания. PTSCB должен быть расположен на границе кратной 4-м словам во внутреннем файле регистров.

Таблица 6.3 – Источники прерываний, векторы и приоритеты

Источник прерывания	Мнемо-ника	Обслуживание контроллера прерываний			Обслуживание PTS		
		Назва-ние	Вектор	При-ори-тет	Назва-ние	Вектор	При-ори-тет
1	2	3	4	5	6	7	8
Немаскируемое прерывание	NMI	INT15	203EH	30	—	—	—
Вывод EXTINT	EXTINT	INT14	203CH	14	PTS14	205CH	29
WF генератор	PI	INT13	203AH	13	PTS13	205AH	28
Зарезервирован	—	INT12	2038H	12	PTS12	2058H	27
Зарезервирован	—	INT11	2036H	11	PTS11	2056H	26
Зарезервирован	—	INT10	2034H	10	PTS10	2054H	25
EPA сравнение 3	COMP3	INT09	2032H	09	PTS09	2052H	24
EPA захват/сравнение 3	EPA3	INT08	2030H	08	PTS08	2050H	23
Некорректный opcode	—	—	2012H	—	—	—	—
Команды TRAP программного обеспечения	—	—	2010H	—	—	—	—
EPA сравнение 2	COMP2	INT07	200EH	07	PTS07	204EH	22
EPA захват/сравнение 2	EPA2	INT06	200CH	06	PTS06	204CH	21
EPA сравнение 1	COMP1	INT05	200AH	05	PTS05	204AH	20
EPA захват/сравнение 1	EPA1	INT04	2008H	04	PTS04	2048H	19
EPA сравнение 0	COMP0	INT03	2006H	03	PTS03	2046H	18
EPA захват/сравнение 0	EPA0	INT02	2004H	02	PTS02	2044H	17
АЦП преобразование завершено	AD_DONE	INT01	2002H	01	PTS01	2042H	16
Переполнение таймеров 1 или 2	OVRTM †	INT00	2000H	00	PTS00	2040H	15

† Обслуживание PTS некорректно для мультиплексных прерываний, потому что PTS не может определить источник прерываний.

Специальные прерывания

Микроконтроллеры имеют три специальных источника прерываний, которые всегда разрешены: некорректный код инструкции, программный TRAP и NMI. Эти прерывания не разрешаются/запрещаются командами EI/DI, и они не могут быть замаскированы. Все они обслуживаются контроллером прерываний и не могут быть обслужены на PTS. Из них только NMI проходит детектор перехода и дешифратор приоритета. Другие два идут непосредственно к контроллеру прерываний для обслуживания.

Некорректный код инструкции

Если центральный процессор пытается выполнить некорректный код инструкции, вырабатывается косвенный вектор через ячейку 2012H. Это предотвращает выполнение произвольной программы в течение отказов программного обеспечения и аппаратных средств. Вектор прерывания должен содержать стартовый адрес программы обслуживания ошибки, который не будет далее развивать ошибочную ситуацию. Прерывание по некорректному коду инструкции предотвращает запрос других прерываний и разрешение их обслуживания, пока следующая инструкция не будет выполнена.

Программный TRAP

Инструкция TRAP (код инструкции F7H) вызывает прерывание по вектору в ячейке 2010H. Инструкция TRAP обеспечивает прерывание, которое является полезным при отладке программного обеспечения или генерации прерываний программного обеспечения. Инструкция TRAP запрещает обслуживание других запросов прерывания, пока следующая инструкция не выполнена.

NMI

Внешний вывод NMI производит немаскируемое прерывание для реализации программ обслуживания критических ситуаций. NMI имеет самый высокий приоритет среди приоритизированных прерываний. Запрос передается непосредственно от детектора перехода в дешифратор приоритета и вызывает вектор косвенно через ячейку 203EH. NMI сканируется в течение фазы 2 (CLKOUT высокий), захватывается микроконтроллером. Так как прерывание производится фронтом, только один запрос прерывания может быть сгенерирован, даже если вывод поддерживается в высоком уровне.

Если система не использует NMI прерывание, необходимо соединить вывод NMI с #0V, чтобы предотвратить несанкционированное прерывание.

Вывод внешнего прерывания

Схема защиты в генераторе формы сигнала (рисунок 6.2) контролирует внешний сигнал EXTINT. Когда она обнаруживает действительное событие на входе, то одновременно запрещает выходы генератора формы сигнала и генерирует запрос прерывания EXTINT. Биты 2 и 3 регистра защиты генератора формы сигнала (WG_PROTECT) (см. рисунок 8.9) выбирают тип внешнего события, которое произведет запрос прерывания: задний или передний фронт, низкий или высокий уровень.

Когда принимается «событие – уровень», внешний сигнал должен остаться постоянным в течение, по крайней мере, 24 T_{BQ1} ($24/F_{BQ1}$), чтобы быть распознанным как действительное прерывание. Когда сигнал установлен, устройство выборки уровня принимает уровень сигнала три раза в течение 24 T_{BQ1} периодов. Когда соответствующий уровень распознан, устройство выборки уровня генерирует один импульс на выходе. Этот импульс производит запрос прерывания EXTINT. Режим «событие – уровень» полезен в оборудовании с высоким уровнем помех, где выброс помехи может вызвать незапланированный запрос прерывания.

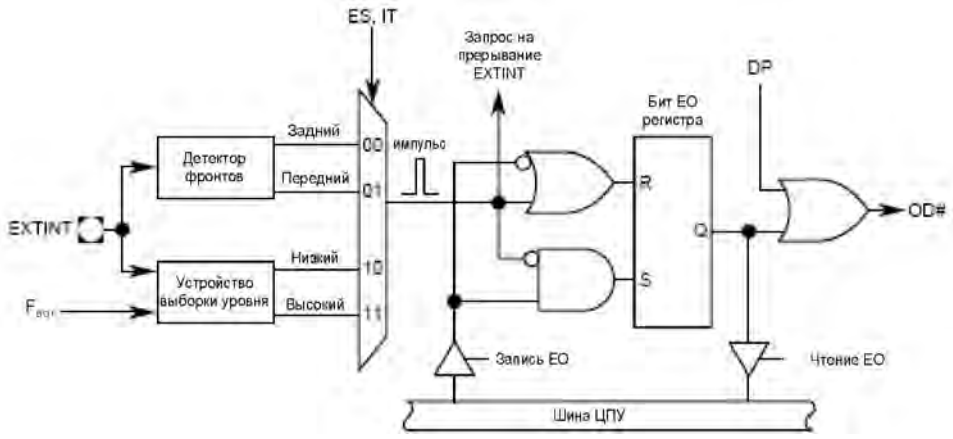


Рисунок 6.2 – Схема защиты генератора формы сигнала

Когда установлен режим «событие – фронт», сигнал на выходе должен остаться постоянным, по крайней мере, в течение двух T_{BQ1} ($2/F_{BQ1}$), чтобы быть распознанным как действительное прерывание. Когда действительный переход уровня произошел, детектор перехода формирует одиночный импульс. Импульс генерирует запрос прерывания EXTINT.

Мультиплексированные источники прерывания

OVRTM прерывание имеет мультиплексный источник (см. таблицу 6.3). Индивидуальный источник произведет прерывание, только если программное обеспечение разрешает источник прерывания и мультиплексное прерывание. Мультиплексное прерывание разрешается установкой соответствующего бита в регистре маски (рисунки 6.7 и 6.8). Чтобы разрешить источник прерывания, необходимо установить соответствующий бит в регистре PI_MASK (рисунок 6.9). Рисунок 6.3 иллюстрирует процедуру обработки по переполнению таймеров (OVRTM).

Примечание – Хотя прерывание PI имеет один источник (генератор формы сигнала), программное обеспечение должно разрешать источник прерывания (WG) в регистре PI_PEND и PI прерывание в регистре INT_MASK.

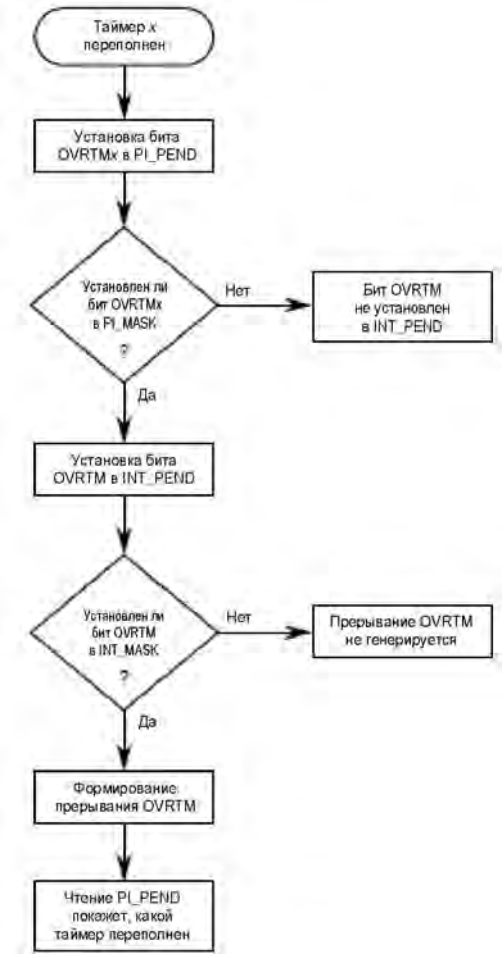


Рисунок 6.3 –Процедура обработки OVRTM прерывания

Программа обслуживания прерывания должна считать PI_PEND регистр (рисунок 6.12), чтобы определить источник прерывания. Перед выполнением инструкции возврата программа обслуживания прерывания должна проверить наличие других задержанных в обслуживании источников прерываний. Обслуживание прерываний PTS не рекомендуется для мультиплексных прерываний, потому что PTS не может определить источник прерывания.

Прерывания по обслуживанию PTS (end-of-PTS)

Когда содержимое регистра PTSCOUNT становится равно нулю в конце однократной передачи, передачи блока, АЦП сканирования или при процедуре последовательного ввода-вывода, аппаратные средства очищают соответствующий бит в регистре PTSSEL (рисунок 6.6), который запрещает PTS обслуживание этого прерывания. Также устанавливается соответственно PTSSRV бит, запрашивающий end-of-PTS прерывание. End-of-PTS прерывание имеет тот же самый приоритет, что и соответствующее стандартное прерывание. Контроллер прерываний обслуживает его программой обработки прерываний, находящейся в ячейке запоминающего устройства, на которую указывает вектор стандартного прерывания. Например, PTS обслуживает EPA0 прерывание, если установлен PTSSEL.2. Векторы прерываний находятся в ячейке 2044H, но вектор end-of-PTS прерывания в ячейке 2004H, стандартный вектор EPA0 прерывания. Когда end-of-PTS прерывание обслуживается программой обработки прерывания, аппаратные средства очищают PTSSRV бит. Программа обслуживания end-of-PTS прерывания должна повторно инициализировать PTSCB, если требуется, и устанавливать соответствующий PTSSEL бит, разрешающий обслуживание PTS прерывания.

6.4 Время ожидания прерывания

Время ожидания прерывания – полная задержка между временем, когда запрос прерывания произведен и временем, когда устройство начинает выполнение стандартной программы обслуживания прерывания или программы обслуживания прерывания PTS. Задержка происходит между временем, когда запрос прерывания обнаружен и временем, когда он подтвержден. Запрос прерывания подтвержден, когда текущая инструкция заканчивает выполняться. Если запрос прерывания происходит в течение одного из последних четырех тактов инструкции, он не может быть подтвержден до окончания следующей инструкции. Эта дополнительная задержка происходит, потому что инструкции предвыбираются за несколько машинных циклов прежде, чем они выполняются. Таким образом, максимальная задержка между запросом прерывания и подтверждением – четыре машинных цикла плюс время выполнения следующей инструкции.

Когда запрос стандартного прерывания подтвержден, аппаратные средства очищают бит задержки прерывания и производят вызов по адресу, содержавшемуся в соответствующем векторе прерывания. Когда запрос прерывания PTS подтвержден, аппаратные средства немедленно обращаются по вектору PTSCB, и начинается выполнение цикла PTS.

Ситуации, которые увеличивают время ожидания прерывания

Если запрос прерывания происходит в то время, когда любая из следующих инструкций выполняется, прерывание не будет подтверждено, пока не выполнится последующая инструкция:

- знаковый префикс кода инструкции (FE) для двухбайтовых команд умножения и деления со знаком;
- любая из этих восьми защищенных инструкций: DI, EI, APTS, EPTS, POPA, POPF, PUSHA, PUSHF;
- любая из инструкций "чтение-модификация-запись": AND, ANDB, OR, ORB, XOR, XORB.

Прерывания по несуществующему коду команды и прерывания программного TRAP запрещают другие запросы прерываний, пока не выполнена последующая инструкция.

Каждый цикл PTS в пределах процедуры PTS не может быть прерван. Цикл PTS – отклик PTS на отдельный запрос прерывания. В режиме передачи блока цикл PTS состоит из передачи полного блока байтов или слов. Это означает худший случай времени ожидания – 500 машинных циклов, если происходит передача блока 32 слов из одной области внешней памяти в другую (см. таблицу 6.4 для информации о времени выполнения циклов PTS).

Вычисление времени ожидания

Максимальное время ожидания получается, когда запрос прерывания происходит слишком поздно для подтверждения во время текущей инструкции. Приведенное ниже вычисление худшего случая предполагает, что текущая инструкция – не защищенная. При вычислении времени ожидания используются следующие термины:

- Время окончания выполнения текущей инструкции (4 цикла) – если инструкция защищенная, следующая за ней должна также выполняться перед подтверждением прерывания. Необходимо добавить время выполнения инструкции, которая следует за защищенной инструкцией.
- Время выполнения следующей инструкции. (Самая длинная инструкция NORML занимает 39 циклов. Однако BMOV инструкция может выполняться дольше, если передает большой блок данных. Если программа содержит процедуры, которые

передают большие блоки данных, можно получить более точное значение худшего случая, если используется инструкция BMOV вместо NORML, см. приложение А).

- Для стандартных прерываний время отклика на получение вектора и осуществление вызова – 11 циклов для внутреннего стека или 13 для внешнего стека (считая шину адреса данных с нулевым состоянием ожидания).

Время ожидания стандартных прерываний

Худший случай задержки для стандартных прерываний – 56 циклов (4+39+11+2), если стек находится во внешнем запоминающем устройстве (см. рисунок 6.4). Это время задержки не включает время выполнения первой инструкции в программе обслуживания или выполнения инструкции после защищенной инструкции.

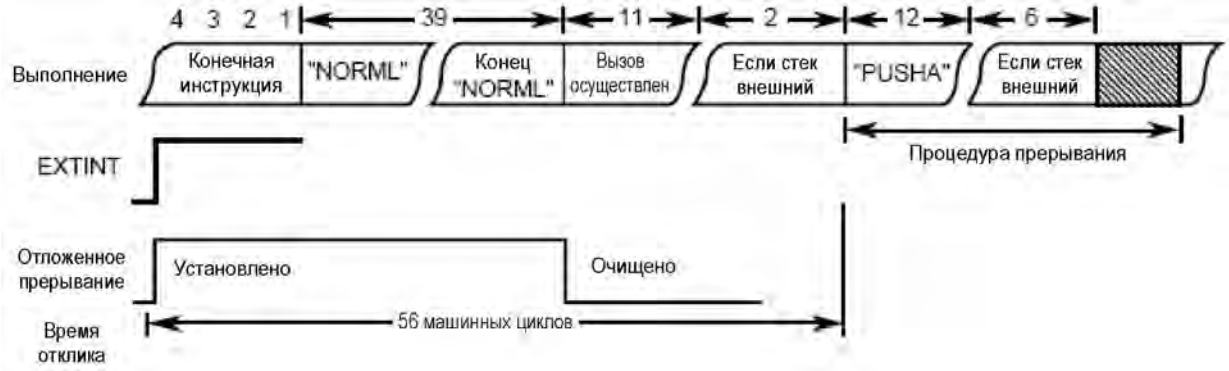


Рисунок 6.4 – Время отклика на стандартное прерывание

Время ожидания PTS прерывания

Максимальная задержка для PTS прерывания – 43 цикла (4 + 39) как показано на рисунке 6.5. Это время задержки не включает добавленную задержку, если выполняется защищенная инструкция или если запрос PTS уже принят (см. таблицу 6.4).

Рисунок 6.5 – Время отклика PTS прерывания

Таблица 6.4 – Время выполнения циклов PTS

Режим PTS	Время выполнения (в машинных циклах)
Режим одиночной передачи регистр/регистр † память/регистр † память/память †	 18 на передачу байта или слова + 1 21 на передачу байта или слова + 1 24 на передачу байта или слова + 1
Режим передачи блока регистр/регистр † память/регистр † память/память †	 13 + 7 на передачу байта или слова (1 минимум) 16 + 7 на передачу байта или слова (1 минимум) 19 + 7 на передачу байта или слова (1 минимум)
Режим сканирования АЦП регистр/регистр † регистр/память †	 21 25
ASIO режим приема Мажоритарность запрещена Мажоритарность разрешена	 24 + 2 (если контроль четности разрешен) 36 + время выборки (2-я выборка) 36 + 7 + время выборки (3-я выборка) 36 + 2 (если контроль четности разрешен)
ASIO режим передачи	29 + 3 (если контроль четности разрешен)
SSIO режим приема	29 (прием бита) 21 (нет приема)
SSIO режим передачи	30 (передача бита данных) 20 (нет передачи)
† «Регистр» указывает доступ к файлу регистров или периферийному SFR. «Память» указывает доступ к занесенному в карту памяти регистру, устройству ввода-вывода или запоминающему устройству. См. таблицу 5.1 для информации о распределении памяти.	

6.5 Программирование прерывания

Регистр выбора PTS (PTSSEL) выбирает или обслуживание PTS или стандартную процедуру обслуживания прерывания для каждого из маскируемых запросов прерываний (см. рисунок 6.6). Биты в регистрах маски прерываний INT_MASK и INT_MASK1 позволяют или запрещают (маскируют) индивидуальные прерывания (см. рисунки 6.7 и 6.8). Для источников мультиплексных прерываний биты в регистре PI_MASK (см. рисунок 6.9) разрешают или запрещают (маскируют) индивидуальные источники прерываний. Установка бита разрешает соответствующий источник прерывания и очистку бита запрещения источника за исключением бита немаскируемого прерывания (NMI) (INT_MASK1.7).

Для запрета любого прерывания необходимо очистить его бит маски. Чтобы разрешить обслуживание стандартного прерывания, устанавливается его маскирующий бит и очищается его бит выбора PTS. Чтобы разрешить прерывание для обслуживания PTS, необходимо установить и маскирующий бит, и бит выбора PTS.

Когда назначается обслуживание прерываниями PTS, необходимо установить блок управления PTS (PTSCB) для каждого источника прерываний (см. подраздел 6.6 “Инициализация блоков управления PTS”) и использовать EPTS инструкцию, чтобы глобально разрешить PTS. Для назначения обслуживания прерываний стандартными процедурами используют инструкцию EI, чтобы глобально разрешить обслуживание прерываний.

Примечание – Инструкция DI (запрет прерывания) не запрещает обслуживание PTS. Однако она запрещает обслуживание для запросов прерываний end-of-PTS. Если запрос прерываний происходит в то время, когда прерывания запрещены, соответствующий бит ожидания устанавливается в регистре INT_PEND1 или INT_PEND.

Обслуживание PTS не рекомендовано для мультиплексных прерываний, потому что PTS не может определить источник прерываний.

PTSSEL

Адрес: 0004H

Состояние Сброса: 0000H

Регистр выбора PTS (PTSSEL) выбирает или PTS микропрограмму, или процедуру обслуживания стандартного прерывания для каждого запроса прерывания. Установка бита выбирает PTS микропрограмму; очистка бита выбирает процедуру обслуживания стандартного прерывания. Когда PTSCOUNT становится равным нулю, аппаратные средства очищают соответственно PTSSEL бит. PTSSEL бит должен быть установлен в соответствии с руководством для перевыбора канала PTS.

15 8

—	EXTINT	PI	—	—	—	COMP3	EPA3

7 0

COMP2	EPA2	COMP1	EPA1	COMP0	EPA0	AD	OVRTM

Номер бита	Функция
15	Зарезервированный; для совместимости с будущими устройствами записать ноль в этот бит.
14:0†	Установка бита разрешает обслуживание соответствующего прерывания микропрограммой PTS.

Адреса ячеек векторов прерываний PTS

Мнемоника	Вектор PTS	Мнемоника	Вектор PTS
EXTINT	205CH	COMP2	204EH
PI ††	205AH	EPA2	204CH
COMP3	2052H	COMP1	204AH
EPA3	2050H	EPA1	2048H
		COMP0	2046H
		EPA0	2044H
		AD	2042H
		OVRTM ††	2040H

_____

† Биты 10–12 зарезервированы. Для совместимости с будущими устройствами должен быть установлен «0».

†† обслуживание PTS некорректно для мультиплексных прерываний, так как PTS не может определить источник прерываний.

Рисунок 6.6 – Регистр выбора PTS (PTSSEL)

INT_MASK

Адрес: 0008H

Состояние Сброса: 00H

Регистр маски прерываний (INT_MASK) разрешает или запрещает индивидуальные запросы прерываний. Команды EI и DI разрешают и запрещают обслуживание всех маскируемых прерываний. INT_MASK – младший байт слова состояния процессора (PSW). PUSHF или PUSHA сохранит содержимое этого регистра в стеке и затем очистит этот регистр. Вызов прерывания не может произойти

немедленно после этих инструкций. POPF или POPA восстанавливают содержимое регистра.

7							0
COMP2	EPA2	COMP1	EPA1	COMP0	EPA0	AD	OVRTM

Номер бита	Функция
7:0	Установка бита разрешает соответствующее прерывание

Адреса ячеек векторов стандартных прерываний

Мнемоника бита	Прерывание	Стандартный вектор
COMP2	EPA сравнение канал 2	200EH
EPA2	EPA захват/сравнение канал 2	200CH
COMP1	EPA сравнение канал 1	200AH
EPA1	EPA захват/сравнение канал 1	2008H
COMP0	EPA сравнение канал 0	2006H
EPA0	EPA захват/сравнение канал 0	2004H
AD	A/D преобразование завершено	2002H
OVRTM †	Переполнение/антипереполнение таймера	2000H

_____

† Таймер 1 и таймер 2 могут производить мультиплексное прерывание по переполнению/антипереполнению таймера. Записывается PI_MASK, чтобы разрешить источники прерываний. Считается PI_PEND, чтобы определить, который источник вызвал прерывание.

Рисунок 6.7 – Регистр маски прерываний (INT_MASK)

INT_MASK1 Адрес: 0013H
Состояние Сброса: 00H

Регистр маски прерывания (INT_MASK1) позволяет или запрещает индивидуальные запросы прерываний. (Инструкции EI и DI позволяют и запрещают обслуживание всех маскируемых прерываний.) INT_MASK1 может считываться или записываться как байтовый регистр. PUSHA сохраняет содержимое регистра в стеке, а POPA восстанавливает его содержимое.

7							0
NMI	EXTINT	PI	-	-	-	COMP3	EPA3

Номер бита	Функция
7:0 †	Установка бита разрешает соответствующее прерывание

Мнемоника бит	Прерывание	Стандартный вектор
NMI	Немаскируемое прерывание	203EH
EXTINT	Вывод EXTINT	203CH
PI	Мультиплексное периферийное прерывание	203AH
COMP3	EPA сравнение Канал 3	2032H
EPA3	EPA захват/сравнение Канал 3	2030H

_____

† Биты 4–2 зарезервированы для совместимости с будущими устройствами, заполняются «0».

Рисунок 6.8 – Регистр маски прерываний 1 (INT_MASK1)

PI_MASK Адрес: 1FBCH
Состояние Сброса: AAH

Регистр маски периферийных прерываний (PI_MASK) разрешает или запрещает запросы прерываний, связанные с периферийными прерываниями (PI), и прерываниями по переполнению/антипереполнению таймеров (OVRTM).

7							0
-	-	-	WG	-	OVRTM2	-	OVRTM1

Номер бита	Мнемоника бита	Функция
7, 6, 5, 3, 1	—	Зарезервированы для совместимости с будущими устройствами, записываются «0».
4	WG	Генератор формы сигнала. Установка бита разрешает прерывание генератора формы сигнала.
2	OVRTM2	Переполнение/антипереполнение таймера 2. Установка этого бита разрешает прерывание переполнения/антипереполнения таймера 2. Прерывания переполнения/антипереполнения таймера 2 и таймера 1 связаны с прерыванием переполнения/антипереполнения таймера (OVRTM). Установка INT_MASK.0 разрешает OVRTM.
0	OVRTM1	Переполнение/антипереполнение таймера 1. Установка этого бита разрешает прерывание переполнения/антипереполнения таймера 1. Прерывания переполнения/антипереполнения таймера 2 и таймера 1 связаны с прерыванием переполнения/антипереполнения таймера (OVRTM). Установка INT_MASK.0 разрешает OVRTM.

Рисунок 6.9 – Регистр маски прерываний периферийных устройств (PI_MASK)

Изменение приоритетов прерываний

Программное обеспечение может изменять заданные по умолчанию приоритеты маскируемых прерываний, управляя регистрами маски прерываний (INT_MASK и INT_MASK1). Например, можно определить какое прерывание, если таковые вообще имеются, может прервать программу обслуживания. Следующая программа показывает один из способов запретить всем прерываниям, кроме EXTINT (приоритет 14), прерывать процедуру обслуживания АЦП-преобразования (приоритет 01).

```
SERIAL_RI_ISR:
    PUSHA                          ; Сохранить PSW, INT_MASK, INT_MASK1 и WSR
                                   ; (Запрет всех прерываний)
    LDB INT_MASK1, *01000000B      ; Разрешено только EXTINT
    EI                             ; Разрешено обслуживание прерывания
                                   ; Обслужить AD_DONE прерывание
    POPA                           ; Восстановить PSW, INT_MASK, INT_MASK1 и
                                   ; WSR регистры
    RET
CSEG B 02002-ом                    ; Заполнить таблицу прерываний
   DCW AD_DONE_ISR                  END
```

Необходимо заметить, что ячейка 2002H в таблице векторов прерываний должна быть загружена значением AD_DONE_ISR прежде, чем происходит запрос на

прерывание и что прерывание завершения АЦП преобразования должно быть разрешено для выполнения этой программы.

Эта программа, подобно всем программам обслуживания прерываний, составлена следующим образом:

1 После того, как аппаратные средства обнаруживают и располагают по приоритетам запрос прерывания, они производят вызов прерывания. Это выталкивает программный счетчик в стек и затем загружает в него содержимое вектора, соответствующего самому высокому приоритету ожидаемого немаскированного прерывания. Аппаратные средства не будут позволять вызов другого прерывания, пока первая инструкция процедуры обслуживания прерывания не выполнена.

2 PUSHA инструкция помещает содержимое PSW, INT_MASK, INT_MASK1 и регистра выбора окна (WSR) в стек и затем очищает PSW, INT_MASK и регистры INT_MASK1. В дополнение к арифметическим флагам PSW содержит бит разрешения глобального прерывания (I) и бит разрешения PTS (PSE). Очищая PSW и регистры масок прерывания, PUSHA фактически запрещает все маскируемые прерывания, запрещает стандартное обслуживание прерывания и запрещает PTS. Поскольку PUSHA – защищенная инструкция, она также запрещает вызовы прерывания, пока следующая инструкция не выполнится.

3 LDB INT_MASK1 инструкция разрешает те прерывания, которые определены для прерывания процедуры обслуживания. В этом примере только EXTINT может прервать принятую процедуру обслуживания прерываний. Разрешая или запрещая прерывания, программное обеспечение устанавливает свои собственные приоритеты обслуживания прерываний.

4 Инструкция EI снова позволяет обработку прерываний и запрещает вызовы прерываний, пока следующая инструкция не выполнится.

5 Реально процедура обслуживания прерываний выполняется в соответствии с приоритетами, установленными программным обеспечением.

6 В конце процедуры обслуживания инструкция POPA восстанавливает первоначальное содержимое регистров PSW, INT_MASK, INT_MASK1 и WSR; любые изменения, сделанные в этих регистрах в течение процедуры обслуживания прерываний, перезаписываются. Поскольку запросы прерываний не могут произойти немедленно после инструкции POPA, последняя инструкция (RET) выполнится прежде, чем может произойти другой запрос прерывания.

"Преамбула" и исполняемый код для этой процедуры не сохраняются и не восстанавливаются в регистровом ОЗУ. Процедура обслуживания прерываний предполагает размещение ее собственного набора регистров в младшем регистровом файле. Кроме того, память в верхнем файле регистров доступна через «окна».

Определение источника прерываний

Когда аппаратные средства обнаруживают прерывание, они устанавливают соответствующий бит в INT_PEND или регистре INT_PEND1 (рисунки 6.10 и 6.11). Установка бита происходит, хотя индивидуальное прерывание может быть замаскировано. Аппаратные средства очищают бит ожидания, когда программа обращается к процедуре обслуживания. Процедура обслуживания может читать INT_PEND и INT_PEND1, чтобы определить, которое из прерываний ожидается.

Программное обеспечение может генерировать прерывание установкой бита в регистре INT_PEND1 или INT_PEND. Мы рекомендуем использование инструкций "чтение-модификация-запись" типа AND и OR для модификации этих регистров.

```
ANDB    INT_PEND, #11111110B;   очистка бита OVRTM
XOR     INT_PEND, #00000001B;   установка бита OVRTM
```

Другие методы также могут быть использованы в цикле прерываний. Например, прерывание могло произойти в течение последовательности инструкций, которые

загружают содержимое регистра ожидаемого прерывания во временный регистр, изменяют содержимое временного регистра, и затем записывают содержимое временного регистра обратно в регистр ожидания прерываний. Если прерывание происходит в течение одного из последних четырех машинных циклов второй инструкции, оно не будет подтверждено до окончания завершения третьей инструкции. Поскольку третья инструкция переписывает содержимое регистра ожидания прерываний, то переход по вектору не будет происходить.

OVRTM прерывание имеет мультиплексные источники прерываний. См. PI_PEND (рисунок 6.12), чтобы определить, какой источник произвел запрос прерывания. Чтение PI_PEND очищает все биты. PI_PEND - только читаемый регистр.

INT_PEND Адрес: 0009H
Состояние сброса: 00H

Когда аппаратные средства обнаруживают запрос прерывания, устанавливается соответствующий бит в регистрах INT_PEND или INT_PEND1. Когда вектор выбран, то аппаратные средства очищают бит ожидания. Программное обеспечение может генерировать прерывание установкой соответствующего бита ожидания прерывания.

7 0

COMP2	EPA2	COMP1	EPA1	COMP0	EPA0	AD	OVRTM

Номер бита	Функция
7:0	Любой установленный бит указывает, что ожидается обслуживание прерывания. Бит прерывания очищается, когда процесс передается по соответствующему вектору прерывания.

Адреса ячеек векторов стандартных прерываний:

Мнемоника бита	Прерывание	Стандартный вектор
COMP2	EPA сравнение канал 2	200EH
EPA2	EPA захват/сравнение канал 2	200CH
COMP1	EPA сравнение канал 1	200AH
EPA1	EPA захват/сравнение канал 1	2008H
COMP0	EPA сравнение канал 0	2006H
EPA0	EPA захват/сравнение канал 0	2004H
AD	A/D преобразование завершено	2002H
OVRTM †	Переполнение/антипереполнение таймера	2000H

_____

† Таймер 1 и таймер 2 могут произвести мультиплексное прерывание по переполнению/антипереполнению. Записав в PI_MASK разрешение источника прерывания, считайте PI_PEND, чтобы определить источник, который вызвал прерывание.

Рисунок 6.10 – Регистр ожидания прерывания (INT_PEND)

INT_PEND1 Адрес: 0012H
Состояние Сброса: 00H

Когда аппаратные средства обнаруживают запрос прерывания, они устанавливают соответствующий бит в регистрах INT_PEND или INT_PEND1. Когда вектор выбран, аппаратные средства очищают соответствующий бит. Программное обеспечение может произвести прерывание установкой бита ожидания прерывания.

7							0
NMI	EXTINT	PI	-	-	-	COMP3	EPA3

Номер бита	Функция
7:0 †	Установленный бит разрешает соответствующее прерывание

Адреса ячеек векторов стандартных прерываний:

Мнемоника бита	Прерывание	Стандартный вектор
NMI	Немаскируемое прерывание	203EH
EXTINT	Вывод EXTINT	203CH
PI ††	Периферийное прерывание	203AH
COMP3	EPA сравнение канал 3	2032H
EPA3	EPA захват/сравнение канал 3	2030H

_____

† Биты 4–2 зарезервированы. Эти биты не определены.

Рисунок 6.11 – Регистр ожидания прерывания 1 (INT_PEND1)

PI_PEND Адрес: 1FBEH
Состояние Сброса: AAH

Когда аппаратные средства обнаруживают ожидаемые периферийное прерывание или прерывание таймера, они устанавливают соответствующий бит в регистрах ожидания прерывания (INT_PEND или INT_PEND1), (PI_PEND). Когда вектор выбран, аппаратные средства очищают соответствующий бит INT_PEND/INT_PEND1. Чтение этого регистра очищает все биты PI_PEND. Программное обеспечение может произвести прерывание установкой бита PI_PEND.

7							0
-	-	-	WG	-	OVRTM2	-	OVRTM1

Номер бита	Мнемоника бита	Функция
7, 6, 5, 3, 1	—	Резервные. Эти биты не определены.
4	WG	Генератор формы сигнала. Установка бита указывает, что ожидается прерывание.
2	OVRTM2	Переполнение/антипереполнение таймера 2. Установленный бит указывает на ожидаемое прерывание по переполнению/антипереполнению таймера 2. Прерывания по переполнению/антипереполнению таймера 2 и таймера 1 связаны с прерыванием по переполнению/антипереполнению таймера (OVRTM). Установка INT_MASK.0 позволяет OVRTM. Установка PI_MASK.2 позволяет OVRTM2.
0	OVRTM1	Переполнение/антипереполнение таймера 1. Установленный бит указывает на ожидаемое прерывание по переполнению/ антипереполнению таймера 1 и таймера 2. Они связаны с прерыванием по переполнению/антипереполнению таймера (OVRTM). Установка INT_MASK.0 позволяет OVRTM. Установка PI_MASK.0 позволяет OVRTM1.

Рисунок 6.12 – Регистр ожидания прерывания периферийных устройств

6.6 Инициализация блоков управления PTS

Каждое PTS прерывание запрашивает блок данных в регистровом ОЗУ, называемый блоком управления PTS (PTSCB). PTSCB определяет PTS микропрограмму, которая вызывает и устанавливает определенные параметры для программы. Необходимо установить PTSCB для каждого источника прерываний перед разрешением соответствующего PTS прерывания.

Адрес первого младшего байта PTSCB находится в таблице векторов PTS в области памяти специального назначения. Рисунок 6.13 описывает PTSCB для каждого режима PTS. Неиспользованные байты PTSCB могут использоваться как дополнительное ОЗУ.

Примечание – PTSCB должен быть расположен во внутреннем файле регистров. Ячейки первого байта PTSCB должны быть выровнены по границе слова кратного четырем.

	Одиночная передача	Передача блока	АЦП сканирование	SIO #1	SIO #2
	*)	*)	*)	PTSVEC1 (H)	*)
	*)	PTSBLOCK	*)	PTSVEC1 (L)	SAMPTIME
	PTSDST (H)	PTSDST (H)	PTSPTR2 (H)	BAUD (H)	DATA (H)
	PTSDST (L)	PTSDST (L)	PTSPTR2 (L)	BAUD (L)	DATA (L)
	PTSSRC (H)	PTSSRC (H)	PTSPTR1 (H)	EPAREG (H)	PTSCON1
	PTSSRC (L)	PTSSRC (L)	PTSPTR1 (L)	EPAREG (L)	PORTMASK
	PTSCON	PTSCON	PTSCON	PTSCON	PORTREG (H)
PTSVECT	PTSCOUNT	PTSCOUNT	PTSCOUNT	PTSCOUNT	PORTREG (L)

*) Неиспользуемый.

Рисунок 6.13 – Блоки управления PTS

Счетчик циклов PTS

Первая ячейка PTSCB, содержащая 8-битную величину, называется PTSCOUNT. Это значение определяет количество прерываний, которые обслуживаются процедурой PTS. Декремент PTS PTSCOUNT происходит после каждого цикла PTS. Когда PTSCOUNT достигает ноля, аппаратные средства очищают соответствующий PTSSEL бит и устанавливают PTSSRV бит (рисунок 6.6), который запрашивает прерывание end-of-PTS. Программа, обслуживающая end-of-PTS, должна повторно инициализировать PTSCB, если требуется, и устанавливать соответствующий PTSSEL бит, вновь разрешающий обслуживание прерывания PTS.

PTSSRV Адрес: 0006H
Состояние Сброса: 0000H

Регистр обслуживания PTS (PTSSRV) используется аппаратными средствами, чтобы указать, что последнее PTS прерывание обслужено программой PTS. Когда PTSCOUNT равен нолю, аппаратные средства очищают соответствующий PTSSEL бит и устанавливают PTSSRV бит, который запрашивает end-of-PTS прерывание. Когда прерывание end-of-PTS вызвано, аппаратные средства очищают PTSSRV бит.

15							8
—	EXYINT	PI	—	—	—	COMP3	EPA3
7							0
COMP2	EPA2	COMP1	EPA1	COMP0	EPA0	AD	OVRTM

15 Резервный. Этот бит не определен.

14:0 † Бит для соответствующего прерывания устанавливается аппаратно через стандартный вектор запроса прерывания end-of-PTS:

Мнемоника бит	Вектор PTS	Мнемоника бит	Вектор PTS
EXTINT	205CH	COMP1	204AH
PI	205AH	EPA1	2048H
COMP3	2052H	COMP0	2046H
EPA3	2050H	EPA0	2044H
COMP2	204EH	AD	2042H
EPA2	204CH	OVRTM	2040H

† Биты устройства 10–12 резервные. Эти биты не определены.

Рисунок 6.14 – Регистр обслуживания PTS (PTSSRV)

Выбор режима PTS

Второй байт каждого PTSCB – всегда 8-битная величина PTSCON. Биты 5–7 выбирают режим PTS (рисунок 6.15). Функция битов 0–4 отличается для каждого режима PTS. Таблица 6.4 содержит время выполнения цикла для каждого режима PTS.

PTSCON Адрес: PTSPCB + 1

Регистр управления PTS (PTSCON) выбирает режим PTS и устанавливает функции управления для этого режима.

7							0
M2	M1	M0	†	†	†	†	†

Номер бита	Мнемоника	Функция
7:5	M2:0	Режим PTS

Эти биты выбирают режим PTS:

M2	M1	M0	
0	0	0	передача блока
0	0	1	последовательный прием
0	1	0	зарезервирован
0	1	1	последовательная передача
1	0	0	однократная передача
1	0	1	зарезервирован
1	1	0	A/D сканирование
1	1	1	зарезервирован

† Функция этого бита зависит от выбранного режима.

Рисунок 6.15 – Режим выбора PTS (PTSCON биты 7–5)

Режимы одиночной передачи и блочной передачи

В однократном режиме передачи прерывание задает PTS передачу отдельного байта или слова, выбранного битом BW в PTSCON, из одной ячейки запоминающего устройства в другую. Этот режим типично используется с EPA, чтобы переместить захваченные значения времени из регистра времени события во внутреннее ОЗУ для дальнейшей обработки. На рисунке 6.16 приведен управляющий блок PTS для однократного режима передачи.

В однократном режиме передачи блок управления PTS содержит источник и адрес назначения (PTSSRC и PTSDST), регистр управления (PTSCON) и счетчик передач (PTSCOUNT).

	7							0
Неиспользуемый	0	0	0	0	0	0	0	0

	15							8
PTSDST (H)	Адрес назначения PTS (старший байт)							
	7							0
PTSDST (L)	Адрес назначения PTS (младший байт)							

	15							8
PTSSRC (H)	Адрес источника PTS (старший байт)							
	7							0
PTSSRC (L)	Адрес источника PTS (младший байт)							

	7							0
PTSCON	M2	M1	M0	BWZ	SU	DU	SI	DI

	7							0
PTSCOUNT	Количество передачи байт или слов							

Регистр	Адрес	Функция
PTSDST	PTSCB +4	Адрес назначения PTS. Содержит адрес ячейки запоминающего устройства назначения. Действительный адрес – любая незарезервированная ячейка запоминающего устройства; однако, это должен быть четный адрес, если выбрана передача слова.
PTSSRC	PTSCB +2	Адрес источника PTS. Адрес ячейки источника запоминающего устройства. Действительный адрес – любая незарезервированная ячейка запоминающего устройства; однако, необходимо указывать четный адрес, если выбрана передача слова.
PTSCON	PTSCB +1	Биты управления PTS
		M2:0 Режим PTS M2 M1 M0 1 0 0 однократный режим передачи
		BW Передача байта/слова; 0 = передача слова; 1 = передача байта.
		SU † Обновление PTSSRC; 0 = перезагружает оригинальный адрес источника PTS после того, как каждый байт или слово передано; 1 = сохраняет текущий адрес источника передачи PTS байта или передачи слова.

DU † Обновление PTSDST; 0 = перезагружает оригинальный адрес назначения PTS после того, как каждый байт или слово передано; 1= сохраняет текущий адрес источника передачи PTS байта или передачи слова.

SI † Автоинкремент PTSSRC; 0 = не увеличивает содержимое PTSSRC после того, как каждый байт или слово передано; 1 = увеличивает содержание PTSSRC после каждого байта или передачи слова.

DI † Автоинкремент PTSDST; 0 = не увеличивает содержимое PTSDST после того, как каждый байт или слово передано; 1 = увеличивает содержание PTSDST после каждого байта или передачи слова.

PTSCOUNT PTSCB +0 Последовательная передача слов или байтов. Определяют число слов или байтов, которые будут переданы в течение отдельной процедуры передачи. Каждая передача слова или байта - один цикл PTS. Максимальное значение – 255.

_____

† Биты DU/DI и биты SU/SI спарены в режиме однократной передачи. Каждая пара должна быть установлена или очищена вместе. Однако, эти две пары, DU/DI и SU/SI, не обязательно должны быть равны.

Рисунок 6.16 – Блок управления PTS – режим однократной передачи

PTSCB в таблице 6.5 определяет девять циклов PTS. Каждый цикл перемещает единственное слово из ячейки 20H во внешнюю ячейку запоминающего устройства. PTS передает первое слово в ячейку 6000H. Затем он увеличивает и обновляет адрес назначения и декрементирует регистр PTSCOUNT, но не увеличивает адрес источника. Когда второй цикл начинается, PTS перемещает второе слово из ячейки 20H в 6002H ячейку. Когда PTSCOUNT равняется нолю, PTS заполнит ячейку 6000-600FH и end-of-PTS прерывание генерируется.

Таблица 6.5 – Однократный режим передачи PTSCB

Неиспользованный
Неиспользованный
PTSDST (H) = 60H
PTSDST (L) = 00H
PTSSRC (H) = 00H
PTSSRC (L) = 20H
PTSCON = 85H (Режим = 100, BW= 0, SI/SU = 0, DI/DU = 1)
PTSCOUNT = 09H

Режим блочной передачи

В режиме блочной передачи прерывание заставляет PTS перемещать блок байтов или слов из одних ячеек запоминающего устройства в другие. На рисунке 6.17 показан блок управления PTS в режиме передачи блока.

В этом режиме каждый цикл PTS состоит из передачи определённого блока байтов или слов. Поскольку цикл PTS не может быть прерван, режим передачи блока может создать самое длительное время ожидания прерывания. Время ожидания в худшем случае может быть 500 циклов, если передача блока 32 слов из одних ячеек внешнего запоминающего устройства в другие использует 8-битную шину без циклов ожидания. См. таблицу 6.4 о времени выполнения циклов PTS.

PTSCB в таблице 6.6 настраивает три цикла PTS, которые передадут пять байтов от ячеек запоминающего устройства, 2024H в 60006004H (цикл 1), 60056009H (цикл 2) и 600A–600EH (цикл 3). Источники адреса и назначения увеличиваются после каждой передачи байта, но только адрес назначения обновляется в PTSSRC в конце каждого цикла передачи блока. В этой процедуре PTS всегда получает первый байт из ячейки 20H .

Таблица 6.6 – Режим передачи блока PTSCB

Неиспользованный
PTSBLOCK = 05H
PTSDST (H) = 60H
PTSDST (L) = 00H
PTSSRC (H) = 00H
PTSSRC (L) = 20H
PTSCON = 17H (Режим = 000; DI, SI, DU, BW = 1; SU = 0)
PTSCOUNT = 03H

В режиме передачи блока, блок управления PTS содержит размер блока (PTSBLOCK), источник и адрес назначения (PTSSRC и PTSDST), регистр управления (PTSCON) и счетчик передач (PTSCOUNT).

	7							0
Неиспользованный	0	0	0	0	0	0	0	0

	7							0
PTSBLOCK	Размер блока PTS							0

	15							8
PTSDST (H)	Адрес назначения PTS (старший байт)							
	7							0
PTSDST (L)	Адрес назначения PTS (младший байт)							

	15							8
PTSSRC (H)	Адрес источника PTS (старший байт)							
	7							0
PTSSRC (L)	Адрес источника PTS (младший байт)							

	7							0
PTSCON	M2	M1	M0	BW	SU	DU	SI	DI

	7							0
PTSCOUNT	Количество передач блока							

Регистр	Адрес	Функция
PTSBLOCK	PTSCB +6	Размер блока PTS. Определяет количество байтов или слов в каждом блоке. Действительные значения 1–32.
PTSDST	PTSCB +4	Адрес назначения PTS. В этот регистр записывается адрес ячейки назначения запоминающего устройства. Действительный адрес - любая нерезервированная ячейка запоминающего устройства; однако, это должен быть четный адрес, если выбрана передача слова.

PTSSRC	PTSCB +2	Адрес источника PTS. В этот регистр записывается адрес ячейки источника запоминающего устройства. Действительный адрес – любая незарезервированная ячейка запоминающего устройства; однако, это должен быть четный адрес, если выбрана передача слова.
PTSCON	PTSCB +1	Биты управления PTS
	M2:0	Режим PTS. Эти биты выбирают режим PTS: M2 M1 M0 0 0 0 режим передачи блоков.
	BW	Передача байта/слова (0 = передача слова; 1 = передача байта)
	SU	Перезагрузка PTSSRC. 0 = перезагружает первоначальный адрес источника PTS после того, как каждая передача блока завершена. 1 = сохраняет текущий адрес источника PTS после того, как каждая передача блока завершена.
	DU	Перезагрузка PTSDST. 0 = перезагружает первоначальный адрес назначения PTS после того, как каждая передача блока завершена 1 = сохраняет текущий адрес назначения PTS после того, как каждая передача блока завершена.
	SI	Автоинкремент PTSSRC. 0 = не увеличивает содержание PTSSRC после того, как каждый байт или слово переданы; 1 = увеличивает содержание PTSSRC на «1» после каждого байта или передачи слова
	DI	Автоинкремент PTSDST. 0 = не увеличивает содержание PTSDST после того, как каждый байт или слово переданы; 1 = увеличивает содержание PTSDST на «1» после каждого байта или передачи слова
PTSCOUNT	PTSCB +0	Последовательная передача блоков. Определяет число блоков, которые будут переданы в течение процедуры поблочной передачи. Передача каждого блока – один цикл PTS. Максимальное число – 255.

Рисунок 6.17 – Блок управления PTS – режим передачи блока

Режим АЦП сканирования

В режиме АЦП сканирования PTS заставляет АЦП исполнять многократные преобразования на одном или более каналах и затем хранит результаты в таблице в запоминающем устройстве. На рисунке 6.18 показан блок управления PTS в режиме АЦП сканирования.

Блок управления содержит указатели для регистра AD_RESULT (PTSPTR1) и таблицы команд АЦП преобразования и результатов (PTSPTR2), регистр управления (PTSCON) и счетчик АЦП преобразований (PTSCOUNT).

	7							0
Неиспользованный	0	0	0	0	0	0	0	0

	7							0
Неиспользованный	0	0	0	0	0	0	0	0

	15							8
PTSPTR2 (H)	Указатель 2 значение (старший байт)							
	7							0
PTSPTR2 (L)	Указатель 2 значение (младший байт)							

	15							8
PTSPTR1 (H)	Указатель 1 значение (старший байт)							
	7							0
PTSPTR1 (L)	Указатель 1 значение (младший байт)							

	7							0
PTSCON	M2	M1	M0	0	UPDT	0	1	0

	7							0
PTSCOUNT	Количество циклов АЦП преобразований							

Регистр	Адрес	Функция
PTSPTR2	PTSCB +4	Значение Указателя 2. Этот регистр содержит адрес регистра результата АЦП преобразования (AD_RESULT).
PTSPTR1	PTSCB +2	Значение Указателя 1. Этот регистр содержит адрес таблицы АЦП команд и результаты преобразований.
PTSCON	PTSCB +1	Биты управления PTS M2:0 Режим PTS. Эти биты выбирают режим PTS. M2 M1 M0 1 1 0 режим сканирования A/D UPDT Коррекция. 0 = перезагружает начальное значение PTSPTR1 после каждого A/D сканирования; 1 = сохраняют текущее значение PTSPTR1 после каждого сканирования A/D.
PTSCOUNT	PTSCB +0	Последовательное АЦП преобразование. Определяет число АЦП преобразований, которые будут закончены в течение АЦП сканирования. Каждый цикл состоит из передачи результатов АЦП преобразований из PTS в таблицу команд/данных и загрузки новой команды в регистр AD_COMMAND. Максимальный номер – 255.

Рисунок 6.18 – Блок управления PTS – режим АЦП сканирования

Используя режим АЦП-сканирования, необходимо установить таблицу команд/данных в запоминающем устройстве (таблица 6.7). Таблица команд/данных содержит АЦП команды, которые записываются в свободные ячейки запоминающего устройства. PTS хранит результаты преобразований в этих свободных ячейках. Только объем свободных ячеек запоминающего устройства ограничивает размер таблицы; они могут храниться во внутреннем или внешнем ОЗУ.

Таблица 6.7 – Таблица команд/данных режима АЦП сканирования

Адрес	Содержание	
XXXX + AH	A/D результат 2	
XXXX + 8H	Неиспользованный	A/D команда 3 †
XXXX + 6H	A/D результат 1	
XXXX + 4H	Неиспользованный	A/D команда 2
XXXX + 2H	A/D результат 0 ††	
XXXX	Неиспользованный	A/D команда 1

† Записывается 0000H, чтобы предотвратить новое преобразование в конце процедуры.

†† Результат A/D преобразования, которое инициализировало процедуру PTS.

Чтобы начать режим АЦП сканирования, разрешают прерывание по законченному циклу и разрешают его обслуживание через PTS. Программное обеспечение должно начать первое преобразование. Когда АЦП заканчивает первое преобразование и генерирует прерывание завершения АЦП преобразования, прерывание адресуется к PTSCB и инициирует процедуру АЦП сканирования. PTS хранит результаты преобразования, загружает новую команду в AD_COMMAND и декрементирует значение в PTSCOUNT. При каждом повторном PTS прерывании повторяется цикл просмотра A/D; он сохраняет результаты преобразования, загружает команду следующего преобразования в регистр AD_COMMAND и декрементирует PTSCOUNT. Процедура продолжается до тех пор, пока PTSCOUNT не уменьшится до нуля. Когда это произойдёт, аппаратные средства очистят бит разрешения в регистре PTSSEL, который запрещает обслуживание PTS, и установят бит PTSSRV, который запрашивает end-of-PTS прерывание.

PTS циклы в режиме АЦП сканирования

Программное обеспечение начинает первое A/D преобразование. После того, как прерывание по завершению A/D инициализирует программу PTS, выполняется следующая последовательность инструкций.

1 PTS читает первую команду (из адреса XXXX), загружает ее во временную ячейку и инкрементирует регистр PTSPTR1 дважды. PTSPTR1 теперь указывает на первую свободную ячейку в таблице команд/данных (адрес XXXX + 2).

2 PTS читает регистр AD_RESULT, хранит результаты первого преобразования в ячейке (XXXX + 2) в таблице команд/данных и увеличивает регистр PTSPTR1 на два. PTSPTR1 теперь указывает на (XXXX + 4).

3 PTS загружает команду из временной ячейки в регистр AD_COMMAND. Это завершает первый цикл АЦП сканирования и начинает следующее АЦП преобразование.

4 Если UPDT (PTSCON.3) очищен, первоначальный адрес перезагружается в регистр PTSPTR1. Следующий цикл использует ту же самую команду и перезаписывает предыдущие данные. Если UPDT установлен, то обновленный адрес остается в PTSPTR1, и следующий цикл использует новую команду и хранит преобразованные результаты в новом адресе.

5 Содержимое PTSCOUNT декрементируется, и центральный процессор возобновляет выполнение программы. Когда формируется следующее прерывание по завершению АЦП преобразования, цикл повторяется. Когда PTSCOUNT достигает нуля, аппаратные средства очищают бит PTSSEL и устанавливают бит PTSSRV, который формирует запрос на прерывание end-of-PTS.

Режим АЦП сканирования (пример 1)

Таблица команд/данных, показанная в таблице 6.8, задаёт ряд АЦП-преобразований, начиная с канала 7 и заканчивая каналом 4. Каждый вход в таблицу – слово (два байта). В таблице 6.9 приведен соответствующий PTSCB.

Программное обеспечение начинает преобразование с канала 7. После завершения преобразования прерывание по завершению АЦП-преобразования инициализирует программу PTS сканирования. Шаг 1 записывает команды из канала 6 во временную ячейку и увеличивает PTSPTR1 до 3002H. Шаг 2 записывает результат преобразований из канала 7 в ячейку 3002H и увеличивает PTSPTR1 до 3004H. Шаг 3 загружает содержимое канала 6 из временной ячейки в AD_COMMAND, чтобы начать следующее преобразование. Шаг 4 обновляет PTSPTR1 (PTSPTR1 теперь указывает на 3004H), и шаг 5 увеличивает PTSCOUNT до 3. Следующий цикл начинается с сохранения содержимого канала 5 команд во временной ячейке. В течение последнего цикла (PTSCOUNT = 1) «пустая» команда загружается в регистр AD_COMMAND, и преобразование не выполняется. PTSCOUNT – уменьшается до ноля, и происходит запрос на end-of-PTS прерывание.

Таблица 6.8 – Таблица команд/данных (пример 1)

Адрес	Содержание	
300EH	AD_RESULT для ACH4	
300CH	Не используется	0000H (холостая команда)
300AH	AD_RESULT для ACH5	
3008H	Не используется	AD_COMMAND для ACH4
3006H	AD_RESULT для ACH6	
3004H	Не используется	AD_COMMAND для ACH5
3002H	AD_RESULT для ACH7	
3000H	Не используется	AD_COMMAND для ACH6

Таблица 6.9 – Режим АЦП сканирования PTSCB (пример 1)

Не используется
Не используется
PTSPTR2 (H) = 1FH
PTSPTR2 (L) = AAH
PTSPTR1 (H) = 30H
PTSPTR1 (L) = 00H
PTSCON = CBH (Режим = 110, UPDT = 1)
PTSCOUNT = 04H

Режим АЦП сканирования (пример 2)

Таблица 6.11 устанавливает серию из десяти циклов PTS, каждый из которых читает один канал АЦП и записывает результат в ячейку (3002H). UPDT бит (PTSCON.3) очищается так, чтобы после цикла было восстановлено первоначальное содержание PTSPTR1. Таблица команд/данных представлена в таблице 6.10.

Таблица 6.10 – Таблица команд/данных (пример 2)

Адрес	Содержание	
3002H	AD_RESULT for ACHx	
3000H	Не используется	AD_COMMAND для ACHx

Таблица 6.11 – Режим просмотра A/D PTSCB (пример 2)

Не используется
Не используется
PTSPTR2 (H) = 1FH
PTSPTR2 (L) = AAH
PTSPTR1 (H) = 30H
PTSPTR1 (L) = 00H
PTSCON = C3H (Режим = 110, UPDT = 0)
PTSCOUNT = 0AH

Программа начинает преобразование на канале (x). Когда преобразование закончено и сгенерировано прерывание завершения АЦП преобразования, начинается режим АЦП сканирования. PTS считывает команду в ячейке 3000H и записывает её во временную ячейку. Затем увеличивает регистр PTSPTR1 на два и записывает значение регистра AD_RESULT в ячейку 3002H. Заключительный шаг – копирование команды преобразования из временной ячейки в регистр AD_COMMAND. Центральный процессор обработает или перенесет данные результатов преобразования из таблицы прежде, чем следующее преобразование закончится и начнется новый цикл PTS. Когда начинается следующий цикл, PTSPTR1 снова обращается в 3000H, и повторяются шаги первого цикла. Значение регистра AD_RESULT записывается в ячейку 3002H, и команда из ячейки 3000H выполняется снова.

Режим последовательного ввода-вывода

Микроконтроллер не имеет последовательного порта ввода-вывода, но режимы последовательного ввода-вывода PTS обеспечивают программируемый канал последовательного ввода-вывода для синхронной и асинхронной передачи и приема. Есть четыре основных режима работы: синхронная передача, синхронный прием, асинхронная передача и асинхронный прием. Стандартный сигнал порта ввода-вывода конфигурируется как для передачи данных (TXD), так и для получения данных (RXD). Канал EPA устанавливает скорость двоичной передачи (baud rate). Возможно конфигурирование любого сигнала порта 2 как TXD или как RXD. В синхронных режимах канал EPA может или производить последовательные синхроимпульсы (SCK) сигнала, или использовать внешний синхросигнал. Может быть передано и принято до 16 битов, включая биты четности и останова в асинхронных режимах. Последовательные режимы ввода-вывода требуют два блока управления PTS для формирования всех вариантов (см. рисунки 6.19 и 6.20). Эти блоки не обязательно должны быть смежными, но каждый из них должен располагаться в границах четырех слов в регистровом файле.

Блок управления 1 PTS в режиме последовательного ввода-вывода

Блок управления 1 PTS содержит указатели на второй блок управления PTS (PTSVEC) и регистр времени EPA, который устанавливает скорость двоичной передачи (EPAREG). Он содержит также 16-битную величину, используемую для вычисления скорости двоичной передачи – регистр управления (PTSCON) и счетчик последовательных циклов PTS (PTSCOUNT).

	15							8
PTSVEC (H)	SIO PTSCB2 Указатель основного адреса (старший байт)							
	7							0
PTSVEC (L)	SIO PTSCB2 Указатель основного адреса (младший байт)							

	15							8
BAUD (H)	Значение BAUD (старший байт)							
	7							0
BAUD (L)	Значение BAUD (младший байт)							

	15							8
EPAREG (H)	Адрес регистра EPA (старший байт)							
	7							0
EPAREG (L)	Адрес регистра EPA (младший байт)							

	7							0
PTSCON	M2	M1	M0	SA1	0	0	SA0	MAJ

	7							0
PTSCOUNT	Последовательные циклы PTS (счётчик)							

Регистр	Адрес	Функция
PTSVEC	PTSCB1 + 6	Указатель базового адреса SIO PTSCB2. Этот регистр содержит основной адрес второго блока управления PTS для последовательного режима ввода-вывода.
BAUD	PTSCB1 + 4	Значение BAUD. Этот регистр содержит 16-битную величину, используемую PTS для задания желаемой скорости передачи. Формула для вычисления значения для загрузки регистра BAUD: $\text{Baud\_value} = \frac{F_{BQ1}}{\text{Multiplier}(\text{Baud\_rate} \times \text{EPA\_prescale})}$, где Baud_value - целое 16-битное число, которое загружено в регистр BAUD; F_{BQ1} - входная частота на BQ1, в МГц; Multiplier = 4 в асинхронных режимах и = 8 в синхронных режимах; Baud_rate - скорость передачи, в битах в секунду; EPA_prescale – число предварительного деления частоты таймера EPA, 1–64.
EPAREG	PTSCB1 + 2	Адрес регистра времени EPA. Этот регистр содержит 16-битный адрес регистра COMPx_TIME или EPAx_TIME.
PTSCON	PTSCB1 + 2	Биты управления PTS M2:0 Режим M2 M1 M0 0 0 1 PTS SIO режим приема 0 1 1 SIO режим передачи

SA1:0 Асинхронный/синхронный режим.

SA1	SA0 †	
0	0	разрешает асинхронные режимы последовательного ввода-вывода
1	1	разрешает синхронные режимы последовательного ввода-вывода

MAJ Мажоритарная выборка.
0 = запрещает мажоритарную выборку в асинхронном режиме приёма; всегда очищен во всех других режимах;
1 = позволяет мажоритарную выборку в асинхронном режиме приёма.

PTSCOUNT PTSCB1 + 0 Последовательные циклы PTS.
Определяет число битов, которые будут переданы или получены, включая биты четности и останова, но без старт-бита. Для асинхронных режимов программируют значения от 1 до 16. Для синхронных режимов программируют удвоенное количество передаваемых (принимаемых) битов – от 2 до 32. PTSCOUNT декрементируется в конце каждого цикла PTS. Когда он достигает нуля, аппаратные средства очищают соответствующий бит PTSSEL и устанавливают бит PTSSRV, который разрешает end-of-PTS прерывание.

† В оба бита всегда заносят одинаковые значения.

Рисунок 6.19 – Режим управления блоком последовательного ввода - вывода PTS1

Блок управления 2 PTS в режиме последовательного ввода-вывода

Блок управления 2 PTS содержит указатели и на регистр порта (PORTREG), и на регистр данных (DATA). Он также содержит 16-битную величину, которая используется для вычисления времени выборки для асинхронного приема при мажоритарной выборке (SAMPTIME), регистр управления (PTSCON1) и 16-битное значение, используемое для выбора разряда порта в качестве TXD или RXD сигнала (PORTMASK).

	7							0
Неиспользованный	0	0	0	0	0	0	0	0

	7							0
SAMPTIME	Значение времени выборки							

	15							8
DATA (H)	Регистр данных (старший байт)							
	7							0
DATA (L)	Регистр данных (младший байт)							

	7							0
PTSCON1 (синхронный)	0	0	0	0	0	0	TRC	0

	7							0
PTSCON1 (асинхронный)	0	RPAR	PEN	0	0	0	FE	TPAR

	7							0
PORTMASK	Регистр маски порта							

	15							8
PORTREG (H)	Указатель адреса порта (старший байт)							
	7							0
PORTREG (L)	Указатель адреса порта (младший байт)							

Регистр	Адрес	Функция
SAMPTIME	PTSCB2 + 6	Значение времени выборки. Этот регистр управляет временем между выборками в течение режима асинхронного приема при мажоритарной выборке. Для вычисления значения, загружаемого в регистр SAMPTIME, используют следующую формулу. $\text{Sample\_time} = \frac{T_{SAM} \times F_{BQ1}}{2} - 9$, где Sample_time – целое число 1–31, загружаемое в регистр SAMPTIME, T_{SAM} - желательное время между выборками, в мкс, F_{BQ1} - входная частота на BQ1, в МГц.
DATA	PTSCB2 + 4	Регистр данных. 16-битный регистр содержит данные для передачи или данные, которые будут приняты. Во время передачи младший (0-й) бит передается первым. Данные сдвигаются вправо с каждой успешной передачей. В течение приема первый бит загружается в старший (15-й) разряд. Данные сдвигаются вправо с каждым успешным приемом.

PTSCON1	PTSCB2 + 3	Биты управления PTS. Синхронный режим TRC Управление режимом передачи/приема. 0 = передают или получают данные в течение четных циклов PTS; 1 = передают или получают данные в течение нечетных циклов PTS. Этот бит инициализируют при старте каждой передачи или приема. Асинхронный режим RPAR Статус и контроль четности при приеме. Инициализируется перед началом приема. 0 = бит TPAR установлен для проверки на четность; 1 = бит TPAR очищен для проверки на нечетность. PEN Разрешение контроля чётности. 0 = запрет контроля чётности, 1 = разрешение контроля чётности. FE Флаг ошибки кадрирования. 0 = бит останова 1, 1 = бит останова 0. Очищать бит в начале каждого приема. TPAR Управление чётностью передачи. 0 = четность, 1 = нечетность
PORTMASK	PTSCB2 + 2	Регистр маски порта. Выбирает сигнал порта, который будет функционировать на передачу данных (TXD) или на прием данных (RXD), установкой соответствующего бита.
PORTREG	PTSCB2 + 0	Указатель адреса порта. 16 битный регистр, содержащий адрес порта, который будет использоваться при приеме и передаче данных.

Рисунок 6.20 – Блок управления 2 PTS режима последовательного ввода-вывода

Пример режима синхронной передачи

В синхронном режиме передачи (SSIO) канал EPA управляет скоростью двоичной передачи, генерируя или захватывая сигнал синхроимпульсов (SCK). Чтобы генерировать SCK сигнал, канал EPA конфигурируется в режиме сравнения, и выставляется опция переключения вывода. Если внешний источник задаёт SCK сигнал, канал EPA конфигурируется в режиме захвата по каждому фронту. В этом случае EPA канал генерирует прерывание по каждому переключению сигнала SCK. По каждому прерыванию EPA периферийный сервер (PTS) сдвигает бит данных на выход порта, который сконфигурирован как TXD. PTSCON1 (рисунок 6.19) задаёт передачу данных в чётных или нечётных циклах PTS. Поскольку передачи происходят только по переднему или заднему фронту синхроимпульсов, два цикла PTS требуются для передачи каждого бита данных (рисунок 6.21). Требуется 16 циклов PTS, чтобы передать восемь битов данных. В SSIO режиме передачи только биты данных могут быть переданы; биты чётности и стоп-биты исключены.

