МИКРОСХЕМА ИНТЕГРАЛЬНАЯ
1867ВМ9Ф
Руководство пользователя

2014

Содержание

1 Общее описание микросхемы 1867ВМ9Ф

1.1 Введение

Интегральная микросхема 1867ВМ9Ф (далее – ИС или микросхема) – это 32-разрядный процессор с плавающей запятой, содержащий набор инструкций, которые ориентированы на эффективную цифровую обработку сигналов (далее – ЦОС) и оптимизированы для реализации параллельных вычислений.

Микросхема совмещает высокую производительность центрального процессорного устройства (далее – ЦПУ) и сопроцессора прямого доступа к памяти (далее – ПДП) с шестью коммуникационными портами, которые предназначены для работы в мультипроцессорной системе и для реализации высокоскоростного ввода-вывода с периферийными устройствами.

Микросхема 1867ВМ9Ф имеет встроенный модуль анализа, который поддерживает аппаратные точки останова для разработки и отладки параллельных программ на эмуляторе. Набор инструкций совместим с семейством инструкций ИС 1867ВЦ6Ф, однако микросхема 1867ВМ9Ф поддерживает дополнительный набор инструкций, повышающих эффективность вычислений и представленных в приложении А «Инструкции языка Ассемблер».

1.2 Описание микросхемы 1867ВМ9Ф

Микросхема 1867ВМ9Ф имеет 13 модификаций. В таблице 1.1 приведены технические характеристики модификаций.

Центральное процессорное устройство обеспечивает производительность до 20 миллионов операций в секунду при работе с фиксированной запятой и до 40 миллионов операций в секунду при работе с плавающей запятой.

ИС 1867ВМ9Ф содержит 8К сверхоперативной памяти (далее – СОЗУ), 128 слов КЭШ-памяти команд (далее – КЭШ команд) и постоянное запоминающее устройство (далее – ПЗУ), в котором хранится начальный загрузчик программ.

Две внешние шины обеспечивают адресацию к 4096М словам непрерывного адресного пространства. Микросхема 1867ВМ9Ф выпускается в 325 выводном металлокерамическом корпусе типа CPGA-325B.

Ключевые характеристики ИС 1867ВМ9Ф:

- 20 миллионов операций в секунду при работе с фиксированной запятой и 40 миллионов операций в секунду при работе с плавающей запятой;
- аппаратная поддержка преобразования числа с плавающей запятой из собственного формата в число с плавающей запятой в формате IEEE;
- однотактный цикл выполнения инструкций при операциях с байтами, полусловами и словами;
- наличие регистров общего назначения;
- аппаратная поддержка деления, вычисления обратной величины и извлечения квадратного корня;
- две внешние шины (глобальная и локальная), обеспечивающие адресацию к 4 Гбайт слов непрерывного адресного пространства;
- два 32-разрядных таймера;
- 6 каналов сопроцессора ПДП;
- 6 коммуникационных портов для организации мультипроцессорных систем и работы с периферийными устройствами;
- режим IDLE для уменьшения энергопотребления.

1.3 Основные технические характеристики микросхемы

В таблице 1.1 приведены основные технические характеристики ИС 1867ВМ9Ф.

Таблица 1.1 – Основные технические характеристики ИС 1867ВМ9Ф

Наименование параметра, единица измерения	Значение параметра
Разрядность АЛУ, бит: - плавающая запятая - фиксированная запятая	 40 32
Разрядность процессора, бит	32
Производительность на операциях с плавающей запятой, MFLOPS	40
Производительность на операциях с фиксированной запятой, MIPS	20
Объем ПЗУ, Кбайт	32
Объем СОЗУ, Кбайт	8
Объем непрерывно адресуемой памяти, Гбайт	16
Число 8 битных коммуникационных портов	6
Число каналов в сопроцессоре ПДП	6
Число 32-разрядных таймеров	2
Напряжение питания ядра, В	3,3
Напряжение питания буферов ввода/вывода, В	3,3
Тактовая частота ядра процессора, МГц	40

Временные параметры сигналов микросхемы 1867ВМ9Ф представлены в таблицах Б.1 – Б.19 и на рисунках Б.1 – Б.21 приложения Б.

Электрические параметры микросхемы представлены в приложении В.

Значения параметров, приведенные в техническом описании КФДЛ.431282.006ТО, являются справочными.

Все значения электрических параметров ИС 1867ВМ9Ф приведены в АЕЯР.431280.902ТУ.

Условное графическое обозначение микросхемы 1867ВМ9Ф приведено в приложении Г.

Функциональное назначение выводов ИС приведено в приложении Д.

Микросхема 1867ВМ9Ф разработана в металлокерамическом корпусе CPGA-325B, габаритные и присоединительные размеры которого приведены в приложении Е.

2 Описание архитектуры

Высокая производительность ИС 1867ВМ9Ф достигнута за счет большой точности и широкого динамического диапазона вычислительного блока с плавающей запятой, двухпортового СОЗУ, высокой степени параллелизма работы коммуникационных портов, сопроцессора ПДП и ЦПУ.

В данном разделе дается описание архитектуры ИС 1867ВМ9Ф.

На рисунке 2.1 приведена структурная схема ИС 1867ВМ9Ф.

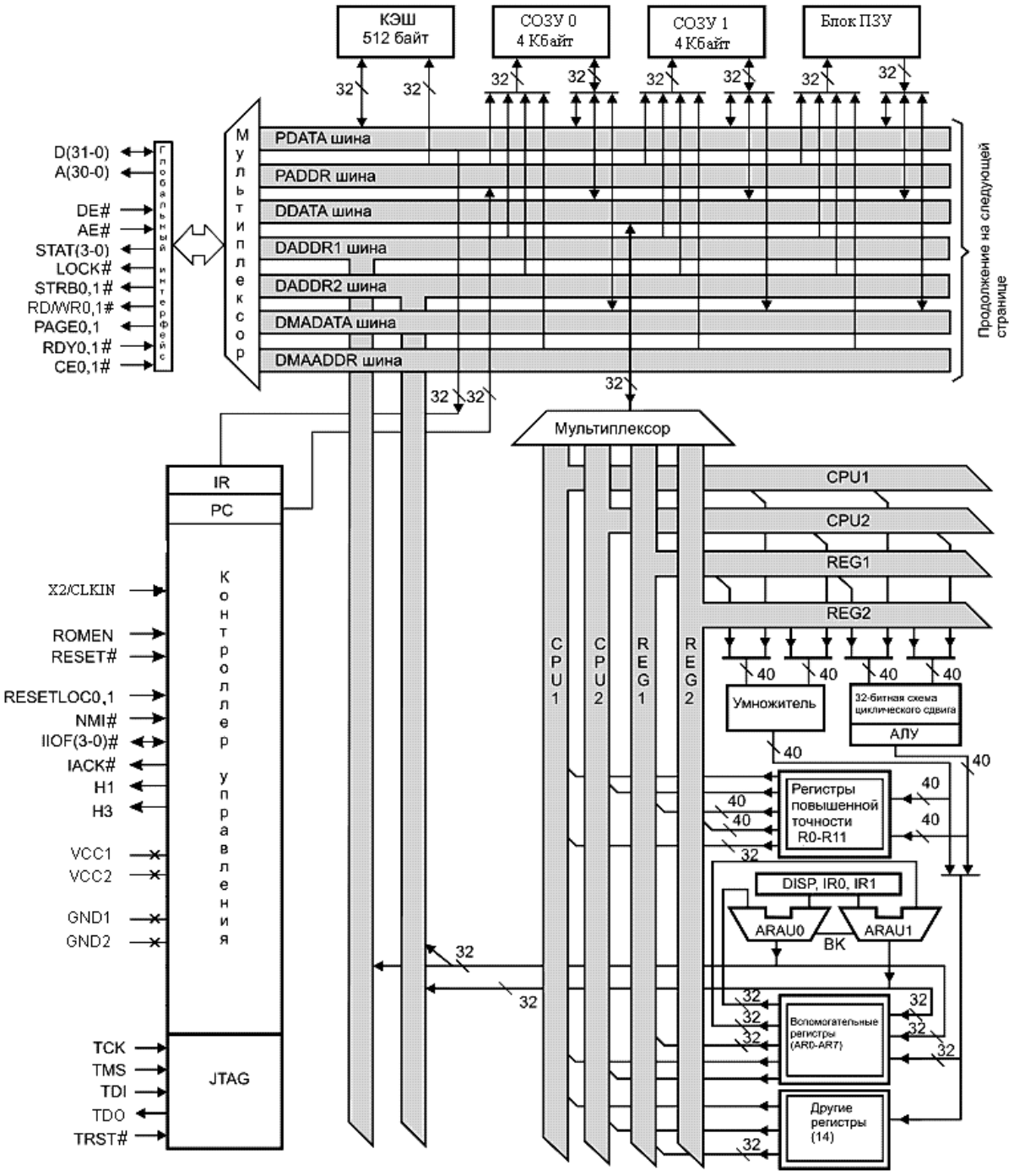


Рисунок 2.1, лист 1 – Структурная схема ИС 1867ВМ9Ф

Начало на предыдущей странице

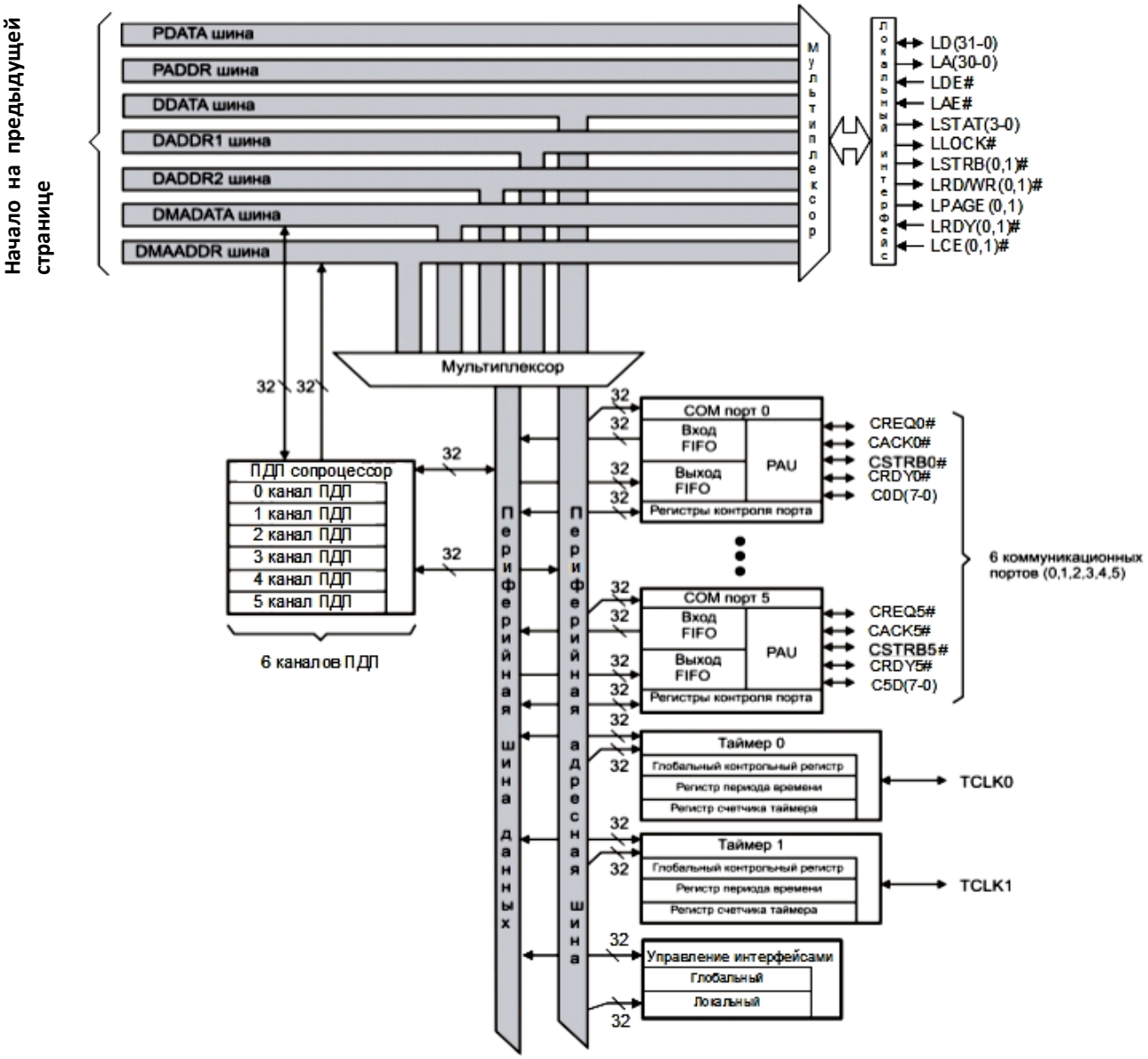


Рисунок 2.1, лист 2

2.1 Центральное процессорное устройство ЦПУ

Архитектура ЦПУ микросхемы 1867ВМ9Ф основана на работе с регистрами. ЦПУ можно разделить на несколько функционально законченных блоков:

- умножитель;
- арифметико-логическое устройство АЛУ;
- внутренние шины CPU1/CPU2 и REG1/REG2;
- арифметическое устройство для выполнения арифметических действий над вспомогательными регистрами ARAUs;
- регистры ЦПУ, представленные в таблице 2.1.

На рисунке 2.2 приведены основные функциональные блоки ЦПУ.

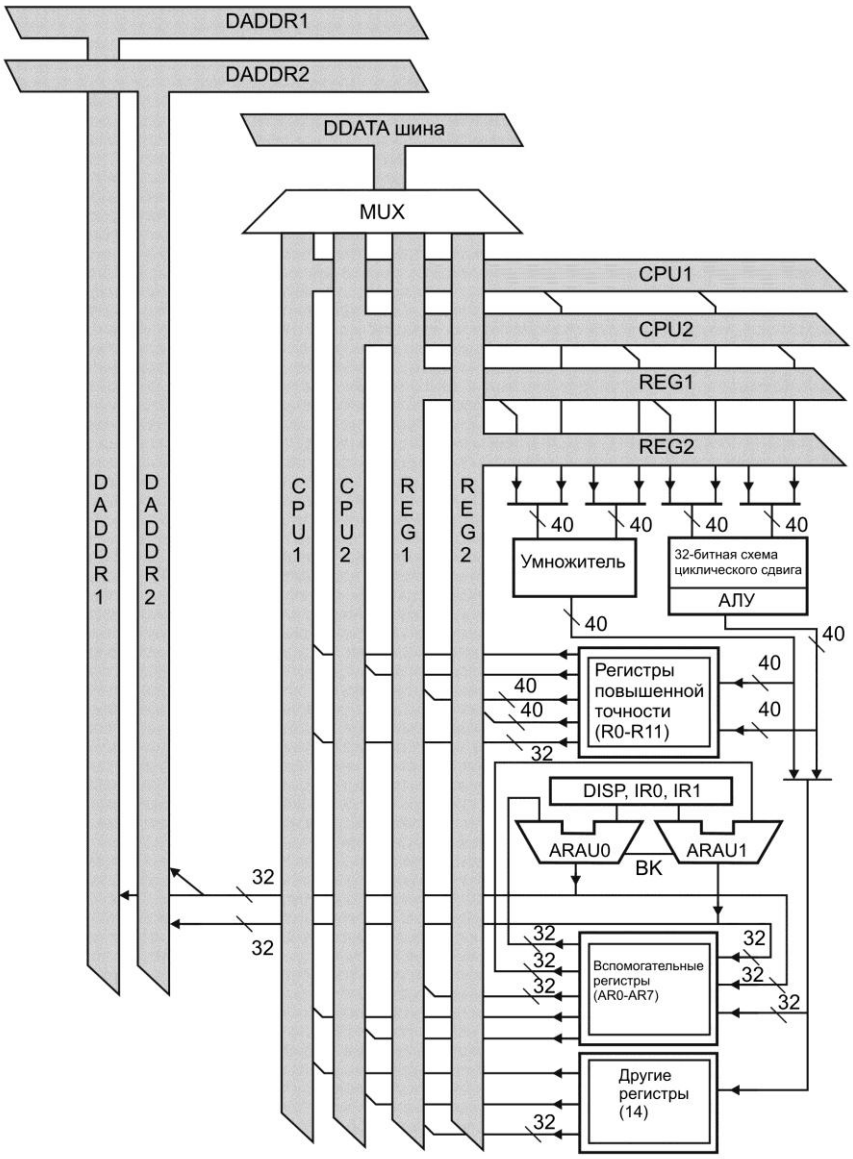


Рисунок 2.2 – Структура центрального процессорного устройства ИС 1867ВМ9

2.1.1 Умножитель

Умножитель выполняет умножение над 32-разрядными целыми числами и 40-разрядными числами с плавающей запятой за один машинный такт (цикл).

Реализация в ИС арифметики с плавающей запятой позволяет выполнять операции с плавающей запятой с такой же скоростью, как и для операций с фиксированной запятой, и с высокой степенью параллелизма. Для получения большей производительности операции АЛУ и умножителя могут быть выполнены за один цикл при использовании параллельных инструкций.

При выполнении операции умножения с плавающей запятой на вход умножителя подаются 40-разрядные числа в формате с плавающей запятой, и результатом являются 40-разрядные значения в формате с плавающей запятой.

При выполнении целочисленного умножения на вход умножителя подаются 32-разрядные целые числа, а результатом являются либо старшие 32 разряда, либо младшие 32 разряда произведения.

2.1.2 Арифметико-логическое устройство АЛУ

АЛУ выполняет одноцикловые операции с 32-разрядными числами в формате с фиксированной запятой (целыми числами), 32-разрядными логическими и 40-разрядными числами в формате с плавающей запятой, включая одноцикловые целочисленные преобразования и преобразования чисел с плавающей запятой из одного формата в другой. Результат работы АЛУ всегда сохраняется или в 32-разрядном целом формате, или в 40-разрядном формате с плавающей запятой. Циклический сдвигатель используется для сдвига числа влево или вправо до 32 бит за один цикл.

Четыре внутренние шины CPU1/CPU2 и REG1/REG2 предназначены для передачи двух операндов из памяти и двух операндов из регистрового файла, обеспечивая, таким образом, параллельное умножение и сложение/вычитание двух пар целых чисел или чисел в формате с плавающей запятой в одном цикле.

2.1.3 Арифметическое устройство для выполнения операций над вспомогательными регистрами ARAUs

Два арифметических устройства над вспомогательными регистрами ARAU0 и ARAU1 могут вычислять два адреса в одном цикле. ARAU функционирует параллельно с умножителем и АЛУ. Они обеспечивают вычисление адреса операнда со смещением, с использование индексных регистров IR0 и IR1, циклическую и бит-реверсную адресацию операндов.

2.1.4 Основные регистры ЦПУ

Микросхема имеет 32 регистра в регистровом файле, который тесно связан с ЦПУ. Все эти регистры могут быть операндами умножителя и АЛУ и могут быть использованы как регистры общего назначения. Однако эти регистры также имеют некоторые специальные функции, например, двенадцать регистров повышенной точности специально предназначены для хранения результатов повышенной точности с плавающей запятой. Восемь вспомогательных регистров поддерживают различные методы косвенной адресации и могут быть использованы как 32-разрядные регистры общего назначения для хранения целочисленных и логических значений. Оставшиеся регистры обеспечивают системные функции, такие как адресация, управление стеком, состояние процессора, прерывания и повторения блока инструкций.

В разделе 3 «Регистры ЦПУ» представлена более детальная информация, примеры управления стеком и использования регистров.

Регистры ЦПУ, выполняемые ими функции и ссылки на более детальную информацию приведены в таблице 2.1.

Таблица 2.1 – Основные регистры ЦПУ

Синтаксис Ассемблера	Соответствующее имя функции	Ссылка в РП
R0	Регистр повышенной точности 0	3.1.1
R1	Регистр повышенной точности 1	3.1.1
R2	Регистр повышенной точности 2	3.1.1
R3	Регистр повышенной точности 3	3.1.1
R4	Регистр повышенной точности 4	3.1.1
R5	Регистр повышенной точности 5	3.1.1
R6	Регистр повышенной точности 6	3.1.1
R7	Регистр повышенной точности 7	3.1.1
R8	Регистр повышенной точности 8	3.1.1
R9	Регистр повышенной точности 9	3.1.1
R10	Регистр повышенной точности 10	3.1.1

Окончание таблицы 2.1

Синтаксис Ассемблера	Соответствующее имя функции	Ссылка в РП
R11	Регистр повышенной точности 11	3.1.1
AR0	Вспомогательный регистр 0	3.1.2
AR1	Вспомогательный регистр 1	3.1.2
AR2	Вспомогательный регистр 2	3.1.2
AR3	Вспомогательный регистр 3	3.1.2
AR4	Вспомогательный регистр 4	3.1.2
AR5	Вспомогательный регистр 5	3.1.2
AR6	Вспомогательный регистр 6	3.1.2
AR7	Вспомогательный регистр 7	3.1.2
DP	Указатель страницы данных	3.1.3
IR0	Индексный регистр 0	3.1.4
IR1	Индексный регистр 1	3.1.4
BK	Регистр размера блока повторения	3.1.5
SP	Указатель системного стека	3.1.6
ST	Регистр состояния	3.1.7
DIE	Регистр разрешения прерывания сопроцессора ПДП	3.1.8
IIE	Регистр разрешения внутренних прерываний ЦПУ	3.1.9
IIF	Регистр флагов ввода-вывода и прерывания IIOF	3.1.10
RS	Регистр адреса начала блока повторения	3.1.11
RE	Регистр адреса конца блока повторения	3.1.11
RC	Счетчик повторений	3.1.11

Регистры повышенной точности R0-R11 имеют возможность хранения и выполнения операций с 32-разрядными целыми и 40-разрядными значениями с плавающей запятой. Любая инструкция, содержащая значения с плавающей запятой, использует разряды с 39 по 0. Если операнды являются целыми числами со знаком или беззнаковыми, то используются только разряды с 31 по 0, разряды 39-32 остаются без изменений. Это также относится ко всем операциям сдвига. В разделе 5 «Форматы данных и операции с плавающей запятой» указаны форматы регистров повышенной точности для значений с плавающей запятой и целочисленных значений.

32-разрядные вспомогательные регистры AR0-AR7 могут быть доступны из ЦПУ и модифицированы двумя арифметическими устройствами вспомогательных регистров ARAU. Основной функцией вспомогательного регистра является генерация 31-разрядных адресов операндов. Они также могут быть использованы для выполнения различных функций, таких как циклические счетчики или как 32-разрядные регистры общего пользования, которые могут быть модифицированы умножителем или АЛУ. В разделе 6 «Методы адресации» дана подробная информация и примеры использования вспомогательных регистров AR0-AR7 для вычисления адреса операнда.

Указатель страницы данных DP является 32-разрядным регистром. Шестнадцать младших разрядов регистра указателя страницы данных используются в режиме прямой адресации как указатели на адресуемую страницу данных. Можно адресовать до 64К страниц и каждая страница содержит 64К слов. Подробнее регистр DP описан в 3.1.3.

32-разрядные индексные регистры IR0 и IR1 используются арифметическим устройством вспомогательных регистров ARAU для вычисления индексированного адреса. В разделе 6 «Методы адресации» приведены примеры использования индексных регистров IR0 и IR1 для адресации операндов.

32-разрядный регистр размера блока повторения BK используется ARAU при циклической адресации для определения размера блока данных. Подробнее регистр BK описан в 3.1.5.

Указатель системного стека SP – это 32-разрядный регистр, содержащий адрес вершины системного стека. Регистр SP всегда указывает на последний элемент, записанный в стек. При проталкивании данных в стек происходит предекрементирование (увеличение на «1»), при выталкивании данных из стека выполняется постдекрементирование (уменьшение на «1») указателя системного стека. Регистр указателя системного стека SP изменяется (модифицируется) прерываниями, переходами, вызовами, возвратами и инструкциями PUSH и POP. Подробнее регистр SP описан в 3.1.6.

Регистр состояния ST содержит глобальную информацию, относящуюся к состоянию ЦПУ. Обычно операции выставляют флаги условий в этом регистре состояния в соответствии с полученным этой операцией результатом: нулевым, отрицательным и т. п., включающие операции загрузки, сохранения регистров и арифметические и логические операции. Когда регистр состояния загружается инструкцией загрузки, то осуществляется замена каждого разряда регистра состояния ST на разряд операнда независимо от значения операнда инструкции загрузки. Следовательно, при следующей загрузке содержимое регистра состояния равно содержимому операнда источника инструкции загрузки. Это позволяет легко сохранять и восстанавливать регистр состояния. Подробнее статусный регистр ST описан в 3.1.7.

Регистр разрешения прерывания DIE сопроцессора ПДП – это 32-битный регистр, содержащий 2- и 3-битные поля для указания прерываний, которые синхронизируют работу каждого из шести каналов ПДП с работой ЦПУ. Это позволяет каждому каналу ПДП работать независимо от работы основного ЦПУ. Каждый канал ПДП также может быть синхронизирован внешними прерываниями или встроенными в кристалл таймерами. Подробнее регистр DIE описан в 3.1.8.

Регистр разрешения внутренних прерываний ЦПУ IIE – это 32-битный регистр, который либо разрешает, либо запрещает прерывания ЦПУ от шести коммуникационных портов, двух таймеров и шести каналов ПДП. Подробнее регистр IIE описан в 3.1.9.

Регистр флагов ввода-вывода и прерывания ЦПУ IIF является также 32-разрядным регистром. Единица в одном бите или битах регистра IIF показывает, что соответствующие запросы прерываний установлены (поступили). Ноль показывает, что соответствующие прерывания не установлены. Регистр IIF управляет четырьмя внешними выводами IIOF3#-IIOF0#, выполняющими либо функции ввода-вывода общего назначения, либо функции входов внешних прерываний. Эти выводы могут быть программно установлены в качестве порта ввода-вывода общего назначения (т. е. через них могут быть считаны или записаны значения) или в качестве входов внешних прерываний. Получить детальную информацию о регистре IIF можно в 3.1.10.

Счетчик повторений RC – это 32-разрядный регистр, используемый для точного определения того, сколько раз должен быть повторен блок инструкций при выполнении операций повторения блока. При работе в режиме повторения 32-разрядный регистр начального адреса повторений RS содержит начальный адрес блока памяти программы, который будет повторяться, а 32-разрядный регистр адреса конца повторений RE содержит конечный адрес блока повторений. Более детальная информация о регистрах RC, RS, RE представлена в 3.1.11.

Счетчик инструкций (команд) PC – это 32-разрядный регистр, содержащий адрес следующей выбираемой инструкций. Хотя регистр PC не является частью регистрового файла ЦПУ, но он может быть изменен командами, которые изменяют

последовательность выполнения программы. Получить детальную информацию о счетчике инструкций PC можно в 3.1.12.

2.1.5 Дополнительные регистры ЦПУ

Кроме основного регистрового файла, ЦПУ содержит дополнительный регистровый файл, который состоит из двух специальных регистров, выполняющих функции указателей:

- регистр IVTP – это указатель адреса таблицы векторов системных прерываний, который определяет адреса векторов для всех системных прерываний;
- регистр TVTP – это указатель на таблицу векторов программных прерываний TRAP.

Эти два регистра полностью описаны в 3.2 «Дополнительный регистровый файл ЦПУ».

2.2 Организация памяти

Общее адресное пространство ИС 1867ВМ9Ф составляет 4096К (4 Гбайт) 32-разрядных слов. Программа, данные и область ввода-вывода содержатся внутри этого 4 Гбайт (пословно адресуемого) адресного пространства, обеспечив, таким образом, хранение таблиц, коэффициентов, кода программы или данных либо и/или в СОЗУ, либо и/или во внутрикристальном ПЗУ, подключенном к локальной/глобальной шине, либо и/или во внешнем ОЗУ/ПЗУ, подключенным к локальной/глобальной шине. В этом случае объем памяти может быть максимальным, и область памяти распределяется так, как это необходимо. Управляя одним выводом ROMEN можно выбирать место расположения области памяти от 0000 0000h до 000F FFFFh либо во внешнем ОЗУ (ROMEN = 0), либо во внутрикристальном ПЗУ (ROMEN = 1).

2.2.1 Сверхоперативное ОЗУ, ПЗУ, КЭШ

На рисунке 2.3 показана организация памяти ИС 1867ВМ9Ф. Микросхема имеет два блока сверхоперативной памяти СОЗУ 0 и СОЗУ 1, ПЗУ. Объем каждого блока СОЗУ составляет 1К × 32 бит, объем блока ПЗУ составляет 4К × 32 бит.

Раздельные шины инструкций, данных и контроллера ПДП обеспечивают параллельную (в одном цикле) выборку кода инструкций, чтение, запись данных и работу контроллера ПДП. Например, ЦПУ может обращаться в одном цикле к двум значениям данных в одном блоке СОЗУ или обращаться к внешнему интерфейсу параллельно с загрузкой сопроцессором ПДП к другому блоку ОЗУ.

КЭШ команд размером 128 × 32 бит обеспечивает кеширование кода программы, что значительно уменьшает число внекристальных обращений. Это позволяет увеличить производительность, так как КЭШ команд поддерживает два обращения за цикл, в то время как внешний интерфейс поддерживают одно обращение за цикл. При этом шины внешнего интерфейса свободны для использования сопроцессором ПДП для выборки из внешней внекристальной памяти, а также для использования другими устройствами в системе.

В разделе 4 «Память и КЭШ команд» приведена более подробная информация о памяти и командах КЭШ.

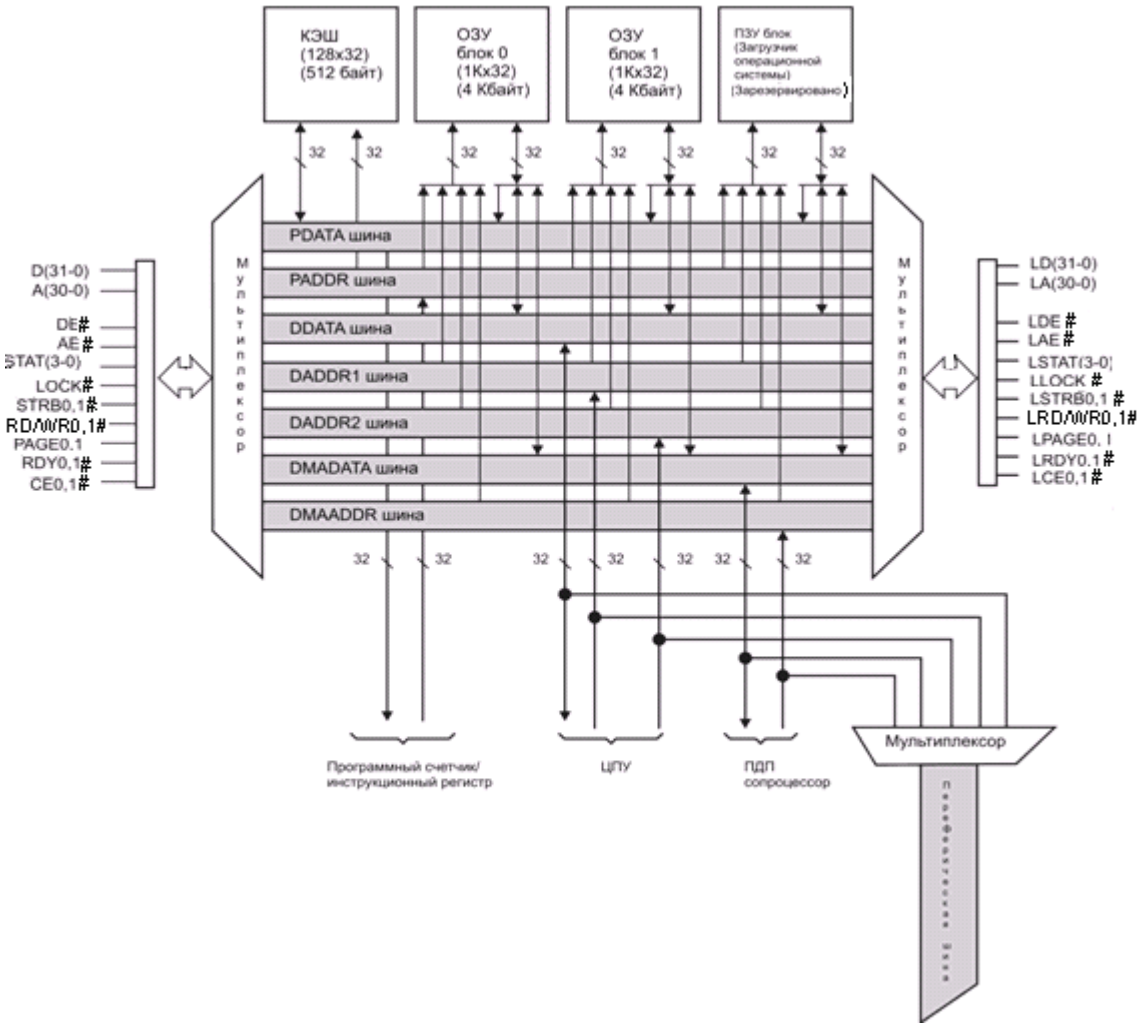


Рисунок 2.3 – Организация памяти

2.2.2 Карта памяти

Карта памяти ИС приведена на рисунке 2.4.

В зависимости от значения внешнего вывода ROMEN карта памяти может иметь разные конфигурации:

- если ROMEN равен единице, то область адресного пространства с 0000 0000h по 000F FFFFh зарезервирована для работы внутреннего ПЗУ, и процессор работает в режиме микрокомпьютера, что показано с правой стороны рисунка 2.4;
- если ROMEN равен нулю, то область адресного пространства с 0000 0000h по 000F FFFFh доступна локальной шине, и процессор работает в режиме микропроцессора. Это показано с левой стороны рисунка 2.4.

Остальная часть карты памяти ЦОС остаётся неизменной и не зависит от значения сигнала ROMEN.

> БУДЬТЕ ВНИМАТЕЛЬНЫ! Обращение к зарезервированной области памяти может привести к непредсказуемым результатам.

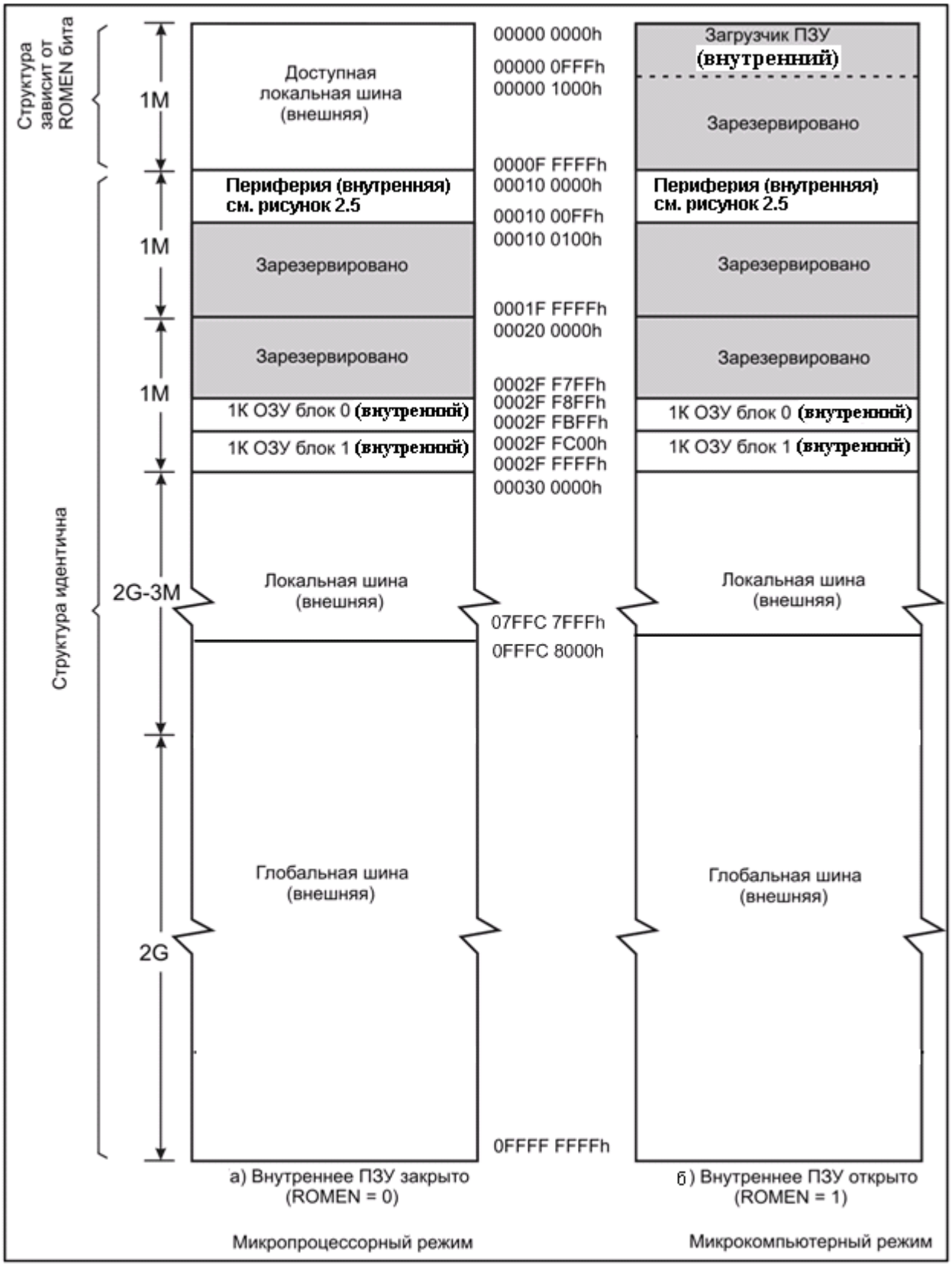


Рисунок 2.4 – Карта памяти

В таблице 2.2 представлена карта памяти регистров периферийных устройств и ссылка на более детальную информацию.

Таблица 2.2 – Карта памяти регистров периферийных устройств

Адрес	Периферийное устройство	Ссылка в РП
0010 0000h 0010 000Fh	Регистры управления локальной и глобальной шинами (16 слов)	4.2.1, рисунок 4.2
0010 0010h 0010 001Fh	Регистры управления работой процессора (16 слов)	4.2.2
0010 0020h 0010 002Fh	Регистры таймера 0 (16 слов)	4.2.3, рисунок 4.3
0010 0030h 0010 003Fh	Регистры таймера 1 (16 слов)	
0010 0040h 0010 004Fh	Регистры коммуникационного порта 0 (16 слов)	4.2.4, рисунок 4.4
0010 0050h 0010 005Fh	Регистры коммуникационного порта 1 (16 слов)	
0010 0060h 0010 006Fh	Регистры коммуникационного порта 2 (16 слов)	
0010 0070h 0010 007Fh	Регистры коммуникационного порта 3 (16 слов)	
0010 0080h 0010 008Fh	Регистры коммуникационного порта 4 (16 слов)	
0010 0090h 0010 009Fh	Регистры коммуникационного порта 5 (16 слов)	
0010 00A0h 0010 00AFh	Регистры 0 канала сопроцессора ПДП (16 слов)	4.2.5, рисунок 4.5
0010 00B0h 0010 00BFh	Регистры 1 канала сопроцессора ПДП (16 слов)	
0010 00C0h 0010 00CFh	Регистры 2 канала сопроцессора ПДП (16 слов)	
0010 00D0h 0010 00DFh	Регистры 3 канала сопроцессора ПДП (16 слов)	
0010 00E0h 0010 00EFh	Регистры 4 канала сопроцессора ПДП (16 слов)	
0010 00F0h 0010 00FFh	Регистры 5 канала сопроцессора ПДП (16 слов)	

2.2.3 Методы адресации памяти

ИС реализует базовый набор инструкций общего назначения, такие как арифметические инструкции, которые ориентированы на цифровую обработку сигналов и другие приложения, требующие объемных числовых вычислений. В разделе 6 «Методы адресации» дана более подробная информация об адресации памяти.

Процессор реализует четыре группы методов адресации. Внутри каждой группы могут быть использованы шесть типов адресации.

Первая группа использует следующие основные виды адресации:

- регистровая адресация – операнд является регистром ЦПУ;
- короткая непосредственная адресация – операнд является 16-разрядным непосредственным значением;
- прямая адресация – операнд является содержимым 32-битного адреса, полученного конкатенацией 16 старших бит регистра DP и 16 младших бит кода инструкции;
- косвенная адресация – вспомогательный регистр указывает адрес операнда.

Вторая группа использует трехоперандные методы адресации:

- регистровый метод, такой как для основного режима адресации;
- косвенный метод, такой как для основного режима адресации;
- непосредственный метод – операнд является 8-разрядным непосредственным значением.

Третья группа использует режимы параллельной адресации:

- регистровый режим – операнд является регистром повышенной точности;
- косвенный режим, такой как для основного режима адресации.

Четвертая группа использует режимы адресации относительно программного счетчика:

- регистровый режим, такой как для основного режима адресации;
- относительно счетчика инструкций PC – 16 разрядов со знаком или 24 разряда смещения прибавляется к содержимому PC.

2.3 Внутренние шины процессора

Высокое быстродействие ИС 1867ВМ9Ф достигается за счет внутреннего разделения и параллельного выполнения необходимых действий. Раздельные шины позволяют выполнять инструкции параллельно, предоставлять доступ к данным и доступ к ПДП. Процессор имеет три группы шин:

- программные шины PADDR и PDATA;
- шины данных DADDR1, DADDR2 и DDATA;
- шины сопроцессора ПДП DMAADDR и DMADATA.

Эти шины охватывают всё физическое пространство процессора (сверхоперативное ОЗУ, регистры внутренних периферийных устройств). На рисунке 2.3 показаны эти внутренние шины и их соединение с блоками процессора.

Программный счетчик PC соединен с 32-битной программной адресной шиной PADDR. Регистр инструкций IR подключен к 32-битной программной шине данных PDATA. Такая шинная структура дает возможность выбирать слово инструкции за один машинный цикл.

32-битные адресные шины DADDR1, DADDR2 и 32-битная шина данных DDATA поддерживают два обращения к внутренней памяти за один машинный цикл. Шина DDATA передает данные к ЦПУ через шины CPU1 и CPU2. Шины CPU1 и CPU2 могут передавать два операнда данных памяти к умножителю, АЛУ и регистровому файлу за каждый машинный цикл. Рисунок 2.3 показывает шины, которые являются внутренними шинами ЦПУ.

К сопроцессору ПДП подходят 32-битная адресная шина DMAADDR и 32-битная шина данных DMADATA. Эти шины позволяют сопроцессору ПДП выполнять доступ к памяти параллельно с доступом к памяти ЦПУ с других шин.

Внешние шины процессора содержат два идентичных интерфейса для доступа к внешней памяти: глобальный интерфейс и локальный интерфейс. Каждый интерфейс содержит 32-битную шину данных, 31-битную адресную шину и управляющие сигналы (сигналы стробирования, готовности и выбора страниц). Подробное описание работы внешних шин представлено в разделе 9 «Операции внешней шины».

2.4 Прерывания

ИС 1867ВМ9Ф поддерживает четыре внешних прерывания IIOF0#-IIOF3# и немаскируемое внешнее прерывание NMI#. К внешним прерываниям можно также отнести и прерывание от сигнала RESET#, который устанавливает процессор и периферийные устройства в первоначальное состояние. Сопроцессор ПДП, таймеры и коммуникационные порты имеют свои собственные внутренние прерывания.

Когда ЦПУ отвечает на прерывание, то на выводе IACK# вырабатывается сигнал подтверждения того, что прерывание получено и обрабатывается. Подробнее работа процессора с внутренними и внешними прерываниями представлена в 7.4 «Прерывания».

2.5 Периферийные устройства

Все периферийные устройства процессора ИС 1867ВМ9Ф управляются через регистры, адресуемые как ячейки памяти, доступ к которым осуществляется через свою (периферийную) внутреннюю шину. Эта периферийная шина состоит из 32-битной шины данных и 32-битной адресной шины. Периферийная шина позволяет ЦПУ напрямую получить доступ к регистрам периферийных устройств. Периферийные устройства микросхемы 1867ВМ9Ф состоят из двух таймеров и шести коммуникационных портов. На рисунке 2.5 представлены периферийные устройства ИС.

2.5.1 Коммуникационные порты

Шесть высокоскоростных коммуникационных портов обеспечивают быстрый межпроцессорный обмен. В сочетании с двумя внешними интерфейсами доступа к внешней памяти (глобальной и локальной) они дают возможность создать мультипроцессорную систему для параллельной обработки ресурсоемких вычислительных задач, в которой достигается оптимальная системная производительность за счет распределения задач между несколькими процессорами. Каждый процессор может передавать результаты своей работы другому процессору через коммуникационный порт.

Более подробное описание работы коммуникационных портов представлено в разделе 12 «Коммуникационные порты».

Коммуникационные порты обеспечивают:

- скорость передачи данных до 640 Мбит/с (80 Мбайт или 20М слов в секунду);
- подключение между процессорами через 8-разрядную шину данных и четыре сигнала управления передачей;
- буферизацию всех передаваемых данных, как на входе, так и на выходе;
- автоматический арбитраж доступа к общей шине данных коммуникационного порта для гарантированной передачи/приёма данных;
- синхронизацию между ЦПУ, сопроцессором ПДП и шестью коммуникационными портами через внутренние прерывания и сигнал готовности.

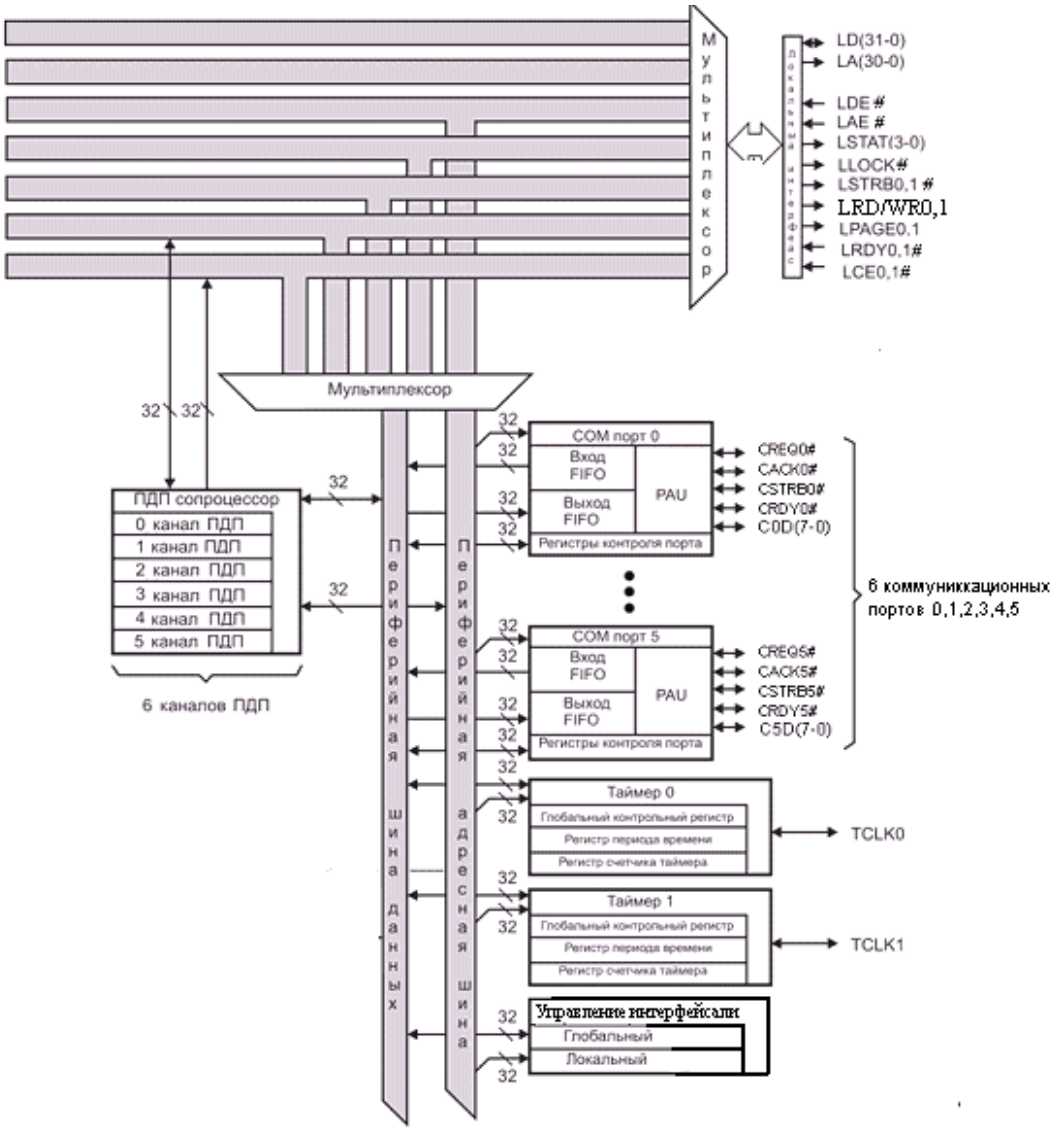


Рисунок 2.5 – Периферийные устройства ИС

2.5.2 Сопроцессор ПДП

Шесть каналов встроенного сопроцессора ПДП могут, независимо от работы основного ЦПУ, читать/записывать данные в любой адрес, который определен в карте памяти. Это позволяет согласовывать работу медленной внешней памяти и периферийных устройств без уменьшения производительности ЦПУ.

Сопроцессор ПДП содержит свои собственные адресные генераторы, регистры адреса источника и получателя данных, а также счетчик передач. Предназначенные для сопроцессора ПДП адресные шины и шины данных позволяют минимизировать конфликты между ЦПУ и сопроцессором ПДП. Сопроцессор ПДП позволяет передавать данные как блоками, так и по одному слову. Ключевая возможность сопроцессора ПДП – это способность к автоматической реинициализации после завершения передачи всех данных. Это даёт возможность снова запустить сопроцессор ПДП с ранее загруженными установками. Подробное рассмотрение работы сопроцессора ПДП представлено в разделе 11 «Сопроцессор прямого доступа к памяти».

2.5.3 Таймеры

Каждый таймер является 32-разрядным универсальным счетчиком времени или числа событий. Таймер имеет два режима тактирования или внешнее, или внутреннее и два режима тактирования (внешний и внутренний). Каждый таймер имеет внешний

вывод (вход/выход), который может быть использован как вход синхронизации таймера или как выходной сигнал, управляемый таймером. Вывод таймера может также быть использован как вход/выход общего назначения. Таймеры детально рассмотрены в разделе 13 «Таймеры».

3 Регистры ЦПУ

Основной регистровый файл ЦПУ включает 32 регистра, которые могут быть использованы как операнды умножителя и АЛУ. Регистровый файл включает вспомогательные регистры, регистры повышенной точности и индексные регистры. Эти регистры поддерживают адресацию, операции с плавающей запятой и целочисленные операции, управление стеком и состоянием процессора, блоковые повторения, ветвления и прерывания.

Дополнительный регистровый файл ЦПУ объединяет два регистра – указатель на адрес таблицы системных векторов IVTP и указатель на адрес таблицы векторов программных прерываний TVTP.

Данный раздел описывает каждый регистр ЦПУ.

3.1 Основной регистровый файл ЦПУ

ИС 1867ВМ9Ф содержит 32 регистра в мультипортовом регистровом файле, который тесно связан с ЦПУ. Счетчик инструкций (команд) PC не включен в эти 32 регистра. Состав регистрового файла представлен в таблице 3.1.

Таблица 3.1 – Основной регистровый файл ЦПУ

Символ регистра	Адрес регистра (шестнадцати-ричное значение)	Назначенное имя функции
R0	00	Регистр повышенной точности 1
R1	01	Регистр повышенной точности 1
R2	02	Регистр повышенной точности 2
R3	03	Регистр повышенной точности 3
R4	04	Регистр повышенной точности 4
R5	05	Регистр повышенной точности 5
R6	06	Регистр повышенной точности 6
R7	07	Регистр повышенной точности 7
R8	1C	Регистр повышенной точности 8
R9	1D	Регистр повышенной точности 9
R10	1E	Регистр повышенной точности 10
R11	1F	Регистр повышенной точности 11
AR0	08	Вспомогательный регистр 0
AR1	09	Вспомогательный регистр 1
AR2	0A	Вспомогательный регистр 2
AR3	0B	Вспомогательный регистр 3
AR4	0C	Вспомогательный регистр 4
AR5	0D	Вспомогательный регистр 5
AR6	0E	Вспомогательный регистр 6
AR7	0F	Вспомогательный регистр 7
DP	10	Указатель страницы данных
IR0	11	Индексный регистр 0
IR1	12	Индексный регистр 1
BK	13	Регистр размера блока повторения
SP	14	Указатель системного стека
ST	15	Регистр состояния
DIE	16	Регистр разрешения прерывания сопроцессора ПДП
IIE	17	Регистр разрешения внутренних прерываний ЦПУ
IIF	18	Регистр флагов ввода-вывода и прерывания IIOF (IIOF3#-IIOF0#, таймеров и ПДП)

Окончание таблицы 3.1

Символ регистра	Адрес регистра (шестнадцатиричное значение)	Назначенное имя функции
RS	19	Регистр адреса начала блока повторения
RE	1A	Регистр адреса конца блока повторения
RC	1B	Счетчик повторений

Все перечисленные в таблице 3.1 регистры могут быть использованы двояко – как операнды умножителя и АЛУ и как универсальные 32-битные регистры. Однако регистры также выполняют несколько специальных функций, например, 12 регистров повышенной точности поддерживают результаты повышенной точности с плавающей запятой. Восемь вспомогательных регистров поддерживают многообразие методов косвенной адресации и могут быть использованы как универсальные 32-битные регистры для хранения целочисленных и логических значений. Оставшиеся регистры поддерживают системные функции, такие как адресацию, управление стеком, состоянием процессора, прерываниями и блоковыми повторениями.

В разделе 6 «Методы адресации» дана более подробная информация и примеры использования регистров ЦПУ для адресации.

3.1.1 Регистры повышенной точности R0-R11

12 регистров повышенной точности R0-R11 могут хранить 32-разрядные целые числа и 40-разрядные числа с плавающей запятой и с ними можно оперировать.

Эти регистры состоят из двух различных областей:

- биты 39-32 хранят порядок (e) числа с плавающей запятой;
- биты 31-0 хранят мантиссу числа с плавающей запятой:
- бит 31 – это знаковый бит (s),
- биты 30-0 – это дробная часть (f).

Любая инструкция, которая предполагает, что операнды – это числа с плавающей запятой, использует разряды 39-0. Рисунок 3.1 иллюстрирует хранение в регистрах повышенной точности 40-разрядных чисел с плавающей запятой.

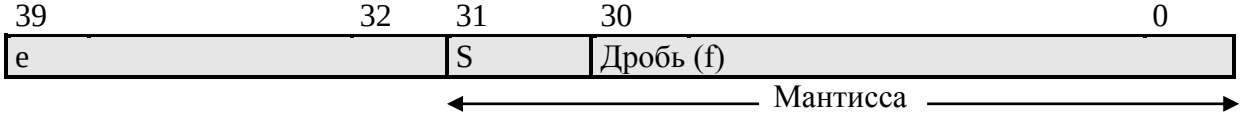


Рисунок 3.1 – Формат числа с плавающей запятой в регистре повышенной точности

Для работы с целыми числами разряды 31-0 регистров повышенной точности содержат целые значения (со знаком или без знака). Любая инструкция, которая содержит операнды с целыми значениями со знаком или без знака, использует только разряды 31-0. Разряды 39-32 остаются без изменений. Это верно для любых операций сдвига. Хранение 32-разрядных целых значений в регистрах повышенной точности показано на рисунке 3.2.

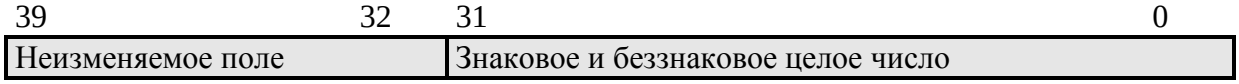


Рисунок 3.2 – Формат целого числа в регистре повышенной точности

3.1.2 Вспомогательные регистры AR0-AR7

Восемь 32-разрядных вспомогательных регистров AR0-AR7 могут быть доступны для ЦПУ и могут быть модифицированы арифметическими устройствами вспомогательных регистров ARAU. Основная функция вспомогательных регистров – это генерация 31-разрядных адресов операндов. Однако они также могут быть

использованы для выполнения различных функций, таких как циклический счетчик для косвенной адресации или 32-разрядные регистры общего назначения, которые могут быть изменены умножителем и АЛУ. В разделе 6 «Методы адресации» дана более подробная информация и примеры использования для адресации вспомогательных регистров.

3.1.3 Указатель страницы данных DP

Указатель страницы данных DP – это 32-разрядный регистр. 16 младших значащих разрядов указателя страницы данных используются в режиме прямой адресации как указатель на страницу адресуемых данных. Страницы данных имеют длину 64К слов. Число страниц равно 64К (65536) страниц. Биты 31-16 зарезервированы; они всегда читаются как нули и не могут быть изменены записью в этот регистр. Регистр DP может быть загружен с использованием инструкций LDP или LDI. На рисунке 6.1 показаны эти функции регистра DP.

3.1.4 Индексные регистры IR0, IR1

32-разрядные индексные регистры IR0, IR1 используются арифметическим устройством вспомогательных регистров ARAU для индексирования адресов. Регистр IR0 также используется в режиме бит-реверсной адресации. В разделе 6 «Методы адресации» дана подробная информация и примеры использования при адресации индексных регистров.

3.1.5 Регистр размера блока повторения BK

32-разрядный регистр размера блока повторения BK используется ARAU при циклической адресации для точного определения размера блока данных.

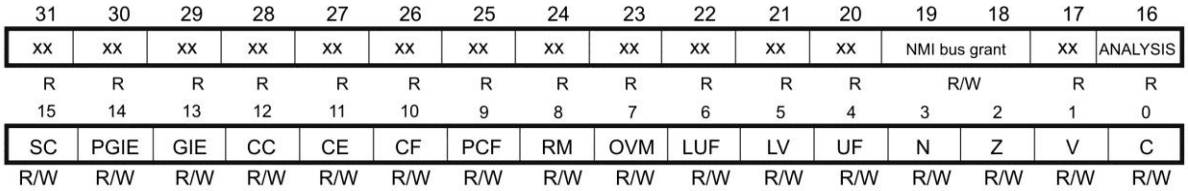
В подразделе 6.8 «Циклическая адресация» дана подробная информация об использовании регистра BK.

3.1.6 Указатель системного стека SP

Указатель системного стека SP – это 32-разрядный регистр, содержащий адрес вершины системного стека. SP всегда указывает на следующий элемент, выталкиваемый из стека. SP модифицируется при программных и системных прерываниях, вызовах подпрограмм и возврата из подпрограмм и инструкциями PUSH, PUSHF, POP, POPF. При выталкивании из стека и проталкивании данных в стек выполняется предекрементирование и постдекрементирование всех 32 разрядов регистра SP.

3.1.7 Регистр состояния ST

Регистр состояния ST содержит общую информацию о состоянии ЦПУ. Обычно выполняемые инструкции устанавливают соответствующие флаги в регистре состояния в зависимости, например, от нулевого или отрицательного результата и т.п. Регистр состояния также модифицируется на операциях загрузки и сохранения регистра, а также после арифметических или логических операций. Однако если регистр состояния ST загружается, то содержимое операнда источника изменяет текущее содержимое разряда «разряд в разряд», независимо от состояния любого из разрядов операнда источника. Таким образом, после записи в этот регистр он будет иметь значение, которое идентично содержимому операнда источника. Это позволяет достаточно просто сохранять и восстанавливать регистр состояния. При системном сбросе в этот регистр записывается «0», после сброса бит CF устанавливается в «1». Формат регистра состояния показан на рисунке 3.3. Далее дано описание каждого поля регистра состояния.



Примечание – Принятые условные обозначения: xx – резервный бит, R – чтение, R/W – чтение/запись.

Рисунок 3.3 – Регистр состояния ST

Описание битов регистра состояния ST:

- бит C – флаг переноса;
- бит V – флаг переполнения;
- бит Z – флаг нулевого значения;
- бит N – флаг отрицательного значения;
- бит UF – флаг потери значимости разрядов числа с плавающей запятой;
- бит LV – флаг, фиксирующий переполнение;
- бит LUF – флаг, фиксирующий потерю значимости разрядов числа с плавающей запятой;
- бит OVM – флаг режима переполнения. Этот флаг модифицируется программно и влияет только на результат целочисленных операций.

Если OVM=0, то режим переполнения отключен; целочисленный результат с переполнением не обрабатывается специальным методом.

Если OVM=1:

а) целочисленный положительный результат с переполнением устанавливается в значение наибольшего положительного 32-разрядного числа в дополнительном коде до двух (7FFF FFFFh);

б) целочисленный отрицательный результат устанавливается в наименьшее отрицательное 32-разрядное число в дополнительном коде до двух (8000 0000h);

- бит RM – флаг режима повторения. Если RM=1, то PC модифицируется или при повторении блока инструкций, или в режиме одиночного повторения;
- бит PCF – предыдущее состояние бита CF. Когда осуществляется программное или системное прерывание, то бит CF устанавливается в «1», а PCF принимает предыдущее значение бита CF;
- бит CF – блокировка изменения КЭШ. Если CF=1, то изменение КЭШ блокировано. Если КЭШ разрешен CE=1, то выбор из КЭШ разрешен, но состояние КЭШ не изменяется. Эта функция может быть использована для сохранения часто используемых кодов, находящихся в КЭШ. При сбросе в этот бит заносится «0». Очистка КЭШ CC=1 разрешается, если CF=0;
- бит CE – разрешение КЭШ. Установка бита CE=1 разрешает КЭШ, позволяя использовать КЭШ в соответствии с алгоритмом КЭШирования LRU (вытесняется наименее использованный). Установка бита CE=0 запрещает КЭШ; КЭШ не изменяется, не происходит выборка из КЭШ. При сбросе в этот бит заносится «0». Очистка КЭШ CC=1 разрешается, если CE=0. Описание битов CE и CF представлено в таблице 3.2;

Таблица 3.2 – Описание битов CE и CF

Бит CE	Бит CF	Действие
0	0	КЭШ не разрешен
0	1	КЭШ не разрешен
1	0	КЭШ разрешен и не блокирован
1	1	КЭШ разрешен, но заблокирован (КЭШ только читается)

- бит CC – очистка КЭШ. Установка бита CC = 1 делает недействительным все содержимое КЭШ. Этот бит всегда очищается после того, как он записан и всегда читается как «0». При сбросе в этот бит записывается «0»;

- бит GIE – разрешение глобальных прерываний. Разрешает или запрещает все маскируемые прерывания. Если GIE=1, то ЦПУ отвечает на все разрешенные прерывания. Если GIE=0, то ЦПУ не отвечает на все разрешенные прерывания. Этот бит не влияет на прерывание NMI#. Инструкции IDLE, LAT, RETI, RETID и TRAP изменяют значение бита GIE. Когда возникает программное или системное прерывание, то GIE устанавливается в «0»;

- бит PGIE – предыдущее состояние бита GIE. GIE устанавливается в «0», когда возникает программное или системное прерывание. Когда это происходит, то PGIE принимает предыдущее значение бита GIE. Заметим, что инструкции RETIcond и RETIcondD копируют PGIE в GIE бит. При сбросе в этот бит заносится «0»;

- бит COND (SC) – этот бит определяет установку флагов условий (биты 0-6 регистра ST). Если SET COND = 0, то флаги условий устанавливаются, если операндом-приемником является какой-либо из регистров повышенной точности R0-R11. При сбросе в этот бит заносится «0». Если SET COND = 1, то флаги условий устанавливаются, если конечным операндом-приемником является любой регистр в основном регистровом файле, исключая регистр состояния. Флаги условий всегда устанавливаются при выполнении инструкций CMPF, CMPI, CMPF3, CMPI3, TSTB или TSTB3 независимо от значения SET COND;

- бит ANALYSIS – бит только читается, он используется в режиме анализа, чтобы обеспечить эмулятор информацией о состоянии микросхемы;

- предоставление шины по NMI# – эта функция полезна для корректировки ошибок работы коммуникационного порта с использованием программного сброса. Если бит 19 = 1, а бит 18 = 0, то внутренний сигнал разрешения передачи по периферийной шине устанавливается по спаду NMI#. Если сигнал NMI# появляется, когда периферийная шина находится в состоянии останова, то NMI# прерывает незаконченный цикл и происходит переход к подпрограмме обслуживания NMI#. Состояние останова может возникнуть при записи в уже заполненный буфер FIFO или при чтении из пустого буфера FIFO;

- xx – зарезервировано. Значение не определено. Эти биты только для чтения.