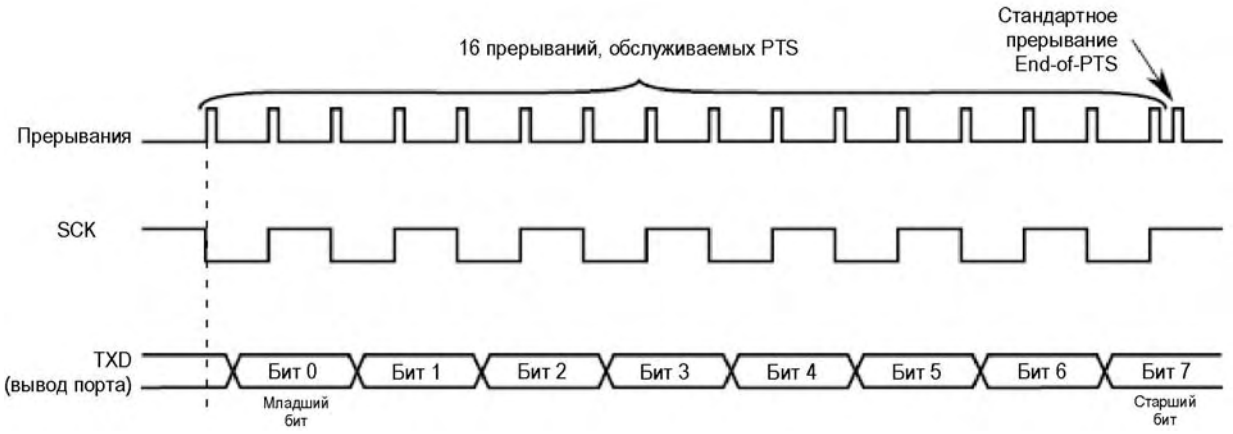


Рисунок 6.21– Временные диаграммы синхронной последовательной передачи

Если SCK сигнал генерирован каналом EPA, первый цикл PTS должен быть начат следующим образом:

- Вывод TXD порта и SCK сигнал инициализируются к требуемому системой логическому уровню перед началом передачи.
- Добавляется содержание регистра таймера к Baud_value (рисунок 6.19) и результат сохраняется в регистре времени EPA. Это устанавливает время первого прерывания и передача первого разряда осуществляется с нужной двоичной скоростью.

Следующий пример использует EPA0 для управления скоростью двоичной передачи и выводом SCK сигнала и P2.2 для вывода данных (TXD). Он устанавливает процедуру синхронного последовательного ввода-вывода PTS, которая передает 16 байтов с восемью битами данных со скоростью 9600 бод. Этот пример использует несколько определенных пользователем регистров. T_COUNT определяет количество байтов для передачи, а TXDDONE – флаг, который устанавливается, когда все байты переданы.

1. Запрет прерываний и PTS.

– Используется инструкция DI для запрета стандартных прерываний и инструкция DPTS для запрета PTS.

2. Установка указателя стека.
3. Сброс всех регистров маски прерываний.

– Очистка INT_MASK, INT_MASK1 и PI_MASK.

4. Инициализация вывода P2.0 как выход EPA0 (SCK) и P2.2 как TXD.

– Очистка P2_DIR (выбор вывода).

– Установка P2_MODE.0 (выбор специальной функции).

– Очистка P2_MODE.2 (выбор функции LSIO).

– Установка битов 0 и 2 P2_REG инициализирует выводы TXD и SCK в “1”.

5. Инициализация и разрешение таймера. Выбирают прямой счет, внутреннюю синхронизацию и запрещают предварительное деление частоты.

– Установка битов 6 и 7 T1CONTROL (рисунок 10.8).

6. Инициализация PTSCB как показано в таблице 6.13.

Таблица 6.13 – SSIO режим передачи PTSCBs

PTSCB1	PTSCB2
PTSVEC (H) = указатель на PTSCB2	Неиспользованный
PTSVEC (L) = указатель на PTSCB2	SAMPTIME = неиспользованный
BAUD (H) = 00H (9600 бод в 16 МГц)	DATA (H) = неиспользованный
BAUD (L) = D0H (9600 бод в 16 МГц)	DATA (L) = nnH (8 битов данных)
EPAREG (H) = 1FH (EPA0_TIME)	PTSCON1 = 02H (передача данных по нечетным циклам PTS)
EPAREG (L) = 42H (EPA0_TIME)	PORTMASK = 04H (P2.2 = TXD)
PTSCON = 72H (режим передачи SSIO)	PORTREG (H) = 1FH (P2_REG)
PTSCOUNT = 10H (8 бит данных × 2)	PORTREG (L) = D4H (P2_REG)

7. Разрешение EPA0 прерывания

– Установка INT_MASK.2.

8. Загрузка количества передаваемых байтов в счетчик T_COUNT и очистка флага окончания передачи (TXDDONE).

– LD T_COUNT, #16.

– CLRB TXDDONE.

9. Выбор PTS обслуживания для EPA0.

– Установка PTSSEL.2.

10. Установка EPA0 как канала только сравнения.

– Установка EPA0_CON.6.

11. Начать операции канала EPA0, записав время первого прерывания в EPA0_TIME. Для установки корректного значения прибавляют baud_value (0D0H) к текущему значению TIMER1 и сохраняют результат в EPA0_TIME. Baud_value определяет время первого PTS прерывания и первое переключение сигнала SCK. PTS передает первый бит данных по первому фронту SCK в этом примере. Baud_value 0D0H выбирает скорость двоичной передачи 9600 бод.

12. Разрешение PTS и стандартных прерываний.

– Используют инструкцию EI для разрешения всех стандартных прерываний и инструкцию EPTS для разрешения PTS прерывания.

13. Начало передачи. Данные сдвигаются, начиная с младшего (самого правого) бита. Каждый раз, когда происходит совпадение между EPA0_TIME и TIMER1, канал EPA0 производит прерывание и переключает SCK сигнал. PTS выводит следующий бит данных на вывод, функционирующий как TXD по нечетным циклам PTS. Когда PTSCOUNT уменьшится до нуля, PTS вызывает end-of-PTS прерывание (рисунок 6.22). Процедура обслуживания прерывания запрещает канал EPA, потому что в конце цикла PTS следующее время переключения SCK загружается в регистр времени события. Если переключение происходит, полярность синхроимпульсов изменится из-за нечетного числа переключений. Процедура обслуживания прерывания должна загрузить следующий байт данных, перезагрузить PTSCOUNT и PTSCON1, выбрать обслуживание PTS для EPA0 и регистров EPA0_TIME и EPA0_CONTROL.

14. Чтобы определить, когда все байты переданы, надо создать циклическую подпрограмму, чтобы проверить состояние TXDDONE флага.

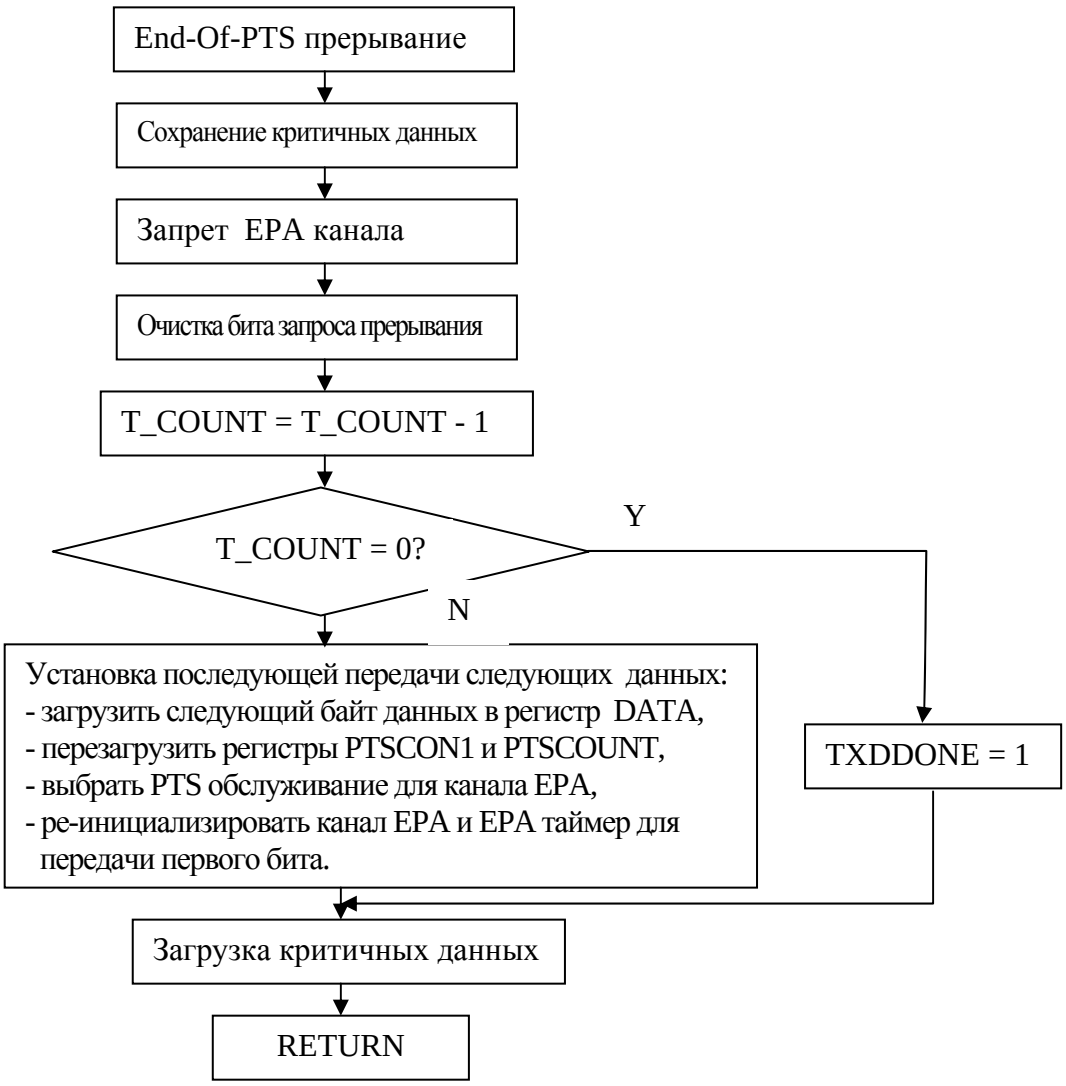


Рисунок 6.22 – Режим синхронного последовательного вывода. Блок-схема прерывания end-of-PTS

Пример режима синхронного приема

В режиме синхронного приёма данных канал EPA управляет двоичной скоростью приема, генерируя или захватывая синхроимпульсы (сигнал SCK). Для генерации SCK сигнала канал EPA формируется для сравнения. Всякий раз, когда происходит совпадение между регистром события-времени EPA и регистром таймера, канал EPA переключает SCK и производит прерывание. Если SCK сигнал обеспечивается внешним источником, канал EPA конфигурируется в режиме захвата с захватом по любому фронту. По каждому следующему EPA прерыванию PTS принимает бит данных со входа порта, функционирующего как RXD. PTSCON1 (рисунок 6.19) управляет режимом приема (по четным или нечетным циклам PTS). Поскольку прием происходит только по передним или задним фронтам синхроимпульсов, два цикла PTS требуется для приема каждого бита данных (рисунок 6.23). Требуется 16 циклов PTS, чтобы получить восемь битов данных. SSIO прием не включает биты чётности или стоп-биты.

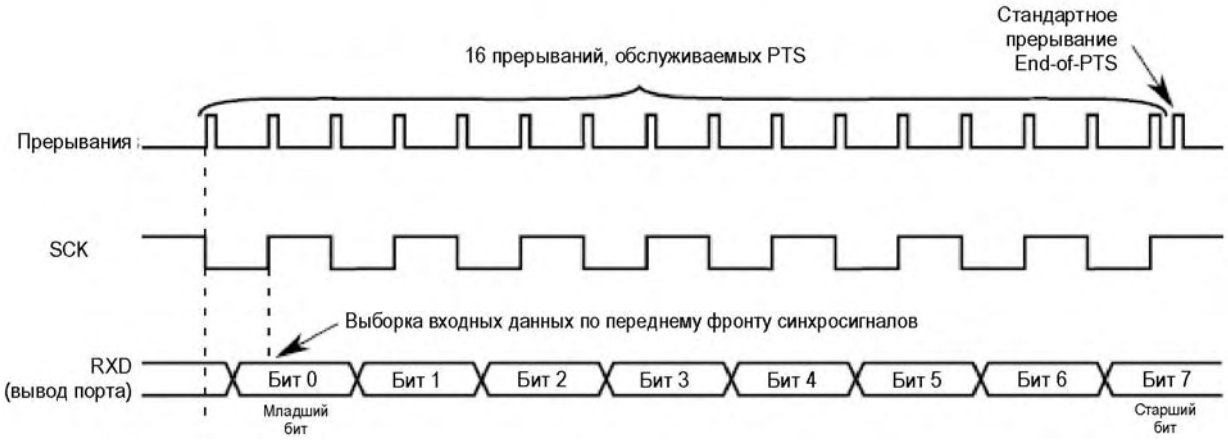


Рисунок 6.23 – Временные диаграммы синхронного последовательного приёма

Если SCK сигнал генерируется каналом EPA, первый цикл PTS должен быть начат следующим образом:

- Вход RXD и сигнал SCK инициализируются к требуемому системой логическому уровню.
- Прибавить содержимое регистра таймера к Baud_value (рисунок 6.19) и сохранить результат в регистре времени EPA. Это устанавливает время для первого прерывания и появление первого прерывания в соответствии со скоростью приёма данных.

В следующем примере EPA0 используется для захвата сигнала SCK, а P2.3 (RXD)_ – для получения данных. Процедура синхронного последовательного ввода PTS настроена на приём 16 байтов с восемью битами данных. Поскольку этот пример использует внешний последовательный ввод синхроимпульсов, TIMER1 и регистр BAUD не используются. Внешний источник синхроимпульсов управляет скоростью приёма. В этом примере используется несколько определяемых пользователем регистров. R_COUNT определяет количество принимаемых байтов, а RXDDONE – флаг, который устанавливается после получения всех байтов.

1. Запретить прерывания и PTS.

– Использовать инструкцию DI, чтобы запретить все стандартные прерывания и инструкцию DPTS, чтобы запретить PTS.

2. Установить указатель стека.

3. Сбросить все регистры маски прерывания.

– Очистите INT_MASK, INT_MASK1 и PI_MASK.

4. Инициализировать P2.0 для функционирования как вход EPA0 (SCK) и P2.3 для функционирования как RXD.

– Установить биты 0 и 3 P2_DIR (выбирает ввод).

– Установить P2_MODE.0 (выбирает специальную функцию).

– Очистить P2_MODE.3 (выбирает функцию LSIO).

– Установить биты 0 и 3 P2_REG (инициализирует SCK и RXD в “1”).

5. Инициализировать PTSCB как показано в таблице 6.14.

Таблица 6.14 –Режим SSIO приёма PTSCBs

PTSCB1	PTSCB2
PTSVEC (H) = указатель на PTSCB2	Неиспользованный
PTSVEC (L) = указатель на PTSCB2	SAMPTIME = неиспользованный
BAUD (H) = неиспользованный	DATA (H) = неиспользованный
BAUD (L) = неиспользованный	DATA (L) = 00H (очистка регистра, чтобы получить данные)
EPAREG (H) = 1FH (EPA0_TIME)	PTSCON1 = 00H (получение данных по чётным циклам PTS)
EPAREG (L) = 42H (EPA0_TIME)	PORTMASK = 08H (P2.3 = RXD)
PTSCON = 32H (режим SSIO приёма)	PORTREG (H) = 1FH (P2_REG)
PTSCOUNT = 10H (8 бит данных x 2)	PORTREG (L) = D4H (P2_REG)

6. Разрешить прерывание EPA0.

– Установить INT_MASK.2.

7. Загрузить число байтов к передаче в определённый пользователем регистр счета передачи (R_COUNT) и очистить определенный пользователем флаг завершенного приема (RXDDONE).

– LD R_COUNT, #16.

– CLRB RXDDONE.

8. Выбрать PTS-сервис для EPA0.

– Установить PTSSEL.2.

9. Установить EPA0 для захвата и передних, и задних фронтов.

– Установить биты 4 и 5 EPA0_CON (рисунок 10.10).

10. Установить PTS и стандартные прерывания.

– Использовать инструкцию EI для разрешения всех стандартных прерываний и EPTS инструкцию для разрешения PTS.

11. Переключить вход SCK, чтобы начать прием. Данные перемещаются в наиболее значащий (крайний левый) бит и сдвигаются вправо с каждым полученным битом. EPA вырабатывает прерывание при каждом переключении на входе SCK. PTS получает следующий бит данных на вход, сконфигурированный как RXD, в четных циклах PTS. Когда PTSCOUNT декрементируется до нуля, PTS вызывает end-of-PTS прерывание (рисунок 6.24). Процедура обслуживания прерываний должна запретить канал EPA, очистить регистр данных (DATA), перезагрузить регистры PTSCOUNT и PTSCON1, перезагрузить EPA0_CON и выбрать обслуживание PTS для EPA0. Если EPA генерирует сигнал SCK, процедура обслуживания end-of-PTS прерывания также должна будет перезагрузить регистр EPA0_TIME.

12. Чтобы определить, когда все байты были переданы, надо создать циклическую подпрограмму, чтобы проверить состояние RXDDONE флага.

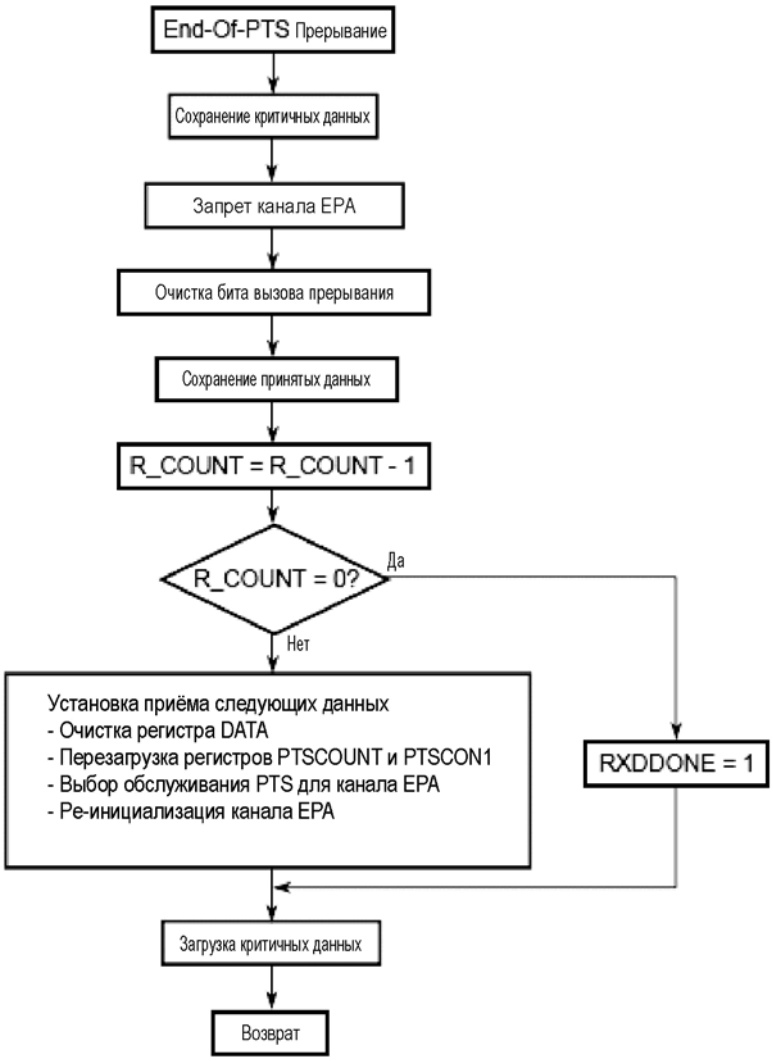


Рисунок 6.24 – Режим синхронного SIO приёма. Блок-схема процедуры прерывания end-of-PTS

Пример режима асинхронной SIO передачи

В режиме асинхронной передачи (ASIO) канал EPA управляет скоростью двоичной передачи, вырабатывая прерывание всякий раз, когда происходит совпадение между регистром времени события EPA и регистром таймера. PTS сдвигает бит данных на вывод порта, сконфигурированного для функционирования как сигнал передачи данных (TXD), когда выбранный канал EPA производит прерывания сравнения (рисунок 6.25). В режиме передачи ASIO PTS автоматически передает до 16 битов (данные + 1 опциональный бит четности + 1 стоп-бит). Максимальное число битов данных – 14 с паритетом или 15 без.

Рисунок 6.25 – Временная диаграмма асинхронной передачи SIO

Первый цикл PTS должен быть начат путем генерации стартового бита с последующей установкой времени для первого EPA прерывания.

- Инициализировать вывод порта TXD в единицу перед стартом передачи.
- Записать нуль в вывод порта TXD, чтобы начать передачу. PTS использует это как стартовый бит.
- Добавить содержимое регистра таймера к Baud_value (рисунок 6.19), и сохранить результат в регистре времени EPA. Это устанавливает время для первого прерывания и задаёт надлежащую скорость передачи первого бита.

Следующий пример использует P2.0 как вывод данных (TXD), а EPA0 для управления скоростью двоичной передачи. Он устанавливает процедуру асинхронного последовательного вывода PTS, которая передает 16 байтов с восемью битами данных, одним паритетным битом и одним стоп-битом со скоростью 9600 бод. В этом примере используется несколько определяемых пользователем регистров. T_COUNT определяет число байтов для передачи, а TXDDONE – флаг, который устанавливается, когда все байты переданы.

1. Запретить прерывания и PTS.

– Использовать инструкцию DI, чтобы запретить все стандартные прерывания и инструкцию DPTS, чтобы запретить PTS.

2. Установить указатель стека.
3. Сбросить все регистры маски прерывания. Очистить INT_MASK, INT_MASK1 и PI_MASK.
4. Инициализировать P2.0 как TXD.

– Очистить P2_DIR.0 (выбор вывода).

– Очистить P2_MODE.0 (выбор функции LSIO).

– Установить P2_REG.0 (инициализирует TXD вывод в “1”).

5. Инициализировать и разрешить таймер; выбрать прямой счет, внутренние синхроимпульсы и запретить предделитель.

– Установить биты 6 и 7 T1CONTROL (рисунок 10.8).

6. Инициализировать PTSCB как показано в таблице 6.15.

Таблица 6.15 – Режим передачи ASIO PTSCB

PTSCB1	PTSCB2
PTSVEC (H) = указатель на PTSCB2	Неиспользуемый
PTSVEC (L) = указатель на PTSCB2	SAMPTIME = неиспользуемый
BAUD (H) = 01H (9600 бод в 16 МГц)	DATA (H) = неиспользуемый
BAUD (L) = A0H (9600 бод в 16 МГц)	DATA (L) = nnH (8 битов данных)
EPAREG (H) = 1FH (EPA0_TIME)	PTSCON1 = 21H (разрешить контроль нечетности)
EPAREG (L) = 42H (EPA0_TIME)	PORTMASK = 01H (P2.0 = TXD)
PTSCON = 60H (ASIO режим передачи)	PORTREG (H) = 1FH (P2_REG)
PTSCOUNT = 0AH (8 битов данных, 1 бит четности и 1 стоп-бит)	PORTREG (L) = D4H (P2_REG)

7. Разрешить прерывание EPA0.

– Установить INT_MASK.2.

8. Загрузить число байтов для передачи в определённый пользователем регистр счета передачи (T_COUNT), и очистить определенный пользователем флаг окончания передачи (TXDDONE).

– LD T_COUNT, #16

– CLRB TXDDONE.

9. Выбрать PTS-сервис для EPA0.

– Установить PTSSEL.2.

10. Установить стартовый бит передачи.

– Очистить P2.0.

11. Установить EPA0 как канал только сравнения.

– Установить EPA0_CON.6 (рисунок 10.10).

12. Начать работу EPA0 канала, записав время первого прерывания в EPA0_TIME. Для установки правильного значения прибавить baud_value (1A0H) к текущему значению TIMER1 и сохранить результат в EPA0_TIME. Baud_value определяет время первого PTS прерывания. Когда прерывание происходит, PTS передает первый бит данных. Baud_value 1A0H выбирает скорость двоичной передачи 9600 бод.

13. Разрешить PTS и стандартные прерывания.

– Использовать инструкцию EI для разрешения всех стандартных прерываний и EPTS инструкцию для разрешения PTS.

14. Передача стартует. Данные сдвигаются, начиная с наименее значимого (самого правого) бита. Каждый раз, когда происходит совпадение в таймере между EPA0_TIME и TIMER1, EPA0 канал производит прерывание, и PTS выводит следующий бит данных на вывод сконфигурированный как TXD. Когда PTSCOUNT декрементируется до нуля, PTS вызывает прерывание end-of-PTS. Процедура обслуживания прерываний должна загрузить следующий байт данных, перезагрузить регистры PTSCOUNT и PTSCON1, очистить TXD бит, чтобы создать стартовый бит для следующего слова, которое будет передано, выбрать обслуживание PTS для EPA0 и перезагрузить регистры EPA0_TIME и EPA0_CONTROL.

15. Чтобы определить, когда все байты будут переданы, надо создать циклическую подпрограмму, чтобы проверять состояние TXDDONE флага.

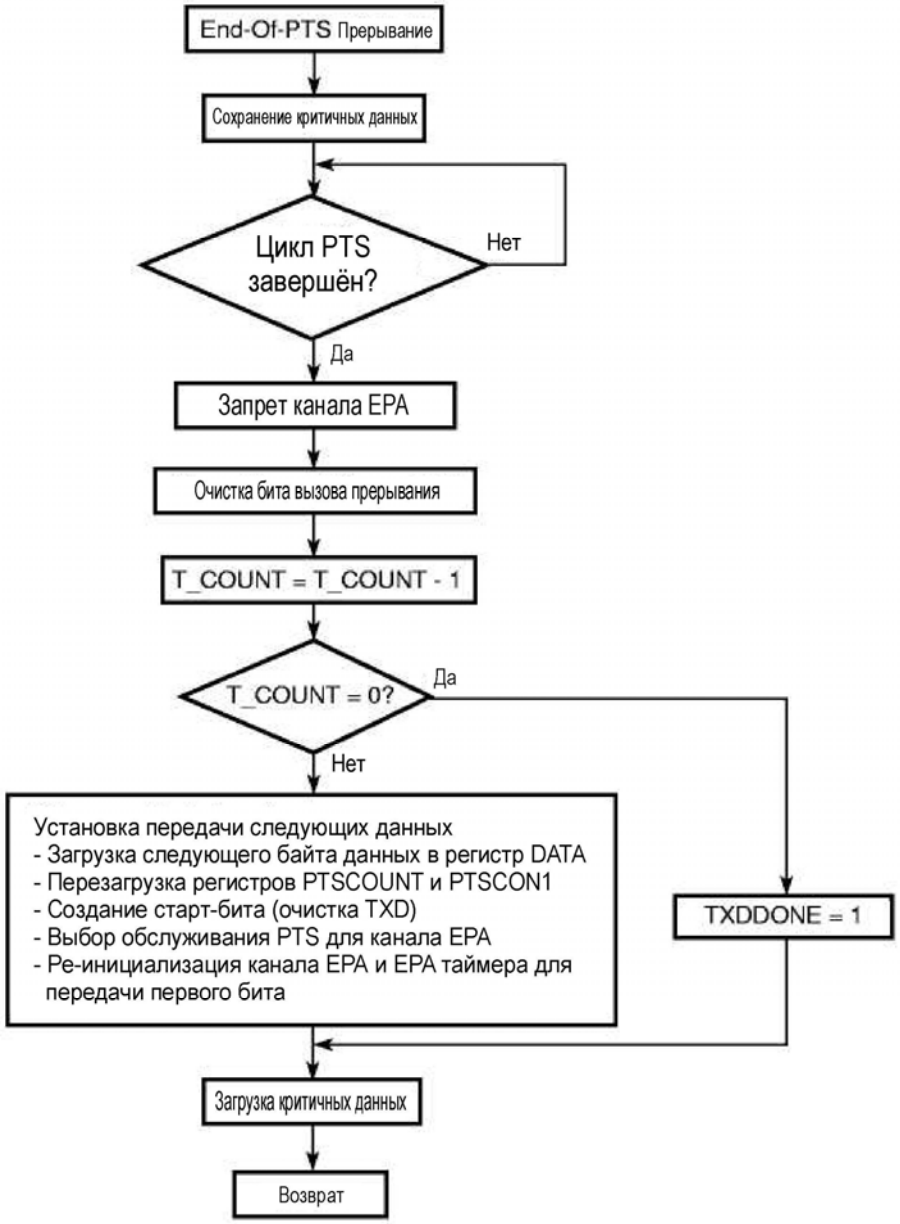


Рисунок 6.26 – Режим асинхронной SIO передачи. Блок-схема процедуры прерывания end-of-PTS

Пример режима асинхронного SIO приёма

В режиме асинхронного приёма (ASIO) канал EPA перехватывает задний фронт старт-бита данных на входе, функционирующем как RXD. При перехвате EPA генерирует стандартное прерывание, стартующее процесс асинхронного приёма. Процедура обслуживания для стандартного прерывания – та же, что и для прерывания end-of-PTS. Она переводит канал EPA в режим сравнения и выставляет время следующего сравнения – время полутора битов и разрешает PTS. Через время полутора битов с начала старт-бита первый цикл PTS принимает входные данные с RXD и сдвигает их в регистр DATA (рисунок 6.27). Если разрешена мажоритарность, производится дополнительная выборка. Если две выборки различны, производится третья, чтобы определить, какая из первых двух верна. Регистр SAMPTIME (см. рисунок 6.19) задаёт временной интервал между выборками. Мажоритарность существенно увеличивает время выполнения цикла PTS (см. таблицу 6.4).

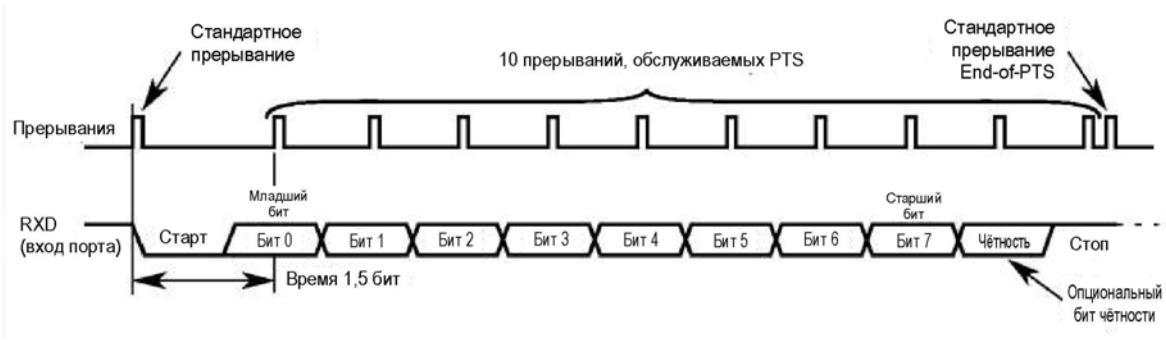


Рисунок 6.27 – Временная диаграмма асинхронного приёма SIO

Первый цикл PTS должен запущен следующим образом:

- Вход RXD и сигнал SCK инициализируются к требуемому системой логическому уровню.
- Прибавить содержимое регистра таймера к Baud_value (рисунок 6.19) и сохранить результат в регистре времени EPA. Это устанавливает время для первого прерывания и появление первого прерывания в соответствии со скоростью приёма данных.

В следующем примере EPA0 используется для захвата старт-бита, а P2.0 (RXD) – для получения данных. Процедура синхронного последовательного ввода PTS настроена на приём 16 байтов с восемью битами данных и битом чётности каждый. В этом примере используется несколько определяемых пользователем регистров. R_COUNT определяет количество принимаемых байтов, а RXDDONE – флаг, который устанавливается после получения всех байтов.

1. Запретить прерывания и PTS.

– Использовать инструкцию DI, чтобы запретить все стандартные прерывания и инструкцию DPTS, чтобы запретить PTS.

2. Установить указатель стека.
3. Сбросить все регистры маски прерывания.

– Очистить INT_MASK, INT_MASK1 и PI_MASK.

4. Инициализировать P2.0 как RXD.

– Очистить P2_DIR.0 (выбор входа).

– Очистить P2_MODE.0 (выбор функции LSIO).

– Установить P2_REG.0 (инициализирует RXD вывод в “1”).

5. Инициализировать и разрешить таймер; выбрать прямой счет, внутреннюю синхронизацию и запретить предделитель.

– Установить биты 6 и 7 T1CONTROL (рисунок 10.8).

6. Инициализировать PTSCB как показано в таблице 6.16.

Таблица 6.16 – PTSCBs в режиме приёма ASIO PTSCBs

PTSCB1	PTSCB2
PTSVEC (H) = указатель на PTSCB2	Неиспользованный
PTSVEC (L) = указатель на PTSCB2	SAMPTIME = 01H
BAUD (H) = 01H	DATA (H) = 00H (очистка регистра для приёма данных)
BAUD (L) = A0H	DATA (L) = 00H (очистка регистра для приёма данных)
EPAREG (H) = 1FH (EPA0_TIME)	PTSCON1 = 60H (разрешить контроль нечетности)
EPAREG (L) = 42H (EPA0_TIME)	PORTMASK = 01H (P2.0 = TXD)
PTSCON = 21H (SSIO режим передачи, мажоритарность)	PORTREG (H) = 1FH (P2_PIN)
PTSCOUNT = 0AH (8 битов данных, 1 бит четности)	PORTREG (L) = D6H (P2_PIN)

7. Разрешить прерывание EPA0.

– Установить INT_MASK.2.

8. Загрузить число байтов к передаче в определённый пользователем регистр счета передачи (R_COUNT) и очистить определенный пользователем флаг завершенного приема (RXDDONE).

– LD R_COUNT, #16.

– CLRB RXDDONE.

9. Выбрать PTS-сервис для EPA0.

– Установить PTSSEL.2.

10. Установить EPA0 для захвата задних фронтов.

– Установить EPA0_CON.4 (рисунок 10.10).

11. Разрешить PTS и стандартные прерывания.

– Использовать инструкцию EI для разрешения всех стандартных прерываний и EPTS инструкцию для разрешения PTS.

12. Переключить вход RXD, чтобы начать прием. EPA генерирует стандартное прерывание. Процедура обслуживания для стандартного прерывания – та же, что и для прерывания end-of-PTS. Она различает прерывание старта приёма от прерывания end-of-PTS чтением регистра EPA0_CON. Если регистр установлен в режим захвата, тогда это прерывание старта приёма.

Процедура обслуживания прерываний состоит из следующих действий:

1. Перевод канала EPA в режим сравнения.

– Установка EPA0_CON.6 (выбор режима сравнения).

2. Инициализировать время первого цикла PTS, записав время первого прерывания в EPA0_TIME. Для установки корректного значения надо умножить baud_value (1A0H) на 1,5, сложить результат с текущим значением TIMER1 и сохранить результат в EPA0_TIME. Baud_value задает время первого прерывания PTS, по которому PTS передаёт первый бит данных. Baud_value = 1A0H задаёт скорость 9600 бод.

3. Выбор PTS-сервиса для EPA0.

– Установить PTSSEL.2.

4. PTS начинает загружать данные через интервалы, равные времени 1,5 бит. Когда PTSCOUNT уменьшается до нуля, PTS вызывает прерывание end-of-PTS (см. рисунок 6.28). Процедура обслуживания прерывания проверяет ошибки фрейминга и чётности (рисунок 6.19), очищает регистры DATA для следующего приёма, перезагружает регистры PTSCOUNT и PTSCON1, переводит канал EPA в режим захвата по заднему фронту. Необходимо помнить, что в это время обслуживание PTS для канала EPA запрещено, т.к. следующее прерывание должно быть стандартным.

5. Чтобы определить, когда все байты были переданы, надо создать циклическую подпрограмму, чтобы проверять состояние флага RXDDONE.

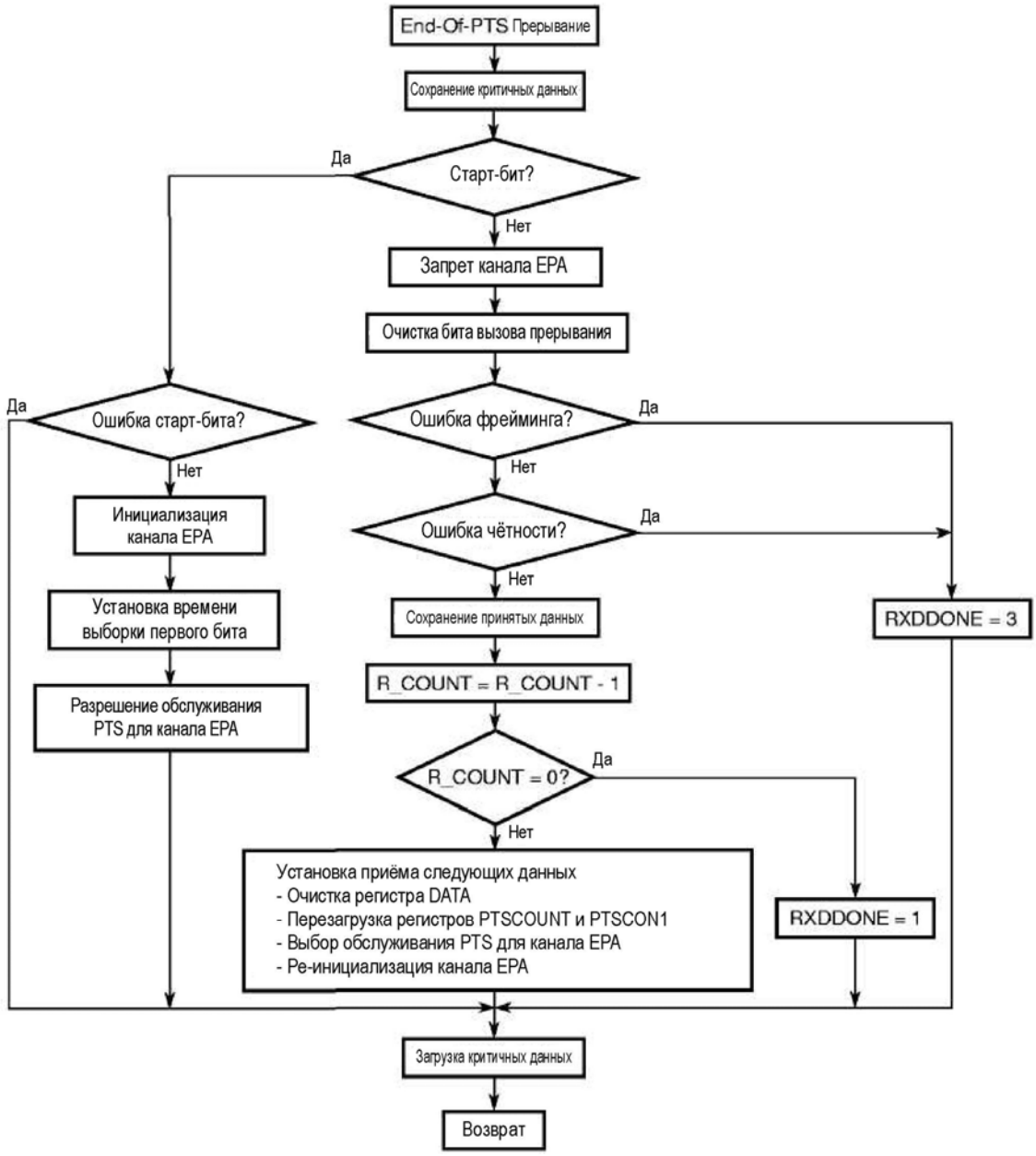


Рисунок 6.28 – Режим асинхронного SIO приёма. Блок-схема процедуры прерывания end-of-PTS

7 Порты ввода-вывода

Порты ввода-вывода обеспечивают механизм передачи информации между устройством и схемами, входящими в периферийное окружение. Они могут считывать состояние системы, осуществлять мониторинг системы, считывать состояние выходных устройств системы, конфигурировать систему, формировать сигналы управления, обеспечивать связь по последовательному порту и так далее.

7.1 Описание портов ввода-вывода

Регистры стандартных портов ввода-вывода расположены в адресном пространстве SFR, к которым обращаются косвенно или через окна. Регистры нестандартных портов ввода-вывода расположены по адресам, соответствующим карте памяти контроллера. К этим регистрам можно обратиться с использованием косвенной или индексной адресации. Через окна в регистровом файле к ним нет доступа. Все порты могут обеспечить медленные инструкции ввода/вывода или обслужить дополнительные функции. Таблица 7.1 показывает краткий обзор портов ввода-вывода устройства. Далее порты описаны более подробно и с объяснением. Разделы, которые описывают соответствующие периферийные устройства, обсуждают использование выводов для их специальных функций.

Таблица 7.1 – Порты ввода-вывода

Порт	Биты	Тип	Направление	Связанные периферийные устройства
Порт 0	8	Стандартный	Только для входа	АЦП
Порт 1	5	Стандартный	Только для входа	АЦП, EPA
Порт 2	8	Стандартный	Двунаправленный	EPA и таймеры
Порт 3	8	Нестандартный	Двунаправленный	Шина адреса/данных
Порт 4	8	Нестандартный	Двунаправленный	Шина адреса/данных
Порт 5	8	Нестандартный	Двунаправленный	Управление шиной
Порт 6	8	Стандартный	Только для выхода	ШИМ, ГФС

7.2 Входные порты 0 и 1

Порт 0 – восьмибитный, высокоимпедансный входной порт, который обеспечивает аналоговые и цифровые входы. Входные выводы могут использоваться как цифровые входы, а также как входы АЦП. Порты 0 и 1 отличаются от других стандартных портов, т. к. их выводы могут использоваться только как входы к цифровой или аналоговой схеме. Поскольку порт 0 и порт 1 постоянно формируется как входной порт, они не имеют регистров конфигурации. У каждого порта имеется свой единственный регистр Px_PIN (x=0, 1). Он может читаться, чтобы определить текущее состояние вывода порта. К регистру можно обратиться с косвенно-регистровой и индексной адресацией, а также с помощью окна в регистровом файле. В таблице 7.2 приведены стандартные выводы входных портов. В таблице 7.3 описаны регистры состояния Px_PIN (x=0, 1).

Таблица 7.2 – Стандартные выводы входных портов 0 и 1

Вывод порта	Сигнал специальной функции	Тип сигнала специальной функции	Связанное периферийное устройство
P0.0-P0.3	ACH0-ACH3	Вход	АЦП
P0.4	ACH4	Вход	АЦП
	PMODE.0	Вход	OTP
P0.5	ACH5	Вход	АЦП
	PMODE.1	Вход	OTP
P0.6	ACH6	Вход	АЦП
	PMODE.2	Вход	OTP
P0.7	ACH7	Вход	АЦП
	PMODE.3	Вход	OTP
P1.0	ACH8	Вход	АЦП
P1.1	ACH9	Вход	АЦП
P1.2	ACH10	Вход	АЦП
	T1CLK	Вход	EPA
P1.3	ACH11	Вход	АЦП
	T1DIR	Вход	EPA
P1.4	ACH12	Вход	АЦП

Таблица 7.3 – Регистры входных портов 0 и 1

Мнемоника	Адрес	Описание
P0_PIN	1FA8H	Каждый бит P0_PIN отражает текущее состояние соответствующего вывода порта 0.
P1_PIN	1FA9H	Каждый бит P1_PIN отражает текущее состояние соответствующего вывода порта 1.

Стандартное решение буферов входных портов 0 и 1

Рисунок 7.1 – схемное решение вывода входного порта. Транзисторы Q1 и Q2 служат для защиты от статического электричества, они подключены к ∩VCC2 и ∩0V. Транзистор Q3 – дополнительное устройство защиты от статического электричества, подключенный к #0V. Резистор R1 ограничивает текущий ток через Q3 до приемлемой величины. Из этого узла входной сигнал идет к аналоговому мультиплексору и на цифровой буфер трансляции уровня. Буфер трансляции уровня конвертирует входные сигналы в рабочие сигналы, совместимые с цифровыми уровнями напряжения #VCC1 и #0V, используемыми ядром центрального процессора. Этот буфер является триггером Шмитта. Он необходим для улучшенной помехоустойчивости. Сигналы защелкиваются в P0_PIN или P1_PIN регистрах и выводятся на внутреннюю шину, когда регистры P0_PIN или P1_PIN читаются.

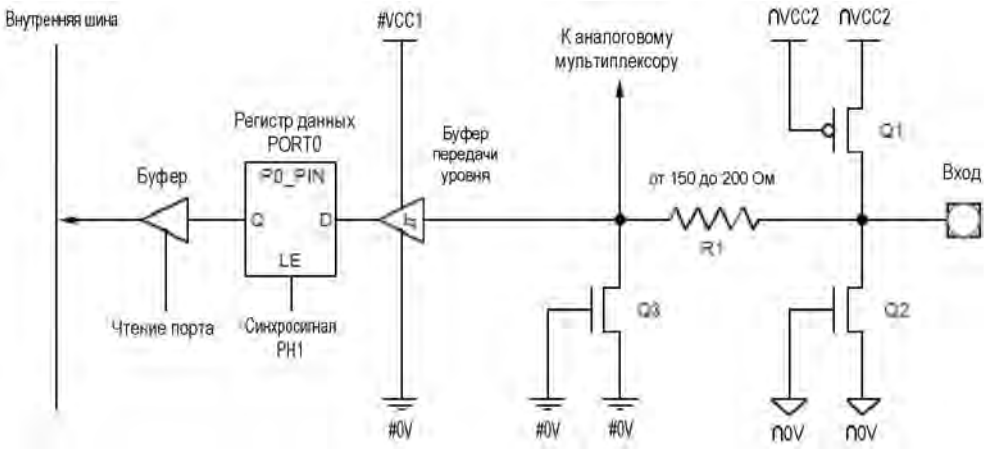


Рисунок 7.1 – Структура входного порта

Анализ стандартных входных портов 0 и 1

Выводы портов 0 и 1 являются уникальными в том, что они могут индивидуально использоваться как цифровые входы и аналоговые входы в одно время. Однако чтение порта усиливает влияние помех на АЦП, уменьшая точность любого аналого-цифрового преобразования. Поэтому строго рекомендуется не читать порты 0, 1 при выполнении аналого-цифрового преобразования. Для уменьшения помех регистры P0_PIN и P1_PIN тактируются только во время чтения порта.

Эти выводы порта запитаны аналоговым напряжением (∩VCC2) и аналоговой землей (∩0V). Если выводы порта должны функционировать или как аналоговые или как цифровые входы, то выводы ∩VCC2 и ∩0V должны обеспечить требуемую мощность. Если напряжение, приложенное к аналоговому входу, превышает ∩VCC2 или ∩0V более, чем на 0,5 В, будут открываться защитные транзисторы Q1 или Q2, уменьшая точность всех аналоговых преобразований.

Значение на выводе порта фиксируется за один такт перед разрешением чтения буфера. Фиксация уровня на входе происходит в течение фазы 1 (в то время как CLKOUT имеет низкий уровень) и передается на внутреннюю шину. Чтобы гарантировать правильное чтение, сигнал должен быть установлен не менее чем за 45 нс до переднего фронта CLKOUT и удерживаться до прохождения заднего фронта CLKOUT.

Как цифровой вход, вывод функционирует как вход с высоким импедансом. В то же время, как аналоговый вход, вывод должен обеспечить ток в течение короткого времени для зарядки внутреннего конденсатора, когда начинается аналого-цифровое преобразование. Это означает, что если аналого-цифровое преобразование имеет место на выводе порта, его входные характеристики изменяются моментально.

7.3 Двунаправленные порты 2, 5

Хотя двунаправленные порты очень похожи и по схеме, и по конфигурации, порт 5 отличается от других. Порт 5, имеющий адрес из адресного пространства, имеет КМОП-буфер, необходимый для выполнения быстрого управления внешней периферией. Порт 2 использует входные буферы на базе триггера Шмитта для улучшения помехоустойчивости.

В таблице 7.4 приведен список выводов двунаправленных портов 2, 5 с их сигналами специальной функции и связанными периферийными устройствами.

Таблица 7.4 – Двунаправленные выводы порта

Вывод порта	Сигнал специальной функции	Тип сигнала специальной функции	Связанное периферийное устройство
P2.0	CAPCOMP0	Вход - выход	EPA
	PVER	Выход	OTP
P2.1	CAPCOMP1	Вход - выход	EPA
	PALE#	Выход	OTP
P2.2	CAPCOMP2	Вход - выход	EPA
	PROG#	Выход	EPA
P2.3	CAPCOMP3	Вход - выход	EPA
P2.4	COMP0	Выход	EPA
P2.5	COMP1	Выход	EPA
P2.6	COMP2	Выход	EPA
P2.7	COMP3	Выход	EPA
P5.0	ALE/ADV#	Выход	Контроллер памяти
P5.1	INST	Выход	Контроллер памяти
P5.2	WR#/WRL#	Выход	Контроллер памяти
P5.3	RD#	Выход	Контроллер памяти
P5.4	ONCE#	Вход	Контроллер памяти
P5.5	BHE#/WRH#	Выход	Контроллер памяти
P5.6	READY	Вход	Контроллер памяти
P5.7	BW	Вход	Контроллер памяти

В таблице 7.5 приведено описание регистров управления портов 2, 5. Каждый порт имеет три регистра управления (Px_MODE, Px_DIR и Px_REG); они могут и читаться, и записываться. Регистры Px_PIN – регистры состояния, повторяющие логический уровень на выводе портов. Они могут только читаться. Регистры порта 2 используют побайтовую адресацию и могут быть доступны через окна. Регистры порта 5 используют 16-битную адресацию и не могут быть доступны через окна.

Таблица 7.5 – Управление портами 2, 5 и регистры состояния

Мнемоника	Адрес	Описание
1	2	3
P2_DIR P5_DIR	1FD2H 1FF3H	Направление порта x (x = 0,1). Каждый бит Px_DIR управляет направлением соответствующего вывода. 0 = комплементарный выход (только выход). 1 = вход или выход с открытым стоком (вход, выход или вход-выход). Выходы с открытым стоком требуют внешний pull-up.
P2_MODE P5_MODE	1FD0H 1FF1H	Режим порта x. Каждый бит Px_MODE определяет, функционирует ли соответствующий вывод как стандартный вывод порта ввода-вывода или как сигнал специальной функции. 0 = стандартный вывод порта ввода – вывода. 1 = сигнал специальной функции.
P2_PIN P5_PIN	1FD6H 1FF7H	Вход порта x. Каждый бит Px_PIN отражает текущее состояние соответствующего вывода независимо от конфигурации вывода.

Продолжение таблицы 7.5

1	2	3
P2_REG P5_REG	1FD4H 1FF5H	Вывод данных порта x. Для входа необходимо установить соответствующий Px_REG бит. Для выхода написать данные, которые будут выведены каждым выводом в соответствующий бит Px_REG. Когда вывод конфигурируется как стандартный вход-выход (Px_MODE.y = 0), результат записи центрального процессора в Px_REG немедленно видим на выводе. Когда вывод формируется как сигнал специальной функции (Px_MODE.y = 1), соответствующая периферия на МК или компонент вне МК управляет выводом. Центральный процессор может все еще писать Px_REG, но состояние вывода не меняется, пока он не переключен к его стандартной функции входа-выхода. Эта особенность позволяет программному обеспечению формировать вывод как стандартный вход-выход (очистить Px_MODE.y), инициализировать или переписывать значение вывода, затем формировать вывод как сигнал специальной функции (необходимо установить Px_MODE.y). Этим способом инициализация, восстановление ошибки, обработка исключения и т.д. могут быть сделаны без изменения работы связанного периферийного устройства.

Работа двунаправленных портов 2, 5

Рисунок 7.2 показывает логику управления выходных транзисторов Q1 и Q2. Q1 может выдать ток до минус 3 мА при выходном напряжении (#VCC1 – 0,7) В. Q2 может пропустить ток не менее 3 мА при выходном напряжении 0,45 В. В режиме ввода-вывода (выбирается обнулением Px_MODE.y) состояние Px_REG и Px_DIR подается на мультиплексоры. Это позволяет управлять транзисторами Q1 и Q2 так, чтобы вывод находился в состоянии высокого или низкого уровня или в высокоимпедансном состоянии. Таблица 7.6 – логическая таблица для функционирования ввода-вывода этих портов.

В режиме специальной функции (выбирается установкой Px_MODE.y) SFDIR и SFDATA – управляющие сигналы для мультиплексоров. Этими сигналами производится управление транзисторами Q1 и Q2 так, чтобы можно было установить на выводе или высокий, или низкий уровень сигнала, или высокоимпедансное состояние. Для выходного сигнала специальной функции обнуляют SFDIR. Для входного сигнала специальной функции устанавливают SFDIR. Таблица 7.7 – логическая таблица для выполнения специальной функции этих портов. Даже если вывод должен использоваться в режиме специальной функции, необходимо инициализировать вывод как вход или выход, записывая Px_DIR.

Резистор R1 обеспечивает защиту от статического электричества. Входные сигналы буферизуются. Стандартные порты используют в качестве буфера триггер Шмитта для улучшенной помехоустойчивости. Порт 5 использует стандартный входной буфер для быстрых переключений, требуемых для управления шиной. Сигналы защелкиваются в регистре Px_PIN и выводятся на внутреннюю шину, когда регистр Px_PIN читается.

Спадающий фронт сигнала RESET# открывает транзистор Q3, который остается открытым приблизительно на 300 нс, заставляя вывод быстро изменить его состояние на состояние в режиме сброса. Активный низкий уровень RESET# включает транзистор

Q4, который осуществляет слабую поддержку высокого уровня. Q4 осуществляет эту слабую поддержку, пока не будет изменен Px_MODE.

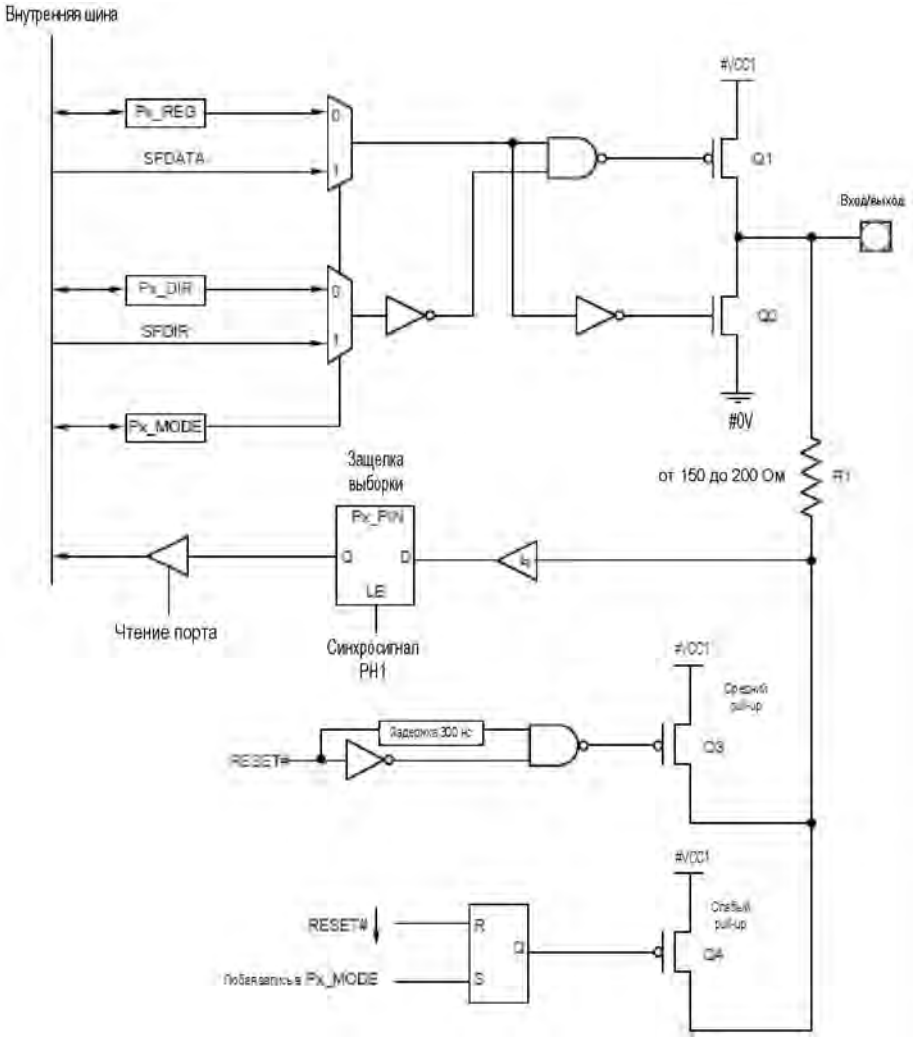


Рисунок 7.2 – Структура двунаправленного порта

Таблица 7.6 – Таблица логических уровней для двунаправленных портов в режиме входа-выхода

Конфигурация	Дополнительный выход		Выход с открытым стоком	Вход
Px_MODE	0	0	0	0
Px_DIR	0	0	1	1
SFDIR	X [1)]	X	X	X
SFDATA	X	X	X	X
Px_REG	0	1	0, 1 [2)]	1
Q1	закрыт	открыт	закрыт	закрыт
Q2	открыт	закрыт	открыт, закрыт [2)]	закрыт
Px_PIN	0	1	X [3)]	высокий импеданс [4)]

[1)] X = любой уровень.

[2)] Если Px_REG очищен, то Q2 открыт (пропускает ток); если Px_REG установлен, то Q2 закрыт.

[3)] Px_PIN содержит текущее значение на выводе.

[4)] В течение сброса и до первой записи в Px_MODE Q4 открыт (пропускает ток).

Таблица 7.7 – Таблица логических уровней для двунаправленных портов в режиме специальной функции

Конфигурация	Дополнительный выход		Выход с открытым стоком	Вход
Px_MODE	1	1	1	1
Px_DIR	0	0	1	1
SFDIR	0	0	1	1
SFDATA	0	1	0, 1 [2)]	1
Px_REG	X [1)]	X	X	1
Q1	закрыт	открыт	закрыт	закрыт
Q2	открыт	закрыт	открыт, закрыт [2)]	закрыт
Px_PIN	0	1	X [3)]	высокий импеданс [4)]

[1)] X = любой уровень.
[2)] Если Px_REG очищен, Q2 открыт (пропускает ток); если Px_REG установлен, Q2 открыт.
[3)] Px_PIN содержит текущее значение на выводе.
[4)] В течение сброса и до первой записи в Px_MODE, Q4 открыт (пропускает ток).

Конфигурация двунаправленных портов 2, 5

Каждый двунаправленный вывод порта может быть индивидуально сконфигурирован, чтобы работать или как вывод входа-выхода, или как вывод сигнала специальной функции. В режиме специальной функции сигнал управляется периферийным устройством МК или компонентом вне МК. В любой конфигурации возможны два режима:

- комплементарный выход (только выход);
- высокоимпедансный вход или выход с открытым стоком (вход, выход или двунаправленный).

Для предотвращения плавающего уровня на выводе порта выводы двунаправленных портов имеют слабую поддержку высокого уровня, пока программное обеспечение не запишет Px_MODE. Заданные по умолчанию значения регистров управления после начальной установки МК (RESET#) конфигурируют выводы портов как высокоимпедансные входы со слабой поддержкой высокого уровня (pull-up). Чтобы гарантировать, что порты инициализируются правильно и что слабая поддержка высокого уровня отключена, выполняют следующую последовательность инициализации:

1. Производится запись в Px_DIR, чтобы установить конкретные выводы или как входы, или как выходы (выводы будут управлять данными, которые определяются в шаге 3).
   - Для комплементарного выхода очистить его Px_DIR бит.
   - Для высокоимпедансного входа или выхода с открытым стоком установить его Px_DIR бит (выходы с открытым стоком требуют внешней слабой поддержки высокого уровня).
2. Производится запись в Px_MODE, чтобы выбрать или режим специальной функции, или вход-выход. Запись Px_MODE (независимо от значения) выключает поддержку высокого уровня. Даже если порт должен использоваться как вход-выход (это выполняется после начальной установки), необходимо записать Px_MODE, чтобы гарантировать, что слабая поддержка высокого уровня была выключена.
   - Для стандартного входа-выхода очистить его бит в Px_MODE. В этом режиме вывод устанавливают как определено в шагах 1 и 3.
   - Для сигнала специальной функции установить его бит в Px_MODE. В этом режиме связанное периферийное устройство управляет выводом.

3. Производится запись в Px_REG.

- Для выводов, работающих как выходы (задается в шаге 1), записать данные в соответствующие биты Px_REG, чтобы получить эти данные на соответствующих выводах. Для выходов в режиме специальной функции значение является несущественным, потому что периферийное устройство управляет выводом. Тем не менее, необходимо сделать запись в Px_REG, чтобы инициализировать вывод.
- Для выводов, установленных как входы (определяется в шаге 1), установить соответствующие биты Px_REG.

В таблице 7.8 приведены значения управляющих регистров для каждой возможной конфигурации. Для выходов в режиме специальной функции значение Px_REG является несущественным, потому что связанное периферийное устройство управляет выводом в режиме специальной функции. Однако необходимо записывать в Px_REG, чтобы инициализировать вывод. Для двунаправленного вывода, чтобы он функционировал как вход (или вывод специальной функции, или вывод порта), необходимо установить Px_REG.

Таблица 7.8 – Значения управляющих регистров для каждой конфигурации

Необходимая конфигурация вывода	Назначения регистра конфигурации		
Стандартный сигнал входа - выхода	Px_DIR	Px_MODE [1)]	Px_REG
Комплементарный выход, вывод 0	0	0	0
Комплементарный выход, вывод 1	0	0	1
Выход с открытым стоком, вывод сильного 0	1	0	0
Выход с открытым стоком, высокий импеданс	1	0	1
Вход	1	0	1
Сигнал специальной функции	Px_DIR	Px_MODE [1)]	Px_REG
Комплементарный выход, выходное значение, управляемое периферийным устройством	0	1	X
Выход с открытым стоком, выходное значение, управляемое периферийным устройством	1	1	X
Вход	1	1	1

[1)] В течение сброса и до первой записи Px_MODE выводы имеют слабую поддержку высокого уровня.

Пример конфигурации выводов двунаправленного порта

Предположим, что необходимо сконфигурировать выводы двунаправленного порта как показано в таблице 7.9.

Таблица 7.9 – Пример конфигурации порта

Выводы порта	Конфигурация	Данные
Px.0, Px.1	высокоимпедансный вход	высокий импеданс
Px.2, Px.3	выход с открытым стоком	0
Px.4	выход с открытым стоком	1 (при условии внешней слабой поддержки высокого уровня)
Px.5, Px.6	комплементарный выход	0
Px.7	комплементарный выход	1

Можно использовать следующий набор команд. Таблица 7.10 показывает состояние каждого вывода после того, как прошла начальная установка МК (сброс) и после выполнения ниже приведенных команд.

LDB Px_DIR, #00011111B
LDB Px_MODE, #00000000B
LDB Px_REG, #10010011B

Таблица 7.10 – Состояние выводов после сброса и после выполнения команд

Действие или код	Итоговые состояния выводов (см. примечание)							
	Px.7	Px.6	Px.5	Px.4	Px.3	Px.2	Px.1	Px.0
RESET	wk1	wk1	wk1	wk1	wk1	wk1	wk1	wk1
LDB Px_DIR, #00011111B	1	1	1	wk1	wk1	wk1	wk1	wk1
LDB Px_MODE, #00000000B	1	1	1	HZ1	HZ1	HZ1	HZ1	HZ1
LDB Px_REG, #10010011B	1	0	0	HZ1	0	0	HZ1	HZ1
Примечание – wk1 = слабая поддержка высокого уровня, HZ1 = высокий импеданс (фактически “1” с внешней поддержкой высокого уровня).								

Анализ двунаправленных портов

Ниже приведен специальный анализ для использования портов.

Порт 2 – После сброса программное обеспечение должно сконфигурировать устройство в соответствии с внешней системой. Это достигается записью соответствующих данных конфигурации в P2_MODE. Запись в P2_MODE не только конфигурирует выводы, но также и выключает транзистор, который осуществляет слабую поддержку высокого уровня (Q4 на рисунке 7.2). По этой причине, даже если порт 2 должен использоваться как это формируется при сбросе, необходимо все равно записать данные в P2_MODE.

Порт 5 – После сброса программное обеспечение должно сконфигурировать устройство в соответствии с внешней системой. В дальнейшем будут описаны состояния выводов порта 5 после сброса до записи в P5_MODE. Запись P5_MODE не только конфигурирует выводы, но также и выключает транзистор, который осуществляет слабую поддержку высокого уровня (Q4 на рисунке 7.2). По этой причине, даже если порт 5 должен использоваться как это формируется при сбросе, необходимо все равно записать данные в P5_MODE.

P5.0/ALE – Если удерживать высокий уровень на EA# при сбросе (доступ внутренней памяти программ), вывод имеет слабую поддержку высокого уровня, пока не будет записан P5_MODE. Если удерживать низкий уровень на EA# при сбросе (доступ внешней памяти программ), то ALE или ADV# активируется как системный управляющий вывод в зависимости от бита ALE в CCR0. В любом случае, вывод становится настоящим комплементарным выходом.

P5.1/INST – Этот вывод имеет слабую поддержку высокого уровня, пока программное обеспечение не запишет конфигурацию в P5_MODE.

P5.2/WR#/WRL# – Этот вывод имеет слабую поддержку высокого уровня, пока программное обеспечение не запишет конфигурацию в P5_MODE.

P5.3/RD# – Если удерживать высокий уровень на EA# при сбросе (доступ к внутренней памяти программ), вывод имеет слабую поддержку высокого уровня, пока не будет записан P5_MODE. Если удерживать низкий уровень на EA# при сбросе (доступ внешней памяти программ), то RD# активируется как системный управляющий вывод и становится настоящим комплементарным выходом.

P5.4 – Этот вывод имеет слабую поддержку высокого уровня, пока программное обеспечение не запишет конфигурацию в P5_MODE. P5.4 – это вывод, разрешающий режим ONCE. Поскольку низкий уровень сигнала в течение сброса может заставить

устройство войти в режим ONCE, необходимо поддерживать высокий уровень на выводе в течение сброса, чтобы предотвратить случайный вход в режим ONCE.

P5.5/BHE#/WRH# – Этот вывод имеет слабую поддержку высокого уровня, пока запись CCB не закончена. Затем, состояние этого вывода зависит от значения BW0 бита CCR0. Если BW0 обнулен, вывод имеет слабую поддержку высокого уровня, пока не будет записан P5_MODE. Если BW0 установлен, BHE# активизирован как вывод управления системы, и вывод становится комплементарным выходом.

P5.6/READY – Этот вывод имеет слабую поддержку высокого уровня, пока запись CCB не закончена. В это время состояние этого вывода зависит от значения битов CCR0,1 в IRC0–IRC2. Если все IRC0–IRC2 установлены (111B), активизирован READY как системный управляющий вывод. Это предотвращает вставку бесконечных циклов ожидания до первого обращения к внешней памяти. При других значениях IRC0–IRC2 вывод конфигурируется после сброса как вход-выход.

Примечание – Если IRC0–IRC2 CCB все установлены (READY активируется как системный управляющий вывод) и P5_MODE.6 очищен (конфигурирование вывода как вход-выход), при обращении к внешней памяти МК может заблокироваться.

P5.7/BW – Этот вывод имеет слабую поддержку высокого уровня, пока программное обеспечение не запишет конфигурацию в P5_MODE.

7.4 Двунаправленные порты 3, 4 (шина адресов/данных)

Порты 3 и 4 – восьмибитные, двунаправленные, имеющие адреса из общего адресного пространства порты ввода-вывода. К ним можно обратиться только с косвенной или индексной адресацией и нельзя обратиться через регистровые окна. Порты 3 и 4 обеспечивают мультиплексную шину адреса/данных. В режиме программирования порты 3 и 4 служат шиной программирования (PBUS). Порт 5 обеспечивает сигналы управления шины.

В течение внешних циклов шины запоминающего устройства МК забирает под свой контроль порты 3 и 4 и автоматически формирует их как комплементарные порты вывода для того, чтобы сконфигурировать их для вывода адреса/данных или как входы для того, чтобы сосчитать данные. По этой причине эти порты не имеют регистров режима.

МК при неактивном EA# (высокий) имеет нерабочее время между внешними шинными циклами. Когда шина адреса/данных простаивает, можно использовать эти порты для ввода-вывода. Подобно порту 5, эти порты используют стандартные входные CMOS буферы. Однако порты 3 и 4 должны полностью функционировать или как комплементарные или порты с открытым стоком. Их выводы не могут конфигурироваться индивидуально. МК при активном EA# (низкий) не могут использовать порты 3 и 4 как стандартные входы-выходы. Если EA# активный (низкий), эти порты будут функционировать только как шина адреса/данных.

В таблице 7.11 приведен список выводов портов 3 и 4 с их сигналами специальной функции и связанными периферийными устройствами. В таблице 7.12 приведен список регистров, которые определяют функцию и указывают состояние портов 3 и 4.

Таблица 7.11 – Выводы портов 3 и 4

Выводы порта	Сигнал(ы) специальной функции	Тип сигнала специальной функции	Связанное периферийное устройство
1	2	3	4
P3.7 – P3.0	AD7:0	Вход - выход	шина адреса/данных, младший байт
	PBUS7:0	Вход - выход	шина программирования, младший байт

1	2	3	4
P4.7 – P4.0	AD15:8	Вход - выход	шина адреса/данных, старший байт
	PBUS15:8	Вход - выход	шина программирования старший байт

Таблица 7.12 – Регистры управления и состояния портов 3, 4

Мнемоника	Адрес	Описание
P3_PIN P4_PIN	1FFEH 1FFFH	Вход порта x (x=3, 4). Каждый бит Px_PIN отражает текущее состояние соответствующего вывода независимо от конфигурации вывода.
P3_REG P4_REG	1FFCH 1FFDH	Выход порта x (x=3, 4). Каждый бит Px_REG содержит данные, которые будут выданы соответствующим выводом. Когда устройство требует доступа к внешнему запоминающему устройству, МК берет управление портом и управляет выдачей адреса/данных бит на вывод. Биты адреса/данных выводятся в это время. Когда доступ к внешней памяти закончен, устройство восстанавливает ваши данные на выводе.

Работа двунаправленных портов 3, 4 (шины адреса/данных)

Рисунок 7.3 показывает логику портов 3, 4. В течение сброса активный низкий уровень RESET# выключает Q1 и Q2 и включает транзистор Q3, который осуществляет слабую поддержку высокого уровня. (Q1 может пропускать ток до 3 мА при (#VCC1 – 0,7) В на выходе; Q2 может пропускать ток до 3 мА при 0,45 В на выходе; и Q3 может пропускать ток около 10 мкА при (#VCC1 – 1,0) В на выходе.) В течение нормальной работы (как порта) внутренний сигнал управления BUS CONTROL SELECT управляет портом.

Когда микроконтроллеру необходим доступ к внешней памяти, он обнуляет сигнал BUS CONTROL SELECT, который подает адрес/данные на вход мультиплексора. Адрес/данные выдаются, управляя транзисторами Q1 и Q2.

Когда доступ к внешней памяти не требуется, микроконтроллер устанавливает сигнал BUS CONTROL SELECT, который выбирает Px_REG как источник данных для мультиплексора. Px_REG управляет транзисторами Q1 и Q2 как выходом с открытым стоком. (Выходы с открытым стоком требуют внешних резисторов поддержки высокого уровня.) В этой конфигурации вывод порта может использоваться как вход. Сигнал на выводе записывается в Px_PIN. В таблице 7.13 приведены данные для конфигурации порта как вход – выход с открытым стоком.

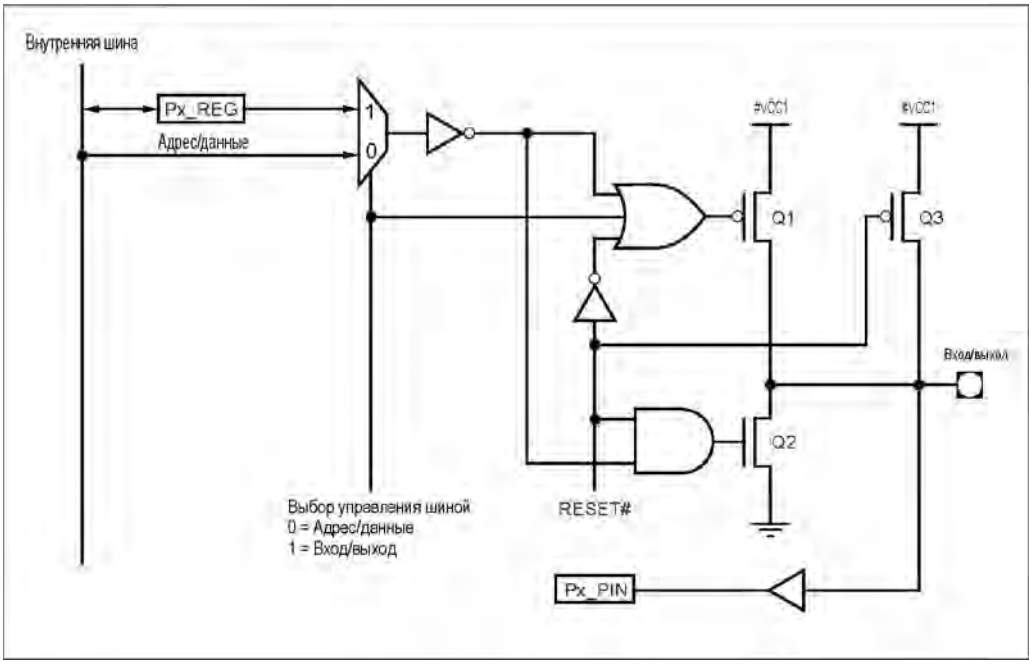


Рисунок 7.3 – Структура портов 3 и 4, шина адресов/данных

Таблица 7.13 – Логическая таблица для портов 3, 4 в режиме входа/выхода с открытым стоком

Конфигурация	Открытый сток	
Px_REG	0	1
Q1	закрыт	закрыт
Q2	открыт	закрыт
Px_PIN	0	высокий импеданс

Использование портов 3 и 4 для входа-выхода

Чтобы использовать вывод порта как выход, необходимо записать выходные данные в соответствующий Px_REG бит. Когда устройство требует доступа к внешней памяти, оно берет под свое управление порт и выводит бит адреса/данных на вывод. Бит адреса/данных выдает выходные данные в течение этого времени. Когда внешний доступ завершен, МК восстанавливает ваши данные на выводе.

Чтобы использовать вывод порта как вход, необходимо установить соответствующий бит Px_REG для перевода вывода в высокоимпедансное состояние. После этого можно читать входное значение вывода в регистр Px_PIN. Когда МК требует доступа к внешней памяти, он берет под свое управление порт. Необходимо сконфигурировать источник входных сигналов порта так, чтобы избежать конфликтных ситуаций в шине.

Анализ работы портов 3 и 4

Когда сигнал EA# активный (низкий уровень), порты 3 и 4 будут функционировать только как шина адреса/данных. В этом случае инструкция, которая работает с P3_REG или P4_REG, организует шинный цикл, в течение которого происходит чтение из или запись во внешнюю память по соответствующему адресу SFR. Например, запись в P4_REG организует шинный цикл, в результате которого во внешнюю память попадает содержимое ячейки 1FFDH. Поскольку P3_REG и P4_REG не формируют выходное состояние на выводе, когда EA# активный, шина будет иметь плавающий уровень в течение многих периодов простоя (например, в течение выполнения инструкций BMOV или TIJMP).

Когда сигнал EA# не активный (высокий уровень), выводы портов 3 и 4 отображают содержание регистров P3_REG и P4_REG, которые при сбросе устанавливаются в FFH. Выводы переходят в состояние высокого импеданса. Уровень напряжения на выводах портов 3 и 4 будет плавать, если через внешние резисторы не организовать слабую поддержку высокого уровня или не записать нули в регистры P4_REG и P3_REG.

7.5 Стандартный выходной порт 6

Порт 6 – выходной порт, который обеспечивает выходы для генератора формы сигнала и сигналов широтно-импульсной модуляции (ШИМ). Выводы порта 6 могут конфигурироваться, чтобы работать или как выводы порта или как выводы генератора формы сигнала или выводы широтно-импульсного модулятора (ШИМ). В таблице 7.14 приведен список выводов с соответствующими сигналами специальной функции и связанными периферийными устройствами.

Таблица 7.14 – Стандартные выводы выходного порта

Вывод порта	Сигнал специальной функции	Тип сигнала специальной функции	Связанное периферийное устройство
P6.0	WG1#	Выход	Генератор формы сигнала
P6.1	WG1	Выход	Генератор формы сигнала
P6.2	WG2#	Выход	Генератор формы сигнала
P6.3	WG2	Выход	Генератор формы сигнала
P6.4	WG3#	Выход	Генератор формы сигнала
P6.5	WG3	Выход	Генератор формы сигнала
P6.6	PWM0	Выход	Широтно-импульсный модулятор
P6.7	PWM1	Выход	Широтно-импульсный модулятор

Таблица 7.15 – Регистр управления выходного порта

Мнемоника	Адрес	Описание
WG_OUTPUT	1FC0H	Управляющий регистр выходного порта 6. Этот регистр управляет выводами порта 6 в режиме входа-выхода. Его функции различны в режиме генератора формы сигнала и в режиме широтно-импульсного модулятора.

Работа выходного порта

Рисунок 7.4 показывает упрощенную схему для порта 6. Порт 6 имеет собственную конфигурацию и регистр управления WG_OUTPUT. Транзистор Q1 может пропускать ток не менее 200 мкА при выходном уровне (#VCC1 – 0,3) В. Для выводов P6.0 – P6.5 транзистор Q2 может пропустить ток не менее 10 мА при выходном уровне 0,45 В. Для выводов P6.6 и P6.7 Q2 может пропустить ток не менее 200 мкА при выходном уровне 0,3 В.

Конфигурирование выводов выходного порта

Порт 6 имеет собственный регистр конфигурации WG_OUTPUT (таблицы 7.15, 7.16). Этот регистр управляет функциями вывода, значениями и полярностью вывода. К этому регистру можно обратиться или как к слову, или как к отдельным байтам, а также можно обращаться через регистровые окна. Функции этого регистра по формированию выходов общего назначения отличны от тех, что формируются для работы в качестве выводов генератора формы сигналов и широтно-импульсного модулятора.

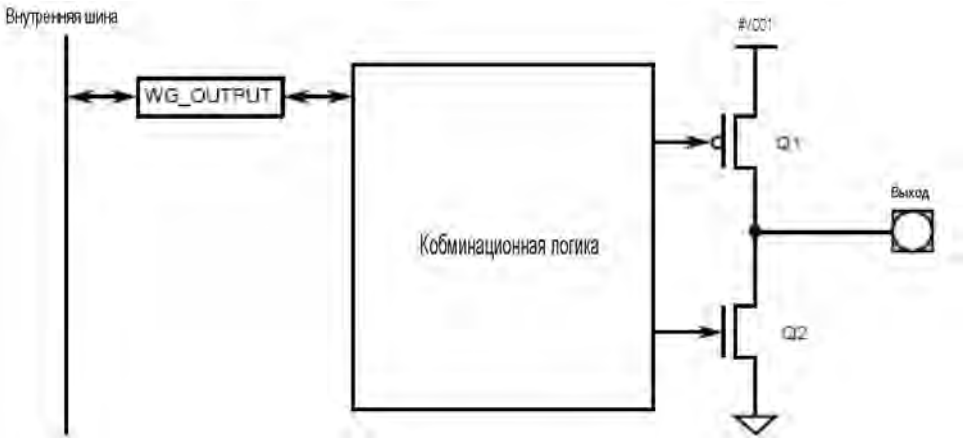


Рисунок 7.4 – Структура выходного порта

WG_OUTPUT (Порт 6) Адрес: 1FC0H
Состояние сброса: 0000H

Регистр WG_OUTPUT управляет функциями порта 6. Если используется порт 6 для выходов общего назначения, то необходимо записать C0H (для активно-высоких выводов) или 00H (для активно-низких выводов) в старший байт WG_OUTPUT и записать желаемые значения выводов в младший байт. Более подробно регистр WG_OUTPUT описан в разделе 8 “Генератор формы сигнала” или в разделе 9 “Широтно-импульсный модулятор”.

Таблица 7.16 – Регистр управления выходного порта WG_OUTPUT

15							8
OP1	OP0	–	M7	M6	M5:4	M3:2	M1:0
7							0
D7	D6	D5	D4	D3	D2	D1	D0

Таблица 7.17 – Поля регистра управления выходного порта WG_OUTPUT

Разряд	Мнемоника	Функция
15:14	OP1:0	Полярность вывода. Эти биты выбирают полярность вывода. 0 – активный низкий уровень, 1 – активный высокий уровень.
13	—	Зарезервирован. Для совместимости с будущими устройствами необходимо записать ноль в этот бит.
12:8	M7:0	Режим. Эти биты выбирают или периферийную функцию, или функцию выхода общего назначения. Необходимо очистить эти биты для выхода общего назначения.
7:0	D7:0	Данные. В режиме выхода общего назначения эти биты выставляются на выводах. Желаемые значения записывают в эти биты (биты 7:0 соответствуют выводам P6.7 - P6.0).

8 Генератор формы сигнала

Генератор формы сигнала (ГФС) упрощает задачу формирования синхронизированного широтно-импульсно-модулированного (ШИМ) выходного сигнала. Данный ГФС оптимизирован для приложений управления двигателями такими, как 3-фазные асинхронные двигатели переменного тока, 3-фазные бесконтактные двигатели постоянного тока или 4-фазные шаговые двигатели. ГФС может выдавать три независимых пары комплементарных ШИМ сигналов, которые совместно используют общий период несущей, «мертвое время» и рабочий режим. После инициализации генератор формы сигнала работает без вмешательства центрального процессора, пока программа не изменит скважность.

Этот раздел описывает работу генератора формы сигнала и объясняет как его конфигурировать.

8.1 Краткий обзор функций генератора формы сигнала

Генератор формы сигнала (рисунок 8.1) имеет три основных части: задающий генератор, формирователи фазированных сигналов и схему управления. Задающий генератор устанавливает период опорного сигнала (несущей), формирователи фазированных сигналов определяют скважность, а схема управления задаёт режим и управляет выработкой прерываний. Максимальная частота ГФС – 15,625 кГц для режимов выравнивания по центру и 31,250 кГц для режимов выравнивания по фронту.

Формируются три независимых фазы, каждая из которых имеет два программируемых комплементарных вывода. Программируемый генератор «мертвого» времени запрещает активацию комплементарных выводов в определенное время. Период несущей, «мертвое» время и рабочий режим, одни и те же для всех трех фаз; скважность программируется независимо.

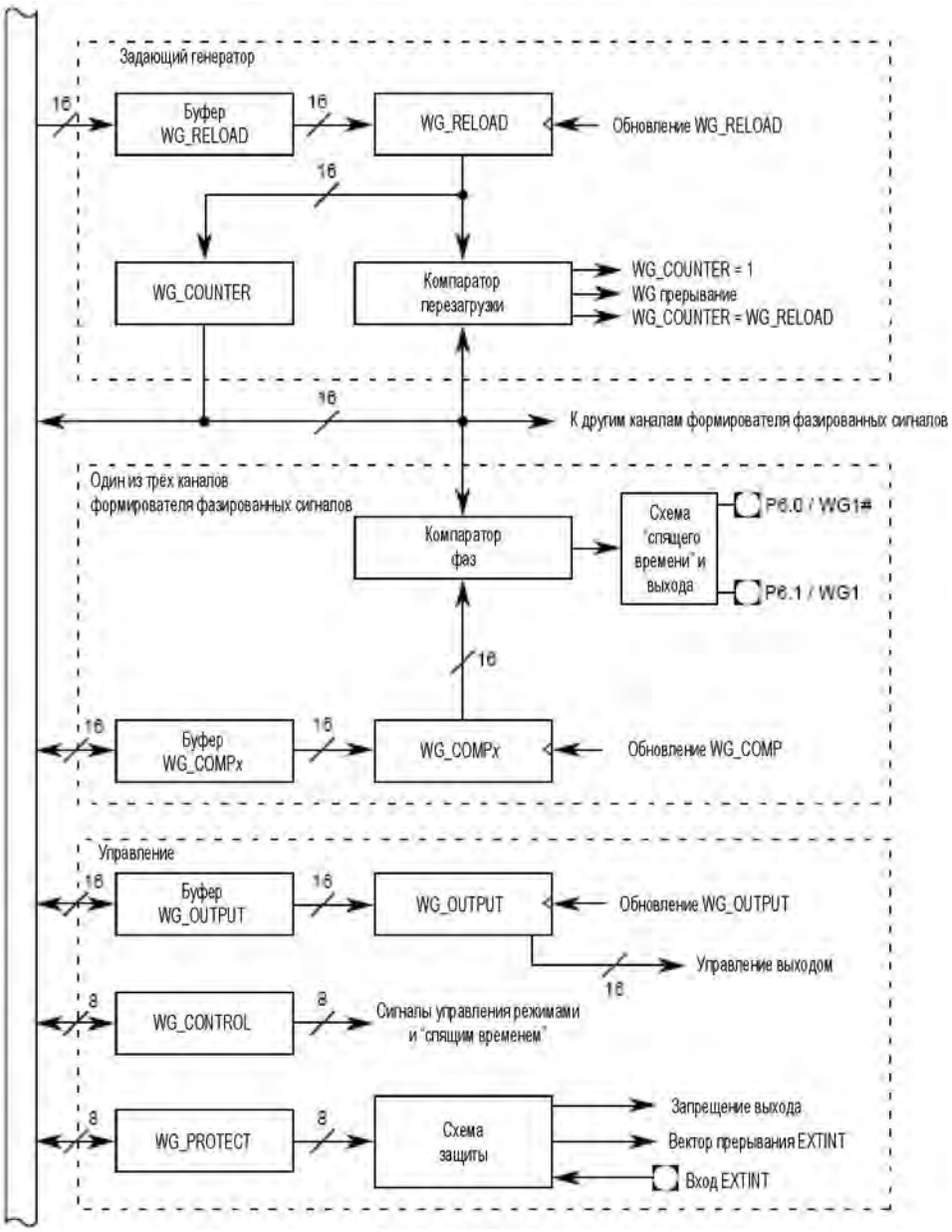


Рисунок 8.1 – Блок-схема генератора формы сигнала

8.2 Сигналы и регистры генератора формы сигнала

Таблица 8.1 описывает сигналы ГФС, а таблица 8.2 кратко описывает регистры состояния и управления.

Таблица 8.1 – Сигналы генератора формы сигнала

Вывод порта	Сигнал ГФС	Тип	Описание
P6.0	WG1#	O	инверсный выход фазы 1 ГФС
P6.1	WG1	O	прямой выход фазы 1 ГФС
P6.2	WG2#	O	инверсный выход фазы 2 ГФС
P6.3	WG2	O	прямой выход фазы 2 ГФС
P6.4	WG3#	O	инверсный выход фазы 3 ГФС
P6.5	WG3	O	прямой выход фазы 3 ГФС
—	EXTINT	I	вход схемы защиты ГФС

Таблица 8.2 – Регистры управления и состояния ГФС

Мнемоника	Адрес	Описание
INT_MASK1	0013H	Маска 1 прерывания. EXTINT бит разрешает или запрещает прерывание EXTINT. PI бит разрешает или запрещает мультиплексированное периферийное прерывание. Соответствующий бит в регистре PI_MASK разрешает или запрещает индивидуальные источники периферийного прерывания.
INT_PEND1	0014H	Ожидание 1 прерывания. Любой установленный бит индицирует ожидаемый запрос прерывания.
PI_MASK	1FBCH	Маска периферийного прерывания. WG бит разрешает или запрещает прерывание ГФС как один из возможных источников мультиплексированного периферийного прерывания. В PI бите в INT_MASK1 должно быть установлено разрешение мультиплексированного периферийного прерывания.
PI_PEND	1FBEH	Ожидание периферийного прерывания. Любой установленный бит индицирует ожидаемый запрос прерывания.
WG_COMP1 WG_COMP2 WG_COMP3	1FC2H 1FC4H 1FC6H	Буферы сравнения ГФС. Каждый буфер сравнения фаз содержит значение, которое сравнивается со значением счётчика. Действие, выполняемое в случае соответствия значений, зависит от операционного режима.
WG_CONTROL	1FCCH	Управление ГФС. Регистр управления определяет операционный режим ГФС, стартует и останавливает счетчик, определяет «мертвое» время для всех фаз и указывает текущее направление счета.
WG_COUNTER	1FCAH	Значение счетчика ГФС. Предназначенный только для чтения регистр-счётчик отражает текущее значение счётчика.
WG_OUTPUT	1FC0H	Управление выходами ГФС. Регистр управления выходами конфигурирует выходы ГФС и выбирает их активную полярность.
WG_PROTECT	1FCEH	Защита ГФС. Защитный регистр разрешает и запрещает схему защиты и выходы, выбирает прерывания по уровню или по фронту и контролирует, какое значение фронта или уровня включит запрос на прерывание.
WG_RELOAD	1FC8H	Значение перезагрузки ГФС. Регистр перезагрузки содержит значение, которое сравнивается со значением счётчика. Действия, выполняемые в случае соответствия значений, зависят от операционного режима.

8.3 Работа ГФС

Этот подраздел описывает главные компоненты ГФС: задающий генератор, формирователи фазированных сигналов, схемы управления и защиты. Далее объясняется, как обновляются буферные регистры, и описывается сходство и различие между «центрированным» и «фронтовым» операционными режимами. Наконец, здесь описывается два типа запросов на прерывание, которые может вырабатывать ГФС и объясняется как разрешать прерывания.

Задающий генератор

Задающий генератор устанавливает период опорного сигнала выводов ШИМ. Определяется этот период записью значения в регистр перезагрузки (WG_RELOAD). Это значение загружается в счётный регистр (WG_COUNTER) при инициализации системы и периодически (в зависимости от операционного режима) впоследствии. Необходимо считать счётный регистр, чтобы определить текущее значение счётчика, и необходимо записывать в регистр перезагрузки, чтобы изменить значение перезагрузки в любое время.

16-битный счетчик опорного сигнала переключается каждый машинный цикл. Регистр управления (WG_CONTROL) разрешает и отключает счетчик, управляет режимом счета и отражает направление счета. Когда счетчик разрешён, он непрерывно считает между 0001H и значением перезагрузки. Запись 0000H в регистр перезагрузки или очистка разрешающего бита в регистре управления останавливает счетчик.

Формирователи фазированных сигналов

Формирователи фазированных сигналов определяют скважность выходных сигналов. Определяется скважность, записывая значение в регистр сравнения каждой фазы (WG_COMPx). Во всех операционных режимах выходы установлены при инициализации, и они остаются установленными, пока значение счётчика (WG_COUNTER) не будет соответствовать значению регистра сравнения фазы (WG_COMPx). В этом случае выходы сброшены и остаются сброшенными, пока не произойдёт другое событие. Событие, которое заставит выводы быть установленными снова, зависит от операционного режима.

Схема генератора «мертвого времени» (рисунок 8.2) предотвращает выход и его инверсный выход от того, чтобы быть установленным в то же самое время. Схема использует два внутренних сигнала WFG и DT, чтобы вырабатывать неперекрывающиеся выходные сигналы. Схема обнаружения фронта формирует сигнал WFG, в то время как 10-битный счетчик «мертвого времени» формирует сигнал DT. Когда обнаружен заданный фронт, в счетчик «мертвого времени» загружается 10-битное значение «мертвого времени» из регистра управления, и DT устанавливается в низкий уровень. Каждый машинный цикл содержимое счётчика уменьшается на 1, пока не достигает нуля, в этом случае счётчик останавливается, и сигнал DT выставляется в высокий уровень. Сигнал WFG логически умножается (AND) с DT, чтобы получить сигнал WG_EVEN; сигнал WFG# логически умножается (AND) с DT, чтобы получить сигнал WG_ODD. Выходы генератора формы сигнала могут быть соединены с сигналами WG_ODD и WG_EVEN.

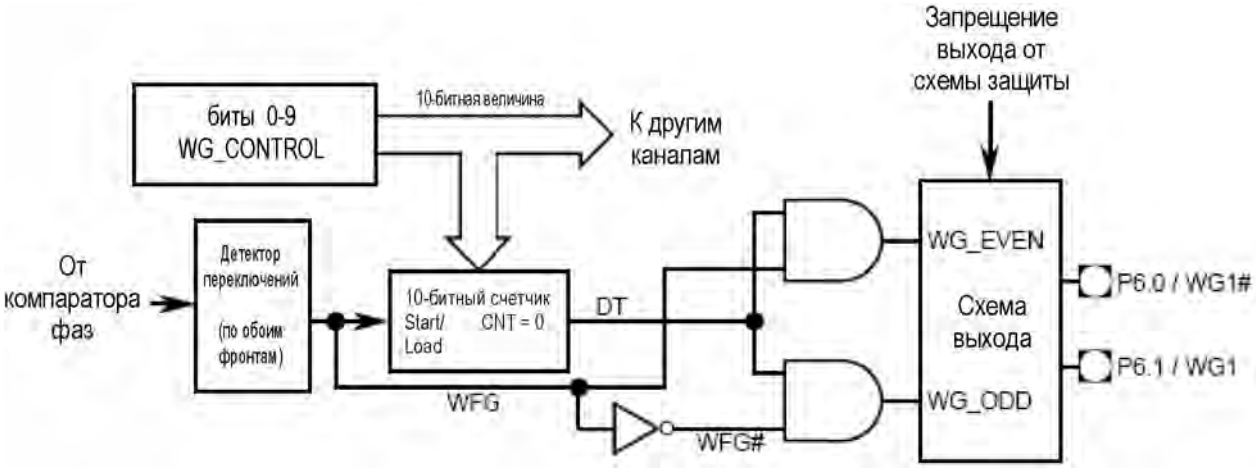


Рисунок 8.2 – Схема генератора «мертвого» времени

Схема управления и защиты

Схема управления содержит регистры управления (WG_CONTROL) и выходные регистры (WG_OUTPUT). Регистр управления разрешает или запрещает счетчик, определяет направление счёта, управляет операционным режимом и определяет «мертвое» время для всех трех фаз. Выходной регистр конфигурирует выходы, определяет полярность выходных сигналов (активный высокий или активный низкий) и задает, как обновляются выходные сигналы (немедленно или синхронно с событием).

Схема защиты (рисунок 8.3) контролирует вход EXTINT. Когда обнаруживается заданное событие на входе, схема одновременно запрещает выходы и вырабатывает запрос на прерывание EXTINT. Программное обеспечение также может запретить выходы, очистив бит разрешения выходов (OE) в регистре защиты (WG_PROTECT).

Запрещённые выходы переходят в неактивные состояния, заданные запрограммированной полярностью.

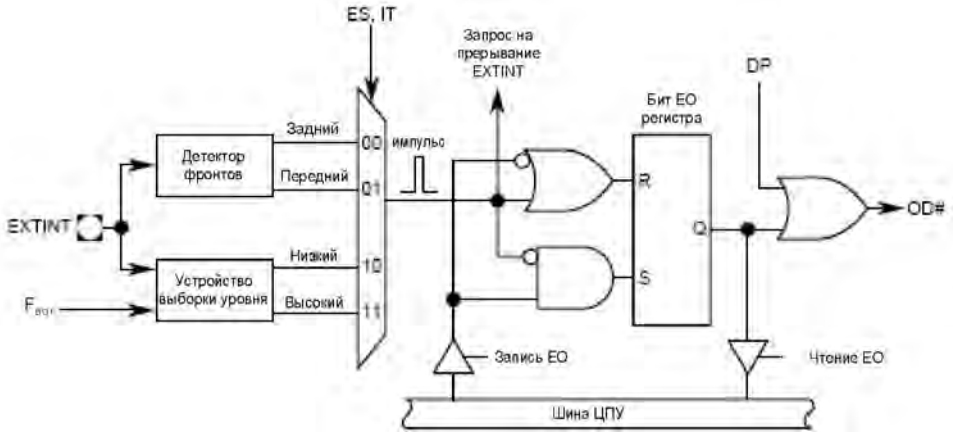


Рисунок 8.3 - Схема защиты

Буферизация и синхронизация регистров

Регистры WG_RELOAD, WG_COMPx и WG_OUTPUT буферизированы. ГФС обновляет регистры синхронно во избежание ошибочных или несимметричных рабочих циклов.

При записи в буферы WG_COMPx при остановленном счетчике (или когда в счётном регистре – нуль, или когда очищен бит разрешения счёта в регистре управления) содержимое регистров обновляется спустя половину машинного цикла.

Регистр WG_RELOAD обновляется, когда значение счётчика достигает значения перезагрузки. В регистр WG_COUNTER загружается обновленное значение WG_RELOAD, так что новое значение перезагрузки вступает в силу для следующего цикла. В режиме 3 регистр WG_RELOAD может быть обновлён, когда происходит

событие EPA. Для этого необходимо, чтобы была разрешена периферийная функция канала EPA.

Регистр WG_OUTPUT содержит бит синхронизации, который задаёт, как изменения отразятся на выходных сигналах – немедленно или синхронно с событием. Бит синхронизации не буферизирован, поэтому изменения в нём происходят немедленно. Необходимо инициализировать синхронизацию в нуль (изменения отражаются немедленно), чтобы гарантировать, что выходы находятся в желаемых состояниях при старте счётчика.

Режимы работы

Генератор формы сигнала может работать в «центрированном» или «фронтовом» режимах. В «центрированных» режимах счётчик считает и вверх, и вниз; во «фронтовых» режимах он считает только вверх. «Центрированные» режимы формируют выходные сигналы ШИМ, более эффективные для управления 3-фазными индукционными двигателями переменного тока, в то время как «фронтовые» режимы формируют стандартные сигналы ШИМ. «Центрированные» выходные сигналы имеют меньше гармоник с двойным периодом несущей, чем «фронтовые».

Начальное состояние ГФС одинаково для всех режимов работы. После системного включения питания или сброса счётчик остановлен, выводы сброшены, и все регистры очищены. Значения, записываемые в регистры, заносятся спустя половину такта.

Главные различия между «центрированными» и «фронтовыми» режимами: значение счётчика после инициализации, направление счета и условия, которые влекут за собой изменение в состоянии выходов. В таблице 8.3 приведены операции блока для «центрированного» и «фронтового» режимов.

Таблица 8.3 – Работа в «центрированных» и «фронтовых» режимах

Шаг	«Центрированные» режимы	«Фронтовые» режимы
1	Загрузка в WG_COUNTER значения WG_RELOAD. Выходы остаются сброшенными.	Загрузка в WG_COUNTER значения 0001H. Выходы остаются сброшенными.
2	Когда счетчик разрешён, он начинает обратный отсчёт. Когда WG_COUNTER достигает 1, ждёт 1 такт, затем начинает считать вверх. С началом счёта вверх выходы установлены.	Когда счетчик разрешён, он начинает считать вверх. С началом счёта вверх выходы установлены.
3	Когда WG_COUNTER достигает значения WG_COMPx при счёте вверх, выходы соответствующих фаз сбрасываются, и счёт вверх продолжается.	Когда WG_COUNTER достигает значения WG_COMPx, выходы соответствующих фаз сбрасываются, и счёт вверх продолжается.
4 5 6	Когда WG_COUNTER достигает значения WG_RELOAD, начинается обратный счёт. Когда WG_COUNTER достигает значения WG_COMPx при обратном счёте, выходы соответствующих фаз устанавливаются, и обратный счёт продолжается. Когда WG_COUNTER достигает 1, выходы сбрасываются, WG_RELOAD обновляется, и происходит переход к шагу 1.	Когда WG_COUNTER достигает значения WG_RELOAD, WG_RELOAD обновляется, и происходит переход к шагу 1.

Главные различия между «центрированными» и «фронтовыми» режимами – события, управляющие обновлениями регистра управления. В таблице 8.4 приведён список событий, которые могут повлечь обновление регистров, и регистры, которые обновляются в каждом режиме.

Таблица 8.4 - Обновление регистров

Событие	«Центрированные» режимы		«Фронтовые» режимы	
	Режим 0	Режим 1	Режим 2	Режим 3
	Обновляемые регистры		Обновляемые регистры	
WG_COUNTER= =WG_RELOAD	WG_RELOAD WG_COUNTER WG_COMPx WG_OUTPUT †	WG_RELOAD WG_COUNTER WG_COMPx WG_OUTPUT †	WG_RELOAD WG_COUNTER WG_COMPx WG_OUTPUT †	WG_RELOAD WG_COUNTER WG_COMPx WG_OUTPUT †
WG_COUNTER = 1	—	WG_COMPx	—	—
Событие EPA	WG_OUTPUT †	WG_OUTPUT †	WG_OUTPUT †	WG_RELOAD WG_COUNTER WG_COMPx WG_OUTPUT †

† Регистр WG_OUTPUT обновляется при этих условиях, если его бит синхронизации установлен; иначе изменения выполняются немедленно.

«Центрированные» режимы

В «центрированных» режимах счетчик считает вниз от значения WG_RELOAD до 1, затем меняет направление счёта и считает вверх от 1 до WG_RELOAD. Когда записывается в регистр WG_RELOAD, в WG_COUNTER загружается значение перезагрузки. Когда устанавливается бит разрешения в регистре управления, счетчик начинает считать вниз и продолжает счёт до достижения 1, ждет один такт и начинает считать вверх вплоть до достижения значения WG_RELOAD. В этот момент WG_RELOAD обновляется, а в WG_COUNTER загружается обновлённое значение, так что новое значение перезагрузки вступает в силу в следующем такте. Счетчик продолжает обратный счёт от WG_RELOAD до 1. Он производит «симметричный» счёт вверх и вниз, иллюстрированный треугольным сигналом на рисунке 8.4, с периодом равным удвоенному значению WG_RELOAD. Рисунок 8.5 показывает поведение выходов и прерываний в «центрированных» режимах.

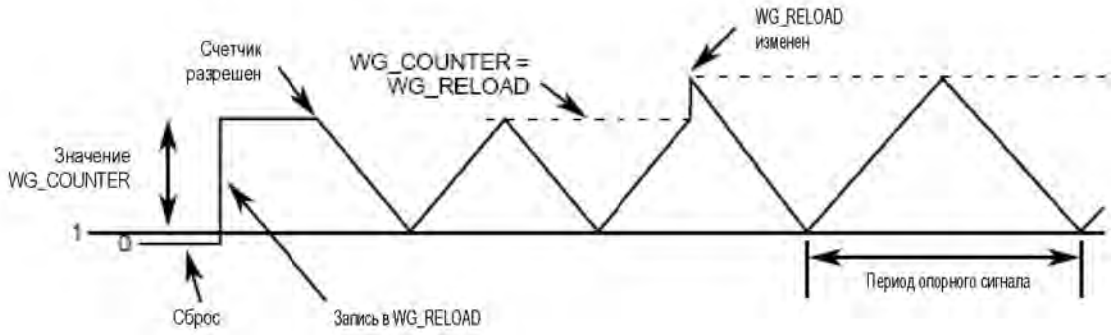


Рисунок 8.4 – Работа счётчика в «центрированных» режимах

В режиме 0 WG_COMPx и регистры WG_OUTPUT обновляются только однажды в течение периода опорного сигнала, когда счетчик достигает значения перезагрузки.

В режиме 1 эти регистры обновляются дважды в течение периода несущей: сначала, когда счетчик установлен в 1, и второй раз, когда он достигает значения перезагрузки.

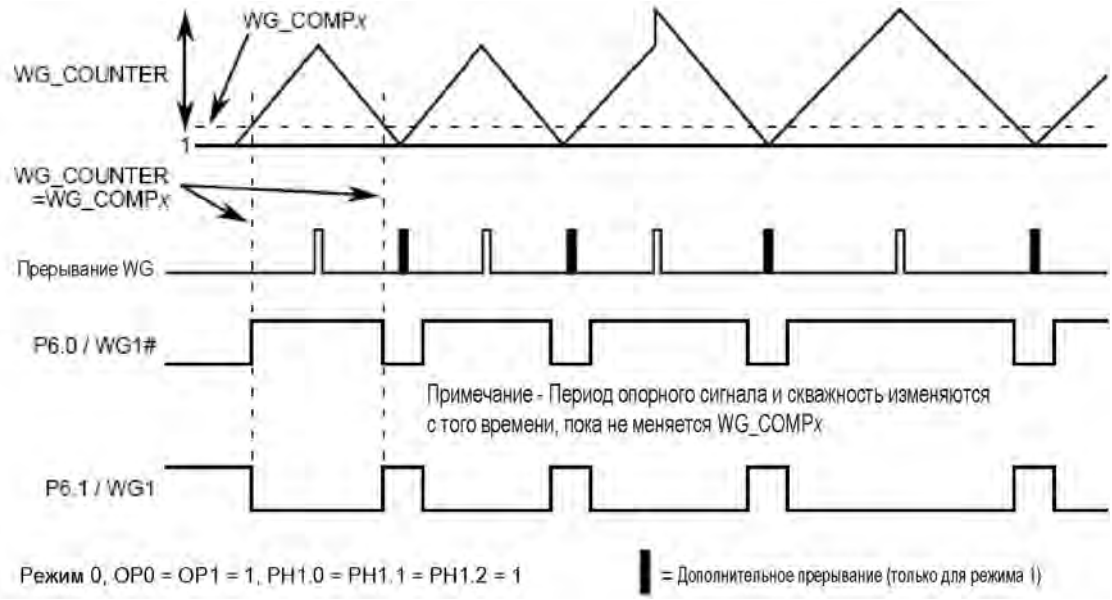


Рисунок 8.5 – Диаграмма выходных сигналов в «центрированных» режимах

«Фронтовые» режимы

Во «фронтовых» режимах счетчик считает вверх, начиная с 1 до значения WG_RELOAD. Когда записывается в регистр WG_RELOAD, в WG_COUNTER загружается 0001H. Когда устанавливается бит разрешения в регистре управления, счетчик начинает считать вверх и продолжает считать, пока не достигает значения WG_RELOAD или (только в режиме 3) пока не произойдёт событие EPA. В этот момент в WG_COUNTER вновь загружается 0001H, и WG_RELOAD обновляется, так что новое значение перезагрузки вступает в силу в следующем такте. Счетчик продолжает считать вверх от 0001H до WG_RELOAD. Он производит «гладко возрастающий» счет, иллюстрированный пилообразным сигналом на рисунке 8.6, с периодом равным значению WG_RELOAD. Рисунок 8.7 показывает поведение выходов и прерываний во «фронтовых» режимах.

В режиме 2 регистры обновляются только однажды в течение периода опорного сигнала, когда счетчик достигает значения перезагрузки. В режиме 3 регистры также обновляются, когда происходит событие внешней функции EPA. (Необходимо конфигурировать канал EPA для этой функции, см. раздел 10 “Процессор событий (EPA)” для информации.

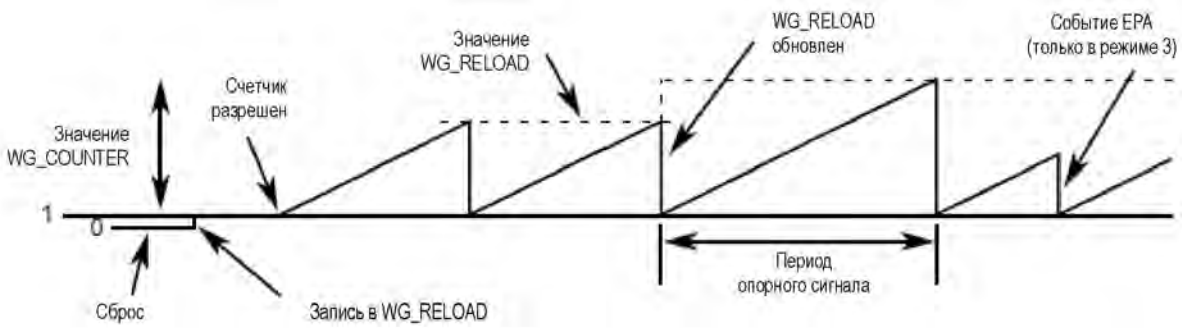


Рисунок 8.6 – Работа счётчика во «фронтовых» режимах

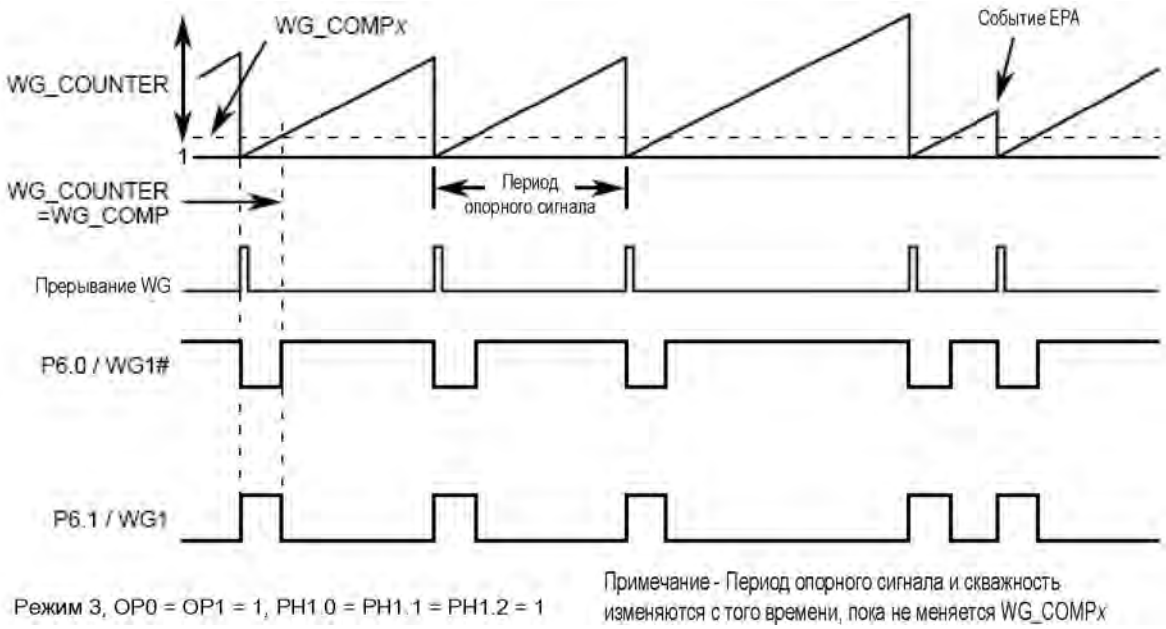


Рисунок 8.7 – Диаграмма выходных сигналов во «фронтовых» режимах

8.4 Программирование ГФС

Этот подраздел объясняет, как конфигурировать ГФС и устанавливать его состояние.

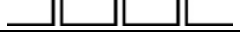
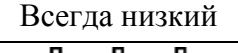
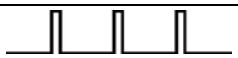
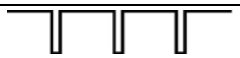
Конфигурирование выходов

Выходы генератора формы сигнала мультиплексированы с выходом общего назначения порта 6 таким образом, что необходимо конфигурировать их как сигналы специальных функций, чтобы использовать их как выводы ГФС. Регистр WG_OUTPUT (рисунок 8.8) конфигурирует выводы, устанавливает полярность выхода и управляет, как изменяются выходные сигналы, синхронно с событием или немедленно.

Четыре бита WG_OUTPUT независимы от ГФС. Они конфигурируют выходы для широтно-импульсного модулятора (ШИМ), который также совместно использует выводы с портом 6. P6 и PE6 биты управляют выводом P6.6/PWM0, а P7 и PE7 биты управляют выводом P6.7/PWM1. Их размещение в этом регистре позволяет конфигурировать все выводы порта 6 единственной записью в WG_OUTPUT.

Таблица 8.5 показывает комбинации битов, необходимые для задания высокого или низкого уровня выходов ГФС или подключения их к сигналам WG_EVEN или WG_ODD. Необходимо обратить внимание, что PHx.2 всегда установлен для выбора функции сигнала ГФС (очистка PHx.2 выбирает функцию общего назначения порта ввода-вывода). Столбец "Полярность выхода" показывает полярность выходных сигналов. Рисунки в таблице 8.5 показывают скважность приблизительно 15 %, и в этих случаях длительность высокого уровня сигналов увеличивается с увеличением «мертвого» времени.

Таблица 8.5 – Конфигурация выходов

PHx.2	PHx.1	PHx.0	Значение выхода		Полярность выхода	
			WGx	WGx#	WGx	WGx#
1	0	0	Низкий	Низкий	Всегда низкий	Всегда низкий
1	0	1	Низкий	WG_EVEN#	Всегда низкий	
1	1	0	WG_ODD	Низкий		Всегда низкий
1	1	1	WG_ODD	WG_EVEN		
Примечание – Эта таблица подразумевает активные высокие выходные сигналы (OP1=OP0=1).						

WG_OUTPUT (Генератор формы сигнала) Адрес: 1FC0H

Состояние после сброса: 0000H

Регистр конфигурации выходов ГФС (WG_OUTPUT) управляет конфигурацией выводов генератора формы сигнала и модуля ШИМ. ГФС и модуль ШИМ совместно используют выводы порта 6. Наличие этих битов управления в одном регистре дает возможность конфигурировать весь порт 6 одной записью в WG_OUTPUT.

15							8
OP1	OP0	SYNC	PE7	PE6	PH3.2	PH2.2	PH1.2
7							0
P7	P6	PH3.1	PH3.0	PH2.1	PH2.0	PH1.1	PH1.0

Номер бита	Мнемоника	Функция
15	OP1	Полярность выхода. Выбирает полярность выхода для сигналов инверсных фаз WG1#, WG2# и WG3#. 0 = активно-низкие выходы. 1 = активно-высокие выходы.
14	OP0	Полярность выхода. Выбирает полярность выхода для сигналов прямых фаз WG1, WG2 и WG3. 0 = активно-низкие выводы. 1 = активно-высокие выводы.
13	SYNC	Синхронизация. Выбирает как происходит обновление регистра WG_OUTPUT: синхронно ли с другим событием или немедленно после того, как он изменится. 0 = обновляет WG_OUTPUT немедленно. 1 = обновляет WG_OUTPUT синхронно с событием. Чтобы гарантировать, что выходы находятся в нужных состояниях при запуске ГФС, необходимо при инициализации очистить этот бит, затем установить его позже, чтобы в последующем WG_OUTPUT обновлялся синхронно с событием. (Таблица 9.4 перечисляет события, которые обновляют WG_OUTPUT в каждом режиме.)
12	PE7	P6.7/PWM1. Выбирает функцию порта или функцию выхода ШИМ на выводе P6.7/PWM1. 0 = P6.7 1 = PWM1
11	PE6	P6.6/PWM0. Выбирает функцию порта или функцию выхода ШИМ на выводе P6.6/PWM0. 0 = P6.6 1 = PWM0
10	PH3.2	Функция фазы 3. Выбирает или функцию порта, или функцию выхода ГФС для выводов P6.4/WG3# и P6.5/WG3. 0 = P6.4, P6.5 1 = WG3#, WG3

Номер бита	Мнемоника	Функция
9	PH2.2	Функция фазы 2. Выбирает или функцию порта, или функцию выхода ГФС для выводов P6.2/WG2# и P6.3/WG2. 0 = P6.2, P6.3 1 = WG2#, WG2
8	PH1.2	Функция фазы 1. Выбирает или функцию порта или функцию выхода ГФС для выводов P6.0/WG1# и P6.1/WG1. 0 = P6.0, P6.1 1 = WG1#, WG1
7	P7	P6.7/PWM1 значение. Нужное значение P6.7/PWM1 записывают в этот бит.
6	P6	P6.6/PWM0 значение. Нужное значение P6.6/PWM0 записывают в этот бит.
5:4	PH3.1:0	P6.4/WG3#, P6.5/WG3 значения. Нужные значения выходов записывают в эти биты, см. таблицу 8.5.
3:2	PH2.1:0	P6.2/WG2#, P6.3/WG2 значения. Нужные значения выходов записывают в эти биты, см. таблицу 8.5.
1:0	PH1.1:0	P6.0/WG1#, P6.1/WG1. Нужные значения выходов записывают в эти биты, см. таблицу 9.5.

Рисунок 8.8 – Регистр конфигурации выходов ГФС (WG_OUTPUT)

Управление схемой защиты и выработкой прерывания EXTINT

Регистр защиты (рисунок 8.9) управляет схемой защиты и запросами прерывания EXTINT.

WG_PROTECT Адрес: 1FCEH
Состояние после сброса (MC, MD) F0H
Состояние после сброса (MH): E0H

Регистр защиты ГФС (WG_PROTECT) разрешает и запрещает выходы и схему защиты. Он также выбирает прерывания EXTINT или по уровню, или по фронту и выбирает, который уровень или фронт произведет запрос на прерывание EXTINT.

7							0
–	–	–	–	ES	IT	DP	EO

Номер бита	Мнемоника	Функция
		Зарезервирован.
7:5	—	Для совместимости с будущими устройствами записывать нули в эти биты.

3:2	ES IT	Разрешение выборки и тип прерывания. Бит ES выбирает, что схема защиты производит выборку по уровню сигнала EXTINT или детектирует перепад сигнала (фронт), в то время как бит IT задаёт, какое значение фронта или уровня вызывает запрос на прерывание. Возможные комбинации следующие: ES IT Событие 0 0 задний фронт 0 1 передний фронт 1 0 низкий уровень 1 1 высокий уровень
1	DP	Запрещение защиты. Этот бит разрешает и запрещает схему защиты. 0 = разрешение защиты 1 = запрещение защиты
0	EO	Разрешение выходов. Этот бит разрешает и запрещает выходы. 0 = запрещение выходов 1 = разрешение выходов

Рисунок 8.9 – Регистр защиты ГФС (WG_PROTECT)

Задание периода опорного сигнала и скважности

Регистр перезагрузки (WG_RELOAD) и регистры сравнения фаз (WG_COMPx) управляют периодом опорного сигнала (несущей) и скважностью. Чтобы установить период опорного сигнала, его значение записывается в регистр перезагрузки (рисунок 8.10). Чтобы определить отрезок времени, в течение которого соответствующие выходы останутся установленными, его значение записывается в каждый регистр сравнения фаз.

WG_RELOAD Адрес: 1FC8H

Состояние сброса: 0000H

Регистр перезагрузки ГФС (WG_RELOAD) и регистры сравнения фаз (WG_COMPx) управляют периодом опорного сигнала (несущей) и скважностью. Запись значения в регистр перезагрузки устанавливает период опорного сигнала.

При изменении значения WG_RELOAD изменяются и период опорного сигнала, и скважность, потому что выводы остаются установленными в течение постоянного отрезка времени, в то время как счетчик требует большего времени, чтобы закончить цикл. Чтобы изменить период несущей, не изменяя скважность, необходимо пропорционально изменить и WG_RELOAD, и WG_COMPx в одно и то же время, немедленно после прерывания.

15	0
Reload	

Номер бита	Функция
15:0	Перезагрузка Этот регистр определяет период опорного сигнала (несущей). Используются следующие формулы, чтобы вычислить период несущей и скважность: $T_{CARRIER}$ = (*множитель* × WG_RELOAD)/F_{BQ1} Скважность = (WG_COMPx/WG_RELOAD) × 100 % , где $T_{CARRIER}$ = период несущей, мкс,

F_{BQ1} = частота входного сигнала на входе BQ1, МГц,
множитель = 4 для «центрированных» режимов; равен 2 для «фронтовых» режимов,
WG_RELOAD = 16-битное значение WG_RELOAD ≥ WG_COMPx,
WG_COMPx = 16-битное значение WG_COMPx ≤ WG_RELOAD.

Рисунок 8.10 – Регистр перезагрузки ГФС (WG_RELOAD)

WG_COMPx — Адрес: 1FC2H, 1FC4H, 1FC6H
x = 1–3 — Состояние Сброса: 0000H

Регистр сравнения фаз (WG_COMPx) управляет скважностью сигнала каждой фазы. Записывается значение в каждый регистр сравнения фаз, чтобы определить отрезок времени, в течение которого соответствующие выходы останутся установленными.

Изменение значения WG_RELOAD изменяет и период несущей и скважность, потому что выводы остаются установленными в течение постоянного отрезка времени, в то время как счетчик требует большего времени, чтобы закончить цикл. Чтобы изменить период опорного сигнала, не изменяя скважность, необходимо пропорционально изменить WG_RELOAD и WG_COMPx в одно и то же время, немедленно после прерывания.

15							0
Сравнение							

Номер бита	Функция
15:0	Сравнение Эти биты определяют отрезок времени, в течение которого соответствующие выходы установлены. Используются следующие формулы, чтобы вычислить время установленного выхода и скважность: T_{OUTPUT} = (*множитель* × WG_RELOAD)/F_{BQ1} Скважность = (WG_COMPx/WG_RELOAD) × 100 %, где T_{OUTPUT} = полное время, когда выход установлен, мкс, F_{BQ1} = частота входного сигнала на входе BQ1, МГц, *множитель* = 4 для «центрированных» режимов; равен 2 для «фронтовых» режимов, WG_RELOAD = 16-битное значение WG_RELOAD ≥ WG_COMPx, WG_COMPx = 16-битное значение WG_COMPx ≤ WG_RELOAD.

Рисунок 8.11 – Регистр сравнения фаз (WG_COMPx)

Определение режима работы, «мёртвого» времени и старт счётчика

Регистр управления ГФС (WG_CONTROL) (рисунок 8.12) определяет «мёртвое» время и режим работы, разрешает и запрещает счётчики. Бит только для чтения CS указывает текущее направление счёта.

WG_CONTROL — Адрес: 1FCCH
Состояние Сброса 00C0H

15							8
—	M2	M1	M0	CS	EC	DT9	DT8
7							0
DT7	DT6	DT5	DT4	DT3	DT2	DT1	DT0

Номер Мнемо- Функция

бита	ника	
15		Зарезервирован.
	—	Для совместимости с последующими устройствами записывается нуль в этот бит.
14:12	M2:0	Режим работы. Это поле управляет режимом работы ГФС. M2 M1 M0 Режим 0 0 0 0 «центрированный»; регистры обновляются однажды 0 0 1 1 «центрированный»; регистры обновляются дважды 0 1 0 2 «фронтовой»; регистры обновляются однажды 0 1 1 3 «фронтовой»; регистры обновляются дважды
11	CS	Состояние счётчика. Этот бит только для чтения указывает, считает ли счетчик вверх или в обратном порядке. 0 = счет вниз 1 = счет вверх
10	EC	Разрешение счетчика. Этот бит запускает или останавливает счетчик. 0 = запретить (остановить) счетчик 1 = разрешить (запустить) счетчик
9:0	DT9:0	«Мёртвое» время. Это поле определяет «мертвое» время для всех трех фаз. Используется следующая формула, чтобы вычислить соответствующее значение DT_VALUE: $DT\_VALUE=T_{DEAD}\times F_{BQ1}/2$, где T_{DEAD} = «мёртвое» время, мкс, F_{BQ1} = частота входного сигнала на входе BQ1, МГц.

Рисунок 8.12 – Регистр управления ГФС (WG_CONTROL)

8.5 Определение состояния ГФС

Считается WG_CONTROL (рисунок 8.12), чтобы определить текущее значение «мёртвого» времени, состояние счётчика, направление счета и режим работы. Считается WG_COUNTER (рисунок 8.13), чтобы определить текущее значение счётчика.

WG_COUNTER Адрес: 1FCAH
Состояние сброса XXXXH

Считается регистр-счетчик генератора формы сигнала (WG_COUNTER), чтобы определить текущее значение счётчика.

15	0
Значение счётчика	

Номер бита	Функция
15:0	Значение счётчика. Этот регистр отражает текущее значение счётчика.

Рисунок 8.13 – Регистр-счётчик генератора формы сигнала (WG_COUNTER)

8.6 Разрешение прерываний генератора формы сигнала

Генератор формы сигнала может производить два типа запросов на прерывание. Запрос на прерывание WG вызывается счетчиком, в то время как прерывание EXTINT вызывается внешним событием.

В режиме 0 запрос на прерывание WG производится однажды в период, когда счетчик достигает значения WG_RELOAD. В режиме 1 запрос на прерывание WG производится дважды в течение периода: сначала – когда счетчик достигает 1 и снова – когда он достигает значения WG_RELOAD. Во «фронтовых» режимах запрос на прерывание WG производится однажды в конце каждого периода, когда в счетчик загружена 1.

Схема защиты управляет прерыванием EXTINT. Два бита в регистре защиты управляют, какой тип внешнего события произведет запрос на прерывание: задний или передний фронт или низкий или высокий уровень.

Схема обнаружения фронта требует, чтобы сигнал оставался установленным в течение по крайней мере 2 тактов, которые будут отсчитаны от нужного фронта. Схема выборки требует, чтобы сигнал оставался установленным в течение по крайней мере 24 тактов, которые будут отсчитаны от нужного уровня. Схема производит выборки входного уровня 3 раза в течение этого 24-тактового периода и признает сигнал «правильным», только если это подтверждено для каждой выборки. Осуществление выборки уровня полезно для сред, в которых шумовые пики могли бы причинить нежелательные прерывания, если бы использовалось обнаружение фронта.

Чтобы разрешить прерывания, необходимо установить соответствующие биты масок в регистре маски и выполнить команду EI, чтобы разрешить обслуживание прерываний. Необходимо сосчитать регистр ожидания прерываний, чтобы определить любые отложенные прерывания.

9 Широтно-импульсный модулятор

Широтно-импульсный модулятор (ШИМ, PWM – Pulse Width Modulator) имеет два выхода, каждый из которых может выводить ШИМ сигнал с фиксированной программируемой частотой и переменной скважностью. Эти выходы могут использоваться, чтобы управлять двигателями, которым необходимы не фильтрованные цифровые сигналы ШИМ для оптимальной эффективности, или они могут быть фильтрованы, чтобы получить сглаженный аналоговый сигнал.

В этом разделе проводится краткий функциональный обзор широтно-импульсного модулятора, описано как программировать его и обеспечивать типовую схему для преобразования ШИМ сигналов в аналоговые сигналы.

9.1 Краткий обзор функционирования блока ШИМ

Блок ШИМ имеет два канала, каждый из которых состоит из регистра управления (PWMx_CONTROL), буфера, компаратора, RS-триггера и выходного блока. Два других компонента, счетчик на восемь битов (PWM_COUNT) и регистр периода (PWM_PERIOD) для двух каналов модуля ШИМ, дополняют схему (см. рисунок 9.1).

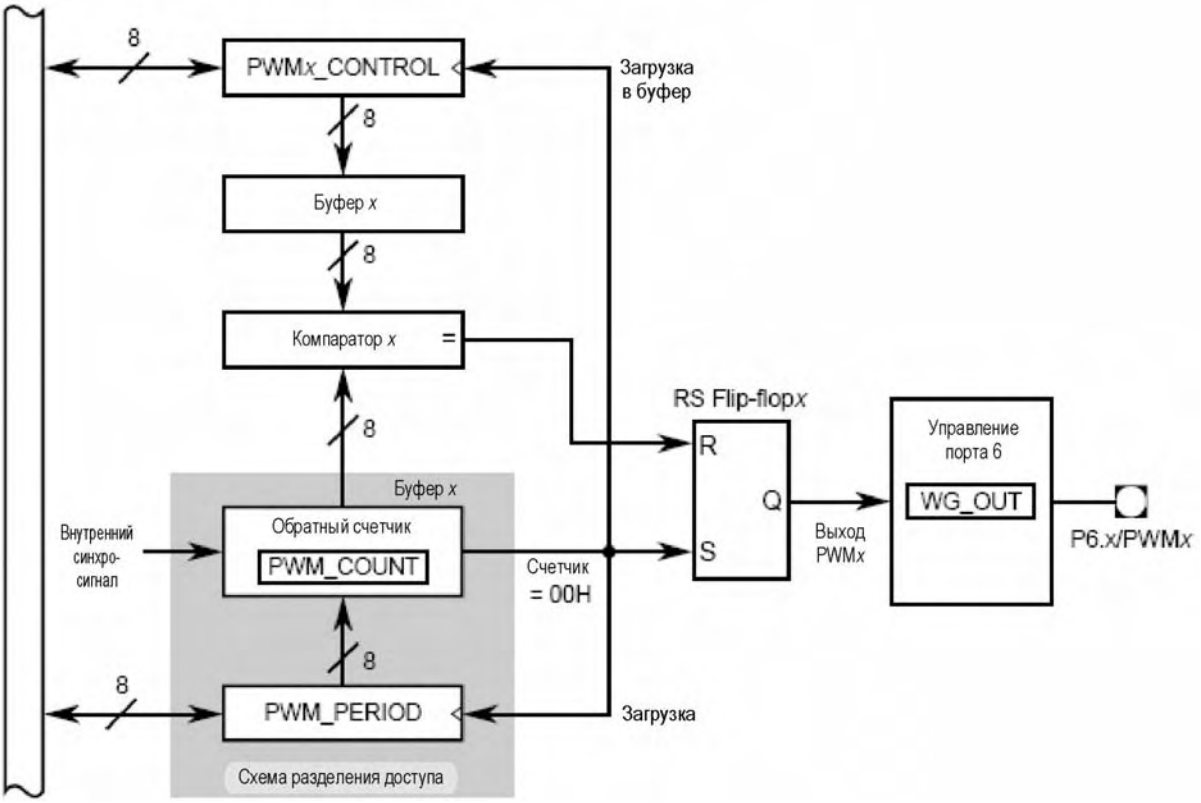


Рисунок 9.1 – Блок-схема блока ШИМ

9.2 Сигналы и регистры блока ШИМ

Таблица 9.1 описывает ШИМ сигналы, а таблица 9.2 кратко описывает управляющие регистры и регистры состояния.

Таблица 9.1 – ШИМ сигналы

Вывод порта	ШИМ сигнал	Тип ШИМ сигнала	Описание
P6.6	PWM0	Выход	Широтно-импульсный выход 0 канала с повышенной нагрузочной способностью.
P6.7	PWM1	Выход	Широтно-импульсный выход 1 канала с повышенной нагрузочной способностью.