3.1.8 Регистр разрешения прерывания DIE сопроцессора ПДП

32-битный регистр разрешения прерывания сопроцессора ПДП DIE разделен на 6 частей, которые определяют, какие прерывания могут быть использованы при управлении синхронизацией для каждого из шести каналов ПДП. Управление синхронизацией происходит при чтении или записи в канал ПДП. При сбросе во все разряды регистра записываются нули.

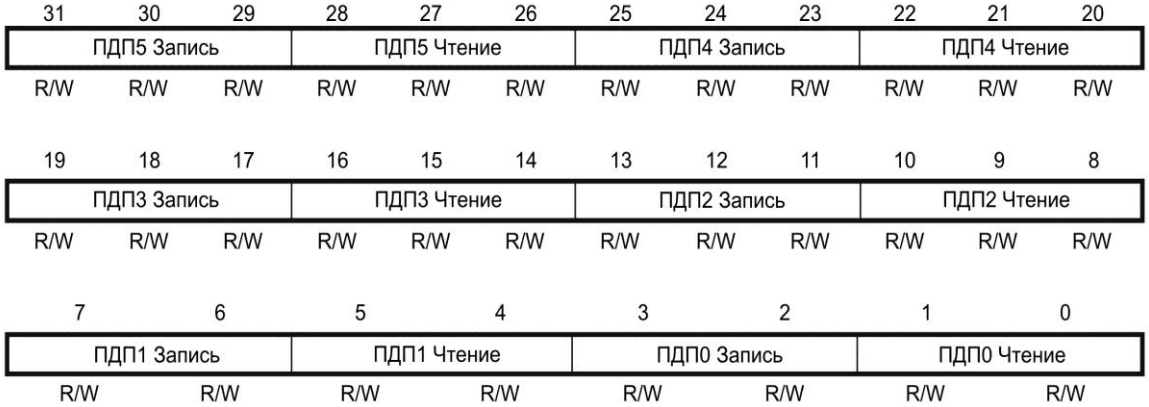
Каждый канал ПДП отслеживает не только выбранные прерывания синхронизации ПДП, но также в режиме синхронизации – какой канал в настоящее время используется, смотри таблицу 11.3. Режим синхронизации определяется полем SYNC MODE в регистрах управления каналами ПДП, которые находятся в сопроцессоре ПДП.

Используя синхронизирующие прерывания, каждый канал ПДП может, например, обслуживать соответствующий коммуникационный порт. Заметим, что канал ПДПi с номером i может быть синхронизирован только сигналами, приходящими от коммуникационного порта с номером i, где $0 \leq i \leq 5$. Также, каждый ПДП канал

может быть синхронизирован внешними прерываниями и прерываниями от встроенных таймеров.

3.1.8.1 Объединенный режим

Рисунок 3.4 показывает регистр разрешения прерывания сопроцессора ПДП DIE для объединенного режима. В таблице 3.3 представлены все возможные комбинации ПДП0 и ПДП1 для объединенного режима. В таблице 3.4 представлены все трехбитные комбинации ПДП2-ПДП5 для объединенного режима.



Примечание – R – чтение, W – запись.

Рисунок 3.4 – Регистр разрешения прерывания DIE сопроцессора ПДП для объединенного режима ПДП

Таблица 3.3 – Синхронизационные прерывания объединенного режима каналов ПДП0 и ПДП1

Значение бита (в ПДП0 и ПДП1)	Разрешение прерывания ПДП0 и ПДП1				Источник прерывания синхронизации ПДП
	ПДП0 чтение	ПДП0 запись	ПДП1 чтение	ПДП1 запись	
00†	нет	нет	нет	нет	--
01	ICRDY0	ICRDY0	ICRDY1	ICRDY1	От коммуникационного порта
10	IIOF0#	IIOF1#	IIOF2#	IIOF3#	От внешних сигналов IIOF3#-IIOF0#
11	TIM0	TIM0	TIM0	TIM0	От таймера TIM0
Примечание – † – Канал ПДП в режиме ожидания (чтение или запись не производится), если используется синхронная передача ПДП.					

Таблица 3.4 – Синхронизирующие прерывания объединенного режима каналов ПДП2-ПДП5

Значение бита (в ПДП2-ПДП5)	Разрешение прерывания в ПДП2-ПДП5†		Источник прерывания синхронизации ПДП
	ПДПх чтение	ПДПх запись	
000 ‡	нет	нет	-
001	ICRDYx†	OCRDYx†	От коммуникационного порта
010	IIOF0#	IIOF0#	От внешних сигналов IIOF3#-IIOF0#
011	IIOF1#	IIOF1#	-
100	IIOF2#	IIOF2#	-
101	IIOF3#	IIOF3#	-
110	TIM0	TIM0	-
111	TIM1	TIM1	-

Примечания

1 † – x в ПДПх – это номер канала ПДП, который также является номером для ICRDYx и OCRDYx прерываний.

2 ‡ – Канал ПДП в режиме ожидания (чтение или запись не производится), если используется синхронная передача ПДП.

3 Прерывания, перечисленные в таблицах 3.3, 3.4 (ICRDYx, OCRDYx, TIM0 и т. д.), не векторные. Сопроцессор ПДП использует их как сигналы для синхронизации передачи, подробное описание в 11.10 «Прямой доступ к памяти и прерывания».

3.1.8.2 Раздельный режим

На рисунке 5 показан регистр разрешения прерываний ПДП для раздельного режима. В таблице 3.5 представлены все возможные комбинации ПДП0 и ПДП1 для раздельного режима. В таблице 3.6 приведены все трехбитные комбинации ПДП2-ПДП5 для раздельного режима.

Примечание – R – чтение, W – запись.

Рисунок 3.5 – Регистр разрешения прерывания DIE сопроцессора ПДП для раздельного режима ПДП

Таблица 3.5 – Синхронизирующие прерывания раздельного режима каналов ПДП0 и ПДП1

Значение бита (в ПДП0 или ПДП1)	Активизированные прерывания				Источник прерывания синхронизации ПДП
	ПДП0		ПДП1		
	Вспомога-тельный Чтение	Основной Запись	Вспомога-тельный Чтение	Основной Запись	
00†	Нет	Нет	Нет	Нет	--
00	ICRDY0	OCRDY0	ICRDY1	OCRDY1	От коммуникацион-ного порта
10	IIOF0#	IIOF1#	IIOF2#	IIOF3#	От внешних сигналов IIOF3#-IIOF0#
10	TIM0	TIM0	TIM0	TIM0	От таймера TIM0
Примечание – † – Канал ПДП в режиме ожидания (чтение или запись не производится), если используется синхронная передача ПДП.					

Таблица 3.6 – Синхронизирующие прерывания раздельного режима каналов ПДП2-ПДП5

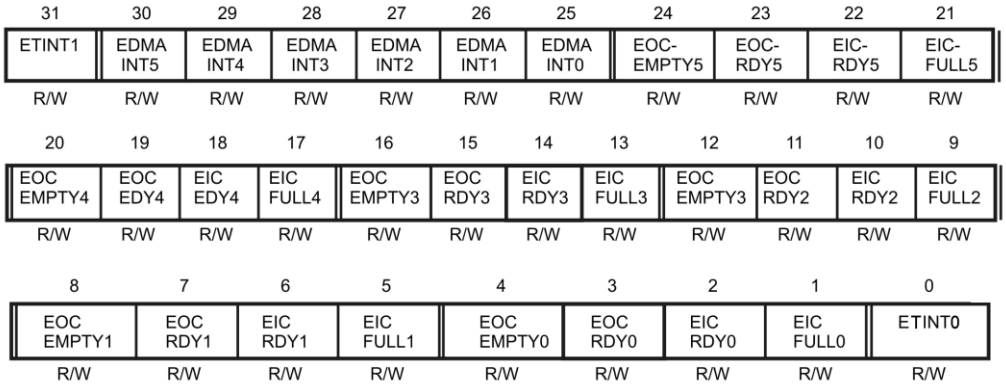
Значение бита (в ПДП2-ПДП5)	Активизированные прерывания в ПДП2-ПДП5†		Ресурс прерывания для ПДП синхронизации
	ПДПx		
	Вспомогательный Чтение†	Основной Запись†	
000‡	Нет	Нет	-
001	ICRDYx†	OCRDYx†	От коммуникационного порта
010	IIOF0#	IIOF0#	От внешних сигналов IIOF3#-IIOF0#
011	IIOF1#	IIOF1#	-
100	IIOF2#	IIOF2#	-
101	IIOF3#	IIOF3#	-
110	TIM0	TIM0	От таймеров TIM0 и TIM1
111	TIM1	TIM1	-
Примечания 1 † – x в ПДПx – это номер канала ПДП, который также является номером для IRCDYx и OCRDYx прерываний. 2 ‡ – Канал ПДП находится в режиме ожидания (чтение или запись не производится), если используется синхронная передача ПДП.			

3.1.9 Регистр разрешения внутренних прерываний ЦПУ IIE

32-разрядный регистр разрешения внутренних прерываний ЦПУ изображен на рисунке 3.6, он разрешает/запрещает следующие прерывания ЦПУ:

- от таймера 0 и таймера 1;
- от коммуникационных портов (0-5) (входной буфер полный, входной буфер готов, выходной буфер готов, выходной буфер пустой);
- от каналов (0-5) сопроцессора ПДП.

На рисунке 3.6 показаны биты регистра IIE. «1» – прерывание разрешено, «0» – прерывание запрещено. При сбросе во все биты регистра записываются нули.



Примечания

1 Поля, отделенные двойными линиями, относятся к разным блокам.

2 Принятое условное обозначение: R/W – чтение/запись.

Рисунок 3.6 – Внутреннее прерывание, активизирующее биты регистра IIE

Описание битов регистра IIE:

- EICFULLx – прерывание, если входной буфер коммуникационного порта x полный;
- EICRDYx – прерывание, если входной буфер коммуникационного порта x готов;
- EOCRDYx – прерывание, если выходной буфер коммуникационного порта x готов;
- EOCEMPTYx прерывание, если выходной буфер коммуникационного порта x пустой;
- EDMAINTx – прерывание канала x сопроцессора ПДП;
- ETINT0 – прерывание таймера 0;
- ETINT1 – прерывание таймера 1.

В каждом случае x обозначает номер коммуникационного порта (0-5) или номер канала ПДП (0-5). Например, «1» в 5 разряде означает то, что разрешено прерывание по заполнению входного буфера первого коммуникационного порта. Установка в «1» соответствующего разряда разрешает прерывание, «0» – запрещает прерывание.

3.1.10 Регистр IIF флагов ввода-вывода и прерывания IIOF

Регистр IIF управляет сигналами внешних выводов IIOF3#-IIOF0#. Этот регистр используется для определения:

- какие из выводов IIOF3#-IIOF0# используются как сигналы ввода-вывода общего назначения, а какие используются для внешних прерываний;
- является ли этот вывод входным (только читается) или выходным (чтение/запись), если этот вывод определен как вывод общего назначения;
- по фронту или по уровню осуществляется прерывание, если вывод определен как вход для внешнего прерывания;
- разрешено ли прерывание от внешнего сигнала или нет.

Регистр IIF также включает в себя флаги прерывания от таймера, ПДП и NMI. На рисунке 3.7 представлены биты регистра IIF. Ниже приведено описание каждого бита.

Биты регистра IIF могут быть прочитаны или записаны программно. Программное управление обеспечивает доступ к сигналам IIOFx#, которые могут

рассматриваться либо как входные/выходные сигналы общего назначения, либо как сигналы прерывания. Например, если в регистре IIF бит FUNCx=0 (вход/выход) и TYPEx=1 (выход), то запись в бит FLAGx переводит внешний сигнал IIOFx# в соответствующее состояние. Если FUNCx=1 (прерывание), то запись «1» в бит FLAGx регистра IIF будет равнозначна появлению соответствующего сигнала прерывания. Следовательно, все прерывания могут быть вызваны как программно, так и аппаратно. Так как биты прерываний могут быть прочитаны, то они могут быть опрошены программно, когда не требуется управление прерываниями.

Внутренние прерывания происходят аналогично. В регистре IIF бит, отвечающий за внутренние прерывания (например, TINT0, TINT1), может быть прочитан и записан программно. Запись «1» вызывает прерывание, запись «0» – сбрасывает его. Все внутренние прерывания выполняются за один цикл H1/H3. Для изменения IIF используются логические операции (AND, OR и т. д.) как показано:

Правильно	Не правильно
LDI @MASK, R0	LDI IIF, R1
AND R0, IIF	AND @MASK, R1
	LDI R1, IIF

Примечание – Программные и аппаратные прерывания кратко описаны в подразделе 3.2 и более детально в подразделах 7.4 «Прерывания» и 7.5 «Программные прерывания».

31	30	29	28	27	26	25	
TINT1	DMAINT5	DMAINT4	DMAINT3	DMAINT2	DMAINT1	DMAINT0	TINT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
23	22	21	20	19	18	17	16
XX	XX	XX	XX	XX	XX	XX	NMI
R	R	R	R	R	R	R	R
15	14	13	12	11	10	9	8
EIIOF3	FLAG3	TYPE3	FUNC3	EIIOF2	FLAG2	TYPE2	FUNC2
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
7	6	5	4	3	2	1	0
EIIOF1	FLAG1	TYPE1	FUNC1	EIIOF0	FLAG0	TYPE0	FUNC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Примечание – Принятые условные обозначения: R – чтение, R/W – чтение/запись.

Рисунок 3.7 – Регистр флагов прерываний IIF

Функциональное назначение битов

FUNCx – режим работы вывода IIOFx#. Если FUNCx=0, то сигнал IIOFx# – это ввод/вывод (R/W) общего назначения. Если FUNCx = 1, то IIOFx# – сигнал прерывания.

TYPEx – тип функции для вывода IIOFx#. Если вывод IIOFx# – это ввод/вывод общего назначения FUNCx = 0, то TYPEx = 0 устанавливает IIOFx# входом, TYPEx=1 устанавливает IIOFx# выходом. Если сигнал IIOFx# определен как сигнал прерывания FUNCx =1, то при TYPEx = 0 по выводу IIOFx# фиксируется прерывание по фронту, при TYPEx = 1 по выводу IIOFx# фиксирует прерывание по уровню.

FLAGx – флаг вывода IIOFx#. Если сигнал IIOFx# – сигнал ввода общего назначения FUNCx = 0, TYPEx = 0, то FLAGx = значению на выводе IIOFx# (только читается). Если сигнал IIOFx# – сигнал выхода общего назначения FUNCx = 0, TYPEx = 1, то FLAGx = значению на выводе IIOFx# (чтение/запись). Если сигнал IIOFx# – определен как сигнал прерывания FUNCx = 1, то FLAGx=0, если прерыва-

ние не пришло и FLAGx = 1, если прерывание пришло. Если в FLAGx записан «0», то соответствующее прерывание сбрасывается.

EIIOFx – запрет/разрешение внешнего прерывания**.** Если EIIOFx = 0 запрещаются внешние прерывания по сигналу IIOFx#. Если EIIOFx=1, то разрешаются внешние прерывания по сигналу IIOFx#.

NMI **–** флаг немаскированного прерывания NMI. NMI прерывание (по внешнему сигналу NMI#) ведет себя подобно другим прерываниям, за исключением того, что его нельзя замаскировать (запретить) посредством бита GIE (ST бит 13) или посредством записи бита NMI. Он временно маскируется во время задержанных переходов и многоцикловых операций ЦПУ. При сбросе этот бит равен «0».

Установленное прерывание очищается только посредством обслуживания прерывания. NMI прерывание фиксируется по переднему и заднему фронту. Данный бит предназначен только для чтения. Если при чтении NMI = 0, то прерывание не пришло. Если при чтении NMI = 1, прерывание поступило.

Reserved – зарезервировано; читается как нуль.

TINT0, TINT1 – флаги прерывания таймеров 0 и 1. Если при чтении TINTx = 0, то прерывание от таймера «x» не пришло. Если при чтении TINTx = 1, то прерывание от таймера «x» пришло. Запись нуля в этот бит сбрасывает прерывание.

DMAINTx – флаг прерывания от каналов (0-5) сопроцессора ПДП. Если при чтении DMAINTx = 0, то прерывание от канала «x» не пришло. Если при чтении DMAINTx = 1, то прерывание от канала «x» поступило. Запись нуля в этот бит сбрасывает прерывание.

Примечания

1 Затененные IIF разряды 0, 1, 2, 3 используются IIOF0#; затененные IIF разряды 4, 5, 6, 7 используются IIOF1# и т. д.

2 «x» означает соответствующий IIOFx# сигнал прерывания IIOF3#-IIOF0#.

3.1.11 Регистры адреса начала RS и конца RE блока повторения и счетчика повторений RC

Регистр адреса начала RS блока повторения – это 32-разрядный регистр, содержащий начальный адрес повторяющегося блока памяти программы, при работе в режиме повторений.

Регистр адреса конца RE блока повторения – это 32-разрядный регистр, содержащий конечный адрес повторяющегося блока памяти программы, при работе в режиме повторений.

Примечание – Если RE<RS, блок программной памяти не повторяется, и код не выполняет возврата к началу блока. Однако ST (RM) бит остается установленным в «1».

Счетчик повторений RC – это 32-разрядный регистр, используемый для определения числа повторного выполнения блока кода. Если в RC записано число n, то цикл (число повторений) осуществляется (n+1) раз.

3.1.12 Счетчик инструкций (команд) PC

Счетчик инструкций PC – это 32-разрядный регистр, содержащий адрес следующей исполняемой инструкции. Счетчик инструкций не является частью регистрового файла, он является регистром, который может изменяться командами в процессе выполнения программы.

3.1.13 Скрытые биты и совместимость

Чтобы обеспечить совместимость с будущими ИС семейства микропроцессоров 1867, резервные разряды, читающиеся как «0», должны быть записаны как «0». Резервные биты, имеющие неопределенные значения, не должны изменять текущее значение. В остальных случаях пользователь должен поддерживать резервные биты точно в определенном состоянии.

3.2 Дополнительный регистровый файл ЦПУ

Дополнительный регистровый файл включает в себя два специальных управляющих регистра:

- регистр-указатель на адрес таблицы векторов системных прерываний IVTP;
- регистр-указатель на адрес таблицы векторов программных прерывания TVTP.

Таблица 3.7 – Дополнительные регистры ЦПУ

Ассемблерный синтаксис	Адрес регистра	Функциональное имя
IVTP	00h	Указатель на адрес таблицы системных векторов прерывания. Точка начала таблицы системных векторов прерывания.
TVTP	01h	Указатель на адрес таблицы программных векторов прерывания. Точка начала таблицы программных векторов прерывания.

Примечание – Необходимо использовать инструкцию LDEP для загрузки (копирования) дополнительного регистра в основной регистр (например, в какой-нибудь вспомогательный регистр AR0-AR7, см. таблицу 3.1).

Например:

LDEP IVTP, AR5; AR5 содержит IVTP

Подобным образом используется инструкция LDPE для загрузки (копирования) основного регистра в дополнительный регистр. Эти инструкции не изменяют флаги условий регистра состояния.

LDPE AR5, IVTP; IVTP содержит AR5

Примечание – Таблица векторов системных прерываний и таблица векторов программных прерываний не должны превышать 512 слов, таким образом, девять наименьших значащих разрядов этих указателей являются нулями (т. е. $10\ 0000\ 0000_2$=512=200h). В эти биты необходимо записывать только нули.

32-разрядный регистр IVTP указывает на таблицу векторов системных прерываний (IVT) в памяти.

32-разрядный регистр TVTP указывает на таблицу векторов программных прерываний (TVT) в памяти. Эта таблица содержит векторы для 512 программных прерываний инструкции TRAP (TRAP0-TRAP511).

Таблицы векторов системных и программных прерываний могут разделять одно и то же пространство в памяти. При такой конфигурации пользователь может расположить векторы программных прерываний там, где нет векторов системных прерываний. Например, если вектор системного прерывания 02Ch не используется, то можно поместить вектор программного прерывания в IVTP+02Ch (который также является TVTP+02Ch, если таблицы перекрываются), и после вызвать это программное прерывание посредством определения адреса 02Ch в инструкции TRAP.

При сбросе IVTP и TVTP устанавливаются в «0».

4 Память и КЭШ команд

Общее адресное пространство ИС составляет 4 Гбайт 32-разрядных слов. Программа, данные и область ввода-вывода содержатся внутри этого 4 Гбайт адресного пространства, обеспечивая, таким образом, хранение таблиц, коэффициентов, программы или данных либо в ОЗУ, либо в ПЗУ.

ИС 1867ВМ9Ф имеет два блока СОЗУ 0 и СОЗУ 1, каждый блок содержит по 1К × 32 бит. Блок ПЗУ содержит 4К × 32 бит. Каждый из блоков СОЗУ и ПЗУ поддерживает два обращения к ним со стороны ЦПУ в одном цикле. Наличие раздельных шин команд, шин данных и шин ПДП обеспечивает параллельно: выборку команд, чтение, запись данных и работу ПДП. Например, ЦПУ может обращаться к двум значениям данных в одном блоке RAM и выполняет выборку внешней программы параллельно с загрузкой ПДП другого блока ОЗУ внутри одного цикла.

КЭШ команд – это 128 × 32-разрядное ОЗУ, которое обеспечивает хранение часто повторяющихся фрагментов кода, что значительно уменьшает число внекристальных обращений и тем самым повышает общую производительность системы, так как обращение к КЭШ осуществляется быстрее, чем к внекристальной памяти.

Кроме того, внешние шины в этом случае свободны для использования ПДП, выборки внешней памяти или для использования другими устройствами в системе. В данном разделе приведена более подробная информация о памяти и командах КЭШ.

4.1 Карта памяти

Карта памяти ИС 1867ВМ9Ф приведена на рисунке 4.1. В зависимости от значения внешнего вывода ROMEN процессор имеет разные конфигурации карты памяти. Далее приведены различия в карте памяти:

- если ROMEN равен единице, то область адресного пространства с 0000 0000h по 000F FFFFh отведена для работы внутреннего ПЗУ, и процессор работает в режиме микрокомпьютера. Это показано с правой стороны рисунка;
- если ROMEN равен нулю, то область адресного пространства с 0000 0000h по 000F FFFFh доступна в адресном пространстве локальной шины, и процессор работает в режиме микропроцессора. Это показано с левой стороны рисунка.

Остальная часть карты памяти процессора остаётся неизменной и не зависит от значения сигнала ROMEN.

Память, начинающаяся от 0010 0000h, не зависит от ROMEN.

Описание каждого выделенного блока в адресном пространстве:

- 0000 0000h-000F FFFFh эта область адресов может быть либо областью локальной шины, либо областью локальной шины встроенного ПЗУ (зависит от значения ROMEN);
- 0010 0000h-0010 00FFh – это адреса регистров внутрикристальных периферийных устройств (сопроцессор ПДП, коммуникационные порты, таймер и т. д.);
- 0010 0100h-002F F7FFh – адреса зарезервированы;
- 002F F800h-002F FBFFh – это 1К СОЗУ, блок 0;
- 002F FC00h-002F FFFFh – это 1К СОЗУ, блок 1;
- 0030 0000h-7FFC 7FFFh – это адреса локальной шины внешнего интерфейса;
- 7FFC 8000h-7FFF FFFFh – это адреса глобальной шины внешнего интерфейса.

БУДЬТЕ ВНИМАТЕЛЬНЫ! Обращение к зарезервированной области памяти может привести к непредсказуемым результатам.

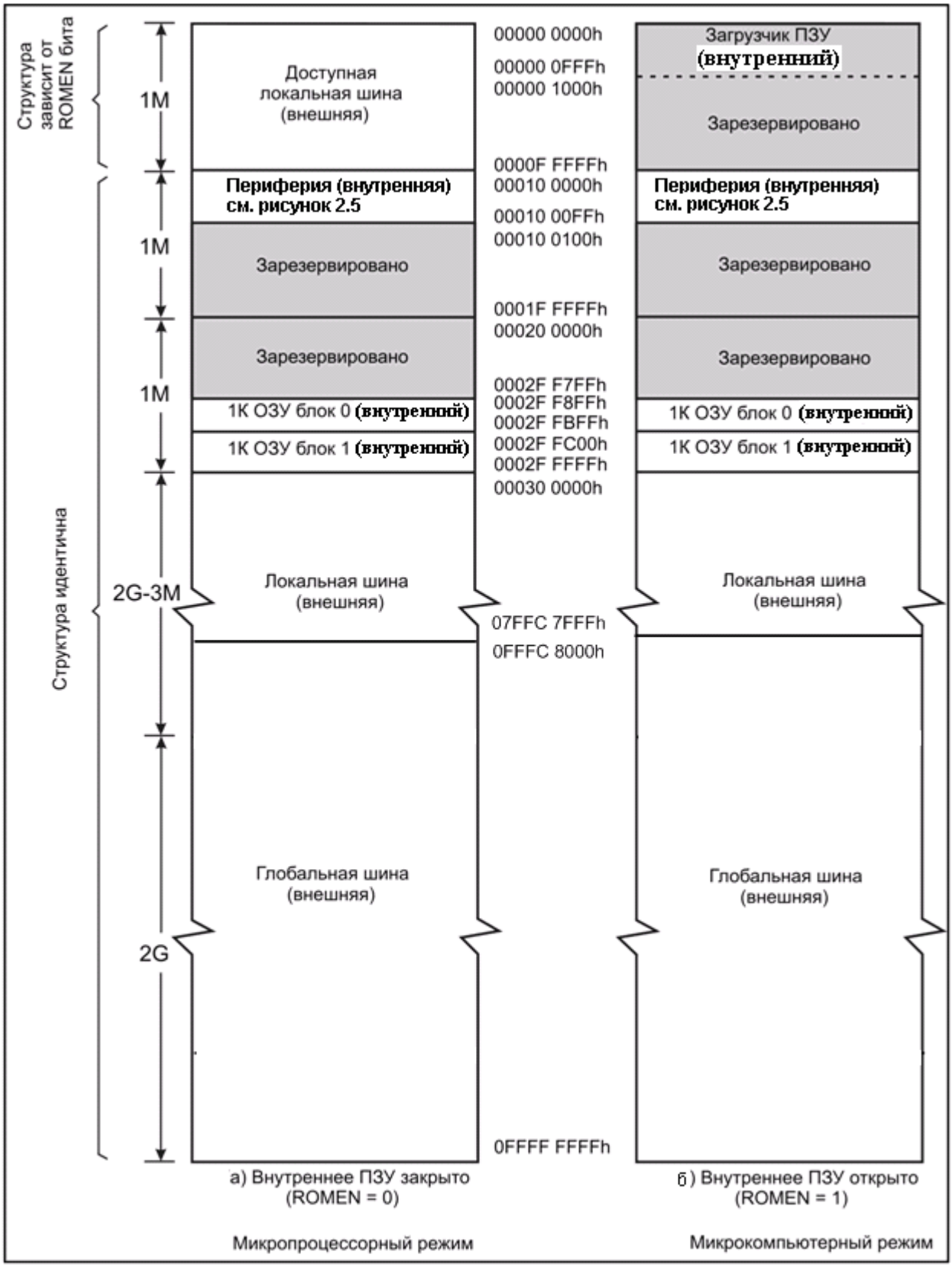


Рисунок 4.1 – Карта памяти ИС 1867ВМ9Ф

4.2 Карта памяти периферийной шины

В таблице 4.1 представлена периферийная карта памяти ИС.

Таблица 4.1 – Карта памяти периферийных устройств

Адрес	Периферийное устройство
0010 0000h 0010 000Fh	Регистры управления локальной и глобальной шинами (16 слов)
0010 0008h	Регистр управления PLL
0010 0010h 0010 001Fh	Регистры управления работой в режиме эмуляции (16 слов)
0010 0020h 0010 002Fh	Регистры таймера 0 (16 слов)
0010 0030h 0010 003Fh	Регистры таймера 1 (16 слов)
0010 0040h 0010 004Fh	Регистры коммуникационного порта 0 (16 слов)
0010 0050h 0010 005Fh	Регистры коммуникационного порта 1 (16 слов)
0010 0060h 0010 006Fh	Регистры коммуникационного порта 2 (16 слов)
0010 0070h 0010 007Fh	Регистры коммуникационного порта 3 (16 слов)
0010 0080h 0010 008Fh	Регистры коммуникационного порта 4 (16 слов)
0010 0090h 0010 009Fh	Регистры коммуникационного порта 5 (16 слов)
0010 00A0h 0010 00AFh	Регистры 0 канала сопроцессора ПДП (16 слов)
0010 00B0h 0010 00BFh	Регистры 1 канала сопроцессора ПДП (16 слов)
0010 00C0h 0010 00CFh	Регистры 2 канала сопроцессора ПДП (16 слов)
0010 00D0h 0010 00DFh	Регистры 3 канала сопроцессора ПДП (16 слов)
0010 00E0h 0010 00EFh	Регистры 4 канала сопроцессора ПДП (16 слов)
0010 00F0h 0010 00FFh	Регистры 5 канала сопроцессора ПДП (16 слов)

4.2.1 Регистры управления глобальной и локальной шинами

Эти регистры управляют работой глобальной и локальной шин. В карте памяти данные регистры занимают блок из 16 слов. На рисунке 4.2 показаны адреса этих регистров, а более подробное описание регистров представлено в разделе 9 «Операции внешней шины».

Регистры управления работой глобальной и локальной шин определяют:

- размеры страниц подключаемой внешней памяти;
- размеры адресных блоков, стробируемых разными стробами;
- состояние ожидания на шине;
- другие операции, которые составляют интерфейс памяти.

0010 0000h	Регистр управления интерфейсом глобальной памяти
0010 0001h-0010 0003h	Зарезервировано
0010 0004h	Регистр управления интерфейсом локальной памяти
0010 0005h 0010 000Fh	Зарезервировано

Рисунок 4.2 – Регистры управления интерфейсом памяти

4.2.2 Регистры управления работой ИС

Следующий 16-словный блок в карте памяти периферийной шины, как показано в таблице 4.1, содержит часть регистров управления работой процессора. Эти регистры зарезервированы для функций эмуляции.

4.2.3 Регистры таймера

Эта группа регистров занимает адресное пространство с 0010 0020h по 0010 003Fh в карте памяти периферийной шины и представлена на рисунке 4.3. Таймеры и их регистры рассмотрены в деталях в разделе 13 «Таймеры».

Таймер 0	0010 0020h	Регистр управления таймера 0
	0010 0021h 0010 0023h	Зарезервировано
	0010 0024h	Регистр счетчика таймера 0
	0010 0025h 0010 0027h	Зарезервировано
	0010 0028h	Регистр периода таймера 0
	0010 0029h	Зарезервировано
Таймер 1	0010 0030h	Регистр управления таймера 1
	0010 0031h 0010 0033h	Зарезервировано
	0010 0034h	Регистр счетчика таймера 1
	0010 0035h 0010 0037h	Зарезервировано
	0010 0038h	Регистр периода таймера 1
	0010 003Fh	Зарезервировано

Рисунок 4.3 – Регистры таймеров 0, 1

4.2.4 Карта памяти коммуникационного порта CPCR

Рисунок 4.4 иллюстрирует расположение адресов регистров управления коммуникационными портами CPCR, входных и выходных буферов FIFO, а также регистр сброса коммуникационных портов. Эти регистры более детально описаны в разделе 12 «Коммуникационные порты».

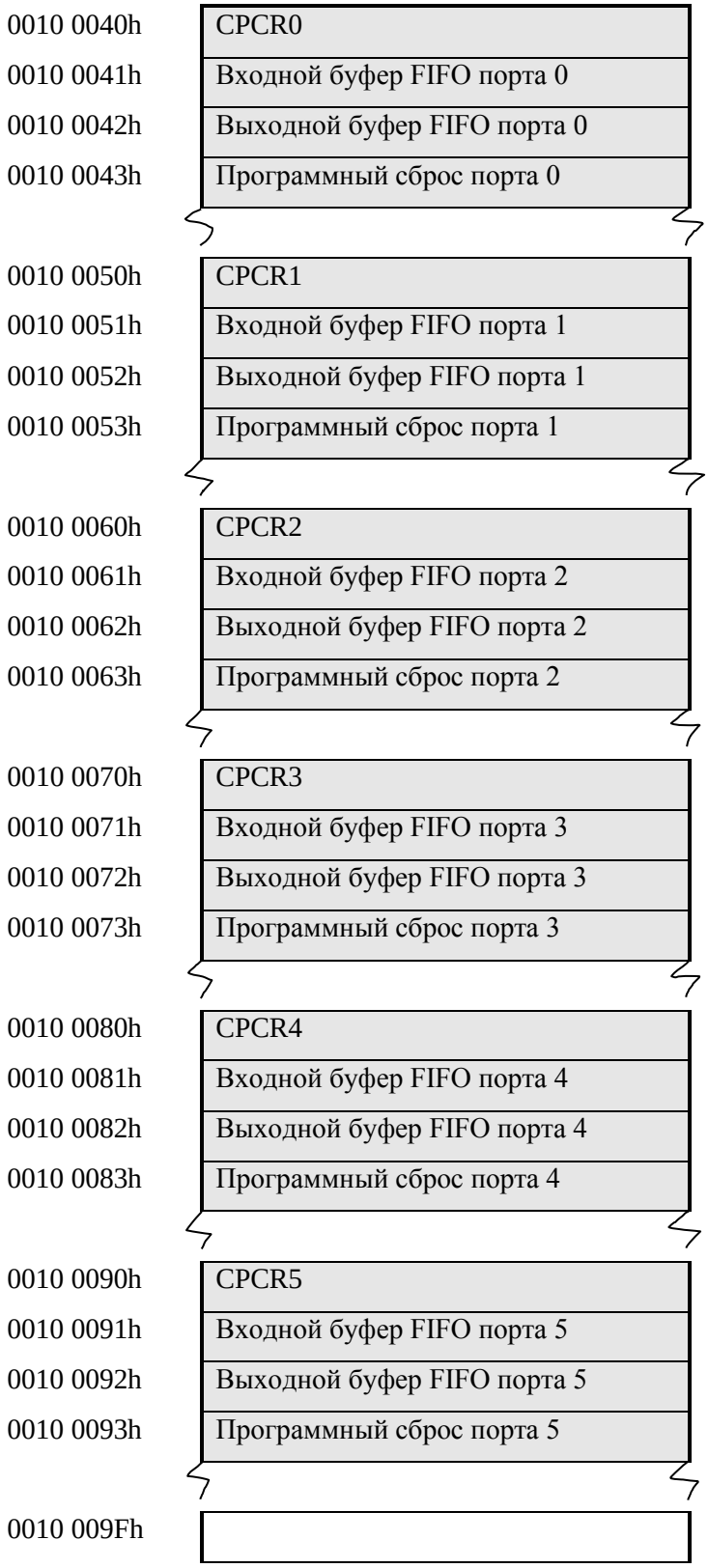


Рисунок 4.4 – Карта памяти коммуникационных портов

4.2.5 Регистры сопроцессора ПДП

Регистры сопроцессора ПДП, показанные на рисунке 4.5 – это нижний блок регистров в карте памяти периферийной шины. Эти регистры описаны в разделе 11 «Сопроцессор прямого доступа к памяти».

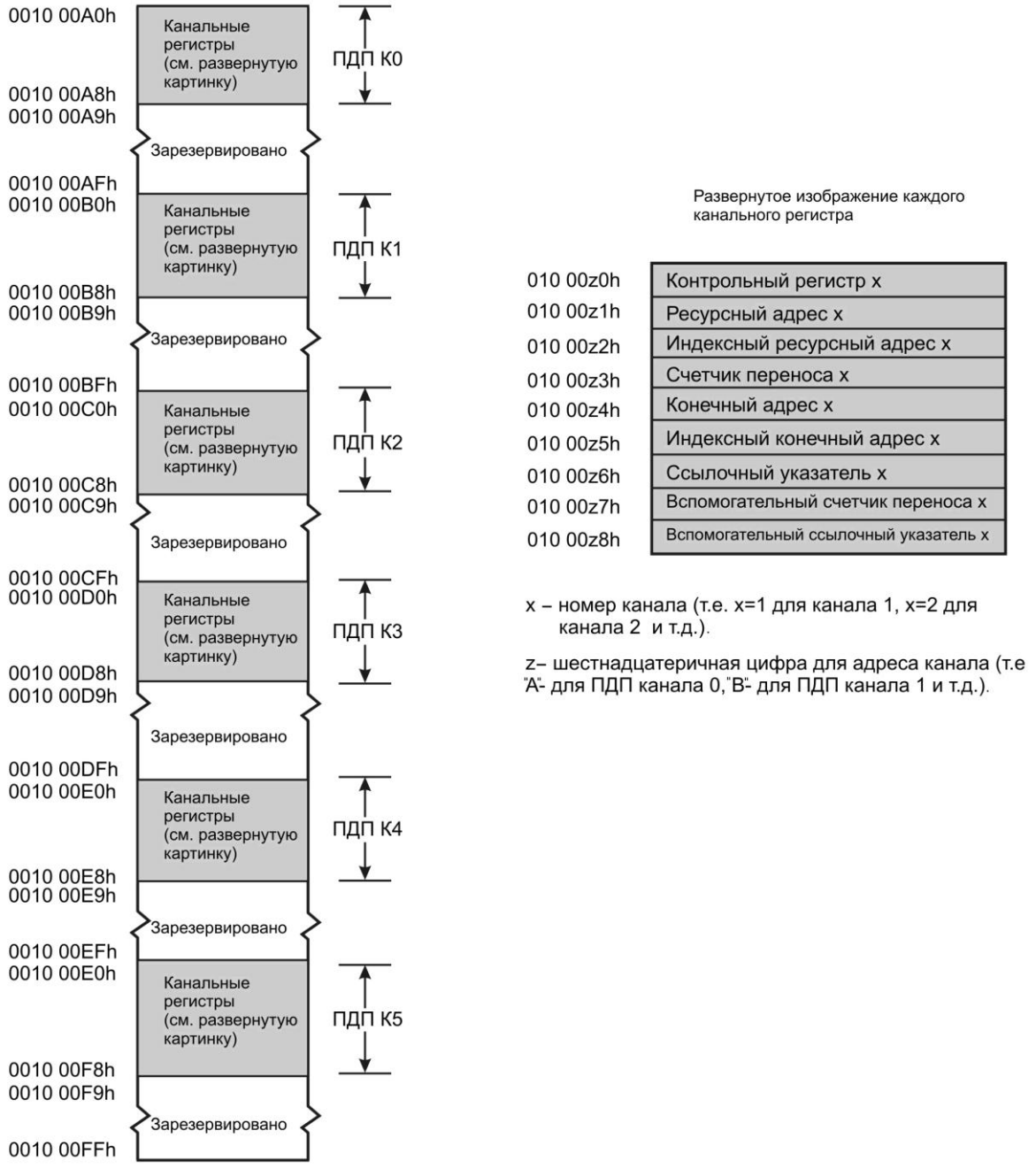


Рисунок 4.5 – Карта памяти сопроцессора ПДП

4.3 КЭШ команд

128 × 32-разрядный КЭШ команд позволяет максимально ускорить выполнение программы при минимальных системных затратах, путем сохранения в памяти КЭШ фрагментов программы, которые могут быть выбраны, когда требуется повторный доступ к этим фрагментам. Внешние шины в этом случае свободны от выборки кодов программы, что может быть использовано сопроцессором ПДП и другими элементами системы.

КЭШ может работать в полностью автоматическом режиме без вмешательства пользователя. В КЭШ реализован алгоритм LRU (Least Recently Used) – вытеснение наименее часто используемых фрагментов программы.

4.3.1 Архитектура КЭШ

КЭШ команд, см. рисунок 4.7, содержит 128 × 32-разрядных слов памяти для хранения 128 слов кода программы. Он делится на четыре 32 словных сегмента. Каждому сегменту соответствует 27-разрядный регистр стартового адреса сегмента SSA. Каждому слову в КЭШ соответствует одноразрядный P (Present) флаг.

Когда ЦПУ выбирает слово команды из внекристальной памяти, то проверяется, не содержится ли слово с таким адесом в КЭШ. Разделение адреса инструкции, используемого алгоритмом управления КЭШ, показано на рисунке 4.6. 27 старших разрядов адреса инструкции сравнивается с двумя регистрами начального адреса сегмента SSA. Если соответствие найдено, то проверяется флаг P. Флаг P показывает, присутствует или нет слово инструкции с таким адресом внутри соответствующего сегмента памяти КЭШ.

Если P = 1, то слово инструкции с таким адресом уже находится в памяти КЭШ, если P = 0, то содержимое памяти КЭШ неактуально (т. е. содержит неправильные данные).

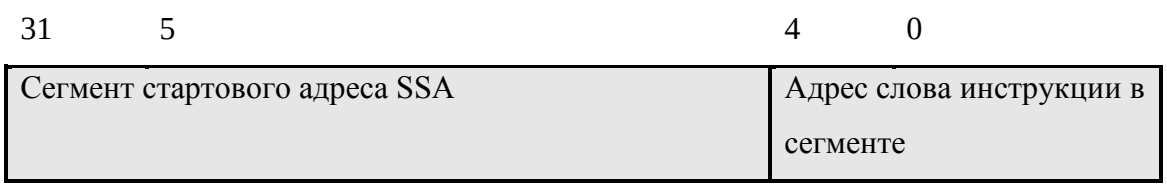


Рисунок 4.6 – Адресное разделение для управляющего алгоритма КЭШ

Если соответствие не найдено, то один из сегментов должен быть заменен новыми данными. Заменяемый сегмент в этом случае определяется алгоритмом LRU. Для этих целей существует стек LRU.

Стек LRU определяет, какой из четырех сегментов определяется как наиболее старый (давно использованный) после каждого доступа к КЭШ. Каждый раз при доступе к сегменту, номер сегмента удаляется из стека LRU и проталкивается в вершину стека LRU. Таким образом, число в вершине стека представляет собой номер последнего использованного сегмента, а число на дне стека – наиболее давнее использование данного сегмента.

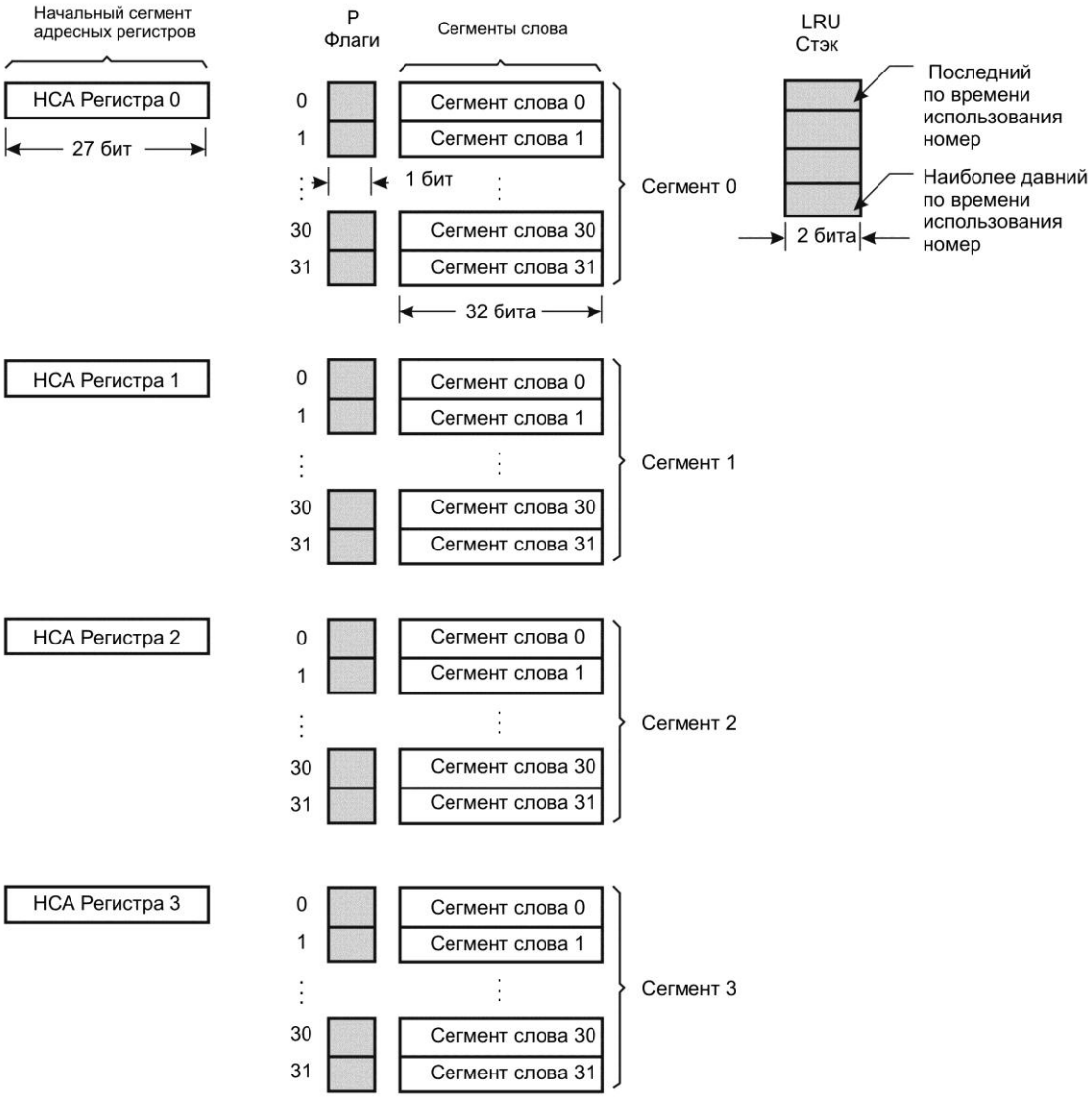


Рисунок 4.7 – Архитектура КЭШ команд

При сбросе происходит следующее:

- КЭШ запрещается ST(CE)=0 и блокируется ST(CF)=1.
- Все флаги P установлены в «0».
- LRU стек инициализируется сегментом «0» в вершине, далее следуют сегменты 1, 2 и 3 ниже. Если любые два регистра SSA эквивалентны (из-за сброса) и КЭШ-сравнение, то выбирается слово команды из наиболее часто используемого сегмента.

Когда необходима замена содержимого сегмента, то заменяется содержимое наиболее давно используемого сегмента. 32 флага P для заменяемого сегмента устанавливаются в «0», и в сегментный регистр SSA записываются 27 старших разрядов адреса инструкции.

4.3.2 Управляющие биты КЭШ

Четыре управляющих бита КЭШ размещены в статусном регистре ЦПУ ST: бит очистки КЭШ – CC, бит активизации КЭШ – CE, бит блокировки КЭШ – CF и бит состояния предыдущей блокировки КЭШ – PCF. Статусный регистр показан на рисунке 3.3.

Описание управляющих битов КЭШ:

- CC – бит очистки КЭШ. Установка CC = 1 делает недействительными все содержимое памяти КЭШ. Этот бит всегда является очищенным после записи; таким

образом, он всегда читается как «0». При сбросе, в этот бит записывается 0. Флаг P = 0, когда КЭШ очищен.

- CE – бит активизации КЭШ. Установка CE = 1 разрешает работу КЭШ, позволяя использовать КЭШ соответственно алгоритму LRU (замещение наиболее поздней по использованию страницы). Установка CE = 0 останавливает работу КЭШ; это не позволяет КЭШ обновляться или изменяться (таким образом, выборка из КЭШ не может выполняться). При сбросе этот бит читается как «0». Очистка КЭШ CC = 1 разрешена, когда CE = 0.

- CF – бит блокировки КЭШ. Установка CF = 1 блокирует КЭШ, включая блокировку LRU (замещения наиболее ранней по использованию страницы) управление стеком. Если КЭШ включен CE = 1 и КЭШ блокирован CF=1, то выборка из КЭШ разрешена, но не разрешены изменения содержимого КЭШ. Очистка КЭШ CC = 1 разрешена, когда CF = 1. При сбросе этот бит очищается в «0», а после сброса устанавливается в «1». Когда CF = 0, то очистка КЭШ разрешена CC = 1. CF устанавливается в «1», когда происходит программное TRAP или системное прерывание. Инструкции RETI и RETID копируют PCF в CF бит.

Таблица 4.2 описывает результат установки различных комбинаций, битов CE и CF.

Таблица 4.2 – Описание различных комбинаций битов CE и CF

Бит CE	Бит CF	Описание
0	0	КЭШ не активизирован
0	1	КЭШ не активизирован
1	0	КЭШ активизирован и не блокирован
1	1	КЭШ активизирован и блокирован

- PCF – бит предыдущего блокирования КЭШ. Когда произошло прерывание, то значение бита CF копируется в бит PCF, и бит CF устанавливается в «1».

Это защищает КЭШ в течение процесса прерывания, и это особенно полезно, когда прерываются циклические участки кода. Программа, обслуживающая прерывания, может быть опционально использована КЭШ под программным управлением. Прерывания могут также быть вложенными, обеспечивая сохранение значений статусного регистра перед прерываниями. Когда инструкции RETIcond и RETIcondD осуществляют процесс выхода из прерывания, то содержимое бита PCF копируется в бит CF.

4.3.2.1 Использование КЭШ

Из КЭШ можно выбирать только коды инструкций. Все чтения и записи данных в/из памяти обходят КЭШ. Все выборки кодов программы из внутренней памяти СОЗУ 0 или СОЗУ 1 не модифицируют КЭШ и не инициируют просмотр КЭШ. Программный КЭШ является блоком памяти с одним доступом за машинный такт (однократный доступ). Фиктивный фрагмент программы, т. е. команды после инструкции перехода, может генерировать несовпадение и обновление КЭШ. Пример 4.1 показывает обычную процедуру сброса и разрешения КЭШ.

Пример 4.1 – Разрешение работы КЭШ

```
OR 1800h, ST
```

Чтобы использовать КЭШ более эффективно, необходимо соблюдать две предосторожности:

1 Нельзя самомодифицировать код программы (изменение кода программы самой программой), поскольку, если инструкция, размещенная в памяти модифицируется, то соответствующая ей инструкция, расположенная в КЭШ, не модифицируется.

2 Необходимо выравнивать программный код. Для этого нужно использовать директиву .align**,** которая выравнивает код языка Ассемблер по 32-словным адресным границам.

4.3.2.2 Алгоритм работы КЭШ

Когда процессор запрашивает слово команды из внешней памяти, то возможны два случая:

- попадание (команда с таким адресом найдена в КЭШ);
- промах (команда с таким адресом отсутствует в КЭШ).

1 случай. КЭШ – команда есть. Если требуемая команда (инструкция с таким же адресом) содержится в КЭШ, то производятся следующие действия:

- первое действие. Командное слово считывается из КЭШ;
- второе действие. Номер сегмента, внутри которого содержится слово инструкции, удаляется из стека LRU и проталкивается на вершину стека LRU, таким образом, перемещая номер другого сегмента на дно стека.

2 случай. КЭШ – команды нет. Команда не содержится в КЭШ, при этом, если слово инструкции не найдено и регистр адреса сегмента совпадает с соответствующими разрядами адреса инструкции, но подходящий флаг не установлен, то следующие действия выполняются параллельно:

- первое действие. Слово инструкции считывается из памяти и копируется в КЭШ;
- второе действие. Номер сегмента, внутри которого содержится слово инструкции, удаляется из стека LRU и проталкивается в вершину стека LRU, таким образом, перемещая номер другого сегмента на дно стека;
- третье действие. Устанавливается соответствующий флаг P.

Если сегмент не найден, т. е. никакой адрес сегмента не соответствует соответствующим разрядам адреса инструкции, то следующие действия выполняются параллельно:

- первое действие. Наиболее редко используемый сегмент выбирается для замены. Флаги P для всех 32 слов очищены;
- второе действие. Регистр SSA для выбранного сегмента загружается 27 старшими разрядами адреса запрошенного командного слова;
- третье действие. Слово инструкции выбирается и копируется в КЭШ. Оно направляется в соответствующую позицию наиболее редко используемого сегмента. Флаг P для этого слова устанавливается в «1»;
- четвертое действие. Номер сегмента, в котором содержится слово, удаляется из стека LRU и проталкивается в вершину стека LRU, таким образом, перемещая номер другого сегмента на дно стека.

5 Форматы данных и операции с плавающей запятой

Архитектура ИС 1867ВМ9Ф поддерживает три основных типа данных: целые, целые без знака и числа в формате с плавающей запятой. Термины целые и целые со знаком будем считать эквивалентными.

ИС 1867ВМ9Ф поддерживает форматы коротких целых и целых с одинарной точностью со знаком и без знака. Микросхема также поддерживает форматы с плавающей запятой – короткие, с одинарной точностью и с повышенной точностью.

Операции с плавающей запятой обеспечивают удобные и безошибочные вычисления. Реализация арифметики с плавающей запятой на ИС 1867ВМ9Ф позволяет производить операции с плавающей запятой со скоростью целочисленных вычислений, и в то же время, предотвращая проблемы с переполнением, выравниванием операндов и с другими общими проблемами целочисленных операций.

В этом разделе подробно обсуждаются форматы данных и операции с плавающей запятой, поддерживаемые ИС 1867ВМ9Ф.

5.1 Целочисленный формат со знаком

ИС 1867ВМ9Ф поддерживает два целочисленных формата: 16-разрядный короткий целый формат и 32-разрядный целый формат с одинарной точностью.

Термин «целый» использован на протяжении всего этого раздела для обозначения целого знакового формата.

Примечание – Когда регистры повышенной точности используются как целые операнды, то используются только биты 31-0; биты 39-32 остаются неизменеными и не используются.

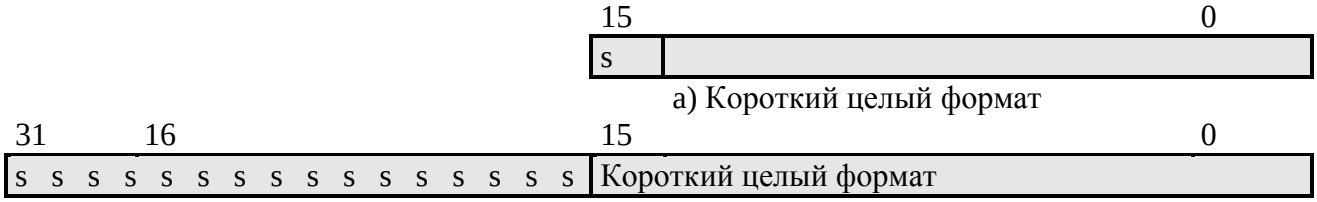
5.1.1 Короткий целочисленный формат

Короткий целочисленный формат – это 16-разрядный формат в двоично-дополнительном коде. Он используется для непосредственных целых операндов. Для тех инструкций, которые содержат целые операнды, происходит расширение знака до 32 разрядов, смотри рисунок 5.1.

Диапазон целого числа si представлен в коротком целом формате

$$-2^{15} \le si \le 2^{15} - 1, \qquad (5.1)$$

На рисунке 5.1 s – знаковый разряд.



a) Короткий целый формат

b) Расширение знака в коротком целом

Рисунок 5.1– Короткий целый формат и расширение знака в коротком целом

5.1.2 Целый формат с одинарной точностью

В целом формате с одинарной точностью целые числа представлены в двоично-дополнительном коде. Диапазон целого числа sp представлен в целом формате с одинарной точностью

$$-2^{31} \le sp \le 2^{31} - 1. \qquad (5.2)$$

На рисунке 5.2 приведен целый формат с одинарной точностью.

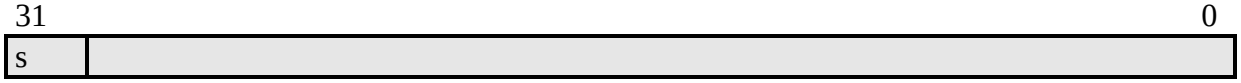


Рисунок 5.2 – Целый формат с одинарной точностью

5.2 Форматы целых чисел без знака

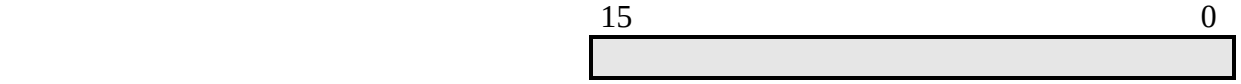
Два целочисленных формата без знака поддерживаются в ИС 1867ВМ9Ф: 16-разрядный короткий формат и 32-разрядный формат с одинарной точностью. В регистрах повышенной точности целые без знака операнды используют только разряды 31-0; разряды 39-32 остаются без изменений.

5.2.1 Короткий целочисленный формат без знака

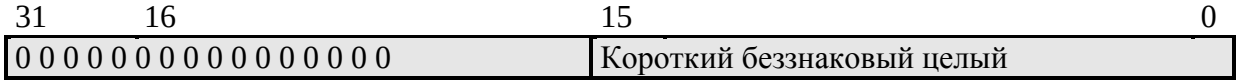
На рисунке 5.3 приведен 16-разрядный короткий целочисленный формат без знака, используемый для непосредственных целых операндов без знака. В инструкциях, которые содержат целые операнды без знака (разряды 16-31 заполнены нулями).

Диапазон целого числа si представлен в коротком целочисленном формате без знака

$$- 0 \le si \le 2^{16}. \qquad (5.3)$$



a) Короткий целый формат без знака



b) Заполнение нулями короткого целого формата без знака

Рисунок 5.3 – Короткий целочисленный формат без знака и с заполнением нулями

5.2.2 Целочисленный формат с одинарной точностью без знака

Число в целочисленном формате с одинарной точностью без знака представлено 32-разрядным значением, как показано на рисунке 5.4.

Диапазон целого числа sp представлен в целочисленном формате с одинарной точностью без знака

$$- 0 \le sp \le 2^{32}. \qquad (5.4)$$

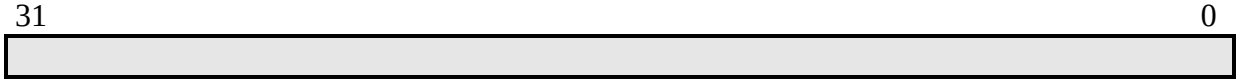


Рисунок 5.4 – Целочисленный формат с одинарной точностью без знака

5.3 Форматы с плавающей запятой

ИС 1867ВМ9Ф поддерживает три формата с плавающей запятой:

- короткий формат с плавающей запятой (для непосредственных операндов с плавающей запятой) содержит 4-разрядный порядок числа, 1 знаковый разряд и 11 разрядов дробной части числа;
- формат числа с одинарной точностью содержит 8-разрядный порядок числа, 1 знаковый разряд и 23-разряда дробной части числа;
- формат числа повышенной точности содержит 8-разрядный порядок числа, 1 знаковый разряд и 31 разряд дробной части числа.

Все форматы чисел с плавающей запятой в ИС 1867ВМ9Ф состоят из трех полей: поля порядка числа (e), поля знакового разряда (s) и поля мантиссы (дробной

части числа) (f). Этот формат показан на рисунке 5.5. Поле знака и поле дробной части числа могут рассматриваться как единая часть и определяются как поле мантиссы (man).

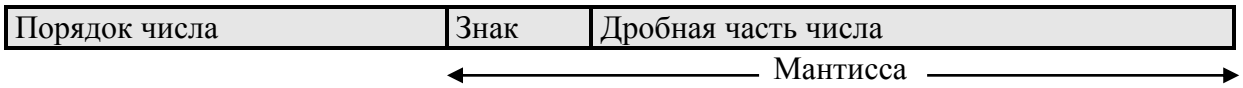


Рисунок 5.5 – Общий формат числа с плавающей запятой

Основное выражение для вычисления значения числа с плавающей запятой дано в формуле 5.5. В выражении s – это значение знакового бита, $\bar{s}$ – это инверсное значение знакового бита, f – двоичное значение поля дробной части числа и e – это десятичный эквивалент поля порядка числа.

$$x = s\bar{s}.f_2 \times 2^e \qquad (5.5)$$

Мантисса представлена нормализованным числом, дополненным до двух. В нормализованном представлении старший незнаковый разряд является неявным, что обеспечивает дополнительный разряд точности.

Использование неявного знакового разряда:

- если s = 0, тогда два старших разряда мантиссы – 01;
- если s = 1, тогда два старших разряда мантиссы – 10.

Если единичный бит s равен «0», то мантисса становится равной «$01.f_2$», где f – двоичное представление дробной части. Если s = 1, то мантисса становится равной «$10.f_2$», где f – двоичное представление дробной части.

Например, если $f=00000000001_2$ и s=0, то значение мантиссы (man) будет равно 01.00000000001_2. Если s=1, то значение man будет 10.00000000001_2.

Поле порядка числа – это число в двоично-дополнительном коде. По существу, поле экспоненты сдвигает двоичную запятую к мантиссе. Если порядок положительный, то двоичная запятая сдвигается вправо. Если порядок числа отрицательный, то двоичная запятая сдвигается влево.

Например, если $man=01.00000000001_2$ и $e=11_{10}$, тогда двоичная запятая переместится на одиннадцать разрядов вправо: число = 0100000000001_2, которое эквивалентно десятичному – 2049_{10}.

5.3.1 Короткий формат числа с плавающей запятой

В коротком формате с плавающей запятой число с плавающей запятой представляется 4-разрядным полем экспоненты (e) в дополнительном коде и 12-разрядным полем мантиссы (man) в дополнительном коде с неявным старшим незнаковым разрядом, смотри рисунок 5.6.

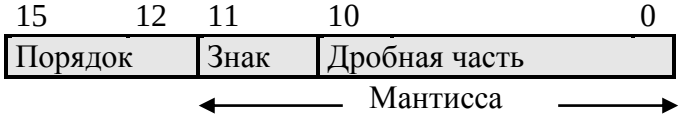


Рисунок 5.6 – Короткий формат числа с плавающей запятой

Для представления нуля в формате с плавающей запятой с одинарной точностью должны быть использованы следующие зарезервированные значения:

e = –8, s = 0 и f = 0.

Операции выполняются с неявной двоичной запятой между 10 и 11 разрядами числа. Число x с плавающей запятой в дополнительном коде в коротком формате с плавающей запятой представляется как:

$x = 01.f_2 \times 2^e$, если s = 0;

$x = 10.f_2 \times 2^e$, если s = 1;

x = 0, если e = –8, s = 0, f = 0.

Следующие примеры иллюстрируют диапазон и точность короткого формата с плавающей запятой:

- Наибольшее положительное число $x = (2 - 2^{-11}) \times 2^7 = 2.5594 \times 10^2$.
- Наименьшее положительное число $x = 1 \times 2^{-7} = 7.8125 \times 10^{-3}$.
- Наибольшее отрицательное число $x = (-1 - 2^{-11}) \times 2^{-7} = -7.8163 \times 10^{-3}$.
- Наименьшее отрицательное число $x = -2 \times 2^7 = -2.5600 \times 10^2$.

5.3.2 Формат числа с плавающей запятой с одинарной точностью

В формате с одинарной точностью число с плавающей запятой представлено 8-разрядным полем порядка числа (e) и 24-разрядным полем мантиссы (man) в дополнительном коде с неявным старшим незнаковым разрядом.

Операции выполняются с неявной двоичной запятой между 23 и 22 разрядами. Когда неявный старший незнаковый разряд определен, то он размещается непосредственно слева от двоичной запятой. Число x с плавающей запятой представляется как:

$x = 01.f \times 2^e$, если s = 0,

$x = 10.f \times 2^e$, если s = 1,

x = 0, если e = –128.

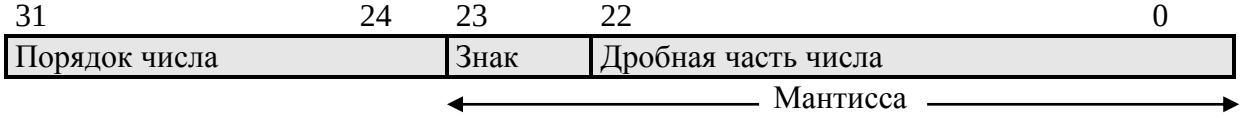


Рисунок 5.7 – Формат числа с плавающей запятой с одинарной точностью

Для представления нуля в формате с плавающей запятой с одинарной точностью должны быть использованы следующие зарезервированные значения:

e = –128, s = 0, f = 0

Следующие примеры иллюстрируют диапазон и точность формата с плавающей запятой с одинарной точностью:

- Наибольшее положительное число $x = (2 - 2^{-23}) \times 2^{127} = 3.4028234 \times 10^{38}$.
- Наименьшее положительное число $x = 1 \times 2^{-127} = 5.8774717 \times 10^{-39}$.
- Наибольшее отрицательное число $x = (-1 - 2^{-23}) \times 2^{-127} = -5.8774724 \times 10^{-39}$.
- Наименьшее отрицательное число $x = -2 \times 2^{127} = -3.4028236 \times 10^{38}$.

5.3.3 Формат числа с плавающей запятой с повышенной точностью

В формате числа с повышенной точностью число с плавающей запятой представляется 8-разрядным полем порядка числа (e) и 32-разрядным полем мантиссы (man) с неявным старшим незнаковым разрядом.

Операции выполняются с неявной двоичной запятой между 31 и 30 разрядами. Когда неявный старший незнаковый разряд определен, то он размещается непосредственно слева от двоичной запятой, см. рисунок 5.8. Число x с плавающей запятой устанавливается как:

- $x = 01.f \times 2^e$, если s = 0.
- $x = 10.f \times 2^e$, если s = 1.
- x = 0, если e = –128, s = 0, f = 0.