Таблица 9.2 – Регистры управления и состояния ШИМ

Мнемонический	Адрес	Описание
PWM0_CONTROL PWM1_CONTROL	1FB0H 1FB2H	Скважность ШИМ. Этот регистр управляет скважностью ШИМ. Обнуление этого регистра ШИМ приводит к формированию сигнала со скважностью близкой к нулю (0 %). FFH в этом регистре заставит ШИМ сформировать сигнал со скважностью близкой к единице (99,6 %).
PWM_PERIOD	1FB4H	Период ШИМ. Этот регистр содержит запрограммированное значение, которое определяет период сигнала ШИМ. Значение перегружается в счетчик каждый раз, когда счетчик переустанавливается в FFH.
PWM_COUNT	1FB6H	Счетчик ШИМ. Это доступный только для чтения регистр, содержит текущее значение счетчика обратного счета.
WG_OUTPUT	1FC0H	Выход генератора формы сигнала. Биты 11 и 12 (PE6 и PE7) определяют, функционирует ли соответствующий вывод как стандартный вывод порта ввода-вывода или как выход ШИМ. Биты 6 и 7 (P6 и P7) определяют выходное состояние вывода, когда выбрана функция выходного порта. PEx Px на выходе 0 0 0 0 1 1 1 X ШИМ выход

9.3 Работа широтно-импульсного модулятора

Регистр периода (PWM_PERIOD) управляет частотой обоих выходных ШИМ сигналов. Каждый регистр управления (PWMx_CONTROL) управляет скважностью (отношением ширины импульса к периоду, в процентах) ШИМ сигнала на соответствующем выводе. Каждый регистр управления содержит 8 битное значение, которое загружается в буфер, когда 8 битный счетчик меняет значение из 00H в FFH. Компараторы сравнивают содержимое буферов со значением счетчика. Так как значение, записанное в регистр управления, сохраняется в буфере, необходимо записать новое 8-битное значение в PWMx_CONTROL в любое время. Однако компараторы не определяют новое значение, пока счетчик не завершил обратный счет по предыдущему значению. Новое значение используется в течение следующего периода ШИМ сигнала.

Счетчик считает в обратном порядке к 00H, в это время ШИМ выход переключается в состояние высокого уровня, счетчик загружает содержимое регистра PWM_PERIOD, и содержимое регистров управления загружается в буферы. На выводе ШИМ уровень остается высоким, пока значение счетчика соответствует значению в буфере, иначе на выводе будет низкий уровень. Необходимо читать регистр PWM_COUNT, чтобы знать текущее значение счетчика. Когда счетчик переустанавливается снова (то есть, когда происходит переполнение), вывод переключается в высокий уровень. Загрузка в PWMx_CONTROL значения 00H переводит вывод в состояние низкого уровня. На рисунке 9.2 показаны типичные формы сигналов на ШИМ выходах.

Примечание – Значение регистра PWMx_CONTROL и соответствующая результирующая скважность (Duty Cycle) на рисунке 9.2 являются верными только тогда, когда значение регистра PWM_PERIOD равно FFH.

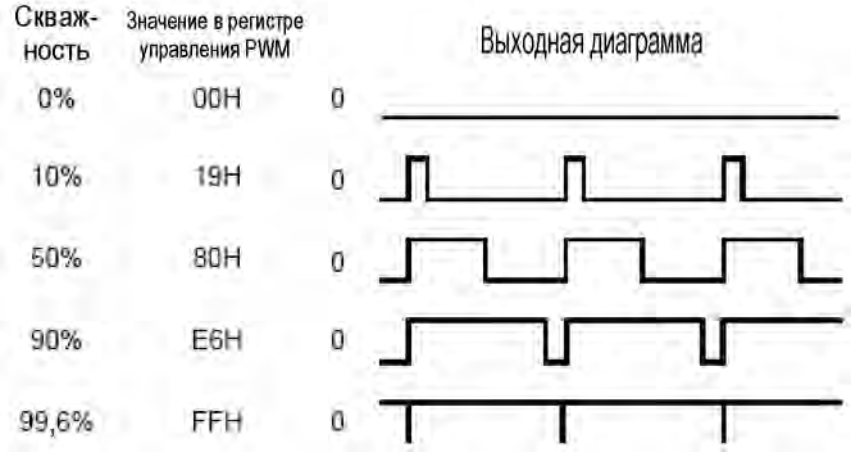


Рисунок 9.2 – Форма ШИМ сигнала

9.4 Программирование частоты и периода

Частота входного сигнала на BQ1 (F_{BQ1}) и содержимое регистра PWM_PERIOD определяет частоту сигнала (F_{PWM}) и период (T_{PWM}) на выводе ШИМ. Таблица 9.3 показывает частоту сигнала на выводе ШИМ при определенном значении F_{BQ1} с различными значениями PWM_PERIOD. Чтобы вычислить желательные значения периода и частоты для сигнала ШИМ вывода и записать их в регистр PWM_PERIOD используют следующие формулы:

$$T_{PWM}(\text{мкс}) = \frac{512 \times (PWM\_PERIOD + 1)}{F_{BQ1}},$$

$$F_{PWM}(\text{МГц}) = \frac{F_{BQ1}}{512 \times (PWM\_PERIOD + 1)},$$

где

PWM_PERIOD – 8 битное значение для загрузки в регистр PWM_PERIOD
F_{BQ1} – частота входного сигнала на выводе BQ1, в МГц
T_{PWM} – период выходного сигнала на выводе ШИМ, в мкс
F_{PWM} – частота выходного сигнала на выводе ШИМ, в МГц

Таблица 9.3 – Частота выходного сигнала ШИМ (F_{PWM})

PWM_PERIOD	Частота выходного сигнала ШИМ (F_{PWM})		
	F_{BQ1}=8 МГц	F_{BQ1}=10 МГц	F_{BQ1}=16 МГц
1	2	3	4
00H	15,6 кГц	19,5 кГц	31,2 кГц
0FH	976,6 Гц	1220,7 Гц	1953,1 Гц
1FH	488,3 Гц	610,3 Гц	976,6 Гц
2FH	325,5 Гц	406,9 Гц	651,0 Гц
3FH	244,1 Гц	395,2 Гц	488,3 Гц
4FH	195,3 Гц	244,1 Гц	390,6 Гц
5FH	162,8 Гц	203,4 Гц	325,5 Гц
6FH	139,5 Гц	174,4 Гц	279,0 Гц
7FH	122,1 Гц	152,6 Гц	244,1 Гц
8FH	108,5 Гц	135,6 Гц	217,0 Гц
9FH	97,7 Гц	122,1 Гц	195,3 Гц

1	2	3	4
AFH	88,8 Гц	111,0 Гц	177,6 Гц
BFH	81,4 Гц	101,7 Гц	1628 Гц
CFH	75,1 Гц	93,9 Гц	150,2 Гц
DFH	69,7 Гц	87,2 Гц	139,5 Гц
EFH	65,1 Гц	81,4 Гц	130,2 Гц
FFH	61,0 Гц	76,0 Гц	122,0 Гц

PWM_PERIOD Адрес: 1FB4H

Состояние сброса: 00H

Регистр периода ШИМ (PWM_PERIOD) управляет периодом выходных ШИМ сигналов. Он содержит значение, которое определяет число состояний счета, необходимое для инкрементирования счетчика ШИМ. Значение PWM_PERIOD загружается в регистр счетчика периода ШИМ всякий раз, когда значение в счетчике равно 0.

Таблица 9.4 – Регистр периода ШИМ (PWM_PERIOD)

7	0
Период сигнала ШИМ	

Номер разряда	Функция
7:0	Период сигнала ШИМ. Этот регистр управляет периодом сигнала на выводах ШИМ. Значение PWM_PERIOD загружается в регистр счетчика периода ШИМ всякий раз, когда значение в счетчике равно 0.

9.5 Программирование скважности (Duty cycle)

Значения, записанные в PWMx_CONTROL и в регистр PWM_PERIOD, управляют шириной пульса высокого уровня, фактически управляя скважностью. 8-битное значение, записанное в регистр управления, загружается в буфер, и это значение используется в течение следующего периода. Для вычисления желаемой скважности по данным значениям PWMx_CONTROL и PWM_PERIOD и последующей записи этих значений в соответствующие регистры применяют следующие формулы:

Скважность (в %) = (PWMx_CONTROL)/(PWM_PERIOD + 1) × 100,

Ширина импульса (в мкс) = (Скважность × T_{PWM})/100,

где
PWMx_CONTROL = 8-битное значение для загрузки регистра PWMx_CONTROL,
PWM_PERIOD = 8-битное значение для загрузки регистра PWM_PERIOD,
ширина импульса = ширина каждого импульса высокого уровня,
T_{PWM} = период сигнала на выводе ШИМ, в мкс.

PWMx_CONTROL, (x = 0, 1) Адреса: 1FB0H, 1FB2H

Состояние Сброса: 00H

Регистр управления ШИМ (PWMx_CONTROL) определяет скважность сигнала ШИМ для канала x. Ноль, загруженный в этот регистр ШИМ, устанавливает состояние низкого уровня сигнала (постоянное, скважность равна 0 %). FFH в этом регистре устанавливает для ШИМ максимальную скважность (99,6 %) и состояние постоянного высокого уровня сигнала.

Таблица 9.5 – Регистр управления ШИМ (PWMx_CONTROL)

7	0
Скважность ШИМ	

Номер разряда	Функция
7:0	Скважность ШИМ. Этот регистр управляет скважностью ШИМ. Ноль, загруженный в этот регистр ШИМ, устанавливает состояние постоянного низкого уровня сигнала (скважность равна 0 %). FFH в этом регистре устанавливает для ШИМ максимальную скважность (99,6 %) и состояние постоянного высокого уровня сигнала.

Примеры вычислений

Например, предположим, что F_{BQ1} равняется 16 МГц, и значение, записанное в регистр PWM_PERIOD – FFH, таким образом, ожидаемый период сигнала на выходе ШИМ равен 8,19 мс. Если значение в PWMx_CONTROL равняется 8AH (десятичное 138), ширина высокого уровня сигнала равна 4,42 мс и низкого уровня сигнала 3,77 мс, что и составляет 8,19 мс для полного периода. Результирующая скважность равна 54 %.

Чтение текущего значения обратного счётчика

Можно считать значение регистра PWM_COUNT, чтобы найти текущее значение обратного счетчика (см. таблицу 9.6).

PWM_COUNT Адрес: 1FB6H

(только чтение) Состояние сброса: 00H

Регистр периода счета ШИМ (PWM_COUNT) обеспечивает текущее значение периода счетчика обратного счета.

Таблица 9.6 – Регистр периода счета ШИМ (PWM_COUNT)

7	0
Значение счета PWM	

Номер разряда	Функция
7:0	Значение периода счета PWM. Регистр содержит текущее значение периода счетчика обратного счета.

Разрешение ШИМ функции выводов

Каждый ШИМ выход мультиплексирован с выводом порта, так что необходимо конфигурировать его как сигнал вывода специальной функции перед использованием функции ШИМ. Чтобы определить, функционирует ли соответствующий вывод как стандартный вывод порта ввода-вывода или как ШИМ вывод, необходимо записать число в регистр WG_OUTPUT (см. таблицу 9.8). Таблица 9.7 показывает выбор альтернативной функции порта (ШИМ) записью регистра WG_OUTPUT.

Таблица 9.7 – Альтернативные функции ШИМ вывода

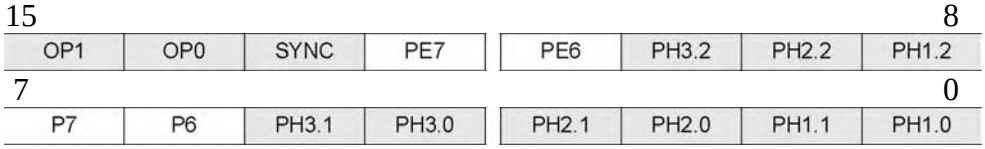
Выход ШИМ	Дополнительная функция порта	Выход ШИМ разрешен
PWM0	P6.6	когда WG_OUT.11 = 1
PWM1	P6.7	когда WG_OUT.12 = 1

WG_OUTPUT (Генератор формы сигнала) Адрес: 1FC0H

Состояние сброса: 0000H

Регистр управления функцией выхода генератора формы сигнала (WG_OUTPUT) управляет функционированием выходов генератора формы сигнала и блока ШИМ. И генератор формы сигнала, и блок ШИМ совмещают выводы с портом 6. Наличие битов управления в отдельном регистре позволяет формировать все выводы порта записью единственного регистра WG_OUTPUT.

Таблица 9.8 – Регистр генератора формы сигнала (WG_OUTPUT) – конфигурация выводов порта 6



Разрядный номер	Мнемоника	Функция
15	OP1	Полярность выхода
14	OP0	Полярность выхода
13	SYNC	Синхронизация
12	PE7	P6.7/PWM1 функция. Выбирает или функцию порта, или ШИМ функцию вывода P6.7/PWM1. 1 = PWM1 0 = P6.7
11	PE6	P6.6/PWM0 функция. Выбирает или функцию порта, или ШИМ функцию вывода P6.6/PWM0. 1 = PWM0 0 = P6.6
10	PH3.2	Функция фазы 3
9	PH2.2	Функция фазы 2
8	PH1.2	Функция фазы 1
7	P7	P6.7/PWM1 значение. В этот бит записывают желаемое значение P6.7/PWM1.
6	P6	P6.6/PWM0 значение. В этот бит записывают желаемое значение P6.6/PWM0.
5:4	PH3.1:0	P6.4/WG3#, P6.5/WG3 значения
3:2	PH2.1:0	P6.2/WG2 #, P6.3/WG2 значения
1:0	PH1.1:0	P6.0/WG1 #, P6.1/WG1 значения

Создание аналоговых выходов

Блоки ШИМ вырабатывают прямоугольные сигналы, которые имеют изменяемые период и скважность. Подключение фильтра к этим выводам позволит создать сглаженный аналоговый сигнал. Чтобы сформировать сигнал, изменяющийся с желаемыми аналоговыми параметрами, сначала надо поставить буфер для формирования сигнала и затем фильтр. Это или простая емкостно-резистивная цепь или

с активный фильтр. На рисунке 9.3 представлена типовая блок-схема устройства, необходимого для создания сглаженного аналогового сигнала.

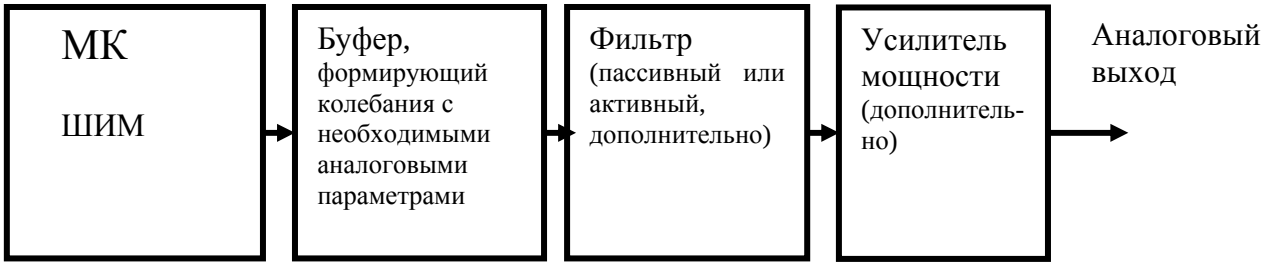


Рисунок 9.3 – Блок-схема преобразования цифрового выхода в аналоговый

Рисунок 9.4 показывает типовую схему, используемую для малых выходных токов (меньше, чем 100 мкА). Необходимо рассчитать температуру и дрейф источника питания при отборе компонентов для внешней цифро-аналоговой схемы. С надлежащими параметрами, используя ШИМ, можно сделать очень точные 8-битные цифро-аналоговые преобразователи.

Необходимые номиналы R и C рассчитываются для конкретного применения.

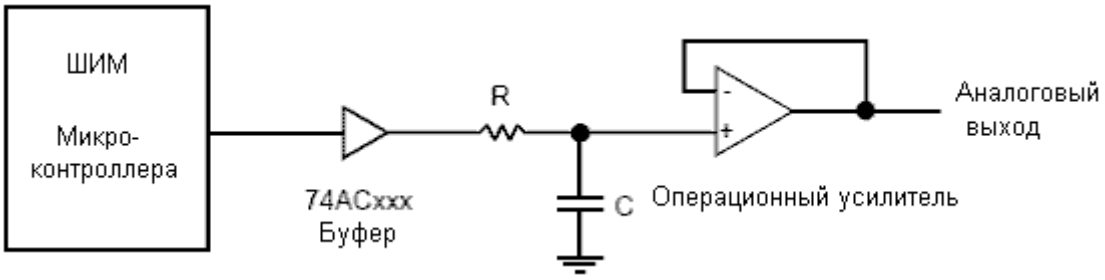


Рисунок 9.4 – Подключение ШИМ к аналоговой конверсионной схеме

10 Процессор событий (EPA)

Управление устройствами часто требует быстродействующего управления событиями. Например, может возникнуть необходимость, чтобы микроконтроллер периодически вырабатывал ШИМ сигналы или прерывание. В другом приложении микроконтроллер может контролировать входной сигнал для определения состояния внешнего устройства. Процессор событий (EPA) был спроектирован, чтобы уменьшить нагрузку на центральный процессор по управлению этими типами событий. Этот раздел описывает EPA, его таймеры и объясняет, как конфигурировать и программировать их.

10.1 Краткий обзор функционирования EPA

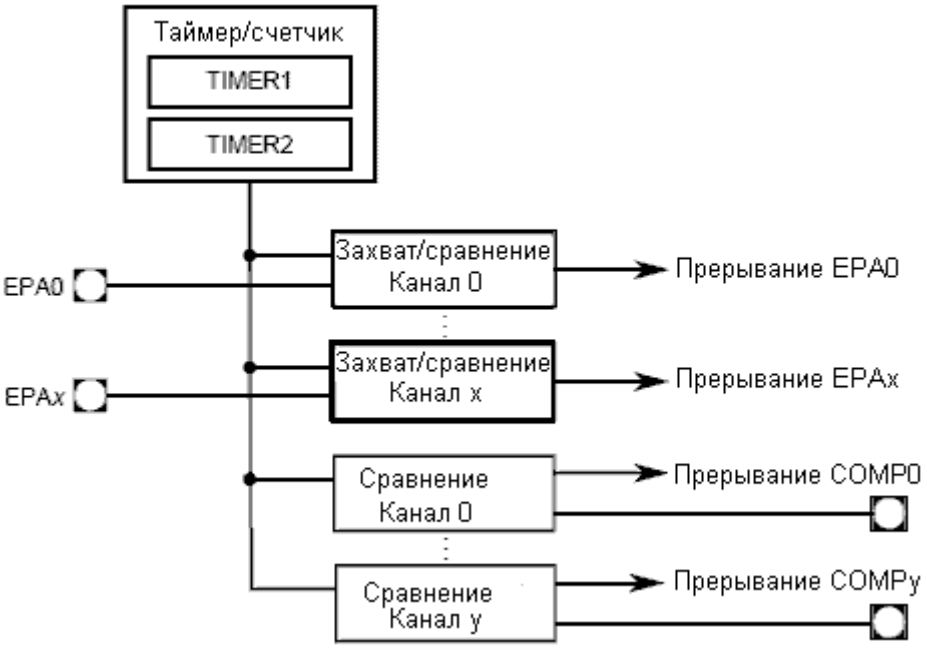
EPA осуществляет входные и выходные функции в соответствии с двумя таймерами/счетчиками – таймером 1 и таймером 2 (см. рисунок 10.1). Во входном режиме EPA фиксирует на выводе входные события: передний фронт, задний фронт или оба фронта входного импульса. Когда событие происходит, EPA делает запись значения таймера/счетчика в регистре и вырабатывает запрос на прерывание, таким образом, событие фиксируется по времени, когда оно произошло. Это называется «захватом по входу». Входные захваты буферизуются, чтобы позволить выполнить два захвата прежде, чем произойдет переполнение.

В выходном режиме EPA контролирует таймер/счетчик и сравнивает его значение со значением, сохраненным в специальном регистре. Когда значение таймера/счетчика соответствует сохраненному значению, EPA может вызвать событие: сброс таймера, преобразование АЦП, перезагрузку генератора формы сигнала или событие на выходе (установить выход в состояние высокого уровня, установить выход в состояние низкого уровня, изменить значение на выходе или не выполнять никакого действия). Такое выходное событие называют выходным сигналом по совпадению или «выходом сравнения».

Каждый захват по входу или выход сравнения устанавливает бит запроса прерывания. Этот бит может быть причиной прерывания. В таблице 10.1 приведен список каналов захвата/сравнения для микроконтроллеров.

Таблица 10.1 – Каналы EPA

Устройство	Каналы захвата/сравнения	Каналы только сравнения
1867ВЕ86Т, 1874ВЕ16Т	EPA3 - EPA0	COMP3 - COMP0



Примечание - x = y = 3.
Рисунок 10.1 – Блок-схема EPA

10.2 Регистры и сигналы таймера/счетчика EPA

В таблице 10.2 описаны сигналы EPA, входные и выходные сигналы таймера/счетчика. Каждый сигнал мультиплексирован с выводом порта, как показано в первой колонке. В таблице 10.3 кратко описаны регистры каналов захвата/сравнения EPA, каналов только сравнения EPA и таймеров/счетчиков.

Таблица 10.2 – Сигналы EPA и таймера/счетчика

Вывод порта	Сигнал EPA	Тип сигнала EPA	Описание
P1.2	T1CLK	Вход	Внешний источник синхроимпульсов для таймера 1.
P1.3	T1DIR	Вход	Внешнее управление направлением счета таймера 1.
P2.0 P2.1 P2.2 P2.3	EPA0 EPA1 EPA2 EPA3	Вход/выход	Быстродействующий ввод/вывод для каналов захвата/сравнения.
P2.4 P2.5 P2.6 P2.7	COMP0 COMP1 COMP2 COMP3	Выход	Выход каналов только сравнения.

Таблица 10.3 – Регистры управления и состояния EPA

Мнемоника	Адрес	Описание
1	2	3
COMP0_CON COMP1_CON COMP2_CON COMP3_CON	1F58H 1F5CH 1F60H 1F64H	Управление сравнения EPAx. Эти регистры управляют функциями каналов только сравнения.
COMP0_TIME COMP1_TIME COMP2_TIME COMP3_TIME	1F5AH 1F5EH 1F62H 1F66H	Время сравнения EPAx. Эти регистры содержат время, в которое произошло событие на каналах сравнения.
EPA0_CON EPA1_CON EPA2_CON EPA3_CON	1F40H 1F44H 1F48H 1F4CH	Управление захвата/сравнения EPAx. Эти регистры управляют функциями каналов захвата/сравнения. EPA1_CON и EPA3_CON требуют дополнительного байта, потому что они содержат дополнительный бит для режима генерации импульсов ШИМ (с помощью EPA). К этим двум регистрам нужно обращаться как к словам; к другим можно обратиться как к байтам.
EPA0_TIME EPA1_TIME EPA2_TIME EPA3_TIME	1F42H 1F46H 1F4AH 1F4EH	Время захвата/сравнения EPAx. В режиме захвата эти регистры содержат захваченное значение таймера. В режиме сравнения эти регистры содержат время, в течение которого должно произойти событие. В режиме захвата эти регистры буферизуются, чтобы позволить два захвата прежде, чем происходит переполнение. Однако они не буферизуются в режиме сравнения.
INT_MASK	0008H	Маска прерываний. Биты в этом 8-битном регистре разрешают и запрещают (маскируют) прерывания, связанные с соответствующими битами в регистре INT_PEND.
INT_MASK1	0013H	Маска прерываний 1. Биты в этом 8-битном регистре разрешают и запрещают (маскируют) прерывания, связанные с соответствующими битами в регистре INT_PEND1.
INT_PEND	0009H	Ожидание прерывания. Любой бит, установленный в этом 8-битном регистре указывает, что ожидается запрос прерывания.
INT_PEND1	0012H	Ожидание прерывания 1. Любой бит, установленный в этом 8-битном регистре указывает, что ожидается запрос прерывания.
P2_DIR	1FD2H	Порт_x Направление. Каждый бит Px_DIR управляет направлением соответствующего вывода. Очистка бита формирует вывод как комплементарный выход; установка бита формирует вывод как выход с открытым стоком или вход. (Выходы с открытым стоком требуют внешней поддержки высокого уровня.)

Продолжение таблицы 10.3

1	2	3
P2_MODE	1FD0H	Порт_x Режим. Каждый бит Px_MODE определяет, функционирует ли соответствующий вывод как стандартный вывод порта ввода-вывода или как сигнал специальной функции. Установка бита формирует вывод как сигнал специальной функции, очистка бита формирует вывод как стандартный выход порта ввода-вывода.
P0_PIN P1_PIN P2_PIN\	1FA8H 1FA9H 1FD6H	Порт_x Вход. Каждый бит Px_PIN отражает текущее состояние соответствующего вывода независимо от конфигурации вывода.
P2_REG	1FD4H	Порт_x Выход данных. Для входа надо установить соответствующий бит Px_REG. Для выхода – записать данные, которые будут выведены каждым выводом из соответствующего бита Px_REG. Когда вывод сконфигурирован как стандартный вход-выход (Px_MODE.y = 0), результат центрального процессора пишется в Px_REG и немедленно отображается на выводе. Когда вывод формируется как сигнал специальной функции (Px_MODE.y = 1), связанный периферийный блок МК или компонент вне МК управляет выводом. Центральный процессор может продолжать запись в Px_REG, но состояние вывода не изменяется, пока не сконфигурирована его стандартная функция входа-выхода. Эта особенность позволяет программному обеспечению формировать вывод как стандартный вход-выход (очистив Px_MODE.y), инициализировать или переустановить значение вывода, затем сконфигурировать вывод как сигнал специальной функции (установив Px_MODE.y). Этим способом инициализация, восстановление ошибки, обработка особых ситуаций и т.д. могут быть выполнены без изменения работы связанного периферийного устройства.
PI_MASK	1FBCH	Маска прерывания периферийного устройства. Биты в этом регистре разрешают и запрещают (маскируют) запрос прерывания по переполнению/антипереполнению таймеров 1 и 2, запросы прерывания генератора формы сигналов.
PI_PEND	1FBEH	Ожидание прерывания от периферийного устройства. Любой установленный бит указывает на ожидание запроса прерывания.
T1CONTROL	1F78H	Управление таймера 1. Этот регистр разрешает/запрещает таймер 1, управляет направлением счета, выбором источника синхросигнала и направления, устанавливает коэффициент деления частоты синхросигналов.
T1RELOAD	1F72H	Перезагрузка таймера 1. Этот регистр содержит значение инициализации для таймера 1. При переполнении/антипереполнении таймера 1 загружается значение T1RELOAD в регистр TIMER1, если и квадратурная синхронизация, и перезагрузка разрешены (T1CONTROL.5:0 = 1).

Продолжение таблицы 10.3

1	2	3
T2CONTROL	1F7CH	Управление таймера 2. Этот регистр разрешает/запрещает таймер 2 управляет направлением счета, выбирает источник синхросигнала и направления, устанавливает коэффициент деления частоты синхросигналов.
TIMER1	1F7AH	Значение таймера 1. Этот регистр содержит текущее значение таймера 1.
TIMER2	1F7EH	Значение таймера 2. Этот регистр содержит текущее значение таймера 2.

10.3 Краткий обзор функционирования таймеров/счетчиков

EPA имеет два 16-битных реверсивных (прямого и обратного счета) таймера/счетчика: таймер 1 и таймер 2, которые могут быть синхронизированы внутренним или внешним синхросигналом. Каждый из них называют таймером, если синхронизация внутренняя, и счетчиком, если синхронизация внешняя. Рисунок 10.2 иллюстрирует структуру таймера/счетчика.

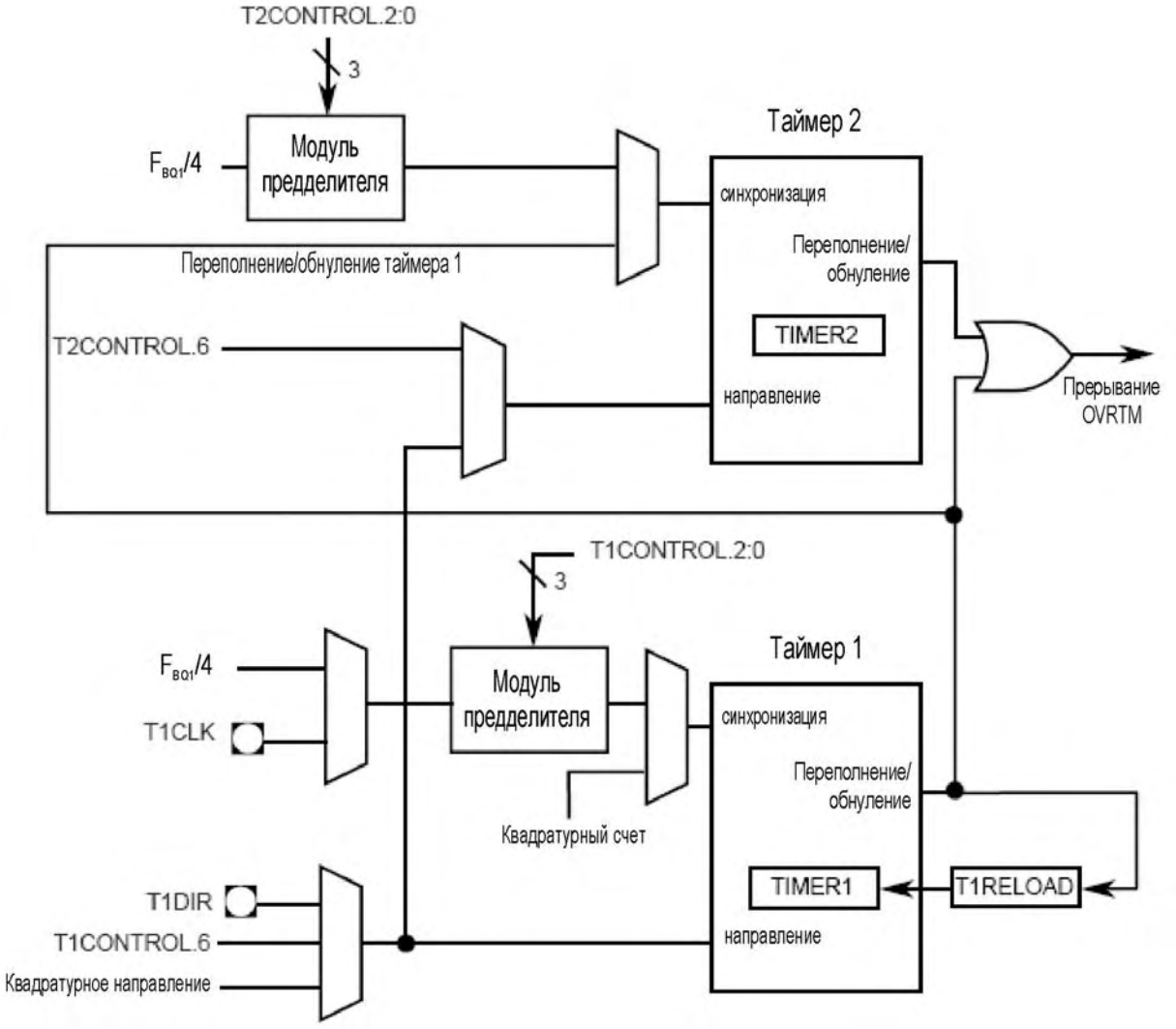


Рисунок 10.2 – Таймеры/счетчики EPA

Таймеры/счетчики могут использоваться как времязадающие блоки для входных захватов, выходных сравнений и программных прерываний (таймеры программного

обеспечения). Когда счетчик наращивается от FFFEH до FFFFH или уменьшается от 0001H до 0000H, устанавливается бит ожидания запроса прерывания по переполнению/антипереполнению. Этот бит может быть причиной прерывания. Источник синхроимпульсов, источник направления счета и разрешение захвата по входу или выхода сравнения программируются. Максимальная скорость счета – половина частоты внутреннего синхросигнала или $F_{BQ1}/4$. Это обеспечивает минимальное разрешение для входного захвата или выходного сравнения 250 нс (при 16 МГц).

Разрешение = $(4\times prescaler\_divisor)/F_{BQ1,}$

где prescaler_divisor – это коэффициент деления частоты из регистров TxCONTROL;

F_{BQ1} - частота входного синхросигнала на выводе BQ1.

Каскадный режим (только таймер 2)

Таймер 2 может использоваться в каскадном режиме. В этом режиме сигнал переполнения таймера 1 используется как сигнал синхронизации таймера 2. При этом направление счета определяется управляющим битом регистра управления таймера 1 или таймера 2. Этот метод, называемый каскадным, может обеспечить синхроимпульсы низкой частоты для управления таймаутом режима холостого хода или для формирования низкочастотных ШИМ сигналов.

Режим квадратурного счета

Таймер 1 может использоваться в двух режимах синхронизации квадратурного счета. Оба режима используют выводы T1CLK и T1DIR как квадратурные входы, как показано на рисунке 10.3. Внешние квадратурно кодированные сигналы (два сигнала одной частоты, которые отличаются по фазе на 90 °) определяют инкремент или декремент таймера при поступлении положительного или отрицательного фронта. Поскольку входы T1CLK и T1DIR тактируются внутренними синхросигналами, переключения должны быть разделены временем, равным не менее двух периодов системного тактового сигнала для правильного выполнения операций. Счет синхронизирован сигналом PH2, который является сигналом PH1, задержанным на половину периода. Последовательность фронтов сигнала и уровней управляет направлением счета, см. рисунок 10.4 и таблицу 10.4 для информации о направлении счета.

Типичный источник сигналов квадратурного счета – датчик угловых перемещений, показанный на рисунке 10.3. Его выходные сигналы X и Y – входы для T1CLK и T1DIR, которые в свою очередь формируют выходные сигналы X_internal и Y_internal. Эти сигналы используются на рисунке 10.4 и в таблице 10.4, чтобы описать направление поворота диска.

В заданном по умолчанию режиме квадратурного счета программное обеспечение должно перезагрузить регистр TIMER1, когда таймер 1 переполняется или антипереполняется. В режиме 2 (T1CONTROL.2:0=1) таймер 1 автоматически загружает значение из регистра T1RELOAD в регистр TIMER1, когда происходит переполнение или антипереполнение. Режим 2 полезен для интерфейса с инкрементным датчиком поворота, при вращении только в одном направлении. Для этого случая инициализируется T1RELOAD со значением, которое на единицу меньше, чем разрешение кодирующего устройства. Этот метод позволяет таймеру 1 отслеживать абсолютное положение устройства кодирования поворота без программных издержек.

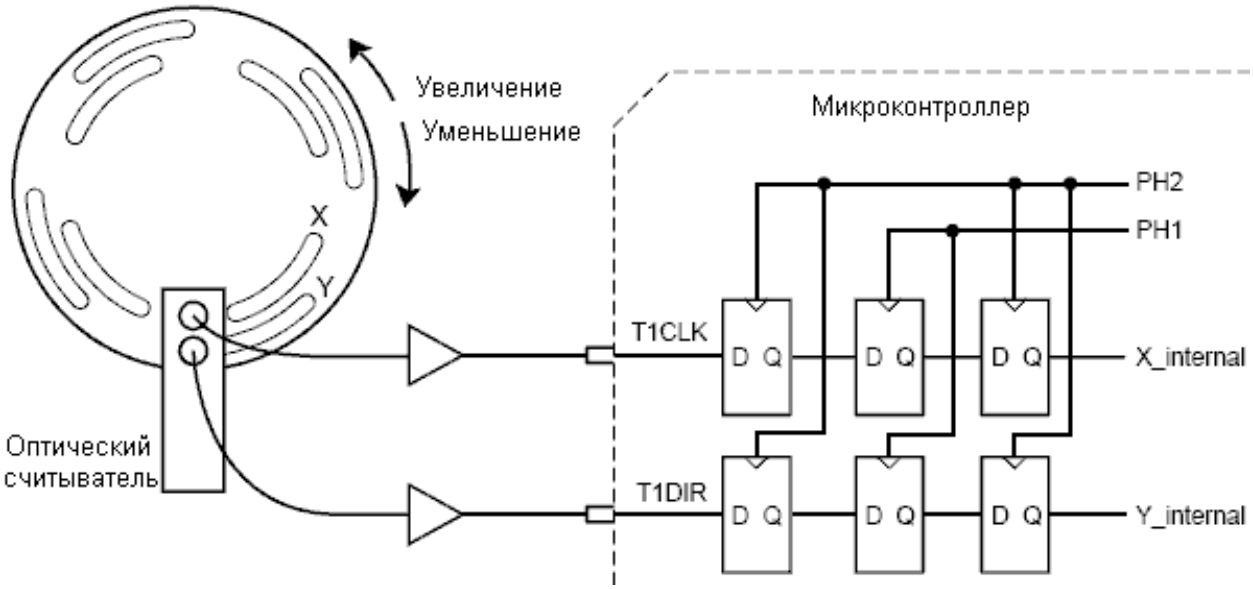


Рисунок 10.3 – Интерфейс режима квадратурного счета

Таблица 10.4 – Таблица направления счета в зависимости от T1CLK и T1DIR

Состояние X_internal (T1CLK)	Состояние Y_internal (T1DIR)	Направление счета
↑	0	Приращение
↓	1	Приращение
0	↓	Приращение
1	↑	Приращение
↓	0	Уменьшение
↑	1	Уменьшение
0	↑	Уменьшение
1	↓	Уменьшение

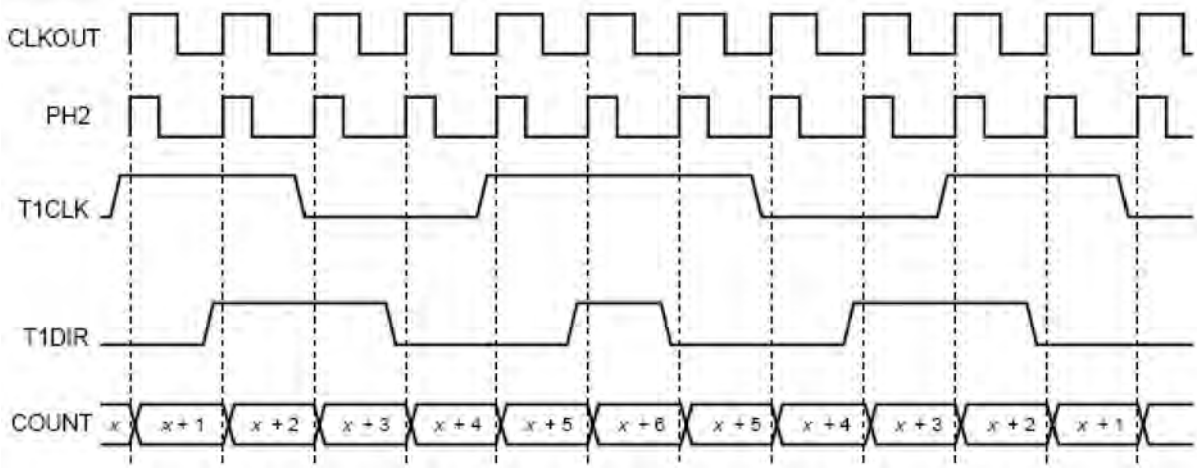


Рисунок 10.4 – Временные диаграммы для квадратурного счета

10.4 Краткое описание функционирования канала EPA

EPA имеет и программируемые каналы захвата/сравнения, и только сравнения. Каждый канал захвата/сравнения может выполнять следующие задачи (каналы только

сравнения имеют те же самые функциональные возможности за исключением того, что они не могут захватить внешнее событие):

- захват значения таймера, когда указанное переключение происходит на выводе EPA;
- старт АЦП преобразования или перезагрузка генератора формы сигнала, когда событие захвачено, или значение таймера соответствует запрограммированному значению в регистре времени события;
- очистка, установка или переключение на выводе EPA, когда значение таймера соответствует запрограммированному значению в регистре времени события;
- генерация прерывания, когда происходит захват или сравнение;
- сброс собственного базового таймера в режиме сравнения;
- сброс другого таймера и в режиме сравнения, и в режиме захвата.

Каждый канал EPA имеет регистр управления – EPAx_CON (каналы захвата/сравнения) или COMPx_CON (только каналы сравнения), регистр времени события, EPAx_TIME (каналы захвата/сравнения) или COMPx_TIME (только каналы сравнения) и вход таймера (рисунок 10.5). Регистр управления выбирает таймер, режим и событие, которое будет захвачено или событие, которое должно произойти. Регистр времени события удерживает захваченное значение таймера в режиме захвата и время события в режиме сравнения.

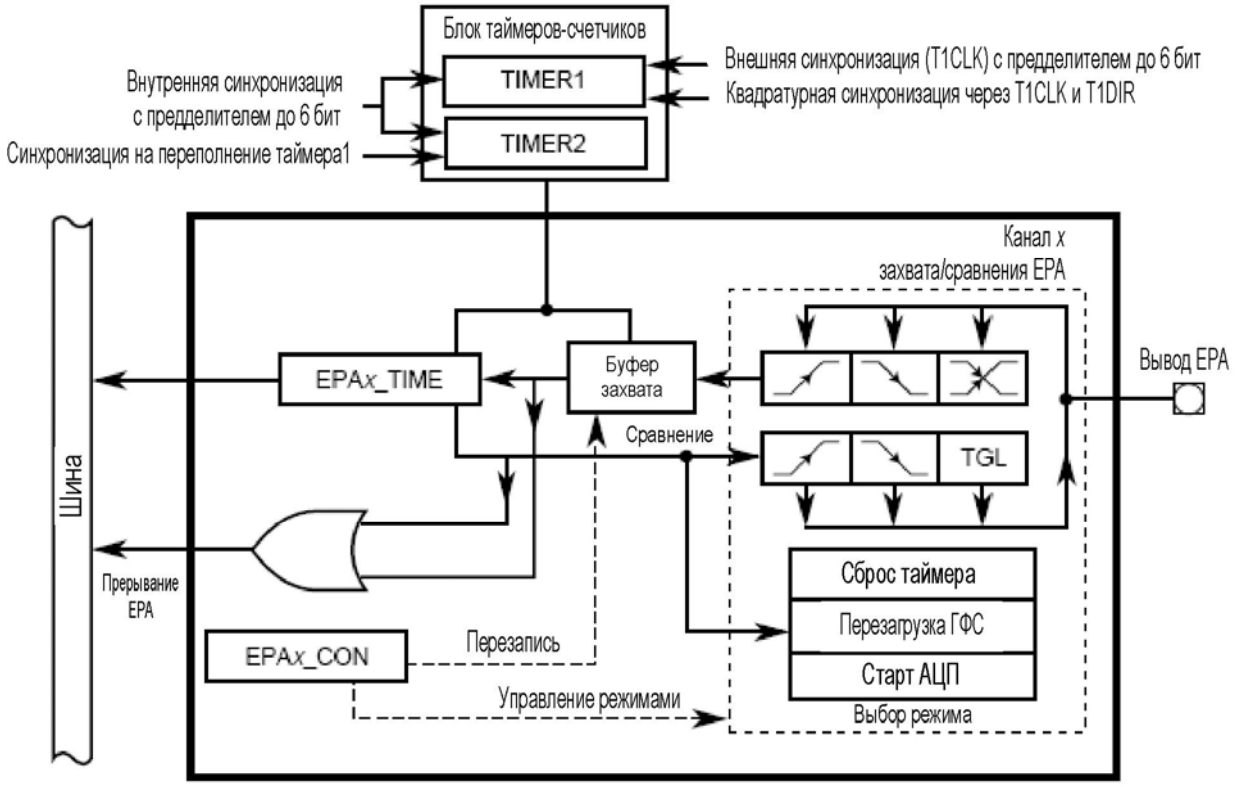


Рисунок 10.5 – Канал захвата/сравнения EPA

Работа в режиме захвата

В режиме захвата, когда действительное событие на выводе имеет место, значение выбранного таймера захватывается в буфер. Значение таймера затем передается из буфера в регистр EPAx_TIME, который устанавливает бит ожидания прерывания EPA, как показано на рисунке 10.6. При разрешении происходит прерывание. Если второе событие происходит прежде, чем центральный процессор читает первое значение таймера в EPAx_TIME, текущее значение таймера загружается в буфер и удерживается там. После того, как центральный процессор читает регистр

EPAx_TIME, содержимое буфера захвата автоматически перемещается в EPAx_TIME, и бит ожидания прерывания EPA устанавливается.

Рисунок 10.6 – Упрощенная структура EPA при захвате

Если третье событие происходит прежде, чем центральный процессор считает регистр времени события, бит перезаписи (EPAx_CON.0) определяет, как EPA будет обрабатывать событие. Если бит очищен, EPA игнорирует третье событие. Если бит установлен, время третьего события перезаписывает в буфере захвата время второго события. Таблица 10.5 суммирует возможные варианты обработки действительного события.

Примечание – Для того, чтобы событие было захвачено, сигнал должен быть установлен в течение, по крайней мере, двух машинных циклов до и после переключения (см. рисунок 10.7).

Рисунок 10.7 – Действующие события на выводе EPA

Таблица 10.5 – Операции канала EPA, когда действующий фронт сигнала появляется на выводе

Бит перезаписи (EPAx_CON.0)	Состояние буфера захвата и EPAx_TIME	Операция EPA, когда на выводе действующий фронт сигнала
0	Пустой	Фронт захвачен, и время события загружено в буфер захвата и регистр EPAx_TIME.
0	Полный	Новые данные игнорируются, нет захвата, прерывания EPA или передачи времени события.
1	Пустой	Фронт захвачен, и время события загружено в буфер захвата и регистр EPAx_TIME.
1	Полный	Старые данные переписаны в буфер захвата.

Захват входного события не устанавливает бит ожидаемого прерывания, пока захваченное значение времени не перемещается фактически из буфера захвата в регистр EPAx_TIME. Если буфер содержит данные, и используется PTS для обслуживания прерывания, то два прерывания PTS происходят почти одно за другим (то есть с одной инструкцией, выполненной между прерываниями).

Переполнение EPA

Переполнение имеет место, когда частота переключения на входах EPA превышает скорость обработки процедурой обслуживания прерывания EPA. Если не осуществлять стратегию обработки переполнений, и если одновременно выполняются указанные ниже три условия, может возникнуть ситуация, когда и буфер захвата, и регистр EPAx_TIME содержат данные, а EPA прерывание не происходит:

- входной сигнал с достаточно высокой частотой, чтобы вызвать переполнение, присутствует на выбранном выводе EPA,
- бит перезаписи установлен (EPAx_CON.0 = 1, старые данные переписаны при переполнении),
- регистр EPAx_TIME читается в тот момент, когда EPA признает захваченный фронт как действующий.

Входная частота события зависит от длительности процедуры обслуживания прерывания так же, как других факторов. Если процедура обслуживания прерывания не включает проверку на переполнение, эта ситуация сохраняется до тех пор, пока МК не будет сброшен или регистр EPAx_TIME сосчитан. Выполнение чтения EPAx_TIME позволяет буферизованному значению времени быть перемещенным в EPAx_TIME. Это очищает буфер и позволяет другому событию быть захваченным. Необходимо помнить, что процедура передачи содержания буфера регистру EPAx_TIME состоит в том, что фактически устанавливается бит ожидания прерывания EPAx, и начинается прерывание.

Предотвращение переполнения EPA

Любой из следующих методов может использоваться, чтобы предотвратить или исправить ситуацию переполнения EPA.

- Очистить EPAx_CON.0. Когда бит перезаписи (EPAx_CON.0) - ноль, EPA не оценивает захваченный фронт, пока регистр EPAx_TIME не считан и данные не переданы из буфера захвата в EPAx_TIME. Это предотвращает переполнение, игнорируя новые входные события для захвата, когда буфер захвата и EPAx_TIME содержат действительные времена захвата.
- Сделать проверку ожидаемых запросов прерываний до выполнения процедуры обслуживания прерываний EPAx. Другой способ избежать переполнения EPA состоит в проверке ожидаемых запросов прерываний перед выходом из процедуры обслуживания прерываний EPAx. Это – легкий способ обнаружения переполнения и дополнительных прерываний. Это может также сохранить время цикла, устраняя время ожидания, необходимое для обслуживания ожидаемых прерываний. Однако этот метод не может использоваться с периферийным сервером (PTS).

Работа в режиме сравнения

Когда отобранное значение таймера соответствует значению времени события, происходит действие, указанное в регистре управления (то есть установка, очистка или переключение выхода, начало АЦП преобразования или перезагрузка генератора формы сигнала). Если бит повторного разрешения (EPAx_CON.3) установлен, действие происходит повторно при каждом совпадении значения таймера. Если очищен бит повторного разрешения, действие не происходит повторно, пока новое значение не будет записано в регистр времени события. В режиме сравнения можно использовать

EPA, чтобы формировать широтно-импульсномодулированный (ШИМ) сигнал на выводе EPA.

Формирование низкочастотного сигнала ШИМ

Можно сформировать низкочастотный ШИМ сигнал на выводе одного канала EPA со стандартной процедурой обслуживания прерывания. Формируется канал EPA следующим образом: режим сравнения, переключение на выводе и функция повторного сравнения разрешены. Выбирается процедура обслуживания стандартных прерываний, разрешается прерывание EPA и глобальное разрешение прерываний инструкцией EI. Когда значение таймера/счетчика соответствует значению в регистре времени события, EPA переключает сигнал на выводе и вызывает прерывание. Процедура обслуживания прерывания загружает новое значение в EPAx_TIME.

Максимальная выходная частота зависит от полной задержки прерываний и от времени выполнения обслуживания прерывания процедурой, используемой системой. Поскольку дополнительные каналы EPA и другие функции микроконтроллера используются, получается уменьшение максимальной частоты ШИМ, потому что общее время ожидания прерываний и обслуживания прерываний увеличивается. Для определения максимума низкочастотного ШИМ сигнала в системе вычисляют худший случай времени задержки прерываний и наихудшее время выполнения обслуживания прерывания и затем складывают их вместе. Наихудший случай времени ожидания прерываний – полное время ожидания всех прерываний (обычных и PTS), используемых в системе. Наихудший случай времени выполнения обслуживания прерываний – полное время выполнения всех процедур обслуживания прерываний и процедур PTS.

Предположим, что система имеет один канал EPA, одно разрешенное прерывание и следующую процедуру обслуживания прерывания.

```
; если EPA0-x  прерывание произошло
EPA0-x_ISR:
    PUSHA
    LD EPAx_CON, #toggle_command
    ADD EPAx_TIME, TIMERx, [next_duty_ptr]; загрузка  времени следующего
                                          ; события
    POPA
    RET
```

Худший случай времени ожидания прерывания для системы с единственным прерыванием – 56 периодов системного тактового сигнала (тактов) при использовании внешнего стека и 54 такта при использовании внутреннего стека. Для определения времени обслуживания прерываний складывают время выполнения инструкций сервисной программы.

Полное время выполнения процедуры обслуживания прерывания, которая обслуживает прерывание EPA – 79 тактов при использовании внешнего стека или 71 такт при использовании внутреннего стека. Поэтому один канал захвата/сравнения может быть обновлен каждые 125 тактов при использовании внутреннего стека (54 + 71). Каждый ШИМ период требует два обновления (одна установка и одна очистка), так что период ШИМ равняется 250 тактам. Когда входная частота на BQ1 – 16 МГц, ШИМ период – 31,25 мкс и максимальная частота ШИМ –32 кГц.

Формирование высокоскоростного ШИМ сигнала

Можно сформировать высокоскоростной импульсный сигнал с модулированной шириной импульса из двух каналов EPA и выделенного таймера/счетчика. Первый канал переключает сигнал на выходе, когда значение таймера соответствует EPAx_TIME, и через некоторое время второй канал переключает сигнал на выходе снова, а затем сбрасывает таймер/счетчик. Это перезапускает цикл. Прерывания не запрашиваются, в

результате возможна максимальная скорость. Программное обеспечение вычисляет соответствующие значения EPAx_TIME и загружает их в соответствующее время в цикле в соответствии с изменением частоты или скважности.

Этот метод настройки EPA (устанавливаемый регистрами TxCONTROL) позволяет получить максимум частоты ШИМ сигнала на выводе. (Разрешение – минимальное время, требуемое между последовательными захватами или сравнениями.) Когда входная частота на BQ1 – 16 МГц, разрешение 250 нс при максимальной частоте сигнала ШИМ 4 МГц.

10.5 Программирование EPA и таймеров/счетчиков

В этом подразделе обсуждается конфигурирование выводов порта для EPA и таймеров/счетчиков, описывается, как программировать таймеры, каналы захвата/сравнения и каналы только сравнения и объясняется, как разрешить прерывания EPA.

Конфигурирование EPA и сигналов таймера/счетчика

Прежде чем использовать EPA, необходимо сформировать соответствующие сигналы порта как сигналы специальных функций для EPA, так и, дополнительно, источник синхронизации и сигналы управления направлением счета для таймера/счетчика.

Сигналы, которые не используются для канала EPA или таймера/счетчика, могут формироваться как стандартный вход-выход.

Программирование таймеров

Регистры управления таймерами – T1CONTROL (рисунок 10.8) и T2CONTROL (рисунок 10.9). Запись значений в эти регистры конфигурирует таймеры. Запись в регистры TIMER1 и TIMER2 (адреса – см. таблицу 10.3) задает определенные значения в регистрах таймеров.

T1CONTROL Адрес: 1F78H

Состояние после сброса: 00H

Регистр управления таймера 1 (T1CONTROL) определяет источник синхроимпульсов, направление счета и период счета для таймера 1.

7							0
CE	UD	M2	M1	M0	P2	P1	P0

Номер бита	Мнемоника	Функция
7	CE	Разрешение счета. Этот бит разрешает или запрещает таймеры. После сброса таймеры запрещены и не разрешен их запуск. 0 = запрещает таймеры 1 = разрешает таймеры
6	UD	Прямой/обратный счет. Этот бит определяет направление счета таймера в выбранных режимах (см. биты режима, M2:0). 0 = счет в обратном направлении 1 = счет в прямом направлении
5:3	M2:0	Биты управления синхронизацией и направлением счета EPA. Эти биты определяют источник синхроимпульсов и источник сигнала, управляющего направлением счета. M2 M1 M0 Источник синхроимпульсов Источник направления 0 0 0 $F_{BQ1}/4$ UD бит (T1CONTROL.6)

		X 0 1	T1CLK вход[1)]	UD бит (T1CONTROL.6)
		0 1 0	$F_{BQ1}/4$	T1DIR вход
		0 1 1	T1CLK вход[1)]	T1DIR вход
		1 1 1	квадратурная синхронизация, использующая T1CLK и T1DIR	
2:0	P2:0	Биты коэффициента деления частоты синхроимпульсов EPA (Prescaler) Эти биты определяют prescaler_divisor для синхроимпульсов.		

P2	P1	P0	Деление	Разрешение [2)]
0	0	0	делятся на 1 (запрещение)	250 нс
0	0	1	делятся на 2	500 нс
0	1	0	делятся на 4	1 мкс
0	1	1	делятся на 8	2 мкс
1	0	0	делятся на 16	4 мкс
1	0	1	делятся на 32	8 мкс
1	1	0	делятся на 64	16 мкс
1	1	1	разрешение T1RELOAD	—

[1)] Если выбрана внешняя синхронизация, таймер считает и на положительный и отрицательный фронт синхроимпульсов.

[2)] Для 16 МГц. Для других частот используют формулу разрешение = (4×prescaler_divisor)/F_{BQ1}.

Рисунок 10.8 – Регистр управления таймера 1 (T1CONTROL)

T2CONTROL Адрес: 1F7CH
Состояние после сброса: 00H

Регистр управления таймера 2 (T2CONTROL) определяет источник синхроимпульсов, направление и диапазон счёта для таймера 2.

7							0
CE	UD	M2	M1	M0	P2	P1	P0

Номер бита	Мнемоника	Функция
7	CE	Разрешение счета. Этот бит разрешает или запрещает таймеры. После сброса таймеры запрещены и не разрешен запуск. 0 = запрещает таймер 1 = разрешает таймер
6	UD	Прямой/обратный счет. Этот бит определяет направление счета таймера в выбранных режимах (см. биты режима M2:0). 0 = счет в обратном направлении 1 = счет в прямом направлении
5:3	M2:0	Биты управления синхронизацией и направлением счета EPA. Эти биты определяют источник синхроимпульсов и источник направления счета.

M2	M1	M0	Источник синхроимпульсов	Источник направления
0	0	0	$F_{BQ1}/4$	UD бит (T2CONTROL.6)

		X	0	1	зарезервирован —
		0	1	0	зарезервирован —
		0	1	1	зарезервирован —
		1	0	0	переполнение таймера 1 UD бит (T2CONTROL.6)
		1	1	0	переполнение таймера 1 как у таймера 1
		1	1	1	зарезервирован —

2:0 P2:0 Биты коэффициента деления частоты синхроимпульсов EPA (Prescaler_divisor).

Эти биты определяют prescaler_divisor для синхроимпульсов.

P2	P1	P0	Деление	Разрешение [1)]
0	0	0	деление на 1 (запрещение)	250 нс
0	0	1	деление на 2	500 нс
0	1	0	деление на 4	1 мкс
0	1	1	деление на 8	2 мкс
1	0	0	деление на 16	4 мкс
1	0	1	деление на 32	8 мкс
1	1	0	деление на 64	16 мкс
1	1	1	зарезервировано	—

[1)] Для 16 МГц. Для других частот используют формулу: Разрешение = (4×prescaler_divisor)/F_{BQ1} .

Рисунок 10.9 – Регистр управления таймера 2 (T2CONTROL)

Программирование каналов захвата/сравнения

Регистр EPAx_CON управляет функционированием соответствующего канала захвата/сравнения. Регистры идентичны за исключением бита 2. Для каналов EPA 0, 2 установка этого бита позволяет EPA по событию вызывать перезагрузку генератора формы сигнала. Для каналов EPA 1, 3 установка этого бита позволяет событию EPA вызывать АЦП преобразование. Чтобы программировать событие сравнения, всегда записывают EPAx_CON (рисунок 10.10) первым для конфигурации EPA канала захвата/сравнения, и затем загружают время события в EPAx_TIME. Чтобы запрограммировать захват события, достаточно записать EPAx_CON. Таблица 10.6 показывает действие от различных комбинаций значений битов в EPAx_CON для каналов 1, 3.

Таблица 10.6 – Пример установок битов в управляющем регистре EPA

Режим захвата								
TB	CE	MODE		PE	WGR/ AD	ROT	ON/ RT	Действие
7	6	5	4	3	2	1	0	
X	0	0	0	—	–	—	0	Нет
X	0	0	1	—	X	X	X	Захват на отрицательном фронте
X	0	1	0	—	X	X	X	Захват на положительном фронте
X	0	1	1	—	X	X	X	Захват на обоих фронтах
X	0	X	1	—	X	1	X	Захват на отрицательном фронте и сброс противоположного таймера
X	0	1	X	—	X	1	X	Захват на положительном фронте и сброс противоположного таймера
X	0	0	1	—	1	X	X	Старт АЦП преобразования (EPA1, 3) или перезагрузка генератора формы сигнала (EPA0, 2) на отрицательном фронте
X	0	1	0	—	1	X	X	Старт АЦП преобразования (EPA1, 3) или перезагрузка генератора формы сигнала (EPA0, 2) на положительном фронте
Режим сравнения								
TB	CE	MODE		PE	WGR/ AD	ROT	ON/ RT	Действие
7	6	5	4	3	2	1	0	
X	1	0	0	X	—	—	0	Нет
X	1	0	0	X	0	X	0	Выработка только прерывания (таймер программного обеспечения)
X	1	0	1	X	X	X	X	Обнуление на выходе
X	1	1	0	X	X	X	X	Установка (высокий уровень) на выходе
X	1	1	1	X	X	X	X	Переключение на выходе
X	1	X	X	X	X	0	1	Сброс текущего таймера
X	1	X	X	X	X	1	1	Сброс противоположного таймера
X	1	X	X	X	1	X	X	Старт АЦП преобразования (EPA1, EPA3) или перезагрузка генератора формы сигнала (EPA0, EPA2)
1 - не используемый бит. 2 - X = бит может использоваться, но нет никакого действия.								

EPAx_CON — Адрес: см. таблицу 10.3

x = 0–3 — Состояние сброса: 00H

Регистры управление EPA (EPAx_CON) управляют функциями соответствующих каналов захвата/сравнения.

x = 0, 2

7							0
TB	CE	M1	M0	RE	WGR	ROT	ON/RT

x = 1, 3

7							0
TB	CE	M1	M0	RE	AD	ROT	ON/RT

Номер бита	Мнемоника	Функция
7	TB	Выбор временной базы. Определяет текущий таймер. 0 = таймер 1 – текущий таймер, а таймер 2 – противоположный таймер 1 = таймер 2 – текущий таймер, а таймер 1 – противоположный таймер Событие сравнения (перезагрузка генератора формы сигнала, старт АЦП преобразования, очистка, установка или переключение выхода, и/или сброс любого таймера) происходит, когда значение в текущем таймере соответствует времени, запрограммированному в регистре времени события. Когда происходит событие захвата (отрицательный фронт, положительный фронт или появление любого фронта на выводе EPAx), значение текущего таймера сохраняется в регистре времени события EPA (EPAx_TIME).
6	CE	Разрешение сравнения. Определяет, работает канал EPA в режиме захвата или сравнения. 0 = режим захвата 1 = режим сравнения
5:4	M1:0	Выбор режима EPA. В режиме захвата определяет тип события, которое вызывает входной захват. В режиме сравнения определяет действие, которое EPA выполняет на выводе, когда текущий таймер соответствует времени события. M1 M0 Событие режима захвата 0 0 никаких захватов 0 1 захват на отрицательном фронте 1 0 захват на положительном фронте 1 1 захват на любом фронте M1 M0 Действие в режиме сравнения 0 0 нет изменений 0 1 переключение в низкий уровень 1 0 переключение в высокий уровень 1 1 переключение уровня
3	RE	Повторное разрешение. Повторное разрешение применимо только к режиму сравнения. Это позволяет продолжать выполнять сравнение события каждый раз, когда регистр (EPAx_TIME) соответствует текущему таймеру, а не только до первого сравнения. 0 = сравнение, функция запрещена после первого события 1 = сравнение, функция всегда разрешена
2	WGR /AD	Перезагрузка генератора формы сигнала, АЦП преобразование. Функция этого бита зависит от канала EPA. Для канала захвата/сравнения EPA0, 2: WGR бит позволяет использовать действия EPA, чтобы вызвать перезагрузку новых значений в генератор формы сигнала. 0 = нет действия 1 = разрешена перезагрузка генератора формы сигнала Для каналов захвата/сравнения EPA1, 3:

		Бит разрешения АЦП преобразования позволяет использовать действия EPA, чтобы начать АЦП преобразование, которое было предварительно подготовлено в регистрах управления АЦП. 0 = не начинать АЦП преобразование 1 = старт АЦП преобразования при сравнении
1	ROT	Сброс противоположного таймера. Управляет различными функциями для режимов захвата и сравнения. В режиме захвата: 0 = не вызывает никакого действия 1 = сброс противоположного таймера В режиме сравнения: Выбирает таймер, который должен быть сброшен, если бит установлен. 0 = выбирает текущий таймер для возможного сброса 1 = выбирает противоположный таймер для возможного сброса ТВ бит (бит 7) выбирает, какой таймер является текущим таймером и какой является противоположным таймером.
0	ON/RT	Перезапись/сброс таймера. Этот бит перезаписывает новое значение в режиме захвата и сбрасывает таймер в режиме сравнения. В режиме захвата (ON): ошибка переполнения происходит во время входного захвата, в то время как регистр (EPAx_TIME) и его буфер переполнены. Когда происходит переполнение, бит ON определяет будут ли перезаписаны старые данные или новые данные игнорируются: 0 = новые данные игнорируются 1 = старые данные в буфере перезаписываются В режиме сравнения (RT): 0 = запрещает функцию сброса 1 = сброс таймера, выбираемого битом ROT

Рисунок 10.10 – Регистр управления EPA (EPAx_CON)

Программирование каналов сравнения

Чтобы программировать события сравнения, необходимо сначала записать в регистр COMPx_CON (рисунок 10.11), чтобы формировать канал только сравнения и затем загрузить время события в COMPx_TIME. COMPx_CON имеет те же самые биты и назначения как EPAx_CON. COMPx_TIME функционально идентичен EPAx_TIME.