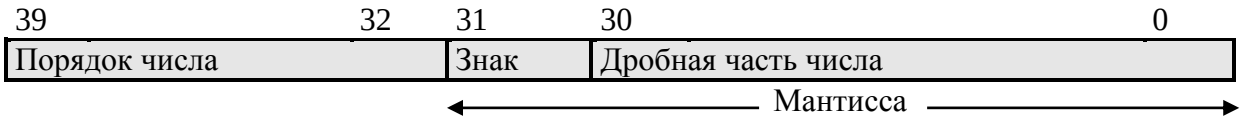


Рисунок 5.8 – Формат числа с плавающей запятой с повышенной точностью

Для представления нуля в формате с плавающей запятой с повышенной точностью должны использоваться следующие зарезервированные значения:

e = –128, s = 0, f = 0.

Следующие примеры иллюстрируют диапазон и точность формата с повышенной точностью с плавающей запятой:

- Наибольшее положительное число $x = (2 - 2^{-31}) \times 2^{127} = 3.4028236683 \times 10^{38}$.
- Наименьшее положительное число $x = 1 \times 2^{-127} = 5.8774717541 \times 10^{-39}$.
- Наибольшее отрицательное число $x = (-1 - 2^{-31}) \times 2^{-127} = -5.8774717569 \times 10^{-39}$.
- Наименьшее отрицательное число $x = -2 \times 2^{127} = -3.4028236691 \times 10^{38}$.

5.3.4 Определение десятичного эквивалента числа с плавающей запятой

Определение десятичного значения числа, сохраненного в формате с плавающей запятой, происходит в два этапа:

- определение значений порядка числа и мантиссы;
- смещение двоичной запятой в мантиссе в соответствии со значением порядка числа и дальнейшее преобразование числа в десятичное значение.

Этап 1: Определение значений порядка числа и мантиссы

Поле порядка числа – число в дополнительном коде, которое зависит от типа преобразуемого числа с плавающей запятой. Например, если преобразуется число с плавающей запятой с одинарной точностью и двоичное значение поля порядка числа равно 00000100, то десятичное значение экспоненты будет равно 4. С другой стороны, если двоичное значение поля порядка числа равно 11111100_2, то десятичное значение порядка числа будет равно –4. Так как первый бит слева равен 1, значит число отрицательное. Значение числа вычисляется путем побитной инверсии числа в дополнительном коде – 11111100_2, которая равна 00000011_2 и добавления «1» к результату.

Примечание – Если значение порядка числа совпадает со значением, зарезервированным для нуля, то число с плавающей запятой равно нулю. Резервное значение для каждого формата числа с плавающей запятой приведено в 5.3.

Мантисса – это двоичное число с неявной двоичной запятой между знаковым разрядом и дробной частью числа. Мантисса формируется двумя способами:

Если s = 0, то мантисса формируется посредством записи «01» и дополнением разрядов в поле дробной части после двоичной запятой.

Например, если $f = 10100000000_2$, тогда $man = 01.1010000000_2$:

s		Дробь										
0	.	1	0	1	0	0	0	0	0	0	0	0

Перезаписывается мантисса как:

		Мантисса										
0	1	.	1	0	1	0	0	0	0	0	0	0

Если s = 1, то мантисса формируется посредством записи 10. и дополнением разрядов в поле дробной части после двоичной запятой.

Например, если $f = 10100000000_2$, тогда $man = 10.1010000000_2$:

s		Дробь										
1	.	1	0	1	0	0	0	0	0	0	0	0

Перезаписывается мантисса как:

		Мантисса										
1	0	.	1	0	1	0	0	0	0	0	0	0

Этап 2: Смещение двоичной запятой в мантиссе в соответствии со значением порядка числа и дальнейшее преобразование числа в десятичное

Если порядок числа (e) имеет положительное значение, то двоичная запятая e смещается вправо.

Если порядок числа (e) имеет отрицательное значение, то двоичная запятая e смещается влево.

Например, если e = 2_{10} и man = 01.11000000000_2, тогда измененная мантисса становится равной 0111.000000000_2, что эквивалентно «7» в десятичной системе счисления.

Если e = $–2_{10}$ и man = 01.10000000000_2, тогда измененная мантисса становится равной $.0110000000000_2$, что эквивалентно «3/8» в десятичной системе счисления.

Следующий пример иллюстрирует, как можно получить эквивалент числа с плавающей запятой в десятичной форме. В каждом примере используется формат числа с плавающей запятой с одинарной точностью.

Пример 5.1 – Положительное число

0	2	4	0	0	0	0	0	Шестнадцатеричное значение
0000	0010	0100	0000	0000	0000	0000	0000	Двоичное значение

Порядок числа = $0000\ 0010_2 = 2$
Знак = 0
Дробная часть = $.10000_2$
Значение = $01.1_2 \times 2^2 = 0110_2. = 6$

Пример 5.2 – Отрицательное число

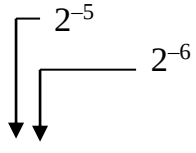
0	1	C	0	0	0	0	0	Шестнадцатеричное значение
0000	0001	1100	0000	0000	0000	0000	0000	Двоичное значение

Экспонента = $0000\ 0001_2 = 1$
Знак = 1
Дробь = $.10000_2$
Значение = $10.1_2 \times 2^1 = 101_2. = -3$

Пример 5.3 – Дробное число

F	B	4	0	0	0	0	0	Шестнадцатеричное значение
1111	1011	0100	0000	0000	0000	0000	0000	Двоичное значение

Порядок числа = $1111\ 1001_2 = –5$
Знак = 0
Дробная часть = $.10000_2$

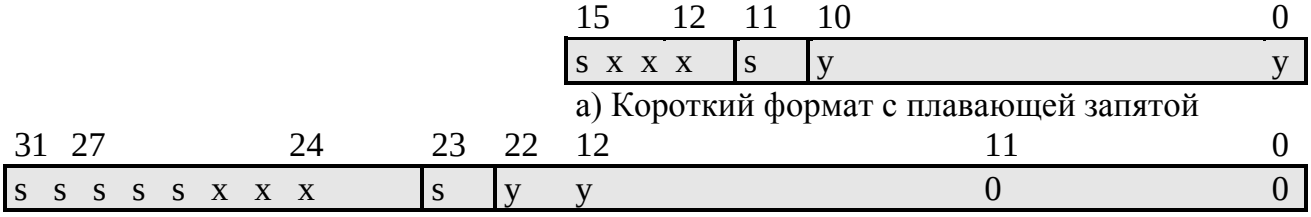


Значение = $01.1_2 \times 2^{-5} = 101_2.= 000011_2 = 3/64$

5.3.5 Преобразование чисел с плавающей запятой из одного формата в другой

Операции с плавающей запятой предполагают различные форматы для ввода и вывода. Эти форматы часто требуют преобразования из одного формата с плавающей запятой в другой (например, короткий формат с плавающей запятой в формат с плавающей запятой с повышенной точностью). Преобразования формата происходят

автоматически (аппаратно), без дополнительных затрат как часть операций с плавающей запятой. Четыре типа преобразования с примерами показаны на рисунках 5.9 – 5.12 (s – знаковый разряд порядка числа). Когда число в формате с плавающей запятой с нулевым значением преобразуется в формат с большей точностью, то оно всегда преобразуется в представление нуля в данном формате.

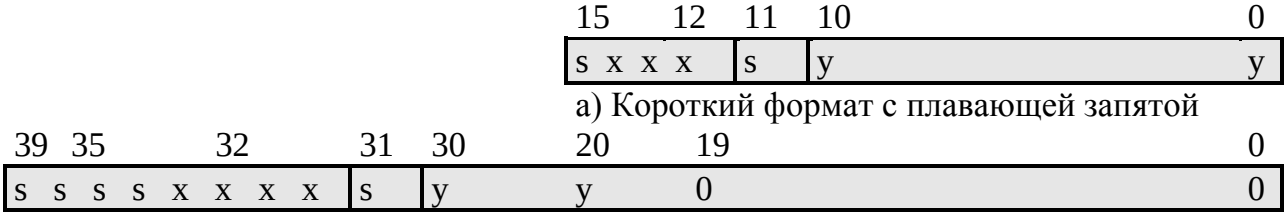


a) Короткий формат с плавающей запятой

b) Формат с плавающей запятой одинарной точности

Рисунок 5.9 – Преобразование короткого формата с плавающей запятой в формат с плавающей запятой одинарной точности

При преобразовании из короткого формата в формат с одинарной точностью поле порядка числа дополняется знаковым разрядом, а крайние справа 12 разрядов дробного поля заполняются нулями.

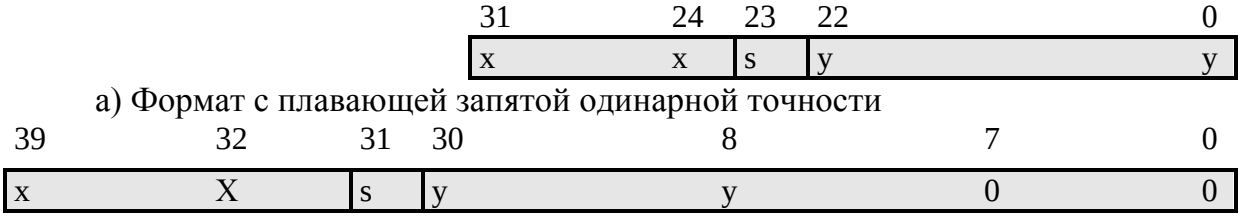


a) Короткий формат с плавающей запятой

b) Формат повышенной точности с плавающей запятой

Рисунок 5.10 – Преобразование короткого формата с плавающей запятой в формат с плавающей запятой повышенной точности

При преобразовании из короткого формата в формат с повышенной точностью поле порядка числа дополняется знаковым разрядом, а крайние справа 20 разрядов дробного поля заполняются нулями



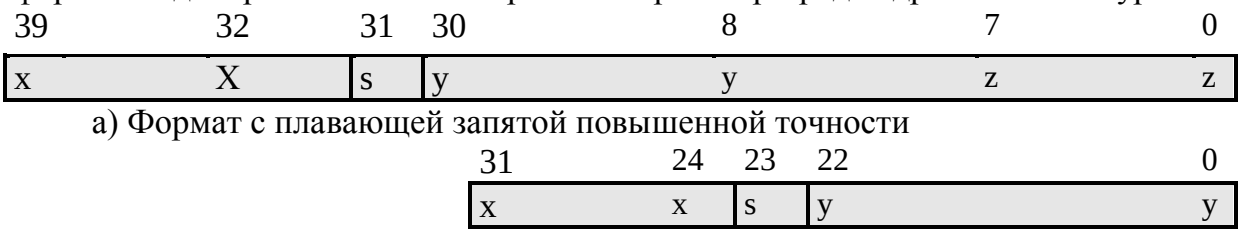
a) Формат с плавающей запятой одинарной точности

b) Формат повышенной точности с плавающей запятой

Рисунок 5.11 – Преобразование формата с плавающей запятой одинарной точности в формат с плавающей запятой повышенной точности

При преобразовании из формата с одинарной точностью в формат с повышенной точностью крайние справа 8 разрядов дробного поля заполняются нулями.

Примечание – При преобразовании из формата с повышенной точностью в формат с одинарной точностью крайние справа 8 разрядов дробного поля урезаются.



a) Формат с плавающей запятой повышенной точности

b) Формат с плавающей запятой одинарной точности

Рисунок 5.12 – Преобразование формата с плавающей запятой повышенной точности в формат с плавающей запятой одинарной точности

5.4 Преобразование формата с плавающей запятой в формат IEEE Std. 754 с помощью инструкций TOIEEE и FRIEEE

Формат с плавающей запятой ИС 1867ВМ9Ф не совместим с форматом IEEE Std. 754. Однако ИС 1867ВМ9Ф имеет инструкции TOIEEE и FRIEEE для преобразования в IEEE формат и обратно, соответственно. Процесс преобразования объясняется в 5.4.1 и 5.4.2. Рисунок 5.13 показывает IEEE формат с плавающей запятой, а рисунок 5.14 показывает формат с плавающей запятой ИС.

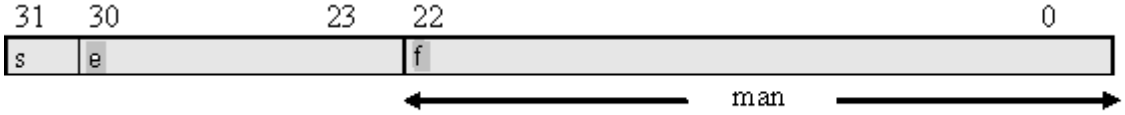


Рисунок 5.13 – IEEE Std. 754 формат с плавающей запятой одинарной точности

Следующие пять случаев определяют значение v чисел, выраженных в IEEE формате:

1) Если e = 255 и f ≠ 0, тогда v = NaN.
2) Если e = 255 и f = 0, тогда v = $(-1)^s$ бесконечность.
3) Если 0<e<255, тогда v = $(-1)^s \times 2^{e-127}(1.f)$.
4) Если e = 0 и f ≠ 0, тогда v = $(-1)^s \times 2^{-126}(0.f)$.
5) Если e = 255 и f = 0, тогда v = $(-1)^s$ x 0(ноль),

где s = знаковый разряд;
e = поле экспоненты;
f = поле дробной части;
NaN = число не определено, v – значение числа.

В представленных выше пяти случаях e рассматривается как целое без знака. Случай 1) генерирует NaN (число не определено) и, в основном, используется для сообщения программе. Случай 4) представляет ненормализованное число. Случай 5) представляет положительный и отрицательный ноль.

Рисунок 5.14 показывает формат числа с плавающей запятой в дополнительном коде, который имеет ИС 1867ВМ9Ф. В этом формате могут быть использованы два случая для определения значения v числа:

- Если e = –128 и f ≠ 0, тогда v = 0.
- Если e ≠ –128 тогда $v = s\bar{s}.f_2 \times 2^e$,

где s = знаковый разряд; e = поле экспоненты; f = поле дробной части числа.

При этом e интерпретируется как целое число в дополнительном коде. Дробная часть числа и знаковый разряд формируют нормализованную мантиссу в дополнительном коде.

Примечание – Необходимо различать символы для IEEE и ИС 1867ВМ9Ф форматов. Различать символы, определяющие эти два формата, позволяют индексы: все IEEE поля имеют индексы IEEE, например, e_{IEEE}, s_{IEEE} и т. д. Таким же образом все поля в двоично-дополнительном коде имеют индекс «two», т. е. e_{two}, s_{two}, f_{two}.

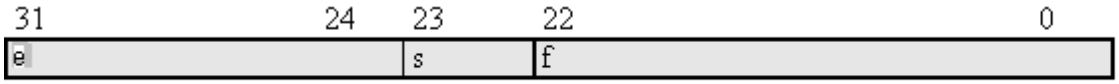


Рисунок 5.14 – Формат ИС 1867ВМ9Ф с плавающей запятой с одинарной точностью в дополнительном коде

5.4.1 Преобразование IEEE формата в формат числа с плавающей запятой в дополнительном коде ИС

Преобразование формата IEEE в дополнительный код является наиболее общим. Это преобразование выполняется в соответствии с правилами, указанными в таблице 5.1.

Таблица 5.1 – Преобразование IEEE формата в формат числа с плавающей запятой в дополнительном коде

Случай	Если имеются эти значения			Эквивалентные значения			
	e_{IEEE}	s_{IEEE}	f_{IEEE}	e_{two}	s_{two}	f_{two}	s_{IEEE}
1	255	1	–	7Fh	1	00 0000h	–
2	255	0	–	7Fh	0	7F FFFFh	–
3	$0< e_{IEEE} <255$	0	–	e_{IEEE}–7Fh	–	f_{IEEE}	0
4	$0< e_{IEEE} <255$	1	≠0	e_{IEEE}–7Fh	–	$\bar{f}_{IEEE}+1$	1
5	$0< e_{IEEE} <255$	1	0	e_{IEEE}–80h	–	0	1
6	0	–	–	80h	0	00 0000h	–

Случай 1 иллюстрирует преобразование положительных NaN чисел и положительных бесконечных чисел в формате IEEE в наибольшее положительное число с одинарной точностью в дополнительном коде. При этом появляется признак переполнения, что позволяет проверить эти специальные случаи.

Случай 2 иллюстрирует преобразование IEEE отрицательных NaN чисел и отрицательных бесконечных чисел в наименьшее отрицательное число с одинарной точностью в дополнительном коде. При этом появляется признак переполнения, что позволяет проверять эти специальные случаи.

Случай 3 иллюстрирует преобразование IEEE положительных нормализованных чисел в эквивалентные положительные значения в дополнительном коде.

Случай 4 иллюстрирует преобразование IEEE отрицательных нормализованных чисел с ненулевой дробной частью в эквивалентные отрицательные числа в дополнительном коде.

Случай 5 иллюстрирует преобразование IEEE отрицательных нормализованных чисел с нулевой дробной частью в эквивалентные отрицательные числа в дополнительном коде.

Случай 6 иллюстрирует преобразование IEEE положительных и отрицательных ненормализованных чисел, а также положительных и отрицательных нулей в эквивалентные числа в дополнительном коде.

ИС 1867ВМ9Ф предполагает, что числа в формате IEEE хранятся в памяти или в регистре как целые числа. Когда процессор преобразует IEEE число, то он помещает это число в регистр повышенной точности, используя в этом регистре поля порядка и дробной части числа. Восемь самых младших разрядов регистра с повышенной точностью устанавливаются в «0». Любые арифметические операции, которые выполняются над дробной частью IEEE числа, должны выполняться только над дробной частью IEEE числа. В случае передачи данных через блок памяти при использовании параллельных инструкций с STF, преобразование формата происходит без потери данных. Пример 5.4 иллюстрирует, как это может быть выполнено.

Пример 5.4 – Преобразование числа в формате IEEE в формат ИС 1867ВМ9Ф при передаче данных через блок памяти

```
* Преобразование IEEE в формат числа ИС 1867ВМ9Ф при передаче данных через блок памяти
*
* ПРОГРАММА ПРЕДПОЛАГАЕТ, ЧТО ВХОДНОЙ БУФЕР FIFO КОММУНИКАЦИОННОГО ПОРТА 0
* ЗАПОЛНЕНИЕ  ДАННЫМИ ФОРМАТА  IEEE. ВОСЕМЬ СЛОВ
* ПЕРЕДАЮТСЯ ИЗ КОММУНИКАЦИОННОГО ПОРТА 0 В БЛОК 0 ВНУТРЕННЕГО ОЗУ
* ФОРМАТ ДАННЫХ ПРЕОБРАЗУЕТСЯ ИЗ  IEEE ФОРМАТА
* В ФОРМАТ ИС 1867ВМ9Ф С ПЛАВАЮЩЕЙ ЗАПЯТОЙ
*
        ...
        …
        …
        LDI @CP0_IN, AR0        ; Загрузка адреса входного буфера FIFO коммуникационного порта 0
        LDI @RAM0, AR1          ; Загрузка адреса БЛОКА 0 внутреннего ОЗУ
        FRIEEE AR0, R0          ; Первое преобразование данных
        RPTS 6
        FRIEEE AR0, R0          ; Следующее преобразование данных
    ||STF R0,*AR1 ++ (1); Сохранение предыдущих данных
        STF R0,*AR1 ++ (1)      ; Сохранение последних данных
        …       …
```

5.4.2 Преобразование числа из формата ИС 1867ВМ9Ф с плавающей запятой в дополнительном коде в IEEE формат

Это преобразование выполняется так, как показано в таблице 5.2.

Таблица 5.2 – Преобразование числа с плавающей запятой в дополнительном коде в формате ИС 1867ВМ9Ф в формат IEEE

Случай	Если имеются эти значения			Тогда эквивалентные им значения		
	e_{two}	s_{two}	f_{two}	e_{IEEE}	s_{IEEE}	f_{IEEE}
1	–128	–	–	00h	0	00 0000h
2	–127	–	–	00h	0	00 0000h
3	$-126 \le e_{two} \le 127$	0	–	e_{two}+7Fh	0	f_{two}
4	$-126 \le e_{two} \le 127$	1	≠0	e_{two}+7Fh	0	$\bar{f}_{IEEE}+1$
5	$-126 \le e_{two} \le 127$	1	0	e_{two}+80h	1	00 0000h
6	127	1	0	FFh	1	00 0000h

Случай 1 иллюстрирует преобразование нуля в дополнительном коде в положительный ноль IEEE.

Случай 2 иллюстрирует преобразование слишком малых чисел в дополнительном коде, которые не могут быть нормализованы в формат IEEE, в положительный ноль IEEE.

Случай 3 иллюстрирует преобразование положительных чисел в дополнительном коде, которые не относятся к случаю 2, в эквивалентные числа IEEE.

Случай 4 иллюстрирует преобразование отрицательных чисел в дополнительном коде с ненулевой дробью, которые не относятся к случаю 2, в эквивалентные числа IEEE.

Случай 5 иллюстрирует преобразование всех отрицательных чисел в дополнительном коде с нулевой дробью, исключая наименьшее отрицательное число в дополнительном коде, а также чисел, которые не относятся к случаю 2, в эквивалентное число IEEE.

Случай 6 иллюстрирует преобразование наименьшего отрицательного числа в дополнительном коде в бесконечное отрицательное число в формате IEEE.

ИС 1867ВМ9Ф предполагает, что числа в дополнительном коде хранятся в памяти или в регистре повышенной точности в полях порядка числа или дробной части числа регистра (как показано на рисунке 5.14). Если число расположено в регистре повышенной точности, то только 24 старших разряда дробной части числа рассматриваются как поле дробной части числа и для выявления специальных случаев. Результат преобразования записывается в 32 старших разряда регистра повышенной точности. В случае передачи данных через блок памяти при использовании параллельных инструкций с STF, преобразование формата происходит без потери данных. Пример 5.5 иллюстрирует, как это может быть выполнено.

Пример 5.5 – Преобразование числа в формате ИС 1867ВМ9Ф в IEEE формат при передаче данных через блок памяти

```
* Преобразование числа в формате ИС 1867ВМ9Ф в формат IEEE при передаче данных через блок памяти
* ПРОГРАММА ПРЕДПОЛАГАЕТ, ЧТО ВЫХОДНОЙ БУФЕР FIFO КОММУНИКАЦИОННОГО ПОРТА 0
* ПУСТОЙ. ВОСЕМЬ СЛОВ  ПЕРЕДАЮТСЯ ИЗ БЛОКА 0 ВНУТРЕННЕГО ОЗУ  В
* КОММУНИКАЦИОННЫЙ ПОРТ 0
* ФОРМАТ ДАННЫХ ПРЕОБРАЗУЕТСЯ ИЗ ФОРМАТА ИС 1867ВМ9Ф С ПЛАВАЮЩЕЙ ЗАПЯТОЙ
* В IEEE ФОРМАТ
  …
  LDI @CP0_OUT, AR0      ; Загрузка адреса выходного буфера FIFO коммуникационного порта 0
  LDI @RAM0, AR1         ; Загрузка адреса БЛОКА 0 внутреннего ОЗУ
  TOIEEE*AR1 ++ (1), R0  ; Первое преобразование данных
  RPTS 6
  TOIEEE*AR1 ++ (1), R0  ; Следующее преобразование данных
  || STF R0,*AR0         ; Сохранение предыдущих данных
  STF R0,*AR0            ; Сохранение последних данных
```

```
  …
```

5.5 Умножение чисел с плавающей запятой

Число с плавающей запятой может быть записано в формате с плавающей запятой согласно формуле, где a(man) – мантисса и a(exp) – порядок числа.

$$a = a(man) \times 2^{a(exp)} \tag{5.6}$$

Результатом умножения чисел a и b является число c, определяемое следующим образом:

$$c = a \times b = a(man) \times b(man) \times 2^{(a(exp) + b(exp))}; \tag{5.7}$$

$$c(man) = a(man) \times b(man);$$

$$c(exp) = a(exp) + b(exp).$$

Когда выполняется умножение с плавающей запятой, то предполагается, что исходные операнды всегда будут в формате с плавающей запятой повышенной точности. Если исходные операнды в коротком формате или в формате с одинарной точностью, то они преобразуются в формат с плавающей запятой повышенной точности.

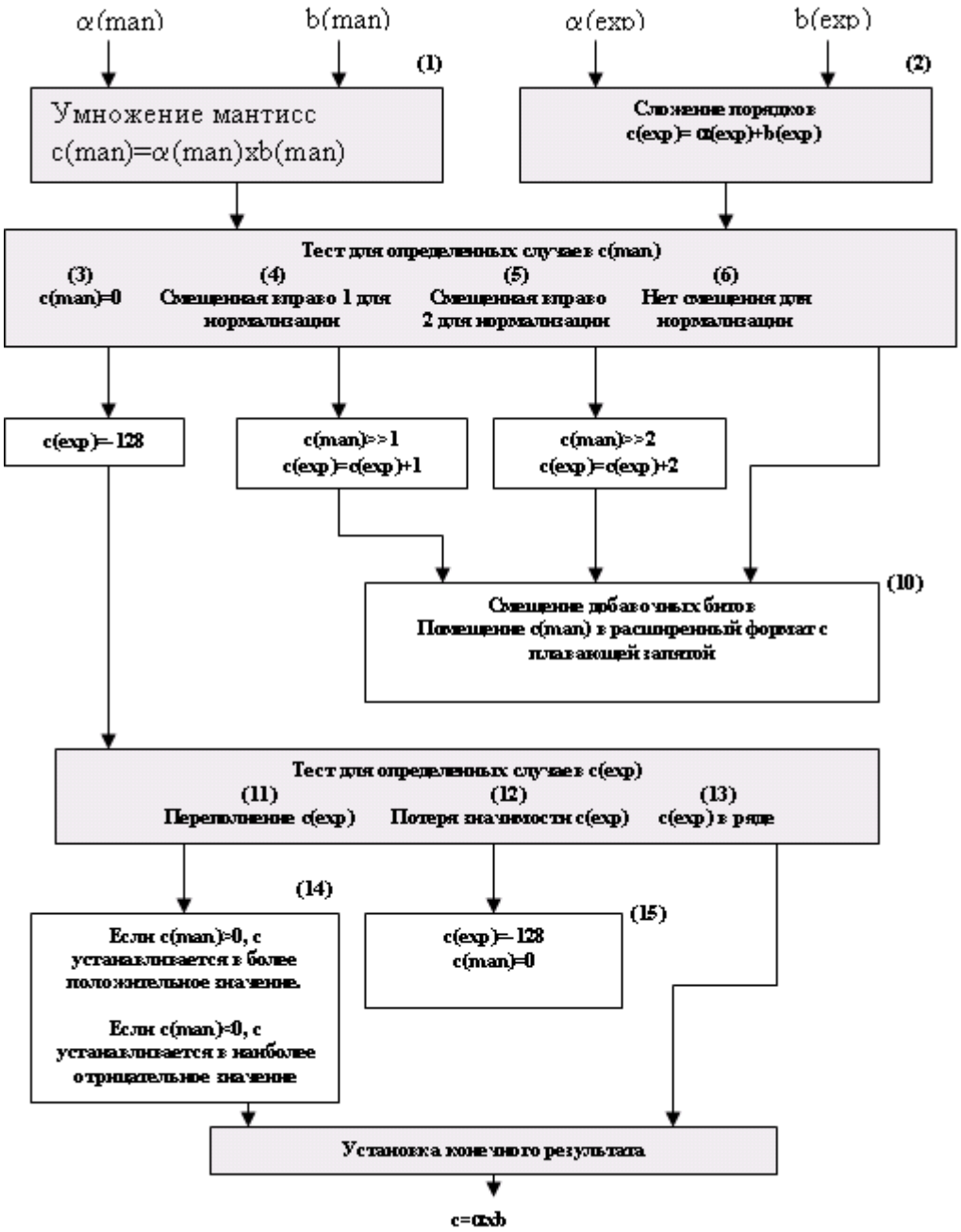


Рисунок 5.15 – Блок-схема операции умножения с плавающей запятой

Эти преобразования производятся автоматически (аппаратно), без дополнительных затрат. Все результаты умножения с плавающей запятой предполагают формат числа с повышенной точностью. Эти умножения происходят в одном цикле.

Рассмотрим блок-схему, представленную на рисунке 5.15.

На 1 этапе 32-разрядные мантиссы операндов источника перемножаются, и на выходе получается 64-разрядный результат c(man). Отметим, что входные и выходные данные всегда представлены нормализованными числами.

На этапе 2 складываются экспоненты, получается c(exp). Этапы 3 – 6 используются для проверки специальных случаев.

На этапе 3 мантисса проверяется на равенство нулю c(man), которая представлена в формате с повышенной точностью. Если c(man) равно нулю, то на этапе 7 устанавливается c(exp) = –128, обеспечивая, таким образом, представление нуля.

Этапы 4 и 5 нормализуют результат. Если необходим сдвиг вправо на один разряд, то на этапе 8 c(man) сдвигается вправо на один разряд и к c(exp) прибавляется единица. Если же необходим сдвиг вправо на два разряда, то на этапе 9 c(man) сдви-

гается вправо на два разряда и к c(exp) добавляется два. Этап 6 выполняется, когда результат нормализован.

На этапе 10 c(man) приводится к формату с плавающей запятой повышенной точности. Этапы с 11 по 13 предназначены для проверки специальных случаев с c(exp). На этапе 14, если c(exp) переполнено в положительном направлении (этап 11), то c(exp) присваивается наибольшее положительное значение в формате с повышенной точностью. Если c(exp) переполнено в отрицательном направлении, то c(exp) присваивается наименьшее отрицательное значение в формате с повышенной точностью. Если произошла потеря значимости c(exp) (исчезновение значащих разрядов) (этап 12), то c присваивается значение ноль (этап 15), т. е. c(man) = 0 и c(exp) = –128.

Следующие примеры иллюстрируют, как выполняется умножение с плавающей запятой. Для этих примеров неявный старший незнаковый разряд задан явно.

Пример 5.6 – Умножение с плавающей запятой (обе мантиссы = –2.0).

Дано:

$a = -2.0 \times 2^{a(exp)} = 10.00000000000000000000000 \times 2^{a(exp)}$

$b = -2.0 \times 2^{b(exp)} = 10.00000000000000000000000 \times 2^{b(exp)}$,

где a и b оба представлены в двоичном виде в соответствии с нормализованным форматом с плавающей запятой с одинарной точностью. Тогда:

$$\begin{array}{l} 10.00000000000000000000000 \times 2^{a(exp)} \\ \times\ 10.00000000000000000000000 \times 2^{b(exp)} \\ \hline 0100.00000000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp))} \end{array}$$

Для приведения этого числа в соответствующий нормализованный формат, необходимо сдвинуть мантиссу на два разряда вправо и прибавить двойку к порядку числа. Получается:

$$\begin{array}{l} 10.00000000000000000000000 \times 2^{a(exp)} \\ \times\ 10.00000000000000000000000 \times 2^{b(exp)} \\ \hline 01.00000000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp)+2)} \end{array}$$

При умножении с плавающей запятой порядок числа результата может переполниться, когда порядок числа изменяется при выполнении нормализации.

Пример 5.7 – Умножение с плавающей запятой (обе мантиссы =1.5).

Дано:

$a = 1.5 \times 2^{a(exp)} = 01.10000000000000000000000 \times 2^{a(exp)}$

$b = 1.5 \times 2^{b(exp)} = 01.10000000000000000000000 \times 2^{b(exp)}$,

где a и b оба представлены в двоичном виде в соответствии с форматом с плавающей запятой повышенной точности. Тогда:

$$\begin{array}{l} 01.10000000000000000000000 \times 2^{a(exp)} \\ \times\ 01.10000000000000000000000 \times 2^{b(exp)} \\ \hline 0010.00000000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp))} \end{array}$$

Для приведения этого числа в соответствующий нормализованный формат, необходимо сдвинуть мантиссу на один разряд вправо и прибавить единицу к порядку числа.

$$\begin{array}{l} 01.10000000000000000000000 \times 2\ a(exp) \\ \times\ 01.10000000000000000000000 \times 2\ b(exp) \\ \hline 01.00100000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp)+1)} \end{array}$$

Пример 5.8 – Умножение с плавающей запятой (обе мантиссы =1.0).

Дано:

$a = 1.0 \times 2^{a(exp)} = 01.00000000000000000000000 \times 2^{a(exp)}$

$b = 1.0 \times 2^{b(exp)} = 01.00000000000000000000000 \times 2^{b(exp)}$,

где a и b оба представлены в двоичном виде соответственно формату с плавающей запятой одинарной точности. Тогда:

$$01.00000000000000000000000 \times 2^{a(exp)}$$
$$\times \quad 01.00000000000000000000000 \times 2^{b(exp)}$$

$$0001.00000000000000000000000000000000000000000000000 * 2^{(a(exp)+b(exp))}$$

Это число представлено в нормализованном формате. Следовательно, нет необходимости сдвигать мантиссу или изменять порядок числа.

В этих примерах рассмотрены случаи, где результатом действий над двумя нормализованными числами может быть нормализованное число со сдвигом на ноль, один или два.

Пример 5.9 – Умножение с плавающей запятой между положительным и отрицательным числами.

Дано:

$a = 1.0 \times 2^{a(exp)} = 01.00000000000000000000000 \times 2^{a(exp)}$

$b = 2.0 \times 2^{b(exp)} = 10.00000000000000000000000 \times 2^{b(exp)}$

Тогда:

$$01.00000000000000000000000 \times 2^{a(exp)}$$
$$\times \quad 10.00000000000000000000000 \times 2^{b(exp)}$$

$$1110.00000000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp))}$$

Результатом будет $c = -2.0 \times 2^{(a(exp)+b(exp))}$

Умножение числа с плавающей запятой на ноль

Любое умножение с плавающей запятой на ноль дает ноль: f = 0, s = 0, exp = –128.

5.6 Сложение и вычитание чисел с плавающей запятой

При сложении и вычитании с плавающей запятой два числа с плавающей запятой a и b должны быть определены как:

$$a = a(man) \times 2^{a(exp)}, \quad (5.8)$$

$$b = b(man) \times 2^{b(exp)} \quad (5.9)$$

Сумма или разность a и b должна быть определена как:

$$c = a \pm b: \quad (5.10)$$

$$= (a(man) \pm (b(man) \times 2^{-(a(exp) - b(exp))})) \times 2^{a(exp)},$$

если $a(exp) \geq b(exp)$;

$$= ((a(man) \times 2^{-(b(exp) - a(exp))}) \pm b(man)) \times 2^{b(exp)},$$

если $a(exp) < b(exp)$.

Блок-схема операции сложения с плавающей запятой приведена на рисунке 5.16. Так как эта блок-схема предполагает данные со знаком, то она так же подходит и для вычитания чисел с плавающей запятой. Для этого примера предполагается, что $a(exp) \leq b(exp)$**.** На этапе 1 исходные порядки чисел сравниваются и c(exp) присваивается наибольшее значение исходной экспоненты. На этапе 2 d присваивается значение разности экспонент. На этапе 3 мантисса с наименьшей экспонентой, в этом случае

a(man), сдвигается вправо на d разрядов с целью выравнивания мантисс. После того как мантиссы выравнены, они складываются (этап 4).

Этапы с 5 по 7 – проверка для специальных случаев c(man). Если c(man) равно нулю (этап 5), то c(exp) присваивается наименьшее отрицательное значение (этап 8) для получения корректного представления нуля. Если происходит переполнение c(man) (этап 6), то на этапе 9 c(man) сдвигается вправо на один разряд и к c(exp) добавляется единица. На этапе 10 результат нормализуется. Этапы 11 и 12 для тестирования c(exp) в специальных случаях. Если происходит переполнение c(exp)**,** то c присваивается наибольшее положительное значение с повышенной точностью, если c(exp) положительно, в противном случае c присваивается наименьшее отрицательное значение с повышенной точностью.

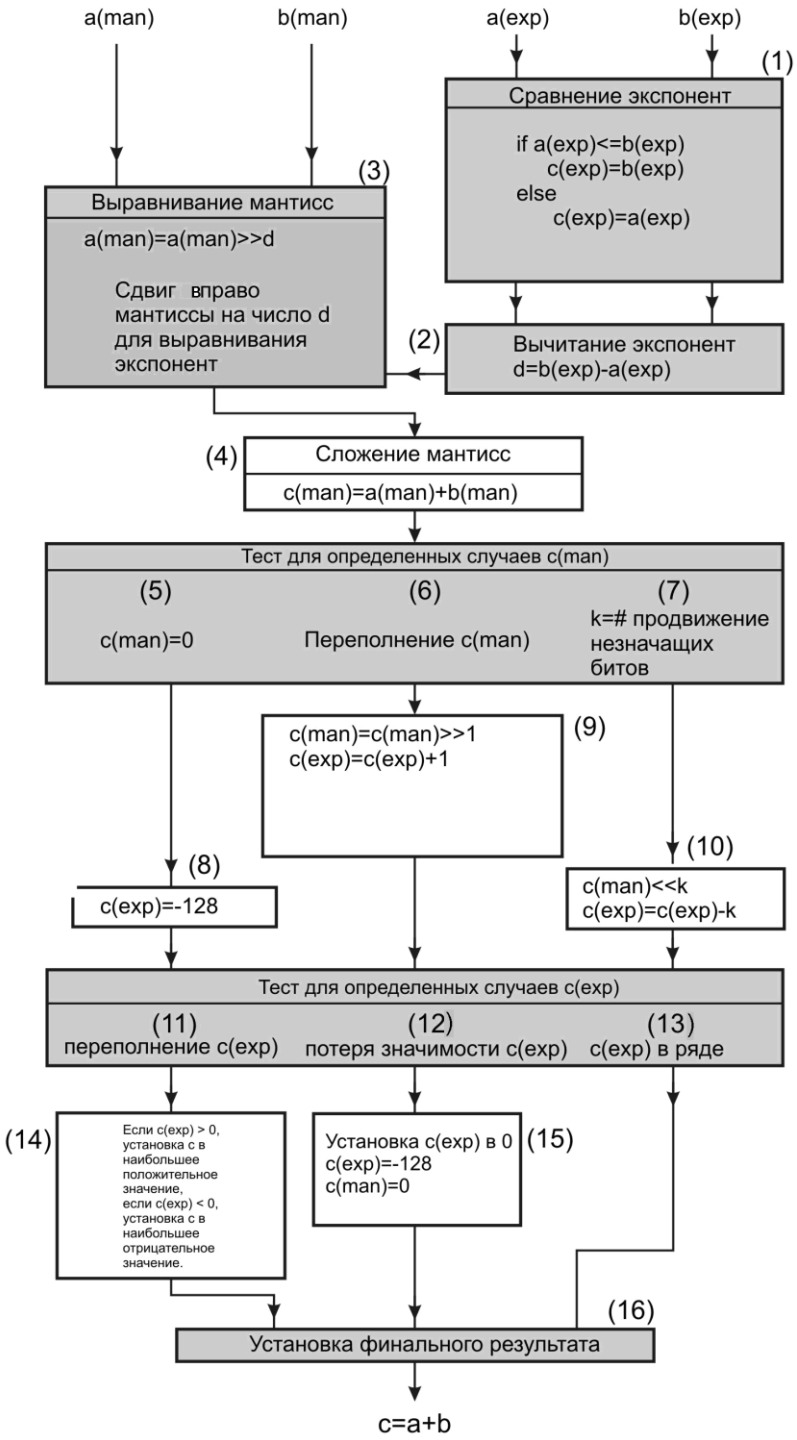


Рисунок 5.16 – Блок-схема для сложения с плавающей запятой

Следующие примеры описывают операции сложения и вычитания с плавающей запятой. Предполагается, что данные находятся в формате с плавающей запятой с повышенной точностью.

Пример 5.10 – Сложение с плавающей запятой

Пусть

$$\alpha = 1.5 = 01.10000000000000000000000000000000 \times 2^{0}$$

$$b = 0.5 = 01.00000000000000000000000000000000 \times 2^{-1}$$

Необходимо сдвинуть b вправо на один разряд так, чтобы a и b имели одинаковые экспоненты. В результате получится:

$$b = 0.5 = 00.10000000000000000000000000000000 \times 2^{0}$$

Тогда:

$$\begin{array}{l} 01.10000000000000000000000000000000 \times 2^{0} \\ +\ 00.10000000000000000000000000000000 \times 2^{0} \\ \hline 010.00000000000000000000000000000000 \times 2^{0} \end{array}$$

Как и в случае умножения, необходимо сдвинуть запятую на один разряд влево и прибавить единицу к экспоненте. В результате получится:

$$\begin{array}{l} 01.10000000000000000000000000000000 \times 2^{0} \\ +\ 00.10000000000000000000000000000000 \times 2^{0} \\ \hline 01.00000000000000000000000000000000 \times 2^{1} \end{array}$$

Пример 5.11 – Вычитание с плавающей запятой

Пусть

$$\alpha = 01.00000000000000000000000000000001 \times 2^{0}$$
$$b = 01.00000000000000000000000000000000 \times 2^{0}$$

Должна быть выполнена операция (a – b). Мантиссы уже выравнены, так как два числа имеют одинаковые экспоненты. В результате происходит большая потеря точности верхних разрядов, как показано ниже:

$$\begin{array}{l} 01.00000000000000000000000000000001 \times 2^{0} \\ -\ 01.00000000000000000000000000000000 \times 2^{0} \\ \hline 00.00000000000000000000000000000001 \times 2^{0} \end{array}$$

Результат должен быть нормализован. В этом случае требуется сдвиг влево на 31, соответственно, изменяется экспонента результата. В результате получится:

$$\begin{array}{l} 01.00000000000000000000000000000001 \times 2^{0} \\ -\ 01.00000000000000000000000000000000 \times 2^{0} \\ \hline 01.00000000000000000000000000000000 \times 2^{-31} \end{array}$$

Пример 5.12 – Сложение с плавающей запятой с 32-битным сдвигом

Этот пример иллюстрирует ситуацию, где полный сдвиг на 32 разряд необходим для нормализации результата. Дано:

$$\alpha = 01.11111111111111111111111111111111 \times 2^{127}$$
$$b = 10.00000000000000000000000000000000 \times 2^{127}$$

Должна быть выполнена операция α + b.

$$\begin{array}{l} 01.11111111111111111111111111111111 \times 2^{127} \\ +\ 10.00000000000000000000000000000000 \times 2^{127} \\ \hline 11.11111111111111111111111111111111 \times 2^{127} \end{array}$$

Для нормализации результата требуется сдвиг влево на 32 разряда и вычитание 32 из экспоненты. В результате получится:

$$
\begin{array}{rl}
 & 01.11111111111111111111111111111111 \times 2^{127} \\
+ & 10.00000000000000000000000000000000 \times 2^{127} \\
\hline
 & 10.00000000000000000000000000000000 \times 2^{95}
\end{array}
$$

Пример 5.13 – Сложение/вычитание с плавающей запятой и нулем

Когда выполняется сложение и вычитание чисел с нулем в формате с плавающей запятой, справедливы следующие соотношения:

$a \pm 0 = a \qquad (a \neq 0)$

$0 \pm 0 = 0$

$0 - a = -a \qquad (a \neq 0)$

5.7 Нормализация числа (инструкция NORM)

Инструкция NORM нормализует число с плавающей запятой с повышенной точностью, если оно не нормализовано.

Примечание – Нормальной формой числа с плавающей запятой называется такая форма, в которой мантисса (без учёта знака) находится на полуинтервале [0; 1).

Инструкция NORM выполняет следующие три этапа:

- 1 этап. Находит наиболее значимый незнаковый разряд числа с плавающей запятой.
- 2 этап. Сдвигает влево число для его нормализации.
- 3 этап. Устанавливает порядок числа.

Пусть число a с плавающей запятой и повышенной точностью не нормализовано, на рисунке 5.17 показана его нормализация.

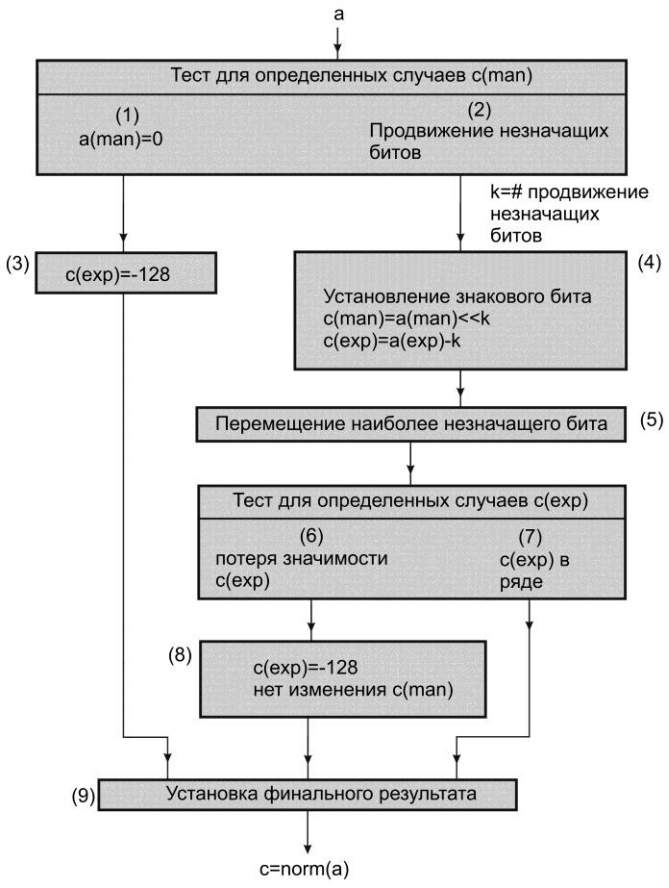


Рисунок 5.17 – Блок-схема выполнения инструкции NORM

Пример 5.14 – Инструкция NORM

Предположим, что регистр повышенной точности содержит значение мантиссы и порядка числа:

man = 00000000000000000001000000000001, exp = 0

При выполнении нормализации числа (предполагается, что число не нормализовано) подразумевается, что число с двоичной запятой имеет вид:

man = 0.00000000000000000001000000000001, exp = 0

Знак этого числа расширяется на один разряд, после этого мантисса содержит 33 разряда.

man = 00.00000000000000000001000000000001, exp = 0

Промежуточный результат после перемещения старшего незнакового разряда и выполнения сдвига определяется следующим образом:

man = 01.00000000000100000000000000000000, exp = –19

После удаления дополнительного бита формируется окончательное 32-разрядное значение нормализованного числа с плавающей запятой:

man = 00000000000100000000000000000000, exp = –19

Инструкция NORM полезна для подсчета количества нулей или единиц находящихся в наиболее значимых разрядах в 32 разрядном слове. Если начальное значение порядка числа равно нулю, то абсолютное окончательное значение результата порядка числа равно количеству единиц или нулей, находящихся в раиболее значимых разрядах числа. Эта инструкция также используется для обработки ненормализованных чисел с плавающей запятой.

5.8 Округление числа (инструкция RND)

Инструкция RND округляет числа из формата с плавающей запятой и повышенной точностью в формат числа с плавающей запятой и одинарной точностью.

Операция округления аналогична операции сложения с плавающей запятой. При округлении данного числа a, сначала выполняется следующая операция:

$$c = a(man) \times 2^{a(exp)} + (1 \times 2^{(a(exp)-24)}) \qquad (5.11)$$

Затем выполняется преобразование из формата с плавающей запятой и повышенной точностью в формат с плавающей запятой и одинарной точностью. Округление данного числа с плавающей запятой и повышенной точностью показано на рисунке 5.18.

Примечание – RND src, dst – где (src) = 0 – не устанавливает флаг нулевого значения (бит 2 в регистре состояния). Вместо этого устанавливается флаг условия потери значимости разрядов числа (бит 4 в регистре состояния).

Примечание – На рисунке 5.18 принято условное обозначение:

МЗР – наименее значимый разряд числа.

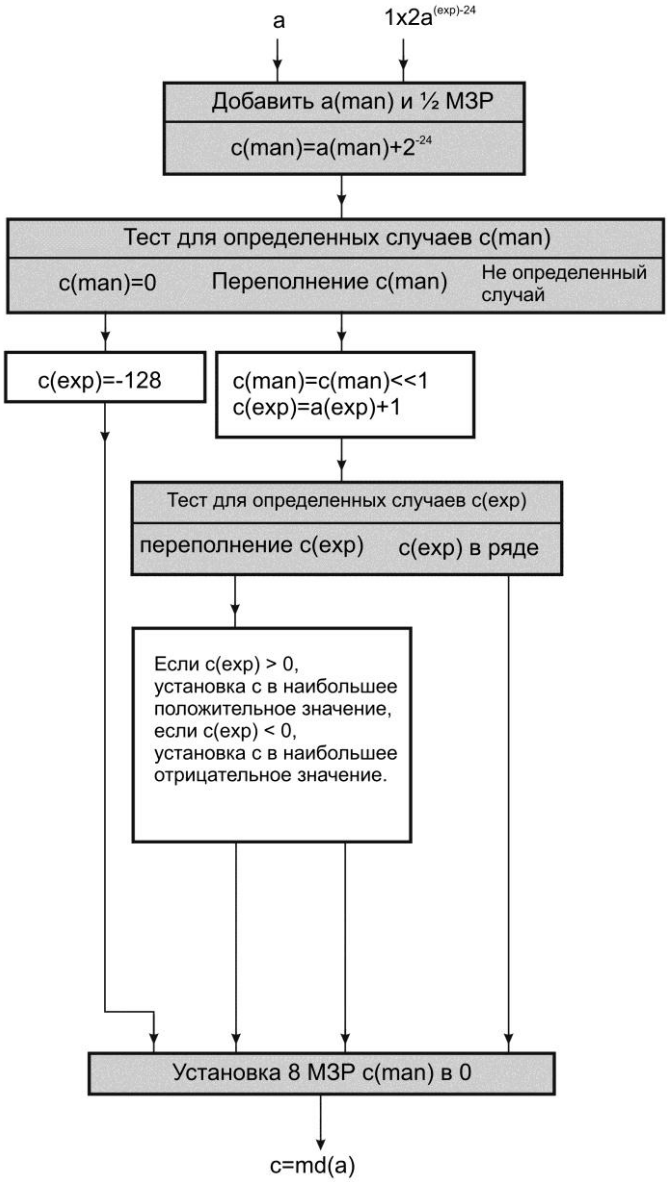


Рисунок 5.18 – Блок-схема выполнения операции округления числа с плавающей запятой с помощью инструкции RND

5.9 Преобразование числа с плавающей запятой в целое число (инструкция FIX)

Для преобразования числа с плавающей запятой в целое, используется инструкция FIX, которая выполняет преобразование чисел в формате с плавающей запятой с одинарной точностью в целые числа с одинарной точностью за один цикл.

Преобразование числа с плавающей запятой x в целое будет обозначаться fix(x). Преобразование не приведет к переполнению, если преобразуемое число a находится в диапазоне: $-2^{31} \le a \le 2^{31}-1$.

Сначала необходимо определить, что a(exp) ≤ 30. Если это не выполняется, то происходит переполнение. Если переполнение произошло в положительном направлении, то на выходе будет наибольшее положительное целое число. Если переполнение произошло в отрицательном направлении, то на выходе будет получено наименьшее отрицательное целое число. Если a(exp) попадает в разрешенный диапазон, то для a(man), включая неявный разряд, производится расширение знака и сдвиг вправо (rs) на величину равную:

$$rs = 31 - a(exp) \qquad (5.12)$$

Этот сдвиг вправо (rs) сдвигает разряды в соответствии с дробной частью мантиссы. Например:

Если $0 \le x < 1$, тогда fix(x) = 0.

Если $-1 \le x < 0$, тогда fix(x) = –1.

Блок-схема для преобразования числа с плавающей запятой в целое число показана на рисунке 5.19.

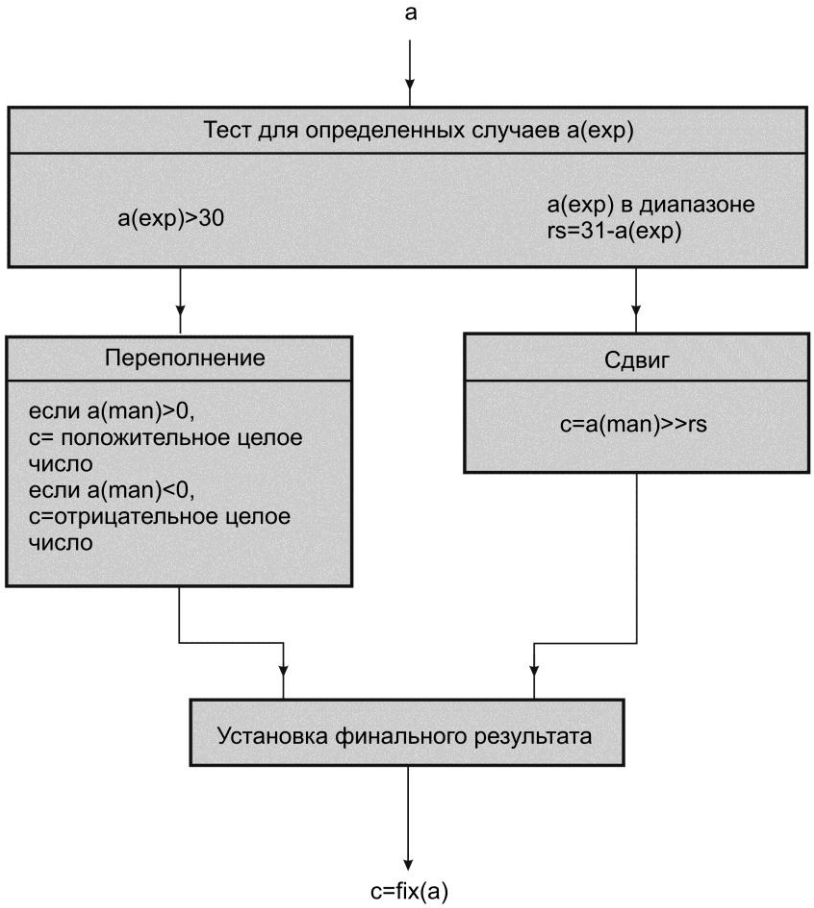


Рисунок 5.19 – Блок-схема преобразования числа с плавающей запятой в целое число с помощью инструкции FIX

5.10 Преобразование целого числа в число с плавающей запятой (инструкция FLOAT)

Инструкция FLOAT осуществляет преобразование целого числа с одинарной точностью в число в формате с плавающей запятой повышенной точности. Блок-схема этого преобразования приведена на рисунке 5.20.

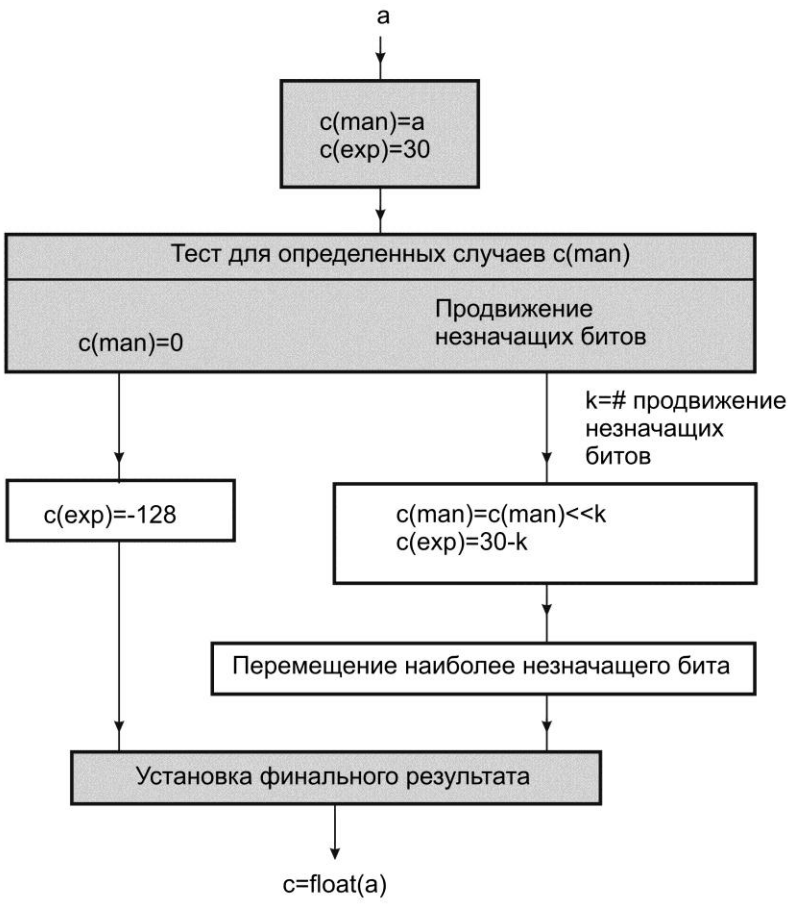


Рисунок 5.20 – Блок-схема преобразования целого числа в число с плавающей запятой при использовании инструкции FLOAT

5.11 Обратное значение числа (инструкция RCPF)

Инструкция RCPF вычисляет приближенное (оценка) значение обратной величины числа с плавающей запятой за один цикл. Значение обратной величины числа имеет правильный порядок числа и мантиссу с точностью до восьми двоичных разрядов (таким образом, ошибка мантиссы $< 2^{-8}$). Оценка обратной величины числа имеет 16-разрядное представление (8 разрядов порядка числа и 8 разрядов мантиссы). Также эта оценка может быть использована как начальное число для алгоритма подсчета более точного значения обратной величины (алгоритм Ньютона-Рафсона, описываемый в этой части).

Рисунок 5.21 показывает алгоритм, используемый инструкцией RCPF.

Предположим, что на входе число $v = vman \times 2^{vexp}$.

Предположим, что на выходе есть $x = xman \times 2^{xexp}$, vexp отрицательна.

Если vexp = –128, то результат будет расширен до наибольшего положительного числа, при этом устанавливается флаг переполнения. Флаг условия N устанавливается в зависимости от vsign.

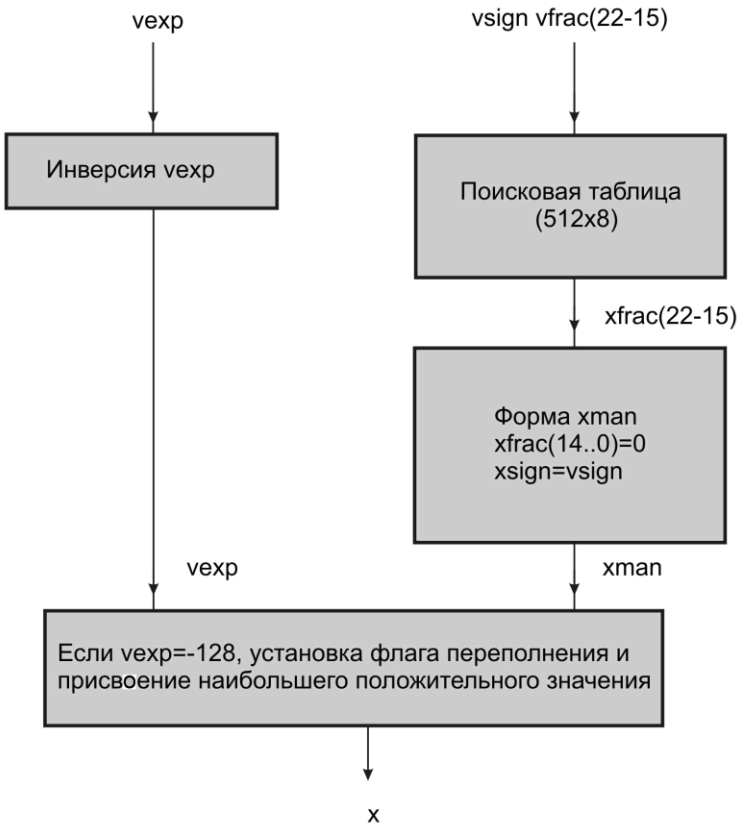


Рисунок 5.21 – Алгоритм выполнения инструкции RCPF

Таблица значений читается посредством формирования 9-разрядного адреса, включающего в себя vsign и разряды 22-15 vfrac. 8-разрядный выход таблицы значений формирует разряды 22-15 xfrac. Разряды 14-0 xfrac сбрасываются в ноль. xsign устанавливается в vsign. Значения таблицы генерируются в зависимости от входных чисел.

5.11.1 Алгоритм получения обратной величины числа

Инструкция RCPF формирует обратное значение числа. Приближенная оценка дает правильное значение показателя степени, а значение мантиссы с точностью до восьмого двоичного разряда (т. е. погрешность мантиссы составляет $< 2^{-8}$). Алгоритм Ньютона-Рафсона (показан в примере 5.15) может использоваться для дальнейшего повышения точности мантиссы:

$$x[n+1] = x[n](2-vx[n]), \qquad (5.13)$$

где v = число, обратное значение которого ищется;

x[0] – начальное значение для алгоритма, задаваемое инструкцией RCPF. Для каждой итерации алгоритма количество точных разрядов в мантиссе удваивается. Используя инструкцию RCPF, можно начать с приближенной точности до 8 разрядов. После одной итерации точность мантиссы достигает 16 разрядов, а после второй – до 32 разрядов.

В примере 5.15 приведена программа для ИС 1867ВМ9Ф, реализующая этот алгоритм. Каждый шаг алгоритма в примере помечен соответствующей точностью, достигнутой в конце шага. Алгоритм занимает всего семь машинных циклов.

Пример 5.15 – Алгоритм Ньютона-Рафсона для вычисления эквивалента

```
RCPF R0, R1    ; R0 = v, R1 = x[0]
;
MPYF R1, R0, R2
SUBRF 2.0, R2
MPYF R2, R1    ; конец первой итерации (16-разрядная точность)
;
MPYF R1, R0, R2
SUBRF 2.0, R2
MPYF R2, R1    ; конец второй итерации (32-разрядная точность)
;
; ; R1 = 1/v
;
```

5.12 Обратная величина квадратного корня числа (инструкция RSQRF)

Во многих приложениях необходима нормализация значений данных. Часто нормализующим множителем является квадратный корень другой величины. Например, когда задан один вектор, и нужно найти единичный вектор в этом же направлении, то это делается путем деления начального вектора на его длину. При этом производится деление на квадратный корень. Инструкция RSQRF дает простую возможность непосредственного определения этой величины вместо выполнения двухступенчатого приближенного отыскания квадратного корня и последующего нахождения обратной величины квадратного корня.

В результате выполнения этого алгоритма квадратный корень находится простым умножением:

$$v=vx[n], \qquad (5.14)$$

где x[n] – это оценка $\frac{1}{\sqrt{v}}$ как решения алгоритма Ньютона-Рафсона или любого другого алгоритма.

Инструкция RSQRF генерирует удовлетворительную приблизительную оценку обратной величины квадратного корня числа с плавающей запятой за один цикл.

Характеристики инструкции RSQRF:

- Инструкция RSQRF генерирует приблизительное значение (в этом случае, значение обратной величины квадратного корня числа с плавающей запятой).

- Точность мантиссы это восьмь двоичных разрядов (ошибка мантиссы $< 2^{-8}$).

Часто, полученное значение является достаточной оценкой обратной величины квадратного корня числа; в остальных случаях, оно может быть использовано как начальное значение для алгоритма вычисления обратной величины квадратного корня с большей точностью.

Рисунок 5.22 иллюстрирует алгоритм выполнения инструкции RSQRF. В алгоритме используется:

- исходное значение $v = vman \times 2^{vexp}$;

- конечное значение $x = xman \times 2^{xexp}$,

где vexp+1 – отрицательное число, которое сдвинуто вправо на один разряд с дополнением знака.

Если vexp = –128, то результат расширяется до наибольшего положительного числа, и устанавливается флаг переполнения.

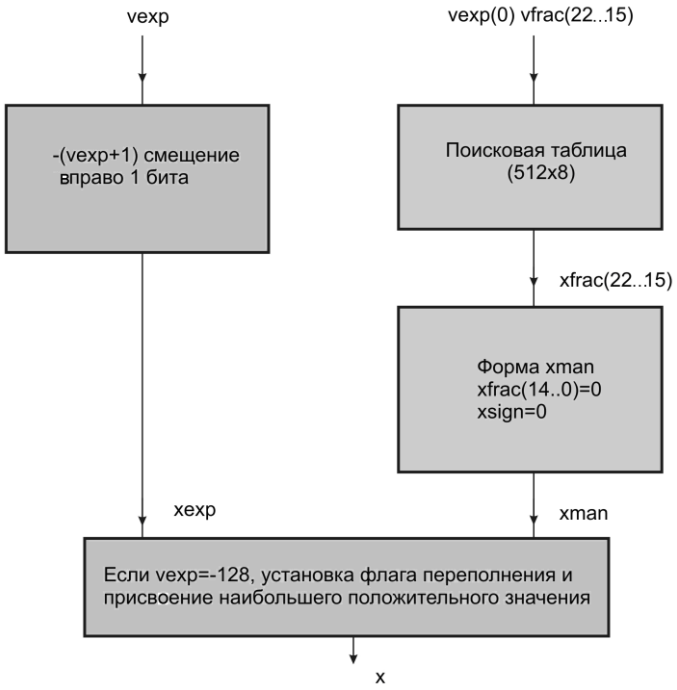


Рисунок 5.22 – Алгоритм выполнения инструкции RSQRF

Таблица значений читается с помощью формирования 9-разрядного адреса, включающего наименьший значащий разряд vexp и разряды 22-15 vfrac. 8-разрядный выход таблицы значений формирует разряды 22-15 xfrac. Разряды 14-0 xfrac сбрасываются в «0». Значение xsign устанавливается в «0».