COMPx_CON Адрес: см. таблицу 10.3

x = 0–3 Состояние после сброса: 00H

Регистры управления EPA сравнением (COMPx_CON) определяют функции сравнения каналов EPA.

x = 0, 2

7							0
TB	CE	M1	M0	RE	WGR	ROT	RT

x = 1, 3

7							0
TB	CE	M1	M0	RE	AD	ROT	RT

7	TB	Выбор временной базы. Определяет текущий таймер. 0 = таймер 1 – текущий таймер, а таймер 2 - противоположный таймер 1 = таймер 2 – текущий таймер, а таймер 1 - противоположный таймер Событие сравнения (старт АЦП преобразования, очистка, установка или переключение выхода, и/или перезагрузка любого таймера) происходит, когда значение в текущем таймере соответствует времени, запрограммированному в регистре времени события.
6	CE	Разрешение сравнения. Разрешает работать каналу EPA в режиме сравнения. 0 = режим сравнения запрещен 1 = режим сравнения разрешен
5:4	M1:0	Выбор режима EPA. Определяет тип сравниваемого события. M1 M0 Действие в режиме сравнения 0 0 нет изменений 0 1 переключение в низкий уровень 1 0 переключение в высокий уровень 1 1 переключение уровня
3	RE	Повторное разрешение. Повторное разрешение применимо только к режиму сравнения. Это позволяет продолжать выполнять сравнение события каждый раз, когда регистр (EPAx_TIME) соответствует текущему таймеру, а не только до первого сравнения. 0 = сравнение, функция запрещена после первого события 1 = сравнение, функция всегда разрешена
2	WGR /AD	Перезагрузка генератора формы сигнала, АЦП преобразование. Функция этого бита зависит от канала EPA. Для канала захвата/сравнения EPA0, EPA2: WGR бит позволяет использовать действия EPA, чтобы вызвать перезагрузку новых значений в генератор формы сигнала. 0 = нет действия 1 = разрешена перезагрузка генератора формы сигнала Для каналов захвата/сравнения EPA1, EPA3: бит разрешения АЦП преобразования позволяет использовать действия EPA, чтобы начать АЦП преобразование, которое было предварительно подготовлено в регистрах управления АЦП. 0 = не начинать АЦП преобразование 1 = старт АЦП преобразования при сравнении
1	ROT	Сброс противоположного таймера. Выбирает таймер, который должен быть сброшен, если RT бит установлен. 0 = выбирает текущий таймер для возможного сброса 1 = выбирает противоположный таймер для возможного сброса. Состояние TB бита определяет, какой таймер является текущим таймером и какой таймер является противоположным таймером.
0	RT	Сброс таймера. Этот бит управляет, будет ли таймер, выбранный битом ROT, сброшен. 1 = сбрасывает таймер, выбранный битом ROT 0 = запрещает функцию сброса

Рисунок 10.11 – Регистр EPA управления сравнением (COMPx_CON)

10.6 Разрешения прерывания EPA

Для разрешения прерывания необходимо установить соответствующие биты в регистре INT_MASK. Для разрешения индивидуальных источников периферийных прерываний устанавливать соответствующие биты в регистре PI_MASK. Раздел 6 подробнее описывает эти прерывания.

10.7 Определение состояния события

В режиме сравнения бит ожидания прерывания устанавливается каждый раз, когда происходит совпадение при разрешенном событии (даже если прерывание специально замаскировано в регистре маски). В режиме захвата бит запроса прерывания устанавливается каждый раз, когда запрограммированное событие захвачено, и время события перемещается из буфера захвата в регистр EPAx_TIME.

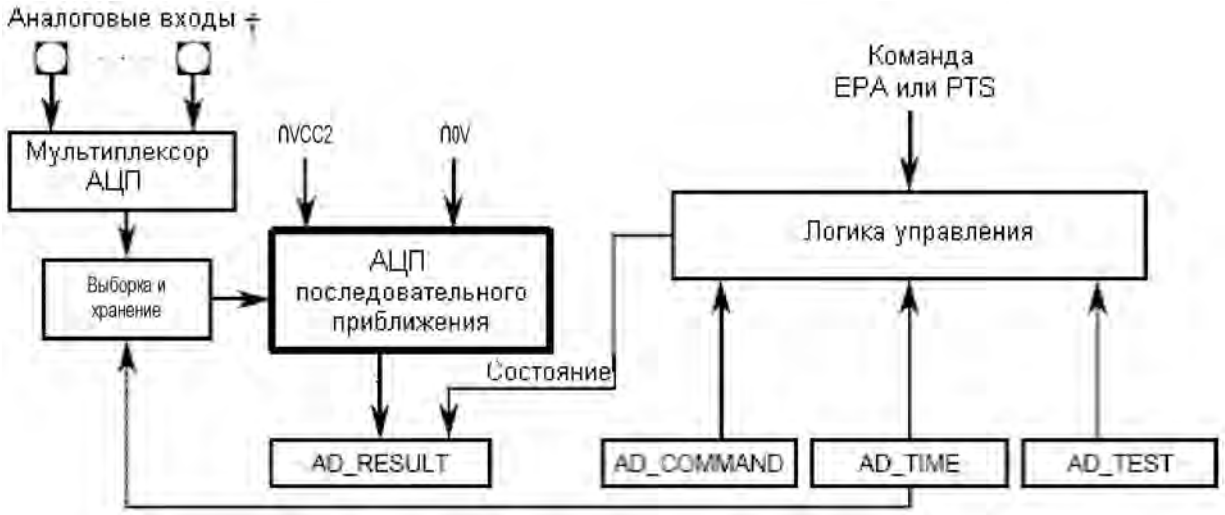
Биты ожидания расположены в регистрах INT_PEND и INT_PEND1. Биты запроса для мультиплексированных прерываний (те, которые входят в периферийные прерывания) расположены в регистре PI_PEND. Переполнение/обнуление таймеров также вызывает установку битов запроса прерывания. Даже если прерывание замаскировано, программное обеспечение может опрашивать регистры запросов прерываний, чтобы определить, было ли событие.

11 Аналого-цифровой преобразователь (АЦП)

Этот раздел описывает работу АЦП и объясняет как его программировать.

11.1 АЦП. Краткий функциональный обзор

Аналого-цифровой преобразователь (АЦП) преобразует аналоговое входное напряжение в 8- или 10-битное цифровое значение и устанавливает бит отложенного прерывания при сохранении результата. Преобразователь также контролирует вход, устанавливая бит отложенного прерывания, когда входное напряжение становится выше или ниже запрограммированного порогового значения. Блок-схема АЦП приведена на рисунке 11.1.



† Мультиплексированы с входами порта

Рисунок 11.1 – Блок-схема АЦП

11.2 Сигналы и регистры АЦП

В таблице 11.1 приведен список сигналов АЦП, в таблице 11.2 описаны регистры состояния и управления. Хотя аналоговые вводы мультиплексированы с выводами порта ввода-вывода, дополнительной конфигурации для этого не требуется.

Таблица 11.1 – Выводы АЦП

Вывод порта	Сигнал АЦП	Тип сигнала АЦП	Описание
P1.4:0	ACH12:8	Вход	Аналоговые входы.
P0.7:0	ACH7:0	Вход	Аналоговые входы.
—	∩0V	GND	Аналоговая земля должна быть подключена для работы АЦП и порта.
—	∩VCC2	PWR	Опорное напряжение должно быть подключено для работы АЦП и порта.

Таблица 11.2 – Регистры управления и состояния АЦП

Мнемоника	Адрес	Описание
AD_COMMAND	1FACH	Регистр команд АЦП. Этот регистр выбирает АЦП канал, определяет, начинается ли АЦП преобразование немедленно или по команде EPA и выбирает режим преобразования.
AD_RESULT	1FAAH, 1FABH	Регистр результата АЦП преобразования. Для АЦ преобразования старший байт содержит восемь старших битов преобразованного кода. Младший байт содержит два младших бита при 10-битном преобразовании (не определены при 8-битном преобразовании), показывает, какой АЦП канал использовался и не занят ли канал. Для определения пороговой величины вычисляют значение для регистра последовательного приближения и записывают значение в старший байт AD_RESULT. Младший байт очищают или оставляют его в заданном по умолчанию состоянии.
AD_TEST	1FAEH	Регистр проверки АЦ преобразования. Регистр осуществляет коррекцию ошибок смещения нуля.
AD_TIME	1FAFH	Регистр времени АЦ преобразования. Этот регистр определяет время выборки и время преобразования для каждого бита.
INT_MASK	0008H	Регистр маски прерывания. Бит AD этого регистра разрешает или запрещает прерывание АЦП. Установка AD бита разрешает запрос этого прерывания.
INT_PEND	0009H	Регистр отложенного прерывания. Установленный бит AD этого регистра показывает, что запрос АЦП прерывания отложен.
P0_PIN	1FA8H	Регистр состояния выводов порта 0. Чтение P0_PIN показывает текущие значения выводов порта 0. Чтение порта вызывает помехи в АЦП, уменьшая точность текущего преобразования. Настоятельно рекомендуется не производить чтение порта во время АЦ преобразования. Для уменьшения помех регистр P0_PIN тактируется только тогда, когда порт читается.
P1_PIN	1FA9H	Регистр состояния выводов порта 1. Чтение P1_PIN показывает текущие значения выводов порта 1. Чтение порта вызывает помехи в АЦП, уменьшая точность текущего преобразования. Настоятельно рекомендуется не производить чтение порта, во время АЦ преобразования. Для уменьшения помех регистр P1_PIN тактируется только тогда, когда порт читается.

11.3 Работа АЦП

АЦП преобразует аналоговое входное напряжение в цифровое значение, хранит результат в регистре AD_RESULT и устанавливает бит отложенного прерывания АЦП. 8-битное преобразование обеспечивает разрешение 20 мВ, 10-битное преобразование обеспечивает разрешение 5 мВ. 8-битное преобразование требует меньше времени, чем 10-битное, потому что используется на два бита меньше для преобразования и

компаратор требует меньшего времени установки для выполнения преобразования с шагом 20 мВ, чем с шагом 5 мВ.

Имеется возможность преобразовывать напряжение на аналоговом входном канале или тестовое напряжение. Преобразование тестовых входов позволяет вычислить ошибку смещения нуля, а коррекция смещения нуля позволяет ее компенсировать. Эта особенность позволяет уменьшить или устранить внешнюю аппаратную компенсацию. Обычно преобразование тестового напряжения используется для коррекции ошибки смещения нуля перед выполнением преобразованиями на входном канале. Регистр AD_TEST позволяет программировать коррекцию смещения нуля.

Пороговый детектор сравнивает входное напряжение с запрограммированным значением опорного напряжения и устанавливает бит ожидания прерывания, когда входное напряжение становится ниже или выше опорного.

Преобразование может быть начато записью в регистр AD_COMMAND или вызвано EPA, который может обеспечивать выборки с равными интервалами или синхронно с внешними событиями. Режим АЦП сканирования сервера периферийных транзакций (PTS) позволяет выполнять многократные преобразования и хранить их результаты.

Как только АЦП получает команду начать преобразование, образуется время задержки до начала выборки. (Преобразования, запущенные модулем EPA, начинаются после события захвата/сравнения. Преобразования, вызванные прямой записью в AD_COMMAND, начинаются в течение трех циклов после завершения команды.) В течение времени этой задержки выборки аппаратные средства очищают регистр последовательного приближения и выбирают заданный канал входного мультиплексора. После этой задержки устройство подключает выход мультиплексора к запоминающей емкости на указанное время выборки. После завершения этого времени устройство отключает выход мультиплексора от запоминающей емкости так, чтобы изменения на входном выводе не изменили хранимый заряд, во время процесса преобразования. Затем устройство обнуляет компаратор и начинает преобразование.

При выполнении аналого-цифрового преобразования используется алгоритм последовательного приближения. В состав аппаратуры преобразователя входят резистивное дерево с 256 резисторами, компаратор, конденсаторы и 10-битный регистр последовательного приближения (SAR) с логикой управления процессом. Резистивное дерево обеспечивает шаг в 20 мВ (∩VCC2 = 5,12 В), в то время как цепочка емкостей создает шаг в 5 мВ внутри шага в 20 мВ. Таким образом, имеется 1024 уровня опорного напряжения для сравнения с аналоговым входом для генерации 10-битного двоичного результата. В 8-битном режиме преобразования используется только резистивное дерево, обеспечивающее 256 уровней опорного напряжения.

Алгоритм последовательного приближения осуществляет сравнение последовательности заданных напряжений резистивного дерева с величиной на аналоговом входе методом половинного деления, чтобы обеспечить лучшее приближение. Первое сравнение – с 1/2 значения полного опорного напряжения. Оно дает 10-битный двоичный код, в котором старший значащий бит = 0, остальные – единицы (0111111111B). Если на аналоговом входе значение меньше, чем контрольное значение, бит 10 в SAR останется нулевым, и вход будет сравниваться с новым контрольным напряжением, равным 1/4 полной шкалы (0011111111b). Если значение на аналоговом входе больше тестового напряжения, устанавливается 9 бит SAR. 8 бит затем очищается для следующего теста (0101111111b). Такой бинарный поиск продолжается, пока не будут произведены 10 (или 8) сравнений, к этому времени результат преобразования будет располагаться в регистре AD_RESULT, откуда может быть программно считан. Результат равен отношению входного напряжения к напряжению аналогового источника. Если отношение равно 1,00, двоичный результат будет содержать все единицы.

11.4 Программирование АЦП

Программируются следующие параметры АЦП:

- вход преобразователя – выбор входного канала;
- подстройка смещения нуля – нет коррекции, плюс 2,5 мВ, минус 2,5 мВ, минус 5,0 мВ;
- временные параметры преобразования – время выборки и время преобразования для каждого бита;
- режим преобразования – 8- или 10-битное преобразование, детектирование низкого или высокого порога;
- запуск преобразования – непосредственный или запуск модулем EPA.

Далее приводится описание регистров АЦП и способы их программирования.

Программирование регистра проверки АЦ преобразования

Регистр AD_TEST (рисунок 11.2) определяет напряжение смещения, которое будет приложено к резистивному дереву. Чтобы использовать коррекцию смещения нуля, сначала выполняют два преобразования: одно на ∩0V, другое на ∩VCC2. Результаты этих преобразований используются для программного вычисления ошибки смещения нуля. Корректировка смещения нуля производится с помощью записи соответствующего значения в AD_TEST. Это напряжение смещения добавляется к резистивному дереву и используется со всеми входными каналами.

AD_TEST	Адрес:	1FAEH
	Состояние сброса	C0H

Регистр проверки АЦП преобразования (AD_TEST) определяет установки корректировки ошибок смещения по постоянному току.

7							0
-	-	-	OFF1	-	OFF0	-	-

Номер разряда	Мнемоника	Функция
7:5	—	Зарезервированы; для совместимости с будущими устройствами в эти биты записывают нули.
4	OFF1	Смещение. Этот бит, наряду с OFF0 (бит 2), позволяет устанавливать точку смещения нуля. OFF1 OFF0 0 0 нет коррекции 0 1 прибавление 2,5 мВ 1 0 вычитание 2,5 мВ 1 1 вычитание 5,0 мВ
3	—	Зарезервирован; для совместимости с будущими устройствами в этот бит записывают ноль.
2	OFF0	См. бит 4 (OFF1).
1:0	—	Зарезервированы; для совместимости с будущими устройствами в эти биты записывают нули.

Рисунок 11.2 – Регистр проверки АЦП преобразования (AD_TEST)

Программирование регистра результата АЦП (только для детектирования порога)

Чтобы использовать режим детектирования порога сначала необходимо записать в старший байт AD_RESULT значение для установки желаемого опорного (порогового) напряжения.

AD_RESULT (запись) Адрес: 1FAAH
Состояние сброса: FFC0H

В старший байт регистра результата АЦП (AD_RESULT) может быть записано значение для установки опорного напряжения в режиме детектирования порога.

15							8
REV7	REV6	REV5	REV4	REV3	REV2	REV1	REV0
7							0
–	–	–	–	–	–	–	–

Номер разряда	Мнемоника	Функция
15:8	REFV7:0	Опорное напряжение. Эти биты определяют пороговое значение. Они выбирают опорное напряжение для сравнения с аналоговым входным напряжением. Когда напряжение на аналоговом входе становится выше или ниже порогового значения, бит отложенного прерывания по завершению АЦ преобразования устанавливается. Для определения значения, записываемого в регистр, необходимо воспользоваться следующей формулой: Опорное напряжение = желаемое пороговое напряжение × 256 /(∩VCC2 – ∩0V)
7:0	—	Зарезервированы; для совместимости с будущими устройствами в эти биты записывают нули.

Рисунок 11.3 – Формат записи регистра результата АЦП (AD_RESULT)

Программирование регистра времени АЦП преобразования

Два параметра – время выборки и время преобразования, управляют временем, требуемым для АЦП преобразования. Время выборки – это время, в течение которого аналоговый вход подключен к запоминающей емкости. Если это время слишком коротко, емкость не зарядится полностью. Если время выборки слишком затянуто, входное напряжение может измениться, что вызовет ошибку преобразования. Время преобразования – это время, требуемое для преобразования величины напряжения, сохраненной на запоминающей емкости, в цифровое значение. Время преобразования должно быть достаточным, чтобы компаратор схема преобразования успели установится и вычислить напряжение. Слишком большое время преобразования позволяет запоминающей емкости разрядиться и уменьшает точность.

Регистр AD_TIME (рисунок 11.4) определяет время выборки и время преобразования АЦП.

AD_TIME Адрес: 1FAFH

Состояние сброса: FFH

Регистр времени АЦП (AD_TIME) программирует время выборки и время преобразования для каждого бита. Этот регистр программирует скорость, с которой может идти АЦ преобразование, но не скорость, с которой это преобразование будет проходить корректно. Необходимо инициализировать регистр AD_TIME перед инициализацией регистра AD_COMMAND. Нельзя записывать значения в этот регистр во время преобразования – результаты непредсказуемы.

7							0
SAM2	SAM1	SAM0	CONV4	CONV3	CONV2	CONV1	CONV0

Номер разряда	Мнемоника бита	Функция
7:5	SAM2:0	Время выборки АЦП. Эти разряды определяют время выборки. Для вычисления времени выборки используется следующая формула $SAM=(T_{SAM} \times F_{BQ1} - 2)/8$, где SAM – от 1 до 7 , T_{SAM} – время выборки, мкс, F_{BQ1} – входная частота на BQ1, МГц
4:0	CONV4:0	Время АЦ преобразования. Эти биты определяют время преобразования для каждого бита. Для вычисления времени преобразования используется следующая формула: $CONV=(T_{CONV} \times F_{BQ1} - 3) / (2 \times B) - 1$, где CONV – от 2 до31, T_{CONV} – время преобразования, мкс, F_{BQ1} – входная частота на BQ1, МГц, B – число битов, которые будут преобразованы (8 или 10).

Рисунок 11.4 – Регистр времени АЦП (AD_TIME)

Программирование регистра команд АЦП

Регистр команд АЦП управляет режимом преобразования, аналоговым входным каналом и запуском преобразования.

AD_COMMAND Адрес: 1FACH

Состояние сброса: 80H

Регистр команд АЦП (AD_COMMAND) выбирает номер АЦП канала, который будет преобразован, определяет, начинается ли АЦП преобразование немедленно или вызывается командой EPA, и выбирает режим преобразования.

7							0
-	M1	M0	GO	ACH3	ACH2	ACH1	ACH0

Номер разряда	Мнемоника	Функция
7	—	Зарезервирован, для совместимости с будущими устройствами записывают в этот бит нуль.
6:5	M1:0	Режим АЦ преобразования[1]. Эти биты определяют режим АЦП. M1 M0 Режим 0 0 10-битное преобразование 0 1 8-битное преобразование 1 0 обнаружение напряжения выше порогового 1 1 обнаружение напряжения ниже порогового
4	GO	Запуск АЦ преобразования [2]. Запись этого бита запускает АЦ преобразователь. Значение, записанное в него, определяет, когда должно начаться преобразование. 0 – EPA запускает преобразование, 1 – начинается немедленно
3:0	ACH3:0	Выбор канала АЦП. В эти биты записывается номер канала АЦП.

[1] Когда выбран режим детектора порога для аналогового входа, никакое другое преобразование не может быть начато. Если другое значение загружено в AD_COMMAND, режим детектора порога запрещен и выполняется новая команда.

[2] Преобразование запускается записью в бит GO, а не его значением. Даже если бит GO имеет нужное значение, его необходимо установить снова, чтобы начать преобразование немедленно или же повторно очистить его для запуска преобразования модулем EPA.

Рисунок 11.5 – Регистр АЦП команд (AD_COMMAND)

Разрешение прерывания АЦП

АЦП может установить бит ожидания прерывания АЦП, когда преобразование заканчивается или когда входное напряжение переходит пороговое значение. Для разрешения прерывания необходимо установить соответствующий маскирующий бит регистра маскирования прерываний и выполнить EI инструкцию, чтобы полностью разрешить обслуживание прерываний. Прерывание АЦП может заставить PTS начать новое преобразование.

11.5 Определение состояния АЦП и результатов преобразования

Чтобы определить состояние АЦП преобразователя, осуществляется чтение из регистра AD_RESULT (рисунок 11.6). Регистр AD_RESULT очищается, когда новое преобразование начато. Поэтому, чтобы предотвратить потерю данных, необходимо прочитать оба байта прежде, чем начнется новое преобразование. Если чтение из AD_RESULT происходит прежде, чем преобразование завершено, не может быть гарантирована точность результата.

Результат преобразования – отношение входного напряжения к эталонному напряжению:

RESULT (8-бит) = $255 \times (V_{IN} - \cap 0V) / (\cap VCC2 - \cap 0V)$

RESULT (10-бит) = $1023 \times (V_{IN} - \cap 0V) / (\cap VCC2 - \cap 0V)$

Можно также читать регистр отложенного прерывания INT_PEND (бит AD) для определения состояния АЦП прерывания.

AD_RESULT (Чтение) Адрес: 1FAAH
Состояние сброса FFC0H

Регистр результата АЦП (AD_RESULT) состоит из двух байтов. Старший байт содержит восемь старших битов АЦ преобразования. Младший байт содержит два младших бита 10-битного АЦ преобразования, указывает номер АЦП канала, который использовался для преобразования, и указывает, происходит ли преобразование в настоящее время.

15							8
ADRLT9	ADRLT8	ADRLT7	ADRLT6	ADRLT5	ADRLT4	ADRLT3	ADRLT2
7							0
ADRLT1	ADRLT0	—	STATUS	ACH3	ACH2	ACH1	ACH0

Номер разряда	Мнемоника	Функция
15:6	ADRLT9:0	Результат АЦП. Эти биты содержат результат АЦ преобразования.
5	—	Зарезервирован. Этот бит не определён.
4	STATUS	Состояние АЦП. Показывает состояние АЦП. До 8 тактов требуется, чтобы установить этот бит после начала команды. При проверке этого бита необходимо ждать, по крайней мере, 8 тактов. 0 – АЦ преобразование не выполняется. 1 – АЦ преобразование выполняется.
3:0	ACH3:0	Номер АЦП канала. Эти биты указывают номер АЦП канала, который использовался для преобразования.

Рисунок 11.6 – Формат чтения регистра результата АЦП (AD_RESULT)

11.6 Примеры схем внешнего интерфейса и ошибок АЦП преобразования

Этот подраздел описывает примеры внешней схемы интерфейса и ошибки, которые могут произойти в любом АЦ преобразователе.

Проектирование внешней схемы интерфейса

Внешняя схема интерфейса с аналоговым входом определяется конкретным применением и влияет на характеристики преобразователя. При разработке внешней схемы такие факторы, как входной ток утечки, запоминающая емкость, последовательное сопротивление мультиплексора от вывода до конденсатора выборки должны быть правильно выбраны. Эти факторы (с идеализированными значениями и схемой) приведены на рисунке 11.7.

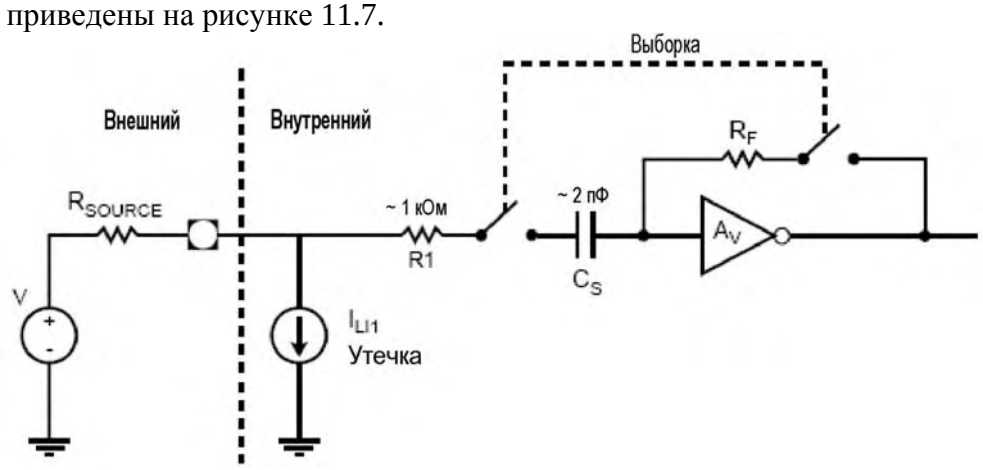


Рисунок 11.7 – Идеализированная схема интерфейса АЦП

В течение времени выборки внешняя входная схема должна быть способна зарядить типовой конденсатор (Cs) через последовательное сопротивление источника (R_{SOURCE}), входное сопротивление схемы (R1) и сопротивление обратной связи компаратора (RF). Полное эффективное последовательное сопротивление рассчитывается с использованием следующей формулы:

$R_T=R_{SOURCE}+R1+R_F/(A_V+1)$,

где A_V – коэффициент усиления схемы компаратора.

Уменьшение эффекта высокого входного исходного сопротивления

При некоторых условиях входное сопротивление источника (R_{SOURCE}) может быть достаточно большим, чтобы повлиять на измерение. Можно минимизировать этот эффект, увеличивая время выборки или подключая внешний конденсатор (C_{EXT}) между входным выводом и ∩0V. Внешний сигнал зарядит C_{EXT} до уровня напряжения источника. Когда канал выбран, C_{EXT} действует как источник с низким импедансом при заряде конденсатора выборки (Cs). Малая часть заряда C_{EXT} передается Cs, происходит снижение выбранного напряжения. Снижение напряжения рассчитывается, исходя из следующей формулы:

Снижение выбранного напряжения, % = $Cs / (C_{EXT} + C_S) \times 100$ %.

Если C_{EXT} равно 0,005 мкФ или больше, ошибка будет меньше, чем (– 0,4) LSB при 10-битном режиме преобразования. Использование C_{EXT} в соединении с R_{SOURCE} формирует фильтр низких частот на входе, снижая помеху на входе.

Высокое R_{SOURCE} может также стать причиной ошибки из-за входного тока утечки (I_{LI1}). I_{LI1} обычно намного ниже, чем его нормы в технической документации. Объединенный эффект тока утечки и высокого R_{SOURCE} рассчитывается по следующей формуле:

Ошибка (LSBs) = $R_{SOURCE} \times I_{LI1} \times 1024 / \cap VCC2$,

где

R_{SOURCE} - является входным сопротивлением источника, Ом;

I_{LI1} - является током входной утечки, А;

∩VCC2 - является опорным напряжением, В.

Во внешних схемах с сопротивлением R_{SOURCE}, равным 1 кОм или ниже и ∩VCC2, равным 5,0 В из-за импеданса источника будет происходить ошибка 0,6 LSB или меньше.

Примеры входных схем АЦП преобразователя

Предложенная схема входа АЦП показана на рисунке 11.8, она обеспечивает защиту против перенапряжения на аналоговом вводе. При входном напряжении ниже ∩0V или выше ∩VCC2 диод D2 или D1 осуществляет прямое смещение около 0,8 В. Защита входа устройства начинает включаться при отклонении приблизительно 0,5 В от напряжения ∩0V или ∩VCC2. Резистор 270 Ом ограничивает входной ток на аналоговом входе до безопасного значения, меньше 1 мА.

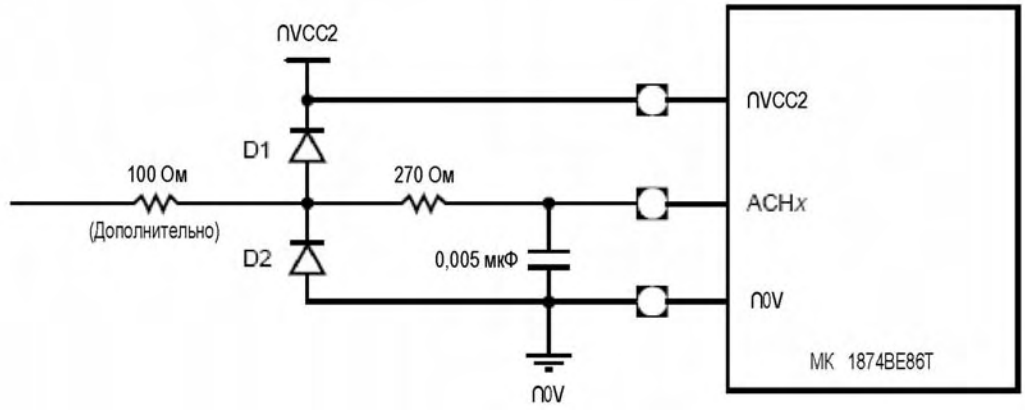


Рисунок 11.8 – Рекомендуемая входная схема АЦП

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

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. I_{REF} может измениться между 2 мА и 5 мА в течение преобразования. Для минимизации этого эффекта следует использовать фильтрующий керамический или танталовый конденсатор 1,0 мкФ между ∩VCC2 и ∩0V. Потенциал ∩0V должен быть в пределах ± 50 мВ по отношению к #0V. ∩VCC2 должен быть стабильным и использоваться только для АЦП. Питание ∩VCC2 может быть между 4,5 В и 5,5 В и должно обеспечивать ток приблизительно 5 мА. ∩VCC2 должно быть приблизительно равно #VCC1. Большие отрицательные пики токов на выводе ∩0V относительно #0V могут вызвать эффект «защёлки» в аналоговой цепи. Это является дополнительной причиной, из-за которой следует тщательно заземлять устройство.

Аналоговое опорное напряжение (∩VCC2) является положительным напряжением, с которым происходит сравнение при АЦ преобразовании. Это также питание и порта 0, и порта 1, если АЦП не используется. Если высокая точность не требуется, ∩VCC2 может быть соединен с #VCC1. Если необходима высокая точность преобразования, ∩VCC2 должен быть очень стабильным. Один из путей достижения 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.

Совместное использование аналоговых и цифровых входов

Порт 0 и порт 1 могут быть использованы и для аналоговых, и для цифровых входных сигналов одновременно. Однако чтение порта может наводить помехи в аналоговую цепь. По этой причине необходимо гарантировать, что не идет процесс АЦ преобразования, когда требуется читать порт. В разделе 7 “Порты ввода-вывода” описано использование порта как цифровых входов.

Передаточная функция и источники ошибок АЦ преобразования

Результат преобразования – это отношение входного напряжения к опорному.

RESULT (8-битный) $=255\times(U_{IN} – ∩0V) / (∩VCC2 - ∩0V)$

RESULT (10-битный) $=1023\times(U_{IN} – ∩0V) / (∩VCC2 - ∩0V)$

Получается ступенчатая передаточная функция, когда выходной код отображает значение входного напряжения. Полученный цифровой код может трактоваться как информация о простом соотношении напряжений, абсолютном значении напряжения на входе или его относительном изменении.

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

Во многих применениях менее критична абсолютная точность на входе, чем изменение на входе. Этот подход допустим, пока идет преобразование непрерывного сигнала, и нет ошибочных кодов.

Увеличение входных напряжений приводит к тому, что выходные коды также увеличиваются. Уменьшение входных напряжений приводит к тому, что выходные коды также уменьшаются. Другими словами, существует определенный диапазон входных напряжений для каждого 10-битного выходного кода, который производится с ошибкой только ±0,25 LSBs (1,5 мВ).

Внутренние ошибки АЦП преобразования следующие: ошибка дискретизации, ошибка смещения нуля, ошибка усиления, дифференциальная и интегральная нелинейности.

Кроме того, должны учитываться температурные коэффициенты, флуктуации #VCC1, перекрытие выборки и хранения, 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.

Неизбежная ошибка следует из преобразования непрерывного напряжения к целому числу в цифровом представлении. Эта ошибка, называемая ошибкой квантования, всегда ±0,5 LSB. Ошибка квантования – единственная ошибка, замеченная в идеальном АЦП преобразователе, и она, очевидно, присутствует в реальных преобразователях. Рисунок 11.9 показывает передаточную функцию идеального 3-битного АЦП.

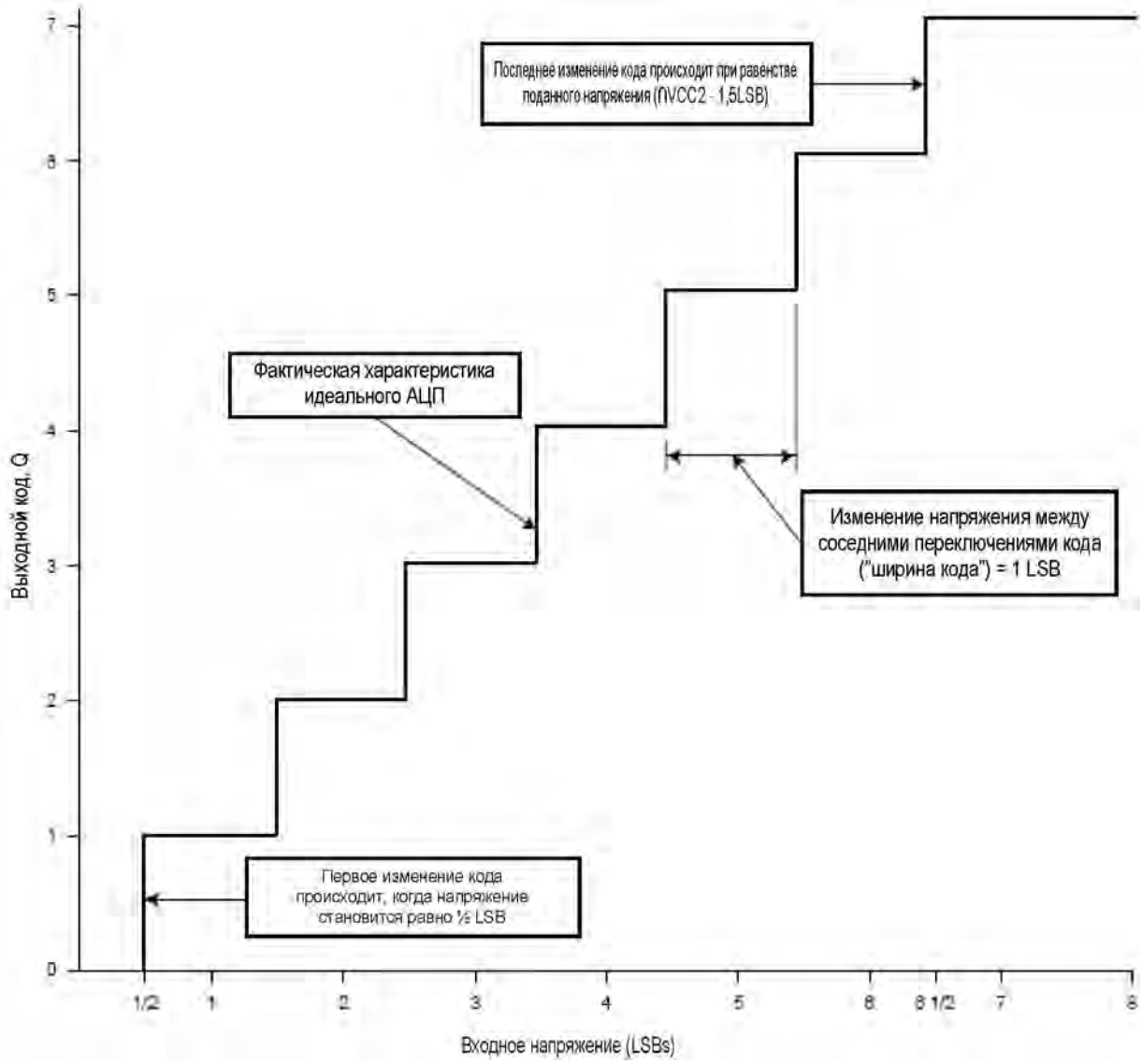


Рисунок 11.9 – Идеальная передаточная характеристика АЦП преобразователя

Идеальная характеристика обладает следующими особенностями:

- появление 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 полного диапазона и нелинейностей. Другими словами, это идеальное преобразование.

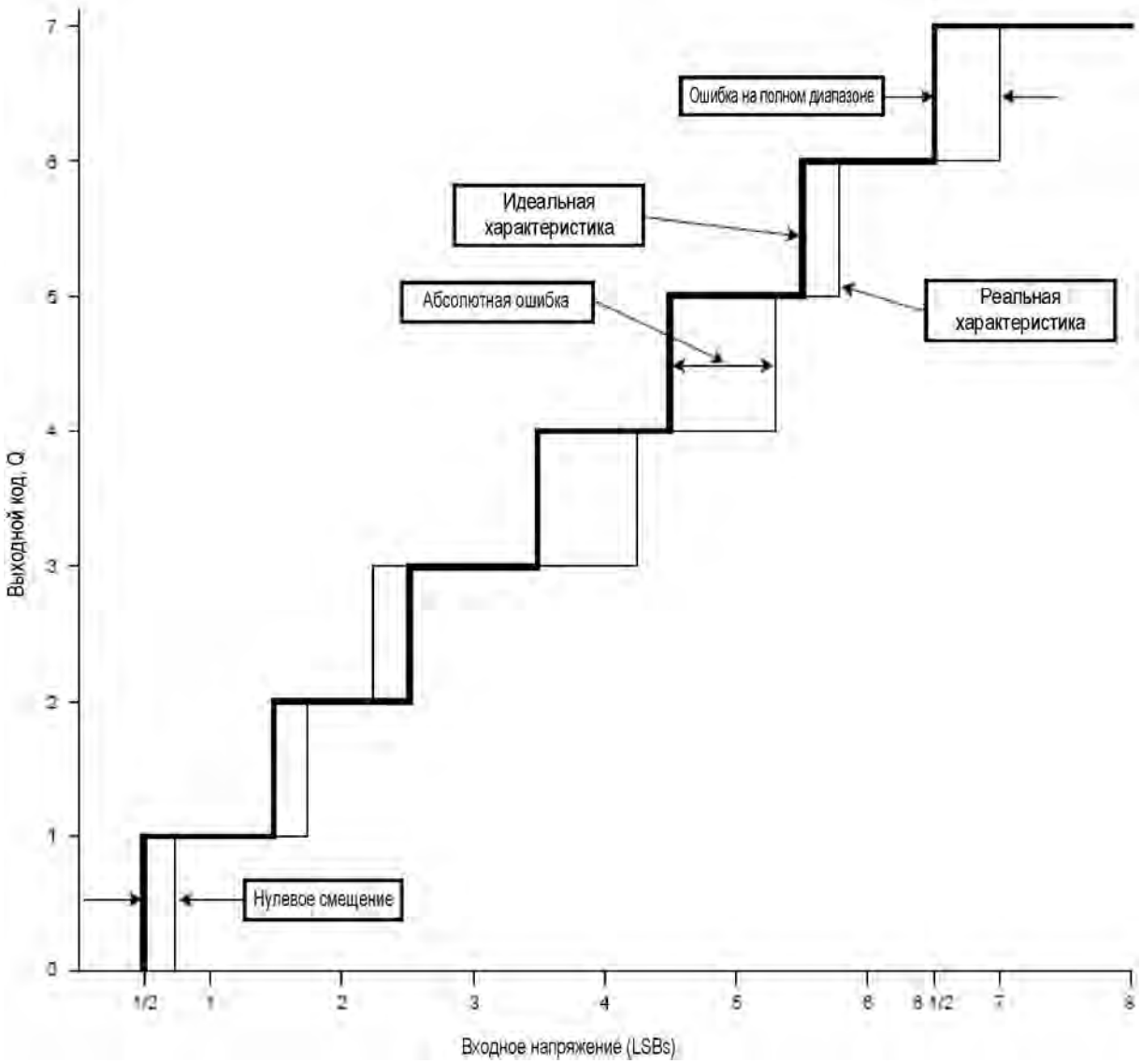


Рисунок 11.10 – Реальная и идеальная передаточные характеристики АЦП преобразователя

Реальная характеристика 3-битного преобразователя отличается от идеальной. Если идеальную характеристику сопоставить с реальной, то реальная характеристика, как показано на рисунке 11.10, имеет как ошибки в расположении минимального и максимального кодового напряжения, так и в величине интервала (кванта). Смещение минимального значения кода от идеального называется ошибкой нулевого смещения, смещение максимального кодового значения от идеального называется ошибкой на полном диапазоне (fullscale 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. Если дифференциальная нелинейность превышает 1 LSB, то в выходном сигнале может отсутствовать одна из кодовых комбинаций (выпадающий код). В 10-битном преобразователе идеальный шаг квантования 5 мВ (∩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.

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 смещения нуля и ошибки на полном диапазоне, как показано на рисунке 11.11. Terminal Based характеристика подобна реальной, в которой ошибки нулевого смещения 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 может точно регулироваться в заданном диапазоне для уменьшения ошибки полного диапазона.

Другие факторы, которые воздействуют на реальную систему АЦП преобразователя, включают: температурный дрейф, импульсные помехи, взаимное влияние каналов мультиплексора и случайный шум. Но обычно эти влияния являются незначительными. Температурный дрейф – это скорость изменения характеристик с изменением температуры. Эти изменения отражаются в температурных коэффициентах.

Основные источники паразитных сигналов - это помехи по питанию, изменение входного сигнала в канале во время преобразования (после заряда эталонной емкости) и сигналы на каналах, не выбранных мультиплексором.

Входное сопротивление мультиплексора немного 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.

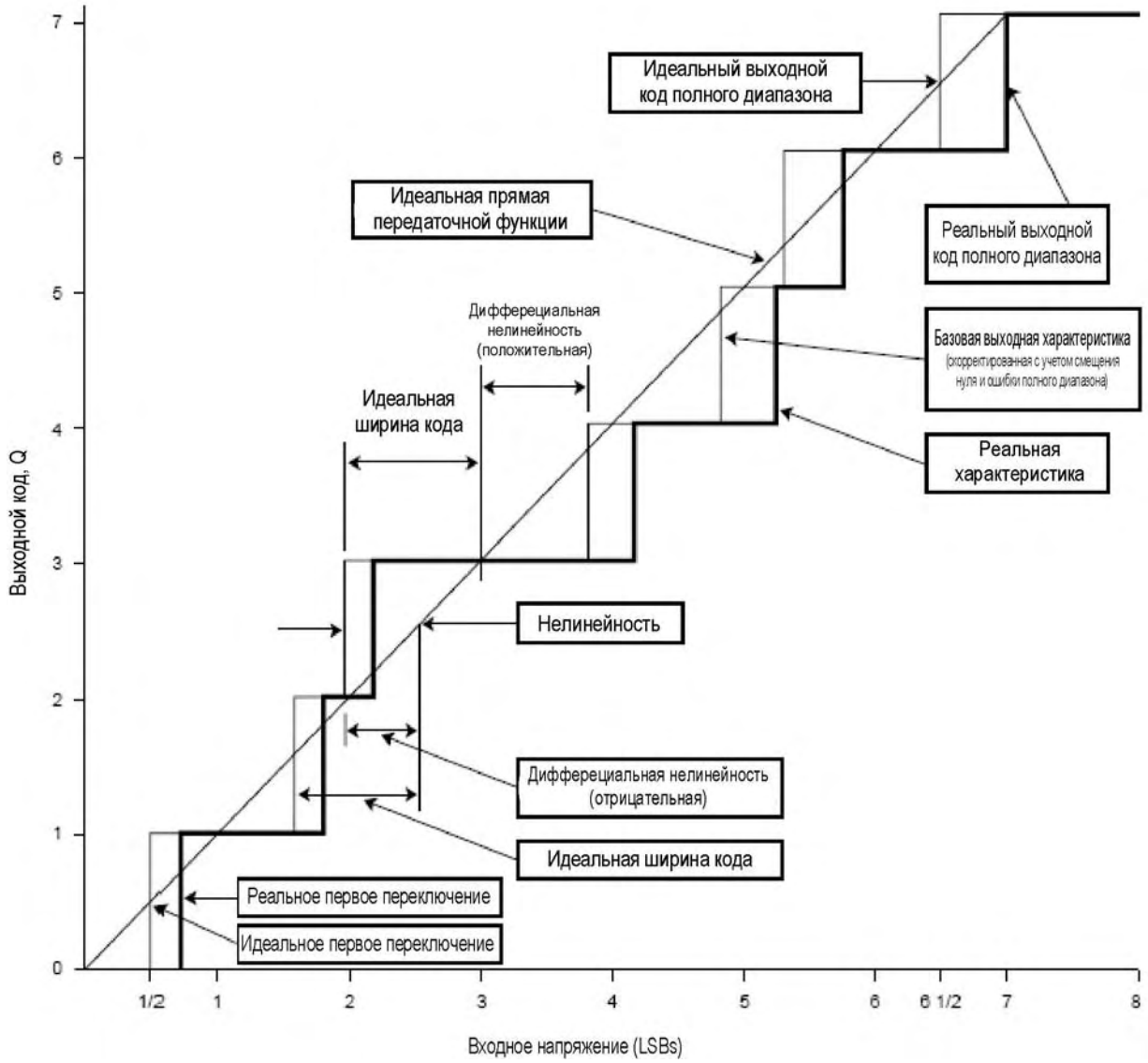


Рисунок 11.11 – Выходная передаточная характеристика АЦП

12 Специальные режимы работы

В 1874BE86T и 1874BE16T предусмотрены два режима экономии мощности: холостой ход (IDLE) и низкое потребление (POWERDOWN). Также предусмотрен режим внутрисхемного эмулятора (ONCE), который электрически изолирует устройство от других компонентов системы. Этот раздел описывает указанные режимы и вход и выход из них.

12.1 Сигналы и регистры управления специальными режимами

В таблице 12.1 приведен список сигналов, в таблице 12.2 приведен список регистров, работа которых в специальных режимах рассматривается в разделе 12.

Таблица 12.1 – Режим работы сигналов управления

Вывод порта	Название сигнала	Тип	Описание
1	2	3	4
P2.7	CLKOUT	Выход	Выход системного тактового сигнала. Выход внутреннего генератора синхроимпульсов. Частота CLKOUT – 1/2 частоты на выводе генератора (F_{BQ1}). CLKOUT имеет 50 % скважность.
—	EXINT	Вход	Внешнее прерывание. Это программируемое прерывание управляется регистром WG_PROTECT. Этот регистр определяет, вызывается прерывание фронтом или уровнем и передним фронтом, высоким уровнем, или задним фронтом, низким уровнем. В режиме POWERDOWN он используется как вход, чувствительный к уровню. Установка сигнала EXINT, по крайней мере, на 50 нс вызывает выход из режима. Прерывание при этом может быть запрещено. Если прерывание от EXINT разрешено, начинает выполняться процедура обслуживания прерывания. Иначе, центральный процессор выполняет инструкцию, которая сразу следует за командой, которая вызвала режим POWERDOWN. В режиме холостого хода установление любого разрешенного прерывания вызывает возобновление нормальной работы.
P5.4	ONCE#	Вход	Внутрисхемная эмуляция. Удержание низкого уровня на выводе во время нарастающего (положительного) фронта сигнала RESET# вызывает режим внутрисхемной эмуляции (ONCE). Этот режим переводит все выводы, кроме BQ1 и BQ2, в состояние высокого импеданса, таким образом, изолируя МК от других компонентов в системе. Значение ONCE# захватывается, когда на вывод RESET# подается неактивный уровень. В то время как МК находится в режиме ONCE, можно производить отладку системы. Для выхода из режима ONCE надо сбросить устройство низким уровнем сигнала RESET#. Чтобы предотвратить незапланированный вход в режим ONCE, можно сконфигурировать его как выход или подавать высокий уровень при выполнении сброса.

Продолжение таблицы 12.1

1	2	3	4
P2.6	Вход в тестовый режим	Вход - выход	Вход в тестовый режим. Если этот вывод удерживать в низком уровне в течение сброса (RESET#), устройство войдет в резервный тестовый режим, поэтому необходимо соблюдать меры предосторожности при использовании этого вывода как входа. Если необходимо сформировать этот вывод как вход, необходимо всегда удерживать высокий уровень на выводе в течение сброса для предотвращения случайного входа в тестовый режим.
—	RESET#	Вход - выход	Сброс. Чувствительный к уровню вход сброса и выход с открытым стоком системного сброса микроконтроллера. Задний фронт RESET# и низкий уровень или внутренний сброс, включающий обнуляющий транзистор, связанный с выводом RESET#, удерживаются не менее 16-ти тактов. Для режимов POWERDOWN и IDLE выполнение сброса переводит МК в нормальный операционный режим. После сброса устройства, адрес первой инструкции 2080H.
—	VPP	Пита-ние	Напряжение программирования. В течение программирования на вывод VPP подается типовое напряжение (+12,5 В). Превышение максимума напряжения на выводе (+13,0 В) может повредить устройство. Вывод можно использовать для выхода из режима POWERDOWN. Используется этот метод для выхода из режима POWERDOWN только при использовании внешнего источника синхроимпульсов. На устройствах без внутреннего энергонезависимого запоминающего устройства надо соединить VPP с #VCC1.

Таблица 12.2 – Регистры состояния и управления режимом

Мнемоника	Адрес	Описание
1	2	3
CCR0	2018H	Регистр конфигурации кристалла 0. Бит 0 этого регистра разрешает и запрещает режим POWERDOWN.
INT_MASK1	0013H	Маска прерывания 1. Бит 6 этого 8-ми битного регистра разрешает и запрещает (маскирует) внешнее прерывание (EXINT).
P2_DIR P5_DIR	1FD2H 1FF3H	Управление порта_x. Каждый бит Px_DIR управляет конфигурацией соответствующего вывода. Очистка бита формирует вывод как комплементарный выход; установка – формирует вывод как выход с открытым стоком или вход. (Выводы с открытым стоком требуют внешнего напряжения поддержки.)
P2_MODE P5_MODE	1FD0H 1FF1H	Режим порта_x. Каждый бит Px_MODE управляет функцией соответствующего вывода как стандартного входа-выхода или как сигнала специальной функции. Установка бита формирует вывод как сигнал специальной функции; очистка бита формирует вывод как стандартный вход-выход.

Продолжение таблицы 12.2

1	2	3
P2_REG P5_REG	1FD4H 1FF5H	Порт_x Вывод данных Для входа устанавливают соответствующий Px_REG бит. Для вывода записывают данные, которые будут выданы каждым выходом в соответствующий бит Px_REG. Когда вывод формируется как стандартный вход-выход (Px_MODE.y = 0), результат записи центрального процессора в Px_REG немедленно появляется на выводе. Когда вывод формируется как сигнал специальной функции (Px_MODE.y = 1), соответствующий периферийный блок МК или компонент вне МК управляет выводом. Центральный процессор может продолжать запись в Px_REG, но вывод не меняет свое состояние, пока он не будет переключен назад к его стандартной функции ввода-вывода. Эта особенность позволяет программному обеспечению формировать вывод как стандартный вход-выход (очистить Px_MODE.y), инициализировать или переписывать значение на выводе, затем формировать вывод как сигнал специальной функции (установить Px_MODE.y). В этом случае инициализация, восстановление ошибки, обработка исключительных ситуаций и т.д. могут быть сделаны без изменения операций связанного периферийного устройства.

12.2 Сокращение потребления мощности

Оба экономящих мощность режима уменьшают потребляемую мощность, запрещая часть сигналов внутренней схемы синхроимпульсов (рисунок 12.1). Подразделы 12.3, 12.4 описывают оба режима подробно.

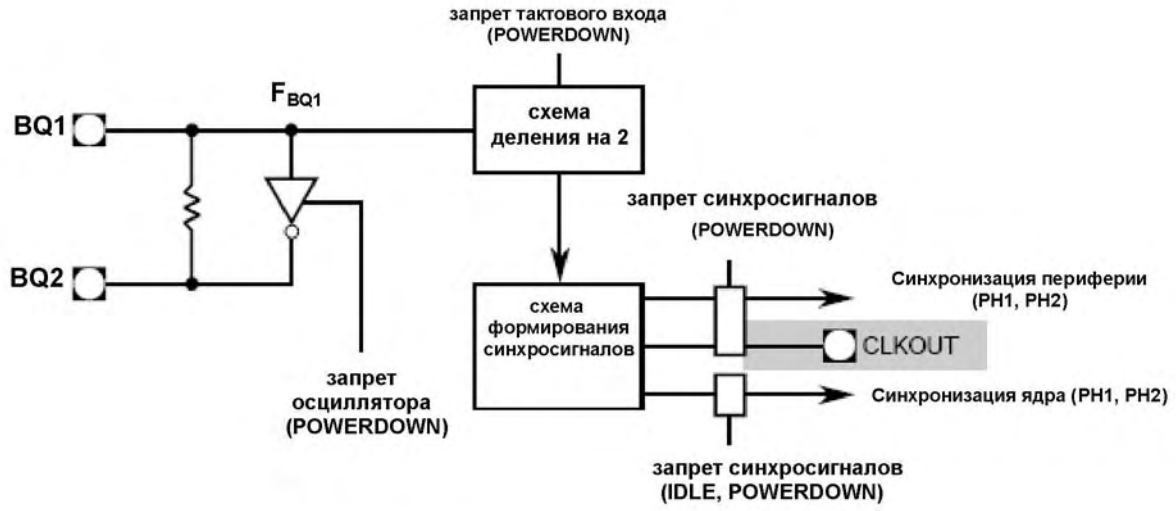


Рисунок 12.1 – Управление синхроимпульсами в течение режимов экономии мощности

12.3 Режим холостого хода (IDLE)

В режиме холостого хода потребление мощности устройства уменьшается приблизительно на 40 % от нормального потребления. Внутренняя логика удерживает сигналы синхронизации центрального процессора в логическом нуле, останавливая выполнение инструкций. Ни периферийные сигналы синхронизации, ни CLKOUT не отключаются, так что регистры специальной функции (SFR) и регистры RAM сохраняют

данные, периферийные устройства и система прерываний остаются активными. МК входит в режим холостого хода после выполнения инструкции IDLPD #1. Любой разрешённый источник прерывания, внутренний или внешний, или аппаратный сброс может вывести устройство из режима холостого хода. Когда происходит прерывание, сигналы синхронизации центрального процессора перезапускаются, и центральный процессор выполняет стандартную процедуру обслуживания прерывания или процедуру обслуживания прерывания PTS. Когда процедура завершена, конвейер центрального процессора восстанавливается и выполняется инструкция, которая следовала за IDLPD #1.

Примечание – Если разрешено, сторожевой таймер продолжает работать в режиме холостого хода. Устройство должно быть выведено из режима холостого хода прежде, чем счетчик переполнится. Иначе таймер сбросит МК. Интервал таймера – всегда 64К тактов.

Чтобы предотвратить случайное возвращение в рабочий режим, необходимо удерживать сигнал EXINT в состоянии низкого уровня, в то время как устройство находится в режиме холостого хода.

12.4 Режим низкого потребления (POWERDOWN)

Режим POWERDOWN переводит устройство в состояние очень низкого потребления мощности, выключая внутренний осциллятор и генератор синхроимпульсов. Внутренняя логика удерживает сигналы синхронизации периферии и центрального процессора в логическом нуле, заставляя центральный процессор прекращать выполнять инструкции, сигналы шинного управления системы становятся неактивными, CLKOUT устанавливается высоким уровнем, и периферийные устройства выключаются. Ток потребления устройства уменьшается до тока утечки. Если напряжение питания UCC не опускается ниже минимального, регистры специальных функций (SFRs) и регистры RAM сохраняют свои данные.

Разрешение и запрещение режима POWERDOWN

Бит PD в регистре конфигурации кристалла 0 (CCR0.0) разрешает или запрещает режим POWERDOWN. Поскольку к CCR0 нельзя обратиться по команде, значение бита PD определено в байте конфигурации кристалла 0 (CCB0.0). Установка бита PD разрешает режим POWERDOWN, а очистка запрещает режим. CCR0 загружается из CCB0, когда устройство выполняет рестарт после сброса.

Вход в режим POWERDOWN

Перед входом в режим низкого потребления надо завершить следующие задачи:

- Закончить все передачи или приемы последовательного порта. Иначе, когда устройство выходит из режима низкого потребления, работа последовательного порта продолжится с того места, где эта работа прервалась, и могут быть переданы или получены неправильные данные.
- Закончить все аналоговые преобразования. Если режим низкого потребления начинается в течение преобразования, результат будет неправилен.
- Если сторожевой таймер (WDT) разрешен, очистка WATCHDOG регистра производится перед инструкцией входа в режим низкого потребления. Это гарантирует, что устройство может выйти из режима низкого потребления корректно. Иначе, WDT может сбросить устройство прежде, чем осциллятор стабилизирует частоту. (WDT не может сбросить устройство в течение режима низкого потребления, потому что сброшены синхроимпульсы.)
- Все другие периферийные устройства перевести в неактивное состояние.

После завершения этих задач выполнить инструкцию IDLPD #2 для входа в режим низкого потребления.

Примечание – Чтобы предотвратить случайное возвращение в режим полной мощности, необходимо удерживать сигнал внешнего прерывания EXINT в состоянии низкого уровня, в то время как устройство находится в режиме низкого потребления.

Выход из режима низкого потребления (POWERDOWN)

Устройство выйдет из режима низкого потребления, когда произойдет любое из следующих событий:

- Внешнее устройство на выводе VPP выставляет низкий уровень не менее чем на 50 нс.
- Произведен системный сброс (RESET).
- Происходит переключение на выводе внешнего прерывания.

Задание низкого уровня на выводе VPP

Если устройство использует внешний тактовый сигнал, а не генерируемый на осцилляторе, то самый быстрый путь к выходу из режима низкого потребления – выставить низкий уровень на выводе VPP не менее, чем на 50 нс. Используется этот метод только при использовании внешнего тактового сигнала, потому что синхронизация центрального процессора и периферийных устройств будет разрешена, а внутренний генератор будет заблокирован.

Выполнение аппаратного сброса (RESET)

Устройство выйдет из режима низкого потребления, если RESET# установлен. Если используется внешний тактовый сигнал, а не осциллятор на кристалле, RESET# должен остаться низким в течение, по крайней мере, 16 тактов. Если используется осциллятор на кристалле, то RESET# должен быть удержан в состоянии низкого уровня до тех пор, пока частота осцилляторане стабилизируется.

Установка сигнала внешнего прерывания (EXTINT)

Последний способ выхода из режима низкого потребления состоит в том, чтобы установить сигнал EXTINT не менее чем на 50 нс. Хотя EXTINT обычно простой вход, схема режима низкого потребления использует его, как чувствительный к уровню вход. Прерывание может не быть разрешено, но вывод должен формироваться как вход специальной функции. Рисунок 12.2 показывает диаграмму последовательности включения и выключения режима низкого потребления при использовании вывода внешнего прерывания.

Когда внешнее прерывание выводит устройство из режима низкого потребления, соответствующий бит устанавливается в регистре ожидания прерывания. Если прерывание разрешено, устройство выполняет процедуру обслуживания прерывания, затем восстанавливает конвейер команд и выполняет инструкцию, следующую после инструкции IDLPD #2. Если прерывание запрещено (замаскировано), устройство восстанавливает конвейер команд и выполняет инструкцию, следующую после инструкции IDLPD #2, а бит ожидания прерывания остается установленным, пока не выполнится обслуживание прерывания или программное обеспечение не очистит бит ожидания прерывания.

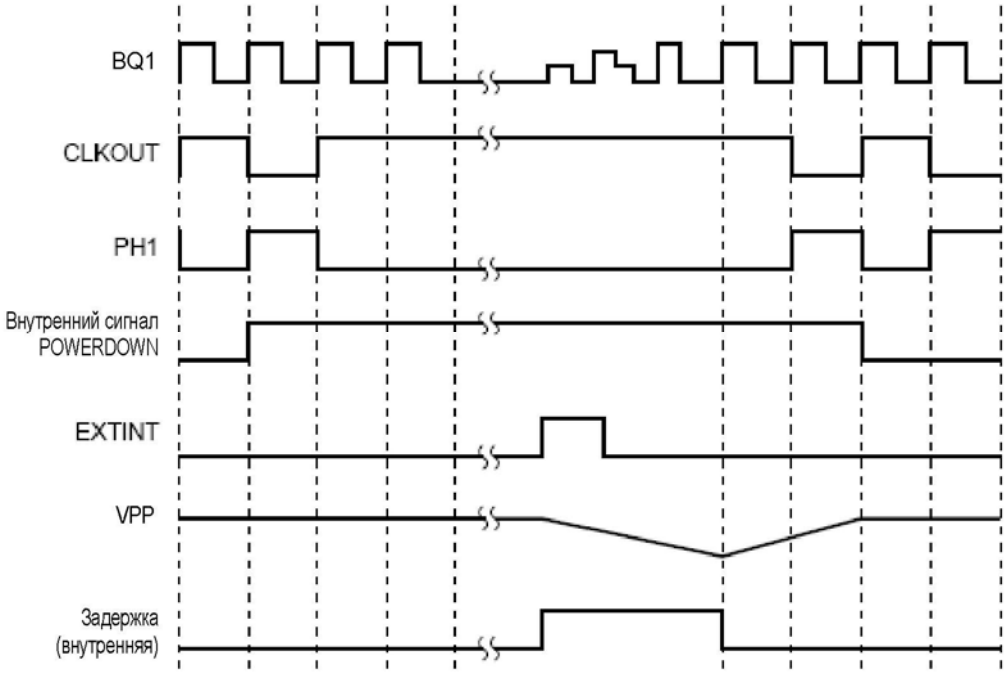


Рисунок 12.2 – Включение-выключение режима экономии мощности с использованием вывода внешнего прерывания

Когда используется сигнал внешнего прерывания для выхода из режима низкого потребления (POWERDOWN), рекомендуется использовать схему, показанную на рисунке 12.3 для вывода VPP. Разряд конденсатора вызывает задержку, которая позволяет осциллятору стабилизироваться прежде, чем будет разрешена синхронизация центрального процессора и периферийная синхронизация.

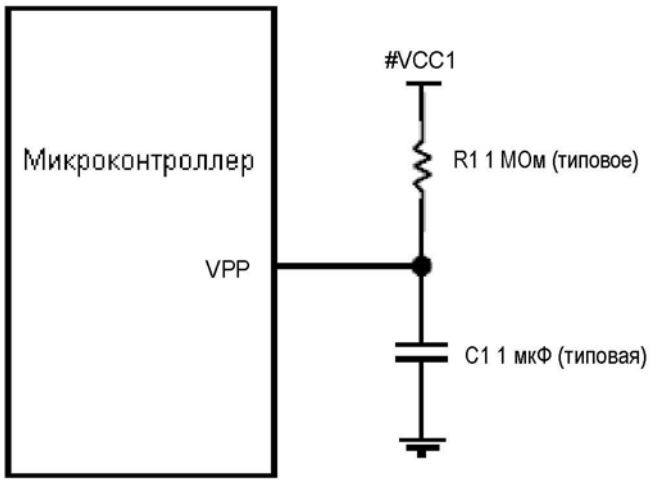


Рисунок 12.3 – Внешняя RC цепь на выводе VPP

В течение нормального режима работы (перед входом в режим POWERDOWN) на выводе VPP внутренне поддерживается до напряжение #VCC1. Когда устанавливается сигнал внешнего прерывания, работа генератора разрешена и включается слабое обнуление на VPP. Эта обнуляющая цепочка вызывает разряд внешнего конденсатора током порядка 200 мкА. При снижении напряжения вывода VPP ниже порогового напряжения (приблизительно 2,5 В), внутренняя синхронизация начинает работать и ЦП начинает выполнять коды команд.

В это время внутренний транзистор поддержки напряжения включается и быстро подтягивает высокий уровень до 3,5 В. После этого внутренняя поддержка становится неэффективной, и внешний резистор (R1) доводит напряжение до #VCC1 (см. время

восстановления на рисунке 12.4). Постоянная времени дает экспоненциально изменяющуюся кривую. Если R1 = 1 МОм и C1 = 1 мкФ, время восстановления будет одна секунда.

Подбор R1 и C1

Значения R1 и C1 (рисунок 12.3) не являются критическими. Необходимо выбрать компоненты, которые обеспечат достаточное время разряда, чтобы разрешить внутренней схеме генератора стабилизироваться. Поскольку многие факторы могут влиять на требования ко времени разряда, необходимо всегда определять время разряда для наихудших условий работы.

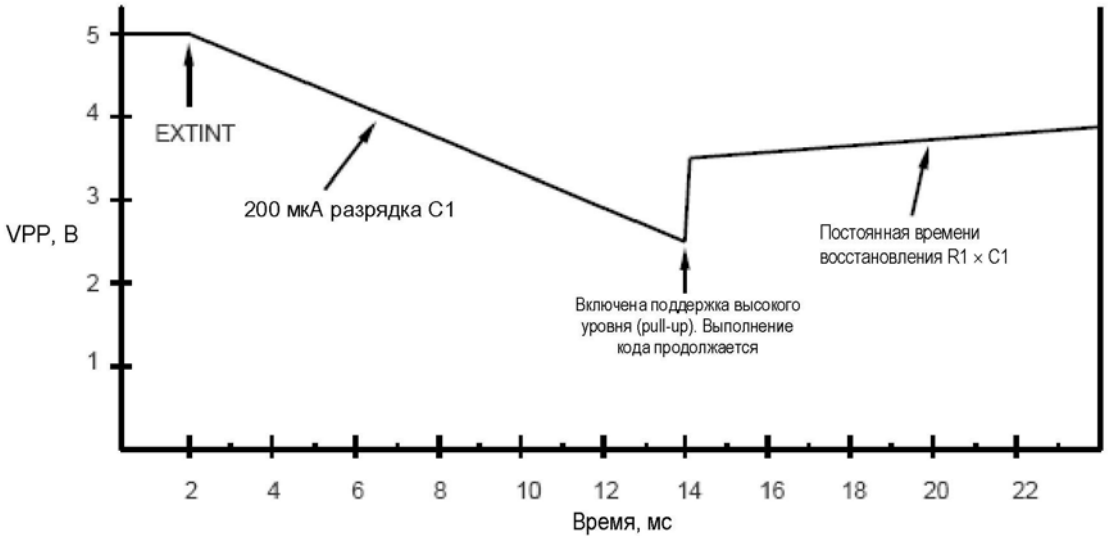


Рисунок 12.4 – Напряжение на выводе VPP при выходе из режима POWERDOWN

Необходимо выбрать резистор, который будет влиять на ток разряда. В большинстве случаев значения его сопротивления – между 200 кОм и 1 МОм.