Примечание – Не обеспечивается для отрицательных значений v.

Значения таблицы генерируются в зависимости от входных чисел.

В данном алгоритме деление заменено выполнением простого умножения:

$$y/v = yx[n] \qquad (5.15)$$

В этом уравнении x[n] – это оценка обратной величины 1/v, вычисляется как решение алгоритма Ньютона-Рафсона или другого алгоритма.

5.12.1 Алгоритм Ньютона-Рафсона

Инструкция RSQRF позволяет получить приближенное значение (оценку) обратной величины квадратного корня числа. Оценка имеет правильный порядок и мантиссу, с точностью в 8 двоичных разрядов (т. е. ошибка мантиссы $< 2^{-8}$).

Алгоритм Ньютона-Рафсона, показанный в примере 5.16, может быть использован для уточнения значения мантиссы

$$x[n+1] = x[n](1{,}5–(v/2)x[n]x[n]), \qquad (5.16)$$

где v = число, обратная величина которого вычисляется.

Начальное значение для алгоритма x[0] задается RSQRF. Для каждой итерации алгоритма количество точных разрядов в мантиссе удваивается. Используя инструкцию RSQRF, можно начать с точности 8 разрядов. После одной итерации точность мантиссы достигает 16 разрядов, а после второй – до 32 разрядов.

Программа ИС 1867ВМ9Ф, реализующая этот алгоритм, приведена в примере 5.16. Каждый шаг алгоритма помечен соответствующей точностью, достигнутой в конце шага, реализация алгоритма занимает всего десять машинных циклов.

Пример 5.16 – Алгоритм Ньютона-Рафсона для вычисления обратной величины квадратного корня числа

```
RSQRF R0, R1        ; R0 = v, R1 = x[0]
MPYF 0.5, R0 ; R0 = v/2
;
MPYF R1, R1, R2
MPYF R0, R2
SUBRF 1.5, R2
MPYF R2, R1 ; конец первой итерации (16-разрядная точность)
;
MPYF R1, R1, R2
MPYF R0, R2
SUBRF 1.5, R2
MPYF R2, R1 ; конец второй итерации (32-разрядная точность)
;
; R1 = 1/(v**0.5)
;
```

6 Методы адресации

Этот раздел описывает методы адресации, поддерживаемые ИС 1867ВМ9Ф, и их использование. Процессор ИС поддерживает пять методов адресации, которые обеспечивают доступ к данным в памяти, регистрах и словах инструкций:

- регистровая;
- прямая;
- косвенная;
- непосредственная (короткая и длинная);
- относительная (относительно PC).

Не все методы адресации подходят для использования во всех инструкциях.

6.1 Типы адресации

Пять типов адресации дают возможность получать доступ к данным в памяти, регистрах и словах инструкций:

- регистровая;
- прямая;
- косвенная;
- непосредственная (короткая и длинная);
- относительная (относительно PC).

Не все типы адресации могут быть использованы во всех инструкциях, поэтому типы адресации разбиты на четыре различные группы режимов адресации:

Группа основных режимов адресации G использует методы:

- регистровой адресации,
- прямой адресации,
- косвенной адресации,
- короткой непосредственной адресации.

Группа трехоперандных режимов адресации T использует методы:

- регистровой адресации,
- косвенной адресации.

Группа режимов параллельной адресации P использует методы:
- регистровой адресации,
- косвенной адресации.

Группа режимов адресации условных переходов B использует методы:
- регистровой адресации,
- относительной адресации.

Для использования в фильтрах и БПФ имеется два специальных метода:
- циклическая адресация;
- бит-реверсивная адресация.

В первую очередь будут рассмотрены пять типов адресации, а затем пять групп режимов адресации.

6.2 Регистровая адресация

В регистровой адресации операнд содержится в регистре ЦПУ, как показано в примере: ABSF R1; R1 = | R1 |

Синтаксис для регистров ЦПУ в языке Ассемблер и назначение этих регистров приведены в таблице 6.1.

Таблица 6.1 – Регистры ЦПУ, их назначение и синтаксис Ассемблера

Адрес регистра	Синтаксис Ассемблера	Назначение
00h	R0	Регистр повышенной точности 0
01h	R1	Регистр повышенной точности 1
02h	R2	Регистр повышенной точности 2
03h	R3	Регистр повышенной точности 3
04h	R4	Регистр повышенной точности 4
05h	R5	Регистр повышенной точности 5
06h	R6	Регистр повышенной точности 6
07h	R7	Регистр повышенной точности 7
1Ch	R8	Регистр повышенной точности 8
1Dh	R9	Регистр повышенной точности 9
1Eh	R10	Регистр повышенной точности 10
1Fh	R11	Регистр повышенной точности 11
08h	AR0	Вспомогательный регистр 0
09h	AR1	Вспомогательный регистр 1
0Ah	AR2	Вспомогательный регистр 2
0Bh	AR3	Вспомогательный регистр 3
0Ch	AR4	Вспомогательный регистр 4
0Dh	AR5	Вспомогательный регистр 5
0Eh	AR6	Вспомогательный регистр 6
0FH	AR7	Вспомогательный регистр 7
10h	DP	Указатель страницы данных
11h	IR0	Индексный регистр 0
12h	IR1	Индексный регистр 1
13h	BK	Регистр размера блока
14h	SP	Указатель системного стека
15h	ST	Регистр состояния
16h	IE	Разрешение прерываний ЦПУ/ПДП
17h	IF	Флаги прерываний ЦПУ
18h	IOF	Флаги ввода-вывода
19h	RS	Регистр адреса начала блока повторения
1Ah	RE	Регистр адреса конца блока повторения
1Bh	RC	Счетчик повторений
–	IVTP	Указатель таблицы векторов системных прерываний
–	TVTP	Указатель таблицы векторов программных прерываний

6.3 Прямая адресация

При прямой адресации адрес данных формируется путем объединения шестнадцати младших разрядов указателя страницы данных DP с 16 младшими разрядами слова инструкции (expr). В результате программист имеет возможность обращаться к большому адресному пространству 65536 страниц (64 К слов в каждой странице без необходимости изменения значения DP).

Синтаксис метода адресации и операция при прямой адресации приведены ниже.

Синтаксис: @expr

Операция: адрес = DP объединен с (expr)

На рисунке 6.1 приведено формирование адреса данных. В примере 6.1 приводится пример инструкции с данными до и после выполнения инструкции.

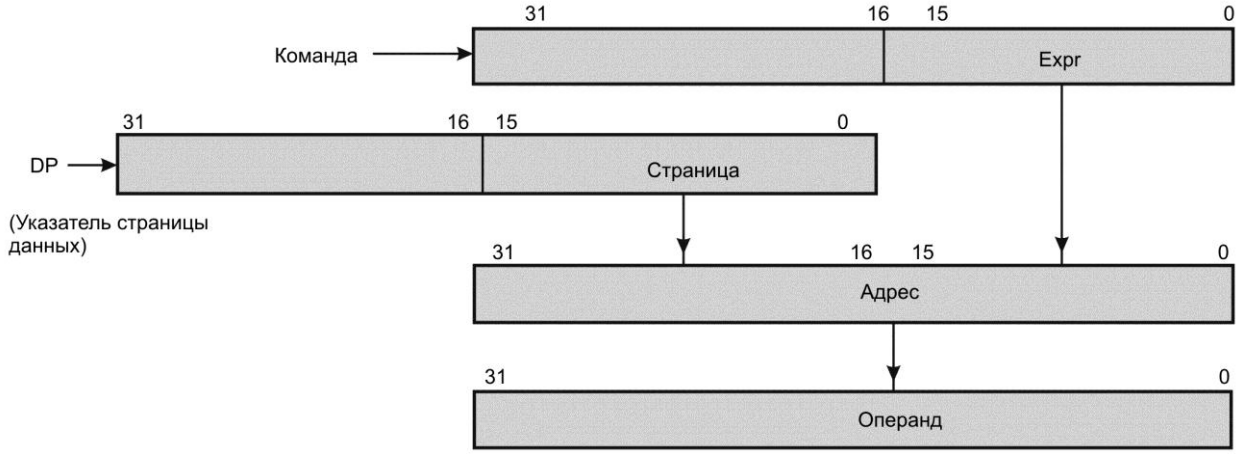


Рисунок 6.1 – Прямая адресация

Пример 6.1 – Прямая адресация

ADDI @0BCDEh, R7

До:	После:
DP=108Ah	DP=108Ah
R7=11h	R7=1234 5689h
Данные 108A BCDEh=1234 5678h	Данные 108A BCDEh=1234 5678h

6.4 Косвенная адресация

Косвенная адресация используется для определения адреса операнда в памяти, с использованием вспомогательного регистра, а также дополнительно, если необходимо, с использованием смещения и индексных регистров. Арифметическое устройство вспомогательных регистров ARAU выполняет беззнаковую арифметику над содержимым регистров для получения адреса операнда. Все 32 бита вспомогательных и индексных регистров используются в косвенной адресации.

Гибкость косвенной адресации обеспечивается тем, что ARAU модифицирует вспомогательные регистры параллельно с операциями ЦПУ. Косвенная адресация определяется пятиразрядным полем в слове инструкции, представленным как поле (mod) (показано с левой стороны рисунка 6.2).

Смещение – это либо восьмиразрядное целое без знака, содержащееся в слове инструкции, либо как неявное смещение на единицу. Два индексных регистра IR0 и IR1 также могут быть использованы в косвенной адресации, разрешая применять 32-битное косвенное смещение. В некоторых случаях дополнительно можно использовать циклическую или бит-реверсную адресацию.

Механизм формирования адресов в циклической адресации рассмотрен в 6.8, бит-реверсной – в подразделе 6.9.

В таблице 6.2 представлены разные виды косвенной адресации с соответствующим каждому виду полем модификации (mod), синтаксисом Ассемблера, операцией и выполняемой функцией. Рисунок 6.2 показывает формат косвенной адресации операнда в коде инструкции. Поле (disp) не существует для некоторых инструкций.

Рисунок 6.2 – Формат косвенной адресации

Примечание – Вспомогательный регистр ARn может быть закодирован в слове инструкции соответственно его двоичному значению n (то есть AR3 кодируется как 11_2, что не соответствует его внутреннему адресу).

Таблица 6.2 – Косвенная адресация

Поле модификации	Синтаксис	Операция	Описание
Косвенная адресация со смещением			
00000	*+ARn(disp)	addr=ARn+disp	С добавлением предварительного смещения
00001	*-ARn(disp)	addr=ARn-disp	С предварительным вычитанием смещения
00010	*++ARn(disp)	addr=ARn+disp ARn=ARn+disp	С предварительным добавлением смещения и модификацией
00011	*--ARn(disp)	addr=ARn-disp ARn=ARn-disp	С предварительным вычитанием смещения и модификацией
00100	*ARn++(disp)	addr=ARn ARn=ARn+disp	С последующим добавлением смещения и модификацией
00101	*ARn - (disp)	addr=ARn ARn=ARn-disp	С последующим вычитанием смещения и модификацией
00110	*ARn++(disp)%	addr=ARn ARn=circ(ARn+disp)	С последующим добавлением смещения и циклической модификацией
00111	*ARn--(disp)%	addr=ARn ARn=circ(ARn-disp)	С последующим вычитанием смещения и циклической модификацией
Косвенная адресация с индексным регистром IR0			
01000	*+ARn(IR0)	addr=ARn+IR0	С предварительным добавлением значения индексного регистра IR0
01001	*-ARn(IR0)	addr=ARn-IR0	С предварительным вычитанием значения индексного регистра IR0
01010	*++ARn(IR0)	addr=ARn+IR0 ARn=ARn+IR0	С предварительным добавлением значения индексного регистра IR0 и модификацией
01011	*--ARn(IR0)	addr=ARn-IR0 ARn=ARn-IR0	С предварительным вычитанием значения индексного регистра IR0 и модификацией
01100	*ARn++(IR0)	addr=ARn ARn=ARn+IR0	С последующим добавлением индексного регистра IR0 и изменение
01101	*ARn-(IR0)	addr=ARn ARn=ARn-IR0	С последующим вычитанием индексного регистра IR0 и изменение

Окончание таблицы 6.2

Поле модификации	Синтаксис	Операция	Описание
01110	*ARn++(IR0)%	addr=ARn ARn=circ(ARn+IR0)	С последующим добавлением значения индексного регистра IR0 и циклическая модификация
01111	*ARn--(IR0)%	addr=ARn ARn=circ(ARn-IR0)	С последующим вычитанием значения индексного регистра IR0 и циркулярная модификация
Косвенная адресация с индексным регистром IR1			
10000	*+ARn(IR1)	addr=ARn+IR1	С предварительным добавлением значения индексного регистра IR1
10001	*-ARn(IR1)	addr=ARn-IR1	С предварительным вычитанием значения индексного регистра IR1
10010	*++ARn(IR1)	addr=ARn+IR1 ARn=ARn+IR1	С предварительным добавлением значения индексного регистра IR1 и модификацией
10011	*--ARn(IR1)	addr=ARn-IR1 ARn=ARn-IR1	С предварительным вычитанием значения индексного регистра IR1 и модификацией
10100	*ARn++(IR1)	addr=ARn ARn=ARn+IR1	С последующим добавлением значения индексного регистра IR1 и модификацией
10101	*ARn--(IR1)	addr=ARn ARn=ARn-IR1	С последующим вычитанием значения индексного регистра IR1 и модификацией
10110	*ARn++(IR1)%	addr=ARn ARn=circ(ARn+IR1)	С последующим добавлением значения индексного регистра IR1 и циклической модификацией
10111	*ARn--(IR1)%	addr=ARn ARn=circ(ARn-IR1)	С последующим вычитанием значения регистра IR1 и циклической модификацией
Косвенная адресация (специальные случаи)			
11000	*ARn	addr=ARn	Косвенная
11001	*ARn++(IR0)B	addr=ARn ARn=B(ARn+IR0)	Сложение и бит-реверсная модификация с последующим индексированием IR0

Примечание – Принятые условные обозначения:
- addr – адрес памяти;
- ARn – вспомогательные регистры AR0-AR7;
- IRn – индексный регистр IR0 или IR1;
- (IRn) – содержимое регистра IRn;
- disp – смещение;
- ++ – сложение и модификация;
- -- – вычитание и модификация;
- circ() – адрес в циклической адресации;
- % – циклическая адресация;
- B – бит-реверсная адресация.

Примеры с 6.2 по 6.19 показывают операцию для каждого типа косвенной адресации.

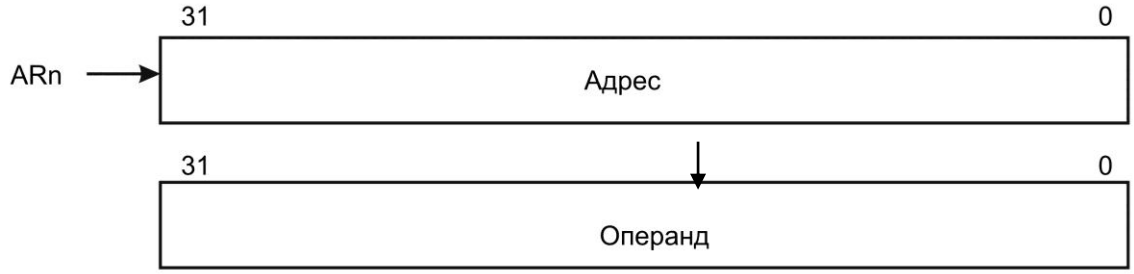
Пример 6.2 – Косвенная адресация через вспомогательный регистр

Вспомогательный регистр ARn содержит адрес операнда.

Операция: адрес операнда = ARn

Синтаксис Ассемблера: *ARn

Поле модификации: 11000



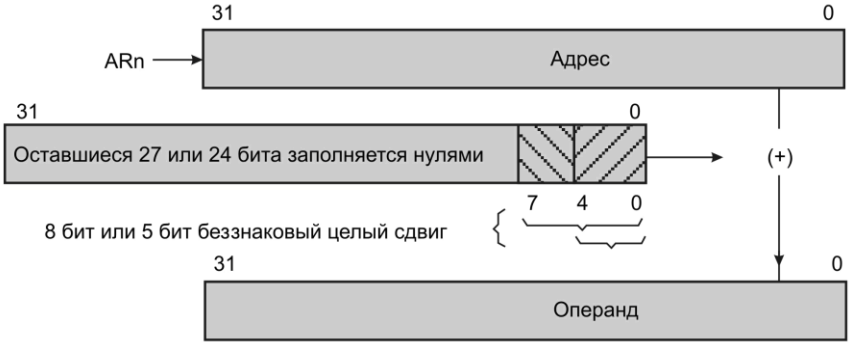
Пример 6.3 – Косвенная адресация с предварительным добавлением смещения

Адрес операнда суммируется со вспомогательным регистром ARn и смещением disp. Смещение – это пять или восемь бит беззнакового целого, содержащегося в слове инструкции или неявное значение «1».

Операция: адрес операнда = ARn+disp

Синтаксис Ассемблера: *ARn(disp)

Поле модификации: 00000



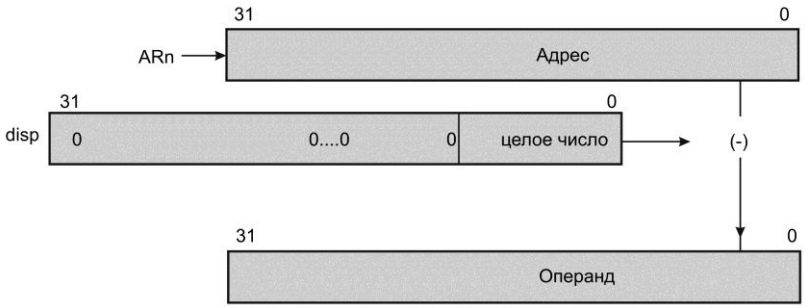
Пример 6.4 – Косвенная адресация с предварительным вычитанием смещения

Адрес выбранного операнда – это содержимое вспомогательного регистра ARn минус смещение disp. Смещение – это восемь бит беззнакового целого, содержащегося в слове инструкции или неявное значение «1».

Операция: адрес операнда = ARn-disp

Синтаксис Ассемблера: *ARn(disp)

Поле модификации: 00001



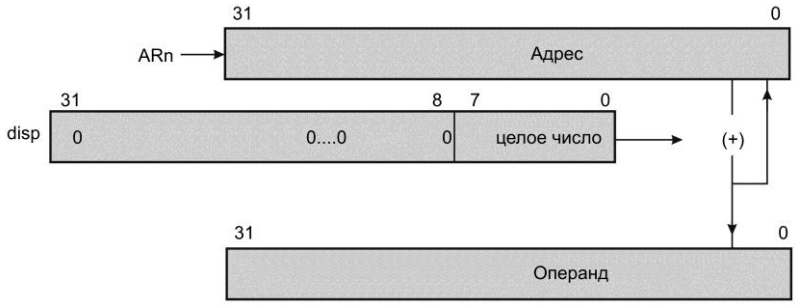
Пример 6.5 – Косвенная адресация с предварительным добавлением смещения и модификацией

Адрес операнда – это сумма содержимого вспомогательного регистра ARn и смещения disp. Смещение – это значение восьми бит беззнакового целого, содержащегося в слове инструкции, или неявное значение «1». После определения адреса операнда данные выбираются, а вспомогательный регистр модифицируется сгенерированным адресом.

Операция: адрес операнда = ARn+disp
ARn = ARn+disp

Синтаксис Ассемблера: *++ARn(disp)

Поле модификации: 00010



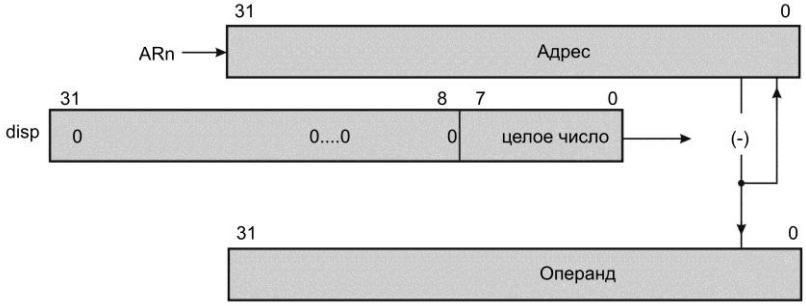
Пример 6.6 – Косвенная адресация с предварительным вычитанием смещения и модификацией

Адрес операнда – это содержимое вспомогательного регистра ARn минус смещение disp. Смещение – это восемь бит беззнакового целого, содержащегося в инструкционном слове или неявное значение «1». После определения адреса операнда и выборки данных вспомогательный регистр обновляется сгенерированным адресом.

Операция: адрес операнда = ARn-disp
ARn = ARn-disp

Синтаксис Ассемблера: *--ARn(disp)

Поле модификации: 00011



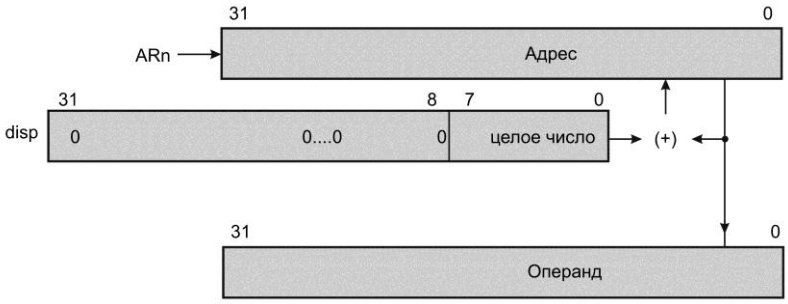
Пример 6.7 – Косвенная адресация с последующим добавлением смещения и модификацией

Адрес выбранного операнда – это содержимое вспомогательного регистра ARn. После выборки операнда добавляется смещение disp к вспомогательному регистру. Смещение – это восемь бит целого без знака, содержащегося в слове инструкции или неявное значение «1».

Операция: адрес операнда = ARn
ARn = Arn+disp

Синтаксис Ассемблера: *Arn++disp

Поле модификации: 00100



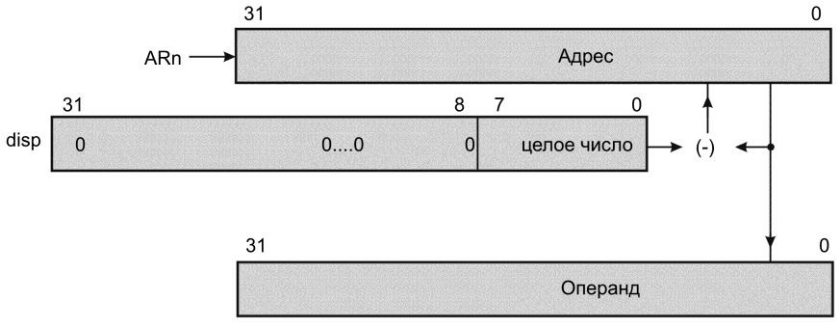
Пример 6.8 – Косвенная адресация с последующим смещением и модификацией

Адрес выбранного операнда – это содержимое вспомогательного регистра ARn. После выборки операнда вычитается смещение disp из вспомогательного регистра. Смещение – это восемь бит целого без знака, содержащегося в слове инструкции или неявное значение «1».

Операция: адрес операнда = ARn
ARn = Arn-disp

Синтаксис Ассемблера: *Arn--disp

Поле модификации: 00101



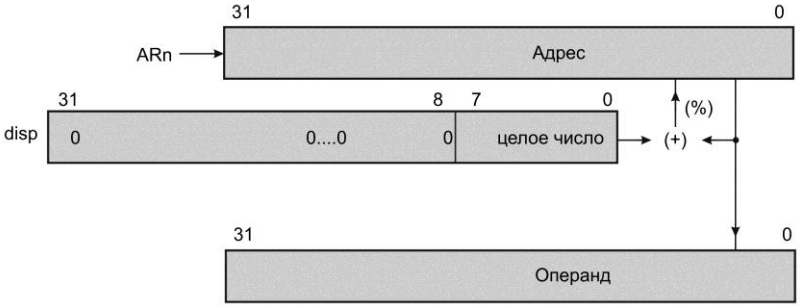
Пример 6.9 – Косвенная адресация с последующим добавлением смещения и циклической модификацией

Адрес выбранного операнда – это содержимое вспомогательного регистра ARn. После выборки операнда добавляется смещение disp к содержимому вспомогательного регистра с циклической адресацией. Смещение – это восемь бит целого без знака, содержащегося в слове инструкции или неявное значение «1».

Операция: адрес операнда = ARn
ARn = circ(Arn+disp)

Синтаксис Ассемблера: * ARn++(disp)%

Поле модификации: 00110

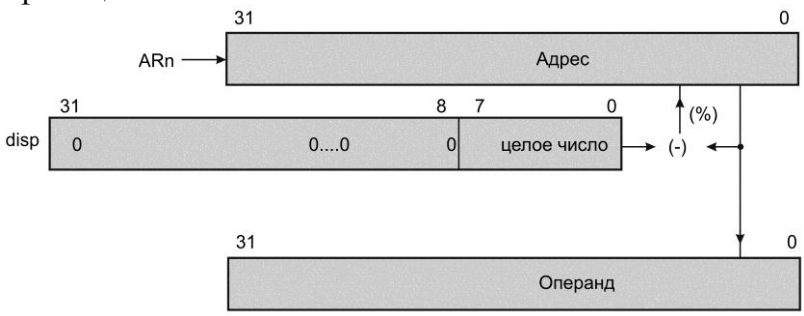


Пример 6.10 – Косвенная адресация с последующим смещением и циклической модификацией

Адрес операнда – это содержимое вспомогательного регистра ARn. После выборки операнда, вычитается смещение disp из содержимого вспомогательного регистра с циклической модификацией. Этот результат используется для модифика-

ции вспомогательного регистра ARn. Смещение – это восемь бит целого без знака, содержащегося в слове инструкции или неявное значение «1».

Операция: адрес операнда = ARn
ARn = circ(Arn-disp)
Синтаксис Ассемблера: *Arn--(disp)%
Поле модификации: 00111



Пример 6.11 – Косвенная адресация с предварительным добавлением индексного регистра

Адрес операнда – это сумма вспомогательного регистра (ARn) и индексного регистра IR0, IR1.

Операция: адрес операнда = ARn+IRm
Синтаксис Ассемблера: *+ARn (IRm)
Поле модификации: 01000, если m=0
10000, если m=1

Пример 6.12 – Косвенная адресация с предварительным вычитанием индексного регистра

Адрес операнда – это разница между содержимым вспомогательного регистра ARn и индексного регистра IR0, IR1.

Операция: адрес операнда = ARn-IRm
Синтаксис Ассемблера: *-ARn(IRm)
Поле модификации: 01001, если m=0
10001, если m=1

Пример 6.13 – Косвенная адресация с предварительным добавлением индексного регистра и модификацией

Адрес операнда – это сумма содержимого вспомогательного регистра ARn и индексного регистра IR0 или IR1. После выборки данных вспомогательный регистр модифицируется вычисленным адресом.

Операция: адрес операнда = ARn+IRm
ARn=ARn+IRm
Синтаксис Ассемблера: *++ARn(IRm)
Поле модификации: 01010, если m=0
10010, если m=1

Пример 6.14 – Косвенная адресация с предварительным вычитанием регистра индекса и модификацией

Адрес операнда – это разница между содержимым вспомогательного регистра ARn и индексного регистра IR0 или IR1. Вычисленный результирующий адрес становится новым содержимым вспомогательного регистра.

Операция: адрес операнда = ARn-IRm
ARn=ARn-IRm
Синтаксис Ассемблера: *--ARn(IRm)
Поле модификации: 01010, если m=0
10010, если m=1

Пример 6.15 – Косвенная адресация с последующим добавлением регистра индекса и модификацией вспомогательного регистра

Адрес операнда – это разница между содержимым вспомогательного регистра ARn и индексного регистра IR0 или IR1. Вычисленный результирующий адрес становится новым содержимым вспомогательного регистра.

Операция: адрес операнда = ARn
ARn=ARn+IRm
Синтаксис Ассемблера: *ARn++(IRm)
Поле модификации: 01100, если m=0
10100, если m=1

Пример 6.16 – Косвенная адресация с последующим вычитанием индекса и модификацией

Адрес операнда – это содержимое вспомогательного регистра ARn. После выбора операнда, индексный регистр IR0 или IR1 вычитается из вспомогательного регистра, который затем модифицируется.

Операция: адрес операнда = ARn
ARn=ARn-IRm

Синтаксис Ассемблера: *ARn--(IRm)

Поле модификации: 01101, если m=0
10101, если m=1

Пример 6.17 – Косвенная адресация с последующим добавлением и циклической модификацией

Адрес операнда – это содержимое вспомогательного регистра (ARn). После выбора операнда, индексный регистр IR0 или IR1 добавляется к вспомогательному регистру. Это значение циклически преобразуется и записывается во вспомогательный регистр.

Операция: адрес операнда = ARn
ARn=circ(ARn+IRm)

Синтаксис Ассемблера: *ARn++(IRm)%

Поле модификации: 01110, если m=0
10110, если m=1

Пример 6.18 – Косвенная адресация с последующим вычитанием и циклической модификацией

Адрес операнда – это содержимое вспомогательного регистра ARn. После выборки операнда индексный регистр IR0 или IR1 вычитается из вспомогательного регистра. Результат циклически преобразуется и записывается во вспомогательный регистр.

Операция: адрес операнда = ARn
ARn=circ(ARn-IRm)

Синтаксис Ассемблера: *ARn--(IRm)%

Поле модификации: 01111, если m=0
10111, если m=1

Пример 6.19 – Косвенная адресация с последующим добавлением и бит-реверсивной модификацией

Адрес операнда – это содержимое вспомогательного регистра ARn. После выборки операнда индексный регистр IR0 добавляется к вспомогательному регистру. Суммирование выполняется с распространением обратного переноса, полученное значение используется для бит-реверсивной B адресации. Это значение заменяет содержимое вспомогательного регистра.

Операция: адрес операнда = ARn

ARn=B(ARn+IR0)

Синтаксис Ассемблера: *ARn++(IR0)B

Поле модификации: 1101

6.5 Непосредственная адресация

При непосредственной адресации операндом является 8- или 16-битное непосредственное значение, хранящееся в восьми или шестнадцати наименьших значащих битах слова инструкции. Допустимые типы данных для инструкции с непосредственным операндом могут быть целый с дополнением до двух, целый без знака, целый со знаком или тип данных с плавающей запятой. Синтаксис для этого метода адресации следующий:

Синтаксис: expr

Пример 6.20 дает пример выполнения инструкции с данными до выполнения и после выполнения. Заметим, что AND и AND3 дают различные результаты.

Пример 6.20 – Непосредственная адресация

Инструкция		Перед выполнением	После выполнения
SUBI	1, R0	R0=0h	R0=00 FFFF FFFFh
LDI	0FFFFh, R0	R0=0h	R0=00 FFFF FFFFh
LDF	5.0, R0	R0=0h	R0=02 2000 0000h
OR	0FFFFh, R0	R0=0h	R0=00 0000 FFFFh
AND3	80h, R0, R0	R0=00 FFFF FFFFh	R0=00 FFFF FF80h
AND	80h, R0	R0=00 FFFF FFFFh	R0=00 0000 0080h

6.6 Относительная адресация (относительно счетчика инструкций PC)

Относительная адресация используется для переходов. При этом методе адресации содержимое 16 или 24 младших битов слова инструкции добавляется к содержимому счетчика команд (регистр PC). Ассемблер берет src (метка или адрес), определенный пользователем и генерирует смещение. Если переход – это стандартный переход, то смещение равно [метка – (адрес инструкции + 1)].

Если переход – это отложенный переход, то смещение равно [метка – (адрес инструкции + 3)].

Смещение хранится как 16-битное или 24-битное целое в младших битах слова инструкции. Смещение добавляется к PC во время фазы декодирования команды. Поскольку PC инкрементируется в фазе выборки, то смещение добавляется к уже инкрементированному значению PC.

Синтаксис: expr (метка или адрес)

Пример 6.21 показывает выполнение инструкции с данными до и после ее выполнения.

Пример 6.21 – PC-относительная адресация

```
BU NEWPC        ; адрес инструкции BU равен 1,
          …     ; адрес метки NEWPC равен 5,
                ; смещение равно 3
NEWPC     …     ; смещение = 5 - (1 + 1)
```

До выполнения	После выполнения
Фаза декодирования	Фаза выполнения
PC = 2h	PC = 5h

24-битный режим адресации используется для кодирования инструкций управления программой (например, BR, BRD, CALL, RPTB, RPTBD, LAJ). В зависимости от инструкции новое значение PC получается добавлением 24-битного значения из слова инструкции к текущему значению PC. Бит 24 инструкции определяет тип ветвления (D = 0 – стандартный переход, D = 1 – отложенный переход). Некоторые коды инструкций представлены на рисунке 6.3.

(a) BR, BRD: безусловные ветвления (не задержанное и задержанное)

31 25		23 0
0 1 1 0 0 0 0	D	Src

(b) CALL: вызов подпрограммы

31	23 0
0 1 1 0 0 0 1 0	Src

(c) RPTB, RPTBD: повторение блока (не задержанное и задержанное)

31		23 0
0 1 1 1 1 0 0	D	Src

(d) LAJ: задержанный переход (возвращает адрес в регистр повышенной точности R11)

31	23 0
0 1 1 0 0 0 1 1	Src

Рисунок 6.3 – Коды инструкций при относительной адресации

6.7 Режимы адресации

Пять типов адресации используются для формирования следующих четырех групп режимов адресации:

- основные режимы адресации (G);

- трехоперандный режим адресации (T);
- параллельные режимы адресации (P);
- режимы адресации условных переходов (B).

6.7.1 Основные режимы адресации

Инструкции, которые используют основные режимы адресации, являются инструкциями общего назначения, такими как ADDI, MPYF и LSH и т.п. Такие инструкции обычно представляются в форме:

dst операция src → dst,

где операнд назначения обозначен dst, операнд источника – src и «операция» определяет операцию, которая будет выполнена, используя основные режимы адресации для определения указанных операндов. Разряды 31-29 – нули, определяющие инструкции основного режима адресации. Разряды 22 и 21 определяют поле основного режима адресации (G), которое определяет назначение разрядов с 15 по 0 для адресации операнда src.

Возможные состояния разрядов 22 и 21 (поле G) следующие:

- 0 0 – регистровая (все регистры ЦПУ, если не определено иначе);
- 0 1 – прямая;
- 1 0 – косвенная;
- 1 1 – непосредственная.

Если поля src и dst специфицируют регистр, то значение в этих полях соответствует адресам регистров ЦПУ, как определено в таблице 6.1. Для основных режимов адресации допустимы следующие значения ARn:

- ARn, $0 \le n \le 7$

На рисунке 6.4 приведено кодирование для основных режимов адресации.

Примечание – «modn» обозначает поле модификации, определенное вместе с полем ARn. Смотри таблицу 6.2 для более подробной информации.

Основные режимы адресации			G		Назначение	Исходные операнды			
31 29		28 23	22 21		20 16	15 11	10 8	7 5	4 0
0 0 0		операция	0 0		dst	00 0 0 0	0 0 0	0 0 0	00 0 0 0
0 0 0		операция	0 1		dst	непосредственная			
0 0 0		операция	1 0		dst	modn	ARn	disp	
0 0 0		операция	1 1		dst	прямая			

Рисунок 6.4 – Кодирование для основных режимов адресации

6.7.2 Трехоперандные режимы адресации

Девятнадцать трехоперандных инструкций используют восемь методов адресации. Такие инструкции как, например, ADDI3, LSH3, CMPF3 или XOR3 обычно представляются в форме:

src1 операция src2 # dst,

где операнд назначения обозначен dst, операнды источников src1 и src2, «операция» определяет выполняемую операцию. Заметим, что цифра «3» может быть опущена в трехоперандных инструкциях.

В разрядах 31-29 установлено значение 001, обозначающее инструкции трехоперандного режима адресации. Разряды 22 и 21 определяют поле (T) трехоперандного режима адресации, которое определяет, как интерпретируются разряды

15-0 для адресации операндов src. Разряды 15-8 используются для определения адреса src1, и разряды 7-0 – для определения адреса src2.

Варианты значений для разрядов 22 и 21 следующие (Т):

Тип 1

Т	src1	src2	dst
00	Регистровый (любой регистр ЦПУ)	Регистровый (любой регистр ЦПУ)	Rx
01	Косвенный (disp = 0, 1, IR0, IR1)	Регистровый (любой регистр ЦПУ)	Rx
10	Регистровый (любой регистр ЦПУ)	Косвенный (disp = 0, 1, IR0, IR1)	Rx
11	Косвенный (disp = 0, 1, IR0, IR1)	Косвенный (disp = 0, 1, IR0, IR1)	Rx

Тип 2

Т	src1	src2	dst
00	Регистровый (любой регистр ЦПУ)	Непосредственный 8-ми битный со знаком	Rx
01	Регистровый (любой регистр ЦПУ)	Косвенный *+ARn (5-битное смещение без знака)	Rx
10	Косвенный *+ARn (5-битное смещение без знака)	Непосредственный 8-ми битный со знаком	Rx
11	Косвенный *+ARn (5-битное смещение без знака)	Косвенный *+ARn (5-битное смещение без знака)	Rx

Примечание – disp – смещение, Rx – любой регистр из основного регистрового файла ЦПУ.

На рисунке 6.5 приведено кодирование для трехоперандной адресации. Если поля src1 и src2 используют одинаковые вспомогательные регистры, то оба адреса генерируются правильно. Однако, только значение, созданное полем src1, сохраняется в заданном вспомогательном регистре. Ассемблер выдает предупреждение, если это используется и определено пользователем.

Допустимы следующие значения ARn и ARm:

ARn, $0 \le n \le 7$

ARm, $0 \le m \le 7$

Сокращения «modm» или «modn» обозначают поле модификации, соответствующее ARn или ARm. В таблице 6.2 приведена полная информация.

В косвенной адресации трехоперандного режима адресации допустимо смещение 0 или 1 (если смещение используется), и могут быть использованы индексные регистры IR0 и IR1. Смещение на «1» – неявное и не закодировано явно в командном слове.

МДФ КОП Т Назначение Источник 1 Источник 2

Тип 1

31 28	27 23	22 21	20 16	15 13	12 11 10 8	7 5	4 2	0
0 0 1 0	операция	0 0	dst	0 0 0	src1	0 0 0	src2	
0 0 1 0	операция	0 1	dst	modn	ARn	0 0 0	src2	
0 0 1 0	операция	1 0	dst	0 0 0	src1	modn	ARn	
0 0 1 0	операция	1 1	dst	modn	ARn	modm	ARm	
Тип 2								
0 0 1 1	операция	0 0	dst	0 0 0	Rn	непосредственная		
0 0 1 0	операция	0 1	dst	0 0 0	Rn	disp	ARn	
0 0 1 0	операция	1 0	dst	disp	ARn	непосредственная		
0 0 1 0	операция	1 1	dst	disp	ARn	disp	ARn	

Примечание – Принятые условные обозначения: disp – смещение; КОП – код операции; МДФ – модификатор.

Рисунок 6.5 – Кодирование для трехоперандных режимов адресации

6.7.3 Режимы параллельной адресации

Инструкции, использующие параллельную адресацию (обозначены || – двумя вертикальными линиями), обеспечивают максимально возможный параллелизм вычислений. Операнды назначения обозначены d1 и d2, обозначающие dst1 и dst2, соответственно, см. рисунок 6.6. Операнды источника, обозначенные src1 и src2, используют регистры повышенной точности. Выполняемые параллельные операции обозначены как «операция».

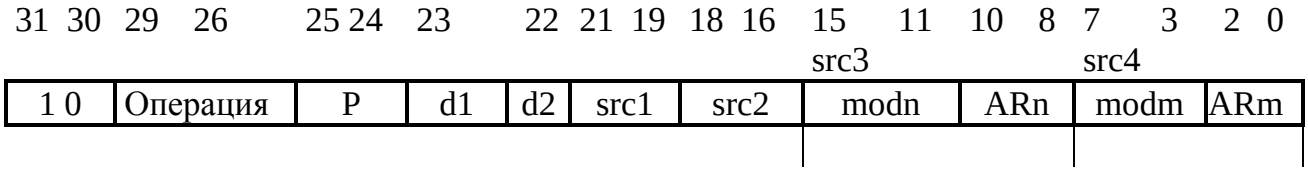


Рисунок 6.6 – Кодирование для параллельных режимов адресации

Поле параллельного режима адресации (P) определяет, как использовать операнды, т. е. являются они операндами источника или назначения. Связь между значением поля P и операндами подробно представлена в описании отдельных параллельных инструкций. Однако операнды всегда кодируются одинаково. В разрядах 31 и 30 установлено значение 10_2, обозначающее инструкции параллельного режима адресации. Разряды 25, 24 определяют поле параллельного режима адресации (P), которое определяет, как разряды 21-0 интерпретируются для адресации операндов источника (src). Разряды 21-19 используются для определения адреса src1, разряды 18-16 определяют адрес src2, разряды 15-8 – адрес src3 и разряды 7-0 – адрес src4.

Запись «modn» и «modm» обозначает поле модификации, соответствующее полю ARn или ARm (вспомогательный регистр).

Операнды параллельной адресации

src1 $0 \le$ src1 ≤ 7 – регистры повышенной точности R0-R7.

src2 $0 \le$ src2 ≤ 7 – регистры повышенной точности R0-R7.

d1 – если 0, dst1 – это R0. Если 1, dst1 – это R1.

d2 – если 0, dst2 – это R2. Если 1, dst2 – это R3.

P $0 \le P \le 3$.

src3 косвенная (disp = 0, 1, IR0, IR1).

src4 косвенная (disp = 0, 1, IR0, IR1).

Как и в трехоперандном режиме адресации, косвенная адресация в параллельном режиме адресации допустима со смещением «1» или «0» и при использовании индексных регистров IR0 и IR1. Смещение на «1» неявное и явно не кодируется в слове инструкции.

Для этого режима адресации, показанного на рисунке 6.6, если поля src3 и src4 используют одинаковые вспомогательные регистры, то оба адреса генерируются правильно, но только значение, созданное полем src3, хранится в определенном вспомогательном регистре. Ассемблер выдает предупреждение, если это условие определено пользователем.

Примечание – Только регистры R0-R7 используются, регистры R8-R11 не используются в параллельных инструкциях.

6.7.4 Режимы адресации с условным переходом

Инструкции, использующие режимы адресации для условных переходов Bcond, Bcond(D), CALLcond, DBcond и DBcond(D), могут выполнять различные условные операции. Разряды 31-27 установлены значением «01101», которые говорят о режиме адресации условных переходов. Разряд 26 установлен в «0» или «1».

Значение «0» выбирает DBcond, а «1» выбирает Bcond. Если B = 0, то используется регистровая адресация. Если B = 1, то используется адресация относительно PC. Разряд 21 устанавливает тип перехода. Если D = 0, то переход стандартный, иначе переход задержанный. Поле условий (cond) задает условие, которое проверяется для определения того, какое действие должно быть выполнено.

Если условие выполнено, то должен выполняться переход, в противном случае переход не выполняется. На рисунке 6.8 показано кодирование для режима адресации условных переходов.

DBcond(D):

31 26	25	24 22	21	20 16	15 5	4 0
0 1 1 0 1 1	B	ARn	D	cond	0 0 0 0 0 0 0 0 0 0 0	src reg
0 1 1 0 1 1	B	ARn	D	cond	непосредственная (относительно PC)	

Bcond(D):

31 26	25	24 22	21	20 16	15 5	4 0
0 1 1 0 1 0	B	00 0	D	cond	0 0 0 0 0 0 0 0 0 0 0	src reg
0 1 1 0 1 0	B	00 0	D	cond	непосредственная (относительно PC)	

CALLcond(D):

31 26	25	24 22	21	20 16	15 5	4 0
0 1 1 1 0 0	B	00 0	D	cond	0 0 0 0 0 0 0 0 0 0 0	src reg
0 1 1 1 0 0	B	00 0	D	cond	непосредственная (относительно PC)	

Рисунок 6.8 – Кодирование для режимов адресации с условным переходом

6.8 Циклическая адресация

Многие алгоритмы, такие как свертка и корреляция, требуют организации циклического буфера в памяти. В процессе свертки и корреляции циклический буфер используется для реализации скользящего окна, которое содержит самые последние значения, подлежащие обработке. По мере того как вводятся новые данные, они стирают предыдущую информацию. Для реализации циклического буфера необходимо использовать режим циклической адресации. В данном подразделе описан режим циклической адресации.

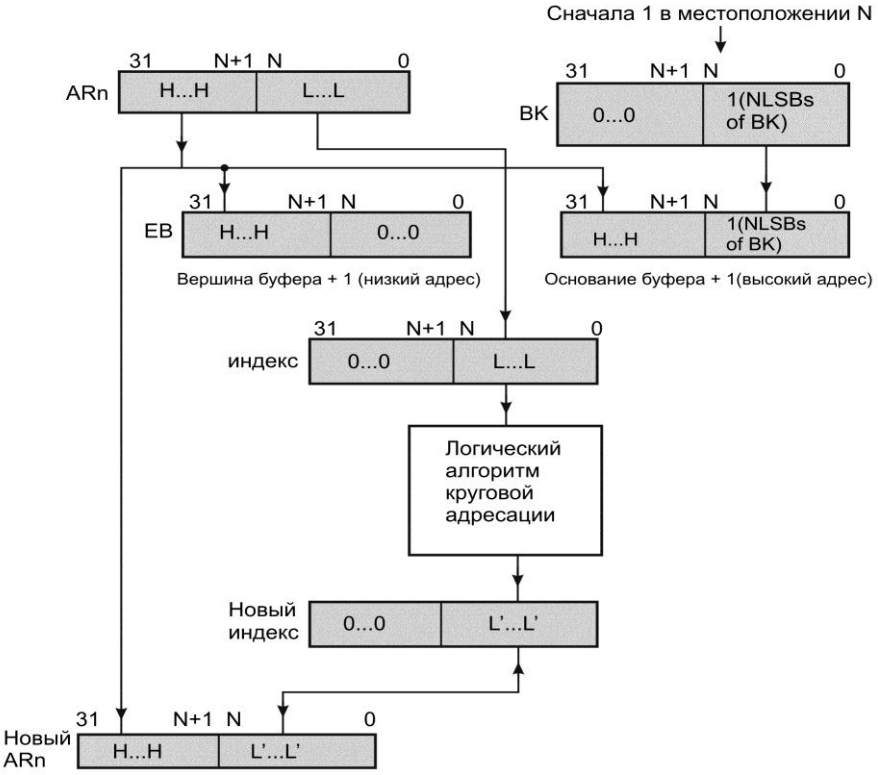
Регистр размера блока BK определяет размер циклического буфера. Если самый старший бит регистра BK, равный «1», обозначить N, где $N \leq 15$, то нижняя часть циклического буфера определяется как конкантенация разрядов от 31 до N+1 выбранного вспомогательного регистра ARn с битами от N до 0 регистра BK.

Адрес в верхней части буфера, именуемый эффективной базой (EB), равен конкатенации разрядов от 31 до (N+1) регистра ARn. Разряды от N до 0 эффективной базы (EB) равны «0».

На рисунке 6.9 показана связь между регистром размера блока BK, вспомогательным регистром ARn, нижней частью циклического буфера, верхней частью циклического буфера и индексным регистром в циклическом буфере.

Циклический буфер размером R должен начинаться на границе K-го разряда (т.е. K младших значащих разрядов циклического буфера должны быть равны «0»), где K – наименьшее целое, удовлетворяющее соотношению $2^K > R$. Значение R также должно быть загружено в регистр BK. Например, 31 – словный циклический буфер должен начинаться с адреса, чьи пять младших значащих разрядов равны «0» (т. е. $XXX\ldots XX00000_2$) и это значение должно быть загружено в регистр BK.

Примечание – Если регистр BK равен «0», то циклический адрес не вычисляется, а вычисляется последовательный линейный адрес.



Примечание –Принятые условные обозначения:

- ARn – вспомогательный регистр n;
- BK – регистр размера блока;
- EB – эффективная база;
- H – старшие разряды;
- L – младшие разряды;
- L'– новые младшие разряды;
- LSB = младший значащий разряд;
- N – значение разряда.

Рисунок 6.9 – Блок-схема циклической адресации

В циклической адресации «index» относится к N младшим значащим разрядам выбранного вспомогательного регистра, а «step» – это величина, которая будет добавлена к вспомогательному регистру или вычтена из него. При использовании циклической адресации необходимо придерживаться следующих двух правил:

- используемый шаг должен быть меньше или равен размеру блока;
- сначала адресуется циклическая очередь, вспомогательный регистр должен указывать на элемент в циклической очереди.

Алгоритм для циклической адресации следующий:

Если $0 \le index + step < BK$:

index = index + step.

Иначе если $index + step \ge BK$:

index = index + step - BK.

Иначе если $index + step < 0$, то

index = index + step + BK.

На рисунке 6.10 показана реализация циклического буфера, который иллюстрирует взаимосвязь между сгенерированными величинами и элементами в циклическом буфере.

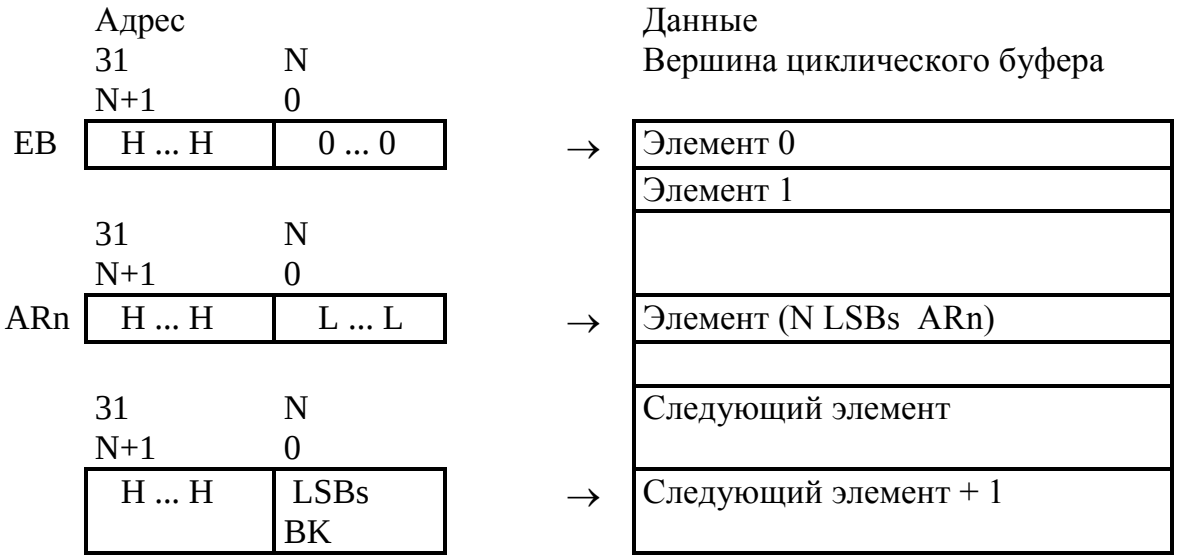


Рисунок 6.10 – Реализация циклического буфера

На рисунке 6.11 приведен пример пошагового выполнения инструкции при циклической адресации. Предполагается, что все AR содержат четыре разряда, AR0=0000 и BK=0110 (размер блока равен 6). Этот пример показывает последовательность модификаций и результирующее значение AR0. Он также иллюстрирует, как указатель продвигается по циклической очереди с разным шагом (как с увеличением, так и с уменьшением).

```
*AR0++(5)%    ; AR0 = 0 (нулевое значение).
*AR0++(2)%    ; AR0 = 5 (первое значение).
*AR0--(3)%    ; AR0 = 1 (второе значение).
*AR0++(6)%    ; AR0 = 4 (третье значение).
*AR0-- %      ; AR0 = 4 (четвертое значение).
*AR0          ; AR0 = 3 (пятое значение).
```

Значение		Данные	Адрес
0-е	→	Элемент 0	0
2-е	→	Элемент 1	1
		Элемент 2	2
5-е	→	Элемент 3	3
4-е, 3-е	→	Элемент 4	4
1-е	→	Элемент 5 (последний элемент)	5
		Последний элемент + 1	6

Рисунок 6.11 – Пример циклической адресации

Циклическая адресация особенно полезна для реализации КИХ-фильтров. На рисунке 6.12 показана одна из возможных структур, данных для КИХ-фильтров. Заметим, что начальное значение AR0 указывает на h(N-1), а начальное значение AR1 указывает на x(0). На рисунке 6.13 показана циклическая адресация, используемая в программах для ИС 1867ВМ9Ф.

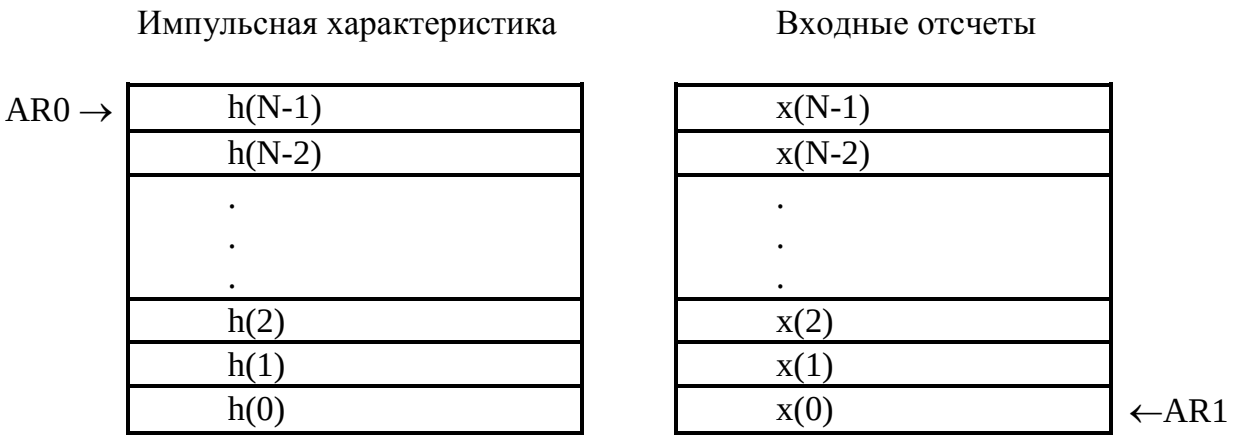


Рисунок 6.12 – Структура данных для КИХ-фильтров

```
* Инициализация
  LDI N, BK          ; Загрузить размер блока
  LDI H, AR0         ; Загрузить указатель на импульсную характеристику
  LDI X, AR1         ; Загрузить указатель на нижнюю часть буфера
                     ; входных отсчетов
**
 TOP LDF IN, R3      ; Считать входной отсчет
 STF R3,*AR1++%      ; Сохранить с другими отсчетами и указать на
                     ; вершину буфера
 LDF 0, R0           ; Инициализировать R0
 LDF 0, R2           ; Инициализировать R2
* Фильтр
RPTS N – 1           ; Повторить последнюю инструкцию
MPYF3  *AR0++%,*AR1++%, R0
||ADDF3 R0, R2, R2 ; Умножить и аккумулировать
ADDF R0, R2          ; Последний результат аккумулирован
*
STF R2, Y            ; Сохранить результат
B TOP                ; Повторить
```

Рисунок 6.13 – Программа для КИХ-фильтра с использованием циклической адресации

6.9 Бит-реверсивная адресация

Бит-реверсивная адресация увеличивает скорость выполнения алгоритмов БПФ и использования памяти программами, реализующие алгоритмы БПФ с различными основаниями. Базовый адрес бит-реверсивной адресации должен быть размещен на границе размера таблицы. Например, если IR0 = 2^{n-1}, то n младших разрядов базового адреса должны быть равны нулю. Базовый адрес данных в памяти должен быть на границе 2^n. Один вспомогательный регистр указывает на физический адрес значения данных. Регистр IR0 определяет половину размера БПФ; например, значение, содержащееся в IR0 должно быть равно 2^{n-1}, где n – целое и размер БПФ = 2^n. При добавлении содержимого IR0 к вспомогательному регистру, используя бит-реверсную адресацию, генерируется адрес в соответствии с бит-реверсным алгоритмом. При вычислении бит-реверсивного адреса разряд переноса распространяется в сторону младших разрядов числа.

Чтобы проиллюстрировать этот вид адресации, рассмотрим восьмиразрядные вспомогательные регистры. Пусть AR2 содержит значение 0110 0000_2 (96_{10}). Это базовый адрес данных в памяти. Пусть IR0 содержит значение 0000 1000_2 (8_{10}).

Рисунок 6.14 показывает последовательную модификацию AR2 и результирующие значения AR2.

```
*AR2++(IR0)B   ; AR2 = 0110 0000 (0-е значение)
*AR2++(IR0)B   ; AR2 = 0110 1000 (1-е значение)
*AR2++(IR0)B   ; AR2 = 0110 0100 (2-е значение)
*AR2++(IR0)B   ; AR2 = 0110 1100 (3-е значение)
*AR2++(IR0)B   ; AR2 = 0110 0010 (4-е значение)
*AR2++(IR0)B   ; AR2 = 0110 1010 (5-е значение)
*AR2++(IR0)B   ; AR2 = 0110 0110 (6-е значение)
*AR2           ; AR2 = 0110 1110 (7-е значение)
```

Рисунок 6.14 – Пример бит-реверсной адресации

В таблице 6.3 приведена взаимосвязь индексного шага и четырех младших разрядов AR2. Можно найти четыре младших значащих разряда путем реверсирования битового шаблона шагов. В таблице 6.3 значение бит-реверсивного шаблона на следующем шаге равно бит-реверсивной сумме (перенос распространяется в сторону младших разрядов) значений бит-реверсивного шаблона на предыдущем шаге и индекса равного 8_{10}.

Таблица 6.3 – Вычисление бит-реверсивного адреса с шагом равным 8_{10}

Шаг	Битовый шаблон	Бит-реверсивный шаблон	Бит-реверсивный шаг
0	0000	0000	0
1	0001	1000	8
2	0010	0100	4
3	0011	1100	12
4	0100	0010	2
5	0101	1010	10
6	0110	0110	6
7	0111	1110	14
8	1000	0001	1
9	1001	1001	9
10	1010	0101	5
11	1011	1101	13
12	1100	0011	3
13	1101	1011	11
14	1110	0111	7
15	1111	1111	15
Примечание – Бит-реверсивная адресация реализуется также и сопроцессором ПДП.			

6.10 Управление системным стеком и стеком пользователя

Процессор ИС 1867ВМ9Ф содержит специальный регистр указателя системного стека SP для создания стеков в памяти. Вспомогательный регистр может также быть использован для построения множества общих линейных списков. В этом разделе описывается создание следующих виды линейных списков:

Стек – это линейный список, для которого запись и удаление из списка выполняются в одном и том же месте списка.

Очередь – это линейный список, для которого запись выполняется в одном конце списка, удаление выполняется в другом конце списка.

Двойная очередь – это линейный список с двумя концами, для которого запись и удаление могут выполняться в/из любого конца списка.

Указатель системного стека SP – это 32-разрядный регистр, который содержит адрес вершины системного стека. Системный стек заполняется от младших адресов памяти к старшим, смотри рисунок 6.15. SP всегда указывает на следующий элемент, проталкиваемый в стек. При проталкивании данных в стек выполняется предекрементирование, а при выталкивании данных из стека выполняется постдекрементирование указателя системного стека.

Счетчик инструкций проталкивается в системный стек во время вызовов подпрограмм, прерываний и системных прерываний. Стек может выполнять проталкивание и выталкивание данных в/из стека, используя инструкции PUSH, POP, PUSHF и POPF.

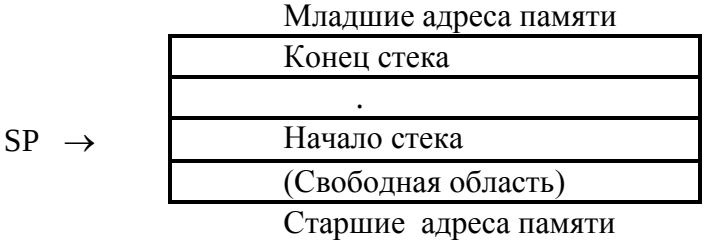


Рисунок 6.15 – Конфигурация системного стека

6.10.1 Стеки

Стеки могут быть построены от младших адресов памяти к старшим или от старших к младшим. Стеки могут быть построены с использованием различных режимов изменения вспомогательных регистров AR – предекрементирование/ декрементирование и постдекрементирование/декрементирование. Изменение стека от старших адресов памяти к младшим может быть выполнено двумя путями:

- 1 вариант: Записывать в память, используя косвенную адресацию *-- ARn для проталкивания данных в стек, и считывать из памяти, используя косвенную адресацию *ARn ++ для выталкивания данных из стека;

- 2 вариант: Записывать в память, используя косвенную адресацию *ARn -- для проталкивания данных в стек, и считывать из памяти, используя косвенную адресацию *++ ARn для выталкивания данных из стека.

Рисунок 6.16 иллюстрирует два этих случая. Различие только в том, что при использовании варианта 1 AR всегда указывает на вершину стека, а во втором варианте AR всегда указывает на следующую свободную область в стеке.

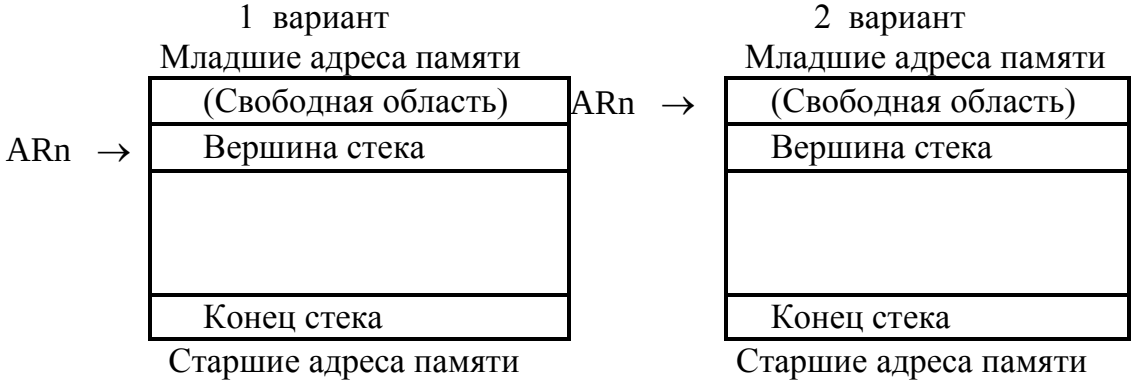


Рисунок 6.16 – Два варианта реализации стека

Изменение стека от младших адресов памяти к старшим может быть реализовано двумя путями:

- 3 вариант: Записывать в память, используя косвенную адресацию *++ ARn для проталкивания данных в стек, и считывать из памяти, используя косвенную адресацию *ARn -- для выталкивания данных из стека;

- 4 вариант: Записывать в память, используя косвенную адресацию *ARn ++ для проталкивания данных в стек, и считывать из памяти, используя косвенную адресацию *-- ARn для выталкивания данных из стека.

На рисунке 6.17 показаны эти два случая. В третьем случае AR всегда указывает на вершину стека. В четвертом случае AR всегда указывает на следующую свободную область в стеке.

Рисунок 6.17 – Реализация стеков от младших адресов памяти к старшим

7 Управление процессом выполнения программы

ИС 1867ВМ9Ф содержит блоки, которые обеспечивают программное и аппаратное управление процессом выполнения программы. Программное управление обеспечивает повторения, переходы (ветвления), вызовы подпрограмм, программные прерывания и возвраты из подпрограмм и прерываний. Аппаратное управление включает в себя системные прерывания.

7.1 Режим повторений

Режим повторений может применяться для организации циклов без дополнительной потери производительности. Для многих алгоритмов наибольшее процессорное время тратится на реализацию внутреннего ядра программы. Использование режима повторений позволяет максимально сократить время выполнения секций программы, критичных ко времени.

ИС 1867ВМ9Ф содержит 3 инструкции для реализации циклов без дополнительной потери производительности: инструкция RPTB (повторение блока кода), RPTBD (повторение блока кода с задержкой) и RPTS (повторение одной инструкции).

Инструкции RPTB и RPTBD позволяют выполнять блок кода определенное количество раз. Инструкция RPTS обеспечивает повторение одной инструкции определенное количество раз и уменьшает нагрузку на шины путем однократного выбора инструкции.