При подборе конденсатора надо определить наихудшее время разряда, необходимое для генератора, чтобы стабилизировать частоту, затем использовать эту формулу, чтобы вычислить соответствующее значение для C1.

$C1 = (T_{DIS} \times I)/V_t$,

где C1 - номинал конденсатора в фарадах;

T_{DIS} - худшее время разряда в секундах;

I - ток разряда в амперах;

V_t - пороговое напряжение в вольтах.

Примечание – Если устройство повторно вошло в POWERDOWN и вышло, прежде чем C1 зарядился до #VCC1, будет требоваться меньше времени для уменьшения напряжения до порога. Поэтому устройству понадобится меньше времени для выхода из POWERDOWN.

Например, предположим, что генератору необходимо по крайней мере 12,5 мс для разряда (T_{DIS} =12,5 мс), V_t = 2,5 В и ток разряда I = 200 мкА. Номинал C1 около 1 мкФ:

$$C1 = \frac{(0{,}0125)(0{,}0002)}{2{,}5} = 1 мкФ$$

При использовании внешнего генератора значение C1 может быть очень маленьким, позволяя быстрый выход из POWERDOWN. Например, емкость в 100 пФ дает время разрядки 1,25 мкс.

$$T_{DIS} = \frac{C1 \times V_t}{I} = \frac{(1{,}0 \times 10^{-10})(2{,}5)}{0{,}0002} = 1{,}25 мкс$$

12.5 Режим внутрисхемной эмуляции (ONCE)

Режим внутрисхемной эмуляции (ONCE) изолирует устройство от других компонентов в системе, чтобы позволить проверять печатные платы или проводить отладку с поставочным эмулятором. В течение режима ONCE все выводы кроме BQ1, BQ2, #0V и #VCC1 имеют слабую поддержку до высокого или до низкого уровня. В течение режима ONCE вывод RESET# должен оставаться в состоянии высокого уровня, иначе устройство выйдет из режима ONCE и войдет в состояние сброса.

Удержание сигнала ONCE# в состоянии низкого уровня в течение положительного фронта сигнала RESET# вводит МК в режим ONCE. Чтобы предотвратить случайный вход в режим ONCE, рекомендуется конфигурирование этого вывода как выхода. Если необходимо сформировать этот вывод как вход, надо на нем держать высокий уровень в течение сброса, чтобы предотвратить случайный вход в режим ONCE.

Выход их режима ONCE происходит при установке сигнала RESET# и при плавающем уровне или слабой поддержке до высокого уровня на выводе ONCE#. Нормальная работа возобновится, когда сигнал RESET# перейдет в высокий уровень.

12.6 Зарезервированные тестовые режимы

Специальный вывод (P2.6) предназначен для тестового режима, который недоступен потребителю. Чтобы предотвратить случайный вход в тестовый режим, рекомендуется конфигурирование вывода P2.6 как выхода. Если надо сформировать этот вывод как вход, необходимо держать этот вывод в состоянии высокого уровня во время сброса.

13 Интерфейс внешней памяти

Микроконтроллеры 1874ВЕ86Т и 1874ВЕ16Т осуществляют поддержку интерфейса с различными типами устройств внешней памяти. Реализована программно-аппаратная поддержка конфигурации шины данных: 8-разрядная фиксированная, 16-разрядная фиксированная, 8-/16-разрядная динамически переключаемая; внутреннее управление длительностью циклов обращения к «медленной» внешней памяти; несколько режимов управления шиной адресов/данных.

В разделе описаны сигналы интерфейса внешней памяти, регистры управления интерфейса, байты конфигурации и режимы работы внешней шины.

13.1 Сигналы и регистры интерфейса внешней памяти

В таблице 13.1 приведены сигналы интерфейса внешней памяти, в таблице 13.2 приведены регистры интерфейса, в таблице 13.3 – установка регистров порта 5 для конфигурации сигналов интерфейса внешней памяти.

Таблица 13.1 – Сигналы интерфейса внешней памяти

Обозначение сигнала	Вывод порта	Тип	Описание
1	2	3	4
AD15:0	P4.7:0 P3.7:0	IO	Шина адресов/данных. Выводы конфигурируют мультиплексированную шину адресов/данных. В течение адресной фазы шинного цикла биты 0–15 адреса выводятся в шину и могут быть зафиксированы внешним интерфейсом сигналами ALE или ADV#. В течение фазы данных происходит передача 8- или 16-разрядных данных.
ADV#	P5.0	O	Адрес действителен. Активный сигнал низкого уровня устанавливается только в течение доступа к внешней памяти. ADV# указывает, что адресная информация выведена в шину адресов/данных. Сигнал сохраняется в низком уровне в продолжении шинного цикла и переходит в высокий уровень после его завершения. Внешний интерфейс может использовать его для формирования сигналов CS («выбор кристалла»), используемых микросхемами памяти.
ALE	P5.0	O	Разрешение записи адреса. Активный высокий уровень сигнала устанавливается в течение цикла обращения к внешней памяти. Сигналом ALE начинается внешний цикл шины и его активный уровень указывает на наличие адресной информации во внешней шине адресов/данных. Отличие ALE от ADV# в том, что его активный уровень не поддерживается до окончания цикла шины. Внешний интерфейс может использовать этот сигнал для демультиплексирования адреса из шины адресов/данных.

Продолжение таблицы 13.1

1	2	3	4
BHE#	P5.5	O	Выбор старшего байта. В течение 16-разрядного цикла шины активный низкий уровень сигнала устанавливается для чтения или записи слова и старшего байта. BHE# указывает, что действительна передача данных в старшем байте шины. Дешифрация BHE# совместно с AD0 определяет, какой конкретно байт слова передается по шине: BHE# AD0 Доступны байты 0 0 оба 0 1 старший 1 0 младший Выбор BHE# задается битом CCR0.2 = 1 в регистре конфигурации 0.
BW	P5.7	I	Разрядность шины данных. Регистры конфигурации 0 и 1 совместно с BW управляют шириной шины данных. CCR0.1 CCR1.2 BW фиксированная: 0 1 X 8-разрядная шина данных 1 0 X 16-разрядная шина данных, управляемая по BW: 1 1 H 16-разрядная шина данных 1 1 L 8-разрядная шина данных
CLKOUT	—	O	Системный тактовый сигнал. Выход внутреннего тактового генератора. Частота F_{CO} равна 1/2 входной частоты F_{BQ1}. Скважность сигнала равна 50 %.
EA#	—	I	Обращение к внешней памяти. Доступ к внутренней памяти программ разрешен установкой EA# в высокий уровень. Доступ к внешней памяти разрешен установкой EA# в низкий уровень. Если EA# устанавливается напряжением равным VPP (типовое +12,5 В) на фронте нарастания RESET#, микроконтроллер с OTPROM входит в режим программирования. Дальнейшее изменение уровня напряжения на входе EA# не оказывает влияние на режим. Если МК не имеет встроенного OTPROM, необходимо подключить вывод EA# к #VCC1.
INST	P5.1	O	Чтение команды. Активный высокий уровень сигнала указывает, что команда считана в МК из внешней памяти. Сигнал сохраняется в высоком уровне до окончания цикла чтения команды. При обращении к данным сигнал устанавливается в низкий уровень, включая чтение векторов прерываний и байтов конфигурации. Также сигнал всегда в низком уровне при чтении внутренней памяти.

Продолжение таблицы 13.1

1	2	3	4
RD#	P5.3	O	Чтение. Устанавливается активным уровнем только в цикле чтения внешней памяти.
READY	P5.6	I	Готовность. Активный сигнал совместно с регистрами конфигурации определяет число тактов задержки в шинном цикле. Регистры конфигурации задают число тактов задержки (0, 1, 2, 3, бесконечность). При низком READY в шинный цикл вводится запрограммированная задержка. Если READY переключается в высокий уровень раньше, чем достигается запрограммированное число тактов задержки, «затяжка» цикла прекращается.
WR#	P5.2	O	Запись. Активный низкий уровень указывает на запись во внешнюю память. Используется только в циклах записи. Выбор WR# задается битом CCR0.2 = 1.
WRH#	P5.5	O	Запись старшего байта. В течение 16-разрядного машинного цикла активный низкий уровень устанавливается для записи старшего байта или слова во внешнюю память. В 8-разрядном цикле устанавливается во всех операциях записи. Выбор WRH# задается битом CCR0.2 = 0.
WRL#	P5.2	O	Запись младшего байта. В течение 16-разрядного машинного цикла активный низкий уровень устанавливается для записи младшего байта или слова во внешнюю память. В 8-разрядном цикле устанавливается во всех операциях записи. Выбор WRL# задается битом CCR0.2 = 0.

Таблица 13.2 – Регистры интерфейса внешней памяти

Обозначение	Адрес	Описание
1	2	3
CCR0	2018H	Конфигурация кристалла 0. Управляет режимом пониженного потребления мощности, сигналами интерфейса шины, защитой внутренней памяти. Три бита регистра совместно с двумя битами CCR1 управляют «затяжкой» шинного цикла и разрядностью шины данных.
CCR1	201AH	Конфигурация кристалла 1. Разрешает сторожевой таймер, выбирает временной режим работы шины. Два бита совместно с тремя битами CCR0 управляют «затяжкой» шинного цикла и разрядностью шины.
P5_DIR	1FF3H	Управление порта 5. Каждый очищенный бит конфигурирует соответствующий вывод как комплементарный выход, установленный бит – как вход или выход с открытым стоком.

Продолжение таблицы 13.2

1	2	3
P5_MODE	1FF1H	Режим порта 5. Каждый установленный бит конфигурирует соответствующий вывод как сигнал специальной функции, очищенный бит – как стандартный вход-выход.
P5_PIN	1FF7H	Входы порта 5. Каждый бит регистра отражает текущее состояние соответствующего вывода, невзирая на конфигурацию.
P5_REG	1FF5H	Выходы порта 5. При выводе данных записанные значения разрядов P5_REG выдаются на соответствующие выводы порта. При конфигурации выводов как стандартные входы-выходы (P5_MODE.y = 0) записанные данные появляются на выходах непосредственно. При конфигурации выводов как сигналов специальных функций (P5_MODE.y = 1) управление выводами осуществляется соответствующей внутренней или внешней периферией МК. Записываемые в P5_REG значения не будут влиять на состояние выводов, пока они не переключатся на функции стандартных входов-выходов. Это позволяет программному обеспечению конфигурировать выводы как стандартные, инициализировать или изменять значения на выводах, затем конфигурировать их как сигналы специальных функций, что позволяет производить коррекцию ошибок, обработку исключительных ситуаций и т. д., не нарушая режим работы периферийных устройств.

Таблица 13.3 – Установка регистров для конфигурации сигналов интерфейса внешней памяти

Вывод порта	Обозначение сигнала интерфейса	Тип сигнала	Значение установки регистров
P5.0	ALE/ADV#	выход	P5_DIR = 110X0000B
P5.1	INST	выход	P5_MODE = 111X 1111B
P5.2	WR# / WRL#*	выход	P5_REG = 11XX XXXXB
P5.3	RD#	выход	
P5.5	BHE# / WRH#*	выход	
P5.6	READY	вход	
P5.7	BW	вход	
* Установка в регистре конфигурации CCR0 бита CCR0.2 = 1 выбирает функции BHE# и WR#, а CCR0.2 = 0 выбирает функции WRL# и WRH#.			

13.2 Регистры и байты конфигурации кристалла

Настройка параметров операций МК и внешних шинных циклов осуществляется двумя регистрами конфигурации. Регистры программно недоступны и их загрузка осуществляется двумя байтами конфигурации, размещаемыми во внутренней энергонезависимой или внешней памяти по адресам 2018H (CCB0) и 201AH (CCB1). Загрузка осуществляется при выполнении микроконтроллером выхода из сброса (каждый раз). Значения бит регистров приведены на рисунках 13.1 и 13.2.

CCR0 нет прямого доступа

Регистр конфигурации 0 (CCR0) управляет режимом пониженного потребления мощности, сигналами управления шины, защитой встроенной памяти, совместно с CCR1 – «затягиванием» шинного цикла и разрядностью шины данных.

7 0

LOC1	LOC0	IRC1	IRC0	ALE	WR	BW0	PD

Номер бита	Мнемоника	Функция
1	2	3
7, 6	LOC1,0	Биты защиты. Управляют доступом чтения или записи OTPROM в режиме нормальных операций LOC1 LOC0 Доступ OTPROM 0 0 запрет чтения и записи 0 1 запрет чтения 1 0 запрет записи 1 1 нет защиты
5, 4	IRC1,0	Управление «затягиванием» шинного цикла. Два бита совместно с IRC1.1 (CCR1) и READY управляют «затягиванием» шинного цикла. Если READY удерживается в низком уровне, цикл шины «затягивается» в соответствии с запрограммированным числом тактов (машинных). Если READY переключается в высокий уровень раньше, чем достигнуто запрограммированное число тактов, «затягивание» цикла прекращается. При выборе варианта неопределённого времени «затягивания» P5.6 конфигурируется как сигнал READY. Рекомендуется дополнить аппаратное обеспечение счётчиком необходимого числа тактов»затягивания» и установки READY высоким уровнем, т. к. дефектная ИМС памяти может занимать шину неопределённое время. IRC2 IRC1 IRC0 «Затягивание» времени цикла шины 0 0 0 нет 0 X 1 запрещённые 0 1 X комбинации 1 0 0 1 такт 1 0 1 2 такта 1 1 0 3 такта 1 1 1 неопределённое
3	ALE	Разрешение записи адреса и запись данных. Определяют комбинацию сигналов управления шиной при внешних циклах чтения и записи. ALE WR режимы 0 0 стробирование действительного адреса и записи (ADV#, RD#, WRL#, WRH#) 0 1 стробирование действительного адреса (ADV#, RD#, WR#, BHE#) 1 0 стробирование записи (ADV#, RD#, WRL#, WRH#) 1 1 стандартный (ALE, RD#, WR#, BHE#)
2	WR	

1	2	3
1	BW0	Управление разрядностью шины. Осуществляется совместно с BW1 (CCR1.2) BW1 BW0 разрядность 0 0 запрещённая комбинация 0 1 только 16-разрядная 1 0 только 8-разрядная 1 1 управление по BW
0	PD	Выбор режима пониженного потребления мощности. Разрешает команде IDLPD #2 перевести МК в режим пониженного потребления мощности. PD = 0 запрет пониженного потребления мощности; PD = 1 разрешение пониженного потребления мощности.

Рисунок 13.1 – Регистр конфигурации кристалла 0 (CCR0)

CCR1 нет прямого доступа

Регистр конфигурации 1 (CCR1) разрешает сторожевой таймер, выбирает временной режим шины, совместно с CCR1 управляет «затягиванием» шинного цикла и разрядностью шины данных.

7 0

1	1	0	1	WDE	BW1	IRC2	0

Номер бита	Мнемоника	Функция
1	2	3
7, 6	1	Записываются «1»
5	0	Записывается «0»
4	1	Записывается «1»
3	WDE	Разрешение сторожевого таймера. 0 – разрешён всегда, 1 – запрещён до очистки регистра WATCHDOG
2	BW1	Управление разрядностью шины. Осуществляется совместно с BW0 (CCR0.1) BW1 BW0 разрядность 0 0 запрещённая комбинация 0 1 только 16-разрядная 1 0 только 8-разрядная 1 1 управление по BW

1	2	3
1	IRC2	Управление затягиванием шинного цикла. Бит совместно с IRC0, IRC1 (CCR0.4, CCR0.5) и READY управляют затягиванием шинного цикла. Если READY удерживается в низком уровне, цикл шины затягивается в соответствии с запрограммированным числом тактов (машинных). Если READY переключается в высокий уровень раньше, чем достигнуто запрограммированное число тактов, затягивание цикла прекращается. При выборе варианта неопределённого времени затягивания P5.6 конфигурируется как сигнал READY. Рекомендуется дополнить аппаратное обеспечение счётчиком необходимого числа тактов «затягивания» и установки READY высоким уровнем, т. к. дефектная ИМС памяти может занимать шину неопределённое время. IRC2 IRC1 IRC0 «Затягивание» времени цикла шины 0 0 0 нет 0 X 1 запрещённые 0 1 X комбинации 1 0 0 1 такт 1 0 1 2 такта 1 1 0 3 такта 1 1 1 неопределённое
0	0	Записывается «0»

Рисунок 13.2 – Регистр конфигурации кристалла 1 (CCR1)

13.3 Мультиплексирование и разрядность шины

Внешняя шина имеет два режима работы: 16-разрядная мультиплексированная шина адресов/данных, мультиплексированная 16-разрядная шина адреса / 8- разрядная шина данных (см. рисунок 13.3).

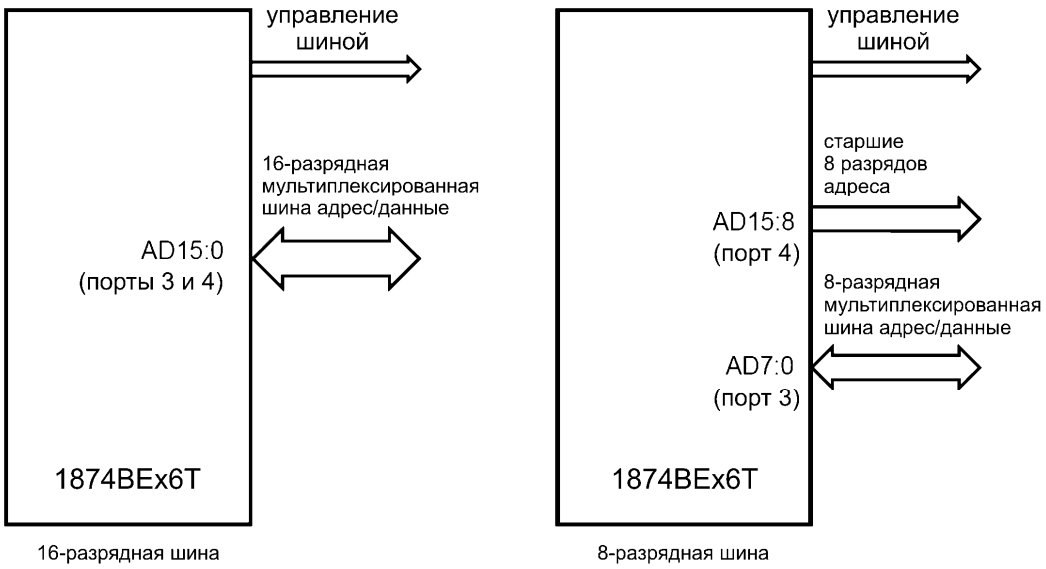


Рисунок 13.3 – Конфигурация шины адресов/данных

После сброса, но до записи ССВ, микроконтроллер конфигурируется в 8-разрядный режим шины независимо от состояния входа BW. Старшие линии шины

(AD15:8) управляются маломощным выходным формирователем сигналов в течение циклов CCB0 и CCB1. Для предотвращения конфликтных ситуаций в шине не следует подключать к этим линиям нагрузки по #VCC и #0V. Старшие байты слов CCB (ячейки 2019H и 201BH) должны быть загружены величиной 20H. При чтении из памяти значения 20H в старшем байте конфликтов в шине не произойдёт. После загрузки CCB в CCR значения BW0 и BW1 будут определять конфигурацию шины (см. рисунки 13.1, 13.2).

При установленных BW0 и BW1 разрядностью шины управляет сигнал BW: высокий уровень сигнала выбирает 16-разрядную шину, низкий уровень – 8-разрядную шину. Сигнал BW действителен после появления адреса в шине (см. рисунок 13.4)

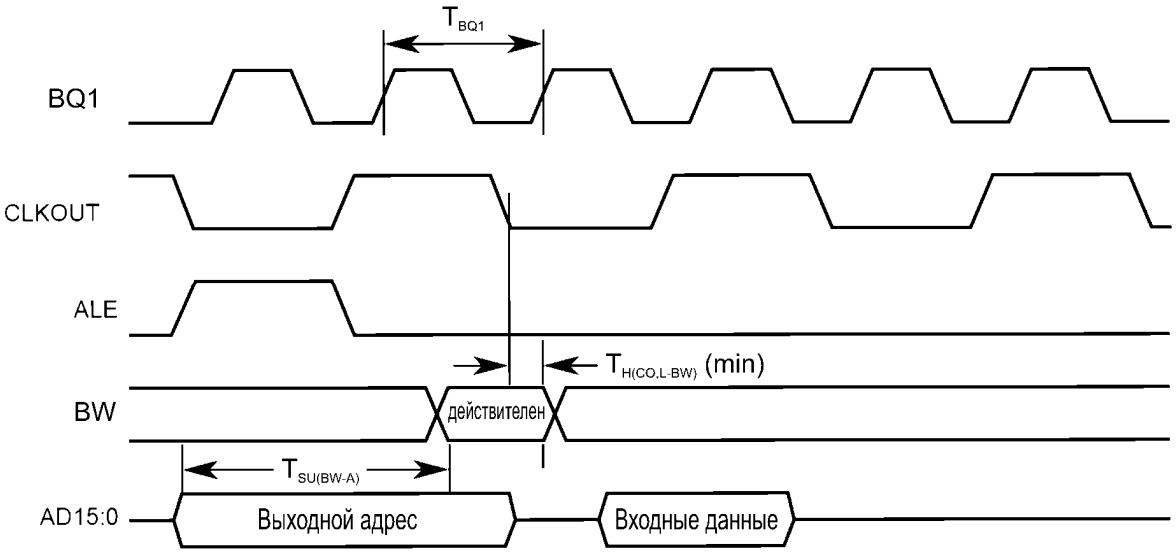


Рисунок 13.4 – Временная диаграмма сигнала BW

Таблица 13.4 – Обозначения временных параметров сигнала BW

Обозначение	Определение
$T_{SU(BW-A)}$	Время установки сигналов адреса относительно сигнала BW
$T_{H(CO,L-BW)}$	Время сохранения сигнала BW относительно низкого уровня сигнала CO (минимальное)
T_{BQ1}	1/ F_{BQ1}. Период следования импульсов входного тактового сигнала.

Временные диаграммы 16-разрядной шины

На рисунке 13.5 приведены идеализированные временные диаграммы сигналов внешней шины в 16-разрядной конфигурации.

Передний фронт сигнала ALE устанавливается перед выдачей микроконтроллером адреса в шину (AD15:0). МК удерживает адрес до переключения ALE в низкий уровень (минимум). Сигнал ALE используется схемами «защёлки» адреса для его записи из шины и хранения до вывода данных в шину.

В 16-разрядном цикле чтения микроконтроллер устанавливает шину в третье состояние и выдает низкий уровень сигнала RD# для чтения данных. Внешняя память должна вывести данные в шину раньше переключения RD# в высокий уровень. При установке микроконтроллером сигнала INST производится чтение команд.

В 16-разрядном цикле записи микроконтроллер устанавливает низкий уровень сигнала WR# и выводит данные в шину. Задний фронт сигнала индицирует действительность данных. В этот момент внешняя система должна «защёлкнуть» данные.

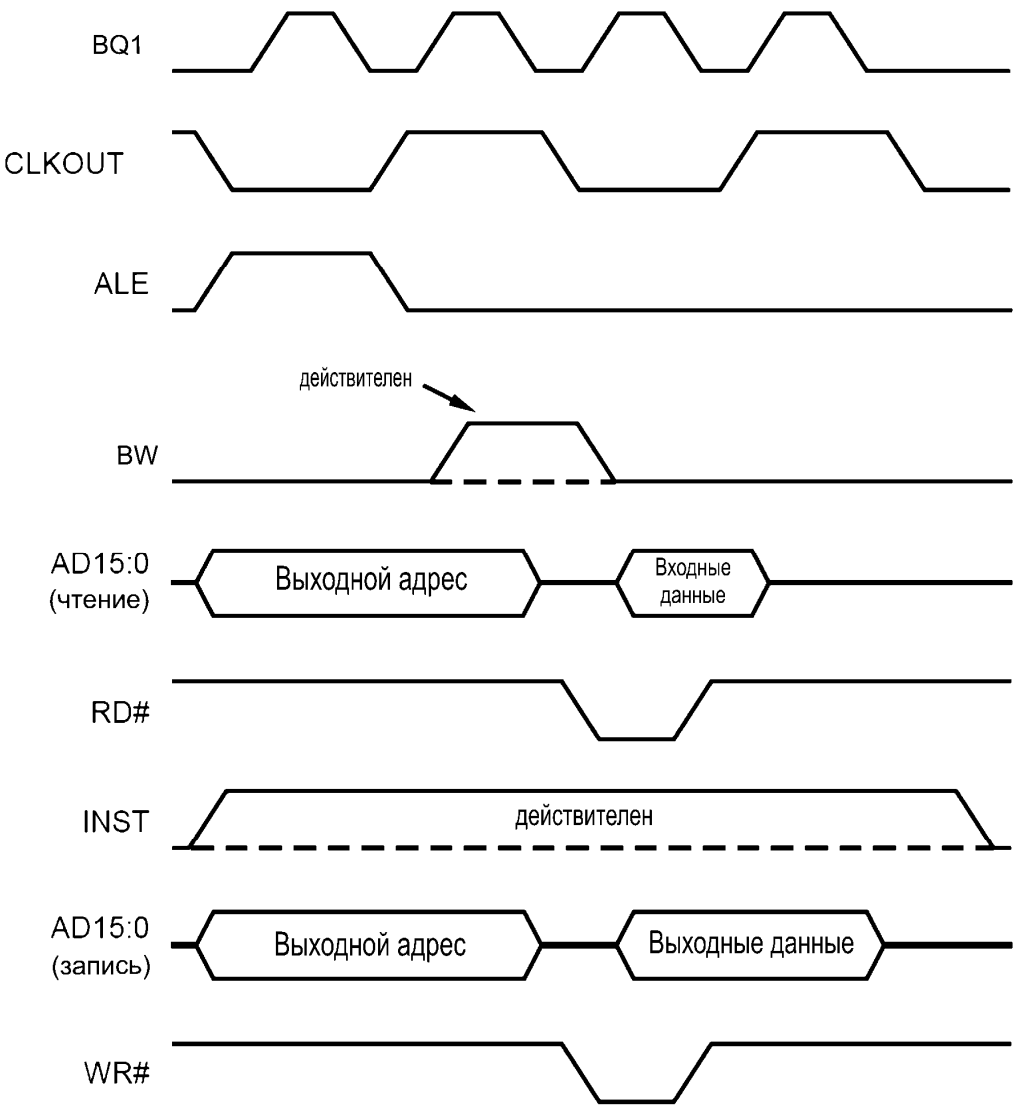


Рисунок 13.5 – Временные диаграммы сигналов 16-разрядной шины

Временные диаграммы 8-разрядной шины

В режиме 8-разрядной шины выводы AD7:0 формируют мультиплексированную шину младшего байта адреса/данных. Выводы AD15:8 не мультиплексированы; в течение цикла на них устанавливается и удерживается старший байт адреса. На рисунке 13.6 приведены идеализированные временные диаграммы сигналов внешней шины в 8-разрядной конфигурации.

Для чтения или записи байта требуется один цикл шины, для слова – два цикла. Первый цикл – доступ к младшему байту, второй цикл – к старшему байту. Временные параметры 16- и 8-разрядных циклов идентичны. ALE используется для демультиплексирования младшего байта адреса.

В цикле 8-разрядного чтения слова сигнал RD# устанавливается дважды: сначала читается младший байт, затем – старший. В цикле 8-разрядной записи слова сигнал WR# также устанавливается дважды: сначала из микроконтроллера выводится младший байт, затем – старший.

Входные и выходные данные должны быть стабильны во время переключения сигналов RD# и WR# соответственно из низкого уровня в высокий.

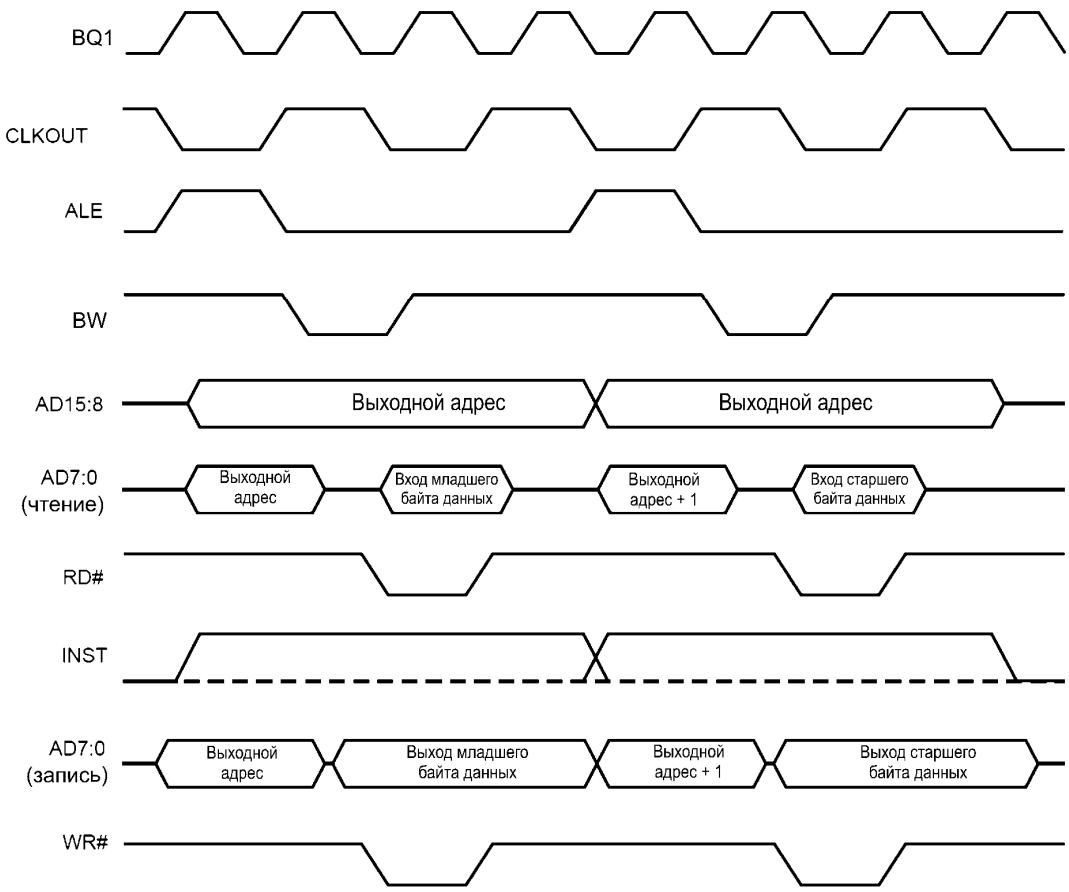


Рисунок 13.6 – Временные диаграммы сигналов 8-разрядной шины

13.4 Управление длительностью шинного цикла сигналом READY

Внешние устройства могут использовать вход READY для управления длительностью шинного цикла. Когда адрес выведен в шину, а устройство внешней памяти не готово для доступа, оно должно установить сигнал READY низким уровнем и удерживать его до готовности завершить операцию. В ответ микроконтроллер увеличивает длительность цикла, добавляя состояние ожидания ($2T_{BQ1}$) до тех пор, пока READY не переключится в высокий уровень. После сброса и до приёма CCB1 микроконтроллер всегда добавляет три состояния ожидания в шинный цикл. Пока P5.6 сконфигурирован как вход READY, биты IRC2:0 регистров конфигурации управляют длительностью цикла, а при установке неопределённой длительности (IRC2:0 = 1) управление осуществляется внешним аппаратным обеспечением. На рисунке 13.7 приведены временные диаграммы для сигнала READY.

В таблице 13.5 приведены определения временных параметров сигнала READY.

Таблица 13.5 – Обозначения временных параметров сигнала READY

Обозначение	Определение
$T_{SU(RDY,L-A)}$	Время установки сигнала адреса относительно установления сигнала READY в низкий уровень.
$T_{H(CO,L-RDY,L)}$	Время удержания сигнала READY относительно системного тактового сигнала (минимальное, максимальное).

Параметр $T_{SU(RDY,L-A)}$ определяет время, необходимое внешней аппаратуре для дешифрации адреса и формирования низкого уровня сигнала READY. Параметр $T_{H(CO,L-RDY,L)}$ определяет минимально необходимое время фиксации микроконтроллером уровня сигнала READY. Норма на максимальное значение параметра не устанавливается.

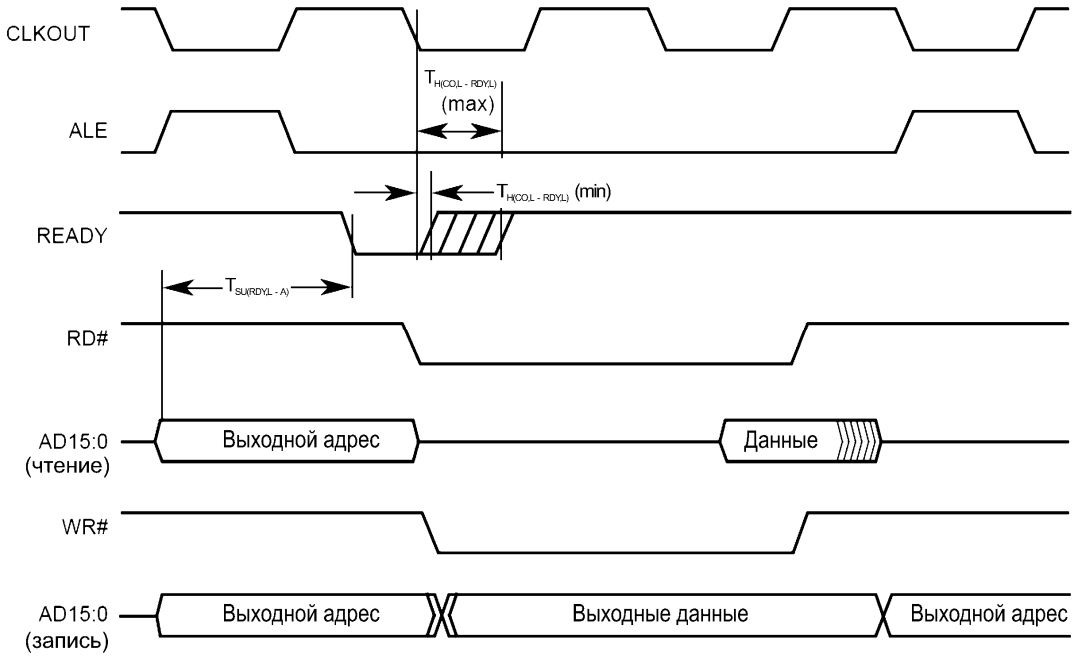


Рисунок 13.7 – Временные диаграммы сигнала READY

13.5 Режимы управления шиной

Биты ALE и WR (CCR0.3 и CCR0.2) определяют комбинацию сигналов управления шиной, формируемую во внешних циклах чтения и записи. В таблице 13.6 приведены устанавливаемые значения битов CCR0.3 и CCR0.2 для четырёх возможных режимов управления шиной.

Таблица 13.6 – Режимы управления шиной

Режим	Сигналы управления	CCR0.3 (ALE)	CCR0.2 (WR)
стандартный	ALE, RD#, WR#, BHE#	1	1
стробирование записи	ALE, RD#, WRL#, WRH#	1	0
стробирование действительного адреса	ADV#, RD#, WR#, BHE#	0	1
стробирование действительного адреса и записи	ADV#, RD#, WRL#, WRH#	0	0

Стандартный режим управления шиной

В стандартном режиме управления шиной микроконтроллер формирует набор сигналов, приведённый в таблице 13.6 и на рисунке 13.8. ALE устанавливается при выводе адреса и может использоваться для фиксации адреса внешними схемами. RD# устанавливается в циклах чтения внешней памяти, WR# – в циклах записи. При установленном BHE# выбирается банк памяти, адресуемый старшим байтом шины данных.

В конфигурации 16-разрядной шины микроконтроллер может формировать отдельные сигналы записи младших и старших байтов для байтовых операций записи. На рисунке 13.9 приведена упрощённая схема дешифрации сигналов WRH# и WRL#. Для циклов чтения подобная пара сигналов не нужна, т. к. при чтении байта из 16-разрядной шины микроконтроллер игнорирует незатребованный байт.

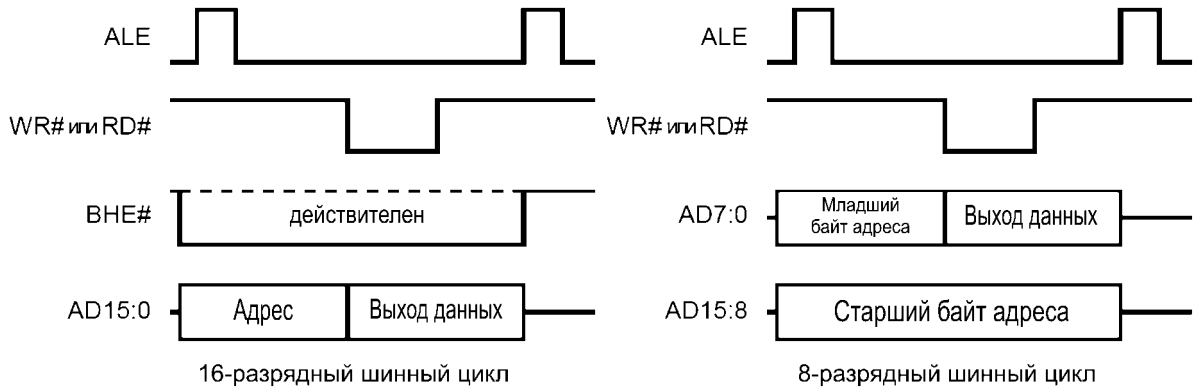


Рисунок 13.8 – Стандартный режим управления шиной

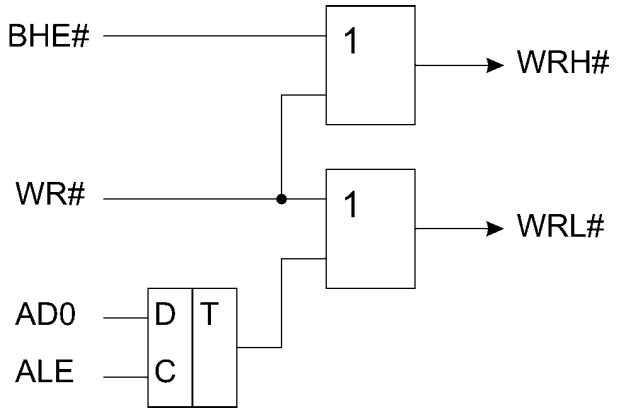


Рисунок 13.9 – Схема дешифрации сигналов WRH# и WRL#

На рисунке 13.10 приведён вариант 8-разрядной системы, использующий схемы ЭППЗУ (типа «флэш») и ОЗУ для младшего и старшего блоков памяти. Старший разряд адреса A15 используется как сигнал «выбор кристалла», сигнал ALE для фиксации адреса.

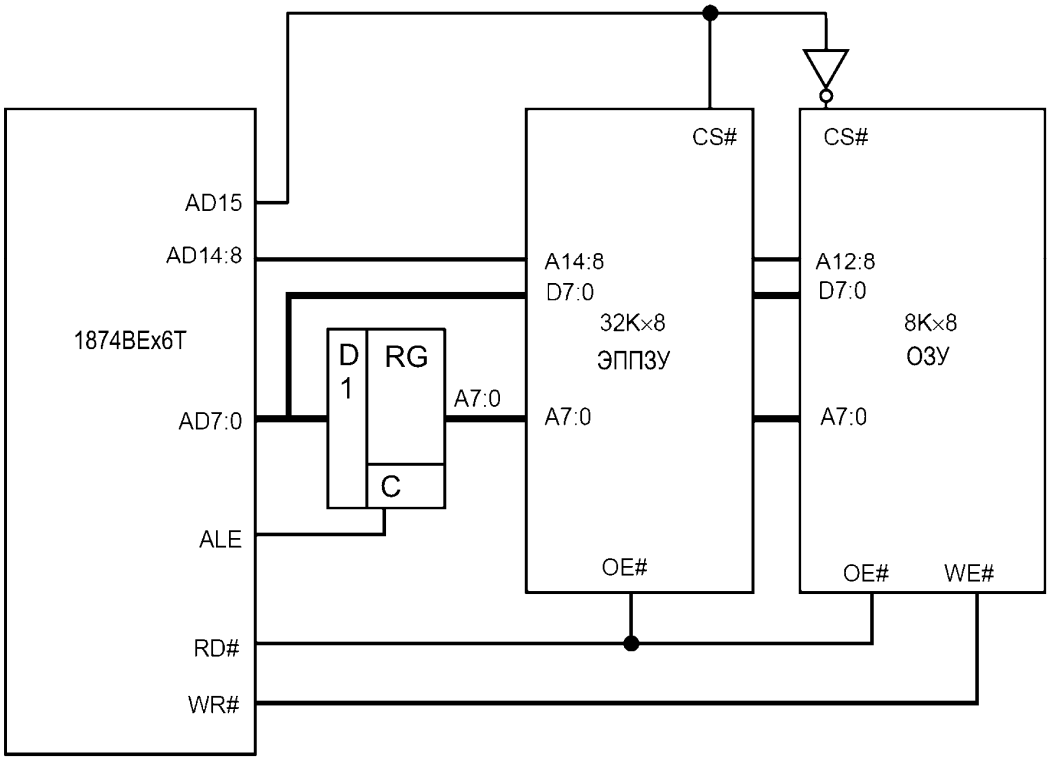


Рисунок 13.10 – 8-разрядная система с ЭППЗУ и ОЗУ

На рисунке 13.11 приведён вариант 16-разрядной системы с динамически переключаемой разрядностью шины (биты BW0 и BW1 в CCR установлены). Программы выполняются из двух СППЗУ, а данные хранятся в 8-разрядном ОЗУ (старший банк памяти), выбираемом по A15 = 1; при этом так же выбирается 8-разрядная шина данных (BW устанавливается в низкий уровень).

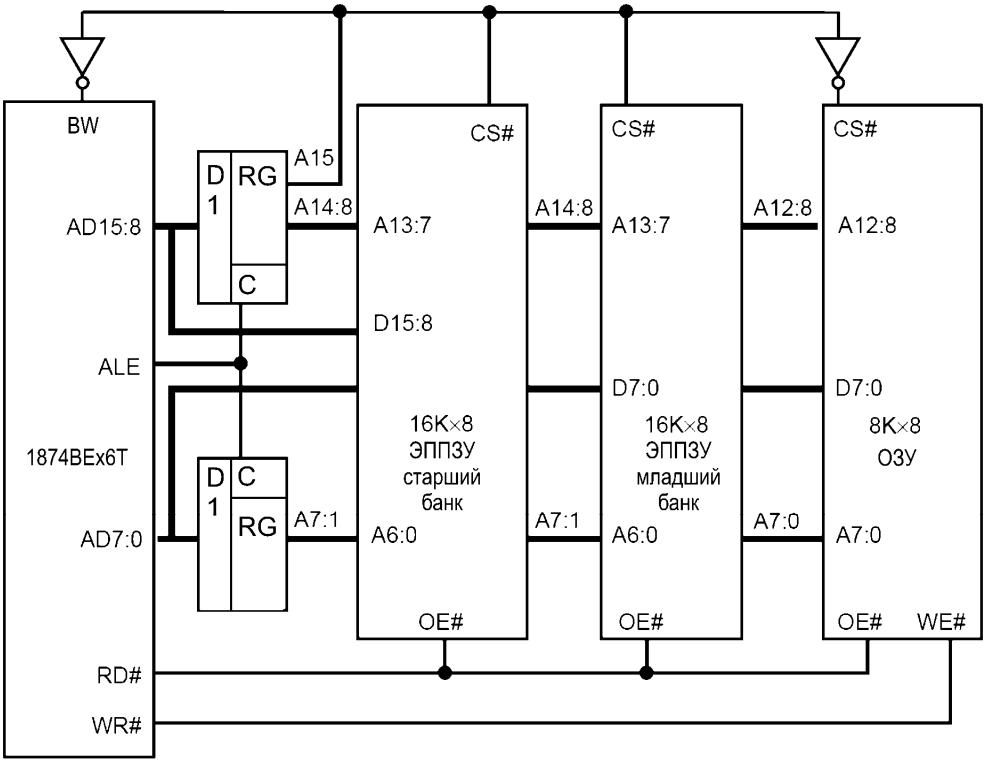


Рисунок 13.11 – 16-разрядная система с динамически переключаемой разрядностью шины

Режим стробирования записи

При выборе режима стробирования записи микроконтроллер формирует вместо сигналов WR# и BHE# сигналы WRL# и WRH# (внешний дешифратор по типу, приведённому на рисунке 13.9, не используется). WRL# устанавливается для записи младших байтов (чётные адреса) и слов. WRH# устанавливается для записи старших байтов (нечётные адреса) и слов. В 8-разрядном режиме шины сигналы WRL# и WRH# устанавливаются и для чётных и для нечётных адресов. На рисунке 13.12 приведены временные диаграммы сигналов режима стробирования записи.

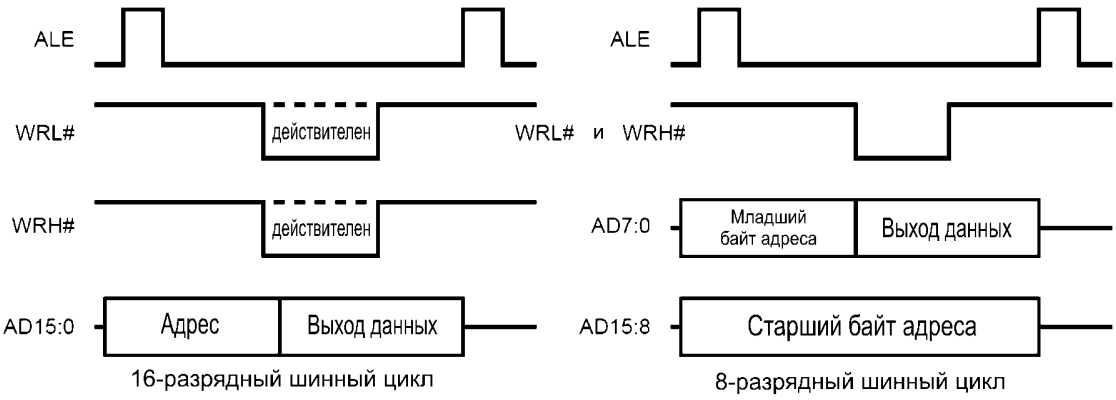


Рисунок 13.12 – Режим стробирования записи

На рисунке 13.13 приведён вариант 16-разрядной системы с двумя устройствами СППЗУ и двумя устройствами ОЗУ, сконфигурированными для использования режима стробирования записи. Сигнал ALE фиксирует адрес. AD15 является сигналом «выбор кристалла» для микросхем памяти. WRL# устанавливается для записи младшего байта и слова, WRH# – для записи старшего байта и слова. A0 не используется (нет внешней дешифрации).

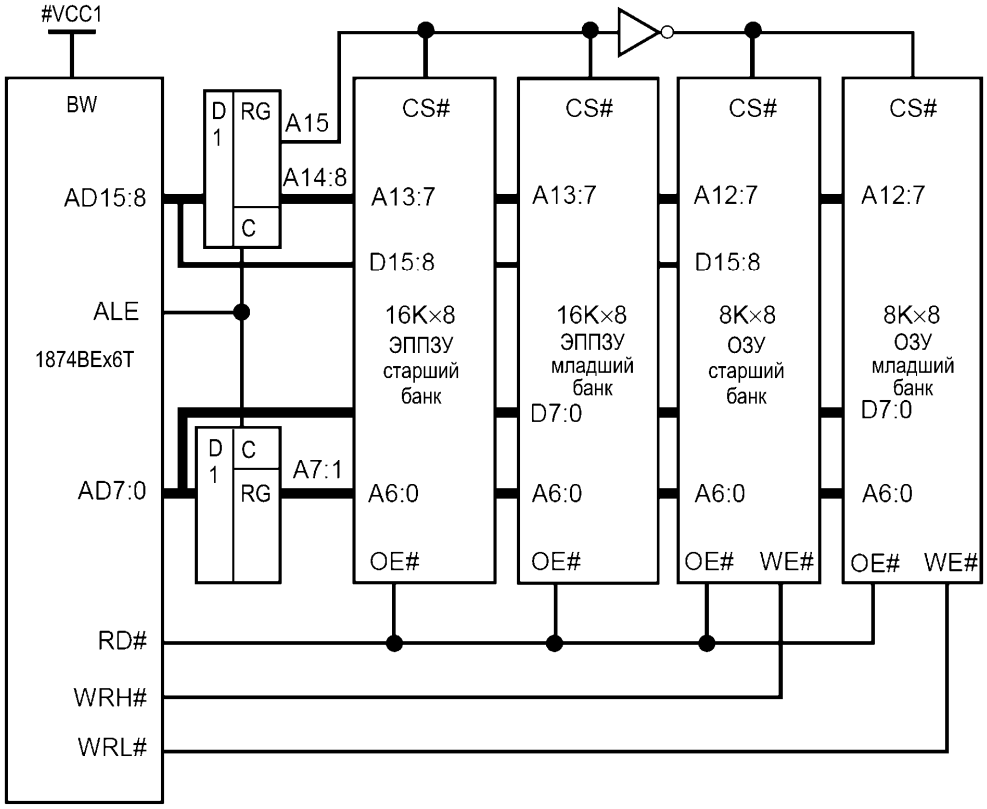


Рисунок 13.13 – 16-разрядная система с записью в 8-разрядное ОЗУ

Режим стробирования действительного адреса

При выборе режима стробирования действительного адреса микроконтроллер формирует сигнал ADV# вместо ALE. ADV# устанавливается после вывода адреса в шину. Сигнал может использоваться для фиксации адреса и разрешать работу микросхем внешней памяти. Разница между сигналами ALE и ADV# в том, что ADV# устанавливается на полный цикл шины, а не только для фиксации адреса. Временные диаграммы режима стробирования действительного адреса приведены на рисунке 13.14, а на рисунке 13.15 показана разница временных диаграмм ALE и ADV# для одного цикла чтения или записи.

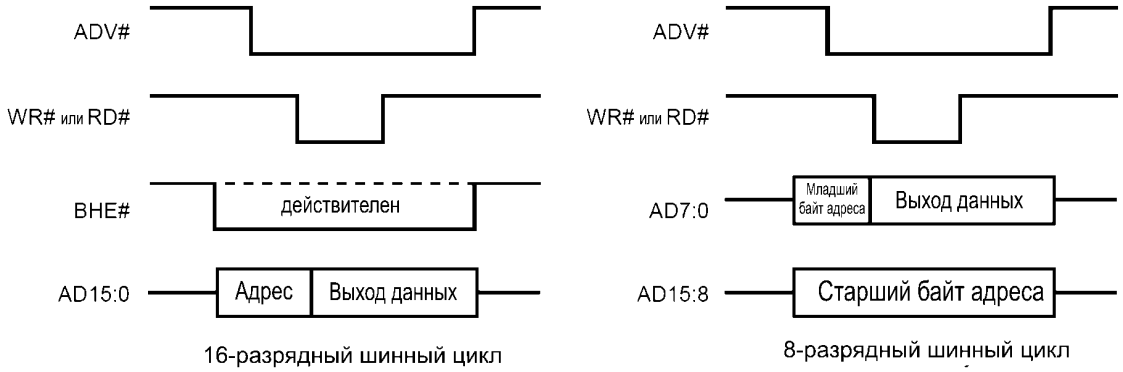


Рисунок 13.14 – Режим стробирования действительного адреса

Если циклы шины следуют один за другим, действие ADV# будет выглядеть идентично действию ALE. Различие заметно при наличии интервалов холостого хода шины. Т. к. ADV# поддерживается в этих интервалах в высоком уровне, внешняя память запрещена и снижается потребление мощности в системе.

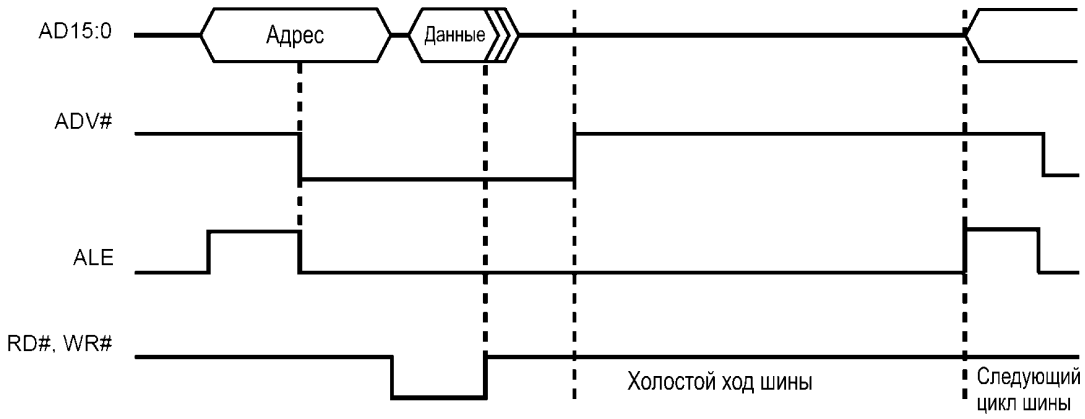


Рисунок 13.15 – Сравнение шинных циклов ALE и ADV#

На рисунках 13.16 и 13.17 показаны упрощённые схемы использования режима стробирования действительного адреса. На рисунке 13.16 показана 8-разрядная система с одной микросхемой ЭППЗУ (типа «флэш»), использующая ADV# для фиксации адреса и в качестве сигнала «выбор кристалла». На рисунке 13.17 показана 16-разрядная система с микросхемами СППЗУ и аналогичным использованием сигнала ADV#.

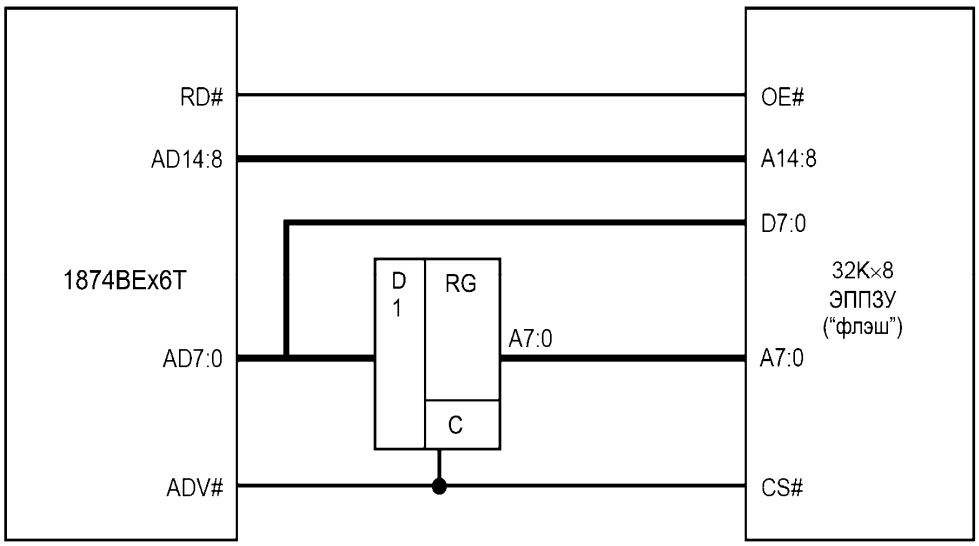


Рисунок 13.16 – 8-разрядная система с ЭППЗУ («флэш»)

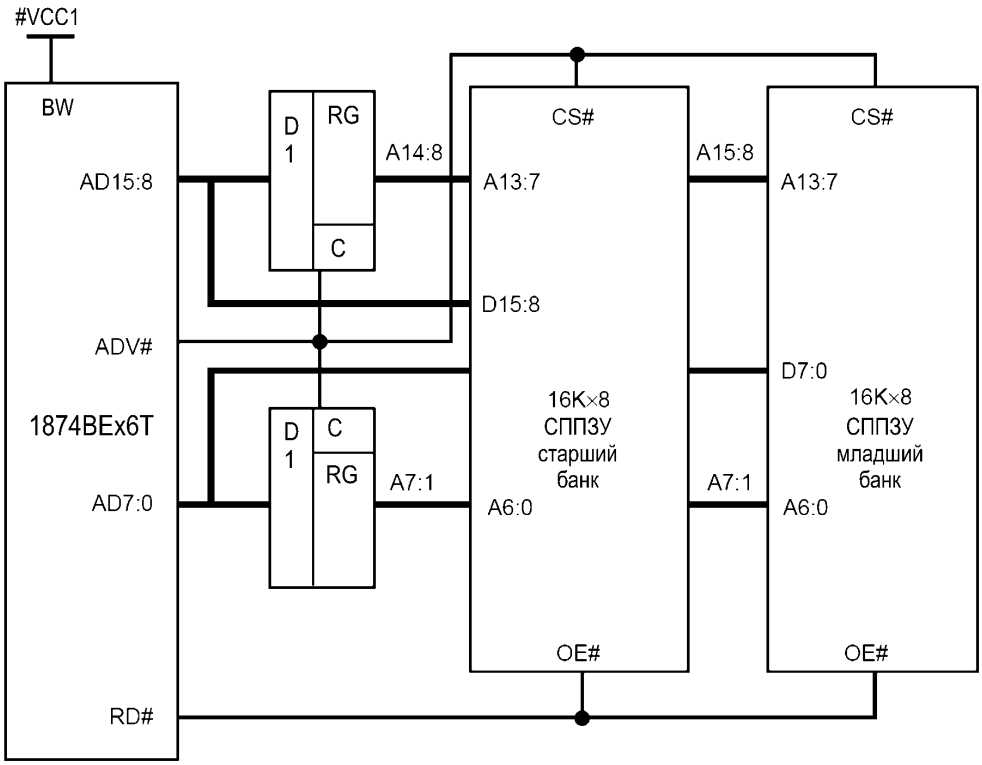


Рисунок 13.17 – 16-разрядная система с СППЗУ

Режим стробирования действительного адреса и записи

В данном режиме микроконтроллер формирует сигналы ADV#, RD#, WRL#, WRH#. Режим обычно используется в простых системах с 16-разрядной шиной. На рисунке 13.18 приведены временные диаграммы сигналов режима стробирования действительного адреса и записи. Сигнал RD# (не показан) аналогичен WRL# и WRH#. На рисунке 13.19 приведен пример системы, использующей режим стробирования действительного адреса и записи для доступа к 8-разрядному ОЗУ с 16-разрядной шиной данных.

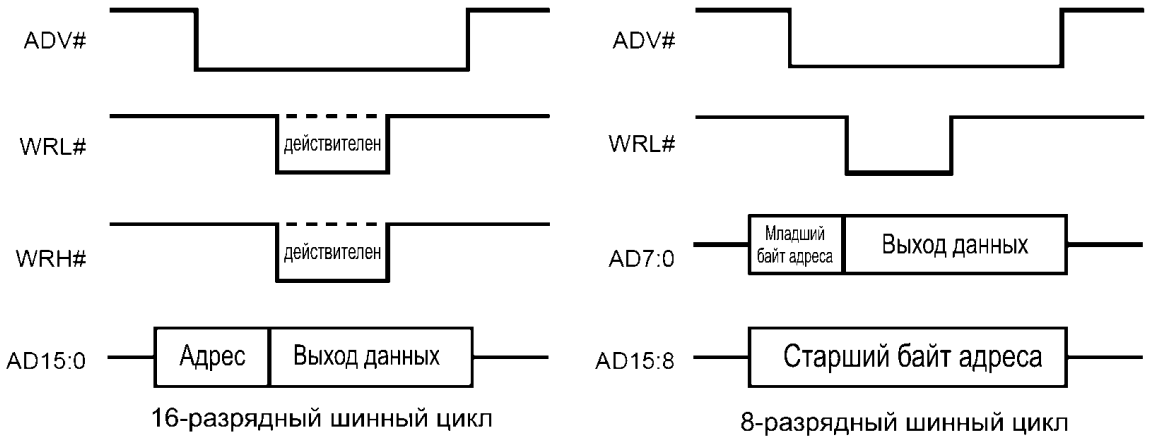


Рисунок 13.18 – Временные диаграммы сигналов в режиме стробирования действительного адреса и записи

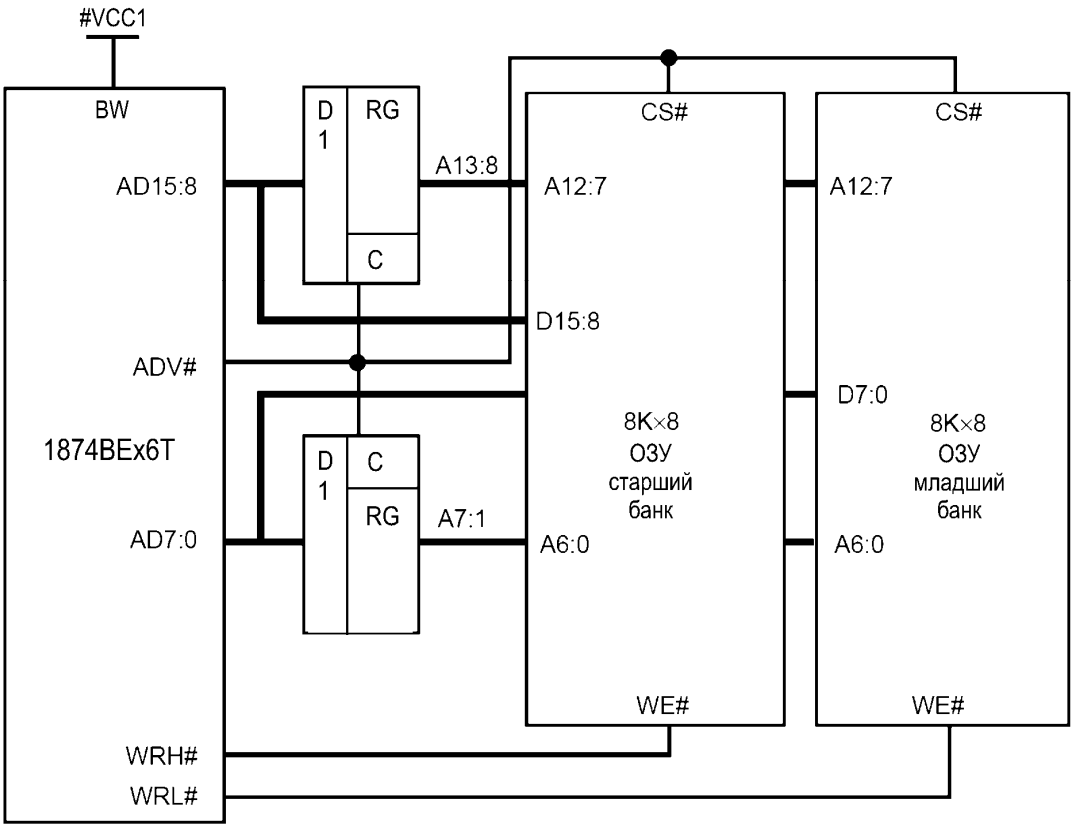


Рисунок 13.19 – 16-разрядная система с 8-разрядными микросхемами ОЗУ

14 Программирование постоянной памяти

Контроллер содержит 16 Кбайт однократно программируемого запоминающего устройства (OTPROM). OTPROM подобен EPROM, но не имеет окна для стирания в корпусе. Этот раздел описывает процедуры и принципы, которые могут помочь программировать устройство. В данном разделе содержится следующая информация:

- краткий обзор режимов программирования;
- карта распределения памяти OTPROM;
- защита памяти;
- программирование ширины программирующего импульса;
- модифицированный алгоритм быстрого программирования;
- выводы в режиме программирования;
- введение устройства в режимы программирования;
- программирование в режиме «SLAVE»;
- автопрограммирование;
- run-time программирование.

14.1 Режимы программирования

Когда выбирается режим программирования, устройство отзывается на него выполнением алгоритма, который находится в области тестового ПЗУ. Устройство поддерживает следующие режимы программирования:

Режим программирования SLAVE. В этом режиме для программирования OTPROM необходимо использовать специальное устройство – программатор. Возможно программировать или верифицировать отдельные слова или наборы слов в OTPROM.

Режим автопрограммирования. В этом режиме OTPROM программируется из внешнего EPROM без использования программатора. Используя этот режим, невозможно запрограммировать биты UPROM и PCCB (для их записи необходимо использовать SLAVE режим). После программирования возможно использовать режим ROM-dump для записи всей OTPROM во внешнюю память для последующей верификации. Режим автопрограммирования обычно используется для программирования небольшого количества микроконтроллеров после того, как закончены этапы разработки и тестирования программного обеспечения (ПО).

Также возможно программирование ячеек OTPROM без ввода контроллера в режим программирования. Такой режим называется run-time режимом. При этом ПО само контролирует количество и длительность программирующих импульсов.