Инструкции RPTB и RPTS – это четырехтактные инструкции. Четыре такта необходимы только при первом выполнении цикла. Все последующие повторения выполняются с нулевыми потерями. Инструкция RPTBD – это однотактная инструкция.

Три регистра RS, RE и RC управляют счетчиком команд при его обновлении в режиме повторения. В таблице 7.1 описаны эти регистры.

Таблица 7.1 – Регистры режима повторения

Регистр	Функция
RS	Регистр начального адреса повторяемого блока содержит адрес первой инструкции повторяемого блока инструкций.
RE	Регистр конечного адреса повторяемого блока содержит адрес последней инструкции повторяемого блока кода. RE ≥ RS.
RC	Регистр количества повторений содержит значение, которое на «1» меньше числа повторений блока кода.

Для корректного выполнения программы в режиме повторений необходимо, чтобы все вышеперечисленные регистры, а также регистр состояния были корректно проинициализированы. Инструкции RPTB, RPTBD и RPTS выполняют эту инициализацию немного по-разному.

7.1.1 Управляющие биты в режиме повторений

Существует два бита, которые очень важны для операций RPTB, RPTBD и RPTS, это биты RM и S.

RM (флаг режима повторений) – это бит в регистре состояния, который определяет, работает ли процессор в режиме повторений. Если RM = 0, то режим повторений не выбран, если RM = 1, то режим повторений выбран.

Бит S скрыт от пользователя, но он необходим для полного описания операций RPTB, RPTBD и RPTS. Если RM = 1 и S = 0, то выполняются RPTB или RPTBD, при этом выборка кода инструкции осуществляется из памяти. Если RM = 1 и S = 1, то выполняется RPTS, при этом после первой выборки кода инструкции последующая выборка осуществляется из регистра инструкций IR.

7.1.2 Операции в режиме повторений

Информация в регистрах режима повторения и соответствующих битах управления используется для управления изменением программного счетчика PC, когда выполняется выборка кода инструкции в режиме повторения. В режимы повторения, после выполнения каждой инструкции, содержимое регистра RE (регистр конца повторяющегося блока) сравнивается с содержимым счетчика инструкций PC. Если значения совпадают и счетчик повторений неотрицательный, то счетчик повторений уменьшается, в PC загружается начальный адрес повторений и обработка продолжается. Выборка и соответствующие биты состояния, при необходимости, изменяются.

Примечание – Счетчик повторений RC никогда не изменяется, если RM = 0.

Более подробно алгоритм изменения PC показан в примере 7.1.

Примечание – Максимальное число повторений возможно, когда RC = 8000 0000h. Это соответствует 8000 0001h повторений. Минимальное число повторений возможно, когда RC = 0. Этому соответствует одно повторение. RE должен быть больше или равен RS (RE ≥ RS), иначе повторений не будет, даже если RM остается установленным в 1. Если записать «0» в счетчик повторений или в бит RM регистра состояния, то можно остановить цикл повторений до его завершения.

Пример 7.1 – Алгоритм управления режимом повторения

```
if RM == 1                    ; Если в режиме повторений (RPTB или RPTS)
if S == 1                     ; Если RPTS
Если это первая выборка, то
```

```
выполнить выборку инструкции
из памяти                    ; Выборка инструкции из памяти
else                         ; Если не первая выборка
выполнить выборку инструкции
из IR                        ; Выборка инструкции из IR
RC – 1 -> RC                 ; Декрементирование RC
if RC < 0                    ; Если RC отрицательное
                             ; Режим повторения завершен
0 -> ST(RM)                  ; Выключить бит режима повторения
0 -> S                       ; Очистить S
PC + 1 -> PC                 ; Инкремент PC
else if S == 0               ; Если RPTB
выполнить выборку инструкции
из памяти                    ; Выборка инструкции из памяти
if PC == RE                  ; Если это конец блока
RC – 1 -> RC                 ; Декрементирование RC
if RC ≥0                     ; Если RC положительное
RS -> PC                     ; Установить PC на начало блока
else if RC < 0               ; Если RC отрицательное
0 -> ST(RM)                  ; Выключить биты режима повторения
0 -> S                       ; Очистить S
      PC + 1 -> PC           ; Инкрементировать PC
```

7.1.3 Инструкции RPTB и RPTBD

Инструкции RPTB и RPTBD повторяют блок кода программы определенное число раз. RPTBD – задержанная форма инструкции RPTB, которая позволяет выполнить еще 3 инструкции после себя. Эти 3 инструкции не являются частью блока повторений, но они выполняются перед началом блока повторения. Поэтому конвейер остается полным и RPTBD инструкция выполняется в одном цикле.

Число повторений блока инструкций равно значению регистра счетчика повторений с добавлением «1». Так как RPTB и RPTBD не загружают RC, то необходимо записать в него нужное значение. Загрузка значения в RC должна быть осуществлена перед выполнением RPTB/RPTBD инструкций. Загрузка регистра RC не может быть одной из трех инструкций, следующих за RPTBD. Пример 7.2 показывает типичную установку блока повторений.

Пример 7.2 – Выполнение RPTB

```
LDI   15, RC                 ; Загружает 15 в счетчик повторений
RPTB        ENDLOP           ; Выполняет блок кода
STLOOP                       ; от STLOOP до ENDLOP 16 раз
                             .
                             .
ENDLOP
```

Повторение блока кода, выполняемое с помощью RPTB и RPTBD, можно прервать. Однако прерывания невозможны в течение выполнения 3 инструкций, следующих за RPTBD. Ни одна из этих трех инструкций не может изменить регистр PC или ход выполнения программы. Это ограничение также применимо к задержанным переходам, которые описаны в подразделе 7.2.

При выполнении RPTB src или RPTBD src выполняется последовательность из 4 операций.

1 операция. Загрузка начального адреса повторений в RS.

Для RPTB – это адрес, следующий за инструкцией

PC (RPTB) + 1 → RS,

Для RPTBD – это четвертый адрес, следующий за инструкцией

PC (RPTBD) + 4 → RS.

2 операция. Загрузка конечного адреса повторений в RE.

Для RPTB в относительном режиме адресации 24-разрядный операнд плюс RS является конечным адресом

src + PC для RPTB + 1 → RE.

Для RPTBD в относительном режиме адресации 24-разрядный операнд плюс RS является конечным адресом

src + PC для RPTBD + 3 → RE.

При регистровом методе адресации содержимое регистра src является конечным адресом, т. е. содержание регистра src → RE

3 операция. Устанавливается регистр состояния для индикации режима повторений.

«1» → RM бит регистра состояния (флаг режима повторений): индикация, что это операция режима повторений.

4 операция. «0» → S разряд (внутренний, непрограммируемый бит).

7.1.4 Инструкция RPTS

Инструкция RPTS src повторяет инструкцию, следующую за RPTS (src +1) раз. Повторение одной инструкции, инициированной с помощью RPTS, не может быть прервано, так как RPTS выбирает код инструкции только один раз и далее хранит его в регистре инструкций для повторного использования. Прерывание в этом случае может привести к потере слова инструкции. Повторная выборка слова инструкции из регистра инструкций уменьшает количество обращений к памяти, и в результате действует как программный КЭШ на одно слово. Если необходимо, то для прерывания программы в режиме повторения используются инструкции RPTB/RPTBD.

Во время выполнения RPTS src производится следующая последовательность операций:

- Шаг 1 – PC + 1 → RS.
- Шаг 2 – PC + 1 → RE.
- Шаг 3 – «1» → RM (бит регистра состояния).
- Шаг 4 – «1» → Бит S.
- Шаг 5 – src → RC (счетчик повторений).

Инструкция RPTS загружает все регистры и биты режима, необходимые для операции повторения одной инструкции. Шаг 1 – загружается начальный адрес блока в RS. Шаг 2 – загружается конечный адрес в RE (конечный адрес блока). Так как это повторение одной инструкции, то начальный и конечный адреса совпадают. Шаг 3 –устанавливается регистр состояния для указания режима повторения операции. Шаг 4 – указывается, что это режим повторения одной инструкции. Далее операнд src загружается в RC – шаг 5.

7.1.5 Правила-ограничения в режиме повторения

Так как режимы повторений изменяют счетчик инструкций, то другие инструкции не могут изменять счетчик инструкций в то же самое время, поэтому применяется 2 правила.

Правило 1. Последней инструкцией в блоке повторений не может быть Bcond, DBcond, CALL, CALLcond, TRAPcond, RETIcond, RETScond, IDLE, RPTB, RPTS.

Пример 7.3 показывает неправильное использование стандартных переходов.

Правило 2. Ни одна из 4 последних инструкций в блоке повторений не может быть BcondD, BRD, DBcondD, RPTBD, LAJ, LAJcond, BcondAF, BcondAT, RETIcondD.

Пример 7.3 – Некорректное использование стандартных переходов

```
LDI    15, RC              ; Загружает 15 в счетчик повторений
RPTB ENDLOP                ; Выполняет блок кода
STLOOP                     ; от STLOOP до ENDLOP 16 раз
…
…
…
ENDLOP BR OOPS             ; нарушение правила 1
```

Пример 7.4 показывает неправильное использование задержанных переходов.

Пример 7.4 – Некорректное использование задержанных переходов

```
LDI    15, RC              ; Загружает 15 в счетчик повторений
RPTB ENDLOP                ; Выполняет блок кода
STLOOP                     ; от STLOOP до ENDLOP 16 раз
…
…
…
BRD   OOPS                 ; нарушение правила 2
ADDF
MPYF
ENDLOP     SUBF
```

7.1.6 Значение регистра RC после завершения режима повторений

Для инструкций RPTB, RPTBD значение регистра RC уменьшается до 0000 0000h, за исключением случая, когда размер блока равен «1». В этом случае значение уменьшается до FFFF FFFFh. Однако, если при выполнении инструкции RPTB или RPTBD и размером блока, равным «1», возникает конфликт конвейера, то RC регистр уменьшается до 0000 0000h. Пример 7.5 показывает конфликт конвейера.

Инструкция RPTS уменьшает значение регистра RC до FFFF FFFFh. Однако если при выполнении RPTS в последнем цикле возникает конфликт конвейера, то значение RC регистра уменьшается до 0000 0000h.

В любом случае, число повторений остается RC+1 независимо от конечного значения RC.

Пример 7.5 – Конфликт конвейера при выполнении инструкции RPTB

```
EDC    .word 4000 0000h    ; программа расположена в 4000 000Fh
LDP    EDC
LDI    @EDC, AR0
LDI    15, RC              ; загрузка 15 в счетчик повторений
RPTB ENDLOP                ; выполнение блока кода
ENDLOP LDI *AR0, R0        ; чтение *AR0 конфликтует с выбором
                           ; инструкции. RC уменьшается до 0
                           ; Если КЭШ разрешен, RC уменьшается
                           ; до FFFF FFFFh
```

7.1.7 Вложенность блоков повторений

Блоки повторений RPTB и RPTBD могут либо иметь вложения, либо сами являться вложенными. Так как управление блоками повторений зависит от регистров RS, RE, RC, ST, то они должны хранить информацию о вложенных блоках повторений. Например, если используется программа, работающая с прерываниями с использованием RPTB или RPTBD, то может возникнуть ситуация, при которой прерывание, связанное с программой, может произойти во время выполнения другого блока повторений. Программа, обслуживающая прерывания, должна проверять RM разряд для определения того, когда режим повторений включен. Если RM установлен, то программа обработки прерываний должна сохранить ST, RS, RE, RC в соответствующем порядке, а после выполнять блок повторений. Перед возвратом из цикла повторений, программа обработки прерываний должна восстановить значения ST, RS, RE, RC в соответствующем порядке. Если разряд RM не установлен, то не требуется сохранять и восстанавливать значения этих регистров.

Инструкция RPTS также может быть использована в блоках повторений, если значения соответствующих регистров сохранены.

Так как счетчик инструкций изменяется в конце блока в соответствии с содержанием регистров RS, RE, RC, то не должно производиться никаких операций в конце блока, способных изменить счетчик повторений или счетчик инструкций.

Сохранение и восстановление значений в этих регистрах занимает 4 цикла. Таким образом, иногда более экономичным может стать применение вложенного цикла с более традиционным способом использования регистра в качестве счетчика, а затем задержанного перехода. Часто использование счетчика для внешнего цикла и инструкций RPTB и RPTBD для внутреннего цикла обеспечивает наилучшее быстродействие.

Примечание – Для обеспечения корректной работы очень важен порядок использования регистров, в которых осуществляется запись/чтение. Значение регистра ST должно восстанавливаться в конце – после восстановления регистров RC, RE, RS. Значение регистра ST должно восстанавливаться после восстановления RC, так как бит RM не может быть установлен в единицу, если RC регистр установлен в «0» или «1». По этой причине, если используется инструкция POP ST (где ST(RM) = 1) до тех пор, пока RC = 0, то инструкция POP перезаписывает все биты ST регистра, за исключением бита RM, который остается равным «0» (режим повторений отключен). Биты RS и RE также должны быть корректно установлены перед включением режима повторений.

7.2 Задержанные переходы

В процессоре ИС 1867ВМ9Ф имеются два основных типа ветвлений: стандартные переходы и задержанные переходы. Стандартные переходы освобождают конвейер до выполнения перехода, чтобы гарантировать корректное управление программным счетчиком. В результате этого ветвление выполняется за четыре цикла. В этот класс переходов включены стандартные переходы Bcond, повторения, вызовы, возвраты и программные прерывания.

Задержанные переходы не освобождают конвейер, но также гарантируют, что следующие три инструкции будут выполнены до того, как программный счетчик будет изменен ветвлением. Результатом является переход, требующий только одного цикла, таким образом, скорость работы задержанного перехода очень близка к оптимальному режиму повторения блоков в процессоре ИС 1867ВМ9Ф. Однако, в отличие от режимов повторения блоков, задержанные переходы могут быть использованы не

только для организации цикла. Каждый задержанный переход имеет копию стандартного перехода, которая применяется, когда задержанный переход не может быть использован.

Условные задержанные переходы используют условия, отражаемые в регистре состояния и устанавливающиеся в конце выполнения инструкции, которая непосредственно предшествует задержанному переходу. Флаги условий не зависят от инструкций, следующих за задержанным переходом. Время выполнения задержанного перехода остается прежним, независимо от того, произошло ветвление или нет.

Когда задержанные переходы выбраны, то они остаются задержанными до тех пор, пока три следующие за эти переходом инструкции не будут выполнены. Ни одна из трех инструкций из перечисленного списка не может следовать за задержанным переходом – Bcond, BcondD, BcondAF, BcondAT, BR, BRD, DBcond, DBcondD, CALL, CALLcond, IDLE, LAJ, LAJcond, LATcond, RETIcond, RETIcondD, RETScond, RPTB, RPTBD, RPTS, TRAPcond. Это ограничение также применяется для инструкции RPTBD, описанной в 7.1.3.

Задержанные переходы запрещают прерывания до тех пор, пока три инструкции, следующие за задержанным переходом, не будут выполнены. Это не зависит от того, произошло ветвление или нет.

При некорректном использовании задержанных переходов PC будет не определен.

Пример 7.6 показывает некорректное размещение задержанного перехода.

Пример 7.6 – Некорректное размещение задержанного перехода.

```
B1:     BD L1
        NOP
        NOP
B2:     B L2        ; Этот переход установлен неправильно
        NOP
        NOP
        NOP
            .
            .
```

Иногда в программе необходим переход, после которого должно выполняться меньше трех инструкций. В этом случае задержанный переход также обеспечивает наилучшее быстродействие. Это отражено в примере 7.7, где вместо третьей неиспользуемой инструкции применяется NOP.

Пример 7.7 – Выполнение задержанного перехода.

```
*Выполнение задержанного перехода
        …
        …
        …
        …
        LDF *+AR1(5), R2    ; Загрузка содержимого памяти в R2
        BGED SKIP           ; Если загружено число ≥ 0, переход
                            ; (задержанный)
        LDFN R2, R1         ; Если загружено число < 0, загрузить его в R1
        SUBF 3.0, R1        ; Вычесть 3 из R1
        NOP                 ; Фиктивная операция для завершения
                            ; задержанного перехода
        MPYF 1.5, R1        ; Продолжить отсюда, если загруженное число < 0
```

…
…
…
SKIP LDF R1, R3 ; Продолжить отсюда, если загруженное число ≥ 0

Существует два типа задержанных ветвлений: переходы без аннулирования и переходы с аннулированием.

7.2.1 Задержанные переходы без аннулирования

Задержанные переходы без аннулирования не очищают конвейер, но гарантируют выполнение следующих за переходом трех инструкций перед тем, как ветвление изменит значение счетчика инструкций. Задержанные переходы без аннулирования – это инструкции BcondD, BRD, DbcondD.

7.2.2 Задержанные переходы с аннулированием

Задержанные переходы с аннулированием могут условно отменять следующие за переходом три инструкции. Задержанные переходы с аннулированием BcondAF, BcondAT:

- BcondAF. Если условие истинно, инструкция BcondAF выполняет три инструкции, следующие за переходом, затем выполняет переход. Если условие ложно, переход не происходит, отменяются фаза выполнения первой инструкции, следующей за переходом, а также фазы чтения и выполнения следующих второй и третьей инструкций.
- BcondAT . Если условие истинно, происходит переход, отменяются фаза выполнения первой инструкции, следующей за переходом, а также фазы чтения и выполнения следующих второй и третьей инструкций. Если условие ложно, инструкция BcondAT выполняет три инструкции, следующие за переходом, и не выполняет переход.

7.3 Вызовы, программные прерывания, ветвления, переходы и возвраты

Вызовы и программные прерывания обеспечивают выполнение подпрограммы или функции с возвратом в вызывавшую программу.

Инструкции CALL, CALLcond и TRAPcond сохраняют значение PC в стеке до изменения содержимого PC. Таким образом, стек обеспечивает возврат из программных прерываний или вызванных подпрограмм при использовании инструкций RETScond или RETIcond.

CALL – это четырехцикловая инструкция, в то время как CALLcond и TRAPcond – пятицикловые. Этим трем, вышеуказанным, инструкциям по выполняемым функциям эквивалентны в соответствующем порядке инструкции LAJ, LAJcond, LATcond, которые являются одноцикловыми.

Инструкция CALL помещает значение следующего PC в стек и записывает содержимое src в PC. src – это 24-разрядное относительное смещение или регистр. На рисунке 7.1 приведена временная диаграмма обработки инструкции CALL.

Инструкция CALLcond подобна инструкции CALL, за исключением:

- она выполняется, если только определенное условие выполнено (20 условий, включая безусловные, приведены в подразделе 10.2);
- src является либо 24-разрядным относительным (относительно PC) смещением, либо регистром.

Инструкция TRAPcond также выполняется, только если определенное условие «истинно» (условия те же, что и для CALLcond).

При ее выполнении:

- значения разрядов GIE и CF регистра состояния сохраняются в разрядах PGIE и PCF регистра состояния;

- прерывания отключены (GIE = 0) и КЭШ заблокирован (CF = 0);

- следующее значение PC сохраняется в стеке;

- определенный вектор из таблицы векторов программных прерываний загружается в PC. Адрес вектора находится в зависимости от номера программного прерывания соответствующей инструкции.

Использование инструкций RETIcond или RETIcondD для возврата повторно разрешает прерывания, если бит GIE регистра состояния был раньше установлен, а также перезаписывает бит CF.

Инструкция RETScond выполняет возврат из любой вышеперечисленной инструкции, загружая вершину стека в PC. Для выполнения инструкции необходимо, чтобы выполнилось заданное условие. Условия те же, что и для CALLcond.

Инструкция RETIcond выполняет возврат из программных прерываний или вызовов. Она аналогична инструкции RETScond и, кроме того, копирует разряды PGIE и PCF в GIE и CF в регистре состояния. Условия те же, что и для CALLcond.

Инструкция RETIcondD выполняет возврат из программных прерываний или вызовов. Она аналогична инструкции RETIcond, но, кроме этого, в первую очередь выполняет три инструкции, следующие непосредственно за RETIcondD. Условия те же, что и для CALLcond.

Инструкции LAJ, LAJcond, LATcond возвращают адрес в регистр повышенной точности R11.

После выполнения трех инструкций, следующих за инструкцией перехода, LAJ обеспечивает переход на адрес, определенный в режиме 24-разрядного относительного смещения, см. подраздел 6.6 «Относительная адресация».

Адрес назначения инструкции LAJcond также определяется PC-относительной адресацией (смещением) или указанным регистром. Если условие истинно, то LAJcond в первую очередь выполняет три инструкции, следующие непосредственно за инструкцией LAJcond, перед тем, как выполнить переход. Если условие «ложно», то выполнение программы продолжается сразу после инструкции LAJcond.

После выполнения трех инструкций, следующих за командой LATcond, вызывается один из 512-ти векторов таблицы программных прерываний в соответствии с TVTP, см. подраздел 3.2 «Дополнительный регистровый файл ЦПУ».

Вызовы функций, подпрограмм и программные прерывания предназначены для одной и той же цели – вызывать и выполнять подпрограмму, а после осуществлять возврат в основную программу. Программные прерывания имеют два преимущества по сравнению с обычными вызовами:

- Системные прерывания автоматически запрещаются, когда выполняется программное прерывание. Это позволяет выполнить критическую программу без риска ее прерывания. Таким образом, программные прерывания обычно заканчиваются одной из инструкций RETIcond или RETIcondD для повторного включения разрешения системных прерываний, если разряд GIE регистра состояния был ранее установлен.

- Также вы можете использовать программные прерывания для косвенного вызова функций. Это особенно удобно, когда ядро кода содержит базовые подпрограммы, используемые приложениями. В этом случае вы можете изменять функции ядра, а также их место расположения без повторной компиляции каждого приложения.

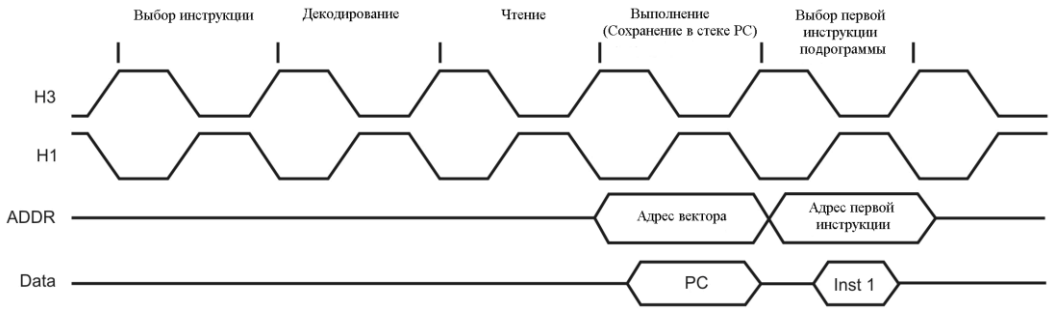


Рисунок 7.1 – Временная диаграмма выполнения инструкции CALL

7.4 Прерывания

ИС 1867ВМ9Ф поддерживает несколько внутренних и внешних прерываний, которые могут использоваться в различных приложениях. Внутренние прерывания генерируются сопроцессором ПДП, таймерами, коммуникационными портами. Пять внешних сигналов прерываний включают 4 внешних маскируемых прерывания IIOF3#-IIOF0# и одно немаскируемое прерывание NMI#. Сигналы прерываний могут посылаться к ЦПУ и сопроцессору ПДП.

Приоритеты прерываний в ИС 1867ВМ9Ф устанавливаются автоматически. Это позволяет одновременно обслуживать поступившие сигналы прерываний в предопределенном порядке.

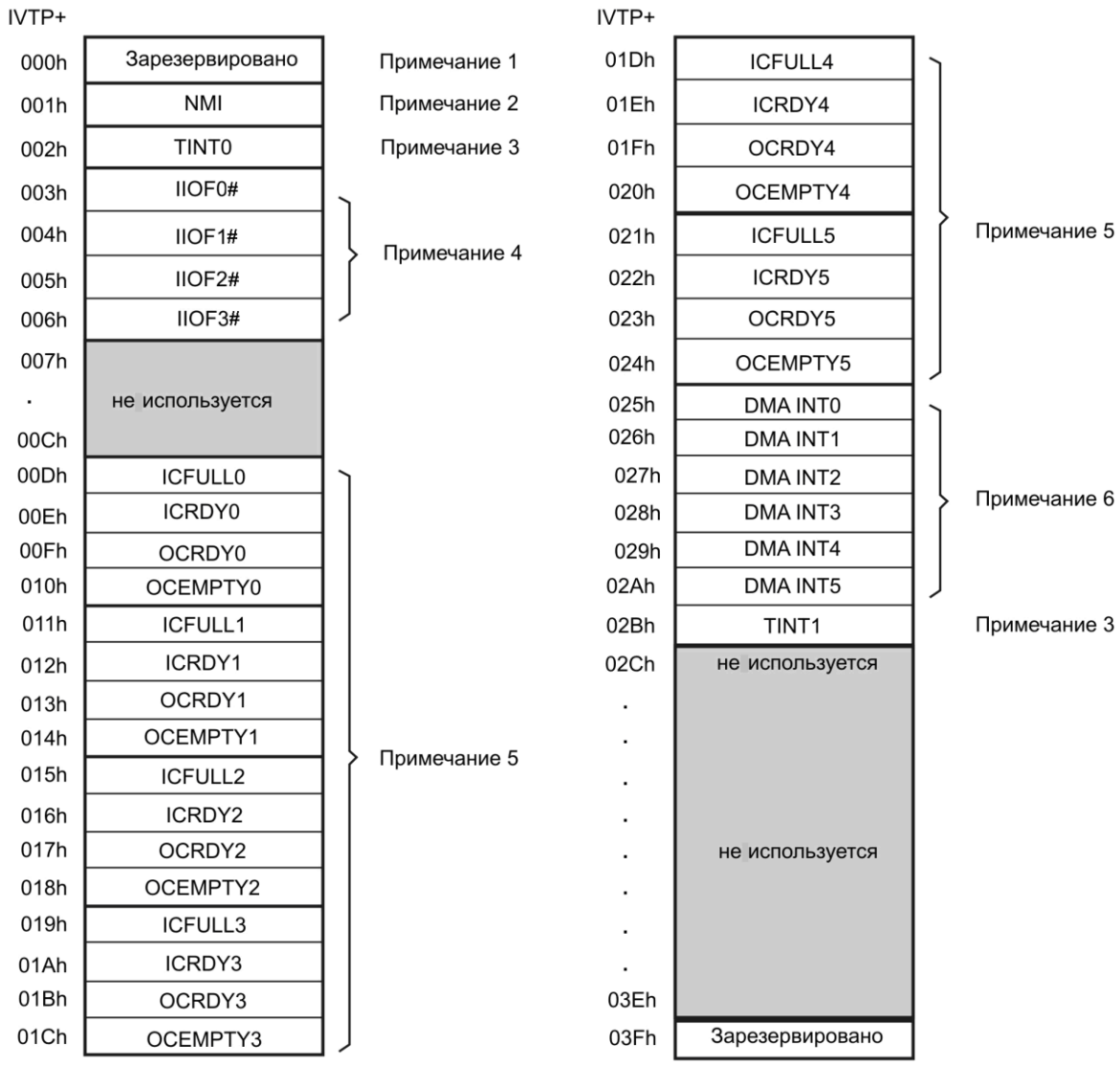
В данном подразделе обсуждается действие данных прерываний.

7.4.1 Таблица векторов системных прерываний и приоритеты

Таблица векторов системных прерываний IVT изображена на рисунке 7.2. Вектор прерывания – это адрес программы, обслуживающей прерывание и которая начинает выполняться при получении соответствующего сигнала прерывания. Таблица IVT должна размещаться в адресном пространстве памяти размером в 512 слов. Расположение таблицы определяется значением регистра IVTP, см. подраздел 3.2.

Приоритеты означают, что прерывания в верхней позиции таблицы векторов прерываний обслуживаются раньше, чем в нижней позиции, в случае, если они произошли в одном машинном цикле или когда два ранее полученных прерывания ожидают обработки. Однако это не означает, что, например, IIOF3# должен ожидать, пока будут обработаны IIOF2#, IIOF1#, IIOF0# (когда ST(GIE) = 1).

Приоритеты прерываний устанавливаются ЦПУ в соответствии с их позициями в таблице векторов прерываний, начиная со старшего (т. е. приоритет NMI старше, чем TINT0, который, в свою очередь, старше IIOF0# и т. д.). Следует заметить, что прерывание TINT0 располагается в IVTP+2, в то время как TINT1 – в IVTP+2Bh после прерываний коммуникационных портов и сопроцессора ПДП.



Примечания

1 «Зарезервировано» – зарезервировано для вектора сброса, см. таблицу 7.4.

2 NMI (немаскируемое прерывание) описано в 7.4.5.

3 Прерывания таймеров TINT0 и TINT1 разрешаются с помощью регистра IIE, см. 3.1.9, и отражаются в регистре IIF, см. 3.1.10.

4 Внешние сигналы IIOF3#-IIOF0# программируются в IIF регистре, смотри 3.1.10.

5 Прерывания по переполнению/очистке/готовности входных/выходных буферов коммуникационного порта разрешаются регистром IIE.

6 Прерывания от ПДП разрешаются регистром IIE и битами TCC и AUXTCC регистра управления каналом ПДП.

Рисунок 7.2 – Таблица векторов системных прерываний

7.4.2 Биты, управляющие прерыванием ЦПУ

Следующие регистры ЦПУ и биты в регистрах служат для управления прерыванием ЦПУ:

- Регистр состояния ЦПУ ST. Бит GIE регистра состояния ST – (глобальное) разрешение всех прерываний ЦПУ. Он управляет всеми маскируемыми прерываниями ЦПУ. Если бит установлен в «1», то прерывания ЦПУ разрешены. Если бит сброшен в «0», то все прерывания ЦПУ запрещены (исключая NMI, немаскируемое прерывание), см. 3.1.7.

- Регистр разрешения внутренних прерываний IIE. Регистр IIE используется для разрешения внутренних системных прерываний от конкретных устройств (от таймеров, коммуникационных портов и каналов ПДП), смотри 3.1.9.

- Регистр флагов IIOF IIF. Регистр IIF содержит биты флагов прерываний и биты для определения функции внешних сигналов на выводах IIOF3#-IIOF0#.

7.4.2.1 Регистр IIF

Регистр IIF содержит флаги прерываний. Когда происходят внешние прерывания или большинство внутренних прерываний, то соответствующие биты в регистре IIF устанавливаются в «1». Только внутренние прерывания от коммуникационного порта не имеют флагов в регистре IIF.

Когда ЦПУ обрабатывает прерывание, которое имеют флаг в регистре IIF, или когда контроллер ПДП фиксирует прерывание по внутреннему сигналу ПДП, то данный бит флага сбрасывается внутренним сигналом подтверждения прерывания в «0». Однако, для прерываний по уровню, если IIOF3#-IIOF0# остается в низком уровне, когда приходит сигнал подтверждения прерывания, бит флага прерывания сбрасывается в «0» только на один машинный цикл, а затем снова устанавливается в «1». По этой причине теоретически возможно, что при чтении регистра IIF, бит флага прерывания может быть равен «0», даже если IIOF3#-IIOF0# в низком (активном) уровне.

После сброса, в регистр флагов прерываний записывается «0», сбрасывая при этом все необслуженные прерывания.

Чтение/запись битов регистра IIF может осуществляться программно. Это обеспечивает доступ к сигналам IIOF3#-IIOF0#, которые могут быть определены как входы/выходы общего назначения или как сигналы прерываний. Например, если в регистре IIF FUNCx=0 (вход/выход) и TYPEx=1 (выход), тогда при записи в бит FLAGx, можно установить внешние выводы IIOF3#-IIOF0# в соответствующее состояние.

Если FUNCx=1 (прерывание), то запись «1» в бит FLAGx регистра IIF будет равнозначна появлению соответствующего прерывания. Поэтому все прерывания могут быть установлены и/или сброшены программно. Так как биты прерываний могут быть прочитаны, то они могут быть опрошены программно и в этом случае управление прерываниями не требуется.

Внутренние прерывания обрабатываются аналогично. В регистре IIF бит, соответствующий внутреннему прерыванию (например, TINT0, TINT1) может быть прочитан или записан программно. Запись «1» устанавливает защелку прерывания, запись «0» сбрасывает его. Все внутренние прерывания имеют продолжительность в 1 цикл H1/H3. Если требуется перед изменением IIF сохранить предыдущее значение неких битов регистра IIF, то изменять регистр следует с помощью логических операций AND, OR и т.д.

На рисунке 7.3 приведено правильная и не правильная модификация регистра IIF.

Правильно	Неправильно
LDI @MASK, R0	LDI IIF, R1
AND R0, IIF	AND @MASK, R1
	LDI R1, IIF

Рисунок 7.3 – Изменение регистра IIF

7.4.3 Описание процесса прерывания

Чтобы произошло прерывание, необходимо 2 условия:

- Все прерывания должны быть разрешены глобально установкой бита GIE в «0» регистра состояния ST.

- Конкретное прерывание должно быть разрешено установкой соответствующего бита в регистре IIE.

Цикл обработки прерывания, см. рисунок 7.4, включает в себя несколько событий. Соответствующий флаг прерывания в регистре IIF сбрасывается, предыдущие значения битов GIE и CF регистра состояния сохраняются, КЭШ «блокирован» (CF=1), прерывания глобально запрещаются (GIE=0) и ЦПУ завершает выборку инструкций. Затем выбирается вектор прерывания и помещается в PC, ЦПУ продолжает обработку с первой инструкции программы обработки прерывания ISR.

Когда используется инструкция RETIcond или RETIcondD для возврата из подпрограммы обработки прерывания, то предыдущие значения GIE и CF восстанавливаются.

Если необходимо, чтобы программа обработки прерываний не прерывалась, нужно установить GIE в «1» после входа в ISR, также можно включить и КЭШ.

Нужно иметь в виду, что биты PGIE и PCF регистра состояния могут хранить только одно предыдущее значение GIE и CF.

Примечание – GIE и CF сначала сохраняются, а затем загружаются новыми значениями после завершения выполнения последней инструкции, которая была выбрана перед произошедшим прерыванием. Это гарантирует дальнейшее корректное восстановление значения флагов.

Прерывания ЦПУ (включая NMI) подтверждаются только между выборкой инструкций. Если выборка инструкций остановлена в связи с конфликтом конвейера или когда выполняется цикл RPTS, прерывания ЦПУ не подтверждаются до следующей выбранной инструкции.

Инструкция подтверждения прерывания IACK может применяться для внешнего уведомления, что прерывание обслуживается. Если в операнде указана внешняя память, то IACK управляет сигналом IACK# и выполняет фиктивное чтение. Чтение выполняется из адреса, указанного операндом инструкции IACK. IACK обычно располагается в начале программы обслуживания прерывания. Однако в зависимости от вашего приложения, она может располагаться и в конце программы обслуживания прерывания или в другом месте. Кроме того, вы не обязаны использовать инструкцию IACK в программе обработки прерывания.

На рисунке 7.4 показан процесс обработки прерывания ЦПУ.

Следует обратить внимание на следующие ситуации:

- Прерывания запрещены в течение выполнения инструкции RPTS и в течение задержанного перехода (до завершения выполнения следующих за переходом трех инструкций). Прерывания удерживаются до завершения перехода.

- Когда происходит прерывание, то фазы декодирования и чтения инструкций продолжаются.

- Эти ситуации не относятся к инструкции, находящейся в фазе выборки инструкции. Если прерывание произошло в первом цикле выборки инструкции, то выбранная инструкция пропускается (не выполняется), а ее адрес заталкивается в стек, или если прерывание произошло после первого цикла выборки в случае многоцикловой выборки до состояния ожидания, то инструкция выполняется, а адрес следующей за ней инструкции заталкивается в стек, или если выборка инструкций в

текущий момент времени не происходит, то новая выборка инструкции не осуществляется.

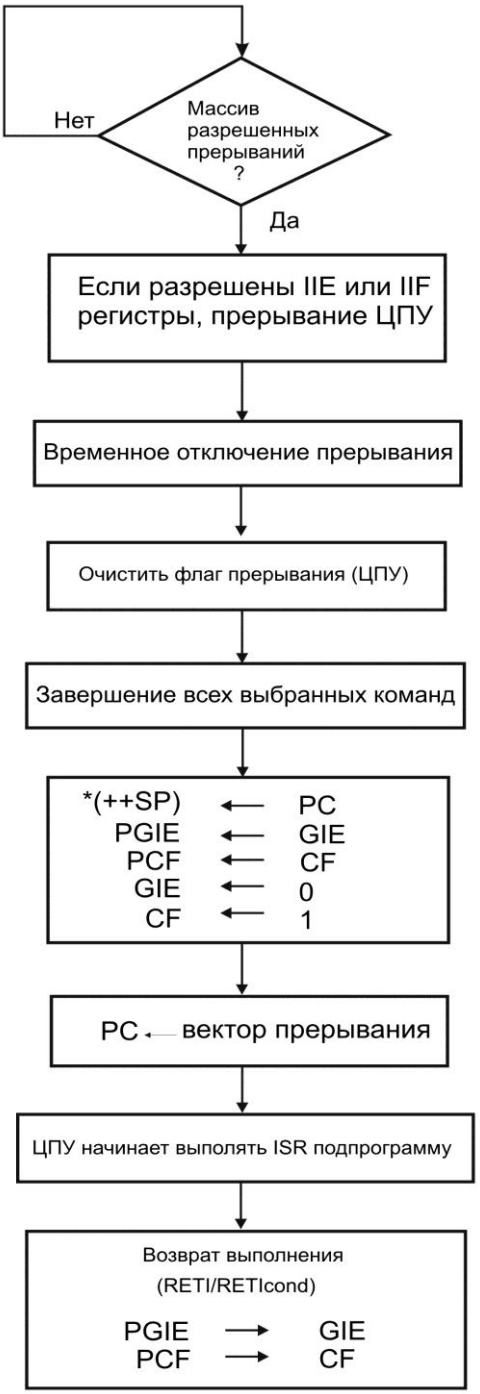


Рисунок 7.4 – Выполнение прерывания ЦПУ

7.4.4 Время запаздывания прерывания ЦПУ

Время запаздывания прерывания, определяемое как время от подтверждения прерывания до выполнения первой инструкции программы обработки прерывания ISR, занимает максимально 8 машинных циклов. Это описывается в таблице 7.2, где прерывание обрабатывается как инструкция, в предположении, что все инструкции одноцикловые.

Таблица 7.2 – Время запаздывания прерывания

Цикл	Описание	Выборка	Декодирование	Чтение	Выполнение
1	Распознавание прерывания в одноцикловой выборке (прог а+1) инструкции	прог а+1	прог а	прог а–1	прог а–2
2	Временное запрещение прерывания до сброса GIE. Сброс соответствующего флага IIF (если применимо)	–	прерывание	прог а	прог а–1
3	Чтение векторной таблицы прерываний	–	–	прерывание	прог а
4	Сохранение адреса в стек, бита GIE в PGIE и бита CF в PCF. Сброс GIE и установка CF в 1	–	–	–	прерывание
5	Конвейер начинает заполняться инструкциями	isr1	–	–	–
6		isr2	isr1	–	–
7		isr3	isr2	isr1	–
8	Выполнение первой инструкции программы обработки прерывания	isr4	isr3	isr2	isr1

7.4.5 Внешние прерывания

Пять внешних сигналов прерывания включают в себя 4 внешних маскируемых прерывания IIOF3#-IIOF0# и одно немаскируемое NMI#.

Четыре внешних асинхронных маскируемых прерывания разрешаются регистром IIF, см. 3.1.10, и синхронизируются внутри ИС. Они определяются по срезу H1 и проходят через серию внутренних задержек H1/H3. Будучи синхронизованным с тактовым сигналом H1, вход прерывания устанавливает соответствующий бит флага в регистре прерываний IIF.

Внешние прерывания и соответствующие им вектора прерываний:

IIOF# прерывание	Расположение адреса вектора прерывания
IIOF0#	IVTP + 003h
IIOF1#	IVTP + 004h
IIOF2#	IVTP + 005h
IIOF3#	IVTP + 006h

Приоритеты прерываний определяются по старшинству в случае прихода двух прерываний в одном машинном цикле (IIOF0# – старший, IIOF1# – следующий и т.д.). Если получено прерывание, то бит регистра состояния ST(GIE) сбрасывается в «0», запрещая любые другие входящие прерывания, исключая прерывание от NMI. Это предотвращает предполагаемое программное управление любыми другими прерываниями (IIOF3#-IIOF0#) до того момента, пока ST(GIE) будет опять установлен в «1». В дополнение к этому бит ST(GIE) сохраняется в ST(PGIE) и ST(CF) сохраняется в ST(PCF). После возврата из программы обработки прерывания, инструкции RETI и RETIcond помещают значение ST(PGIE) в ST(GIE) и ST(PCF) в ST(CF), тем самым, устанавливая их в прежние значения.

Внешние прерывания могут устанавливаться либо по фронту, либо по уровню в зависимости от полей TYPE, установленных в регистре IIF, см. 3.1.10.

Для прерываний по фронту внешние сигналы должны изменяться из «1» в «0», а затем должны удерживаться в низком уровне как минимум в течение одного цикла H1/H3.

Для прерываний по уровню внешние сигналы должны удерживаться в низком уровне в течение 1 или 2 циклов (1 ≤ время низкого уровня ≤ 2). Если прерывание удерживается в низком уровне более 2 циклов, то оно может быть определено как многократное прерывание, в этом случае нет необходимости обеспечивать фронт сигнала.

Примечание – Прерывания по уровню не защелкиваются, поэтому процессор их фиксирует, если низкий уровень присутствует в течение фаз выборки-декодирования инструкций в конвейере. Это означает, что если конвейер находится в режиме ожидания, то прерывания по уровню могут быть пропущены, даже если удерживаются в низком уровне в течение 1, 2 циклов. Это замечание не относится к прерываниям по фронтам, так как они являются фиксируемыми (они определяются, даже если конвейер остановлен).

Немаскируемое прерывание NMI не маскируется битом ST(GIE). Хотя NMI – немаскируемое прерывание, но процесс его обработки временно задерживается в течение выполнения задержанных переходов или многоцикловых операций ЦПУ. NMI – фиксируемое прерывание, определяемое по переднему и заднему фронтам. Необходимо быть осторожными при использовании NMI в качестве прерывания второго уровня.

Когда ИС 1867ВМ9Ф обслуживает прерывание, то другие прерывания запрещаются за исключением NMI. Это представляет проблему, так как в регистр состояния ST может быть записано неверное значение, если NMI произойдет перед первой инструкцией ISR (программы обработки прерываний), при которой предыдущее значение ST регистра сохраняется.

ИС 1867ВМ9Ф программно конфигурируется, позволяя ускорить готовность внутренней периферийной шины, когда подтверждается сигнал NMI#. «NMI разрешение передачи по шине» включается, если разряды 18 и 19 регистра состояния ST установлены в «10_2». Если это разрешение включено, то сигнал разрешения передачи по периферийной шине генерируется по заднему фронту NMI#. Если NMI# подтверждается, но не разрешена передача, то ЦПУ задерживает доступ к периферийной шине, если шина не готова. Данное условие возникает, если происходит запись в полный выходной буфер FIFO или чтение из пустого входного буфера FIFO.

Эта функция полезна при коррекции ошибок коммуникационного порта, когда он используется в совокупности с его программным сбросом.

7.5 Программные прерывания

Программные и системные прерывания в ИС 1867ВМ9Ф обрабатываются одинаково, за исключением их инициализации.

7.5.1 Инициализация программных и системных прерываний

Программные и системные прерывания инициализируются по-разному.

Программные прерывания всегда вызываются программно с помощью инструкций TRAPcond и LATcond.

Системные прерывания всегда вызываются аппаратно, например, внешними прерываниями, прерываниями ПДП или коммуникационного порта.

По этой причине бит GIE в регистре состояния ST и маскирующие биты в IIE не применяются к программным прерываниям.

7.5.2 Операции при программных прерываниях

На рисунке 7.5 изображен процесс программного прерывания, а также системных прерываний.

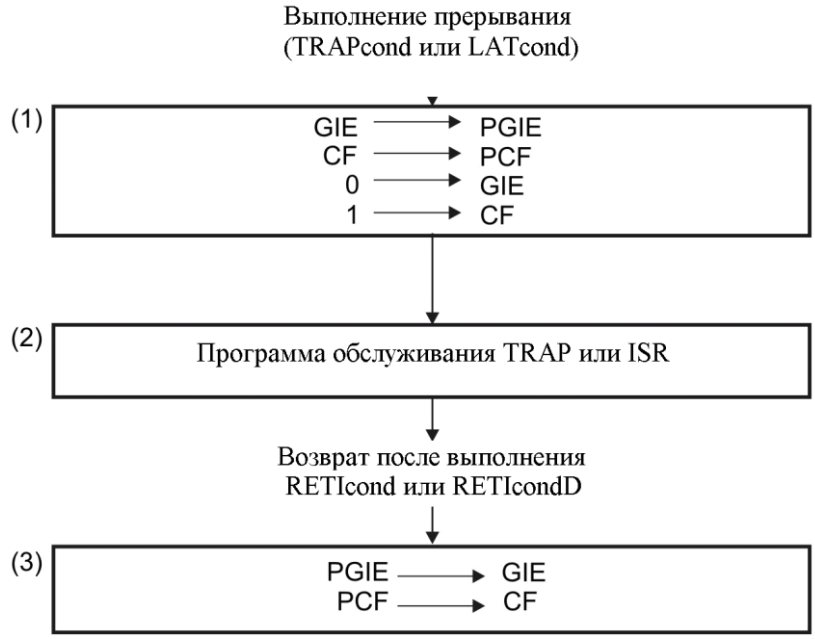


Рисунок 7.5 – Процесс программного прерывания

Инструкции RETIcond и RETIcondD управляют флагами состояния как показано в блоке (3) на рисунке 7.5. RETIcond/RETIcondD обеспечивают возврат/задержанный возврат из программного прерывания.

В основном, не требуется напрямую изменять биты PGIE или PCF регистра состояния за исключением ситуации, когда значение регистра состояния помещается в стек при вложенных системных или программных прерываниях.

ИС 1867ВМ9Ф поддерживает 512 различных программных прерываний. Когда выполняются инструкции TRAPcond n или LATcond n, то ЦПУ переходит по адресу, хранящемуся в области памяти, указанной TVTP+n, где TVTP – регистр-указатель таблицы векторов программных прерываний. Регистр TVTP – это 32-разрядный регистр, который содержит начальный адрес таблицы векторов прерываний. Эта таблица показана на рисунке 7.6.

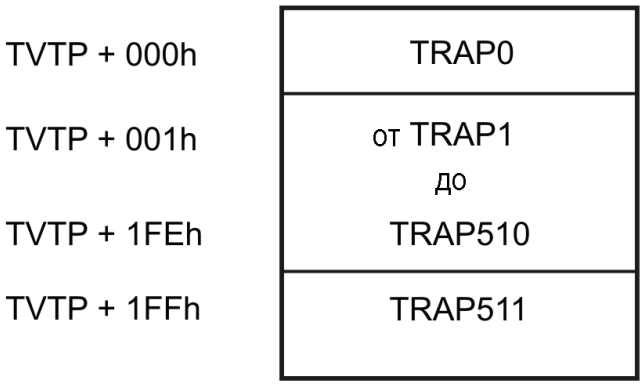


Рисунок 7.6 – Таблица программных прерываний

Как и таблица системных прерываний IVT, таблица программных прерываний TVT занимает область памяти в 512 слов. Регистр-указатель TVT указывает на начало таблицы TVT, см. подраздел 3.2 «Дополнительный регистровый файл ЦПУ».

Инструкции TRAP или LATcond могут использоваться для генерации программных прерываний и управления флагами состояния, как показано на рисунке 7.5. Инструкция LATcond обеспечивает одноцикловую «ловушку», которая полезна для нахождения и исправления ошибок.

Примечание – Так как LATcond – задержанная инструкция, то три инструкции, следующие за ней, не должны изменять биты GIE и CF регистра состояния.

7.5.3 Перекрытие таблиц программных и системных прерываний

Таблицы программных и системных прерываний могут разделять между собой одно и то же пространство памяти, размером в 512 байт. В этом случае векторы программных прерываний должны располагаться там, где нет векторов системных прерываний. Например, если вектор 02Ch не используется, то на этом месте можно разместить вектор программного прерывания в IVTP+02Ch (который, в свою очередь, является также TVTP+02Ch, если таблицы перекрываются), а затем вызвать программное прерывание, указав 02Ch в инструкции TRAP.

7.6 Прерывания сопроцессора ПДП

Прерывания могут вызывать операции чтения и записи ПДП. Это называется синхронизацией ПДП. Цикл обработки прерывания ПДП аналогичен циклу прерывания ЦПУ. После сброса соответствующего флага прерывания, сопроцессор ПДП работает в соответствии с битами SYNC регистра глобального управления сопроцессора ПДП.

Если прерывание разрешено регистром разрешения прерываний ПДП DIE, то контроллер прерываний автоматически фиксирует его и сохраняет для дальнейшего использования ПДП. Что же касается флагов прерываний (таймера, внешнего прерывания), то флаги IIF сбрасываются, когда контроллер прерываний фиксирует прерывание, но не тогда, когда ПДП отвечает на него. Даже если ПДП не был включен, фиксирование прерываний происходит, исключая тот случай, если стартовые биты START (AUX START) регистра управления ПДП сброшены в 00_2. Системный сброс ПДП сбрасывает внутренние зафиксированные прерывания.

7.6.1 Биты управления прерываниями ПДП

Два регистра содержат биты управления операциями прерывания ПДП:

- Регистр разрешения прерываний ПДП DIE. Все прерывания ПДП управляются битами регистра DIE и битами регистра SYNC управления каналом ПДП, смотри рисунок 11.2. Прерывания ПДП не зависят от ST(GIE) и локализованы в сопроцессоре ПДП.

- Регистр управления каналом ПДП. Каждый канал сопроцессора ПДП использует регистр управления каналом для определения режима работы. Этот регистр изображен на рисунке 11.2.

Регистр DIE разбит на 6 полей, которые определяют, какое прерывание будет использовано для управления синхронизацией каждого из 6 каналов ПДП.

Например, биты в каждом поле позволяют выбрать, будет ли ПДП синхронизован с коммуникационным портом или с таймером, или с внешним сигналом прерывания.

7.6.2 Процесс прерывания ПДП

Рисунок 7.7 показывает процесс прерывания, выполняемый сопроцессором ПДП. Для более подробной информации см. подраздел 11.10 «Прямой доступ к памяти и прерывания».

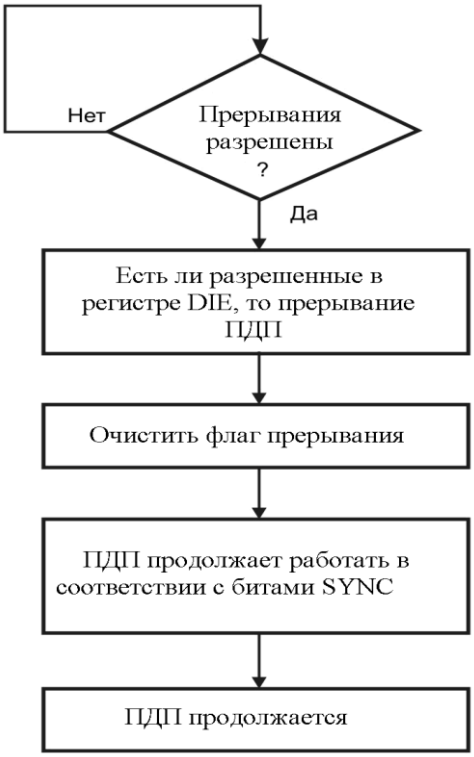


Рисунок 7.7 – Процесс прерывания ПДП

7.6.3 Взаимодействие прерываний ЦПУ/ПДП

Сопроцессор ПДП в ИС 1867ВМ9Ф не затрагивается обработкой прерываний ЦПУ, даже когда ПДП использует прерывания для синхронизации передачи. Кроме этого, ПДП не затрагивается, даже когда конвейер инструкций остановлен.

ИС 1867ВМ9Ф позволяет ЦПУ и сопроцессору ПДП и отвечать, и обрабатывать прерывания параллельно. Рисунок 7.8 показывает последовательность событий при обработке прерываний для ЦПУ и сопроцессора ПДП; точная последовательность событий приведена в таблице 7.2.

Таким образом, возможно прервать ЦПУ и сопроцессор ПДП одновременно одинаковыми или разными прерываниями и в сущности их синхронизировать. Однако так как сопроцессор ПДП и ЦПУ делят между собой установку флагов прерываний, то в некоторых случаях сопроцессор ПДП может сбросить флаг прерывания до того, как ЦПУ ответит на него. Например, прерывания ЦПУ отключены или если выборка инструкций остановлена, а сопроцессор ПДП может зафиксировать прерывание и сбросить соответствующий флаг. Если прерывание разрешено в регистре DIE, то ЦПУ никогда не перехватит прерывание ПДП, потому что ПДП отвечает на прерывание, либо с такой же скоростью, как и ЦПУ, либо быстрее.

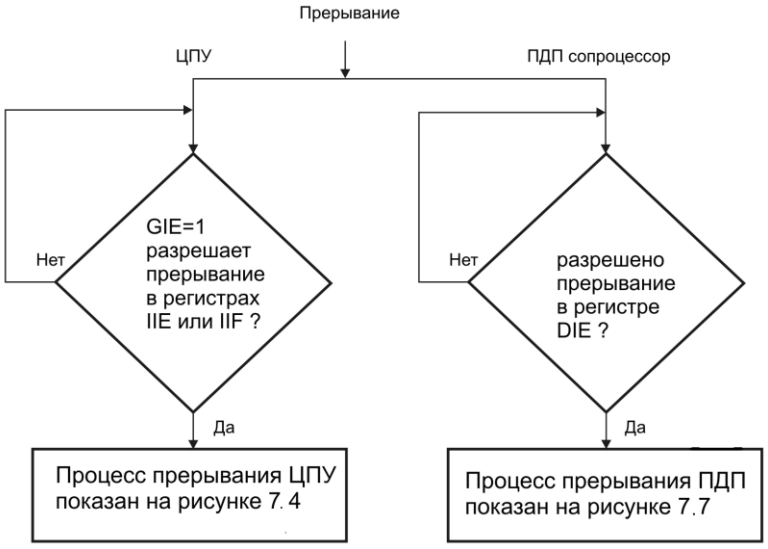


Рисунок 7.8 – Параллельная обработка прерываний ЦПУ и ПДП

7.7 Системный сброс

ИС 1867ВМ9Ф поддерживает немаскируемый внешний сигнал системного сброса RESET#, который используется для выполнения системного сброса регистров ЦПУ, сопроцессора ПДП и периферийных устройств. Данный подраздел описывает операцию системного сброса.

После включения питания состояние блоков ИС 1867ВМ9Ф не определено. Для установки блоков ИС в известное состояние можно использовать сигнал RESET#. Сигнал должен быть установлен в низком уровне в течение 10_{10} или более циклов H1 для обеспечения гарантированного системного сброса всех блоков.

Сигнал H1 – выходной тактовый сигнал, генерируемый ИС 1867ВМ9Ф.

Сброс влияет:

- на состояние некоторых выводов ИС,
- на состояние некоторых регистров ИС,
- на выполнение программы.

7.7.1 Состояние выводов после системного сброса

Системный сброс влияет синхронно или асинхронно на состояние выводов устройства. Синхронный сброс происходит посредством внутренних тактов от генератора ИС. Асинхронный сброс напрямую воздействует на состояние выводов, и он быстрее, чем синхронный сброс.

Таблица 7.3 показывает состояние выводов в течение RESET#=0 и после того, как RESET# опять установится в «1». Каждый сигнал описан в соответствии с тем, синхронно происходит сброс или асинхронно.

Таблица 7.3 – Состояние выводов после системного сброса

Сигнал	Число выводов	Тип §	Тип †	Описание
а) Тактовые сигналы (4 вывода) и вывод управления осциллятором				
H1	1	O	S	Начинает тактировать, когда RESET# изменяется из 1 в 0
H3	1	O	S	Начинает тактировать, когда RESET# изменяется из 1 в 0
X1	1	O	-	Не влияет
X2/CLKIN	1	I	-	Не влияет
б) Интерфейс коммуникационного порта 0 (12 выводов)				
C0D(7-0)	8	I/O	S	Устанавливается в неопределенное значение
CACK0#	1	I/O/Z	A	Устанавливается в Z, когда сброс находится в низком уровне, а затем устанавливается в 1, когда сброс становится высоким уровнем
CRDY0#	1	I/O/Z	A	Устанавливается в Z
CREQ0#	1	I/O/Z	A	Устанавливается в Z
CSTRB0#	1	I/O/Z	A	Устанавливается в Z, когда сброс находится в низком уровне, а затем устанавливается в 1, когда сброс становится высоким уровнем
в) Интерфейс коммуникационного порта 1 (12 выводов)				
C1D(7-0)	8	I/O	S	Устанавливается в неопределенное значение
CACK1#	1	I/O/Z	A	Устанавливается в Z, когда сброс находится в низком уровне, а затем устанавливается в 1, когда сброс становится высоким уровнем
CRDY1#	1	I/O/Z	A	Устанавливается в Z
CREQ1#	1	I/O/Z	A	Устанавливается в Z
CSTRB1#	1	I/O/Z	A	Устанавливается в Z, когда сброс находится в низком уровне, а затем устанавливается в 1, когда сброс становится высоким уровнем
г) Интерфейс коммуникационного порта 2 (12 выводов)				
C2D(7-0)	8	I/O	S	Устанавливается в неопределенное значение
CACK2#	1	I/O/Z	A	Устанавливается в Z, когда сброс находится в низком уровне, а затем устанавливается в 1, когда сброс становится высоким уровнем
CRDY2#	1	I/O/Z	A	Устанавливается в Z
CREQ2#	1	I/O/Z	A	Устанавливается в Z
CSTRB2#	1	I/O/Z	A	Устанавливается в Z, когда сброс находится в низком уровне, а затем устанавливается в 1, когда сброс становится высоким уровнем
д) Интерфейс коммуникационного порта 3 (12 выводов)				
C3D(7-0)	8	I/O/Z	S	Устанавливается в Z
CACK3#	1	I/O/Z	A	Устанавливается в Z
CRDY3#	1	I/O/Z	A	Устанавливается в Z, когда сброс находится в низком уровне, а затем устанавливается в 1, когда сброс становится высоким уровнем
CREQ3#	1	I/O/Z	A	Устанавливается в Z, когда сброс находится в низком уровне, а затем устанавливается в 1, когда сброс становится высоким уровнем
CSTRB3#	1	I/O/Z	A	Устанавливается в Z
е) Интерфейс коммуникационного порта 4 (12 выводов)				
C4D(7-0)	8	I/O/Z	S	Устанавливается в Z

Продолжение таблицы 7.3

Сигнал	Число выводов	Тип §	Тип †	Описание
CACK4#	1	I/O/Z	A	Устанавливается в Z
CRDY4#	1	I/O/Z	A	Устанавливается в Z, когда сброс находится в низком уровне, а затем устанавливается в 1, когда сброс становится высоким уровнем
CREQ4#	1	I/O/Z	A	Устанавливается в Z, когда сброс находится в низком уровне, а затем устанавливается в 1, когда сброс становится высоким уровнем
CSTRB4#	1	I/O/Z	A	Устанавливается в Z
ж) Интерфейс коммуникационного порта 5 (12 выводов)				
C5D(7-0)	8	I/O/Z	S	Устанавливается в Z
CACK5#	1	I/O/Z	A	Устанавливается в Z
CRDY5#	1	I/O/Z	A	Устанавливается в Z, когда сброс проходит в низком уровне, а затем устанавливается в 1, когда сброс проходит в высоком уровне
CREQ5#	1	I/O/Z	A	Устанавливается в Z, когда сброс проходит в низком уровне, а затем устанавливается в 1, когда сброс проходит в высоком уровне
CSTRB5#	1	I/O/Z	A	Устанавливается в Z
и) Эмуляция (7 выводов)				
EMU0	1	I/O	-	Не определено
EMU1	1	I/O	-	Не определено
TCK	1	I	-	Не влияет
TDI	1	I	-	Не влияет
TDO	1	O	-	Не влияет
TMS	1	I	-	Не влияет
TRST#	1	I	-	Не влияет
к) Внешний интерфейс глобальной шины (80 выводов)				
A(30-0)	31	O/Z	S	Устанавливается в Z
AE#	1	I	-	Не влияет
CE0#	1	I	-	Не влияет
CE1#	1	I	-	Не влияет
D(31-0)	32	I/O/Z	S	Устанавливается в Z
DE#	1	I	-	Не влияет
LOCK#	1	O	S	Устанавливается в 1
PAGE0	1	O/Z	S	Устанавливается в 0
PAGE1	1	O/Z	S	Устанавливается в 0
RDY0#	1	I	-	Не влияет
RDY1#	1	I	-	Не влияет
RD/WR0#	1	O/Z	S	Устанавливается в 1
RD/WR1#	1	O/Z	S	Устанавливается в 1
STAT(3-0)	4	O	S	Все устанавливаются в 1
STRB0#	1	O/Z	S	Устанавливается в 1
STRB1#	1	O/Z	S	Устанавливается в 1
л) Внешний интерфейс локальной шины (80 выводов)				
LA(30-0)	31	O/Z	S	Устанавливается в Z
LAE#	1	I	-	Не влияет
LCE0#	1	I	-	Не влияет
LCE1#	1	I	-	Не влияет

Продолжение таблицы 7.3

Сигнал	Число выводов	Тип §	Тип †	Описание
LD(31-0)	32	I/O/Z	S	Устанавливается в Z
LDE#	1	I	-	Не влияет
LLOCK#	1	O	S	Устанавливается в 1
LPAGE0/1	1	O/Z	S	Устанавливается в 0
LRDY0#	1	I	-	Не влияет
LRDY1#	1	I	-	Не влияет
LRD/WR0#	1	O/Z	S	Устанавливается в 1
LRD/WR1#	1	O/Z	S	Устанавливается в 1
LSTAT(3-0)	4	O	S	Все устанавливаются в 1
LSTRB0#	1	O/Z	S	Устанавливается в 1
LSTRB1#	1	O/Z	S	Устанавливается в 1
м) Прерывания, флаги ввода/вывода, сброс, таймер (12 выводов)				
IACK	1	I	S	Устанавливается в 1
IIOF0#-IIOF3#	4	I/O/Z	A	Устанавливается в Z
NMI#	1	I	-	Не влияет
RESET#	1	I	-	Вход RESET#
RESETLOC1	1	I	-	Не влияет
RESETLOC0	1	I	-	Не влияет
ROMEN	1	I	-	Не влияет
TCLK0	1	I/O/Z	A	Устанавливается в Z
TCLK1	1	I/O/Z	A	Устанавливается в Z

Примечания

1 Принятые условные обозначения:

- в графе «Тип †» – A – асинхронный, S – синхронный;
- в графе «Тип §» – I – вход, O – выход, Z – состояние высокого импеданса.

2 Рекомендуемые развязывающие конденсаторы, подключаемые вокруг ИС, должны быть кратными 0,1 мкФ и 4,7 мкФ. Их количество зависит от уровня шумов на плате.

7.7.2 Размещение вектора системного сброса

После осуществления сброса ИС начинает выполнение программы. Начальный адрес программы хранится в векторе сброса. ИС позволяет выбрать один из четырех адресов вектора сброса. Выборка нужного адреса осуществляется посредством установки соответствующих уровней на выводы RESETLOC1 и RESETLOC0 при сбросе. В таблице 7.4 показаны их возможные конфигурации.

Таблица 7.4 – Размещение вектора RESET#

RESETLOC1	RESETLOC0	Получить вектор сброса из памяти по адресу, hex	Примечание
0	0	00000 0000	Локальная шина
0	1	07FFF FFFF	Локальная шина
1	0	08000 0000	Глобальная шина
1	1	0FFFF FFFF	Глобальная шина

7.7.3 Дополнительные операции системного сброса

После системного сброса (после того, как RESET# установился из «0» в «1»), выполняются следующие дополнительные операции:

- Устанавливаются регистры таймеров – регистр глобального управления таймером устанавливается в «0», исключая бит DATIN, который принимает значение сигнала TCLK; счетчик таймера и регистр периода устанавливаются в «0».
- Регистры управления коммуникационными портами 0-2 устанавливаются в «0» (состояние выхода), регистры управления коммуникационными портами 3-5 устанавливаются в 04h (состояние входа).
- Регистры управления внешней памятью устанавливаются в 3E39_FFF0h (7 состояний ожидания).
- Регистр управления каналом ПДП, счетчик передач ПДП и вспомогательный счетчик передач ПДП устанавливаются в «0».
- Регистры ЦПУ: IIE, IIF, DIE, IVTP, TVTP – устанавливаются в «0».
- Регистр состояния ЦПУ ST устанавливается в 0400h, при этом КЭШ становится «блокирован».
- Вектор сброса считывается из памяти по соответствующему адресу и загружается в PC.
- Если ROMEN=1 (внутреннее ПЗУ включено), RESETLOC1, RESETLOC0 в низком уровне и IIOF0# вывод находится в высоком уровне, то ИС 1867ВМ9Ф начинает выполнять код загрузчика, иначе ИС 1867ВМ9Ф начинает выполнение программы с адреса, указанного вектором сброса в соответствии с RESETLOC1, RESETLOC0. Многопроцессорные системы на базе ИС 1867ВМ9Ф могут сбрасываться и синхронизироваться по одному и тому же тактовому сигналу.