14.2 Карта распределения памяти OTPROM

OTPROM содержит ячейки памяти специального назначения и память программ (таблица 14.1). 128 байт памяти специального назначения используются для хранения векторов прерываний, байтов конфигурации кристалла (CCB) и ключа безопасности. Также некоторые ячейки зарезервированы для тестирования. Необходимо записывать значения (20H и FFH) в зарезервированные ячейки согласно таблице 14.1. Остальная часть OTPROM может быть использована для хранения кода.

Таблица 14.1 – Карта распределения памяти

Диапазон адресов (HEX)	Описание
1	2
5FFF 2080	Память программ
207F 205E	Зарезервированы (должны содержать FFH)

Продолжение таблицы 14.1

1	2
205D 2040	PTS векторы
203F 2030	Старшие векторы прерываний
202F 2020	Ключ безопасности
201F 201C	Зарезервированы (должны содержать FFH)
201B	Зарезервирован (должен содержать 20H)
201A	CCB1
2019	Зарезервирован (должен содержать 20H)
2018	CCB0
2017 2014	Зарезервированы (должны содержать FFH)
2013 2000	Младшие векторы прерываний

14.3 Защита памяти

Существуют несколько механизмов, обеспечивающих ограничение доступа к внутренней и внешней памяти. Комбинирование битов защиты чтения и записи в байте конфигурации кристалла (CCR0) с ключом безопасности обеспечивают множество разнообразных уровней защиты памяти. Два UPROM бита запрещают выборку команд и данных из внешней памяти (см. рисунок 14.1).

Доступ к внутренней памяти

Биты защиты в регистре конфигурации кристалла (CCR0) управляют доступом к OTPROM. Процедура сброса загружает CCR из CCB в нормальном режиме и из PCCB в режиме программирования. Необходимо программировать CCB, используя любой из режимов программирования, но для записи PCCB нужно использовать SLAVE режим.

Доступ к OTPROM в нормальном режиме функционирования

В нормальном режиме программирования биты защиты CCB0 управляют разрешением чтения и записи в OTPROM. Таблица 14.2 описывает варианты защиты. Необходимо программировать CCB, используя любой из режимов программирования.

Таблица 14.2 – Защита памяти в нормальном режиме функционирования

Защита чтения LOC1 (CCR0.7)	Защита записи LOC0 (CCR0.6)	Режим защиты
1	1	Защиты нет. OTPROM может читаться. Разрешён Run-time режим.
1	0	Защита записи. Run-time программирование запрещено. OTPROM может читаться.
0	1	Защита чтения. Run-time программирование запрещено. Если идет выполнение программы из внешней памяти, то могут читаться только векторы прерываний и CCB. Ключ безопасности защищен от записи.
0	0	Защита записи и чтения. Run-time программирование запрещено. Если идет выполнение программы из внешней памяти, то могут читаться только векторы прерываний и CCB.

Очистка CCB0.6 обеспечивает защиту всей OTPROM от случайной или преднамеренной записи. Очистка CCB0.7 обеспечивает защиту чтения OTPROM и защиту ключа безопасности от записи. С этой защитой контроллер шины не будет читать защищенную область OTPROM. Попытка загрузить вторичный программный счетчик внешним адресом приводит к сбросу контроллера. Поскольку значение вторичного программного счетчика может опережать главный счётчик на четыре байта, то контроллер блокирует чтение кодов из последних четырех байт внутренней памяти. Векторы прерываний и CCBs не защищены от чтения, так как прерывания могут возникать и при выполнении команд из внешней памяти.

Доступ к OTPROM в режиме программирования

Существуют три уровня защиты в режиме программирования:

- Полный запрет программирования.
- Полный запрет программирования, но разрешение режима ROM-dump после проверки ключа безопасности.
- Разрешение режимов ROM-dump после проверки ключа безопасности, авто-программирования и SLAVE режима.

Эти уровни защиты обеспечиваются битами защиты PCCB0, битами защиты CCB0 и внутренним ключом безопасности (таблица 14.3). При входе в режимы программирования процедура сброса загружает PCCB в регистр конфигурации кристалла. Также во внутреннюю RAM загружается CCB0 для обеспечения дополнительного уровня безопасности.

Можно запрограммировать байты CCB, используя любой из режимов, но только в SLAVE режиме разрешается доступ к байтам PCCBs, и только в режимах SLAVE и автопрограммирования возможна запись внутреннего ключа безопасности.

Таблица 14.3 – Варианты защиты памяти в режиме программирования

LOC1 (CCR0.7)		LOC0 (CCR0.6)		Ключ безопасности запрограммирован?	Состояние защиты
PCCB	CCB	PCCB	CCB		
1	1	1	1	Нет	Защиты нет. Разрешены все режимы программирования.
1	X	0	X	Да	Запрещены все режимы программирования. Режим ROM-dump разрешен после проверки ключа безопасности.
1	0	1	0	Да	Авто и SLAVE режимы разрешены после проверки ключа безопасности.
0	X	0	X	X	Полный запрет программирования

Если необходимо полностью запретить программирование, необходимо очистить оба бита защиты PCCB0. Если эти биты очищены, то запрещен вход в любой из режимов программирования.

Если необходимо запретить программирование, но разрешить ROM-dump, надо оставить бит защиты чтения PCCB0.7 незапрограммированным и очистить бит защиты записи. В PCCB0.6 для защиты от неавторизованного чтения надо записать ключ безопасности. В режиме ROM-dump производится сравнение ключа безопасности, записанного во внутреннюю память, с ключом во внешней памяти, независимо от битов защиты CCB0. Если проверка прошла успешно, то продолжается выполнение последующих инструкций, если нет, контроллер входит в бесконечный цикл.

Если необходимо разрешить режимы SLAVE, автопрограммирования и ROM-dump нужно оставить оба бита защиты PCCB0 незапрограммированными. Чтобы защитить контроллер от неавторизованного программирования, надо очистить биты

защиты CCB0 и записать ключ безопасности. После того, как устройство войдет в SLAVE режим или режим автопрограммирования, соответствующая процедура из тестовой области ROM читает биты защиты CCB0. Если любой бит защиты памяти установлен, процедура сравнивает ключ безопасности, расположенный во внутренней памяти, с внешним ключом. Если проверка прошла успешно, процедура продолжается; в противном случае, устройство входит в бесконечный внутренний цикл.

Можно программировать внутренний ключ безопасности либо в авто-, либо в SLAVE режимах.

Примечание – Если ключ безопасности остается незапрограммированным (заполненным FFFFH), возможен неавторизованный доступ к OTPROM, используя внешнюю EPROM с незапрограммированным внешним ключом или используя режим SLAVE.

Управление выборкой из внешней памяти

Два UPROM бита запрещают выборку команд и данных из внешней памяти. Если UPROM биты записаны, попытка обращения к внешней памяти приведет к сбросу контроллера. UPROM биты можно программировать, используя SLAVE режим.

Программируя бит DEI, можно запретить выборку команд из внешней памяти. Попытка загрузить внешний адрес во вторичный программный счетчик приведет к сбросу устройства. Поскольку значение вторичного программного счетчика может опережать значение главного счётчика на четыре байта, то контроллер блокирует чтение кодов из последних четырех байт внутренней памяти. Автоматический сброс также дает дополнительную защиту от выполнения неконтролируемых команд.

Программируя бит DED, можно запретить как запись, так и чтение данных из внешней памяти. Попытка обратиться к внешней памяти приведёт к сбросу контроллера. Установка этого бита запрещает режим ROM-dump.

Для правильного программирования этих битов см. таблицу 14.4. Программирование осуществляется в SLAVE режиме. В нормальном режиме можно определить значения этих битов, читая регистр UPROM (рисунок 14.1).

USFR Адрес: 1FF6h

Состояние после сброса: 02h

Нестираемый PROM (USFR) регистр содержит два бита, которые запрещают доступ к внешней памяти. Эти биты могут быть запрограммированы, но не могут быть стерты.

7 бит	6 бит	5 бит	4 бит	3 бит	2 бит	1 бит	0 бит
–	–	–	–	DEI	DED	–	–

Номер бита	Обозначение	Функция
7:4	—	Зарезервированы. Всегда записывать нули.
3	DEI	Установка этого бита запрещает выборку команд из внешней памяти. Любая попытка загрузить внешний адрес приводит к сбросу устройства.
2	DED	Установка этого бита запрещает обращение к внешним данным. Любая попытка записи или чтения из внешней памяти приведет к сбросу.
1:0	—	Зарезервированы. Всегда записывать нули.

Рисунок 14.1 – Нестираемый PROM (USFR) регистр

Пользователь может проверить UPROM биты, чтобы удостовериться, что они запрограммированы, но не может стереть их.

Таблица 14.4 – Значения UPROM и их расположение для SLAVE режима

Бит	Значение	Адрес
DEI	08H	0718H
DED	04H	0758H

14.4 Ширина программирующего импульса

Ширина программирующего импульса задается различными способами, в зависимости от режима программирования. В SLAVE режиме ширина импульса управляется PALE# сигналом. В режиме автопрограммирования значение программирующего импульса загружается в PPW из внешней памяти (EPROM). (В run-time режиме программное обеспечение управляет шириной импульса.)

Чтобы правильно определить PPW_VALUE для нужной частоты устройства, используется следующую формулу. Результат округляют до следующего старшего целого числа.

$PPW\_VALUE = 62{,}5 \times F_{BQ1}$,

где PPW_VALUE – 16-битное слово;

F_{BQ1} - входная частота в МГц;

PPW не имеет прямого доступа.

Регистр ширины программирующего импульса загружается из внешней EPROM (адреса 14H и 15H) в режиме автопрограммирования. PPW_VALUE определяет ширину программирующего импульса.

15 бит	14 бит	13 бит	12 бит	11 бит	10 бит	9 бит	8 бит
PPW15	PPW14	PPW13	PPW12	PPW11	PPW10	PPW9	PPW8

7 бит	6 бит	5 бит	4 бит	3 бит	2 бит	1 бит	0 бит
PPW7	PPW6	PPW5	PPW4	PPW3	PPW2	PPW1	PPW0

Номер бита	Обозначение	Функция
15:0	PPW15:0	PPW_VALUE Это значение устанавливает ширину программирующего импульса для режима автопрограммирования. Чтобы вычислить PPW_VALUE используют вышеприведённую формулу. В таблице 14.5 приведены значения для различных частот.

Рисунок 14.2 – Программирование ширины импульса (PPW) регистра

В таблице 14.5 представлены вычисления и значения PPW_VALUE для частот 8 МГц и 16 МГц. Минимальная ширина программирующего импульса составляет 250 мкс.

Таблица 14.5 – Пример PPW_VALUE вычисления

F_{BQ1}	Требуются два импульса по 250 мкс
8 МГц	$PPW\_VALUE = 62{,}5 \times 8 = 504 = 01F4H$
16 МГц	$PPW\_VALUE = 62{,}5 \times 16 = 1000 = 03F8H$

14.5 Модифицированный QUICK-PULSE алгоритм

Режимы SLAVE и автопрограммирования используют модифицированный quick-pulse алгоритм (рисунок 14.3). Модифицированный quick-pulse алгоритм формирует программирующий импульс для каждого слова OTPROM. После необходимого количества программирующих импульсов происходит верификация записанных данных. Ошибка верификации переключает PVER сигнал, но не приводит к остановке процесса программирования. Этот процесс будет повторяться до тех пор, пока каждое слово OTPROM не запишется и не верифицируется.

Примечание – Контроллер использует два программирующих импульса.

Рисунок 14.3 – Модифицированный quick-pulse алгоритм

В режиме автопрограммирования программирующий импульс повторяется дважды с той длительностью, которая задается во внешнем EPROM. В SLAVE режиме ширина программирующего импульса управляется сигналом PALE#. Во всех случаях, для успешного программирования ширина импульса должна быть не менее 100 мкс.

14.6 Выводы в режиме программирования

Рисунок 14.4 показывает сигналы, используемые для программирования, а таблица 14.6 описывает их. EA#, VPP и PMODE выводы обеспечивают вход контроллера в режим программирования. Необходимо задать PMODE (P0.7:4) выводы таким образом, чтобы выбрать нужный режим программирования (см. таблицу 14.7). В каждой процедуре программирования выводы порта 2 работают в режиме специальной функции сигнала. Порты 3 и 4 автоматически используют PBUS в течение программирования.

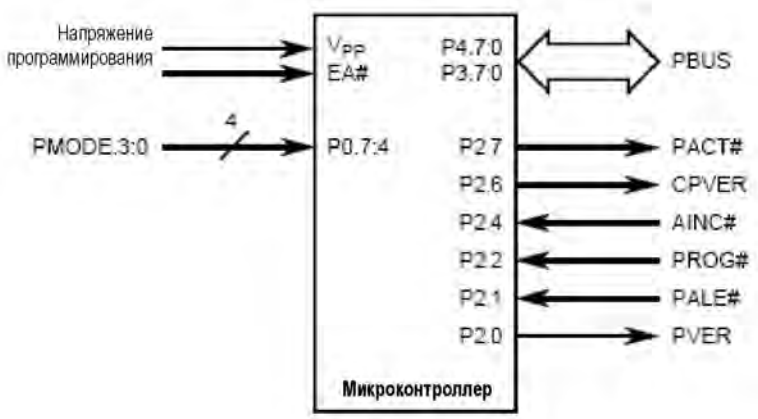


Рисунок 14.4 – Функции выводов в режиме программирования

Таблица 14.6 – Описание выводов (I – вход, O – выход, I/O – вход/выход)

Вывод порта	Сигнал специальной функции	Тип	Режим програм-мирования	Описание
1	2	3	4	5
P0.7: P0.4	PMODE.3: PMODE.0	I	Все	Выбор режима программирования. Определяет режим программирования. Значения PMODE читаются после сброса устройства и не должны изменяться при программировании.
P2.0	PVER	O	Авто, SLAVE	Флаг верификации. В течение SLAVE или автопрограммирования, PVER обновляется после каждого импульса программирования. Высокий уровень показывает, что программирование ячейки прошло успешно, низкий – говорит об ошибке.
P2.1	PALE#	I	SLAVE	Вход ALE в режиме программирования. В SLAVE режиме спадающий фронт заставляет контроллер читать команду и адрес с шины PBUS.
P2.2	PROG#	I	SLAVE	Программирование. В режиме программирования по спаду сигнала фиксируются данные на PBUS и начинается программирование. По нарастающему фронту программирование заканчивается. В режиме чтения слова (dump) по спаду сигнала содержимое ячейки OTPROM выводится в PBUS, по нарастанию – заканчивается передача данных.
P2.4	AINC#	I	SLAVE	Автоинкремент. В течение SLAVE режима активный низкий уровень разрешает автоинкремент. (Автоинкремент позволяет считывать или записывать ячейки последовательно без передачи адреса при каждом обмене).
P2.6	CPVER	O	SLAVE	Полная верификация программирования. Высокий уровень на этом выводе в режиме SLAVE показывает, что все ячейки запрограммированы верно.

Продолжение таблицы 14.6

1	2	3	4	5
P2.7	PACT#	O	Авто, ROM-dump	Программирование активно. В режиме программирования AUTO или ROM-dump низкий уровень на выходе показывает, что идет процесс программирования или чтения, а высокий уровень индицирует окончание операции.
P4.7:0, P3.7:0	PBUS	I/O	SLAVE	Шина адрес/команды/данные. В течение SLAVE режима порты 3 и 4 служат двунаправленными портами с открытыми стоками для передачи команд, адресов и данных. В SLAVE режиме необходимо наличие внешних pull-up резисторов (поддержки высокого уровня).
P4.7:0, P3.7:0	PBUS	I/O	Авто ROM-dump	Шина адрес/команды/данные. В течение автопрограммирования и ROM-dump порты 3 и 4 служат системной шиной для доступа к внешней памяти.
—	EA#	I	Все	Внешний доступ. Управляет переходом контроллера в режим программирования. Если напряжение на EA# равно VPP, то при нарастающем фронте RESET# устройство переходит в режим программирования. EA# фиксируется только по нарастающему фронту сигнала RESET#.
—	VPP	I	Все	Напряжение программирования. В течение программирования, типичное значение VPP плюс 12,5 В.

14.7 Вход в режимы программирования

Для правильного выполнения программ необходимо обеспечить минимально необходимую аппаратную конфигурацию: BQ1 подключен, неиспользованные входы объединены и подключены к питанию или земле. Перевод устройства в режим программирования производится подачей VPP напряжения на EA# (плюс 12,5 В) во время нарастающего RESET#.

Выбор режима программирования

Значение PMODE (P0.7:4) управляет выбором режима программирования. PMODE фиксируется при нарастающем фронте сигнала RESET#. Для переключения режимов программирования необходимо сбросить устройство. Таблица 14.7 содержит значения PMODE для каждого из режимов. Все другие значения PMODE зарезервированы.

Таблица 14.7 – Значения PMODE

Значение PMODE	Режим
5H	SLAVE режим
6H	ROM-dump (вывод содержимого памяти)
CH	Автопрограммирование

Включение и выключение питания при программировании

При начале программирования, необходимо следовать следующим инструкциям по включению и выключению питания.

Примечание – Невыполнение этих требований может привести к выходу устройства из строя.

- Нельзя прикладывать напряжение к VPP при низком #VCC1.
- VPP напряжение должно быть в пределах 1 В от #VCC1, если #VCC1 меньше 4,5 В. VPP не должно превышать 4,5 В, пока #VCC1 не достигло 4,5 В.
- Нельзя превышать максимальное значение VPP.
- EA# должен достигать напряжения программирования раньше VPP.
- Выводы PMODE (P0.7:4) должны быть установлены до возрастающего фронта RESET#.
- Все напряжения должны быть в пределах диапазонов, указанных в ТУ, генератор должен стабилизироваться до возрастающего фронта RESET#.
- Напряжения, прикладываемые к #VCC1, VPP, EA# и RESET# должны быть стабильны, без шумов, выбросов и помех.
- Все выводы #0V должны быть хорошо заземлены.

Последовательность включения питания

1. Удерживать низкий уровень на RESET#, пока #VCC1 стабилизируется. В это время выводы EA# и VPP могут быть не подключены.
2. После стабилизации #VCC1 и генератора удерживать RESET# в низком уровне и подать напряжение VPP на EA#.
3. После стабилизации EA# подать напряжение программирования на вывод VPP.
4. Установить значение PMODE для выбора алгоритма программирования.
5. Подать высокий уровень на RESET#..
6. Закончить работу с выбранным алгоритмом программирования.

Последовательность выключения питания

1. Установить и удерживать RESET# в низком уровне в процессе выключения питания.
2. Снять напряжение программирования с вывода VPP и оставить его неподключенным.
3. Снять напряжение программирования с вывода EA# и оставить его неподключенным.
4. Снять питание #VCC1 и подождать, пока установится 0 В.

14.8 Режим программирования SLAVE

SLAVE режим позволяет записывать и считывать OTPROM, включая биты PCCB и UPROM, используя EPROM программатор.

В этом режиме порты 3 и 4 работают как PBUS, передавая команды, адреса и данные. Наименьший значащий бит PBUS (P3.0) управляет типом команды (1 = программирование слова; 0 = чтение слова). Остальные 15 битов содержат адрес слова, которое будет запрограммировано или прочитано. Некоторые выводы порта 2 обеспечивают сигналы квитирования. AINC# сигнал управляет автоинкрементом, позволяя программировать или считывать последовательность ячеек OTPROM. Это ускоряет процесс программирования, так как при этом не требуется формировать и декодировать последовательные адреса.

Примечание – Если во время программирования ключа безопасности произойдет сбой или возникнет состояние сброса, возможна запись неправильного значения. Поэтому вначале необходимо записать всю программу и только потом биты защиты CCB (CCB0.6 и CCB0.7).

Чтение слова-сигнатуры и значения программирующего напряжения

Слово-сигнатура идентифицирует устройство; напряжение программирования определяет напряжения VPP и #VCC1, требуемые для программирования. Эта информация содержится в тестовом ROM по адресам 2070H, 2072H и 2073H, однако непосредственный доступ к ней возможен по адресам 0070H, 0072H и 0073H. Можно использовать команду чтения слова в SLAVE режиме для чтения сигнатуры и напряжения программировнаия. Внешний программатор может использовать эту информацию, чтобы определить тип устройства и эксплуатационные режимы. Никогда нельзя записывать в эти ячейки. Напряжения рассчитываются, используя следующее уравнение (после преобразования значения ROM в десятичное число).

Напряжение = 20 × значение ROM / 256

$$\#\text{VCC1(40H)} = \frac{20 \times 64}{256} = 5 \text{ (В)}, \quad \text{VPP(0A0H)} = \frac{20 \times 160}{256} = 12{,}5 \text{ (В)}$$

Таблица 14.8 – Слово-сигнатура

Слово-сигнатура		#VCC1		VPP	
Адрес	Значение	Адрес	Значение	Адрес	Значение
0070H	8794H	0072H	40H	0073H	0A0H

Схема включения и распределение памяти в SLAVE режиме

На рисунке 14.5 показана схема включения, а таблица 14.9 показывает распределение памяти для SLAVE режима.

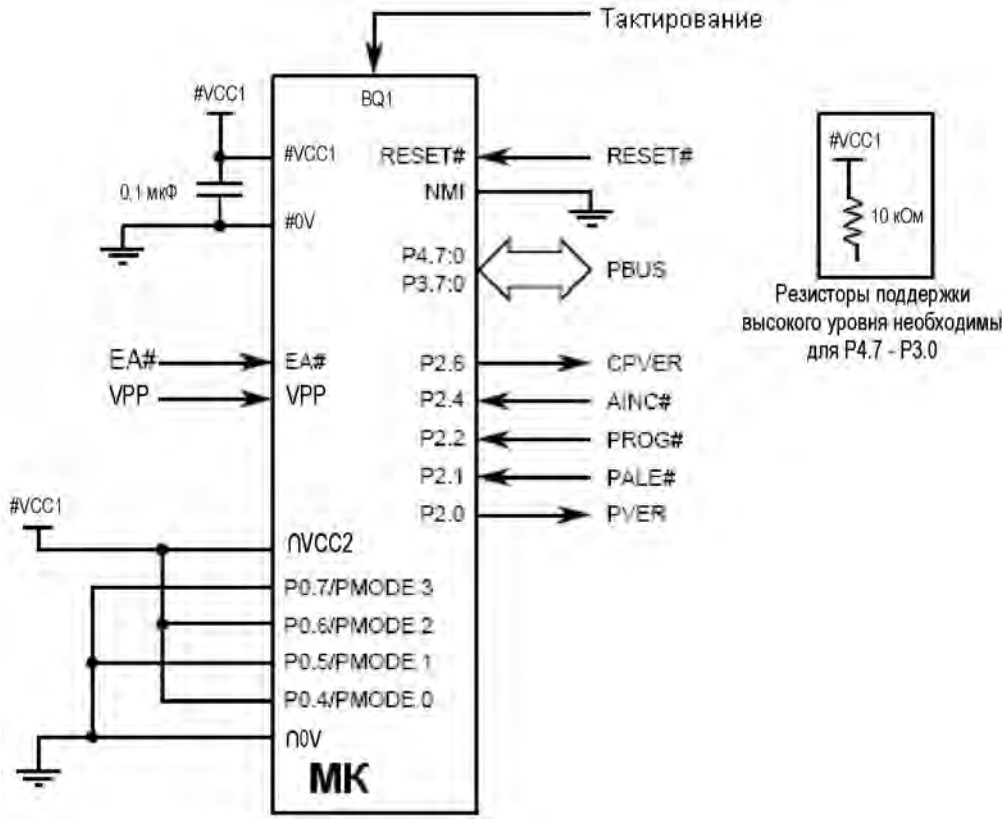


Рисунок 14.5 – Схема включения МК для режима SLAVE

Таблица 14.9 – Распределение памяти в SLAVE режиме

Описание	Адрес	Комментарии
OTPROM	2000–5FFFH	OTPROM ячейки
DED	0758H	UPROM ячейка
DEI	0718H	UPROM ячейка
PCCB	0218H	Тестовая EPROM
Напряжения программирования (см. таблицу 14.8)	0072H, 0073H	Только чтение
Слово-сигнатура	0070H	Только чтение

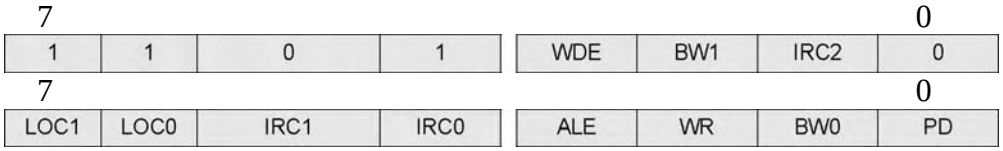
Операционные режимы работы

Регистры конфигурации кристалла (CCR) определяют операционные режимы системы. Так как конфигурация в режиме программирования не обязательно совпадает с конфигурацией в обычном режиме работы, МК поддерживает эти различные конфигурации. Конфигурацию определяют, задав необходимые значения в регистрах конфигурации кристалла (CCB), расположенных в OTPROM. Конфигурацию в режиме программирования определяют, задав необходимые значения в регистрах PCCB, расположенных в тестовом OTPROM.

На рисунке 14.6 представлено схематичное описание CCR. Во время сброса контроллер загружает байты CCB в CCR в нормальном режиме функционирования и PCCB – в режимах программировании. Возможно программировать CCB, используя любой из методов программирования, но PCCB возможно программировать только в режиме SLAVE.

Регистры конфигурации кристалла (CCR) управляют циклами ожидания, режимом пониженного энергопотребления и внутренней защитой памяти. Эти регистры загружаются байтами PCCB в режимах программирования и байтами CCB в нормальном режиме.

CCR1, CCR0 нет прямого доступа



Обозначение	Функция
WDE	Разрешение сторожевого таймера. По умолчанию PCCB запрещает функционирование сторожевого таймера.
BW1	Выбор ширины шины. По умолчанию PCCB содержит бит, выбирающий управление выводом BUSWIDTH.
IRC2	Управление сигналом READY. По умолчанию PCCB выбирает управление выводом READY.
LOC1:0	Биты защиты. По умолчанию PCCB не активирует защиту.
IRC1:0	Управление сигналом READY. По умолчанию PCCB выбирает управление выводом READY.
ALE	Выбор режима стробирования адреса. По умолчанию PCCB выбирает сигнал ALE для стробирования.
WR	Выбор режима стробирования записи. По умолчанию PCCB выбирает сигналы WR# и BHE# для стробирования.
BW0	Выбор ширина шины.

	По умолчанию PCCB содержит бит, выбирающий управление выводом BUSWIDTH.
PD	Режим пониженного энергопотребления. По умолчанию PCCB разрешает режим пониженного энергопотребления.

Рисунок 14.6 – Регистры конфигурации кристалла (CCR)

Алгоритм программирования в SLAVE режиме

Алгоритм данного режима программирования состоит из трех процедур: процедуры декодирования адреса/команды, процедуры записи слова и процедуры чтения слова.

Процедура декодирования адреса/команды (рисунок 14.7) читает информацию с PBUS и передает управление процедурам записи или чтения слова, в зависимости от значения P3.0. Если P3.0 – единица, то происходит программирование слова, а остающиеся биты определяют адрес. Если P3.0 – ноль, то происходит чтение слова. Остающиеся биты также определяют адрес.

Процедура записи слова (рисунок 14.8) проверяет биты защиты CCB. Если любой из битов защиты был запрограммирован (CCB0.6 или CCB0.7), то необходимо иметь соответствующий ключ безопасности, чтобы получить доступ к устройству. Используя команду записи слова, записывают восемь последовательных слов, начиная с адреса 2020H и кончая 202FH. Процедура сохраняет эти восемь слов во внутреннем регистре и сравнивает их значение с внутренним ключом. Если ключи совпадают, то возможна дальнейшая работа с OTPROM, иначе устройство входит в бесконечный цикл.

Процедура чтения слова (рисунок 14.10) также проверяет биты защиты CCB. Если биты защиты незапрограммированы, то возможна выдача данных из OTPROM на шину PBUS. Если любой бит запрограммирован, то возможно проводить циклы записи без чтения данных из OTPROM.

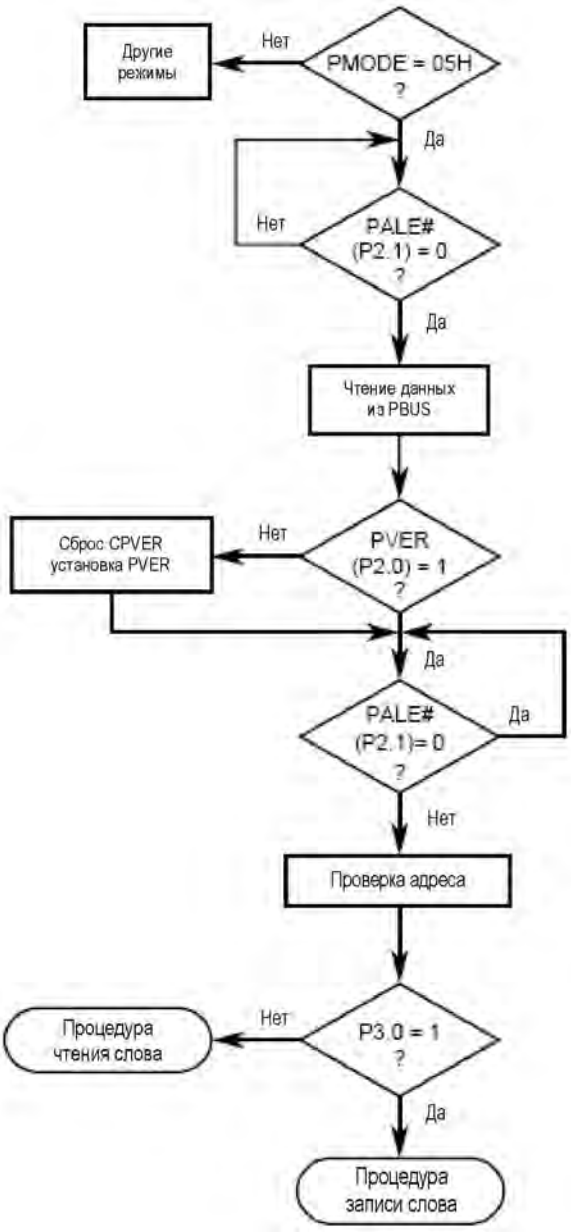


Рисунок 14.7 – Процедура декодирования адреса/команды

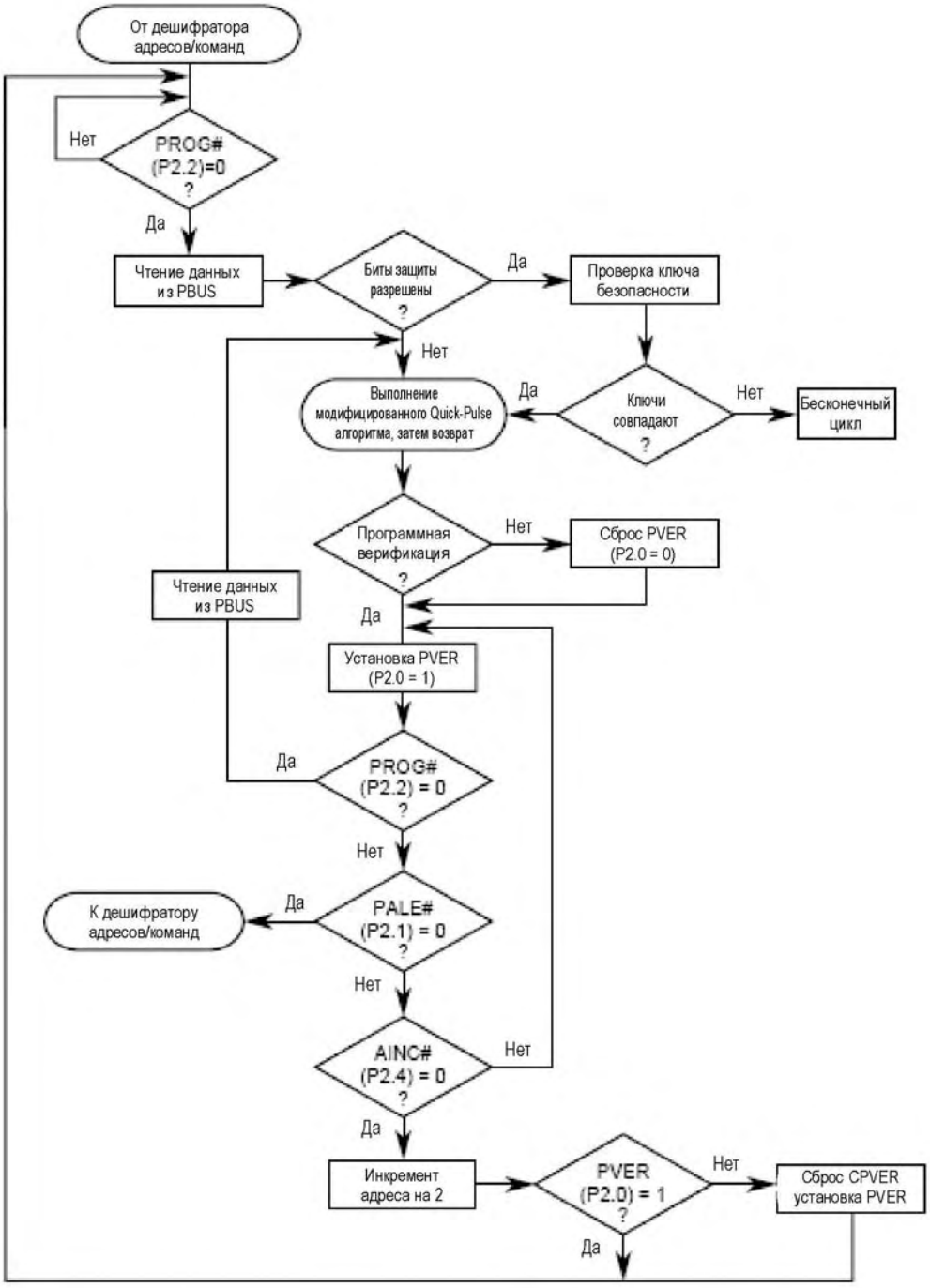


Рисунок 14.8 – Процедура программирования слов

Рисунок 14.9 показывает временные диаграммы для режима записи слова с автоинкрементом. По спадающему фронту сигнала PALE# происходит чтение команд и адресов с PBUS. По спадающему фронту сигнала PROG# происходит фиксация данных и начинается программирование. PROG# сигнал управляет длительностью программирующего импульса. По возрастающему фронту PROG# происходит верификация записанного значения и выставляется сигнал PVER. AINC# сигнал является дополнительным и служит для указания режима автоинкремента адреса. Если не используется сигнал AINC#, то необходимо послать новую команду для указания нового адреса.

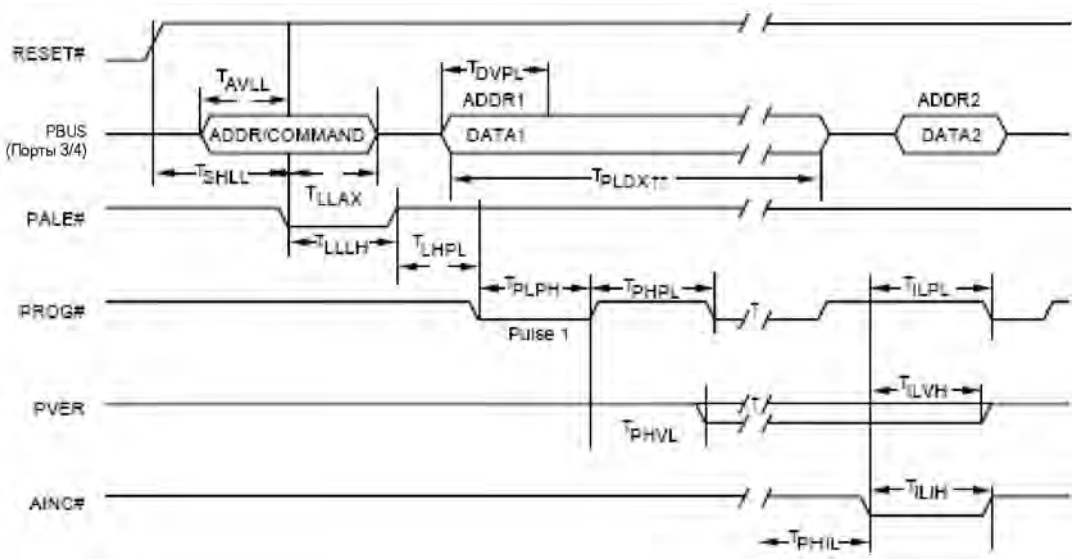


Рисунок 14.9 – Диаграммы программирования слова

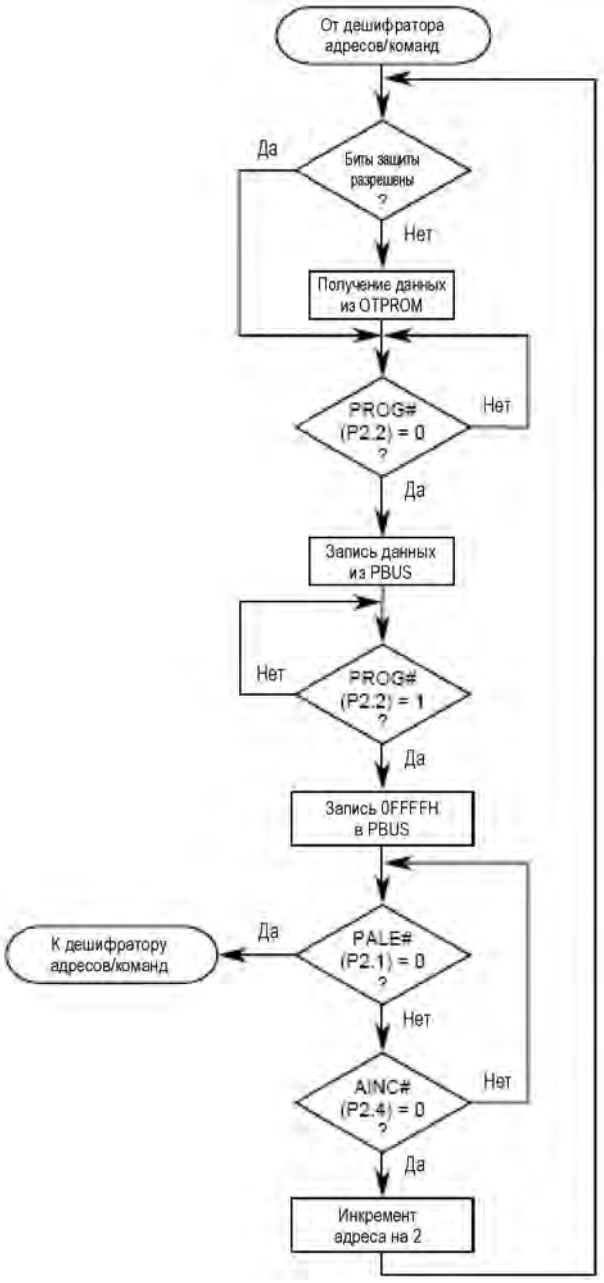


Рисунок 14.10 – Процедура чтения слова

На рисунке 14.11 показаны диаграммы в режиме чтения слов из памяти. Сигнал PROG# управляет шиной. В режиме выдачи слова вывод AINC# может остаться активным. Сигнал PROG# автоматически увеличивает адрес.

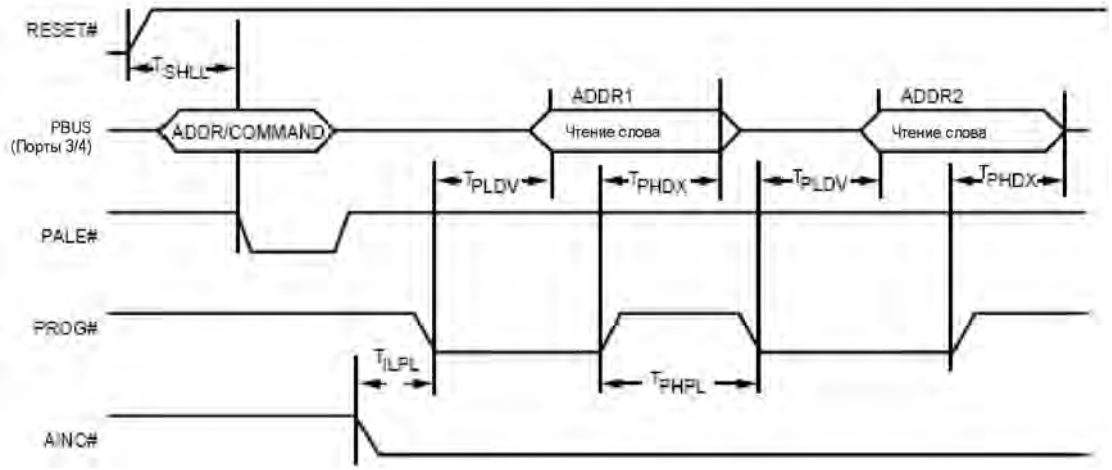


Рисунок 14.11 – Диаграммы чтения слова

Временные параметры и их обозначения

В таблице 14.10 представлены обозначения, использующиеся в диаграммах.

Таблица 14.10 – Временные обозначения

Обозначение	Описание
T_{SHLL}	Сброс высокого уровня до первого низкого PALE#.
T_{LLLH}	Длительность PALE# импульса.
T_{AVLL}	Время установки адреса.
T_{LLAX}	Время удержания адреса.
T_{PLDV}	Время от низкого PROG# до действительного вывода слова.
T_{PHDX}	Время удержания слова данных.
T_{DVPL}	Время установки данных.
T_{PLDX}	Время удержания данных.
T_{PLPH}	Длительность импульса PROG#.
T_{PHLL}	Время от высокого PROG# до следующего низкого PALE#.
T_{LHPL}	Время от высокого PALE# до низкого PROG#.
T_{PHPL}	Время от высокого PROG# до следующего низкого PROG#.
T_{PHIL}	Время от высокого PROG# до низкого AINC #.
T_{ILIH}	Длительность импульса AINC# .
T_{ILVH}	Время удержания PVER после низкого AINC#.
T_{ILPL}	Время от низкого AINC# до низкого PROG#.
T_{PHVL}	Время от высокого PROG# до действительного PVER.

14.9 Режим автопрограммирования

Режим автопрограммирования – альтернатива программированию с помощью программатора. Используя этот режим программирования, контроллер самостоятельно программирует себя, используя данные из внешней EPROM (адреса 4000H и старше, см. таблицу 14.11).

Схема включения в режиме автопрограммирования и распределение памяти

На рисунке 14.12 представлена рекомендованная схема для режима автопрограммирования. Таблица 14.11 показывает распределение памяти контроллера. Автопрограммирование определено для частот (6 – 8) МГц. На 8 МГц используют EPROM с t_{ACC} = 250 нс и t_{OE} = 100 нс или более быстродействующие устройства.

Подается низкий уровень на BUSWIDTH для формирования шины данных шириной 8 бит. Соединяется P0.7:4 с #0V и #VCC1, чтобы выбрать режим автопрограммирования (1100B = 0CH). PACT# и PVER – статусные буферизированные выводы. Соединяются все неиспользованные входы с #0V, а все неиспользуемые выводы остаются неподключенными. Входы READY и NMI активны, их соединяют как обозначено на рисунке 14.12.

Примечание – Все внешние адреса EPROM, указанные в этом подразделе, даются для схемы на рисунке 14.12. Если выбирается другая схема включения, соответственно необходимо изменить адреса.

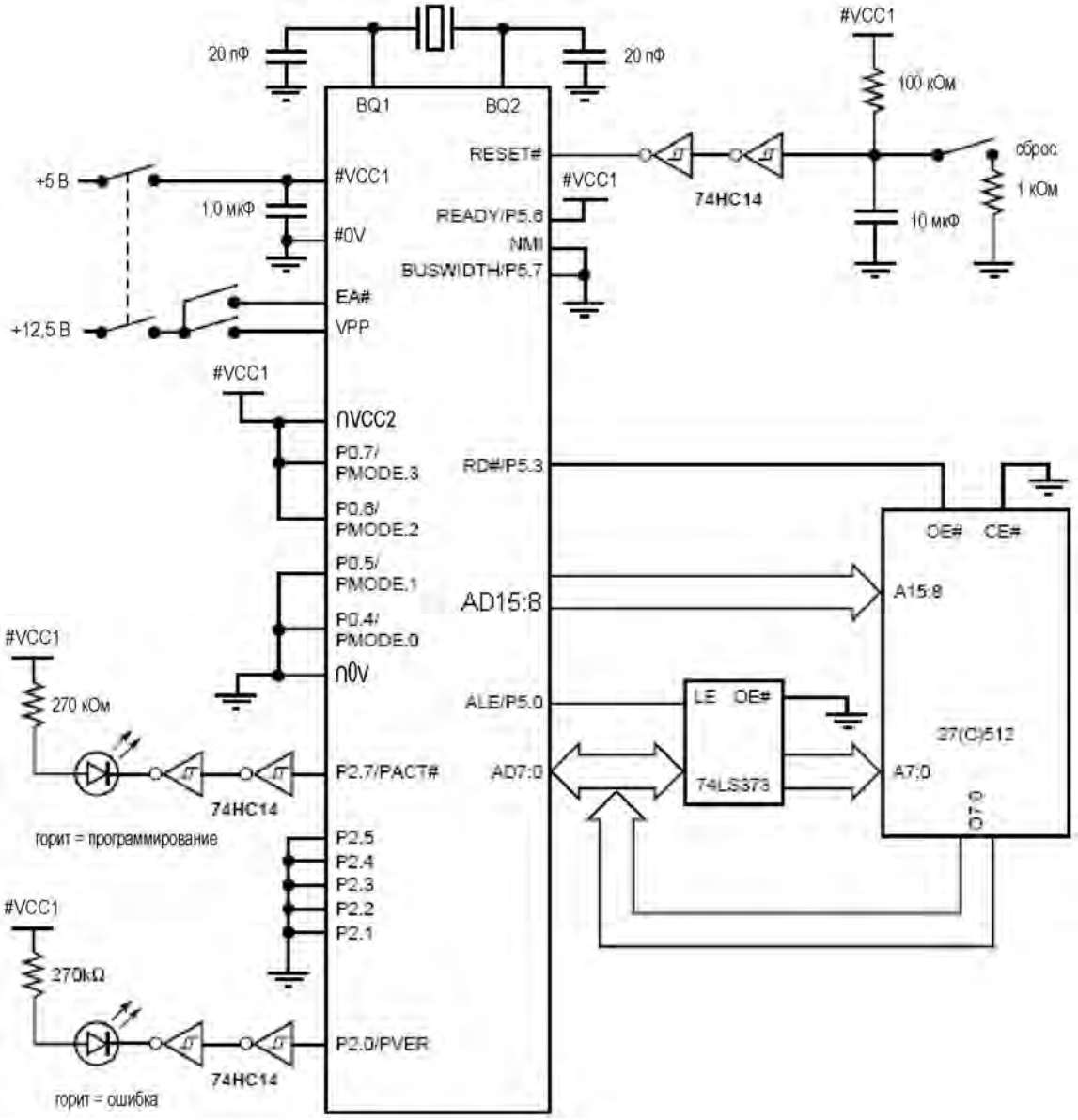


Рисунок 14.12 – Схема для режима автопрограммирования

Таблица 14.11 – Распределение памяти для режима автопрограммирования

Выходной адрес	Адрес внутреннего OTPROM	Адрес в схеме на рисунке 14.12 (A15:0)	Описание
4014H	N/A	14H	PPW младшего байта.
4015H	N/A	15H	PPW старшего байта.
4020–402FH	2020–202FH	0020–002FH	Ключ безопасности.
4000–7FFFH	2000–5FFFH	4000–7FFFH	Команды, данные и зарезервированные байты.

Операционный режим работы

В режиме автопрограммирования байты PCCB загружаются в регистры конфигурации кристалла. Так как устройство получает данные для программирования через внешнюю шину, устройство памяти в программирующей системе должно соответствовать конфигурации по умолчанию (рисунок 14.6). Автопрограммирование требует ширину шины 8 бит, так что в схеме сигнал BUSWIDTH должен быть заземлен. По умолчанию значение PCCB позволяет использовать любую стандартную EPROM, которая удовлетворяет требованиям к временным параметрам, приведенным в ТУ на микросхемы.

В режиме автопрограммирования также загружается байт CCB0 во внутреннюю RAM и проверяются биты защиты. Если любой бит защиты запрограммирован, процедура автопрограммирования сравнивает внутренний ключ безопасности с внешним. Если ключи не совпали, устройство входит в бесконечный внутренний цикл. Если ключи совпали, то процедура продолжается. Процедура автопрограммирования использует модифицированный quick-pulse алгоритм с длительностью программирующих импульсов, записанный во внешнем EPROM.

Алгоритм процедуры автопрограммирования

На рисунке 14.13 представлен алгоритм процедуры программирования в авторежиме. Эта процедура проверяет биты защиты в CCB0; если любой из битов запрограммирован, то происходит сравнение внутреннего ключа безопасности с внешним. Если ключи совпадают, то процедура продолжается, иначе устройство входит в бесконечный цикл.

Если проверка ключа безопасности прошла успешно, то загружается значение длительности программирующих импульсов (PPW) из внешней EPROM во внутренний регистр PPW. После этого устанавливается сигнал PACT#, показывая, что программирование началось. (PACT# также активен во время сброса, хотя процесс программирования не идет.) В начальном состоянии сигнал PVER установлен. В случае обнаружения ошибки программирования, PVER переключится.

После этого процедура считывает данные из внешней EPROM, начиная с адресов 4000H. При этом пропускаются любые слова, содержащие FFFFH (незапрограммированное состояние). Если слово отлично от FFFFH, начинается выполнение модифицированного quick-pulse алгоритма, который записывает это значение в OTPROM, после чего проводится проверка записи. Так продолжается до тех пор, пока не запишется вся OTPROM, после чего сбрасывается сигнал PACT#, и устройство входит в бесконечный цикл.

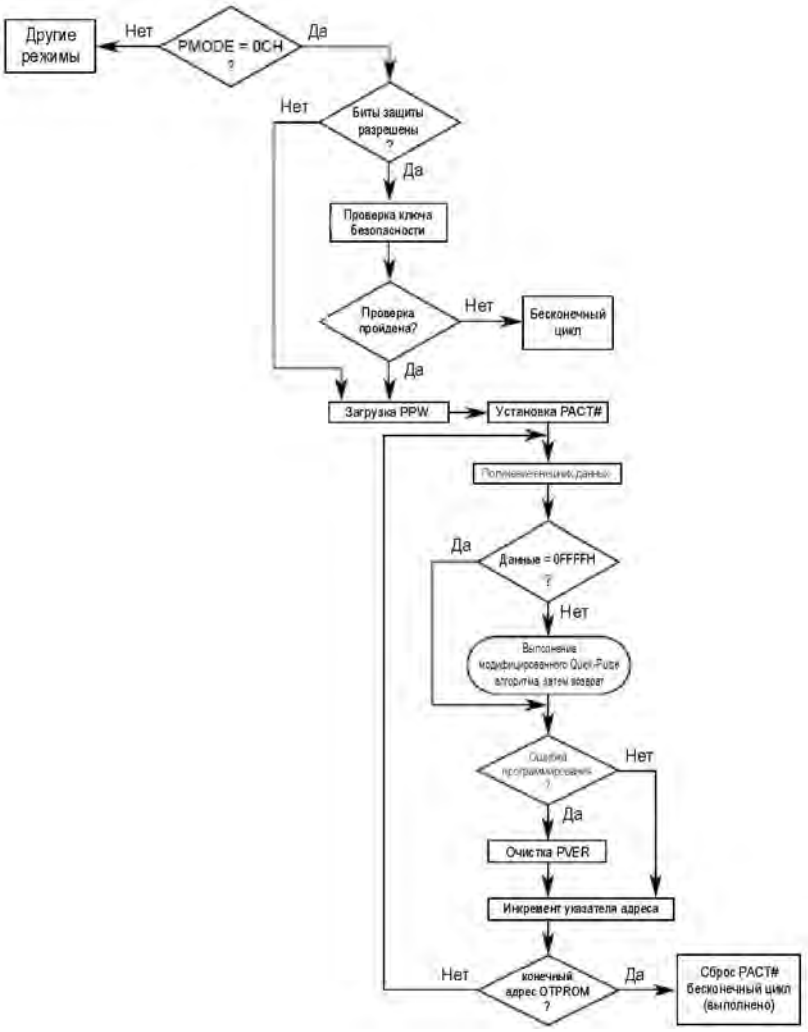


Рисунок 14.13 – Процедура автопрограммирования

Процедура автопрограммирования

В случае сбоя или сброса устройства во время программирования ключа безопасности и битов защиты, дальнейшее программирование невозможно. Чтобы этого не произошло, необходимо следовать следующим рекомендациям.

Примечание – Все адреса приводятся для схемы, показанной на рисунке 14.12. Если используется другая схема, то придется соответственно изменить адреса.

1 Используя «чистую» EPROM, необходимо следовать следующим инструкциям, чтобы пропустить программирование CCB0 и выполнить программирование оставшейся части OTPROM, включая ключ безопасности:

- Поместить значение длительности программирующего импульса (PPW) во внешнюю EPROM по адресам 14H и 15H.
- Оставить байт CCB0 во внешней памяти (4018H) незапрограммированным (0FFFFH).
- Поместить значение CCB1 во внешней памяти по адресу 401AH.
- Поместить ключ безопасности, который будет запрограммирован, во внешней EPROM по адресам 4020H-402FH.
- Поместить значение 20H во внешней памяти по адресам 4019H и 401BH (это требуется для зарезервированных ячеек OTPROM).
- Разместить требуемые коды в EPROM по оставшимся адресам 4000–7FFFH.
- Выполнить последовательность включения питания, чтобы начать автопрограммирование.

- Когда программирование закончено, выполнить последовательность выключения питания перед переходом к шагу 2.

2 Используя другую чистую EPROM, необходимо следовать следующим инструкциям для программирования только CCB0:

- Поместить значение длительности программирующего импульса (PPW) во внешнюю EPROM по адресам 14H и 15H.
- Поместить соответствующее значение CCB0 по адресу 4018H.
- Поместить ключ безопасности во внешней EPROM по адресам 0020H-002FH. Это значение должно соответствовать ключу безопасности, запрограммированному в шаге 1.
- Оставить остальные ячейки EPROM незапрограммированными (0FFFFH).
- Выполнить последовательность включения питания, чтобы начать автопрограммирование.
- Когда программирование закончено, выполнить последовательность выключения питания.

Режим вывода содержимого памяти ROM-DUMP

Режим ROM-DUMP обеспечивает простой способ вывода содержимого OTPROM после автопрограммирования. Используют ту же самую схему, которая используется при автопрограммировании, но с измененным подключением выводов PMODE (P0.7:4). Чтобы выбрать ROM-DUMP режим (PMODE=6H), надо соединить P0.6 и P0.5 с #VCC1 и заземлить P0.7 и P0.4. Примерная схема включения (рисунок 14.12) не показывает необходимые подключения WR# и VPP для обеспечения записи в EPROM. И хотя в примере используется EPROM, можно также использовать RAM.

Примечание – Если запрограммирован бит DED (USFR.2), режим ROM-DUMP не возможен.

Чтобы войти в режим ROM-DUMP, надо следовать рекомендациям по включению питания, приведенным выше. Режим ROM-DUMP проверяет ключ безопасности, независимо от битов защиты CCR. Если запрограммирован ключ безопасности, соответствующий ключ должен находиться во внешней памяти, иначе устройство войдет в бесконечный цикл. Если ключ безопасности пройдет проверку, режим ROM-DUMP фиксирует PPW, после чего вся OTPROM записывается во внешнюю память. PACT# остается в низком уровне, пока идет выдача содержимого OTPROM, а после ее окончания переключается в высокий.

14.10 Режим RUN-TIME

Можно программировать ячейки OTPROM во время обычного выполнения программы. Чтобы сделать OTPROM доступной, надо установить на EA# напряжение #VCC1 во время сброса устройства. Подать напряжение программирования на вывод VPP на время всего процесса программирования. Затем просто записывать информацию в ячейку, которую необходимо запрограммировать.

Примечание – Запись любого бита защиты в CCB0 делает RUN-TIME режим недоступным.

Сразу же после окончания записи OTPROM устройство должно войти в IDLE режим или 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 пять программирующих циклов, каждый из которых должен быть приблизительно 100 мкс.

15 Рекомендации по отладочным средствам ИМС

15.1 Программаторы для программируемого варианта ИМС

Могут использоваться программаторы любой фирмы, обеспечивающие программирование аналогов разработанной ИМС (87C196MC 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 микроконтроллера 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 разработанного микроконтроллера.

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 функциональным аналогом 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 MCA-96
- Кросс-компилятор языка «Си» MCC-96
- Отладчик-симулятор PDS-96
- Внутрисхемный эмулятор PICE-196**.**

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 1 М при использовании основной платы М-196/Х (до 128К 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" - характеризует 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 МСА-96, но и Си-компилятор МСС-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 исправит;
- 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-19;
- 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 1874ВЕ16Т, 1874ВЕ86Т.

16 Заключение

В настоящем руководстве КФДЛ.431295.021 приведено подробное описание архитектуры, функционального построения, системы команд и особенностей применения ИМС 1874ВЕ86Т и 1874ВЕ16Т, которые представляют собой СБИС однокристального 16-разрядного микроконтроллера с функцией управления двигателями, в том числе модификация с внутренней программной памятью объемом 16К типа OTPROM.

Микроконтроллер предназначен для применения в цифровых системах управления, для управления робототехническими комплексами, в системах автоматизации технологических процессов, в системах автоматизированного управления электроприводом, оргтехнике, вычислительной технике, телекоммуникационной технике и т. д.

Все значения электрических параметров ИМС приведены в АЕЯР.431280.496 на изделие, значения параметров, приведенные в настоящем руководстве, являются справочными.

КФДЛ.431295.021 может служить практическим пособием по применению микроконтроллеров для разработчиков систем на основе ИМС 1874ВЕ86Т, 1874ВЕ16Т и программистов.

Приложение А

(обязательное)

Система команд

Это приложение даёт справочную информацию о системе команд микроконтроллеров 1874ВЕ86Т, 1874ВЕ16Т. В нём описана каждая команда, показана связь между командами и флагами PSW, приведены 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 вместе с соответствующей мнемоникой.

В таблице А.8 приводится карта кодов команд.

В таблице А.9 приводятся длины и коды команд для каждого используемого режима адресации.

В таблице А.10 приводится время выполнения команд, измеряемое в машинных циклах.

Таблица А.1 – Переменные, используемые в операндах

Пере-менные	Описание
1	2
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 определённого 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 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.

Продолжение таблицы А.1

1	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 S или D.	

Таблица А.2 – Система команд

Обозначение	Операция	Формат команды
1	2	3
ADD (2 операнд-ное)	«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 операнд-ное)	«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 операнд-ное)	«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 операнд-ное)	«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	«Сложение слов с переносом». Сложение слов источника, приёмника, флага переноса (0 или 1) и сохранение суммы в приемнике. (DEST) <– (DEST)+(SRC)+C	DEST, SRC ADDC wreg, waop (101001aa)(waop)(wreg)
ADDCB	«Сложение байтов с переносом». Сложение байтов источника, приёмника, флага переноса (0 или 1) и сохранение суммы в приемнике. (DEST) <– (DEST)+(SRC)+C	DEST, SRC ADDCB breg, baop (101101aa)(baop)(breg)

Продолжение таблицы А.2

1	2	3
AND (2 операнд-ное)	«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)
ANDB (3 операнд-ное)	«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	«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 указатели источника и приёмника, которые хранятся в соседних регистрах слов. Регистр слова Wreg (CNTREG) определяет количество перемещений. Блоки данных могут находиться в любом месте регистрового ОЗУ, 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 end_if	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.

Продолжение таблицы А.2

1	2	3
BMOVI	«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 регистрового ОЗУ, 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 end_if	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	«Косвенное разветвление». Продолжает выполнение с адреса, определяемого операндом регистра слова. PC <– (DEST)	DEST BR [wreg] (11100011)(wreg)
CLR	«Очистка слова». Обнуляет значение операнда. (DEST) <– 0	DEST CLR wreg (00000001)(wreg)
CLRB	«Очистка байта». Обнуляет значение операнда. (DEST) <– 0	DEST CLRB breg (00010001)(breg)
CLRC	«Очистка флага переноса». Обнуляет флаг переноса. C <– 0	CLRC (11111000)
CLRVT	«Очистка дополнительного флага переполнения». Обнуляет флаг VT VT <– 0	CLRVT (11111100)
CMP	«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)

Продолжение таблицы А.2

1	2	3
CMPB	«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	«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	«Декремент слова». Декрементирует величину операнда на 1. (DEST) <– (DEST)-1	DEST DEC wreg (00000101)(wreg)
DECB	«Декремент байта». Декрементирует величину операнда на 1. (DEST) <– (DEST)-1	DEST DECB breg (00010101)(breg)
DI	«Запрещение прерываний». Запрещает прерывания. Запросы на прерывания не удовлетворяются после этой команды. Interrupt Enable (PSW.1) <– 0	DI (11111010)
DIV	«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	«Деление чисел типа SHORT INTEGER». Делит содержимое приёмника – операнд типа INTEGER на содержимое источника – операнд типа SHORTINTEGER, 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

1	2	3
DIVU	«Деление чисел типа WORD, незнаковых». Делит содержимое приёмника - операнд типа DOUBLEWORD 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	«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	«Декремент и переход, если не равно нулю». 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)	DJNZ breg, cadd (11100000)(breg)(disp)
DJNZW	«Декремент и переход, если не равно нулю». 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 диапазоне от минус 128 до плюс 127. (COUNT) <– (COUNT) - 1 if (COUNT) ≠ 0 then PC <– PC + disp (примечание 1)	DJNZW wreg, cadd (11100001)(wreg)(disp)

Продолжение таблицы А.2

1	2	3
DPTS	«Блокирование периферийного сервера – PTS». Блокирует PTS. PTS Disable (PSW.2) <– 0	DPTS (11101100)
EI	«Разрешение прерываний». Разрешает прерывания, запрашиваемые после выполнения следующего оператора. Запросы на прерывания не могут удовлетворяться немедленно после выполнения этой команды. Interrupt Enable (PSW.1) <– 1	EI (11111011)
EPTS	«Разблокировка периферийного сервера – PTS». Разблокирует PTS. PTS Enable (PSW.2) <– 1	EPTS (11101101)
EXT	«Знаковое расширение INTEGER в LONG-INTEGER». Расширяет знаком младшее слово операнда до двойного слова. if (low word DEST) < 8000h then (high word DEST) <– 0 else (high word DEST) <– 0FFFFh end_if	EXT Ireg (00000110)(Ireg)
EXTB	«Знаковое расширение SHORTINTEGER 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 end_if	EXTB wreg (00010110)(wreg)
IDLPD	«Холостой ход/Пониженное потребление». В зависимости от 8-битной величины операнда KEY эта команда выбирает: - вход в режим холостого хода (IDLE) (KEY=1); - вход в режим пониженного потребления (POWERDOWN) (KEY=2); - выполнить последовательность сброса (любое другое значение KEY, не равное 1 или 2) Контроллер шины завершает цикл упреждающей выборки перед остановкой CPU или сбросом. if KEY=1 – режим IDLE, if KEY=2 – режим POWERDOWN, другие значения – сброс.	IDLPD #key (11110110)(key)
INC	«Инкремент слова». Увеличение значение слова операнда на 1. (DEST) <– (DEST)+1	INC wreg (00000111)(wreg)
INCB	«Инкремент байта». Инкрементирует байт операнда на 1. (DEST) <– (DEST)+1	INCB breg (00010111)(breg)

Продолжение таблицы А.2

1	2	3
JBC	«Переход, если бит очищен». 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	«Переход, если бит установлен». 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	«Переход при установленном флаге переноса». Тестирует С-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	«Переход при равенстве нулю флага нулевого результата». Тестирует Z-флаг нулевого результата. Если флаг обнулен, 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 Z=1 then PC <– PC + disp (примечание 1)	JE cadd (11011111)(disp)

Продолжение таблицы А.2

1	2	3
JGE	«Переход, если знак больше или равен». 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)
JGT	«Переход, если знак 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	«Переход, если больше (беззнаковый)». 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	«Переход, если знак меньше или равен». Тестирует N-флаг отрицательного результата и C-флаг переноса. 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)

Продолжение таблицы А.2

1	2	3
JLT	«Переход, если знак меньше». Тестирует N-флаг отрицательного результата. 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)
JNC	«Переход, если флаг переноса очищен». Тестирует C-флаг переноса. 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	«Переход, если не равно». Тестирует Z-флаг нулевого результата. 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 =0 then PC <– PC + disp (примечание 1)	JNE cadd (11010111)(disp)
JNH	«Переход, если не больше (беззнаковый)». Тестирует флаг нулевого результата и флаг переноса. Если флаг переноса установлен, и флаг нулевого результата 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)

Продолжение таблицы А.2

1	2	3
JNST	«Переход, если флаг ST очищен». 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)
JNV	«Переход, если флаг переполнения очищен». Тестирует V-флаг переполнения. 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	«Переход, если дополнительный флаг переполнения очищен». Тестирует флаг 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	«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)

Продолжение таблицы А.2

1	2	3
JV	«Переход, если установлен флаг переполнения». Тестируется V-флаг переполнения. 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)
JVT	«Переход, если дополнительный флаг переполнения установлен». Тестируется флаг VT. 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 VT=1 then PC <– PC + disp (примечание 1)	JST cadd (11011100)(disp)
LCALL	«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 (16 бит)	LCALL cadd (11101111)(disp-low)(disp-high)
LD	«Загрузка слова». Загружает значение слова источника в приёмник. (DEST) <– (SRC)	DEST, SRC LD wreg, waop (101000aa)(waop)(wreg)
LDB	«Загрузка байта». Загружает значение байта источника в приёмник. (DEST) <– (SRC)	DEST, SRC LDB breg, baop (101100aa)(baop)(breg)
LDBSE	«Загрузка байта со знаковым расширением». Расширяет знаком операнд 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)

Продолжение таблицы А.2

1	2	3
LDBZE	«Загрузка байта, расширенного нулём». Значение операнда источника типа BYTE расширяется нулём и загружается в приёмник типа WORD. (low byte DEST) <– (SRC) (high byte DEST) <– 0	DEST, SRC LDBZE wreg, baop (101011aa)(baop)(wreg)
LJMP	«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)	«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)	«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)	«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)	«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)

Продолжение таблицы А.2

1	2	3
MULU (2 операнда)	«Умножение чисел типа WORD, беззнаковое». 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)
MULU (3 операнда)	«Умножение чисел типа WORD, беззнаковое». Перемножает операнды двух источников типа WORD, используя беззнаковую арифметику, и сохраняет 32-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)	«Умножение чисел типа BYTE, беззнаковое». 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)	«Умножение чисел типа BYTE, беззнаковое». 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	«Изменение знака числа типа INTEGER». Изменяет знак операнда типа INTEGER. (DEST) <– (DEST)	NEG wreg (00000011)(wreg)
NEGB	«Изменение знака числа типа SHORT INTEGER». Изменяет знак операнда типа SHORT INTEGER. (DEST) <– (DEST)	NEGB breg (00010011)(breg)
NOP	«Нет операции». Не выполняет никаких действий. Управление переходит к следующей по порядку команде.	NOP (11111101)

Продолжение таблицы А.2

1	2	3
NORML	«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 Z. Команда сохраняет число совершённых сдвигов в приёмнике (правый операнд) (COUNT) <– 0 do while (MSB(DEST) = 0) AND (COUNT) < 31) (DEST) <– (DEST) * 2 (COUNT) <– (COUNT) + 1	SRC, DEST NORML Ireg, breg (00001111)(breg)(Ireg)
NOT	«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	«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	«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, в котором хотя бы один из операндов имеет 1. (DEST) <– (DEST)OR(SRC)	DEST, SRC OR breg, waop (100000aa)(waop)(breg)
ORB	«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, в котором хотя бы один из операндов имеет 1. (DEST) <– (DEST)OR(SRC)	DEST, SRC ORB breg, baop (100100aa)(baop)(breg)
POP	«Чтение слова из стека». Читает слово из вершины стека и помещает его в приёмник. (DEST) <– (SP) SP <– SP + 2	POP waop (110011aa)(waop)

Продолжение таблицы А.2

1	2	3
POPA	«Чтение стека». Эта команда используется взамен POPF для поддержки восьми добавочных прерываний. Она читает два слова из стека и помещает первое слово в пару регистров INT_MASK1/WSR и второе слово – в PSW/INT_MASK. Эта команда инкрементирует указатель стека SP на 4. Вызовы прерываний не могут осуществляться непосредственно после этой команды. INT_MASK1/WSR <– (SP) SP <– SP + 2 PSW/INT_MASK <– (SP) SP <– SP + 2	POPA (11110101)
POPF	«Чтение флагов из стека». Читает слово из вершины стека и помещает его в PSW. Вызовы прерываний не могут осуществляться непосредственно после этой команды. PSW/INT_MASK <– (SP) SP <– SP + 2	POPF (11110011)
PUSH	«Загрузка слова в стек». Загружает операнд типа word в стек. SP <– SP – 2 (SP) <– (DEST)	PUSH waop (110010aa)(waop)
PUSHA	«Загрузка в стек». Эта команда используется взамен PUSHF для поддержки восьми добавочных прерываний. Она загружает два слова из регистровых пар PSW/INT_MASK и INT_MASK1/WSR в стек. Команда очищает вышеуказанные регистры (кроме WSR) и уменьшает указатель стека на 4. Вызовы прерываний не могут осуществляться 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	«Загрузка флагов в стек». Загружает PSW в вершину стека, затем очищает его. Очистка PSW запрещает обслуживание прерываний. Вызовы прерываний не могут осуществляться сразу после этой команды. SP <– SP – 2 (SP) <– PSW/INT_MASK PSW/INT_MASK <– 0	PUSHF (11110010)

Продолжение таблицы А.2

1	2	3
RET	«Возврат из подпрограммы». Загружает программный счётчик (PC) из вершины стека. PC <– (SP) SP <– SP + 2	RET (11110000)
RST	«Сброс системы». Инициализирует PSW в 0, PC в 2080h, а SFRs – в их значения по сбросу. Выполнение этой команды ведёт к тому, что вывод RESET# находится в низком состоянии 16 машинных циклов. SFR <– Reset Status Pin <– Reset Status PSW <– 0 PC <– 2080h	RST (11111111)
SCALL	«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 (11 бит) (примечание 1)	SCALL cadd (00101xxx)(disp-low)
SETC	«Установка флага переноса». Устанавливает флаг переноса. C <– 1	SETC (11111001)
SHL	«Сдвиг слова влево». Сдвигает операнд типа word приёмника влево столько раз, сколько установлено операндом счёта. Значение операнда может быть задано непосредственно в пределах от 0 до 15 (0FH) включительно или как содержимое какого-либо регистра (10H-0FFH) со значением в пределах от 0 до 31 (1FH) включительно. Правые биты результата заполняются нулями. Последний сдвинутый бит хранится во флаге переноса. Temp <– (COUNT) do while Temp ≠ 0 C <– High order bit of (DEST) (DEST) <– (DEST) * 2 Temp <– Temp – 1 end_while	SHL wreg, #count (00001001)(count)(wreg) или SHL wreg, breg (00001001)(breg)(wreg)

Продолжение таблицы А.2

1	2	3
SHLB	«Сдвиг байта влево». Сдвигает операнд типа byte приёмника влево столько раз, сколько установлено операндом счёта. Значение операнда может быть задано непосредственно в пределах от 0 до 15 (0FH) включительно или как содержимое регистра (10H-0FFH) со значением в пределах от 0 до 31 (1FH) включительно. Правые биты результата заполняются нулями. Последний сдвинутый бит хранится во флаге переноса. Temp <– (COUNT) do while Temp ≠ 0 C <– High order bit of (DEST) (DEST) <– (DEST) * 2 Temp <– Temp – 1 end_while	SHLB breg, #count (00011001)(count)(breg) или SHLB breg, breg (00011001)(breg)(breg)
SHLL	«Сдвиг числа типа DOUBLE-WORD влево». Сдвигает операнд типа double-word приёмника влево столько раз, сколько установлено специальным операндом счётчиком. Значение операнда может быть задано непосредственно в пределах от 0 до 15 (0FH) включительно или как содержимое регистра (10H – 0FFH) со значением в пределах от 0 до 31 (1FH) включительно. Правые биты результата заполняются нулями. Последний сдвинутый бит хранится во флаге переноса. Temp <– (COUNT) do while Temp ≠ 0 C <– High order bit of (DEST) (DEST) <– (DEST) * 2 Temp <– Temp – 1 end_while	SHLL Ireg, #count (00001101)(count)(breg) или SHLL Ireg, breg (00001101)(breg)(Ireg)

Продолжение таблицы А.2

1	2	3
SHR	«Логический сдвиг слова вправо». Сдвигает операнд типа word приёмника вправо столько раз, сколько установлено операндом счётчика. Значение операнда может быть задано непосредственно в пределах от 0 до 15 (0FH) включительно или как содержимое какого-либо регистра (10H – 0FFH) со значением в пределах от 0 до 31 (1FH) включительно. Левые биты результата заполняются нулями. Последний сдвинутый бит хранится во флаге переноса. Команда сначала очищает флаг ST. 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 end_while	SHR wreg, #count (00001000)(count)(wreg) или SHR wreg, breg (00001000)(breg)(wreg)
SHRA	«Арифметический сдвиг слова вправо». Сдвигает операнд типа word приёмника вправо столько раз, сколько установлено специальным операндом счёта. Значение операнда может быть задано непосредственно в пределах от 0 до 15 (0FH) включительно или как содержимое какого-либо регистра (10H – 0FFH) со значением в пределах от 0 до 31 (1FH) включительно. Если значение старшего бита было “0”, сдвигаются нули. Если значение было “1”, сдвигаются единицы. Последний сдвинутый бит хранится во флаге переноса. Команда сначала очищает флаг ST. 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 end_while	SHRA wreg, #count (00001010)(count)(wreg) или SHRA wreg, breg (00001010)(breg)(wreg)

Продолжение таблицы А.2

1	2	3
SHRAB	«Арифметический сдвиг байта вправо». Сдвигает операнд типа byte приёмника вправо столько раз, сколько установлено операндом счёта. Значение операнда может быть задано непосредственно в пределах от 0 до 15 (0FH) включительно или как содержимое какого-либо регистра (10H – 0FFH) со значением в пределах от 0 до 31 (1FH) включительно. Если величина старшего бита была “0”, сдвигаются нули. Если величина была “1”, сдвигаются единицы. Последний сдвинутый бит хранится во флаге переноса. Команда сначала очищает флаг ST. 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	SHRAB breg, #count (00011010)(count)(breg) или SHRAB breg, breg (00011010)(breg)(breg)
SHRAL	«Арифметический сдвиг вправо числа типа DOUBLE-WORD». Сдвигает операнд типа double word приёмника вправо столько раз, сколько установлено операндом счёта (правый операнд). Значение операнда может быть задано непосредственно в пределах от 0 до 15 (0FH) включительно или как содержимое какого-либо регистра (10H – 0FFH) со значением в пределах от 0 до 31 (1FH) включительно. Если величина старшего бита была “0”, сдвигаются нули. Если величина была “1”, сдвигаются единицы. Команда сначала очищает флаг ST. 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	SHRAL Ireg, #count (00001110)(count)(Ireg) или SHRAL Ireg, breg (00001110)(breg)(Ireg)

Продолжение таблицы А.2

1	2	3
SHRB	«Логический сдвиг байта вправо». Сдвигает операнд типа byte приёмника вправо столько раз, сколько установлено операндом счёта. Значение операнда может быть задано непосредственно в пределах от 0 до 15 (0FH) включительно или как содержимое какого-либо регистра (10H – 0FFH) со значением в пределах от 0 до 31 (1FH) включительно. Левые биты результата заполняются нулями. Последний сдвинутый бит хранится во флаге переноса. Команда сначала очищает флаг ST. 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	«Логический сдвиг вправо числа типа DOUBLE-WORD». Сдвигает операнд типа double-word приёмника вправо столько раз, сколько установлено специальным операндом счёта. Значение операнда может быть задано непосредственно в пределах от 0 до 15 (0FH) включительно или как содержимое какого-либо регистра регистра (10H – 0FFH) со значением в пределах от 0 до 31 (1FH) включительно. Левые биты результата заполняются нулями. Последний сдвинутый бит хранится во флаге переноса. Команда сначала очищает флаг ST. 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	«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 (11 бит) (примечание 1)	SJMP cadd (00100xxx)(disp-low)

Продолжение таблицы А.2

1	2	3
SKIP	«Нет операции, двухбайтная». Нет действий. Управление передаётся следующей по порядку команде. Это двухбайтная NOP, где второй байт – любая величина, которая просто игнорируется.	SKIP breg (00000000)(breg)
ST	«Сохранение слова». Загружает величину операнда источника типа word (левый операнд) в приёмник (правый операнд). (DEST) <– (SRC)	SRC, DEST ST wreg,waop (110000aa)(waop)(wreg)
STB	«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)	«Вычитание слов». Вычитает операнд источника типа word из операнда приёмника типа word, сохраняет результат в приёмнике и устанавливает флаг переноса как дополнение заёма. (DEST) <– (DEST) – (SRC)	DEST, SRC SUB wreg, waop (011010aa)(waop)(wreg)
SUB (3 операнда)	«Вычитание слов». Вычитает операнд первого источника типа word из операнда второго источника, сохраняет результат в приёмнике и устанавливает флаг переноса как дополнение заёма. (DEST) <– (SRC1) – (SRC2)	DEST, SRC1, SRC2 SUB Dwreg, Swreg, waop (010010aa)(waop)(Swreg)(Dwreg)
SUBB (2 операнда)	«Вычитание байтов». Вычитает операнд источника типа byte из операнда приёмника типа byte, сохраняет результат в приёмнике и устанавливает флаг переноса как дополнение заёма. (DEST) <– (DEST) – (SRC)	DEST, SRC SUBB breg, baop (010110aa)(baop)(breg)
SUBB (3 операнда)	«Вычитание байтов». Вычитает операнд первого источника типа byte из операнда второго источника, сохраняет результат в приёмнике и устанавливает флаг переноса как дополнение заёма. (DEST) <– (SRC1) – (SRC2)	DEST, SRC1, SRC2 SUBB Dbreg, Sbreg, baop (010110aa)(baop)(Sbreg)(Dbreg)
SUBC	«Вычитание слов с заёмом». Вычитает операнд источника типа word из операнда приёмника типа word. Если флаг переноса очищен, SUBC вычитает 1 из результата. Сохраняет результат в приёмнике и устанавливает флаг переноса как дополнение заёма. (DEST) <– (DEST) – (SRC) – (1 – C)	DEST, SRC SUBC wreg, waop (101010aa)(waop)(wreg)

Продолжение таблицы А.2

1	2	3
SUBCB	«Вычитание байтов с заёмом». Вычитает операнд источника типа byte из операнда приёмника типа byte. Если флаг переноса очищен, SUBC вычитает 1 из результата. Сохраняет результат в приёмнике и устанавливает флаг переноса как дополнение заёма. (DEST) <– (DEST) – (SRC) – (1 – C)	DEST, SRC SUBCB breg, baop (101110aa)(baop)(breg)
TIJMP	«Табличный косвенный переход». Вызывает продолжение выполнения программы по адресу, выбранному из таблицы адресов. Первый регистр типа word, TBASE, содержит16-битный адрес начала таблицы. TBASE может быть размещён в регистровом ОЗУ до адреса FEH (нет режима окон) или выше FFH (режим окон). Таблица переходов может размещаться в любых незарезервированных ячейках памяти, выровненных по границе слова. Второй регистр типа word, INDEX, содержит16-битный адрес регистра, который содержит 7-битную величину. Эта величина используется для вычисления смещения в таблице переходов. INDEX размещается в регистровом ОЗУ аналогично TBASE. 16-разрядный адрес в INDEX абсолютен, не изменяется при выполнении команды с использованием режима окон. Байтовый операнд #MASK – 7-битная непосредственная величина для маскирования регистра INDEX. Логическое AND над #MASK и INDEX определяет смещение (OFFSET). OFFSET умножается на 2, выравнивается по границе слова, затем добавляется к базовому адресу (TBASE) для определения адреса приёмника (DEST X). [INDEX] AND #MASK =OFFSET (2*OFFSET)+ TBASE=DEST X PC <– (DEST X)	TIJMP wreg, [wreg], #byte (11100010)(wreg)(#byte)(wreg)

Продолжение таблицы А.2

1	2	3
TRAP	«Программное прерывание». Эта команда осуществляет вызов прерывания, вектор которого расположен в 2010H. Эта операция не влияет на состояние флага разрешения прерываний (I) в PSW. Вызовы прерываний не могут следовать сразу после этой команды. SP <– SP – 2 (SP) <– PC PC <– (2010H) Примечание - Эта команда не поддерживается ассемблером. Команда TRAP предназначена для использования инструментальными средствами разработки. Эти средства могут не поддерживать применение пользователем этой команды.	TRAP (11110111)
XCH	«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	«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)
XOR	«Логическое “исключающее ИЛИ” слов». Выполняет операцию «исключающее ИЛИ» над операндами 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). Выполняет операцию «исключающее ИЛИ» над операндами 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 Смещение (disp) расширяется до 16 бит знаком. 2 В этой операции DEST/2 результат деления без знака. 3 В этой операции DEST/2 результат деления 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 по результату операции.
–	Команда не модифицирует флаг.
↓	Команда сбрасывает флаг как определено, но не устанавливает.
↑	Команда устанавливает флаг как определено, 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
1	2	3	4	5	6	7
ADD, ADDB	+	+	+	+	↑	–
ADDC, ADDCB	↓	+	+	+	↑	–
AND, ANDB	+	+	0	0	–	–
BMOV, BMOVI	–	–	–	–	–	–
BR (Косвенный)	–	–	–	–	–	–
CLR, CLRB	1	0	0	0	–	–
CLRC	–	–	0	–	–	–
CLRVT	–	–	–	–	0	–
CMP, CMPB	+	+	+	+	↑	–
CMPL	+	+	+	+	+	–
DEC, DECB	+	+	+	+	↑	–
DI	–	–	–	–	–	–
DIV,DIVB,DIVU,DIVUB	–	–	–	+	↑	–
DJNZ, DJNZW	–	–	–	–	–	–
DPTS	–	–	–	–	–	–
EI	–	–	–	–	–	–
EPTS	–	–	–	–	–	–
EXT, EXTB	+	+	0	0	–	–
IDLPD Правильный параметр	–	–	–	–	–	–
Неверный параметр	0	0	0	0	0	0
INC	+	+	+	+	↑	0
INCB	+	+	+	+	↑	–

Продолжение таблицы А.5

1	2	3	4	5	6	7
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	0
RET	–	–	–	–	–	–
RST	0	0	0	0	0	0
SCALL	–	–	–	–	–	–
SETC	–	–	1	–	–	–
SHL, SHLB, SHLL	+	?	+	+	↑	–
SHR	+	0	+	0	–	+
SHRA, SHRAB, SHRAL	+	+	+	0	–	+
SHRB, SHRL	+	0	+	0	–	+
SJMP	–	–	–	–	–	–
SKIP	–	–	–	–	–	–
ST, STB	–	–	–	–	–	–
SUB, SUBB	+	+	+	+	↑	–
SUBC, SUBCB	↓	+	+	+	↑	–
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	Продолжается выполнение, если
1	2	3
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

Продолжение таблицы А.6

1	2	3
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 (очищает VT)
JVT	VT = 1 (очищает 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

16-ричный код	Обозначение команды
1	2
00	SKIP
01	CLR
02	NOT
03	NEG
04	XCH
05	DEC
06	EXT
07	INC
08	SHR
09	SHL
0A	SHRA
0B	XCH
0C	SHRL
0D	SHLL
0E	SHRAL
0F	NORML
10	Зарезервирован
11	CLRB
12	NOTB
13	NEGB
14	XCHB
15	DECB
16	EXTB
17	INCB
18	SHRB
19	SHLB
1A	SHRAB
1B	XCHB
1C-1F	Зарезервирован
20-27	SJMP

1	2
28-2F	SCALL
30-37	JBC
38-3F	JBS
40	AND с прямой адресацией (3 операнда)
41	AND с непосредственной адресацией (3 операнда)
42	AND с косвенной адресацией (3 операнда)
43	AND с индексной адресацией (3 операнда)
44	ADD с прямой адресацией (3 операнда)
45	ADD с непосредственной адресацией (3 операнда)
46	ADD с косвенной адресацией (3 операнда)
47	ADD с индексной адресацией (3 операнда)
48	SUB с прямой адресацией (3 операнда)
49	SUB с непосредственной адресацией (3 операнда)
4A	SUB с косвенной адресацией (3 операнда)
4B	SUB с индексной адресацией (3 операнда)
4C	MULU с прямой адресацией (3 операнда)
4D	MULU с непосредственной адресацией (3 операнда)
4E	MULU с косвенной адресацией (3 операнда)
4F	MULU с индексной адресацией (3 операнда)
50	ANDB с прямой адресацией (3 операнда)
51	ANDB с непосредственной адресацией (3 операнда)
52	ANDB с косвенной адресацией (3 операнда)
53	ANDB с индексной адресацией (3 операнда)
54	ADDB с прямой адресацией (3 операнда)
55	ADDB с непосредственной адресацией (3 операнда)
56	ADDB с косвенной адресацией (3 операнда)
57	ADDB с индексной адресацией (3 операнда)
58	SUBB с прямой адресацией (3 операнда)
59	SUBB с непосредственной адресацией (3 операнда)
5A	SUBB с косвенной адресацией (3 операнда)
5B	SUBB с индексной адресацией (3 операнда)
5C	MULUB с прямой адресацией (3 операнда)
5D	MULUB с непосредственной адресацией (3 операнда)
5E	MULUB с косвенной адресацией (3 операнда)
5F	MULUB с индексной адресацией (3 операнда)
60	AND с прямой адресацией (3 операнда)
61	AND с непосредственной адресацией (3 операнда)
62	AND с косвенной адресацией (3 операнда)
63	AND с индексной адресацией (3 операнда)
64	ADD с прямой адресацией (3 операнда)
65	ADD с непосредственной адресацией (3 операнда)
66	ADD с косвенной адресацией (3 операнда)
67	ADD с индексной адресацией (3 операнда)
68	SUB с прямой адресацией (3 операнда)
69	SUB с непосредственной адресацией (3 операнда)
6A	SUB с косвенной адресацией (3 операнда)
6B	SUB с индексной адресацией (3 операнда)
6C	MULU с прямой адресацией (3 операнда)
6D	MULU с непосредственной адресацией (3 операнда)

1	2
6E	MULU с косвенной адресацией (3 операнда)
6F	MULU с индексной адресацией (3 операнда)
70	ANDB с прямой адресацией (3 операнда)
71	ANDB с непосредственной адресацией (3 операнда)
72	ANDB с косвенной адресацией (3 операнда)
73	ANDB с индексной адресацией (3 операнда)
74	ADDB с прямой адресацией (3 операнда)
75	ADDB с непосредственной адресацией (3 операнда)
76	ADDB с косвенной адресацией (3 операнда)
77	ADDB с индексной адресацией (3 операнда)
78	SUBB с прямой адресацией (3 операнда)
79	SUBB с непосредственной адресацией (3 операнда)
7A	SUBB с косвенной адресацией (3 операнда)
7B	SUBB с индексной адресацией (3 операнда)
7C	MULUB с прямой адресацией (3 операнда)
7D	MULUB с непосредственной адресацией (3 операнда)
7E	MULUB с косвенной адресацией (3 операнда)
7F	MULUB с индексной адресацией (3 операнда)
80	OR с прямой адресацией
81	OR с непосредственной адресацией
82	OR с косвенной адресацией
83	OR с индексной адресацией
84	XOR с прямой адресацией
85	XOR с непосредственной адресацией
86	XOR с косвенной адресацией
87	XOR с индексной адресацией
88	CMP с прямой адресацией
89	CMP с непосредственной адресацией
8A	CMP с косвенной адресацией
8B	CMP с индексной адресацией
8C	DIVU с прямой адресацией
8D	DIVU с непосредственной адресацией
8E	DIVU с косвенной адресацией
8F	DIVU с индексной адресацией
90	ORB с прямой адресацией
91	ORB с непосредственной адресацией
92	ORB с косвенной адресацией
93	ORB с индексной адресацией
94	XORB с прямой адресацией
95	XORB с непосредственной адресацией
96	XORB с косвенной адресацией
97	XORB с индексной адресацией
98	CMPB с прямой адресацией
99	CMPB с непосредственной адресацией
9A	CMPB с косвенной адресацией
9B	CMPB с индексной адресацией
9C	DIVUB с прямой адресацией
9D	DIVUB с непосредственной адресацией
9E	DIVUB с косвенной адресацией

1	2
9F	DIVUB с индексной адресацией
A0	LD с прямой адресацией
A1	LD с непосредственной адресацией
A2	LD с косвенной адресацией
A3	LD с индексной адресацией
A4	ADDC с прямой адресацией
A5	ADDC с непосредственной адресацией
A6	ADDC с косвенной адресацией
A7	ADDC с индексной адресацией
A8	SUBC с прямой адресацией
A9	SUBC с непосредственной адресацией
AA	SUBC с косвенной адресацией
AB	SUBC с индексной адресацией
AC	LDBZE с прямой адресацией
AD	LDBZE с непосредственной адресацией
AE	LDBZE с косвенной адресацией
AF	LDBZE с индексной адресацией
B0	LDB с прямой адресацией
B1	LDB с непосредственной адресацией
B2	LDB с косвенной адресацией
B3	LDB с индексной адресацией
B4	ADDCB с прямой адресацией
B5	ADDCB с непосредственной адресацией
B6	ADDCB с косвенной адресацией
B7	ADDCB с индексной адресацией
B8	SUBCB с прямой адресацией
B9	SUBCB с непосредственной адресацией
BA	SUBCB с косвенной адресацией
BB	SUBCB с индексной адресацией
BC	LDBSE с прямой адресацией
BD	LDBSE с непосредственной адресацией
BE	LDBSE с косвенной адресацией
BF	LDBSE с индексной адресацией
C0	ST с прямой адресацией
C1	BMOV
C2	ST с косвенной адресацией
C3	ST с индексной адресацией
C4	STB с прямой адресацией
C5	CMPL
C6	STB с косвенной адресацией
C7	STB с индексной адресацией
C8	PUSH с прямой адресацией
C9	PUSH с непосредственной адресацией
CA	PUSH с косвенной адресацией
CB	PUSH с индексной адресацией
CC	POP с прямой адресацией
CD	BMOVI
CE	POP с косвенной адресацией
CF	POP с индексной адресацией

1	2
D0	JNST
D1	JNH
D2	JGT
D3	JNC
D4	JNVT
D5	JNV
D6	JGE
D7	JNE
D8	JST
D9	JH
DA	JLE
DB	JC
DC	JVT
DD	JV
DE	JLT
DF	JE
E0	DJNZ
E1	FJNZW
E2	TIJMP
E3	BR (Indirect)
E4-E6	Зарезервирован
E7	LJMP
E8-EB	Зарезервирован
EC	DPTS
ED	EPTS
EE	Зарезервирован*
EF	LCALL
F0	RET
F1	Зарезервирован
F2	PUSHF
F3	POPF
F4	PUSHA
F5	POPA
F6	IDLPD
F7	TRAP
F8	CLRC
F9	SETC
FA	DI
FB	EI
FC	CLRVT
FD	NOP
FE	**DIV/DIVB/MUL/MULB
FF	RST

* 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 беззнаковой команды. Например: код MULU (3 операнда) с прямой индексацией “4C”, поэтому код MUL (3 операнда) с прямой индексацией – “FE4C”.

Таблица А.8 – это карта кодов команд микроконтроллеров. Первая цифра кода команды по вертикали, вторая – по горизонтали. Соответствующее обозначение команды показано на пересечении двух чисел. Метод адресации: di – прямая, im – непосредственная, in – косвенная, ix – индексная.

Таблица А.8 – Карта кодов

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							
	бит 0	бит 1	бит 2	бит 3	бит 4	бит 5	бит 6	бит 7
4x	AND (3 операнда)				ADD (3 операнда)			
	di	im	in	ix	di	im	in	ix
5x	ANDB (3 операнда)				ADDB (3 операнда)			
	di	im	in	ix	di	im	in	ix
6x	AND (2 операнда)				ADD (2 операнда)			
	di	im	in	ix	di	im	in	ix
7x	ANDB (2 операнда)				ADDB (2 операнда)			
	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

Код	x8	x9	xA	xB	xC	xD	xE	xF
1	2	3	4	5	6	7	8	9
0x	SHR	SHL	SHRA	XCH in	SHRL	SHLL	SHRAL	NORML
1x	SHRB	SHLB	SHRAB	XCHB in	***	***	***	***
2x	SCALL							
3x	JBS							
	бит 0	бит 1	бит 2	бит 3	бит 4	бит 5	бит 6	бит 7
4x	SUB (3 операнда)				MULU (3 операнда) *			
	di	im	in	ix	di	im	in	ix
5x	SUBB (3 операнда)				MULUB (3 операнда) *			
	di	im	in	ix	di	im	in	ix
6x	SUB (2 операнда)				MULU (2 операнда) *			
	di	im	in	ix	di	im	in	ix

Продолжение таблицы А.8

1	2	3	4	5	6	7	8	9
7x	SUBB (2 операнда)				MULUB (2 операнда) *			
	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 операнда)	3	64	4	65	3	66	4/5	67
ADD (3 операнда)	4	44	5	45	4	46	5/6	47
ADDB (2 операнда)	3	74	3	75	3	76	4/5	77
ADDB (3 операнда)	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
CMP	3	88	4	89	3	8A	4/5	8B
CMPB	3	98	3	99	3	9A	4/5	9B
SUB (2 операнда)	3	68	4	69	3	6A	4/5	6B
SUB (3 операнда)	4	48	5	49	4	4A	5/6	4B
SUBB (2 операнда)	3	78	3	79	3	7A	4/5	7B
SUBB (3 операнда)	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

Продолжение таблицы А.9

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 операнда)	4	FE 6C	5	FE 6D	4	FE 6E	5/6	FE 6F
MUL (3 операнда)	5	FE 4C	6	FE 4D	5	FE 4E	6/7	FE 4F
MULB (2 операнда)	4	FE 7C	4	FE 7D	4	FE 7E	5/6	FE 7F
MULB (3 операнда)	5	FE 5C	5	FE 5D	5	FE 5E	6/7	FE 5F
MULU (2 операнда)	3	6C	4	6D	3	6E	4/5	6F
MULU (3 операнда)	4	4C	5	4D	4	4E	5/6	4F
MULUB(2 операнда)	3	7C	3	7D	3	7E	4/5	7F
MULUB(3 операнда)	4	5C	4	5D	4	5E	5/6	5F
Логические команды								
Обозначение	Прямая		Непосредственная		Косвенная (1)		Индексная S/L (1)	
AND (2 операнда)	3	60	4	61	3	62	4/5	63
AND (3 операнда)	4	40	5	41	4	42	5/6	43
ANDB (2 операнда)	3	70	3	71	3	72	4/4	73
ANDB (3 операнда)	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	3/4	CF
POPA	1	F5						
POPF	1	F3						
PUSH	2	C8	3	C9	2	CA	3/4	CB
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		
LJMP							-/3	E7
SJMP							2/-	20 -27 (2)
TIJMP	4	E2	4	E2			-/4	E2

Продолжение таблицы А.9

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						
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							2	DB
JE							2	DF
JGE							2	D6
JGT							2	D2
JH							2	D9
JLE							2	DA
JLT							2	DE
JNC							2	D3
JNE							2	D7
JNH							2	D1
JNST							2	D0
JNV							2	D5
JNVT							2	D4
JST							2	D8
JV							2	DD
JVT							2	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		

Продолжение таблицы А.9

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						
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	2	01						
CLRB	2	11						
CMPL	3	C5						
DEC	2	05						
DECB	2	15						
EXT	2	06						
EXTB	2	16						
INC	2	07						
INCB	2	17						
NEG	2	03						
NEGB	2	13						
NOT	2	02						
NOTB	2	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 машинных циклах.

Таблица А.10 – Время выполнения команд (в машинных тактах)

Арифметические команды (группа 1)						
Обозначение	Прямая	Непосредственная	Косвенная*		Индексная*	
			Обычная	Автоинкрементная	Короткая	Длинная
ADD (2 операнда)	4	5	6/8	7/9	6/8	7/9
ADD (3 операнда)	5	6	7/10	8/11	7/10	8/11
ADDB (2 операнда)	4	4	6/8	7/9	6/8	7/9
ADDB (3 операнда)	5	5	7/10	8/11	7/10	8/11
ADDC	4	5	6/8	7/9	6/8	7/9
ADDCB	4	4	6/8	7/9	6/8	7/9
CMP	4	5	6/8	7/9	6/8	7/9
CMPB	4	4	6/8	7/9	6/8	7/9
SUB (2 операнда)	4	5	6/8	7/9	6/8	7/9
SUB (3 операнда)	5	6	7/10	8/11	7/10	8/11
SUBB (2 операнда)	4	4	6/8	7/9	6/8	7/9
SUBB (3 операнда)	5	5	7/10	8/11	7/10	8/11
SUBC	4	5	6/8	7/9	6/8	7/9
SUBCB	4	4	6/8	7/9	6/8	7/9
Арифметические команды (группа 2)						
Обозначение	Прямая	Непосредственная	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 операнда)	16	17	18/21	19/22	19/22	20/23
MUL (3 операнда)	16	17	18/21	19/22	19/22	20/23
MULB (2 операнда)	12	12	14/17	15/18	15/18	16/19
MULB (3 операнда)	12	12	14/17	15/18	15/18	16/19
MULU (2 операнда)	14	15	16/19	17/19	17/20	18/21
MULU (3 операнда)	14	15	16/19	17/19	17/20	18/21
MULUB (2 операнда)	10	10	12/15	13/15	12/16	14/17
MULUB (3 операнда)	10	10	12/15	13/15	12/16	14/17
Логические команды						
Обозначение	Прямая	Непосредственная	Косвенная*		Индексная*	
			Обычная	Автоинкрементная	Короткая	Длинная
AND (2 операнда)	4	5	6/8	7/9	6/8	7/9
AND (3 операнда)	5	6	7/10	8/11	7/10	8/11
ANDB (2 операнда)	4	4	6/8	7/9	6/8	7/9
ANDB (3 операнда)	5	5	7/10	8/11	7/10	8/11
OR	4	5	6/8	7/9	6/8	7/9
ORB	4	4	6/8	7/9	6/8	7/9
XOR	4	5	6/8	7/9	6/8	7/9
XORB	4	4	6/8	7/9	6/8	7/9

Продолжение таблицы А.10

Стековые команды (внутренний стек)						
Обозначение	Прямая	Непосредственная	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	Непосредственная	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	Непосредственная	Косвенная*		Индексная*	
			Обычная	Автоинкрементная	Короткая	Длинная
LD	4	5	5/8	6/8	6/9	7/10
LDB	4	4	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	Непосредственная	Косвенная*		Индексная*	
			Обычная	Автоинкрементная	Короткая	Длинная
BR (Indirect)			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	Непосредственная	Косвенная*		Индексная*	
			Обычная	Автоинкрементная	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	Автоинкре-ментная	Корот-кая	Длин-ная
LCALL						13
RET			14			
SCALL					13	
TRAP	18					

Команды условного перехода	
Обозначение	Коротко-индексная адресация
DJNZ	5 (перехода нет), 9 (переход есть)
DJNZW	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)
SHLL	7 + 1 на сдвиг (8 – сдвига нет)

Блочные команды	
Обозначение	Косвенная 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

Продолжение таблицы А.10

Специальные команды						
Обозначение	Прямая	Непосредственная	Косвенная*		Индексная*	
			Обычная	Автоинкрементная	Короткая	Длинная
CLRC	2					
CLRVT	2					
DI	2					
EI	2					
IDLPD неверный параметр правильный параметр	–	 28 12	–	–	–	–
NOP	2					
RST	4					
SETC	2					
SKIP	3					
Команды управления PTS						
Обозначение	Прямая	Непосредственная	Косвенная*		Индексная*	
			Обычная	Автоинкрементная	Короткая	Длинная
DPTS	2					
EPTS	2					
Остальные команды						
Обозначение	Прямая	Непосредственная	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					

* В таблице время выполнения для косвенного и индексного режимов адресации обозначены в формате R/M, где R – время выполнения с использованием SFRs и внутреннего RAM (0H – 1FFH), а M – время выполнения с использованием контроллера памяти (200H – 0FFFFH).

Приложение Б

(обязательное)

Описание сигналов

Это приложение содержит информацию о функциях выводов микроконтроллеров 1867ВЕ86Т и 1867ВЕ16Т.

Б.1 Функциональные группы сигналов

В таблице Б.1 приведен список сигналов, сгруппированных по функциональному назначению.

Таблица Б.1 – Сигналы, сгруппированные по функциональному назначению

Адрес и данные	Управление программирования	Вход/выход	Вход/выход
AD15:0	AINC #	P0.7:0/ACH7:0	P6.5/WG3
	CPVER	P1.0/ACH8	P6.6/PWM0
Управление шины и состояние	PACT#	P1.1/ACH9	P6.7/PWM1
ALE/ADV#	PALE#	P1.2/ACH10/T1CLK	
BHE #/WRH #	PBUS.15:0	P1.3/ACH11/T1DIR	
BW	PMODE.3:0	P1.4/ACH12	
INST	PROG#	P2.3:0/EPA3:0	
READY	PVER	P2.6:4/COMP2:0	
RD#		P2.7/COMP3	
WR #/WRL #	Управление процессора	P3.7:0	
	CLKOUT	P4.7:0	
Питание и земля	EA#	P5.1	
∩0V	EXTINT	P5.7:0	
#VCC1	NMI	P6.0/WG1 #	
VPP	ONCE#	P6.1/WG1	
∩VCC2	RESET#	P6.2/WG2 #	
#0V	BQ1	P6.3/WG2	
	BQ2	P6.4/WG3 #	

Б.2 Описание сигналов

Таблица Б.2 определяет колонки, используемые в таблице Б.3, в которой описываются сигналы.

Таблица Б.2 – Описание колонок таблицы Б.3

Заголовок колонки	Описание
Обозначение	Список обозначений сигналов, расположенных в алфавитном порядке. Многие выводы имеют несколько функций, поэтому в этой колонке сигналов больше, чем выводов.
Тип	Идентифицирует функцию вывода, внесенную в список в колонке Обозначения: вход (I), выход (O), двунаправленный (I/O), питание (PWR) или земля (GND). Все выводы, кроме RESET#, являются фиксирующими входами. RESET# является чувствительным к уровню входом. В режиме POWERDOWN схема использует EXTINT как чувствительный к уровню вход.
Наименование и описание	Кратко описывает функцию определенного сигнала, внесенного в список в колонке «Обозначение». Также перечисляет другие дополнительные функции, мультиплексированные с сигналом.

Таблица Б.3 – Описание сигналов

Обозначение	Тип	Наименование и описание
1	2	3
ACH12:0	I	Аналоговые каналы. Эти выводы – аналоговые входы на АЦ преобразователь. Эти выводы могут использоваться как аналоговые входы (ACHx) или цифровые входы (P0.y). Хотя выводы могут функционировать одновременно как аналоговые и цифровые вводы, это не рекомендуется, так как чтение порта 0, во время преобразования может привести к искажению результатов преобразования. ∩0V и выводы ∩VCC2 должны быть подключены для функционирования АЦП преобразователя и порта 0. ACH12:0 мультиплексированы следующим образом: ACH0/P0.0, ACH1/P0.1, ACH2/P0.2, ACH3/P0.3, ACH4/P0.4/PMODE.0, ACH5/P0.5/PMODE.1, ACH6/P0.6/ PMODE.2, ACH7/P0.7/PMODE.3, ACH8/P1.0, ACH9/P1.1, ACH10/P1.2/T1CLK, ACH11/P1.3/T1DIR и ACH12/P1.4.
AD15:0	I/O	Системная шина адрес/данных. Эти выводы поддерживают мультиплексированную шину адреса/данных. В течение адресной фазы шинного цикла биты адреса 0–15 выводятся на шину и могут быть зафиксированы с помощью сигналов ALE или ADV#. В течение фазы данных на шину выводятся 8-ми или 16-тибитные данные. AD7:0 мультиплексированы с P3.7:0 и PBUS.7:0. AD15:8 мультиплексированы с P4.7:0 и PBUS.15:8.
ADV#	O	Адрес действителен. Этот выходной сигнал с активным низким уровнем устанавливается только во время доступа к внешней памяти. ADV# показывает, что адресная информация действительно находится на шине адрес/данные. Сигнал остается низким пока идет шинный цикл и возвращается в высокий уровень, как только шинный цикл заканчивает. Внешняя защелка может использовать этот сигнал для демультиплексирования адреса из шины адреса/данных. Декодер может также использовать этот сигнал для генерации сигналов «выбор кристалла» внешней памяти. ADV# мультиплексирован с P5.0 и ALE.

Продолжение таблицы Б.3

1	2	3
AINC#	I	Автоинкремент. Во время программирования этот вход с активным низким уровнем сигнала позволяет использовать автоинкрементирование. (Автоинкремент позволяет читать или писать из последовательных ячеек OTPROM, не требуя адреса и через PBUS для каждого чтения или записи.) AINC# устанавливается после того, как каждая ячейка запрограммирована или разгружена. Если AINC# установлен, адрес увеличивается, и следующее слово данных программируется или разгружается. AINC# мультиплексирован с P2.4 и COMP0.
ALE	O	Разрешение записи адреса. Выходной сигнал с активным высоким уровнем устанавливается только во время доступа к внешней памяти. ALE сигнализирует о начале шинного цикла и показывает, что значение адресной информации действительно находится на шине адреса/данных. ALE отличается из ADV# тем, что не остается активным в течение всего шинного цикла. Внешняя защелка может использовать этот сигнал для демультиплексирования адреса из шины адреса/данных. ALE мультиплексировано с P5.0 и ADV#.
∩0V	GND	Аналоговая Земля ∩0V должна быть подключена для работы АЦП, порта 0 и порта 1. Потенциалы ∩0V и #0V должны быть равны.
BHE #	O	Разрешение старшего байта †. Во время 16-битного шинного цикла этот выходной сигнал с активным низким уровнем установлен для записи и чтения слова и старшего байта во внешнюю память. BHE # показывает, что данные действительно передаются по старшим разрядам шины данных. Необходимо использовать BHE# в сочетании с AD0 для определения, какой байт запоминающего устройства передается по системной шине: BHE # AD0 Доступны 0 0 оба байта 0 1 только старший байт 1 0 только младший байт BHE# мультиплексирован с P5.5 и WRH#. † Регистр конфигурации кристалла (CCR0) определяет, функционирует ли этот вывод как BHE# или WRH#. CCR0.2 = 1 выбран BHE#; CCR0.2 = 0 выбран WRH#.
BW	I	Ширина шины. Два бита регистра конфигурации контроллера, CCR0.1 и CCR1.2, наряду с выводом BW, управляют шириной шины данных. Когда оба бита CCR установлены, сигнал BW выбирает ширину шины данных. Если установлен только один бит CCR, ширина шины установлена 8 или 16 бит, и сигнал BW игнорируется. CCR0.1 CCR1.2 BW 0 1 X установлена 8-битная шина данных 1 0 X установлена 16-битная шина данных 1 1 высокий 16-битная шина данных 1 1 низкий 8-битная шина данных BW мультиплексирован с P5.7.

Продолжение таблицы Б.3

1	2	3
CLKOUT	O	Системный тактовый сигнал. Выход внутреннего тактового генератора. Частота CLKOUT равна 1/2 частоты на входе генератора (F_{BQ1}). Скважность сигнала CLKOUT равна 2.
COMP3:0	O	Выходы каналов сравнения процессора событий (EPA). Эти выводы мультиплексированы с другими сигналами и могут иметь конфигурацию стандартного I/O. COMP3:0 мультиплексированы следующим образом: COMP0/P2.4/AINC#, COMP1/P2.5/PACT#, COMP2/P2.6/CPVER, COMP3/P2.7.
CPVER	O	Общая верификация программы. Во время программирования (режим SLAVE) высокий уровень этого сигнала показывает, что все ячейки памяти запрограммированы правильно, низкий сигнал указывает, что произошла ошибка во время хотя бы одного из этапов программирования. CPVER мультиплексирован с P2.6 и COMP2.
EA#	I	Доступ к внешней памяти. Этот сигнал с низким уровнем разрешает доступ к памяти вне кристалла. Если уровень высокий, разрешен доступ к внутренней памяти OTPROM. EA# также контролирует вхождение микроконтроллера в режим программирования. Если EA# равно VPP (обычно + 12,5 В) при нарастающем фронте сигнала RESET#, микроконтроллер входит в режим программирования. Примечание – В промежутке между циклами обращения к внешней памяти шина адреса/данных при неактивном EA# находится в режиме IDLE, порты 3 и 4 можно использовать для стандартного ввода-вывода. При активном EA# эти порты выполняют только функцию шины адрес/данные. EA# фиксируется только по нарастающему фронту сигнала RESET#. Изменение уровня EA# после сброса игнорируется. Необходимо всегда соединять EA# с #0V при использовании микроконтроллера без внутреннего OTPROM.
EPA3:0	I/O	Каналы захвата/сравнения процессора событий (EPA). Высокоскоростные сигналы ввода/вывода для каналов сравнения/захвата EPA. EPA3:0 мультиплексированы следующим образом: EPA0/P2.0/PVER, EPA1/P2.1/PALE#, EPA2/P2.2/PROG#, EPA3/P2.3.
EXTINT	I	Внешнее прерывание. Программируемое прерывание управляется регистром WG_PROTECT. Этот регистр определяет, фиксируется ли фронт сигнала прерывания или уровень, и нарастающим фронтом/высоким уровнем или спадающим фронтом/низким уровнем активируется прерывание. В режиме POWERDOWN сигнал EXTINT должен удерживаться больше 50 нс, чтобы восстановить нормальное выполнение операций. Необходимо при этом не разрешать прерывание. Если выполняется прерывание EXTINT, центральный процессор выполняет программу обслуживания прерывания. В противном случае центральный процессор выполняет инструкцию, которая немедленно следует за командой, вызвавшей энергосберегающий режим. В режиме IDLE установление любого разрешенного прерывания позволяет возобновить нормальное функционирование.

Продолжение таблицы Б.3

1	2	3
INST	O	Выбор команды. Этот сигнал с активным высоким уровнем действует только в цикле обращения к внешней памяти. Если уровень сигнала высокий, INST показывает, что выбирается команда из внешней памяти. Если уровень сигнала низкий, то происходит считывание данных из внутренней памяти. INST мультиплексирован с P5.1.
NMI	I	Немаскируемое прерывание. Нарастающий фронт сигнала NMI производит немаскируемое прерывание. NMI имеет самый высокий приоритет над остальными приоритетами прерываний. Необходимо устанавливать длительность NMI больше одного машинного цикла, чтобы гарантировать его фиксацию.
ONCE#	I	Режим внутрисхемной эмуляции. Удержание ONCE# в низком уровне во время нарастающего фронта RESET#, переводит устройство в режим эмуляции (ONCE). Этот режим устанавливает все выводы, кроме BQ1 и BQ2, в высокоимпедансное состояние, изолируя таким образом устройство от других компонентов системы. Значение ONCE# фиксируют, когда вывод RESET# неактивен. В то время как устройство находится в режиме ONCE, имеется возможность произвести отладку системы, используя схемный эмулятор. Для выхода из режима ONCE, необходимо произвести сброс устройства путем установки низкого уровня сигнала RESET#. Чтобы предотвратить случайный вход в режим ONCE, необходимо сконфигурировать этот вывод как выход или поддерживать его в высоком состоянии в течение сброса, предполагая, что система соответствует U_{IH} спецификации. ONCE# мультиплексирован с P5.4.
P0.7:0	I	Порт 0. Это входной высокоимпедансный порт. Выводы порта 0 нельзя оставить неподключенными. Эти выводы могут использоваться как аналоговые входы (ACHx) или цифровые входы (P0.y). Хотя возможно функционирование выводов одновременно как аналоговые и цифровые входы, это не рекомендуется, так как чтение порта 0 во время преобразования может привести к ненадежным результатам преобразования. ∩0V и ∩VCC2 должны быть подключены для работы порта 0. Порт 0 мультиплексирован следующим образом: P0.0/ACH0, P0.1/ACH1, P0.2/ACH2, P0.3/ACH3, P0.4/ACH4/PMODE.0, P0.5/ACH5/PMODE.1, P0.6/ACH6/PMODE.2, P0.7/ACH7/ PMODE.3.
P1.4:0	I	Порт 1. Это входной высокоимпедансный порт. Выводы порта 1 могут использоваться как аналоговые вводы (ACHx) или цифровые вводы (P1.y). Хотя возможно функционирование выводов одновременно как аналоговые и цифровые входы, это не рекомендуется, так как чтение порта 1 во время преобразования может привести к ненадежным результатам преобразования. ∩0V и ∩VCC2 должны быть подключены для работы порта 1. Порт 1 мультиплексирован следующим образом: P1.0/ACH8, P1.1/ACH9, P1.2/ACH10/T1CLK, P1.3/ACH11/T1DIR, P1.4/ACH12.

Продолжение таблицы Б.3

1	2	3
P2.7:0	I/O	Порт 2 .Это стандартный 8-битный двунаправленный порт, который мультиплексирован с индивидуально выбираемыми сигналами специальной функции. P2.6 мультиплексирован со специальной функцией тестового режима. Если этот вывод удерживается в низком уровне во время сброса, устройство войдет в резервный тестовый режим, поэтому соблюдается осторожность, если этот вывод используется как вход. Если необходима конфигурация этого вывода как входа, надо удерживать его в высоком состоянии в течение сброса и гарантировать, что система соответствует U_{IH} спецификации для предотвращения случайного входа в режим ONCE. Порт 2 мультиплексирован следующим образом: P2.0/EPA0/PVER, P2.1/EPA1/PALE#, P2.2/EPA2/PROG#, P2.3/EPA3, P2.4/COMP0/AINC#, P2.5/COMP1/PACT#, P2.6/COMP2/CPVER и P2.7/COMP3.
P3.7:0	I/O	Порт 3. Это 8-битный двунаправленный порт с выходами. Выводы объединены с мультиплексной шиной адреса/данных, которая имеет комплементарные выходы. P3.7:0 мультиплексированы с AD7:0 и PBUS.7:0.
P4.7:0	I/O	Порт 4. Это 8-битный двунаправленный порт с выходами, программируемыми с открытым стоком или комплементарными. Выводы объединены с мультиплексной шиной адреса/данных, которая имеет комплементарные выходы. P4.7:0 мультиплексированы с AD15:8 и PBUS15:8.
P5.7:0	I/O	Порт 5. Это 8-битный двунаправленный порт, который мультиплексирован с индивидуально выбираемыми сигналами управления. P5.4 мультиплексирован с функцией ONCE#. Если этот вывод удерживать в низком уровне во время сброса, устройство войдет в режим ONCE. Если необходима конфигурация этого вывода как входа, надо удерживать его в высоком состоянии в течение сброса и гарантировать, что система соответствует U_{IH} спецификации для предотвращения случайного входа в режим ONCE. Порт 5 мультиплексирован следующим образом: P5.0/ALE/ADV#, P5.1/INST, P5.2/WR#/WRL#, P5.3/RD#, P5.4/ONCE#, P5.5/BHE#/WRH#, P5.6/READY и P5.7/BW.
P6.7:0	O	Порт 6. Это стандартный 8-битный порт только для вывода, который мультиплексирован со специальными функциями генератора формы сигнала и периферией PWM. Регистр WG_OUT конфигурирует выводы, устанавливает полярность выхода и управляет, будут ли синхронизированы изменения на выходах с событием или происходят немедленно. Порт 6 мультиплексирован следующим образом: P6.0/WG1#, P6.1/WG1, P6.2/WG2#, P6.3/WG2, P6.4/WG3#, P6.5/WG3, P6.6/PWM0 и P6.7/PWM1.

Продолжение таблицы Б.3

1	2	3
PACT#	O	Активное программирование. В режиме автоматического программирования или вывода содержимого памяти (ROM-dump) низкий уровень этого сигнала показывает, что происходит программирование или процесс вывода данных, высокий уровень сигнала указывает, что операция завершена. PACT# мультиплексирован с P2.5 и COMP1.
PALE#	I	Программный ALE. В ходе программирования спадающий фронт этого сигнала заставляет устройство читать команду и адрес из PBUS. PALE# мультиплексирован с P2.1 и EPA1.
PBUS.15:0	I/O	Шина адрес/команда/данные. Во время программирования (режим SLAVE) порты 3 и 4 служат двунаправленными портами с выводами с открытым стоком для передачи команд, адресов и данных к микроконтроллеру или от него. Программирование требует внешних резисторов для поддержки высокого уровня (режим SLAVE). В течение автоматического программирования и ROM-dump, порты 3 и 4 служат системной шиной для доступа к внешней памяти. P4.6 и P4.7 остаются неподключенными; P1.1 и P1.2 являются выводами старших адресов. Программирование (режим SLAVE): PBUS.7:0 мультиплексированы с AD7:0 и P3.7:0. PBUS.15:8 мультиплексированы с AD15:8 и P4.7:0. Автоматическое программирование: PBUS.7:0 мультиплексированы с AD7:0 и P3.7:0. PBUS.13:8 мультиплексированы с AD13:8 и P4.5:0; PBUS15:14 мультиплексированы с P1.2:1.
PMODE.3:0	I	Выбор режима программирования. Определяют режим программирования. PMODE.x фиксируются после сброса устройства и должен быть стабильным во время работы микроконтроллера. PMODE.3:0 мультиплексированы с P0.7:4 и ACH7:4.
PROG#	I	Начало программирования. Во время программирования спадающий фронт этого сигнала фиксирует данные на PBUS и начинает программирование, нарастающий фронт заканчивает программирование. Текущая ячейка памяти программируется теми же самыми данными, пока PROG# остается установленным, поэтому данные на PBUS должны остаться устойчивыми, пока PROG# является активным. При выводе слова спадающий фронт сигнала выводит содержимое ячеек OTPROM на PBUS, в то время как нарастающий фронт заканчивает передачу данных. PROG# мультиплексирован с P2.2 и EPA2.
PVER	O	Верификация программы. В процессе программирования (SLAVE) или автопрограммирования PVER обновляется после каждого импульса программирования. Высокий сигнал вывода указывает на успешное программирование ячейки памяти, низкий сигнал указывает на обнаружение ошибки. PVER мультиплексирован с P2.0 и EPA0.

Продолжение таблицы Б.3

1	2	3
PWM1:0	O	Выводы широтно-импульсного модулятора. Это выходы PWM с повышенной нагрузочной способностью. PWM1:0 мультиплексированы с P6.7:6.
RD#	O	Внешнее чтение. Это выходной сигнал чтения.RD# установлен только в процессе чтения внешней памяти. RD# мультиплексирован с P5.3.
READY	I	Вход готовности. Это вход с активным высоким уровнем сигнала, который наряду с регистрами конфигурации кристалла определяет число циклов ожидания, включенных в шинный цикл. Регистры конфигурации кристалла выбирают максимальное число тактов ожидания (0, 1, 2, 3 или бесконечно большое), которое может быть включено в шинный цикл. Во время низкого уровня READY циклы ожидания включаются в шинный цикл, пока запрограммированное число тактов ожидания не будет достигнуто. Если READY становится высоким уровнем прежде, чем запрограммированное число тактов ожидания достигнуто, никаких дополнительных тактов ожидания не будет включено в шинный цикл. READY мультиплексирован с P5.6.
RESET#	I/O	Вход «сброс» и выход с открытым стоком. Спадающий фронт сигнала RESET# или внутренний сброс инициирует включение транзистора с открытым стоком на выводе RESET# в течение 16 циклов. В режимах POWERDOWN и IDLE установка RESET# заставляет кристалл осуществлять сброс и возвращаться к нормальному рабочему режиму. После сброса устройства первая команда выбирается по адресу FF2080H.
T1CLK	I	Вход внешнего тактирования таймера 1 и вход генератора, задающего скорость передачи для программного последовательного обмена. Таймер 1 увеличивается на 1 (уменьшается на 1) и на нарастающем фронте T1CLK и на спадающем. Используется вместе с T1DIR для квадратурного режима счета. T1CLK мультиплексирован с P1.2 и ACH10.
T1DIR	I	Вход внешнего управления направлением счета таймера 1 (прямое /обратное). Таймер 1 увеличивается на 1 при высоком уровне T1DIR и уменьшается на 1 – при низком. Используется вместе с T1CLK для квадратурного режима счета. T1DIR мультиплексирован с P1.3 и ACH11.
#VCC1	PWR	Напряжение питания цифровой части устройства.
VPP	PWR	Напряжение программирования. Во время программирования напряжение на выводе VPP обычно +12,5 В. Превышение максимального VPP, приведенного в ТУ, может повредить устройство. VPP также управляет выходом устройства из режима POWERDOWN когда удерживается в низком состоянии больше 50 нс. Использовать этот метод для выхода из POWERDOWN можно только при использовании внешнего источника синхроимпульсов. На устройствах без внутреннего OTPROM необходимо соединить VPP с #VCC1.
∩VCC2	PWR	Опорное напряжение для АЦП. Также является напряжением питания для аналоговой части АЦП и логики, используемой для чтения порта 0 и порта 1.

Продолжение таблицы Б.3

1	2	3
#0V	GND	Цифровая схемная земля. Имеется несколько выводов #0V, все они должны быть соединены.
WG3:1	O	Выходы положительных фаз генератора формы сигнала 1–3. 3-фазные выходные сигналы используются в устройствах управления двигателями. WG1 мультиплексирован с P6.1, WG2 мультиплексирован с P6.3 и WG3 мультиплексирован с P6.5.
WG3:1#	O	Выходы отрицательных фаз генератора формы сигнала 1–3. Комплементарные 3-фазные выходные сигналы используются в устройствах управления двигателями.. WG1# мультиплексирован с P6.0, WG2# мультиплексирован с P6.2 и WG3# мультиплексирован с P6.4.
WR#	O	Внешняя запись †. Это выходной сигнал с активным низким уровнем, устанавливающийся при записи во внешнюю память.. WR# мультиплексирован с P5.2 и WRL#. _____ † Регистр конфигурации кристалла (CCR0) определяет, функционирует ли этот вывод как WR# или WRL#. При CCR0.2 = 1 выбран WR#; при CCR0.2 = 0 выбран WRL#.
WRH#	O	Запись старшего байта †. В 16-битном шинном режиме сигнал WRH# с активным низким уровнем устанавливается при записи старшего байта и слова, в 8-битном шинном режиме - при записи старшего и младшего байтов, а также слова. WRH# мультиплексирован с P5.5 и BHE#. _____ † Регистр конфигурации кристалла (CCR0) определяет, функционирует ли этот вывод как BHE# или WRH#. При CCR0.2 = 1 выбран BHE#; при CCR0.2 = 0 выбран WRH#.
WRL#	O	Запись младшего байта †. В 16-битном шинном режиме сигнал WRL# с активным низким уровнем устанавливается при записи младшего байта или слова, в 8-битном шинном режиме – при записи старшего и младшего байтов, а также слова. WRL# мультиплексирован с P5.2 и WR#. _____ † Регистр конфигурации кристалла (CCR0) определяет, функционирует ли этот вывод как WR# или WRL#. При CCR0.2 = 1 выбран WR #; при CCR0.2 = 0 выбран WRL #.
BQ1	I	Вход кварцевого резонатора или внешнего тактового сигнала. Внутренние генераторы синхроимпульсов формируют периферийные синхроимпульсы, синхроимпульсы центрального процессора и сигнал CLKOUT. Если используется внешний генератор вместо внутрикристального, BQ1 служит входом тактов. Внешний сигнал синхроимпульсов должен удовлетворять U_{IH} спецификации для BQ1.
BQ2	O	Выход инвертора внутрикристального генератора. Необходимо оставить BQ2 неподключенным, если используется внешний источник синхроимпульсов.

Б.3 Заданные по умолчанию состояния

Таблица Б.4 определяет символы, которыми обозначено состояние вывода. Для определения норм U_{OL}, U_{IL}, U_{OH} и U_{IH} необходимо обратиться к техническим условиям на микросхемы АЕЯР.431280.496ТУ.

В таблице Б.5 приведены значения сигналов в различных режимах.

Таблица Б.4 – Определение символов состояния

Символ	Определение
0	Напряжение меньше или равное U_{OL}, U_{IL}
1	Напряжение больше или равное U_{OH}, U_{IH}
HiZ	Высокоимпедансное состояние
LoZ0	Низкий импеданс, формирователь «0» с повышенной нагрузочной способностью
LoZ1	Низкий импеданс, формирователь «1» с повышенной нагрузочной способностью
MD0	Средняя поддержка «0»
MD1	Средняя поддержка «1»
WK0	Слабая поддержка «0»
WK1	Слабая поддержка «1»
ODIO	Вход-выход с открытым стоком

Таблица Б.5 – Заданные по умолчанию состояния сигналов

Сигналы порта	Дополни-тельные функции	Во время сброса	После сброса (примечание 14)	IDLE	POWERDOWN
1	2	3	4	5	6
P0.7:0	ACH7:0	HiZ	—	HiZ	HiZ
P1.1:0	ACH9:8	HiZ	—	HiZ	HiZ
P1.2	ACH10/ T1CLK	HiZ	—	HiZ	HiZ
P1.3	ACH11/ T1DIR	HiZ	—	HiZ	HiZ
P1.4	ACH12	HiZ	—	HiZ	HiZ
P2.0	EPA0	WK1 (примечание 1)	WK1	(примеч. 12)	(примеч. 12)
P2.1	EPA1	WK1	WK1	(примеч. 12)	(примеч. 12)
P2.3:2	EPA3:2	WK1	WK1	(примеч. 12)	(примеч. 12)
P2.6:4	COMP2:0	WK1	WK1; кроме P2.5 = LZ (примечание 1)	(примеч. 12)	(примеч. 12)
P2.7	COMP3	WK1	WK1	(примеч. 12)	(примеч. 12)
P3.7:0	AD7:0	WK1	HiZ	(примеч. 3)	(примечание 3)
P4.7:0	AD15:8	WK1	HiZ	(примеч. 3)	(примечание 3)
P5.0	ADV# /ALE	WK1 (примечание 1)	WK1 (примечание 4)	(примеч. 8)	(примечание 8)
P5.1	INST	WK1	WK1	(примеч. 9)	(примечание 9)
P5.2	WR# / WRL#	WK1 (примечание 1)	WK1	(примеч. 10)	(примеч. 10)
P5.3	RD#	WK1 (примечание 1)	WK1 (примечание 4)	(примеч. 10)	(примеч. 10)
P5.4	ONCE#	MD1	LZ (примечание 1)	(примеч. 10)	(примеч. 10)
P5.5	BHE# / WRH#	WK1	WK1 (примечание 4)	(примеч. 10)	(примеч. 10)
P5.6	READY	WK1	(примечание 5)	(примеч. 11)	(примеч. 11)
P5.7	BW	WK1	(примечание 6)	(примеч. 11)	(примеч. 11)
P6.0	WG1#	WK1	WK1	(примеч. 13)	(примеч. 13)
P6.1	WG1	WK1	WK1	(примеч. 13)	(примеч. 13)
P6.2	WG2#	WK1	WK1	(примеч. 13)	(примеч. 13)
P6.3	WG2	WK1	WK1	(примеч. 13)	(примеч. 13)
P6.4	WG3#	WK1	WK1	(примеч. 13)	(примеч. 13)
P6.5	WG3	WK1	WK1	(примеч. 13)	(примеч. 13)
P6.6	PWM0	WK0	—	(примеч. 13)	(примеч. 13)
P6.7	PWM1	WK0	—	(примеч. 13)	(примеч. 13)

Продолжение таблицы В.5

1	2	3	4	5	6
—	CLKOUT	При действующем CLKOUT, LoZ0/1	—	CLKOUT активный, LoZ0/1	LoZ0
—	EA#	HiZ	—	HiZ	HiZ
—	EXTINT	HiZ	—	HiZ	HiZ
—	NMI	WK0	—	WK0	WK0
—	RESET#	LoZ0/HiZ (примечание 2)	—	HiZ	HiZ
—	VPP	HiZ	—	LoZ1	LoZ1
—	BQ1	Osc вход, HiZ	Osc вход, HiZ	Osc вход, HiZ	Osc вход, HiZ
—	BQ2	Osc выход, LoZ0/1	Osc выход, LoZ0/1	Osc выход, LoZ0/1	(примечание 7)

Примечания

1 Эти выводы также управляют тестовым режимом.

2 Если выход RESET = 0, то значение на выводе LoZ0. Если выход RESET =1, то значение на выводе HiZ.

3 Если EA# = 0, Port3 и Port4 = HiZ. Если EA# = 1, Port3 и Порт 4 = ODIO.

4 Если EA# = 1, на выводе WK1. Если EA#= 0, P5.0, P5.3 и P5.5 формируются как выходы, а их функции: ADV#, RD# или BHE # соответственно.

5 Функция READY не выбрана, необходимо три машинных цикла, чтобы осуществить фиксацию ССВ.

6 Функция BW выбрана.

7 Если BQ1 = 1, вывод - LoZ0. Если BQ1 = 0, вывод - LoZ1.

8 Если P5_MODE.0 = 0, порт работает как запрограммирован.
Если P5_MODE.0 = 1 и CCR.3 = 1 (режим ALE), вывод - LoZ0. Если P5_MODE.0 = 1 и CCR.3 = 0 (ADV# режим), вывод - LoZ1.

9 Если P5_MODE.1 = 0, порт работает как запрограммирован.
Если P5_MODE.1 = 1, вывод - LoZ0.

10 Если P5_MODE.y = 0, порт работает как запрограммирован.
Если P5_MODE.y = 1, вывод - LoZ1.

11 Если P5_MODE.y = 0, порт работает как запрограммирован.
Если P5_MODE.y = 1, вывод - HiZ.

12 Если Px_MODE.y = 0, порт работает как запрограммирован.
Если Px_MODE.y = 1, то работа вывода управляется периферийным устройством.

13 Выводы порта в режиме выхода работают как запрограммированы. В режиме специальной функции работа вывода управляется периферийным устройством.

14 Состояния сигналов в этой колонке действительны, пока программное обеспечение не запишет в регистр Px_MODE.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ докум.	Подп.	Дата
	изме-ненных	заме-ненных	новых	анну-лиро-ванных				
-	-	-	все	-	261			04.12.06
1	-	4, 208	-	-	-			19.02.13