8 Операции конвейера

Две особенности, которые повышают производительность ИС 1867ВМ9Ф, – это конвейеризация и параллельное выполнение операций ЦПУ и операций ввода/вывода.

Четыре функциональных блока управляют работой ИС – это блоки выборки, декодирования, чтения и выполнения. Конвейеризация – это перекрытие или параллельное выполнение в одно и то же время нескольких стадий выполнения базовой инструкции (выборка инструкции, ее декодирование, чтение операндов и выполнение).

Выполняя операции ввода/вывода, сопроцессор ПДП освобождает ЦПУ от этих операций и тем самым снижает нагрузку на ЦПУ, увеличивая вычислительную мощность ИС в целом.

Основные темы, обсуждаемые в данном разделе:

- структура конвейера;
- конфликты конвейера;
- конфликты переходов;
- конфликты регистров;
- конфликты памяти;
- разрешение конфликтов конвейера;
- синхронизация обращений к памяти;
- выборка программы;
- загрузка и сохранение данных;
- обращения ПДП.

8.1 Структура конвейера

Конвейер ИС состоит из четырех функциональных блоков:

- блок выборки F. Выбирает слово инструкции из памяти и изменяет счетчик инструкций PC;

- блок декодирования D. Декодирует слово инструкции и выполняет генерацию адреса. Также этот блок управляет любыми изменениями вспомогательных регистров и указателем стека;

- блок чтения R. Если требуется, то считывает операнд из памяти;

- блок выполнения E. Если требуется, то считывает операнд из регистрового файла, выполняет необходимые операции и записывает результат в регистровый файл. Если требуется, то результат предыдущей операции записывает в память.

Базовые инструкции исполняются на четырех уровня конвейера – это выборка, декодирование, чтение и исполнение. На рисунке 8.1 иллюстрируются эти уровни в структуре конвейера. Уровни пронумерованы в соответствии с циклом выполнения инструкции и исполнения. Полное перекрытие в конвейере, где все четыре элемента работают параллельно, возникает на цикле (m). Те уровни, которые должны быть почти выполнены на m+1, уже выполнены на уровне m-1. Устройство управления конвейером обеспечивает высокоскоростное выполнение каждым блоком своих функций за один машинный такт. Устройство также управляет конфликтами конвейера таким образом, что они не видны для пользователя, поэтому не возникает необходимости предпринимать каких-либо мер для гарантии корректного выполнения операции.

Цикл	F	D	R	E	
m-3	W	-	-	-	
m-2	X	W	-	-	
m-1	Y	X	W	-	
m	Z	Y	X	W	←полное перекрытие
m+1	-	Z	Y	X	
m+2	-	-	Z	Y	
m+3	-	-	-	Z	

Примечание – W, X, Y, Z – представляют исполняемые инструкции. F, D, R, E = выборка, декодирование, чтение и выполнение, соответственно.

Рисунок 8.1 – Структура конвейера ЦПОС

Приоритеты от старшего к младшему, присвоенные для каждого из функциональных элементов и сопроцессора ПДП, следующие:

- ПДП (если сконфигурирован как высокоприоритетный);
- исполнение;
- чтение;
- декодирование;
- выборка;
- ПДП (если сконфигурирован как низкоприоритетный).

Когда процесс обработки инструкции готов к переходу на следующий более высокий уровень конвейера, но этот уровень не готов к приему новой инструкции, то возникает конфликт конвейера. В этом случае элемент нижнего уровня ждет, пока устройство с высшим приоритетом завершит свою текущую функцию.

Конфликты ПДП с ЦПУ могут быть минимизированы путем соответствующего структурирования данных, так как сопроцессор ПДП имеет собственную шину данных и адреса.

8.2 Конфликты конвейера

Конфликты конвейера в ИС могут быть разделены на следующие три типа:

- конфликты переходов**.** Включают большинство инструкций или операций, которые читают и/или изменяют PC;
- конфликты регистров**.** Включают задержки, которые могут возникнуть при чтении/записи в регистры и используются для генерации адреса;
- конфликты памяти**.** Возникают, когда внутренние устройства ИС состязаются за ресурсы памяти.

Каждый из этих трех типов конфликтов описан в следующих подразделах с поясняющими примерами. В этих примерах необходимо обратить внимание на повторную выборку или повторение операции, при этом символ, представляющий стадию конвейера, дополняется номером. Например, если выборка выполняется снова, то мнемоника инструкции повторяется. Когда обращение задерживается на несколько циклов из-за отсутствия готовности, то используются символы ~RDY и RDY для указания отсутствия или наличия готовности, соответственно.

8.2.1 Конфликты переходов

Первый тип конфликтов конвейера возникает при стандартных (незадержанных) переходах, т. е. BR, Bcond, DBcond, CALL, IDLE, RPTB, RPTS, RETIcond, RETScond, прерываниях и сбросе. Конфликты возникают при выполнении этих инструкций и операциях, т. к. в течение их исполнения конвейер используется только для завершения операции; другая информация, находящаяся в конвейере, отбрасыва-

ется или перевыбирается, или конвейер становится неактивным. Это называется «очисткой» конвейера. «Очистка» конвейера необходима в этих случаях, чтобы предотвратить частичное исполнение последующих инструкций. TRAPcond и CALLcond классифицируются отдельно от других типов переходов и рассматриваются позже.

В примере 8.1 приведена программа и работа конвейера для стандартного перехода: выполняется одна пустая выборка MPYF и после того, как доступен адрес перехода, выполняется новая выборка (инструкция OR). Эта пустая выборка вносит инструкцию MPYF в КЭШ.

Пример 8.1 – Стандартное ветвление

```
        BR  THREE       ; Безусловный переход
        MPYF            ; Не выполняется
        ADD             ; Не выполняется
        SUBF            ; Не выполняется
        AND             ; Не выполняется
          .
          .
THREE   OR              ; Выбрана инструкция OR после выборки BR
        STI
                        .
                        .
```

Работа конвейера:

PC	F	D	R	E
n	BR	-	-	-
n+1	MPYF	BR	-	-
n+1	(nop)	(nop)	BR	-
n+1	(nop)	(nop)	(nop)	BR
THREE	OR	(nop)	(nop)	(nop)
	STI	OR	(nop)	(nop)

THREE → PC Выборка задержана для нового значения PC

Примечание – Обе инструкции RPTS и RPTB очищают конвейер, обеспечивая возможность загрузки регистров RS, RE и RC в соответствующее время работы конвейера. Если эти регистры загружены без использования RPTS и RPTB, то не возникает «очистки» конвейера. Если не используется ни один из режимов повторения, то RS, RE и RC могут быть использованы как 32-разрядные регистры общего назначения без возникновения конфликтов конвейера. В таких случаях, как вложенность RPTB, обусловленные вложенными прерываниями, необходимо загрузить и сохранить эти регистры прямо во время использования режима повторения.

Поскольку может быть выбрано до четырех инструкций перед введением режима повторения, то эти загрузки должны предшествовать очистке конвейера. Если RC изменяется командой, то прямая загрузка имеет более высокий приоритет, чем логика повторов.

Задержанные переходы используются для гарантии выборки следующих трех инструкций. Задержанные переходы включают следующие инструкции: BRD, BcondD и DBcondD. В примере 8.2 приведены программа и работа конвейера для задержанного перехода.

Пример 8.2 – Задержанный переход

```
BRD   THREE         ; Безусловный задержанный переход
```

```
MPYF                ; Выполняется
ADD                 ; Выполняется
SUBF                ; Выполняется
AND                 ; Не выполняется
          .
          .
THREE  MPYF         ; Выбирается после выборки SUBF
          .
          .
```

Работа конвейера:

PC	F	D	R	E	
n	BRD	-	-	-	
n+1	MPYF	BRD	-	-	задержка не выполняется
n+2	ADDF	MPYF	BRD	-	
n+3	SUBF	ADDF	MPYF	BRD	
THREE	MPYF	SUBF	ADDF	MPYF	

THREE → PC

8.2.2 Конфликты при обращении к регистрам

Конфликты регистров представляют собой чтение или запись в регистры, которые используются для адресации. Эти конфликты возникают, когда соответствующий регистр не готов для использования. Некоторые условия, при которых можно избежать конфликтов регистров, обсуждаются в подразделе 8.3.

Регистры образуют три функциональных группы:

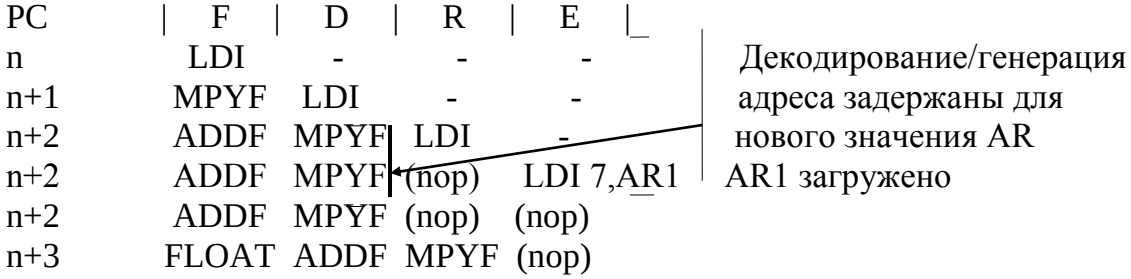
- группа 1. Вспомогательные регистры (AR0-AR7), индексные регистры (IR0, IR1) и регистр размера блока (BK);
- группа 2. Указатель страницы данных (DP);
- группа 3. Указатель системного стека (SP).

Если инструкция производит запись в одну из этих групп, то блок декодирования не может использовать любой регистр из одной и той же группы до завершения в него записи, т.е. до завершения выполнения инструкции. В примере 8.3 вспомогательный регистр загружается, а другой вспомогательный регистр используется для следующей инструкции. Т.к. на стадии декодирования необходимо значение, которое получается в результате записи во вспомогательный регистр, то декодирование этой второй инструкции задерживается на два цикла. Каждый раз при задержке декодирования выполняется повторная выборка кода инструкции, т. е. инструкция ADDF выбирается три раза. Поскольку они являются реальными выборками, то они могут привести не только к конфликтам с сопроцессором ПДП, но и к КЭШ-совпадению и КЭШ-несовпадению.

Пример 8.3 – Запись в AR с последующей генерацией адреса AR

```
LDI   7, AR1                ; 7 → AR1
NEXT  MPYF  *AR2, R0        ; Декодирование задержано на 2 цикла
ADDF
FLOAT
```

Работа конвейера:



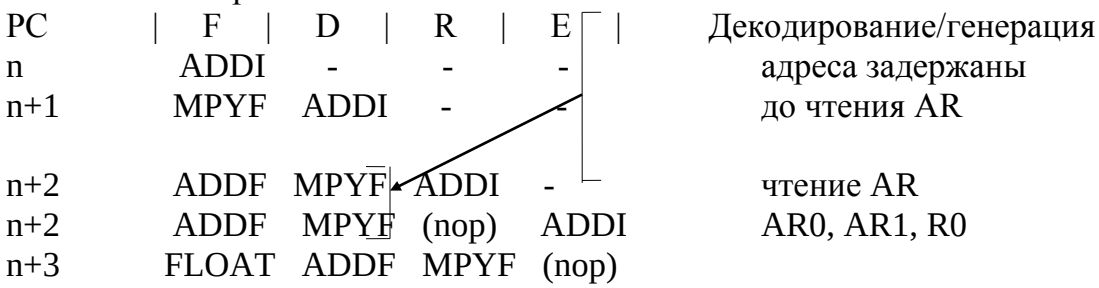
Чтение для этих групп аналогично записи. Если инструкция должна считать значение в одном из элементов группы, то использование регистра из этой же группы при декодировании следующей инструкции задерживается до завершения чтения. Регистры считываются в начале цикла исполнения, поэтому требуется только один цикл задержки последующего декодирования. Для регистров IR0, IR1, BK или DP задержка не возникает. Для всех остальных, включая SP, задержка возникает.

В примере 8.4 производится сложение содержимого двух вспомогательных регистров с записью в регистр расширенной точности. Следующая инструкция использует другой вспомогательный регистр как адресный.

Пример 8.4 – Чтение AR с последующим использованием AR для генерации адреса.

```
ADDI AR0, AR1, R1              ; AR0 + AR1 → R1
NEXT    MPYF *++AR2,R0              ; Декодирование задерживается на 1 цикл
ADDF
FLOAT
```

Работа конвейера:



Инструкции DBR (декремент и ветвление) используют вспомогательные регистры в качестве счетчиков цикла, что аналогично использованию их для адресации. Следовательно, операции, приведенные в вышеописанных примерах, также присутствуют и при выполнении этих инструкций.

8.2.3 Конфликты при обращении к памяти

Конфликты памяти могут возникать при превышении границы физической памяти. Блоки СОЗУ 0, СОЗУ 1 и ПЗУ могут поддерживать только 2 обращения за один цикл. Внешний интерфейс может поддерживать только одно обращение каждый цикл. Некоторые условия, при которых можно избежать конфликтов памяти, обсуждаются в подразделе 8.4.

Конфликты конвейера при работе с памятью состоят из следующих четырех типов:

- ожидание программы. Выборка кода инструкции предотвращена;
- выборка программы не завершена. Выборка кода инструкции начата, но еще не завершена;

- только выполнение**.** Последовательность инструкций требует трех обращений за один цикл к данным со стороны ЦПУ;

- задержка всего. Операция на глобальной или локальной шине должна быть завершена до начала любой другой операции.

Эти четыре типа конфликтов памяти описаны и проиллюстрированы примерами ниже.

8.2.3.1 Ожидание программы

Ожидание выполнения программой происходит в следующих случаях.

1 случай. Происходит большее число обращений к одному и тому же блоку памяти, чем он может позволить:

- происходит два обращения ЦПУ за данными во внутренних блоках ОЗУ или ПЗУ, и необходима выборка программы из тех же блоков;

- ЦПУ начинает доступ к данным, находящимся в одном из внешних портов и требуется выборка инструкции из этого же порта.

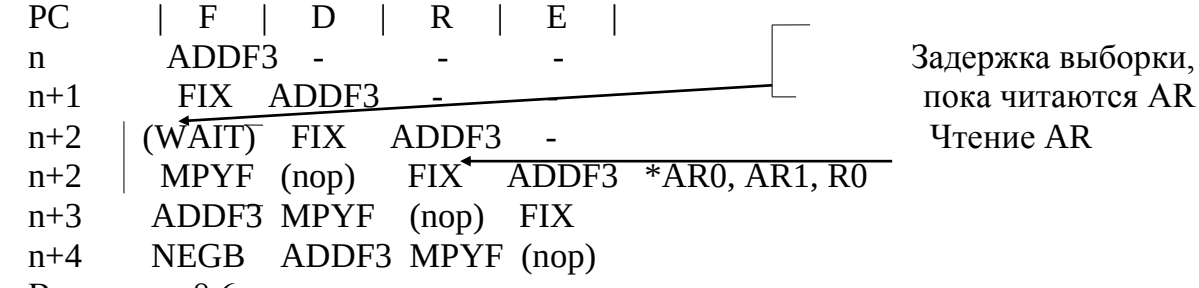
2 случай. Требуется многоцикловое обращение ЦПУ или сопроцессора ПДП к данным через внешнюю шину.

В примере 8.5 иллюстрируется ожидание завершения обращения ЦПУ к данным. В этом случае *AR0 и *AR1 указывают на блок СОЗУ 0, а инструкция MPYF также выбирается из блока СОЗУ 0. Это приводит к конфликту. Так как может быть произведено не более двух обращений к блоку СОЗУ 0 за один цикл, то выборка программы не может быть начата и должна ждать завершения доступа ЦПУ к данным.

Пример 8.5 – Программа ожидает завершения обращения ЦПУ к данным.

```
ADDF3   *AR0, *AR1, R0
FIX
MPYF
ADDF3
 NEGB
```

Работа конвейера:



В примере 8.6 приведено программное ожидание, связанное с многоцикловым обращением «данные-данные» или обращением ПДП. Инструкции ADDF, MPYF и SUBF выбраны из некоторой другой области памяти, чем требуется для ПДП.

Сопроцессор ПДП начинает многоцикловое обращение. Выборка программы, относящаяся к CALL, делается по той же внешней шине, которую использует ПДП. Даже если сопроцессор ПДП имеет самый низкий приоритет, то его многоцикловое обращение не может быть прервано. Выборка программы должна ожидать завершения многоциклового обращения сопроцессора ПДП к данным.

Пример 8.6 – Ожидание программы в связи с многоцикловым обращением сопроцессора ПДП

Работа конвейера:

PC	F	D	R	E
n	ADDF	-	-	-
n+1	MPYF	ADDF	-	-
n+2	SUBF	MPYF	ADDF	-
n+3	(WAIT)	SUBF	MPYF	ADDF
n+3	CALL	(nop)	SUBF	MPYF
n+4	-	CALL	(nop)	SUBF


2-цикловое обращение ПДП

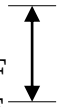
8.2.3.2 Выборка инструкции не завершена

Незавершенность выборки инструкции возникает, когда выборка слова инструкции требует более одного цикла для завершения из-за включенных состояний ожидания. В примере 8.7 инструкции MPYF и ADDF выбраны из памяти, которая поддерживает одноцикловое обращение, а инструкция SUBF выбирается из памяти, требующей одного состояния ожидания. Один пример, демонстрирующий этот конфликт, – выборка через границу банка в порте глобальной шины.

Пример 8.7 – Многоцикловые выборки из памяти, где находится программный код

Работа конвейера:

PC		F	D	R	E
n		MPYF	-	-	-
n+1		ADDF	MPYF	-	-
n+2	RDY	SUBF	ADDF	MPYF	-
n+2	RDY	SUBF	(nop)	ADDF	MPYF
n+3		ADDI	SUBF	(nop)	ADDF


требуется 1 состояние ожидания

8.2.3.3 Конфликт при выполнении

Тип конфликта конвейера «Конфликт при выполнении» возникает, когда последовательность инструкций требует трех обращений ЦПУ к данным в одном цикле. Имеется три случая, при которых возникает данный конфликт:

- инструкция выполняет сохранение, а затем следует инструкция, которая должна выполнить два чтения памяти;
- инструкция выполняет два сохранения, а затем следует инструкция, которая должна выполнить, по меньшей мере, одно чтение из памяти.

Первый случай приведен в примере 8.8. Поскольку эта последовательность инструкций требует трех обращений к памяти данных, а поддерживается только два, то конвейером выполняется только фаза исполнения. Двойные чтения, требуемые LDF||LDF, задерживаются на один цикл. Обратите внимание на то, что может возникнуть двойная выборка следующей инструкции.

Пример 8.8 – Два чтения следуют за одиночным сохранением.

```
STF     R0, *AR1        ; R0 → *AR1
LDF     *AR2, R1        ; *AR2 → R1     параллельно с
||  LDF     *AR3, R2    ; *AR3 → R2
```

Работа конвейера:

PC	F	D	R	E
n	STF	-	-	-
n+1	LDF||LDF	STF	-	-

n+2 W LDF||LDF STF - Запись должна завершиться
n+3 X W LDF||LDF STF до того, как смогут
n+4 X W LDF||LDF (nop) завершиться два чтения.
n+4 Y X W LDF||LDF

В примере 8.9 приводится параллельное сохранение, за которым следует одна запись или чтение. Т.к. требуются два параллельных сохранения, то следующее чтение данных ЦПУ должно ожидать один цикл до их начала. Может произойти одна двойная выборка памяти.

Пример 8.9 – Одно чтение следует за параллельным сохранением

```
STF   R0, *AR0      ; R0 → *AR0    параллельно с
||  STF   R2, *AR1    ; R2 → *AR1
ADDF @SUM, R1       ; R1 + @SUM → R1
IACK
ASH
```

Работа конвейера:

PC	F	D	R	E	
n	STF||STF	-	-	-	Чтение должно ожидать, пока
n+1	ADDF	STF||STF	-	-	завершатся записи
n+2	IACK	ADDF	STF||STF	-	
n+3	ASH	IACK	ADDF	STF||STF	
n+4	ASH	IACK	ADDF	(nop)	
n+4	-	ASH	IACK	ADDF	

8.2.3.4 Задержка всего

Существует три типа конфликтов конвейера при работе с памятью типа «задержка всего»:

- операция загрузки или сохранения ЦПУ не может быть выполнена из-за того, что занята внешняя шина;
- загрузка с внешней шины занимает более одного цикла;
- выполнение условных вызовов и программных прерываний, которые берут на один цикл больше, чем условные переходы.

Первый тип конфликта «Задержка всего» возникает, когда одна из внешних шин занята обращением, которое началось, но не завершилось. В примере 8.10 первое сохранение – двухцикловое. ЦПУ записывает данные во внешнюю шину. Управление шиной требует два цикла для завершения записи «данные-данные». Инструкция LDF читает через тот же внешний интерфейс. Так как сохранение не завершено, ЦПУ продолжает попытки выполнить LDF, пока шина не станет доступна.

Пример 8.10 – Занята внешняя шина

```
STF R0, @DMA1
LDF @DMA2, R0
```

Работа конвейера:

PC	F	D	R	E	
n	STF	-	-	-	
n+1	LDF	STF	-	-	
n+2	W	LDF	STF	-	

```
n+2          W      LDF    (nop)    STF          2-цикловое обращение
записи
n+2          W      LDF    (nop)    (nop)        к внешней шине
n+3          X      W      LDF      (nop)
n+4          Y      X      W        LDF
```

Второй тип конфликта «Задержка всего» представляет собой многоцикловое чтение данных. Чтение началось и продолжается до его завершения. В примере 8.11 выполняется инструкция LDF из внешней памяти, что требует нескольких циклов для своего завершения.

Пример 8.11 – Многоцикловое чтение данных

LDF @DMA, R0

Работа конвейера:

```
PC          |  F  |  D  |  R  |  E  |
n             LDF     -     -     -
n+1            I     LDF    -     -
n+2            J      I     LDF   -     2-цикловое обращение записи
n+3       K (пустой)  I     LDF   -      к внешней памяти
n+3           K2      J      I    LDF
```

Последний тип конфликта «Задержка всего» связан с условными вызовами и системными прерываниями, которые отличаются от выполнения других инструкций ветвления. Поскольку другие инструкции ветвления выполняют условное сохранение и условные вызовы, а системные прерывания также выполняют условное сохранение, что занимает на один цикл больше, чем условный переход, см. пример 8.12. Дополнительный цикл необходим для проталкивания адреса возврата после определения условия вызова.

Пример 8.12 – Условные вызовы и системные прерывания.

Работа конвейера:

```
PC              |   F    |    D    |    R    |    E    |
n             CALLcond      -         -         -
n+1              I       CALLcond     -         -
n+1            (nop)     (nop)    CALLcond     -
n+1            (nop)     (nop)      (nop)   CALLcond
n+1            (nop)     (nop)      (nop)   CALLcond      Цикл сохранения
n+2/CALLaddr   I         (nop)      (nop)      (nop)       PC
```

8.3 Разрешение конфликтов регистров

Если осуществляется обращение к вспомогательным регистрам AR11-AR0, индексным IR0, IR1 регистрам, указателю страницы данных DP, указателю стека SP с какой либо иной целью, кроме как для генерации адреса, то может возникнуть конфликт конвейера, связанный со следующим обращением к памяти. Конфликты конвейера и задержки описаны в 8.2.2.

Примеры с 8.13 по 8.15 демонстрируют использование этих регистров, которые не приводят к возникновению конфликта или способы их использования, с помощью которых можно избежать этих конфликтов.

Пример 8.13 – Изменение генерации адреса AR с последующей генерацией AR адреса

```
LDF 7.0,   R0          ;      7.0 → R0
```

```
MPYF  *++AR0(IR1), R0
ADDF  *AR2, R0
FIX
MPYF
ADDF
```

Работа конвейера:

PC	F	D	R	E	
n	LDF	-	-	-	
n+1	MPYF	LDF	-	-	Генерация адреса и изменение AR
n+2	ADDF	MPYF	LDF	-	Генерация адреса
n+3	FIX	ADDF	MPYF	LDF	
n+4	MPYF	FIX	ADDF	MPYF	
n+5	ADDF	MPYF	FIX	ADDF	

Пример 8.14 – Запись в AR с последующим использованием AR для генерации адреса без конфликта конвейера

```
LDI  @TABLE,  AR2
MPYF @VALUE,  R1
ADDF R2, R1
MPYF *AR2++, R1
SUBF
STF
```

Работа конвейера:

PC	F	D	R	E	
n	LDI	-	-	-	Нет генерации AR адреса для этих двух инструкций
n+1	MPYF	LDI	-	-	
n+2	ADDF	MPYF	LDI	-	AR2 использован для
n+3	MPYF	ADDF	MPYF	LDI 7, AR2	генерации адреса
n+4	SUBF	MPYF	ADDF	MPYF	AR2 загружен
n+5	STF	SUBF	MPYF	ADDF	

Пример 8.15 – Запись в DP с последующим прямым чтением памяти без конфликтов конвейера

```
LDP TABLE_ADDR
POP R0
LDF *-AR3(2), R1
LDI @TABLE_ADDR, AR0
PUSHF  R6
PUSH   R4
```

Работа конвейера:

PC	F	D	R	E	
n	LDP	-	-	-	
n+1	POP	LDP	-	-	
n+2	LDF	POP	LDP	-	DP загружен
n+3	LDI	LDF	POP	LDP	
n+4	PUSHF	LDI	LDF	POP	
n+5	PUSH	PUSHF	LDI	LDF	

8.4 Разрешение конфликтов памяти

Если выполняется выборка инструкции и обращение к данным таким образом, что ресурсы, которые будут использованы, не обеспечивают необходимое быстродействие, то выборка инструкции задерживается до завершения выборки данных. Определенные конфигурации размещения данных, программы и порядок обращения к памяти позволяют обеспечить максимальную производительность ИС 1867ВМ9Ф.

В таблице 8.1 приведены сведения о том, сколько выборок данных может быть произведено для разных областей памяти, в случае, когда необходимо произвести выборку инструкции и одиночное обращение к данным и при этом получить максимальную производительность (один цикл). Четыре варианта дают одноцикловое выполнение инструкций.

Таблица 8.1 – Одна выборка инструкции и одно обращение к данным с максимальной производительностью

Вариант	Обращения по глобальной шине	Выборки из внутренней памяти	Обращения по локальной или периферийной шине
1	1	1	–
2	1	–	1
3	–	2 из любой комбинации внутренней памяти	–
4	–	1	1

В таблице 8.2 показано, какое количество выборок может производиться из различных областей памяти, когда необходимо производить выборку инструкции и две выборки данных, с целью обеспечения максимальной производительности (один цикл). Шесть вариантов дают максимальную производительность.

Таблица 8.2 – Одна выборка инструкции и два обращения к данным с максимальной производительностью

Вариант	Обращения по глобальной шине	Выборки из внутренней памяти	Обращения по локальной или периферийной шине
1	1	2 из любой конфигурации памяти	–
2	1 программа	1 данные	1 данные
3	1 данные	1 данные	1 программа
4	1 данные	1 программа, 1 данные	1 ПДП
5	–	2 из одного блока внутренней памяти и одна из другого блока внутренней памяти	–
6	–	3 из разных блоков внутренней памяти	1 ПДП
7	–	2 из любой конфигурации памяти	1
8	1 программа	2 данные	1 ПДП
9	1 ПДП	2 данные	1 программа

8.5 Синхронизация доступа к памяти

В подразделе описывается организация доступа к памяти, внутренние фазы синхронизации H1 и H3 и их соотношения, см. рисунок 8.1, чтобы показать, каким образом процессор управляет несколькими обращениями к памяти. В то время как в подразделе 8.4 обсуждалось взаимодействие между последовательностями инструкций, то в этом разделе описывается поток данных на основе отдельной инструкции.

Каждый основной период синхронизации равный 12,5 нс состоит из двух меньших периодов синхросигнала по 6,25 нс, называемых H3 и H1 (предполагается, что частота, поступающая на вход ЦПУ от PLL равна 160 МГц).

Примечание – Активный период синхросигнала для H3 и H1 это время, когда сигнал находится в высоком уровне.

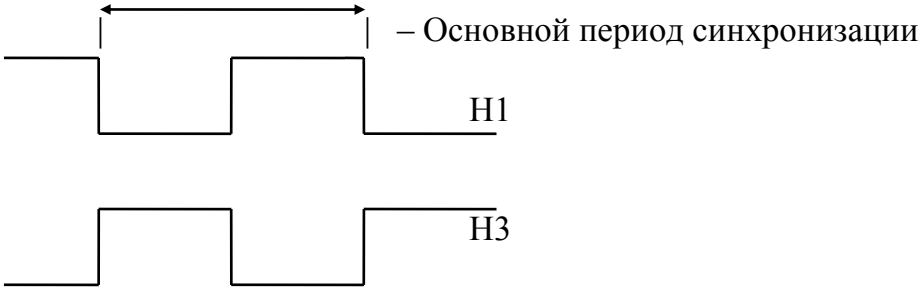


Рисунок 8.1 – Соотношение сигналов H3 и H1

Операции чтения и записи памяти могут быть определены в соответствии с этими меньшими периодами синхронизации. Типы операций памяти, которые могут произойти – это выборка инструкции, загрузка и сохранение данных и обращения сопроцессора ПДП.

8.5.1 Выборка инструкции

Внутренние выборки инструкции всегда выполняются в течение H3, если другая инструкция в конвейере не должна произвести одно сохранение данных в то же самое время. В этом случае выборка инструкции производится в течение H1, а сохранение данных в течение H3.

Внешние выборки программы всегда начинаются вначале H3 с адресом, представляемым на внешней шине. В конце H1 они завершаются с фиксированием слова инструкции.

8.5.2 Загрузка и сохранение данных

Загрузку, чтение и сохранение данных выполняют 4 типа инструкций: двухоперандные инструкции, трехоперандные инструкции, операции умножителя/АЛУ с инструкциями сохранения и инструкции параллельного умножения и сложения.

В разделе 6 «Методы адресации» более детально представлена информация о способах адресации.

Как описано ранее, число циклов шины для обращений к внешней памяти отличается в некоторых случаях от числа циклов исполнения ЦПУ. Для внешнего чтения число циклов шины и исполнения ЦПУ идентично. При внешней записи присутствует как минимум два цикла шины, но кроме случаев конфликта «доступ к шине» имеется только один цикл исполнения ЦПУ. В последующих примерах приведены различия в количестве циклов шины и ЦПУ.

8.5.2.1 Обращение к памяти двухоперандных инструкций

Двухоперандные инструкции включают все те инструкции, которые в разрядах 31-29 имеют значения 000_2 или 010_2, см. рисунок 8.2. В случае чтения данных разряды 15-0 являются операндом src. Внутреннее чтение всегда производится в течение H1. Внешние чтения данных всегда начинаются в начале H3 с адресом, выставляемым на внешней шине, и завершаются защелкиванием слова данных в конце H1.

В случае сохранения данных разряды 15-0 являются операндом dst. Внутренние данные сохраняются в течение H3. Внешняя запись данных (запись данных на внешней шине) всегда начинается вначале H3 с адресом и данными, выставляемыми на внешней шине.

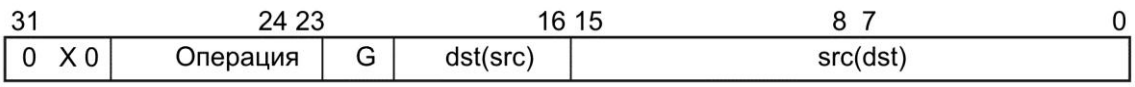


Рисунок 8.2 – Слово двухоперандной инструкции

8.5.2.2 Чтение памяти трехоперандной инструкцией

Трехоперандные инструкции включают все инструкции, разряды 31-29 у которых равны 001, см. рисунок 8.3. Операнды источников src1 и src2 поступают либо из регистра, либо из памяти. В случае, когда один или более операндов источника находятся в памяти, эти инструкции всегда выполняют чтение из памяти.

Если только один из операндов источника (либо src1, либо src2) расположен во внутренней памяти, то данные читаются в течение H1. Если один операнд источника находится во внешней памяти, то чтение данных начинается по H3 с адресом, выставляемым на внешней шине, и завершается с защелкиванием слова данных в конце H1.

Если оба операнда должны выбираться из памяти, то могут возникнуть различные варианты. Если оба операнда расположены во внутренней памяти, то чтение src1 производится в течение H3, а src2 – в течение H1, таким образом, завершая два чтения памяти в один цикл.

Если src1 во внутренней памяти, а src2 – во внешней, то выборка src2 начинается по началу H3 и защелкивается по концу H1. В то же самое время, выборка src1 из внешней памяти осуществляется в течение H3. Таким образом, опять получается два чтения памяти в один цикл.

Если src1 во внешней памяти, а src2 – во внутренней, то для завершения чтения требуются два цикла. В первом цикле производится внутренняя выборка src2. Выборка src1 также производится, но не фиксируется до следующего H3.

Если src1 и src2 оба во внешней памяти, то для завершения чтения требуются два цикла. В первом цикле производится выборка src1 и загрузка по следующему H3. Во втором цикле производится выборка src2 и загрузка по H1 того же цикла.

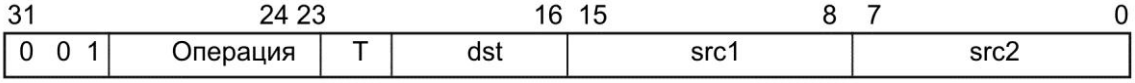


Рисунок 8.3 – Слово трехоперандной инструкции

8.5.2.3 Операции с параллельным сохранением

Следующий класс инструкций включает все инструкции, которые выполняют сохранение параллельно с другой инструкцией. Разряды 31 и 30 для этих инструкций равны 11_2.

Для тех операций, которые производят умножение или операцию АЛУ параллельно с сохранением, формат слова инструкции приведен на рисунке 8.4.

Если это операция сохранения для внешнего или внутреннего dst2, то сохранение выполняется в течение H3. Два цикла шины требуются для внешних сохранений, но только один цикл ЦПУ необходим для завершения записи.

Если операция чтения из памяти – внешняя, то она начинается в начале H3 и данные защелкиваются в конце H1. Если операция чтения памяти является внутренней, то она выполняется в течение H1. Необходимо помнить, что чтение памяти выполняется ЦПУ в течение фазы чтения (R) конвейера и сохранение выполняется в течения фазы исполнения (E).

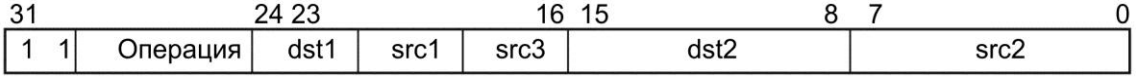


Рисунок 8.4 – Операция умножения или ЦПУ с параллельным сохранением

Формат слова инструкции для тех инструкций, в которых присутствуют параллельные сохранения в памяти, приведены на рисунке 8.5. Если оба операнда назначения dst1 и dst2, расположены во внутренней памяти, то dst1 сохраняется в

течение H3, а dst2 в течение H1, завершая, таким образом, два сохранения в памяти за один цикл.

Если dst1 – во внешней памяти, а dst2 – во внутренней, то сохранение dst1 начинается в начале H3. Сохранение dst2 во внутренней памяти выполняется в течение H1. Два сохранения в памяти завершаются за один цикл.

Для внешнего сохранения требуется два цикла, но только один цикл ЦПУ требуется для завершения записи. Если dst1 – во внутренней памяти, а dst2 – во внешней, то требуется дополнительный (экстра) цикл шины для завершения сохранения dst2. Для завершения записи требуется только один цикл ЦПУ, но доступ к шине требуется в течение трех циклов шины. В первом цикле выполняется внутреннее сохранение dst1 в течение H3, а dst2 записывается во внешнюю память в течение H1. В течение следующего цикла выполняется сохранение dst2 на внешней шине, начиная с H3, и выполняется как нормальная операция на шине в течение следующего цикла.

Если оба операнда dst1 и dst2 пишутся во внешнюю память, то по-прежнему требуется только один цикл ЦПУ для завершения сохранения. В этом случае требуется четыре цикла шины.

В первом цикле оба операнда (dst1 и dst2) пишутся в порт, и начинается доступ к внешней шине dst1.

Сохранение dst1 завершается во втором цикле и сохранение dst2 начинается в третьем цикле. Сохранение dst2 завершается на четвертом цикле внешней шины.

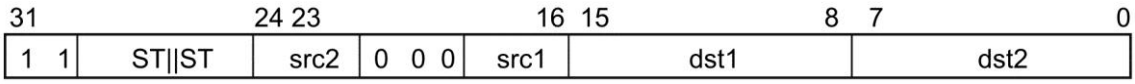


Рисунок 8.5 – Два параллельных сохранения

8.5.2.4 Параллельное умножение и сложение

Адресация памяти для параллельных умножений и сложений подобна адресации для трехоперандных инструкций. Параллельное умножение и сложение включает все инструкции, разряды 31-30 которых равны 10_2, см. рисунок 8.6.

Для этих операций операнды src3 и src4 расположены в памяти. Если оба операнда расположены во внутренней памяти, то src3 выполняется в течение H3, а src4 – в течение H1, таким образом, два чтения памяти завершаются в один цикл.

Если src3 – во внутренней памяти, а src4 – во внешней, то выборка src4 начинается в начале H3, а данные защелкиваются в конце H1. В то же самое время, обращение src4 во внутреннюю память выполняется в течение H3.

Если src3 – во внешней памяти, а src4 – во внутренней, то для завершения двух операций чтения требуется два цикла. В первом цикле выполняется внутренняя выборка src4. В течение H3 следующего цикла выполняется выборка src3.

Если оба src3 и src4 – во внешней памяти, для завершения двух чтений требуется два цикла. В первом цикле производится выборка src3, во втором цикле производится выборка src4.

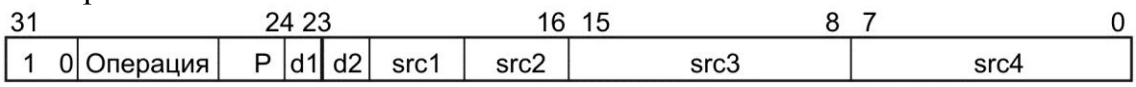


Рисунок 8.6 – Формат инструкций параллельного умножения и сложения

9 Операции внешней шины

ИС 1867ВМ9Ф имеет два идентичных интерфейса внешних шин. Одна шина называется интерфейсом глобальной памяти, а другая шина называется интерфейсом локальной памяти. Эти интерфейсы необходимы для увеличения пропускной способности микросхемы, для обеспечения одновременного обращения к данным в различные области внешней памяти.

Представленная в разделе информация описывает глобальный и локальный интерфейсы памяти, однако в некоторых подразделах описан только глобальный интерфейс памяти.

9.1 Общее описание

Микросхема 1867ВМ9Ф имеет два идентичных параллельных внешних интерфейса: интерфейс глобальной памяти и интерфейс локальной памяти. Каждый из интерфейсов имеет следующие особенности:

- раздельную конфигурацию, где каждый интерфейс имеет собственную 32-разрядную шину данных и 31-битную адресную шину;
- одно цикловое чтение и конвейерную запись;
- независимые сигналы разрешения для данных, адресов и управляющих шин;
- сигналы запроса шины и блокировки шины для реализации параллельных процессов в разделяемой памяти;
- управляемое пользователем распределение адресов для каждой из двух независимых стробов для различных скоростей памяти;
- предварительный просмотр сигналов состояния шины для определения текущей и запрашиваемой операций для организации параллельных процессов;
- выбираемые состояния ожидания (управляемые программно или аппаратно);
- сигналы, которые указывают пересечение границы страницы памяти.

Примечания

1 Интерфейс глобальной памяти идентичен интерфейсу локальной памяти, за исключением того, что они имеют различное расположение в карте памяти, и сигналы управления интерфейсом локальной памяти помечены дополнительным префиксом «L». В данном разделе не делается различий между сигналами локального и глобального интерфейсов, а также между сигналами STRB0# и STRB1#, за исключением случаев, когда это делается для большей ясности.

2 Сигналы, которые показывают, когда границы страниц пересекаются, поддерживают три типа памяти: память со страничной организацией и статическим декодированием столбцов – DRAM, высокоскоростную память – SRAM, низкоскоростную память и устройства ввода/вывода.

9.2 Сигналы интерфейса памяти

Как показано на рисунке 9.1, интерфейс глобальной памяти имеет два множества управляющих сигналов STRB0# и STRB1#. Регистры управления глобальной памятью, смотри подраздел 9.3, определяют, какой из сигналов активен.

Примечание – Сигналы на рисунке 9.1 отражают интерфейс глобальной памяти. Сигналы интерфейса локальной памяти имеют сходную конфигурацию и дополнительный префикс «L» к каждому сигналу, например, STRB0# становится LSTRB0# и т. д.

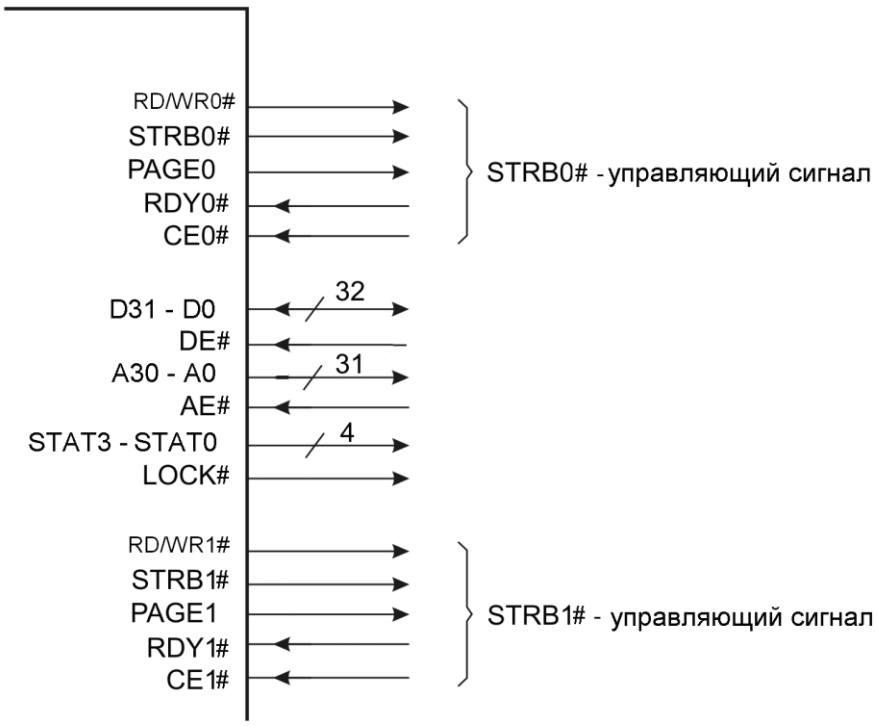


Рисунок 9.1 – Сигналы управления интерфейсами глобальной и локальной памяти

Таблица 9.1 – Сигналы интерфейса глобальной памяти

Сигнал*	Тип	Описание	Значение после сброса	Состояние Idle
AE#**		Сигнал разрешения адресной шины для интерфейса глобальной памяти. При высоком уровне (установлен в «1») устанавливает A30-A0 в высокоимпедансное состояние	Не влияет	Игнорируется
CE0#**, CE1#**		Сигнал разрешения для RD/WR0#, RD/WR1#, STRB0#, STRB1#, PAGE0, PAGE1 сигналов. При высоком уровне (установлен в «1») сигналы RD/WR0#, RD/WR1#, STRB0#, STRB1#, PAGE0, PAGE1 находятся в высоко-импедансном состоянии	Не влияет	Игнорируется
DE#**		Сигнал разрешения шины данных для интерфейса глобальной памяти. При высоком уровне (установлен в «1») D31-D0 находятся в высокоимпедансном состоянии. Чтение возможно, но запись не возможна	Не влияет	Игнорируется
LOCK#		Сигнал блокировки для интерфейса глобальной шины показывает, есть ли блокировка доступа: 0 – доступ разрешен, 1 – доступ запрещен. LOCK# изменяется только посредством инструкции блокировки	1	1

Окончание таблицы 9.1

Сигнал*	Тип	Описание	Значение после сброса	Состояние Idle
PAGE0, PAGE1	O/Z O/Z	Сигнал разрешения доступа к страницам памяти для STRB0#, STRB1#	0	0
RDY0#, RDY1#		Индикаторы готовности внешней памяти к доступу	Не влияет	Игнорируется
RD/WR0#, RD/WR1#	O/Z O/Z	Определяет чтение из памяти (высокий уровень) или запись в память (низкий уровень)	Все 1	Все 1
STAT3, STAT2, STAT1, STAT0	O O O O	Четыре сигнала, определяющие состояние или функцию порта памяти, как показано в таблице 9.2	Все 1	Все 1
STRB0#, STRB1#	O/Z O/Z	Строб доступа к интерфейсу	1	1
A30-A0	O/Z	Адресная шина. Она сохраняет адрес последнего доступа. Адресные сигналы всегда управляемы.	Hi-Z	Адрес последнего доступа
D31-D0	I/O/Z	Шина данных. Эти сигналы переходят в высокоимпедансное состояние между циклами записи	Hi-Z	Hi-Z

Примечания

1 Сигналы STAT3-STAT0 и LOCK# не управляются внешним сигналом управления.

2 Принятые условные обозначения: I – вход; O – выход, I/O – вход/выход, Z или Hi-Z – высокоимпедансное состояние.

_____

* Ноль в обозначении сигнала указывает на сигналы, управляемые STRB0#, а единица – на сигналы, управляемые STRB1#, как показано на рисунке 9.1.

** Этот сигнал может быть использован в системе, состоящей из ИС 1867ВМ9Ф и нескольких (одного или большего количества) активных устройств, способных управлять шиной для того, чтобы разделять общую шину доступа к памяти или периферии и удерживать ее до конца обмена.

В таблице 9.2 приведены значения сигналов STAT3-STAT0, определяющие текущее состояние порта (интерфейса) глобальной памяти. Эти сигналы дают информацию о доступе, который будет выполняться. Значение сигналов STAT3-STAT0 для инструкции SIGI полезно, чтобы отличить состояния шины между SIGI чтением и LDII или LDFI чтением.

Значение для «холостого хода» равно 1111_2 (приведено внизу таблицы 9.2). Это упрощает разделение шин в многопроцессорной системе, так как подтягивающие к «1» резисторы могут быть использованы для уведомления системе о бездействующем процессоре, находящемся в состоянии Idle, когда один из процессоров не присоединен к общей шине.

Таблица 9.2* – Состояние интерфейса глобальной памяти для сигналов STRB0# и STRB1#

Значение сигнала				Состояние
STAT3	STAT2	STAT1	STAT0	
0	0	0	0	STRB0# доступ, чтение программы
0	0	0	1	STRB0# доступ, чтение данных
0	0	1	0	STRB0# доступ, чтение ПДП
0	0	1	1	STRB0# доступ, чтение SIGI (инструкция)
0	1	0	0	Зарезервировано
0	1	0	1	STRB0# доступ, запись данных
0	1	1	0	STRB0# доступ, запись ПДП
0	1	1	1	Зарезервировано
1	0	0	0	STRB1# доступ, чтение программы
1	0	0	1	STRB1# доступ, чтение данных
1	0	1	0	STRB1# доступ, ПДП чтение
1	0	1	1	STRB1# доступ, чтение SIGI (инструкция)
1	1	0	0	Зарезервировано
1	1	0	1	STRB1# доступ, запись данных
1	1	1	0	STRB1# доступ, запись ПДП
1	1	1	1	Не активен

_____

*Данная таблица применима к обоим интерфейсам – глобальной памяти и локальной памяти. Для сигналов интерфейса локальной памяти добавляется префикс «L»: LSTAT3, LSTAT2 и т. д.

9.3 Регистры управления интерфейсом памяти

Рисунок 9.2 показывает карту памяти для регистров управления интерфейсами глобальной и локальной памяти. Рисунок 9.3 показывает поля в каждом регистре. Каждый регистр может быть запрограммирован для управления этими соответствующими интерфейсами памяти путем определения:

- размера страницы, использованной для двух стробов каждого порта;
- области адресов, в которой стробы активны;
- состояния ожидания;
- других операций управления интерфейсом памяти.

При сбросе двоичные значения, показанные для каждого бита и приведенные на рисунке 9.2, записываются в регистр управления интерфейсом глобальной памяти.

Значения разрядов 3-0 – это значения сигналов AE#, DE#, CE1# и CE0#. Сброс устанавливает следующую конфигурацию (для локальной и глобальной шин):

- Поля PAGESIZE для STRB0# (разряды 18-14) и STRB1# (разряды 23-19) устанавливаются в 00111_2, что соответствует 256 словам.

- Поля WTCNT для STRB0# (разряды 10-8) и STRB1# (разряды 13-11) устанавливаются в 111_2, которые соответствуют семи состояниям ожидания.

- Поле ACTIVE для STRB0# (разряды 28-24) устанавливается для всех адресов глобального (или локального для LSTRB0#) интерфейса памяти.

- Поле STRB SWITCH (бит 29) устанавливается в «1» для добавления цикла между чтением, которое переключает STRB0# в STRB1# или (STRB1# в STRB0#).

- Поля SWW для STRB0# (разряды 5, 4) и STRB1# (разряды 7, 6) устанавливаются в 11_2 для установки сигнала внутренней готовности, представляющего из себя

логическое «И» внешнего сигнала готовности READY (RDY_{int}#) и сигнала готовности, генерируемого внутренним счетчиком состояний ожидания (RDY_{wtcnt}).

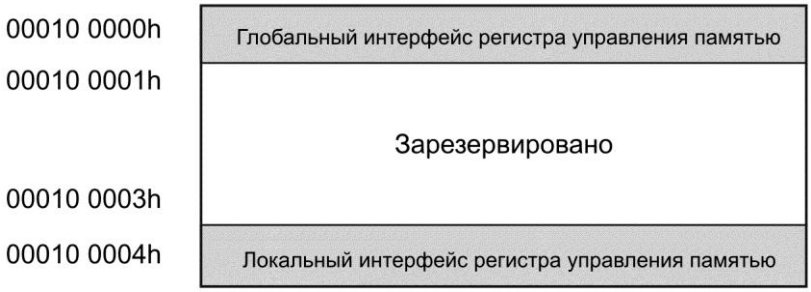
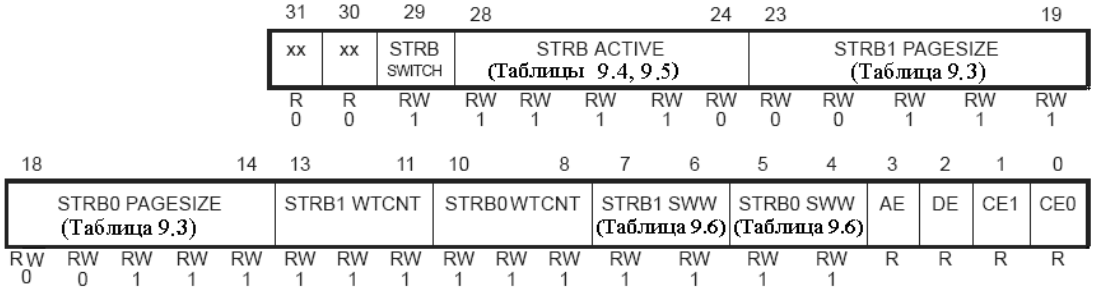


Рисунок 9.2 – Размещение регистров управления интерфейсом памяти



Примечания

1 В ячейке регистра содержится символ регистра управления интерфейсом глобальной памяти. Для символов регистра управления интерфейсом локальной памяти добавляется префикс «L» к каждому символу, т.е. LSTRB0 SWW, LCE0# и т. д.

2 «1» и «0» внизу каждого разряда – это двоичное значение, записываемое в регистр при сбросе. Значения разрядов 3-0 определены значениями соответствующих им внешних сигналов AE#, DE#, CE1# и CE0#.

3 Условно обозначены: RW – чтение/запись; R – чтение.

Рисунок 9.3 – Поля регистров управления интерфейсом памяти

Регистры управления интерфейсом памяти показаны в карте памяти на рисунках 4.1, 4.3.

Описание полей и битов регистра управления интерфейсом глобальной памяти:

CE0 Значение внешнего сигнала CE0# (после его прохождения через внутренний синхронизатор). Значение не фиксируется.

CE1 Значение внешнего сигнала CE1# (после его прохождения через внутренний синхронизатор). Значение не фиксируется.

DE Значение внешнего сигнала DE# (после его прохождения через внутренний синхронизатор). Значение не фиксируется.

AE Значение внешнего сигнала AE# (после его прохождения через внутренний синхронизатор). Значение не фиксируется.

STRB0 SWW Состояние ожидания программы для доступа STRB0#. Совместно с STRB0 WTCNT это поле определяет режим генерации состояния ожидания. Реальные состояния ожидания приведены в подразделе 9.4 и в таблице 9.6.

STRB1 SWW	Состояние ожидания программы для доступа STRB1#. Совместно с STRB0 WTCNT это поле определяет режим генерации состояния ожидания. Реальные состояния ожидания рассматриваются в подразделе 9.4 и в таблице 9.6.
STRB0 WTCNT	Программный счет состояний ожидания для доступов STRB0#. Определяет число циклов, используемых при включении программных состояний ожидания. Диапазон трех разрядов – от 000_2 (нуля) до 111_2 (семи).
STRB1 WTCNT	Программный счет состояний ожидания для доступов STRB1#. Определяет число циклов, используемых при включении программных состояний ожидания. Диапазон трех разрядов – от 000_2 (нуля) до 111_2 (семи).
STRB0 PAGESIZE	Размер страницы для STRB0# доступов. Определяет число старших разрядов адреса, используемых для определения размера банка для STRB0# доступов. Смотрите диапазоны в таблице 9.3 и 9.3.2.
STRB1 PAGESIZE	Размер страницы для STRB1# доступов. Определяет число старших разрядов адреса, используемых для определения размера банка для STRB0# доступов. Смотрите диапазоны в таблице 9.3 и 9.3.2.
STRB ACTIVE	Определяет диапазоны адресов, при которых STRB0# и STRB1# активны. Смотри диапазоны в таблице 9.4 для STRB ACTIVE и в таблице 9.5 для LSTRB ACTIVE.
STRB SWITCH	Вставляет один цикл между повторными обратными чтениями, который переключает STRB0# в STRB1# (или наоборот). Когда «1», то добавляется цикл. Когда «0», то цикл не добавляется.
Резерв	Читается как нули.

Примечание – Описание полей и битов остается таким же и для регистра управления интерфейсом локальной памяти, для которого добавляется префикс «L» к каждому символу, т. е. LCE0#, LCE1#, LSTRB1# и т. д.

Таблица 9.3 – Размер страницы, определяемый битами STRB0 PAGESIZE, STRB1 PAGESIZE †

STRBx PAGESIZE (разряды 14-18, 19-23) ‡	Разряды внешней адресной шины, определяющие текущую страницу	Разряды внешней адресной шины, определяющие адрес на странице	Размер страницы (32-битное слово)
00000-00110	Зарезервированы	Зарезервированы	Зарезервированы
00111*	30-8	7-0	2^8=256
01000	30-9	8-0	2^9=512
01001	30-10	9-0	2^{10}=1К
01010	30-11	10-0	2^{11}=2К
01011	30-12	11-0	2^{12}=4К
01100	30-13	12-0	2^{13}=8К
01101	30-14	13-0	2^{14}=16К
01110	30-15	14-0	2^{15}=32К
01111	30-16	15-0	2^{16}=64К
10000	30-17	16-0	2^{17}=128К
10001	30-18	17-0	2^{18}=256К
10010	30-19	18-0	2^{19}=512К
10011	30-20	19-0	2^{20}=1М
10100	30-21	20-0	2^{21}=2М
10101	30-22	21-0	2^{22}=4М
10110 §	30-23	22-0	2^{23}=8М
10111	30-24	23-0	2^{24}=16М
11000	30-25	24-0	2^{25}=32М
11001	30-26	25-0	2^{26}=64М
11010	30-27	26-0	2^{27}=128М
11011	30-28	27-0	2^{28}=256М
11100	30, 29	28-0	2^{29}=512М
11101	30	29-0	2^{30}=1Г
11110	Нет	30-0	2^{31}=2Г
11111	Зарезервирован	Зарезервирован	Зарезервирован

Примечание – Принятые условные обозначения:

- † – Использованы символы для регистров управления интерфейсом глобальной памяти. Для регистров управления интерфейсом локальной памяти добавляется префикс «L» вначале каждого символа, например, LSTRB0 PAGESIZE, LSTRB1 PAGESIZE и т. д. Описание для регистра управления интерфейсом локальной памяти такое же.
- ‡ – «x» в STRBx означает, что данные в колонках справедливы для STRB0#, STRB1#.
- § – Поле STRBx PAGESIZE 10110_2 изображено на рисунке 9.5.
- * – Значение при сбросе.

Таблица 9.4 – Диапазоны адресов, определяемые битами STRBx ACTIVE †

STRBx PAGESIZE (разряды 24-28) ‡	Диапазон адресов STRB0 ACTIVE	Размер диапазона адресов STRB0 ACTIVE	Диапазон адресов STRB1 ACTIVE
00000-01110	Зарезервированы	Зарезервированы	Зарезервированы
01111	8000 0000-8000 FFFF	2^{16}=64К	8001 0000-FFFF FFFF
10000	8000 0000-8001 FFFF	2^{17}=128К	80020000-FFFF FFFF
10001	8000 0000-8003 FFFF	2^{18}=256К	8004 0000-FFFF FFFF
10010	8000 0000-8007 FFFF	2^{19}=512К	8008 0000-FFFF FFFF
10011	80000000-800F FFFF	2^{20}=1М	8010 0000-FFFF FFFF
10100	8000 0000-801F FFFF	2^{21}=2М	8020 0000-FFFF FFFF
10101	8000 0000-803F FFFF	2^{22}=4М	8040 0000-FFFF FFFF
10110	8000 0000-807F FFFF	2^{23}=8М	8080 0000-FFFF FFFF
10111	8000 0000-80FF FFFF	2^{24}=16М	81000000-FFFF FFFF
11000	8000 0000-81FF FFFF	2^{25}=32М	8200 0000-FFFF FFFF
11001	8000 0000-83FF FFFF	2^{26}=64М	84000000-FFFF FFFF
11010	8000 0000-87FF FFFF	2^{27}=128М	8800 0000-FFFF FFFF
11011	8000 0000-8FFF FFFF	2^{28}=256М	9000 0000-FFFF FFFF
11100	8000 0000-9FFF FFFF	2^{29}=512М	A000 0000-FFFF FFFF
11101	8000 0000-BFFF FFFF	2^{30}=1Г	C000 0000-FFFF FFFF
11110	8000 0000-FFFF FFFF	2^{31}=2Г	Нет
11111	Зарезервирован	Зарезервирован	Зарезервирован

Примечания

1 † – Диапазон адресов определен битами LSTRBx ACTIVE, список которых приведен в таблице 9.5.

2 ‡ – Значение при сбросе.

Таблица 9.5 – Диапазоны адресов, определяемых битами LSTRBx ACTIVE

LSTRBx PAGESIZE (разряды 24-28) †	Диапазон адресов LSTRB0 ACTIVE	Размер диапазона адресов LSTRB0 ACTIVE	Диапазон адресов LSTRB1 ACTIVE
00000-01110	Зарезервированы	Зарезервированы	Зарезервированы
01111	0000 0000-0000 FFFF	2^{16}=64К	0001 0000-7FFF FFFF
10000	0000 0000-0001 FFFF	2^{17}=128К	0002 0000-7FFF FFFF
10001	0000 0000-0003 FFFF	2^{18}=256К	0004 0000-7FFF FFFF
10010	0000 0000-0007 FFFF	2^{19}=512К	0008 0000-7FFF FFFF
10011	0000 0000-000F FFFF	2^{20}=1М	0010 0000-7FFF FFFF
10100	0000 0000-001F FFFF	2^{21}=2М	0020 0000-7FFF FFFF
10101	0000 0000-003F FFFF	2^{22}=4М	0040 0000-7FFF FFFF
10110	0000 0000-007F FFFF	2^{23}=8М	0080 0000-7FFF FFFF
10111	0000 0000-00FF FFFF	2^{24}=16М	0100 0000-7FFF FFFF
11000	0000 0000-01FF FFFF	2^{25}=32М	0200 0000-7FFF FFFF
11001	0000 0000-03FF FFFF	2^{26}=64М	0400 0000-7FFF FFFF
11010	0000 0000-07FF FFFF	2^{27}=128М	0800 0000-7FFF FFFF
11011	0000 0000-0FFF FFFF	2^{28}=256М	9000 0000-7FFF FFFF
11100	0000 0000-1FFF FFFF	2^{29}=512М	A000 0000-7FFF FFFF
11101	0000 0000-3FFF FFFF	2^{30}=1Г	C000 0000-7FFF FFFF
11110	0000 0000-7FFF FFFF	2^{31}=2Г	Нет
11111	Зарезервирован	Зарезервирован	Зарезервирован

Примечание – † – Значение при сбросе.

9.3.1 Карта адресов для стробов

Рисунок 9.4 показывает взаимосвязь между битами STRBx ACTIVE, смотри рисунок 9.3 для более детальной информации, и диапазонами адресов, при которых сигналы STRB0# и STRB1# активны. Заметим, что области адресов STRBx# и LSTRBx# также определяют области ассоциированных с ними сигналов – RDYx#, LRDYx#, RD/WRx#, LRD/WRx#, PAGEx, LPAGEx и т. п., где x = 1 или 0.

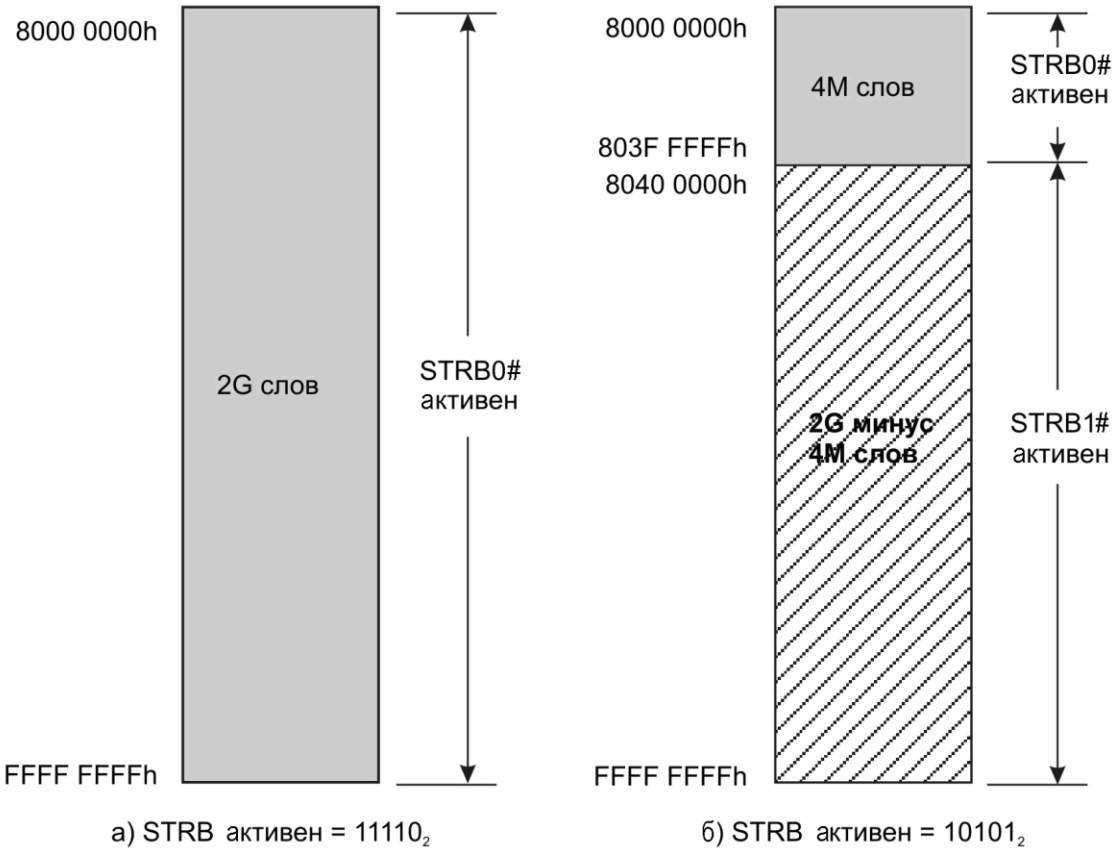


Рисунок 9.4 – Влияние бит STRBx ACTIVE на карту памяти шины глобальной памяти

Примечание – На рисунке 9.4 показаны два примера для карты глобальной памяти. Полная карта памяти (локальная и глобальная) показана на рисунке 4.1. Необходимо заметить, что старший адрес для LSTRB1# (локальная шина) – 7FFF FFFFh.

Пример а) на рисунке 9.4 показывает конфигурацию карты после сброса STRB ACTIVE = 11110_2. В этом случае сигнал STRB0# активен на всей адресной области шины глобальной памяти, см. таблицу 9.1.

Пример б) на рисунке 9.4 показывает карту глобальной шины памяти, когда STRB ACTIVE=10101_2. В этом случае STRB0# активен в области адресов 8000 0000h-803F FFFFh, и STRB1# активен в области адресов 8040 0000h-FFFF FFFFh, как показано в таблице 9.4 для STRB ACTIVE = 10101_2.

9.3.2 Установка размера страницы

С учетом области памяти, выбранной любыми четырьмя стробами, внешний интерфейс ИС 1867ВМ9Ф позволяет в дальнейшем разделить эту область на страницы выбранного размера. Это позволяет достичь гибкости в разработке высокоскоростных и высокоплотных систем памяти, объединенных с более медленными периферийными устройствами, поскольку позволяет добавить экстра цикл при пересечении границы страницы, который позволяет внешней логике переконфигурировать себя.

Каждое поле PAGESIZE в регистре управления интерфейсом памяти, как показано на рисунке 9.3, работает сходным образом с определением размера страницы для соответствующего строба. Таблица 9.5 иллюстрирует взаимосвязь между полем PAGESIZE и разрядами адреса, используемого для определения текущей страницы и конечного размера страницы. Размер страницы начинается с 256 слов (с разрядами 7-0 внешней адресной шины, определяющими адрес на странице) и достигает до 2Г слов с разрядами 30-0 внешней адресной шины, определяющими положение на странице. Пример на рисунке 9.5 показывает, как значение поля размера страницы 10110_2 переводится в разряды 30-23, определяющие текущую страницу и разряды 22-0, определяющие адрес на странице.

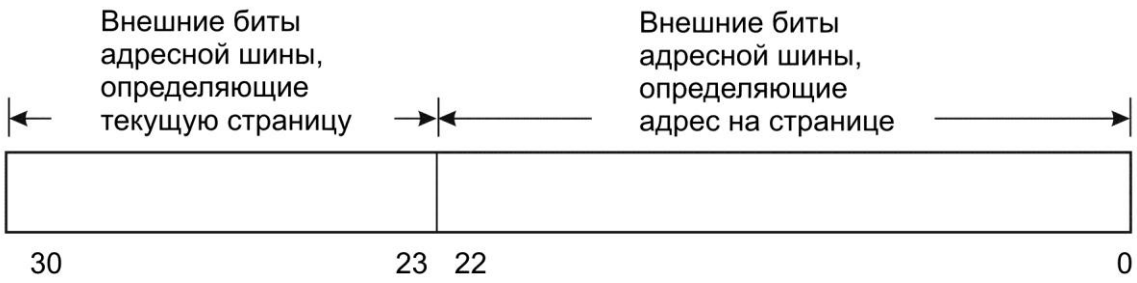


Рисунок 9.5 – Пример полей STRBx PAGESIZE

Примечание – Рисунок 9.5 представляет значение поля STRBx PAGESIZE 10110_2, как показано в таблице 9.3.

При переходе от одной страницы к другой добавляется экстра цикл в последовательность внешнего доступа, позволяя внешней логике переконфигурировать себя. Например, добавочный цикл дает время для медленного устройства освободить шину, таким образом, устраняется конфликт шины. Логика управления интерфейсом памяти сохраняет последовательность использованных адресов для последнего доступа для каждого STRBx#. Когда начинается обращение, сигнал PAGE, соответствующий активному STRBx#, становится неактивным (высокий уровень), если обращение происходит к новой странице. Сигналы PAGE0 и PAGE1 независимы один от другого, каждый имеет собственную логику определения размера страницы.

При сбросе логика управления страницей инициализируется, так что дополнительный цикл добавляется для первого доступа к двум стробам интерфейса.

Регистры управления интерфейсом локальной памяти функционируют аналогично регистрам управления интерфейсом глобальной памяти.

9.4 Программируемые состояния ожидания

ИС 1867ВМ9Ф имеет возможность программно устанавливать циклы ожидания для каждого строба и способы генерации сигнала готовности. Программный генератор состояния ожидания управляется конфигурированием двух полей в регистрах управления глобальным и локальным интерфейсами. Поле STRBx WTCNT (разряды 8-10 и 11-13) используется для определения числа сгенерированных программных состояний ожидания, а поле STRBx SWW (разряды 6, 7 и 4, 5) используется для выбора одного из следующих четырех режимов генерации состояния ожидания:

- Внешний RDY# (SWW=0). Состояния ожидания генерируются исключительно внешним RDY# (программное состояние ожидания игнорируется).

- Полученный из WTCNT RDY_{wtcnt}# (SWW=01_2). Состояния ожидания генерируются исключительно программой-генератором состояний ожидания (внешний RDY# игнорируется).

- Логическое OR («ИЛИ») над RDY# и RDY_{wtcnt}# (SWW=10_2). Состояние ожидания генерируется логическим «ИЛИ» внутреннего и внешнего сигналов готовности. Любой из сигналов может генерировать готовность.

- Логическое AND («И») над RDY# и RDY_{wtcnt}# (SWW=11_2). Состояния ожидания генерируются логическим «И» внутреннего и внешнего сигналов готовности. Оба сигнала должны существовать.

Четыре режима используются для выработки внутреннего сигнала готовности RDY_{int}#, который управляет доступом (обращениями). Пока RDY_{int}# = 1, то текущий внешний доступ по шине расширен. Когда RDY_{int}# = 0, то выполняемый текущий доступ может быть завершен. Так как использование программируемых состояний ожидания для обоих внешних интерфейсов идентично, то далее описывается только интерфейс глобальной шины. RDY_{wtcnt}# – внутренний сигнал готовности. Когда начинается внешнее обращение, то значение WTCNT загружается в счетчик. WTCNT может иметь значение от 0 до 7. Счетчик уменьшается каждый цикл H1/H3, пока не станет равным «0». После установки счетчика в «0» его значение не меняется до следующего обращения. Если счетчик не «0», то RDY_{wtcnt}# = 1. Если счетчик равен «0», то RDY_{wtcnt}# = 0.

Примечание – При сбросе ИС добавляется 7 состояний ожидания для каждого доступа к внешней памяти. Это делается для того, чтобы гарантировать начало функционирования системы с медленной памятью после сброса. Для повышения производительности системы при использовании быстрой памяти необходимо уменьшать число состояний ожидания.

В таблице 9.6 приведены значения SWW для различных значений RDY#, RDY_{wtcnt}#, RDY_{int}#.

Таблица 9.6 – Генерация состояний ожидания для каждого значения SWW

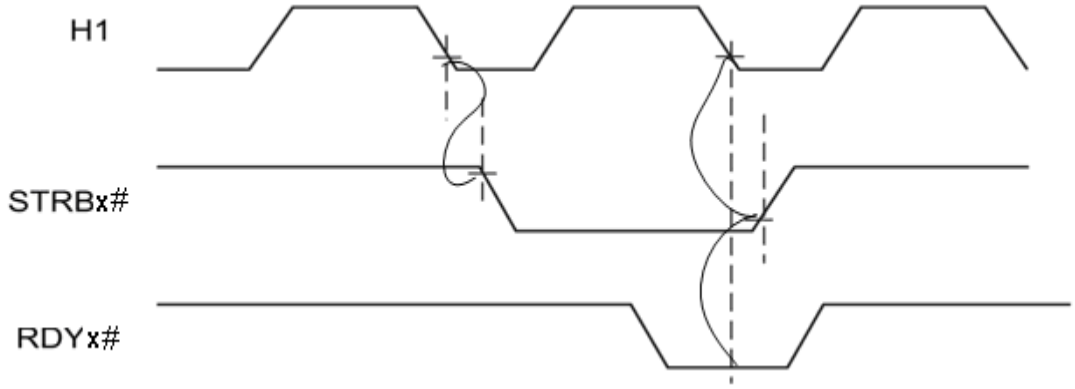
Значение SWW	RDY#	RDY_{wtcnt}#	RDY_{int}#	RDY_{int}#
00	0	0	0	RDY_{int}# зависит только от RDY#,
00	0	1	0	RDY_{wtcnt}# игнорирован
00	1	0	1	
00	1	1	1	
01	0	0	0	RDY_{int}# зависит только от RDY_{wtcnt}#,
01	0	1	1	RDY# игнорирован
01	1	0	0	
01	1	1	1	
10	0	0	0	RDY_{int}# – логическое «ИЛИ» (электрическое «И», так как эти сигналы истинны в низком уровне) RDY# и RDY_{wtcnt}#
10	0	1	0	
10	1	0	0	
10	1	1	1	
11	0	0	0	RDY_{int}# – логическое «И» (электрическое «ИЛИ», так как эти сигналы истинны в низком уровне) RDY# и RDY_{wtcnt}#
11	0	1	1	
11	1	0	1	
11	1	1	1	

9.5 Временная диаграмма интерфейса памяти

ИС 1867ВМ9Ф выполняет одноцикловое внешнее чтение и конвейерную внешнюю запись, за исключением некоторых случаев, которые рассмотрены в данном подразделе.

Запись производится в 2 этапа: в первом цикле данные записываются в буфер порта внешней памяти, а в следующем цикле данные пересылаются во внешнюю память.

Примечание – Для дальнейшей работы ПДП или ЦПУ операция записи заканчивается в первом цикле и ПДП или ЦПУ продолжают работать. Однако если следующее обращение ПДП или ЦПУ происходит к одной и той же внешней шине, то ПДП или ЦПУ должны ждать, и запись, таким образом, занимает 2 цикла.



Примечание – Пунктирная линия на рисунке 9.6 подчеркивает взаимосвязь между сигналами.

Рисунок 9.6 – Временная диаграмма сигналов STRBx# и RDYx#

Как показано на рисунке 9.6, STRBx# изменяется по срезу H1, а RDYx# фиксируется по срезу H1.

Для других временных диаграмм, указанных в данном подразделе, применяются следующие правила:

- Изменение RD/WRx# определяется STRBx#.
- Пересечение границы страниц для каждого STRBx# отражается в установке PAGEx в высокий уровень для одного цикла.
- Изменение RD/WDx# всегда происходит по фронту H1.
- Изменение STRBx# всегда происходит по срезу H1.
- RDYx# фиксируется по срезу H1.
- Данные при чтении всегда защелкиваются по срезу H1.
- Данные при записи всегда управляются срезом H1.
- Данные никогда не изменяются и не управляются в течение записи по переднему фронту H1.
- Сигналы состояния и PAGEx, следующие за чтением, изменяются по срезу H1. Адрес также изменяется по срезу H1.
- Выборка вектора прерывания для внешнего интерфейса определяется сигналами состояния для этого интерфейса STAT3-STAT0 и LSTAT3-LSTAT0 как чтение данных.
- Сигналы состояния операций блокировки LOCK# и LLOCK# имеют одну и ту же временную диаграмму, что и сигналы состояния STAT3-STAT0 и LSTAT3-LSTAT0, соответственно.

- Если сигнал PAGEx переходит в высокий уровень, то STRBx# также переходит в высокий уровень.

Примечание – Если нет обращения к внешнему порту (состояние бездействия – IDLE), то сигналы управления находятся в неактивном состоянии (RDYx# игнорируется, STRBx# находится в высоком уровне, а сигналы STAT3-STAT0 переходят в высокий уровень, шина адреса сохраняет последнее значение адреса и переходит в высокоимпедансное состояние, см. рисунок 9.16.

Рисунок 9.7 иллюстрирует последовательность «чтение, чтение, запись» и предполагается, что по STRB1# происходит обращение, три обращения к одной и той же странице. Временная диаграмма показывает следующее:

- Два чтения в одной и той же странице как одноцикловое обращение.
- STRBx# остается в низком уровне в течение двухкратного чтения.
- После перехода от чтения к записи сигнал STRBx# переходит в высокий уровень на один цикл для изменения сигнала RD/WRx#.

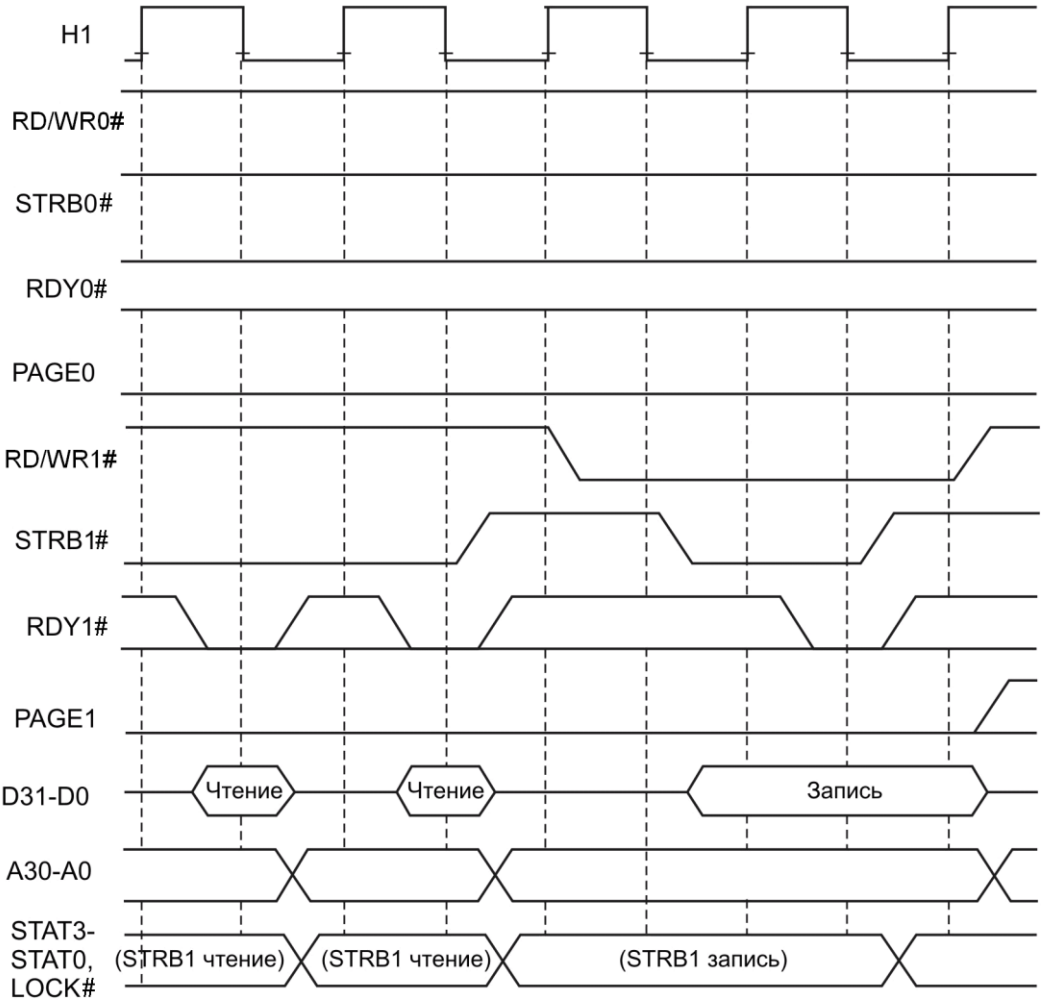


Рисунок 9.7 – Последовательность «чтение, чтение, запись» в одной и той же странице

Рисунок 9.8 показывает следующее:

- для предотвращения нежелательной записи STRB1# переходит в высокий уровень между повторными записями для отключения памяти, пока изменится адрес;
- как показано на рисунке 9.7, STRB1# переходит в высокий уровень между записью и чтением для изменения;

- чтение, следующее после записи с той же шины, занимает 2 цикла. Это происходит независимо от того, что чтение происходит с тем же стробом и/или в одной и той же странице;

- последовательная запись занимает 2 цикла.

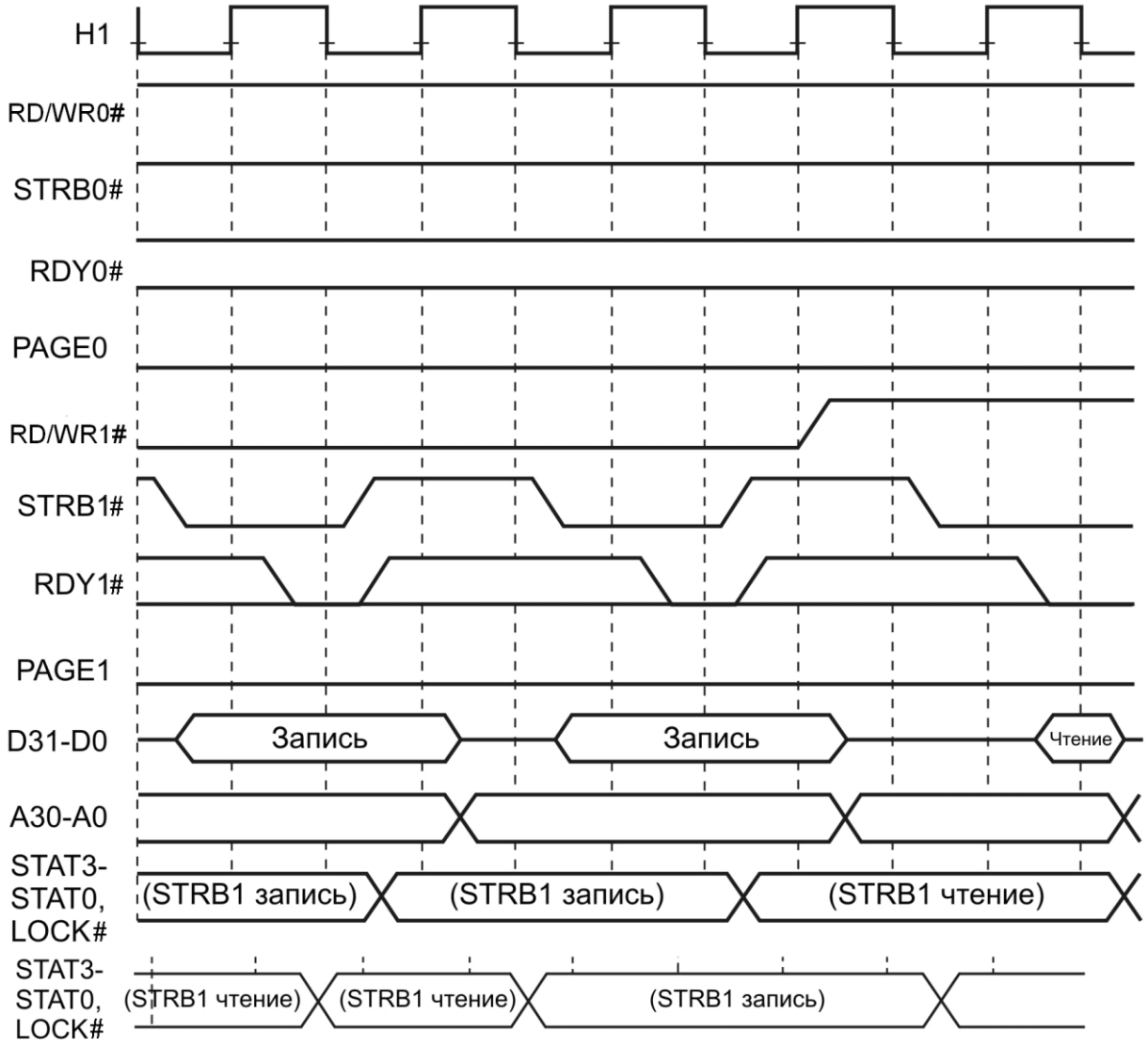


Рисунок 9.8 – Последовательность «запись, запись, чтение» в одной и той же странице

На рисунке 9.9 приведена временная диаграмма последовательности изменения сигналов при чтении в разных страницах:

- добавляется цикл, чтобы выбрать следующую область памяти;

- переход сигнализируется сигналом PAGEx, который переходит в высокий уровень за один цикл;

- STRB1# переходит в высокий уровень на один цикл.

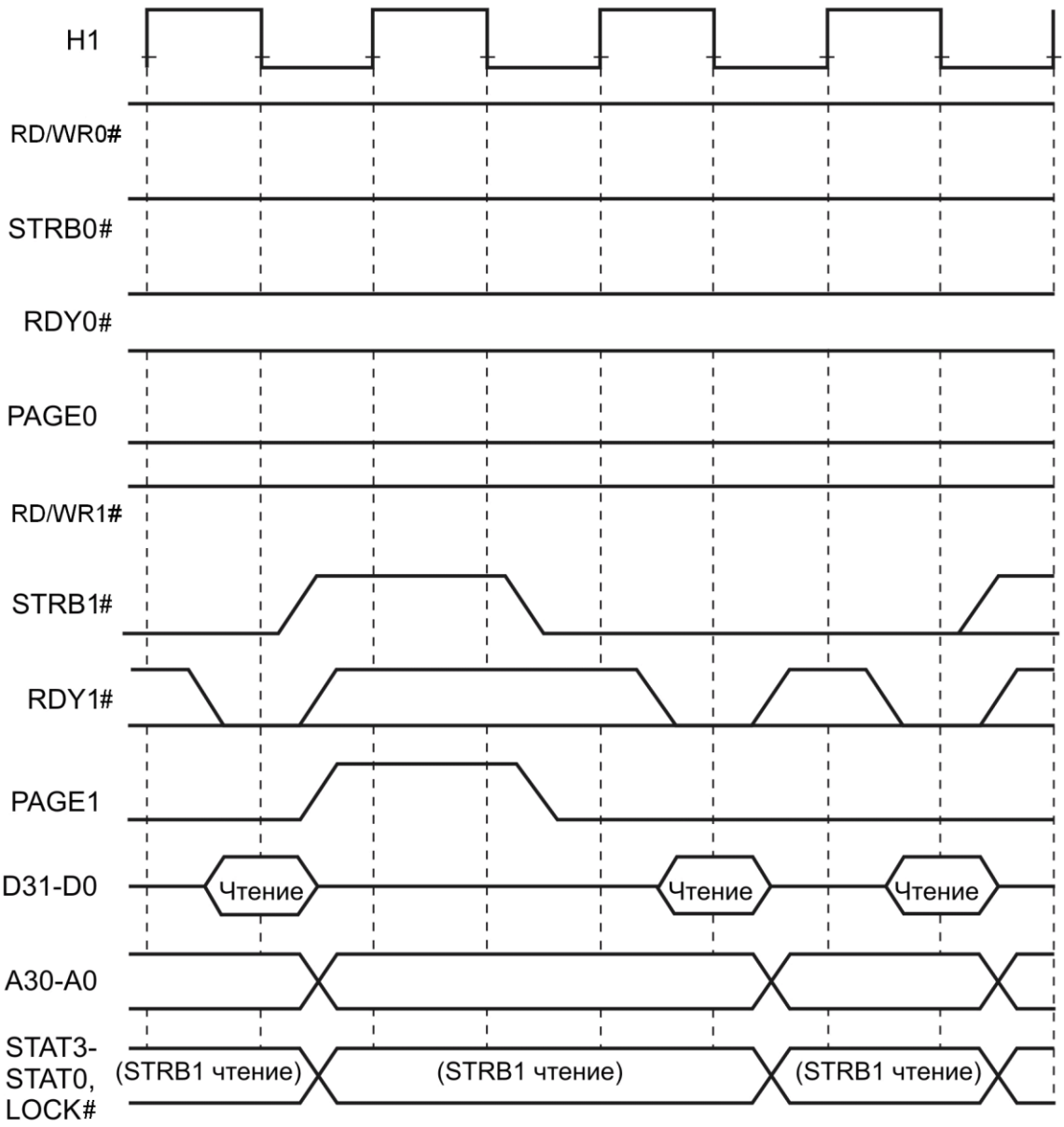


Рисунок 9.9 – Последовательность «чтение в одной странице, чтение в другой странице, чтение в предыдущей странице»

На рисунке 9.10 приведена временная диаграмма последовательности изменения сигналов при записи в разные страницы:

- сигнал PAGE1 переходит в высокий уровень на один цикл;
- дополнительный цикл не добавляется, так как в цикле записи информация об адресе устанавливается в полуцикле H1 перед тем, как сигнал STRB# перейдет в низкий уровень.

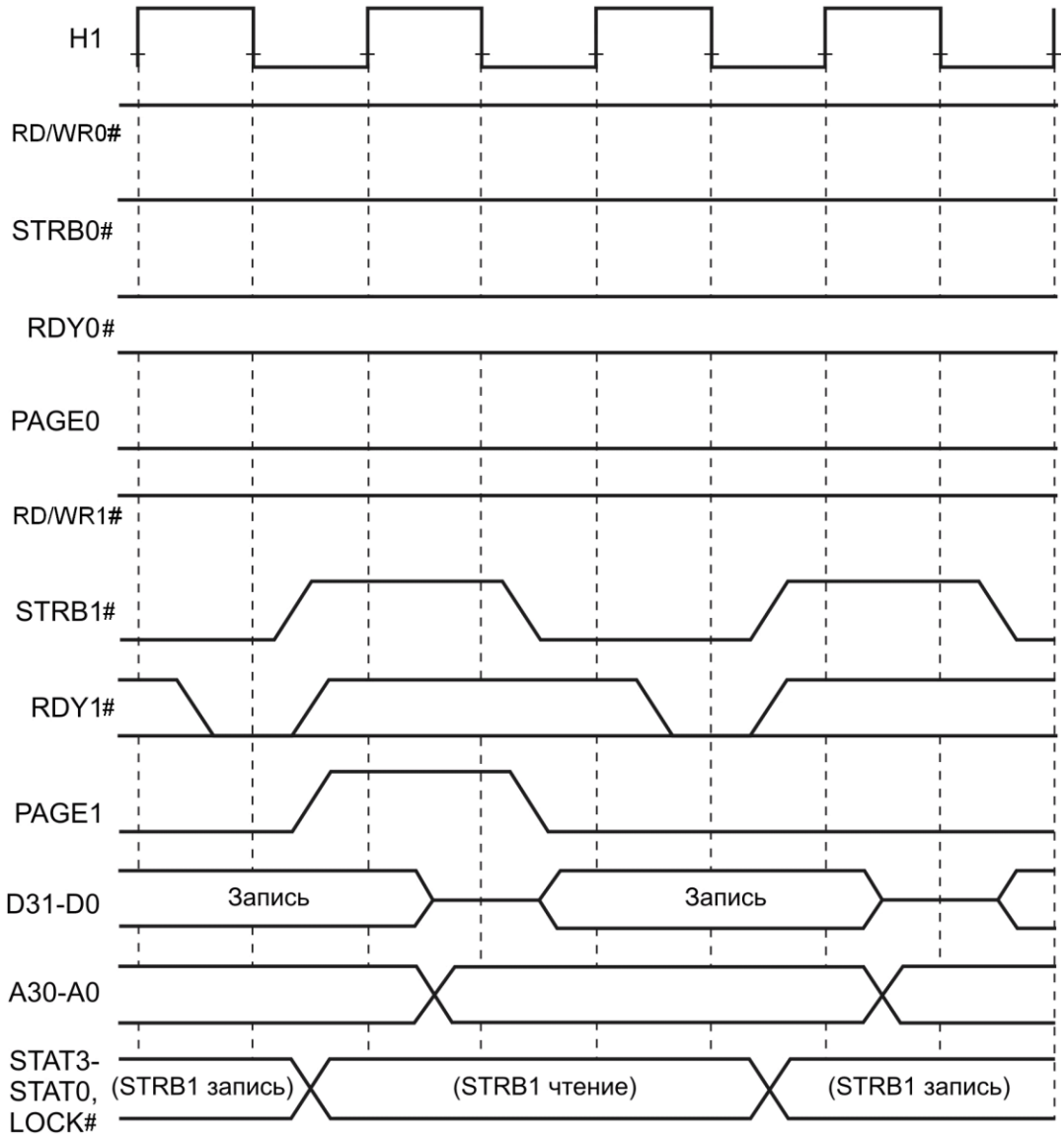


Рисунок 9.10 – Последовательность «запись в одной странице, запись в другой странице, запись в предыдущей странице»

На рисунке 9.11 приведена временная диаграмма последовательности «запись в одной странице, чтение в другой странице, запись в другой странице».

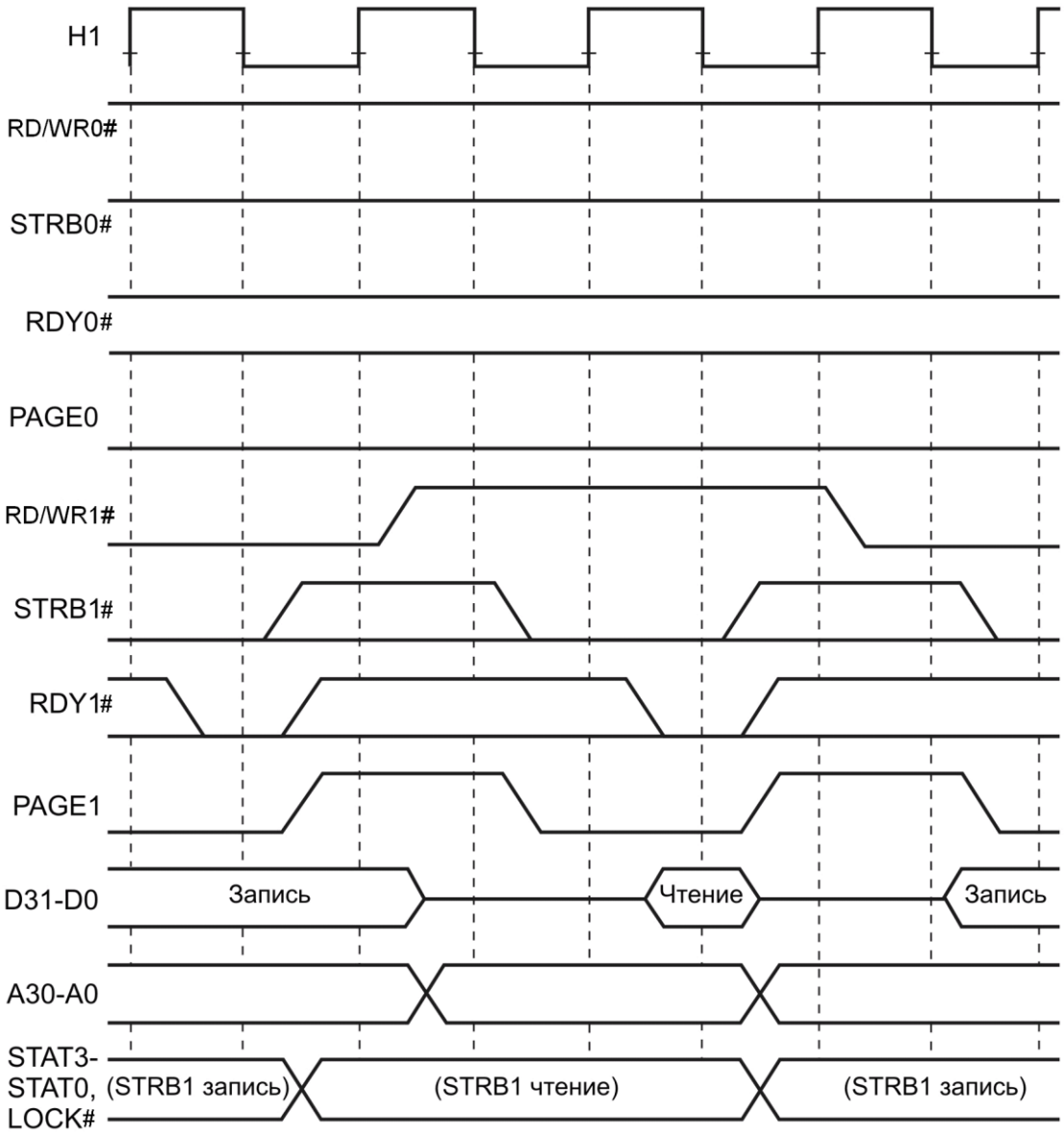


Рисунок 9.11 – Последовательность «запись в одной странице, чтение в другой странице, запись в другой странице»

На рисунке 9.12 приведена временная диаграмма последовательности «чтение в другой странице, чтение в другой странице, запись в предыдущей странице».

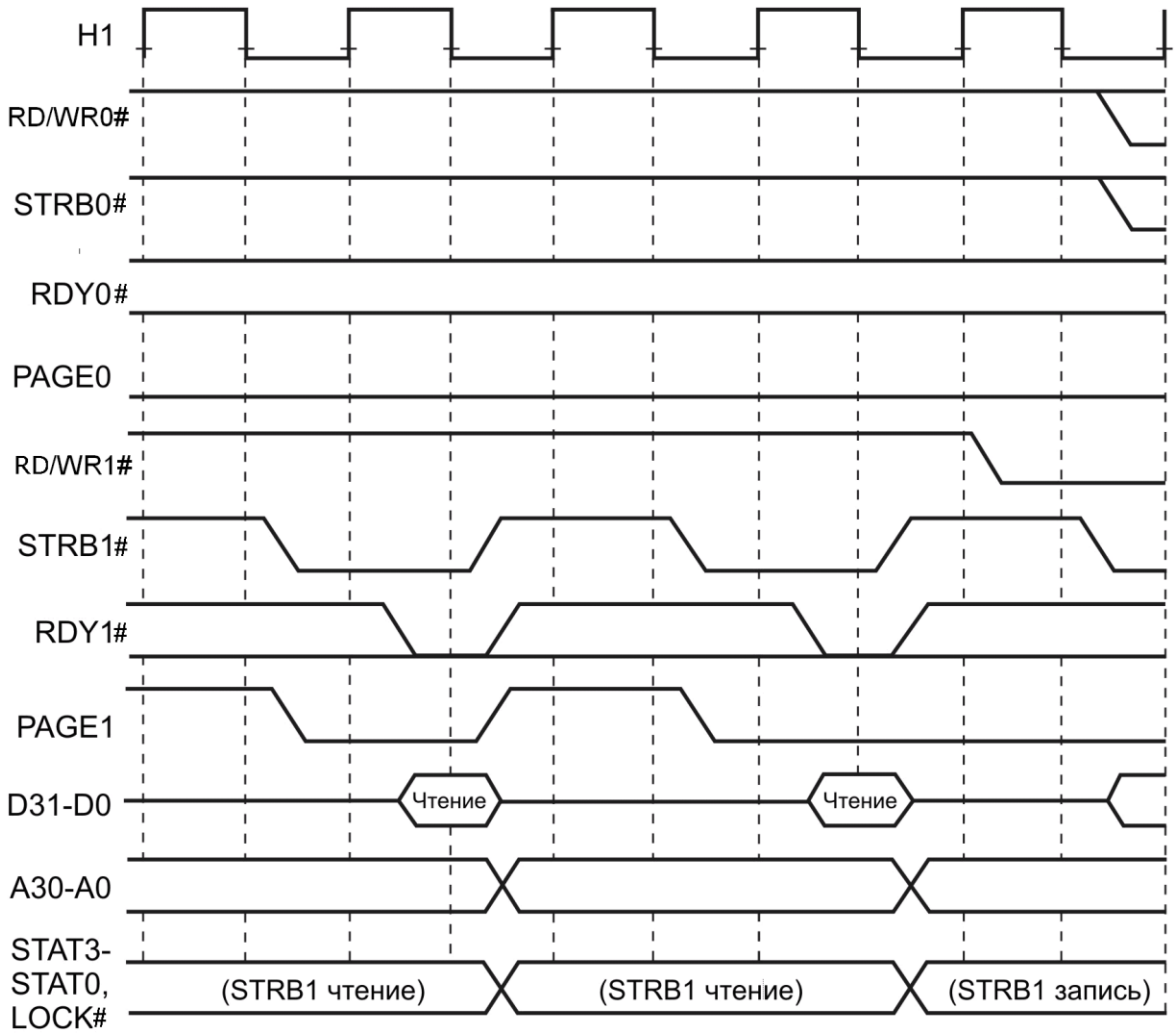


Рисунок 9.12 – Последовательность «чтение в другой странице, чтение в другой странице, запись в предыдущей странице»

На рисунке 9.13 приведена временная диаграмма последовательности «запись в другой странице, запись в другой странице, чтение в предыдущей странице»

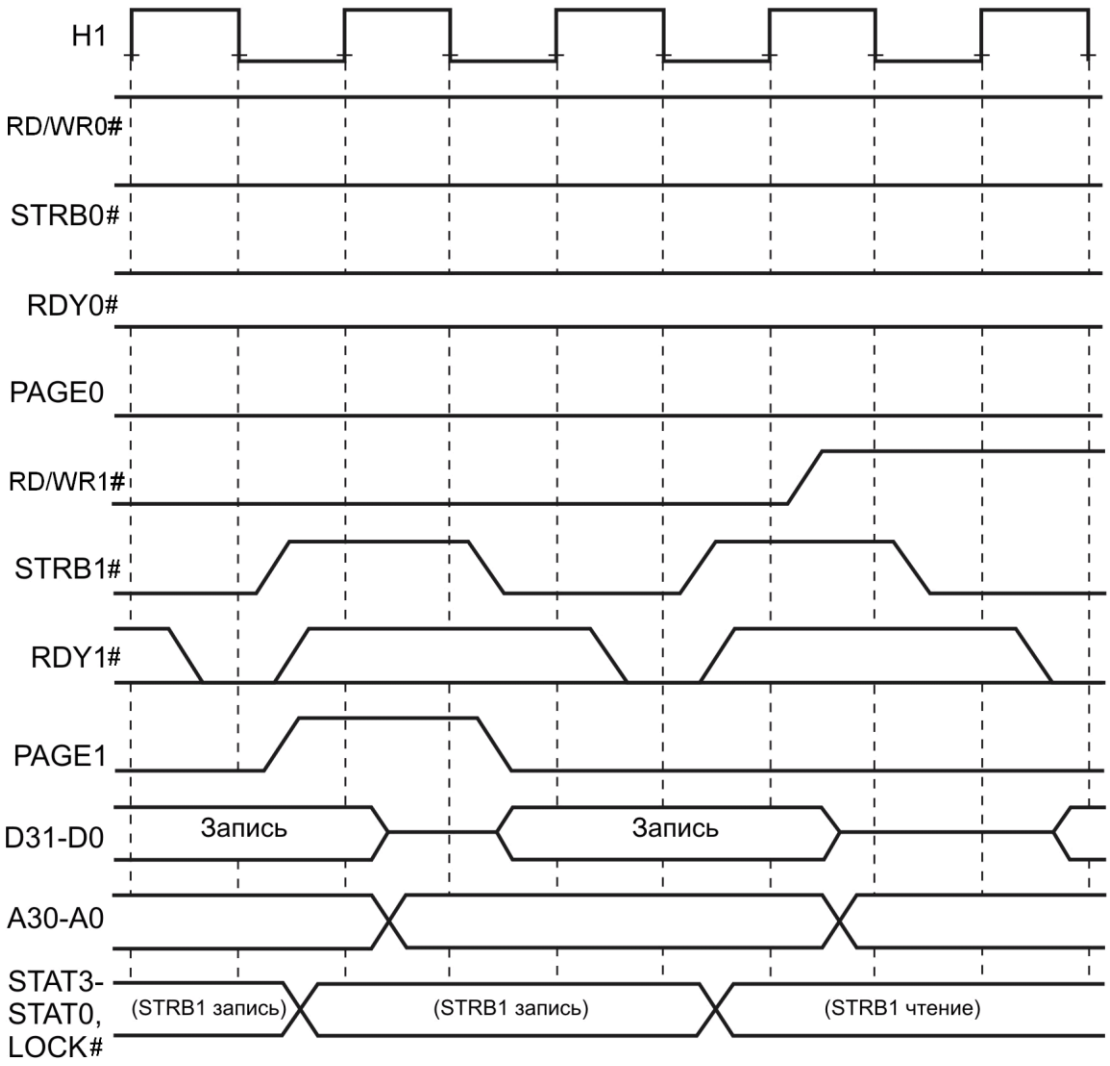


Рисунок 9.13 – Последовательность «запись в другой странице, запись в другой странице, чтение в предыдущей странице»

На рисунке 9.14 приведена временная диаграмма последовательности «чтение в одной странице, запись в другой странице, чтение в другой странице»

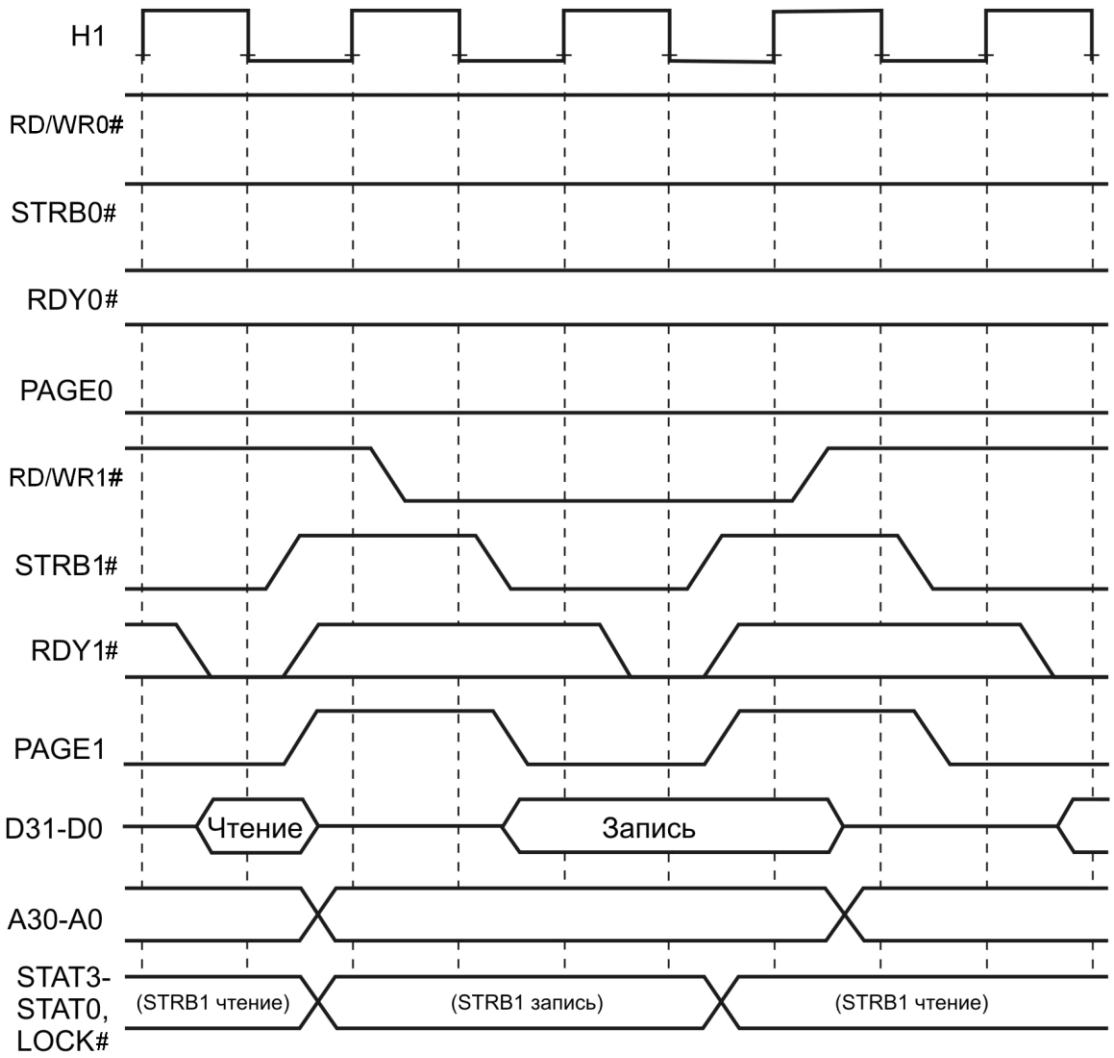


Рисунок 9.14 – Последовательность «чтение в одной странице, запись в другой странице, чтение в другой странице»

Рисунки 9.15 – 9.19 показывают циклы шины в режиме ожидания. Временная диаграмма цикла шины в режиме ожидания аналогична временной диаграмме цикла чтения. Основное различие – данные не считываются, а STRBx# удерживается в высоком уровне, RDYx# игнорируется.

На рисунке 9.15 приведена временная диаграмма «чтение в одной странице, один цикл бездействия, чтение в той же странице».

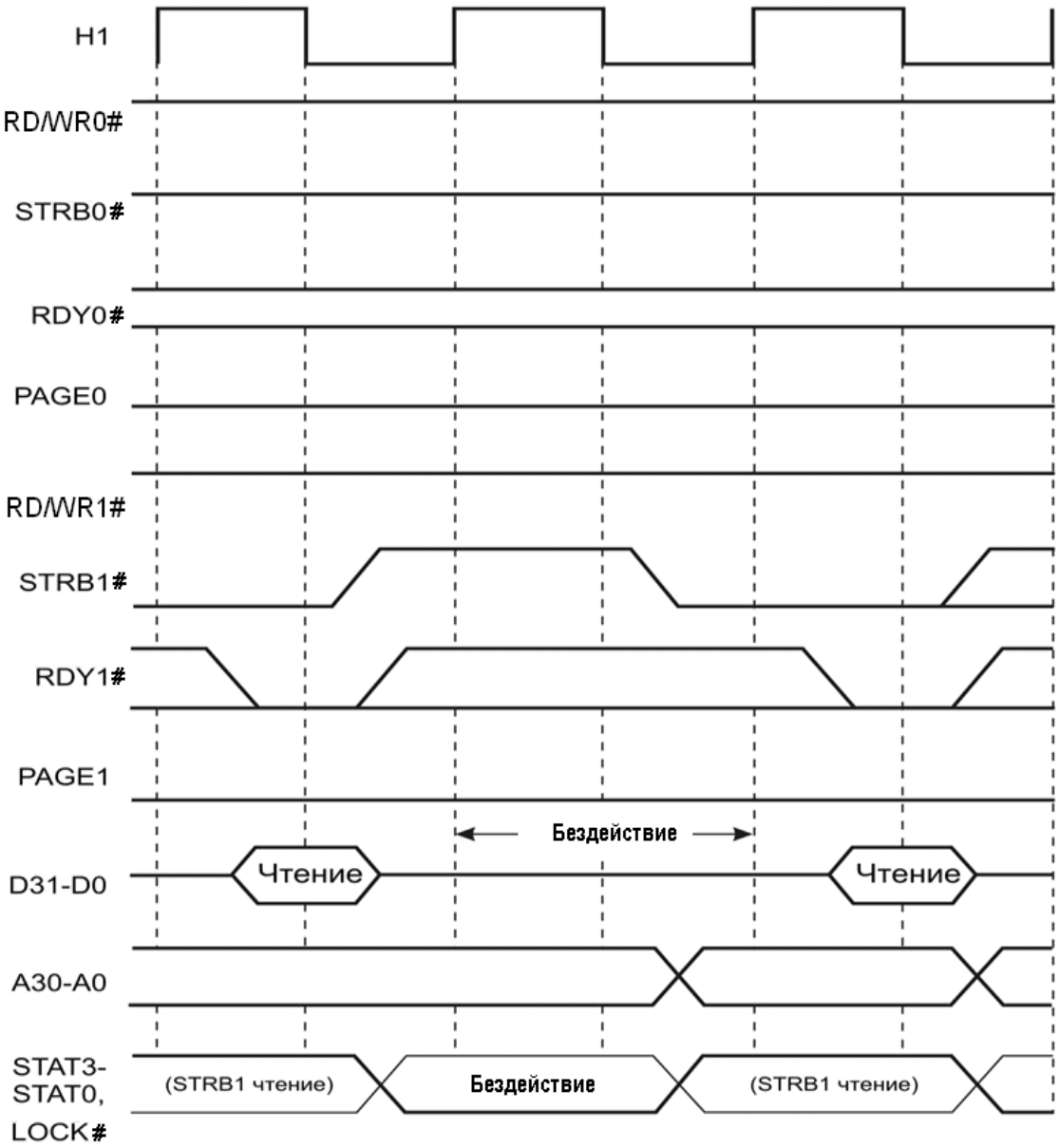


Рисунок 9.15 – Последовательность «чтение в одной странице, один цикл бездействия, чтение в той же странице»

На рисунке 9.16 приведена временная диаграмма последовательности «запись в одной странице, один цикл бездействия, запись в другой странице».

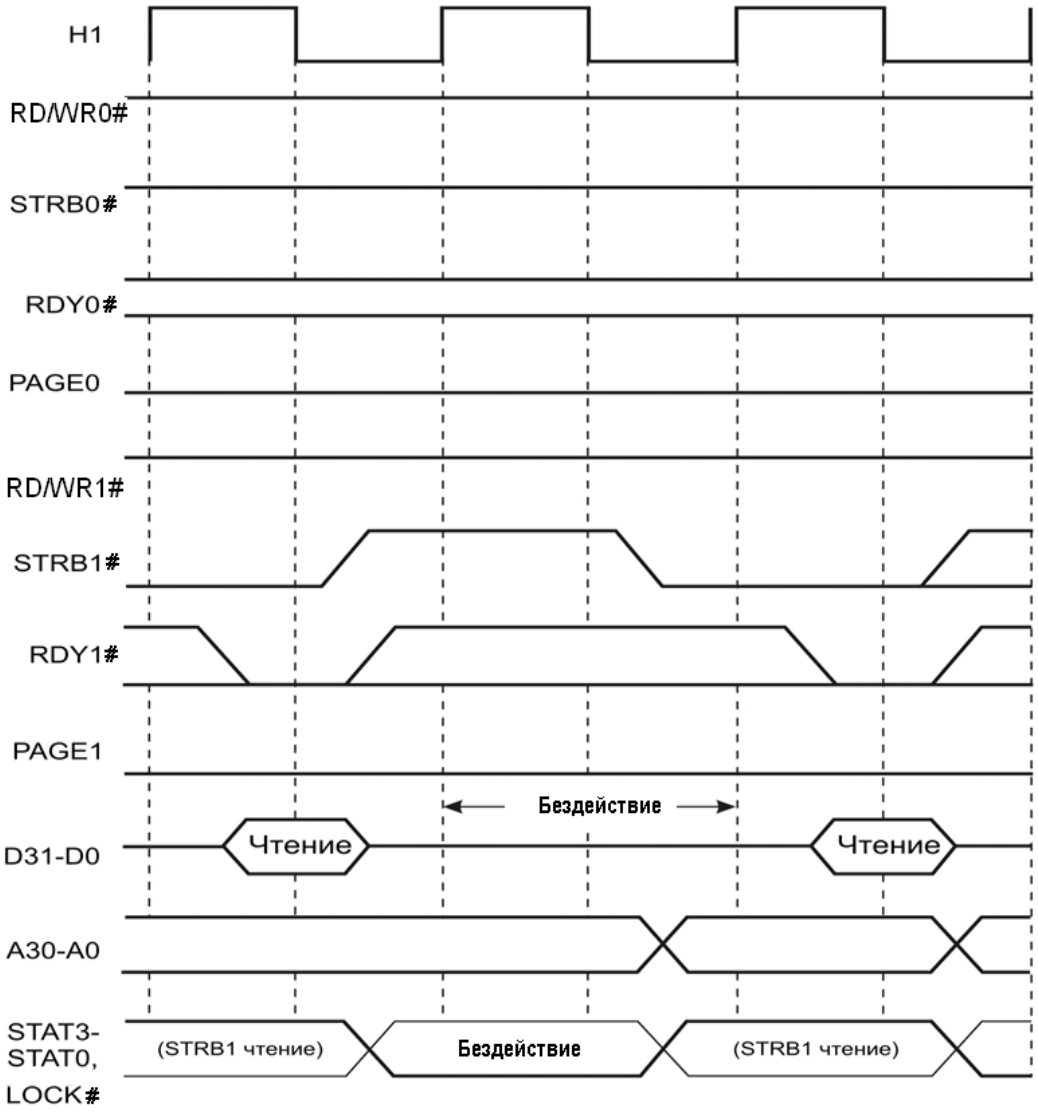


Рисунок 9.16 – Последовательность «запись в одной странице, один цикл бездействия, запись в другой странице»

На рисунке 9.17 приведена временная диаграмма последовательности «бездействие, чтение в другой странице, бездействие».

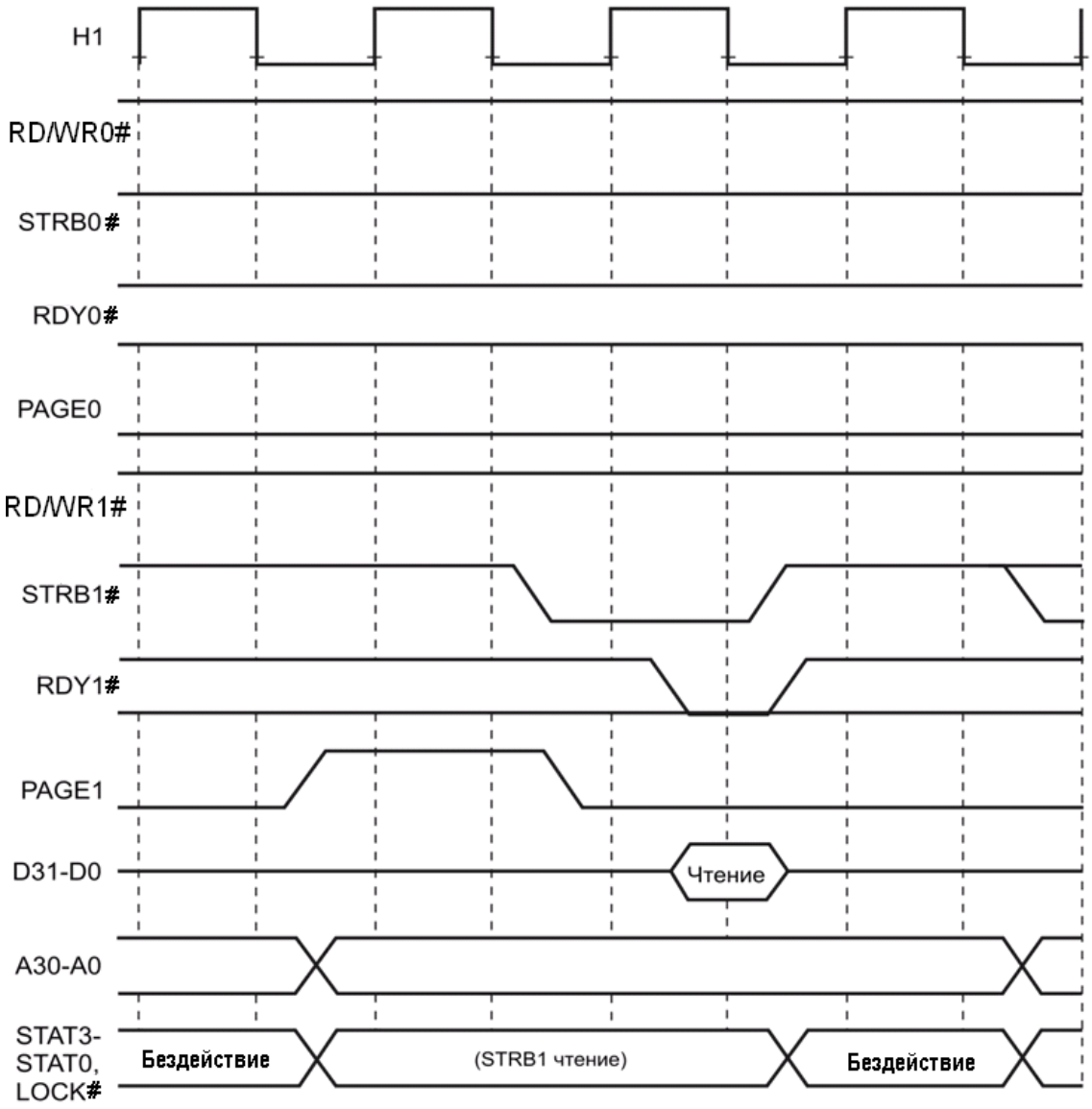


Рисунок 9.17 – Последовательность «бездействие, чтение в другой странице, бездействие»

На рисунке 9.18 приведена временная диаграмма последовательности «бездействие, запись в одной странице, бездействие».

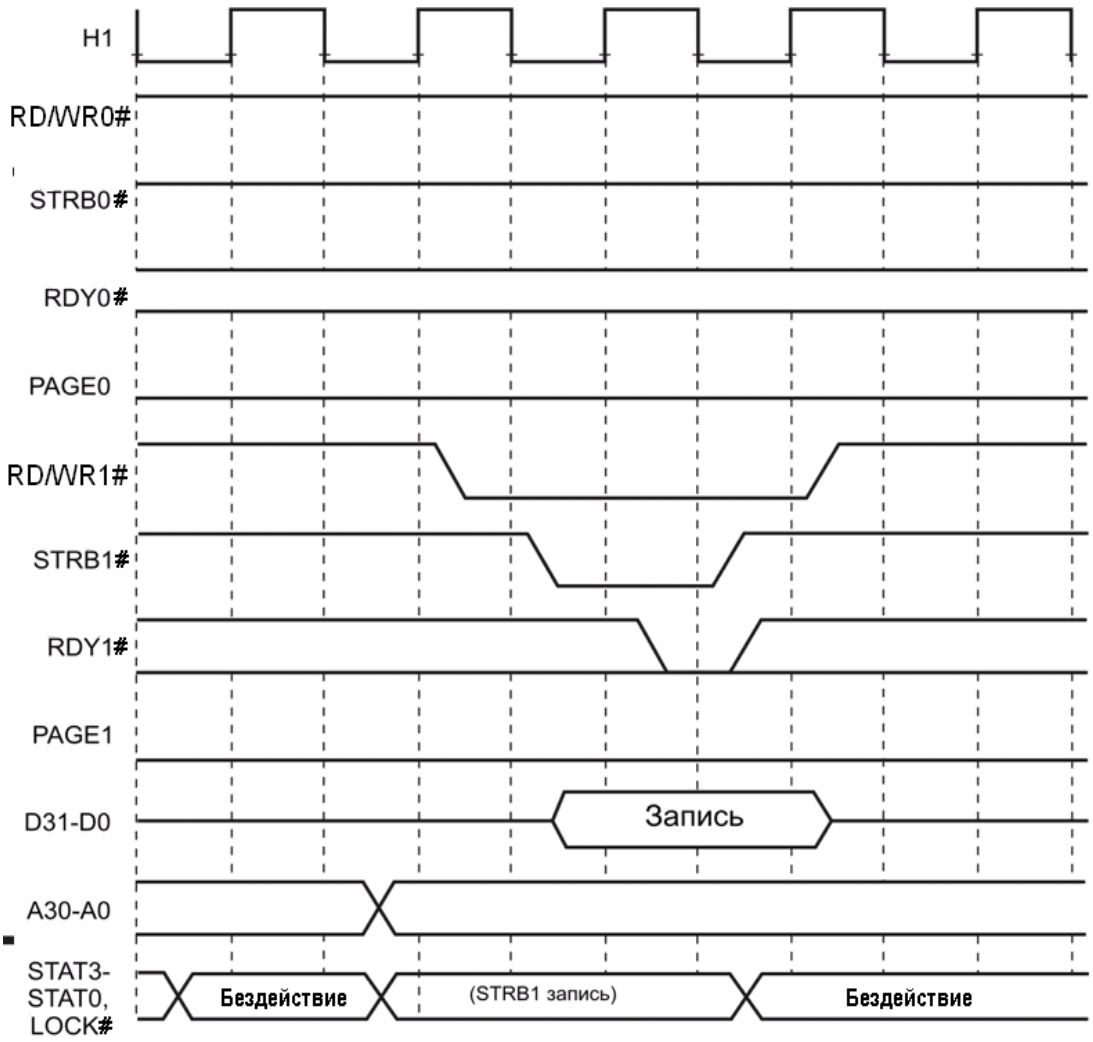


Рисунок 9.18 – Последовательность «бездействие, запись в одной странице, бездействие»

На рисунке 9.19 приведена временная диаграмма последовательности «запись в другой или предыдущей странице, бездействие, бездействие».

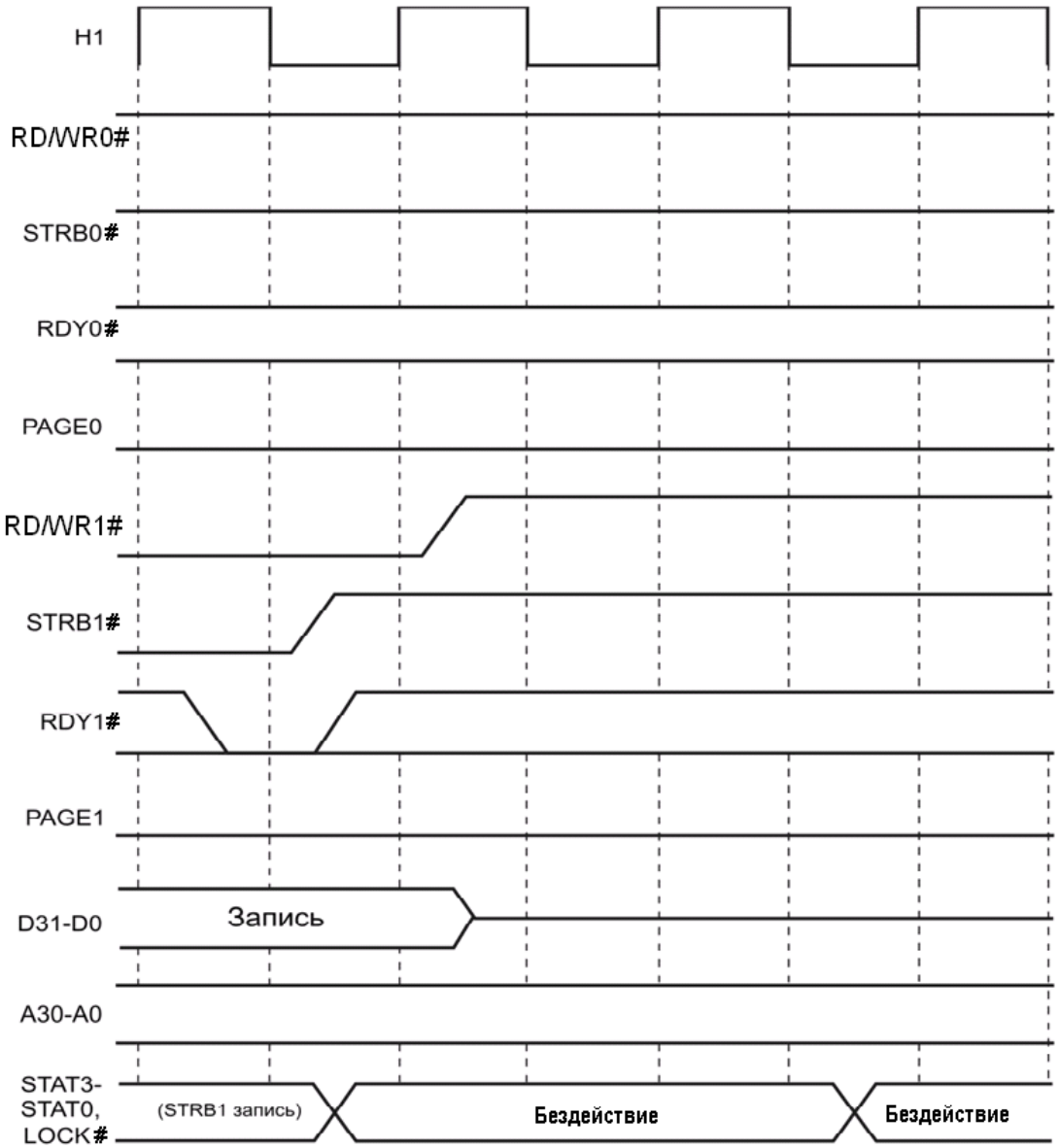


Рисунок 9.19 – Последовательность «запись в другой или предыдущей странице, бездействие, бездействие»

Рисунок 9.20 показывает чтение по STRB1#, следующее за чтением по STRB0# при STRB SWITCH = 0. Это позволяет осуществлять повторное чтение на внешней шине без добавления цикла между последовательными чтениями, когда они активированы разными стробами.

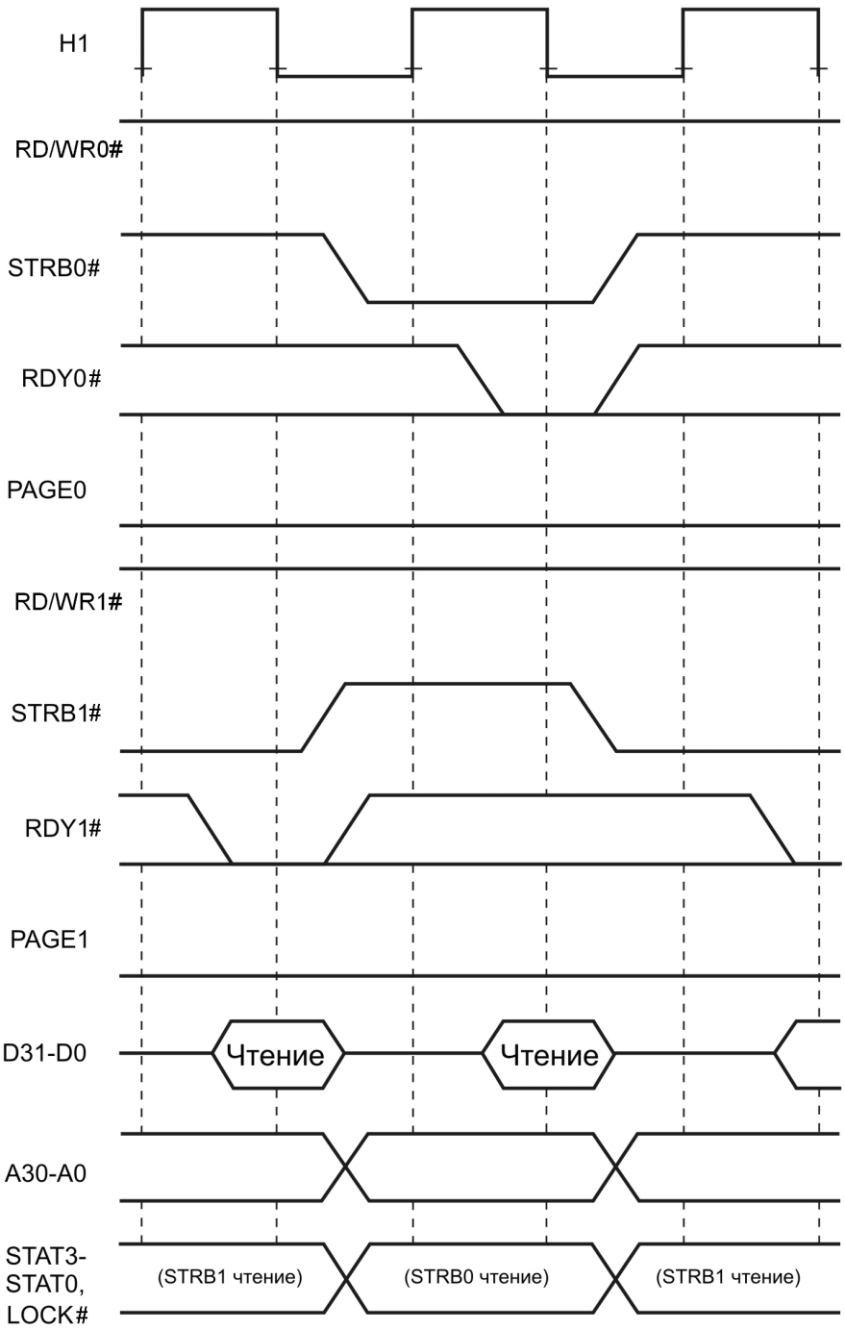


Рисунок 9.20 – Последовательность «чтение в одной странице по STRB1#, STRB0#, STRB1#» при STRB SWITCH = 0

Рисунок 9.21 аналогичен рисунку 9.20 за исключением того, что второе чтение по STRB1# происходит в другой странице.

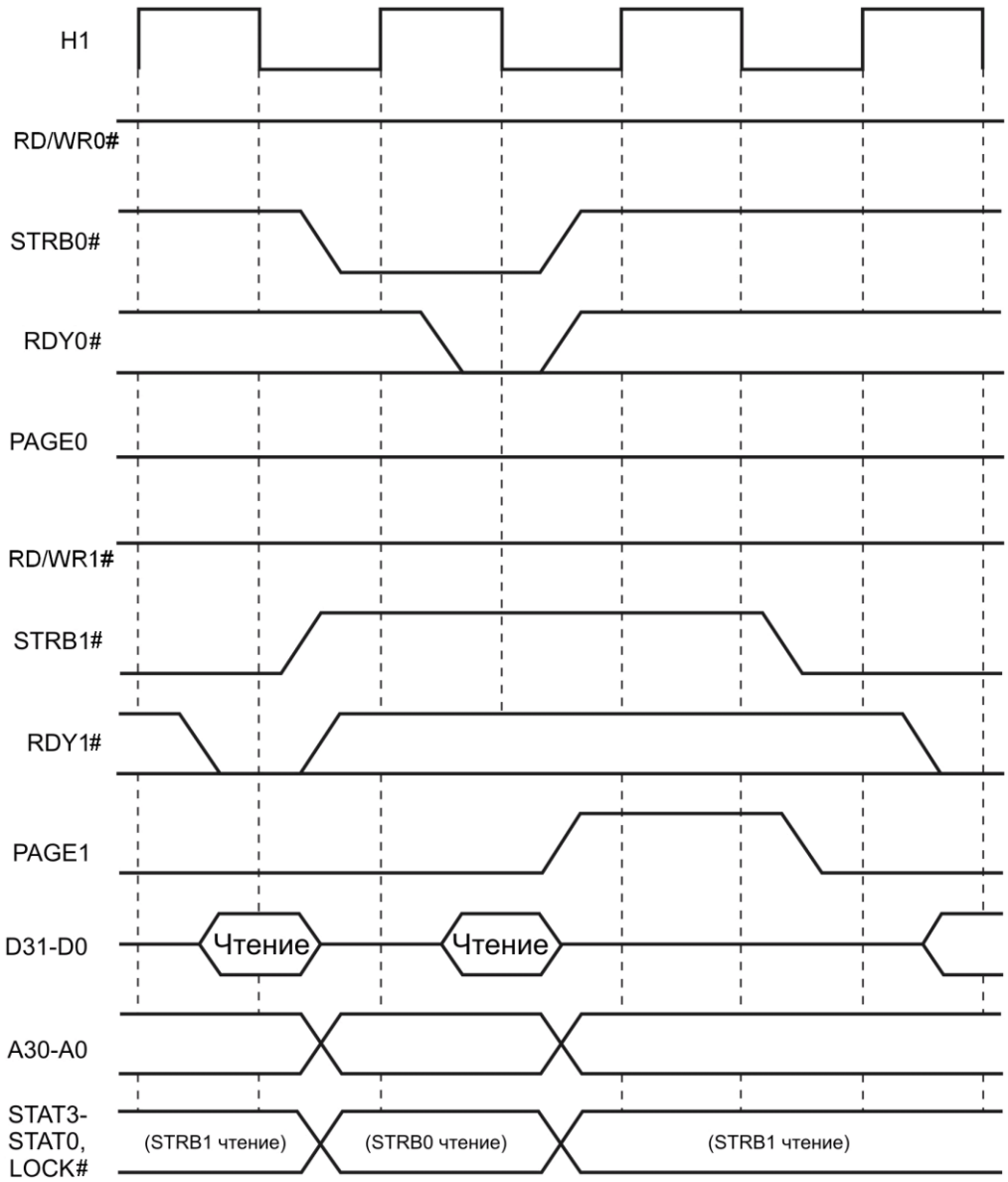


Рисунок 9.21 – Последовательность «чтение в одной странице по STRB1#, STRB0#, чтение в другой странице по STRB1#» при STRB SWITCH = 0

Рисунок 9.22 показывает чтение по STRB1#, следующее чтение по STRB0# при STRB SWITCH = 1. В этом режиме добавляется цикл между чтениями, при которых устанавливаются разные стробы. Некоторые конфигурации памяти требуют этот цикл между стробами для предотвращения конфликтов в течение повторного чтения по разным стробам.

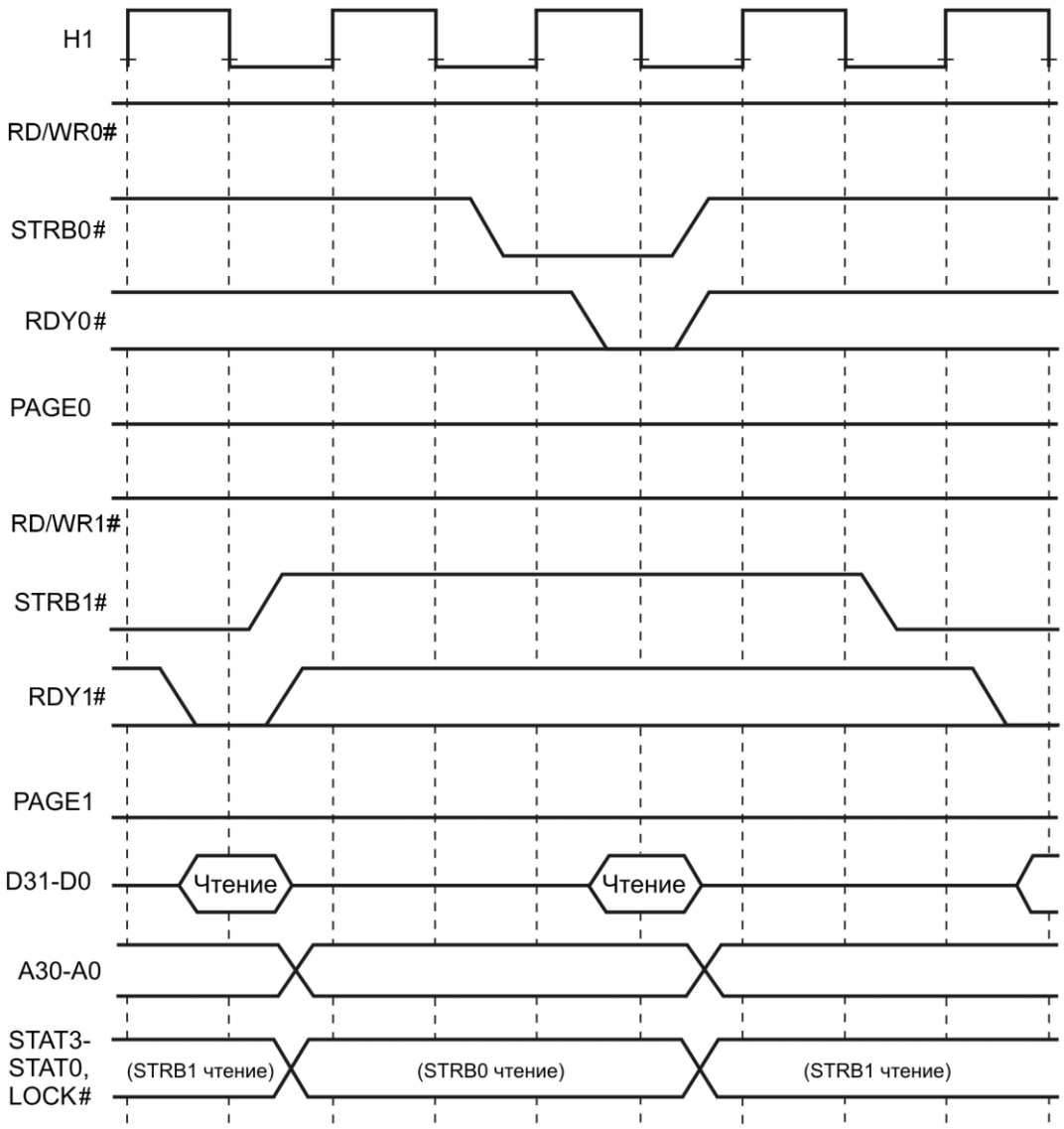


Рисунок 9.22 – Последовательность «чтение в одной странице по STRB1#, STRB0#, STRB1#» при STRB SWITCH = 1

Рисунок 9.23 аналогичен рисунку 9.22 за исключением того, что второе чтение по STRB1# происходит в другой странице.

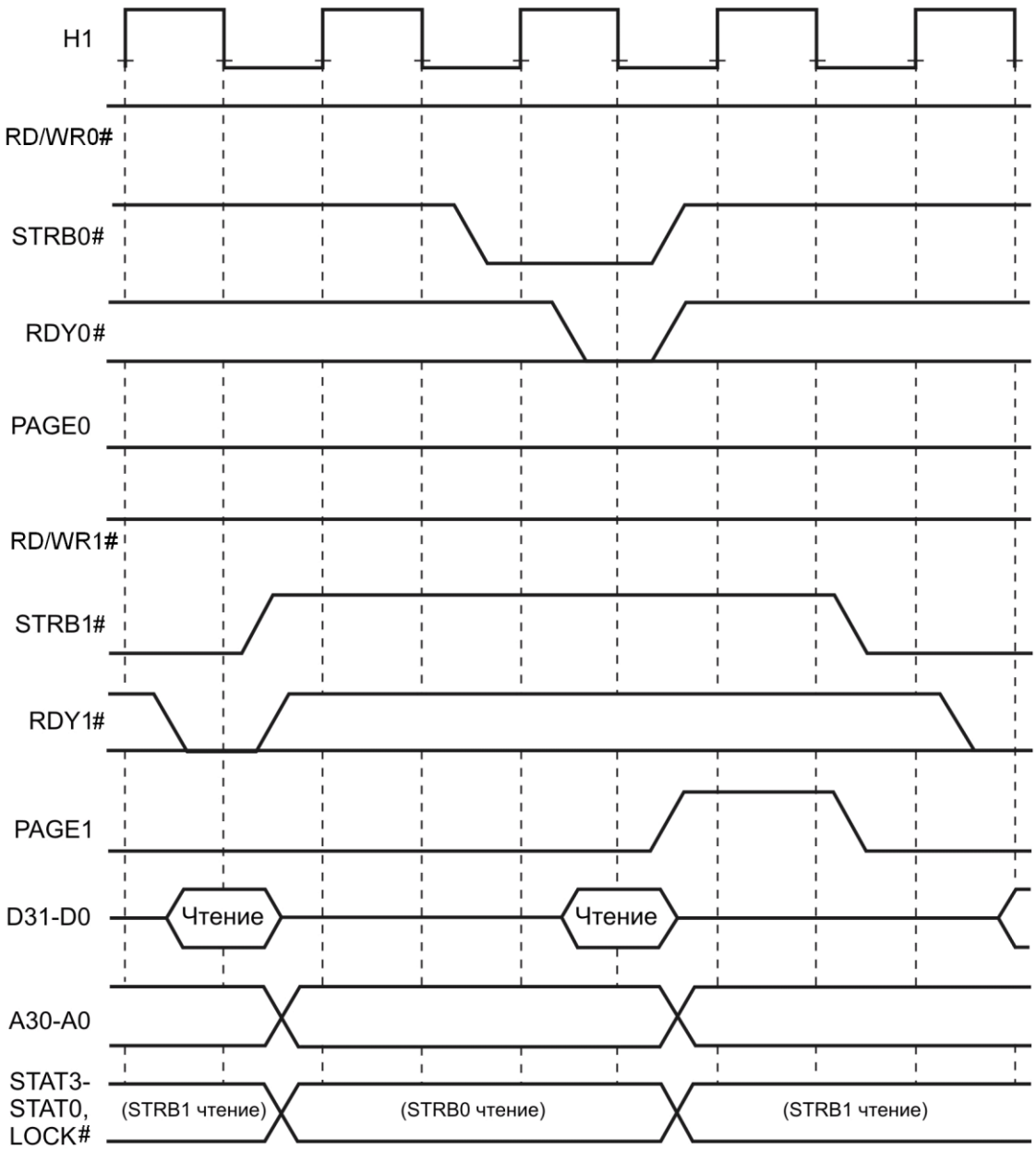


Рисунок 9.23 – Последовательность «чтение в одной странице по STRB1#, STRB0#, чтение другой странице по STRB1#» при STRB SWITCH = 1

На рисунке 9.24 приведена временная диаграмма последовательности «запись в одной странице по STRB1#, STRB0#, чтение в той же странице по STRB1#».

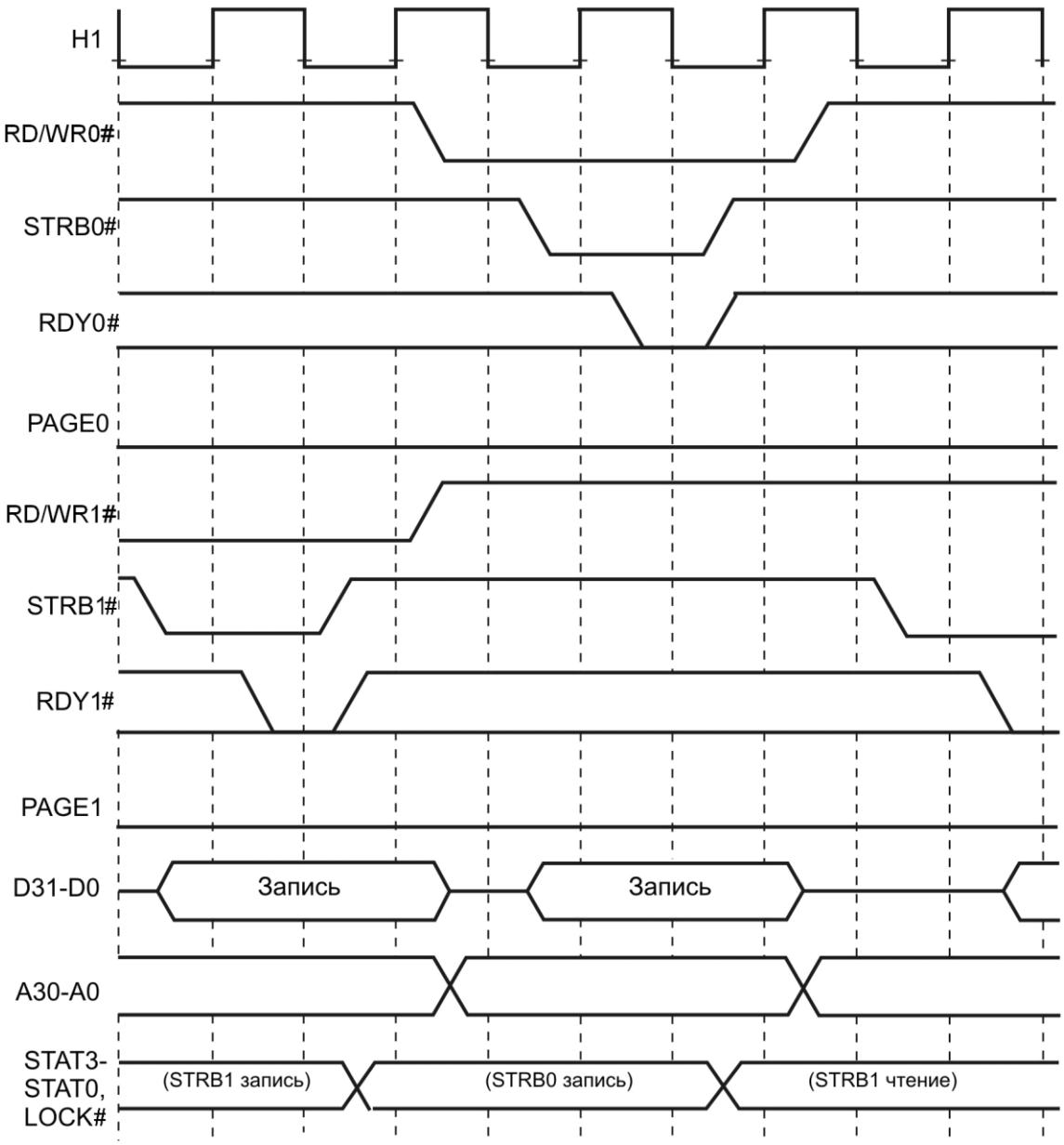


Рисунок 9.24 – Последовательность «запись в одной странице по STRB1#, STRB0#, чтение в той же странице по STRB1#»

На рисунке 9.25 приведена временная диаграмма операции чтения с одним циклом ожидания.

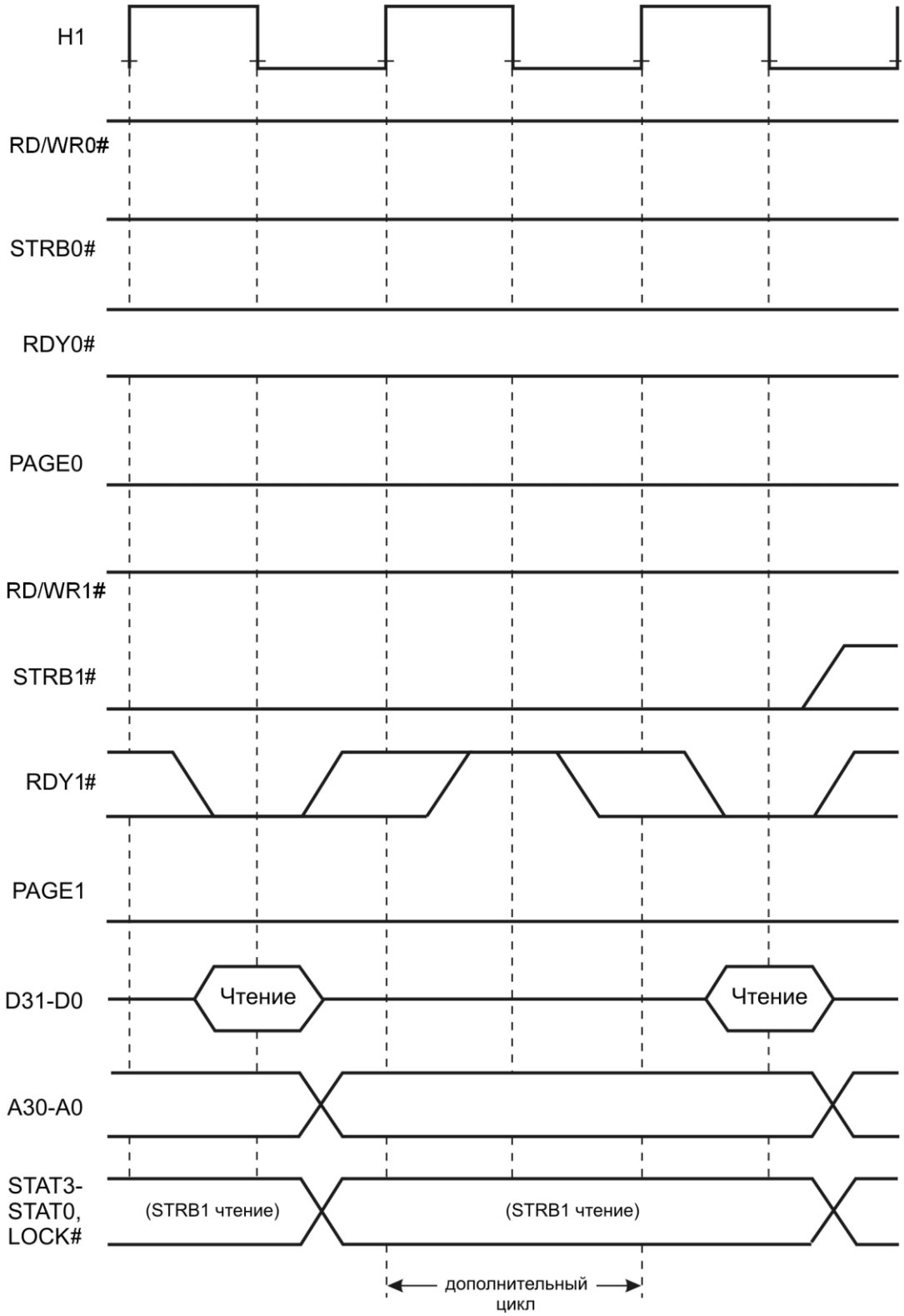


Рисунок 9.25 – Операция чтения с одним циклом ожидания

На рисунке 9.26 приведена временная диаграмма операции записи с одним циклом ожидания.

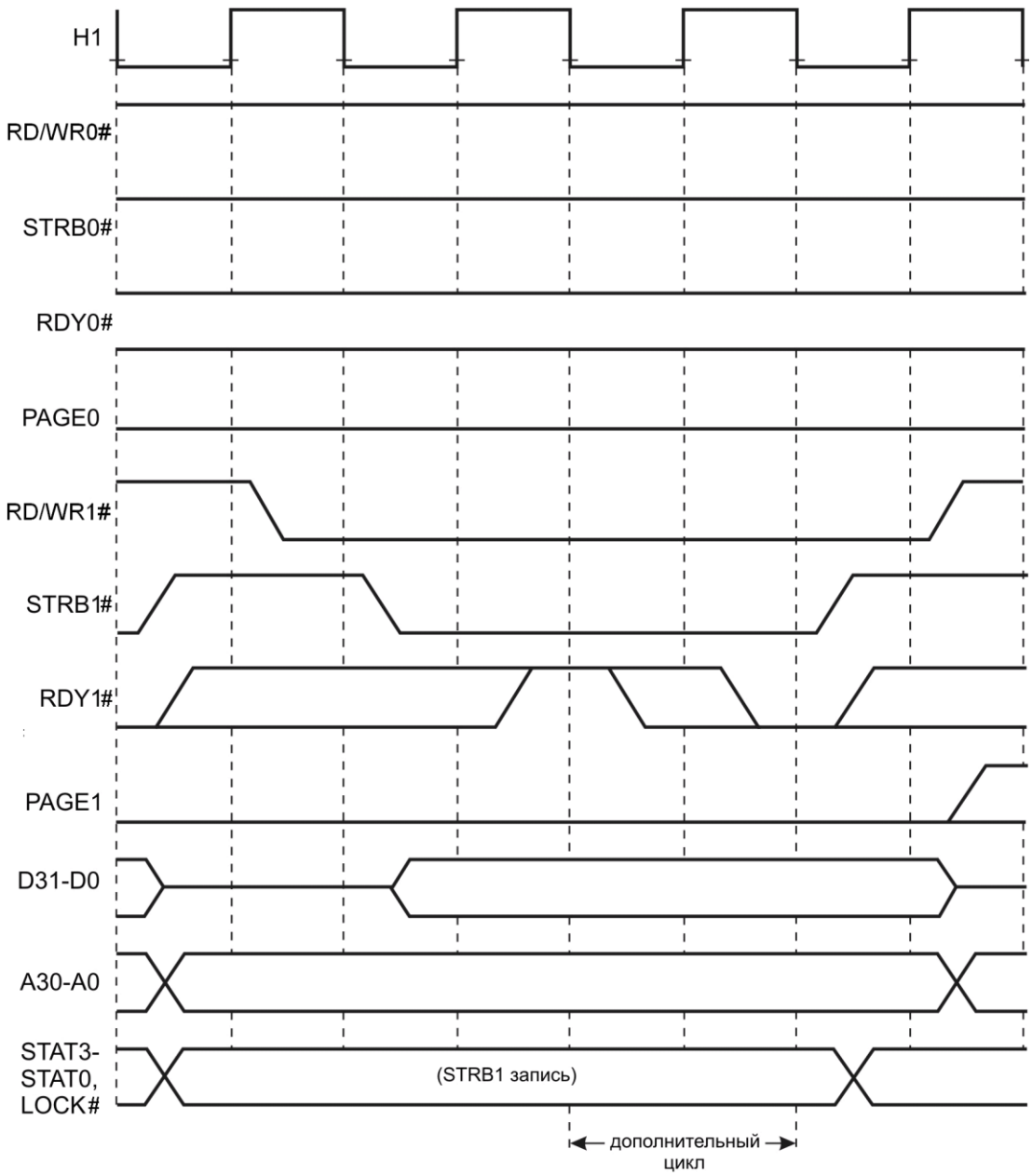


Рисунок 9.26 – Операция записи с одним циклом ожидания

9.6 Использование сигналов разрешения для управления группами сигналов

На рисунке 9.27 приведено управление разрешающим сигналом соответствующей группы сигналов.

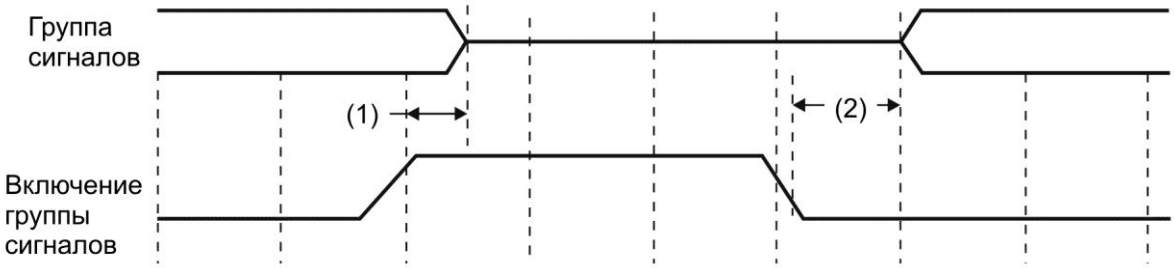


Рисунок 9.27 – Использование сигналов разрешения для установки сигнальных групп в состояние высокого импеданса

Например, сигнал DE# управляет сигналами данных внешнего глобального интерфейса. Сигналы разрешения – не синхронные по отношению к тактирующему сигналу входы, которые выключают соответствующие выходные буферы. После перехода сигнала разрешения в высокий уровень плюс время (1) на рисунке 9.27, соответствующая группа сигналов переходит в высокоимпедансное состояние. Затем после перехода сигнала разрешения в низкий уровень плюс время (2) на рисунке 9.27, группа сигналов выходит из высокоимпедансного состояния. Если сигнальная группа уже была в высокоимпедансном состоянии перед тем, как сигнал разрешения перешел в высокий уровень, то группа выйдет из высокоимпедансного состояния (когда сигнал разрешения станет низкого уровня), только если это потребуется. Например, шина данных, которая до этого не управлялась, после включения сигнала разрешения будет управляться, если ожидается обращение к шине данных.

Примечание – Если планируется использовать внутренне генерируемые состояния ожидания, то необходимо проверять, чтобы данные не читались с шины и не записывались в нее, когда она отключена. При внутренне генерируемых состояниях ожидания шина может быть в режиме высокоимпедансного состояния. В этом случае записанные данные не будут доступны из вне, а читаемые данные, в любом случае, будут иметь значение высокого импеданса.

9.7 Операции блокировки

Разделение общей памяти между процессорами, является одним из большинства решений организации параллельных систем. В многопроцессорной системе доступ к глобальной памяти со стороны процессоров организован одинаково, при этом необходимо обращение к блоку арбитража и подтверждение захвата разделяемой памяти. Подробнее об этом описано в 9.7.5.

Пять инструкций ИС 1867ВМ9Ф выполняют операции блокировки. Используя внешние сигналы, эти инструкции обеспечивают мощные механизмы синхронизации. Они также гарантируют целостность коммуникации и обеспечивают быстрые операции с разделяемыми ресурсами. Группа инструкций блокировки представлена в таблице 9.7.

Таблица 9.7 – Операции блокировки

Инструкция	Описание	Операция
LDFI	Загрузка значения с ПЗ из памяти в регистр; установка сигнала блокировки LOCK# (LLOCK#) при доступе к внешней памяти	Сигнал блокировки src → dst
LDII	Загрузка целого из памяти в регистр; установка сигнала блокировки LOCK# (LLOCK#) при доступе к внешней памяти	Сигнал блокировки src → dst
SIGI	Загрузка значения с ПЗ из памяти в регистр; сброс сигнала блокировки LOCK# (LLOCK#) при доступе к внешней памяти	Сигнал блокировки Сброс блокировки
STFI	Сохранение значения с ПЗ из регистра в память; сброс сигнала блокировки LOCK# (LLOCK#) при доступе к внешней памяти	src → dst Сброс блокировки
STII	Сохранение целого из регистра в память; сброс сигнала блокировки LOCK# (LLOCK#) при доступе к внешней памяти	src → dst Сброс блокировки

Операции блокировки используют сигналы LOCK# глобальной и LLOCK# локальной шины для отражения текущей выполняемой операции блокировки. Эти сигналы становятся активными (низкий уровень), когда выполняется какая-либо инструкция блокировки из таблицы 9.7.

Временная диаграмма записи с блокированием и сохранением такая же, как и для стандартных записей и сохранений. Можно увеличить время записи и сохранения стандартными обращениями, используя соответствующие сигналы RDYx# или LRDYx#.

9.7.1 Инструкции LDFI и LDII

Инструкции LDFI и LDII выполняют следующее:

- переводят LOCK# (LLOCK#) в низкий уровень;
- выполняют LDF или LDI инструкции;
- увеличивают цикл чтения, пока не придет соответствующий сигнал готовности;
- выполняют операцию;
- оставляют LOCK# (LLOCK#) активным в низком уровне, пока он не будет изменен с помощью STFI, STII, SIGI.

Операции чтения/записи аналогичны другим циклам чтения/записи, исключая специальное использование LOCK# (LLOCK#). Операнд src для LDFI и LDII всегда имеет прямой или косвенный адрес. LOCK# (LLOCK#) устанавливается в «0», только если src расположен вне ИС 1867ВМ9Ф, т. е. STRB# или LSTRB# активны. Если имеет место доступ к внутренней памяти, то LOCK# (LLOCK#) не устанавливается, и операция проходит как LDF или LDI из внутренней памяти.

9.7.2 Инструкции STFI и STII

Инструкции STFI и STII выполняют следующее:

- начинается цикл записи. Состояние LOCK# (LLOCK#) не изменяется. Если он в низком уровне, происходит операция блокировки. Если он в высоком уровне, то операция выполняется как STF или STI (без блокировки);
- выполняется инструкция STF или STI и добавляется цикл записи, пока не придет соответствующий сигнал готовности;

- после цикла записи LOCK# (LLOCK#) становится неактивным (высокий уровень).

Как и в случае с LDFI и LDII, dst инструкции STFI и STII влияют на LOCK# (LLOCK#). Если dst расположен вне ИС 1867ВМ9Ф, т. е. STRB0#, STRB1# (LSTRB0#, LSTRB1#) активны, то LOCK# (LLOCK#) устанавливается в «1». Если доступ происходит к внутренней памяти, то LOCK# (LLOCK#) не устанавливается, и операция выполняется как STF или STI во внутреннюю память.

9.7.3 Инструкция SIGI

Инструкция SIGI может использоваться по-разному. В некоторых приложениях, можно модифицировать семафоры внешним образом, а можно логикой специального назначения. Если это так, то SIGI может использоваться для выполнения одноцикловой блокировки семафора. Инструкция SIGI также может использоваться просто для выполнения внешнего чтения и для сообщения, что соответствующее место в программе достигнуто.

Инструкция SIGI выполняет следующее:

- переводит LOCK# (LLOCK#) в низкий уровень;
- выполняет LDI инструкцию;
- увеличивает цикл чтения, пока не придет соответствующий сигнал готовности;
- выполняет операцию;
- переводит LOCK# (LLOCK#) обратно в неактивное состояние (высокий уровень).

Операции блокировки могут использоваться для выполнения циклов ожидания-занятости, для управления многопроцессорным счетчиком, для выполнения простого механизма семафора или для синхронизации между двумя ИС 1867ВМ9Ф. Следующие примеры показывают полезность операций блокировки.

9.7.4 Примеры использования механизма блокировки

Примеры в этом разделе показывают, как использовать операции блокировки.

Пример 9.1 – Цикл ожидания-занятости для программной синхронизации процессоров.

Пример 9.2 – Счетчик, распределенный между совокупностью процессоров, определяющий число выполнений заданий процессорами.

Примеры 9.3 и 9.4 – Семафор для программирования критических секций.

Пример 9.1 показывает применение цикла ожидания-занятости. ИС 1867ВМ9Ф остается в цикле до тех пор, пока другой процессор не запишет «0» в ячейку @LOCK (не сбросит сигнал блокировки). Если в ячейке LOCK размещена переменная, отражающая состояние блокировки для критической секции программы, и не ноль в этой ячейке означает, что критическая секция блокирована, то алгоритм для цикла ожидания-занятости может использоваться, как показано в примере.

Пример 9.1 – Цикл ожидания-занятости

```
        LDI 1, R0          ; помещает 1 в R0
L1: LDII @LOCK, R1 ; загружает значение ячейки LOCK в R1
STII R0, @LOCK       ; устанавливает значение блокировки в 1
BNZ L1                     ; если R1 (предыдущее значение блокировки) не 0,
                           ; то R1 считывается снова
```

Пример 9.2 показывает, как ячейка COUNT содержит счетчик, подсчитывающий количество выполнений конкретной операции. Эта операция может выполняться любым процессором системы. Если счет равен «0», то процессор ожидает, пока ста-

нет не равным «0» перед началом некоторого вычисления. Этот пример показывает, как корректно изменять COUNT.

Пример 9.2 – Изменение счетчика заданий

```
LDI 0, R0
WAIT LDII   @COUNT, R1      ; чтение текущего значения счетчика
BZD WAIT                    ; если COUNT = 0, пробовать снова
LDNZ        1, R0           ; если COUNT не 0, декрементировать его
SUBI R0, R1
STII  R1, @COUNT            ; обновить COUNT
```

На рисунке 9.28 показано, как две ИС 1867ВМ9Ф распределяют между собой глобальную память и используют инструкции блокировки, приведенные в примерах 9.3, 9.4.

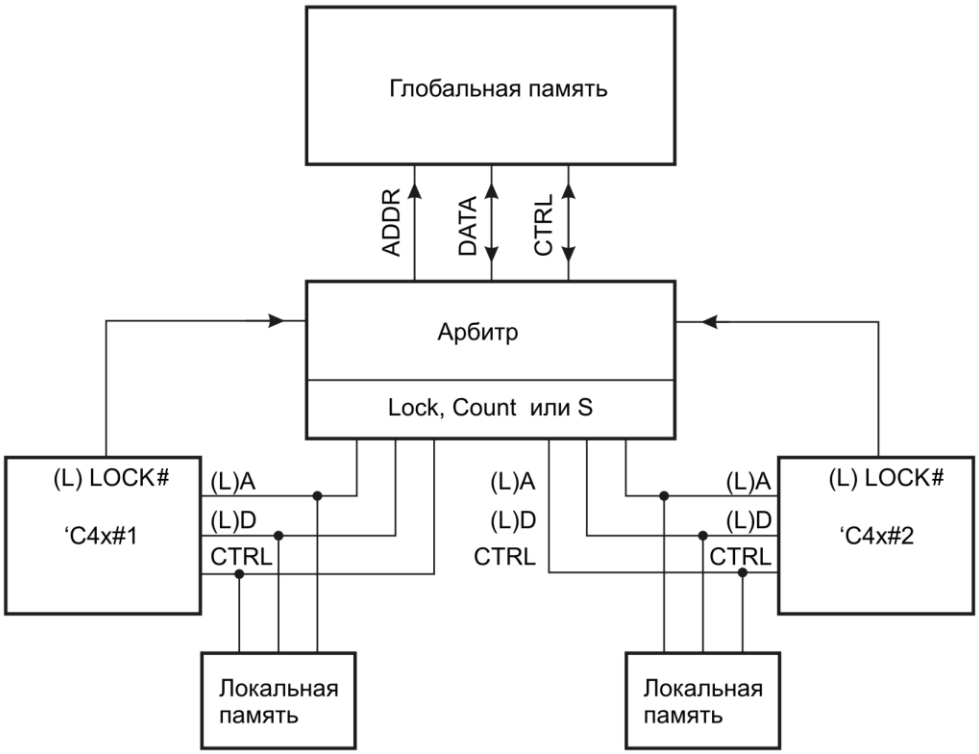


Рисунок 9.28 – Система микропроцессоров ИС 1867ВМ9Ф с распределенной между ними глобальной памятью

Пример 9.3 – Реализация операции V(S) – увеличение семафора на «1»

```
V: LDII @S, R0
ADDI        1, R0
STII  R0, @S        ; S + 1 " S
```

Пример 9.4 – Реализация операции P(S) – уменьшение семафора на «1»

```
LDI 0, R0
P:  LDII @S, R1     ; чтение текущего значения семафора
BZD P               ; если S = 0, переход к P и пробовать снова
LDNZ 1, R0          ; если S не 0, декрементировать его
SUBI R0, R1
STII R1, @S         ; модификация S
```

В мультипроцессорных системах нескольким процессорам необходимо обратиться к распределенным между ними данным или другим общим ресурсам. Часть программы, которая обращается к разделяемым ресурсам, называется критической секцией.

Для упрощения программирования критических секций необходимо использовать семафоры. Семафоры – это переменные, которые синхронизируют доступ к общим ресурсам. Они могут принимать только неотрицательные целочисленные значения.

Две простые неделимые операции, производящиеся над семафорами, где S – семафор, это:

- Операция увеличения семафора на «1» V(S)

$S + 1 \rightarrow S$.

- Операция уменьшения семафора на «1» P(S):

P: Если (S = 0), то

перейти к P.

Иначе $S - 1 \rightarrow S$.

Неделимость V(S) и P(S) означает, что когда происходит обращение к разделяемому ресурсу, то чтение и изменение семафора, выполняется как единая команда, не смотря на то, что реально она состоит из нескольких инструкций. Для обеспечения неделимости инструкций чтения и модификации семафора необходимо закрывать все прерывания перед выполнением этих операций.

Перед входом в критическую секцию проверяется состояние семафора, если он не равен «0», то выполняется операция P над общим семафором, например, S (S инициализируется в «1»). Процессор, выполнивший операцию P(S), может войти в критическую секцию. Все остальные процессоры блокируются, так как S переходит в «0». После выхода из критической секции процессор выполняет V(S), таким образом, позволяя другому процессору выполнить операцию P(S). Код ИС 1867ВМ9Ф для V(S) показан в примере 9.3, код для P(S) показан в примере 9.4. Сравните код в примере 9.4 с кодом в примере 9.2, который не использует семафоры.

9.7.5 Временные диаграммы сигналов шины и блокировки шины

Временные диаграммы для сигналов LOCK# и LLOCK# такие же, как и для STAT3-STAT0 и LSTAT3-LSTAT0. Инструкции LDII, LDFI, STII, STFI, SIGI управляют сигналами блокировки шины, только при обращении к внешней памяти. Инструкции LDII, LDFI, SIGI устанавливают LOCK# или LLOCK# в низкий уровень в начале цикла чтения по срезу H1, а инструкции STII, STFI, SIGI устанавливают LOCK# или LLOCK# в высокий уровень в конце цикла доступа по срезу H1. Инструкции блокировки объяснены в подразделе 9.7. Рисунки 9.29 – 9.32 показывают временные диаграммы шины при доступе к внешней шине с использованием инструкций STII, LDII, STFI, STFI и SIGI.

На рисунке 9.29 представлен пример обращения инструкций LDII или LDFI к внешней шине.

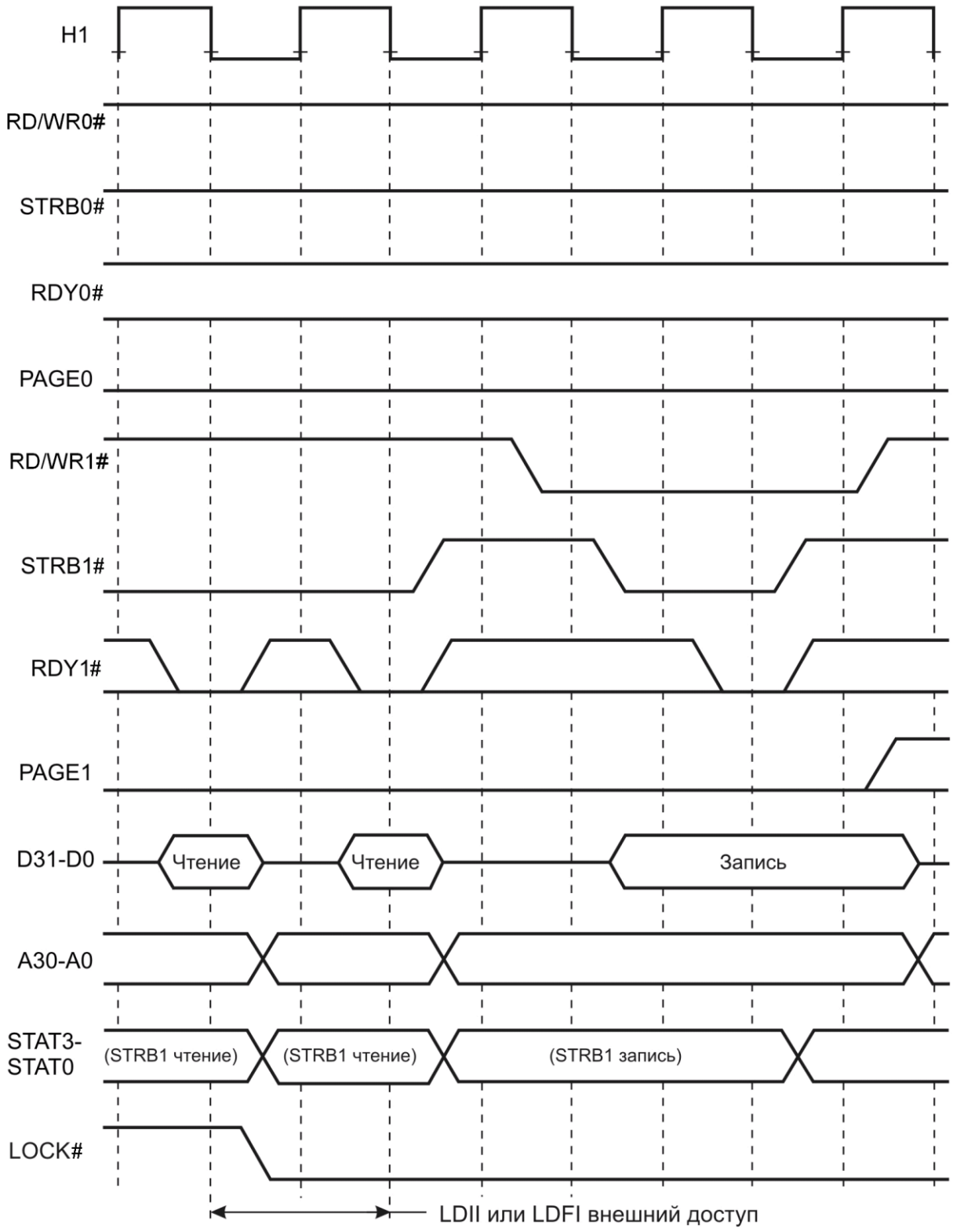


Рисунок 9.29 – Обращение к внешней шине инструкциями LDII или LDFI

Рисунок 9.30 – это пример обращения к внешней шине инструкциями LDII или LDFI и STII или STFI, следующими за предшествующей записью (рисунок 9.29), и цикл ожидания (бездействие).

Это временная диаграмма для последовательности блокирующих загрузок/ сохранений.

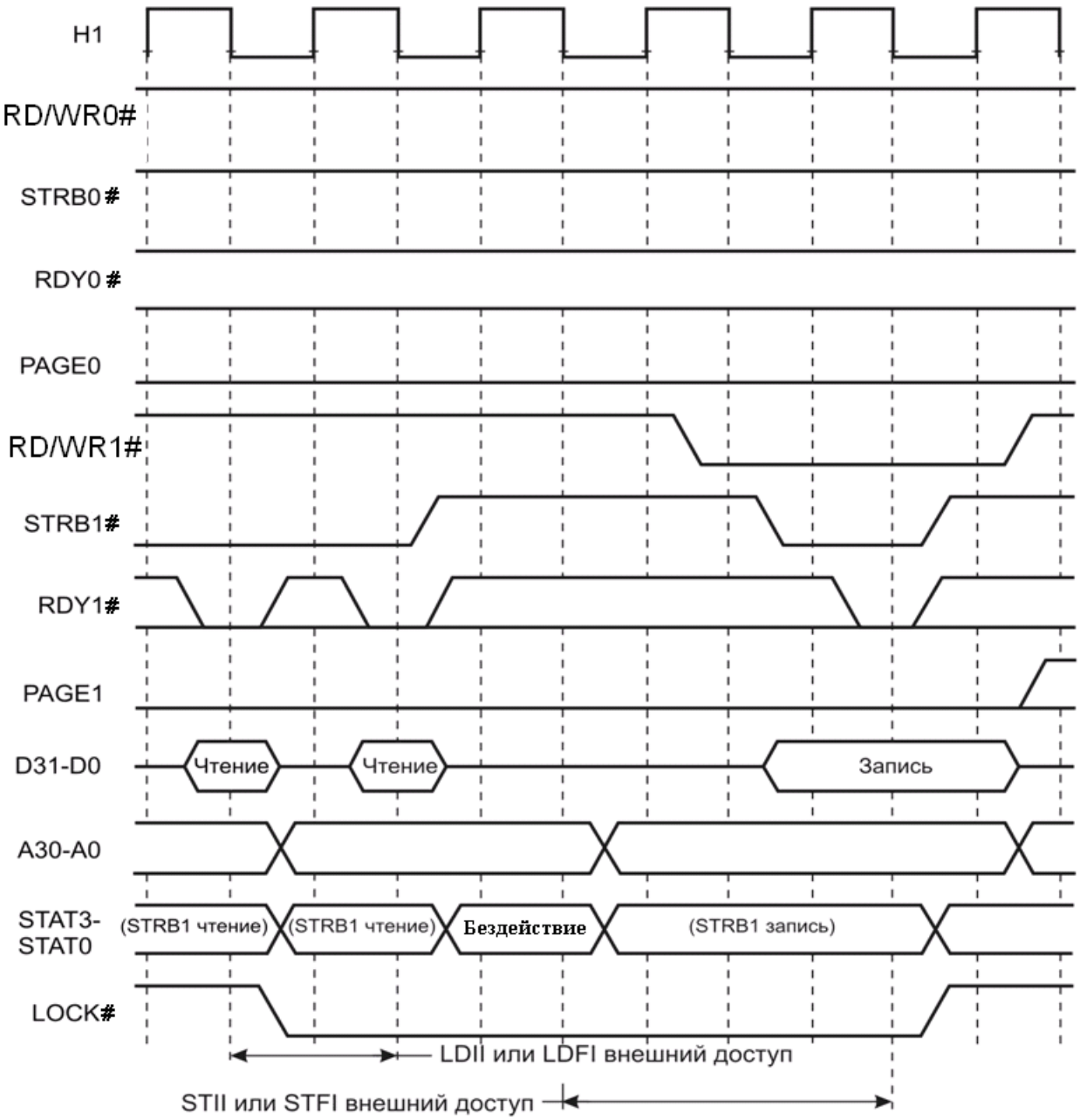


Рисунок 9.30 – Обращение к внешней шине инструкциями LDII или LDFI и STII или STFI и цикл ожидания (бездействие)

На рисунке 9.31 приведена временная диаграмма выполнения инструкции SIGI при доступе к внешней шине.

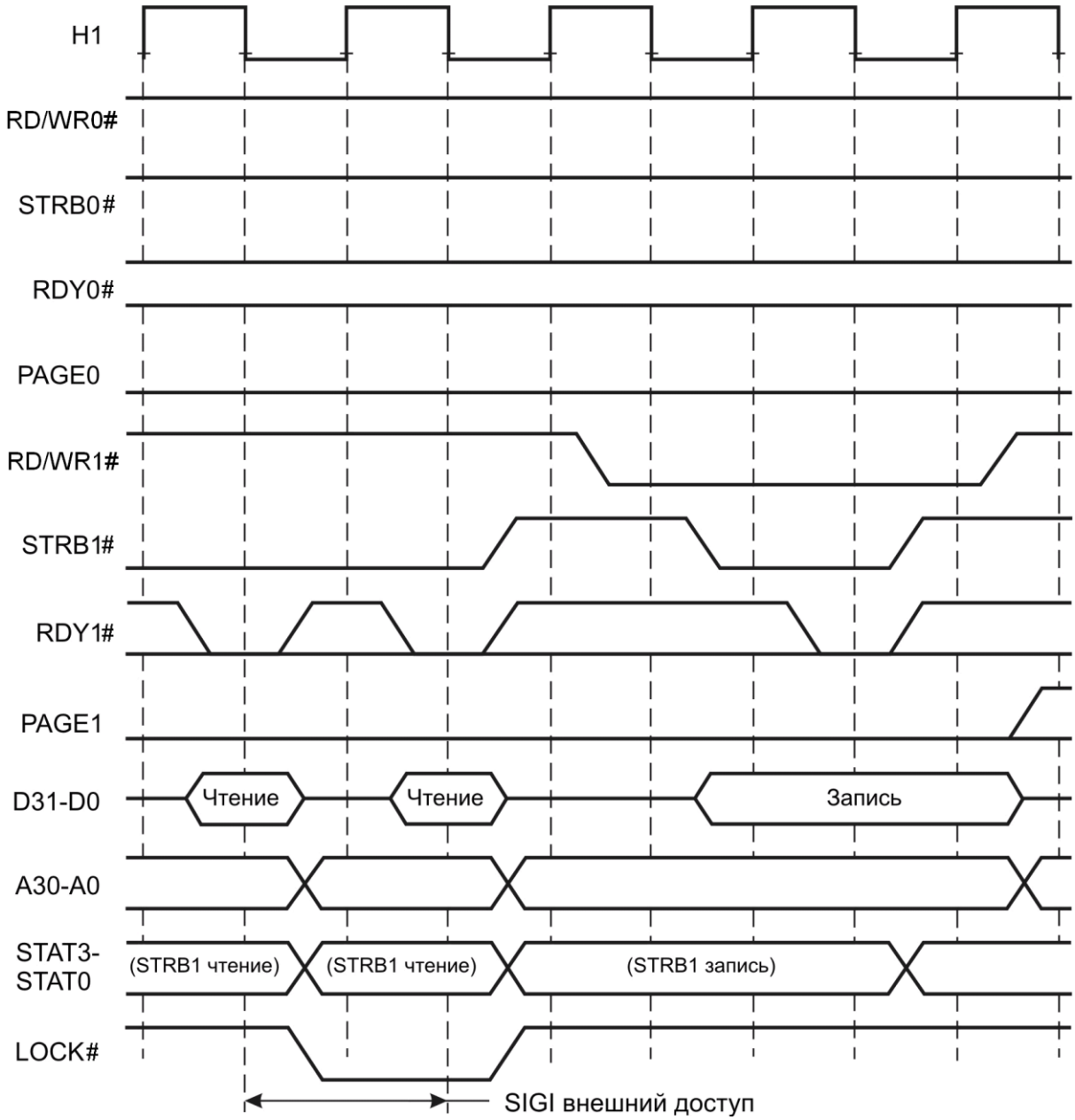


Рисунок 9.31 – Временная диаграмма обращения SIGI к адресу на внешней шине

Рисунок 9.32 показывает временную диаграмму для инструкции SIGI, если сигнал LOCK# уже находится в низком уровне. Это может произойти, если SIGI следует за инструкцией LDII. Так как LOCK# уже в низком уровне, то единственное, что может выполнить SIGI – это перевести его в высокий уровень.

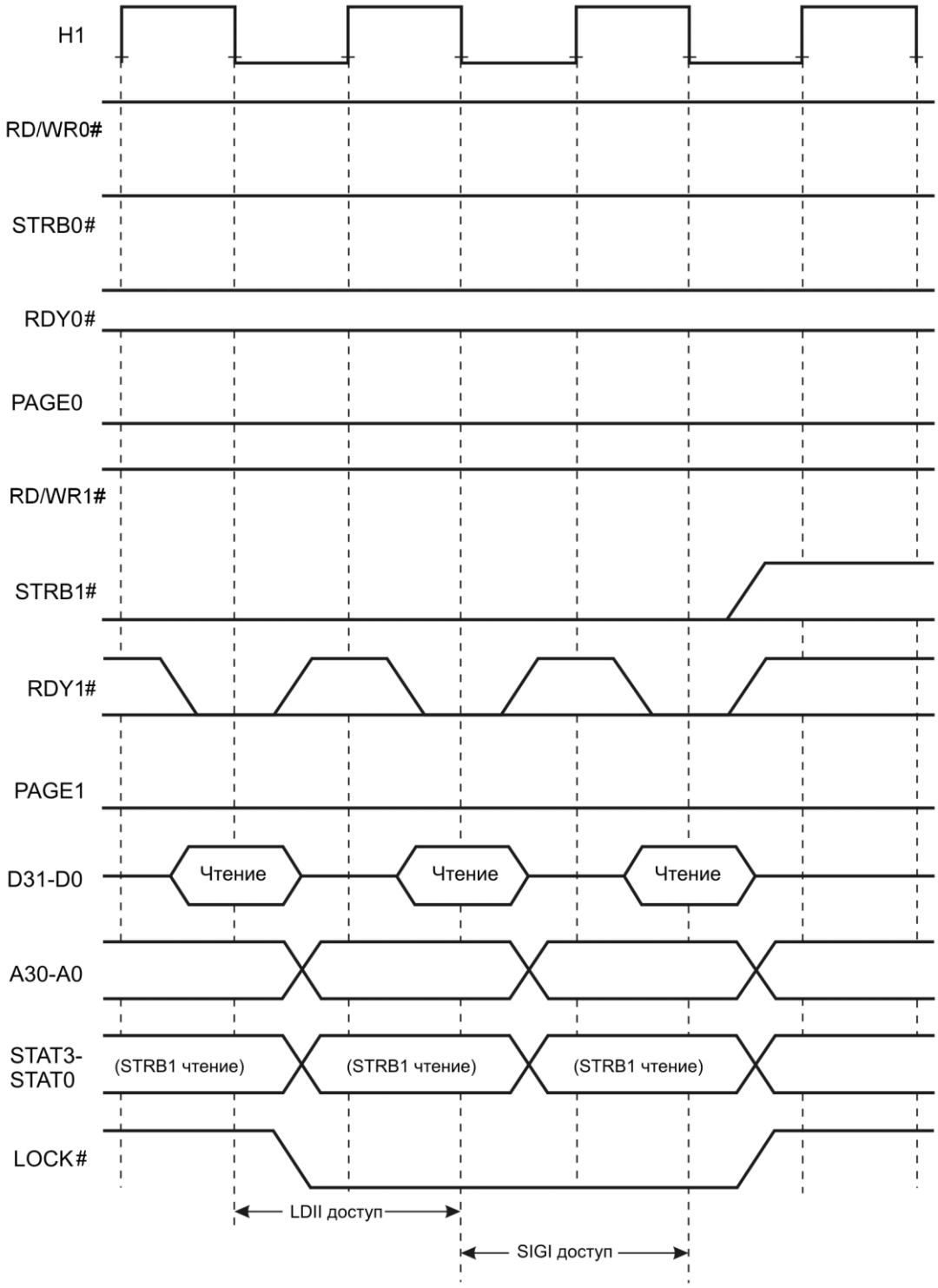


Рисунок 9.32 – Временная диаграмма для инструкции SIGI, если LOCK# уже в низком уровне

9.8 Временная диаграмма для формирования сигнала IACK#

Сигнал IACK# управляется инструкцией IACK. Его временная диаграмма аналогична сигналу LOCK#, когда используется инструкция SIGI. Поведение IACK# похоже на поведение LOCK# или STATx (где x принимает значения от 3 до 0). Различие состоит в том, что существует только один сигнал IACK#.

Временная диаграмма для формирования сигнала IACK# показана на рисунке 9.33.

Как и при операциях блокировки, инструкция IACK воздействует на вывод IACK# только при доступе на внешнюю шину.

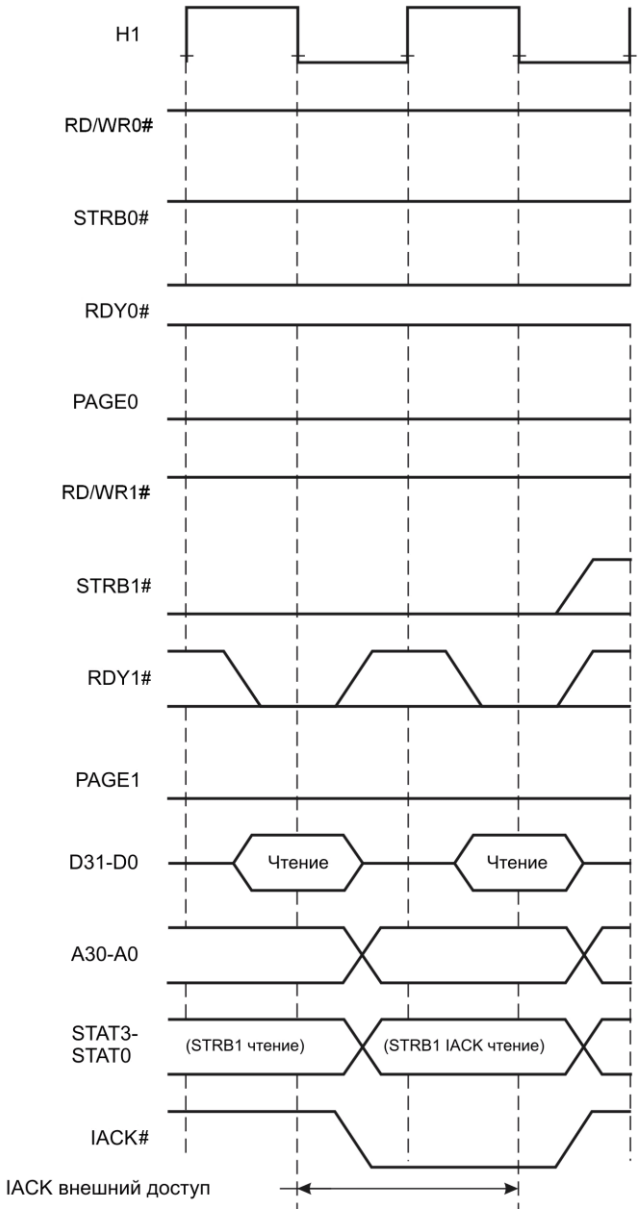


Рисунок 9.33 – Временная диаграмма для формирования сигнала IACK#

10 Первоначальный загрузчик

Загрузчик ИС 1867ВМ9Ф (далее загрузчик) может загрузить и выполнить программы, которые либо были приняты от главного микропроцессора (в мультипроцессорной системе), либо считаны из ПЗУ, либо из внешней памяти. Главная функция загрузчика, как и любого загрузчика, это загрузка программ из памяти или через коммуникационные порты после системного сброса ИС.

10.1 Описание загрузчика

Начало программы загрузчика размещается с адреса 0x0000_01bc 0x0000_11bc и находится во внутреннем ПЗУ ИС. Программа загрузчика приведена в подразделе 10.7.

10.2 Выбор режима загрузки

Функцией загрузчика является, прежде всего, загрузка программы из памяти или коммуникационного порта. Выбор режима работы загрузчика определяется выводами IIOF3#-IIOF0#. Выбор режима работы загрузчика, в зависимости от этих выводов, представлен в таблице 10.1 и показан на рисунке 10.1. Существует два вида загрузки:

- загрузка из внешней памяти. При этом режиме загрузки – загрузчик выполняет загрузку исходной программы из блока памяти, следующих разрядностей: 8 бит, 16 бит или 32 бита. Исходные программы для загрузки должны постоянно находиться в одной из шести предопределенных адресных областях внешней памяти, которые перечислены в таблице 10.1. Стробирование загружаемых программ должно производиться сигналами STRB0# (LSTRB0#), потому что именно эти стробы активны после системного сброса. Рисунок 10.2 показывает порядок установки режима работы ИС в ходе работы загрузки из внешней памяти;

- загрузка через коммуникационные порты. Загрузчик ждет первого ввода данных от одного коммуникационного порта из шести. Формат поступающего потока данных является подобным потоку данных при загрузке из внешней памяти, за исключением того, что нет шага выбора разрядности входных данных. Рисунок 10.3 показывает порядок установки режима работы ИС в ходе загрузки исходной программы через коммуникационные порты.

Таблица 10.1 – Выбор режима загрузчика в зависимости от значения выводов IIOF3#-IIOF0#

Внешние выводы				Стартовый адрес загрузки
IIOF3#	IIOF2#	IIOF1#	IIOF0#	
1	1	0	1	0030 0000h
1	0	1	1	4000 0000h
1	0	0	1	60000000h
0	1	1	1	8000 0000h
0	1	0	1	A000 0000h
0	0	1	1	C000 0000h
0	0	0	1	Зарезервировано
1	1	1	1	Коммуникационные порты
0	1	0	1	A000 0000h
0	0	1	1	C000 0000h
0	0	0	1	Зарезервировано
1	1	1	1	Коммуникационные порты

10.3 Порядок работы загрузчика

В подразделе приведена общая последовательность шагов для инициализации загрузчика, который загружает исходную программу:

- Установка выводов RESETLOC (1,0) в низкий уровень.

- Установка вывода ROMEN в высоком уровне.

Примечание – Вывод ROMEN должен быть в высоком уровне в течение всей работы загрузчика и может быть изменен в любое время после выполнения работы загрузчика.

- Установка вывода IIOF0# в высокий уровень.

- Установка выводов IIOF3#-IIOF1# в соответствующее состояние.

Вывод ROMEN в высоком уровне разрешает доступ к внутреннему ПЗУ. Состояние внешних выводов IIOF3#-IIOF1# указывает, в каком адресном пространстве находится загружаемая программа. Эти варианты были перечислены в таблице 10.1. Выводы IIOF3#-IIOF1# читаются как флаги IIOF регистра IIF. Загрузчик выполняет следующие шаги для определения того, где размещена исходная программа, показанная на рисунке 10.1:

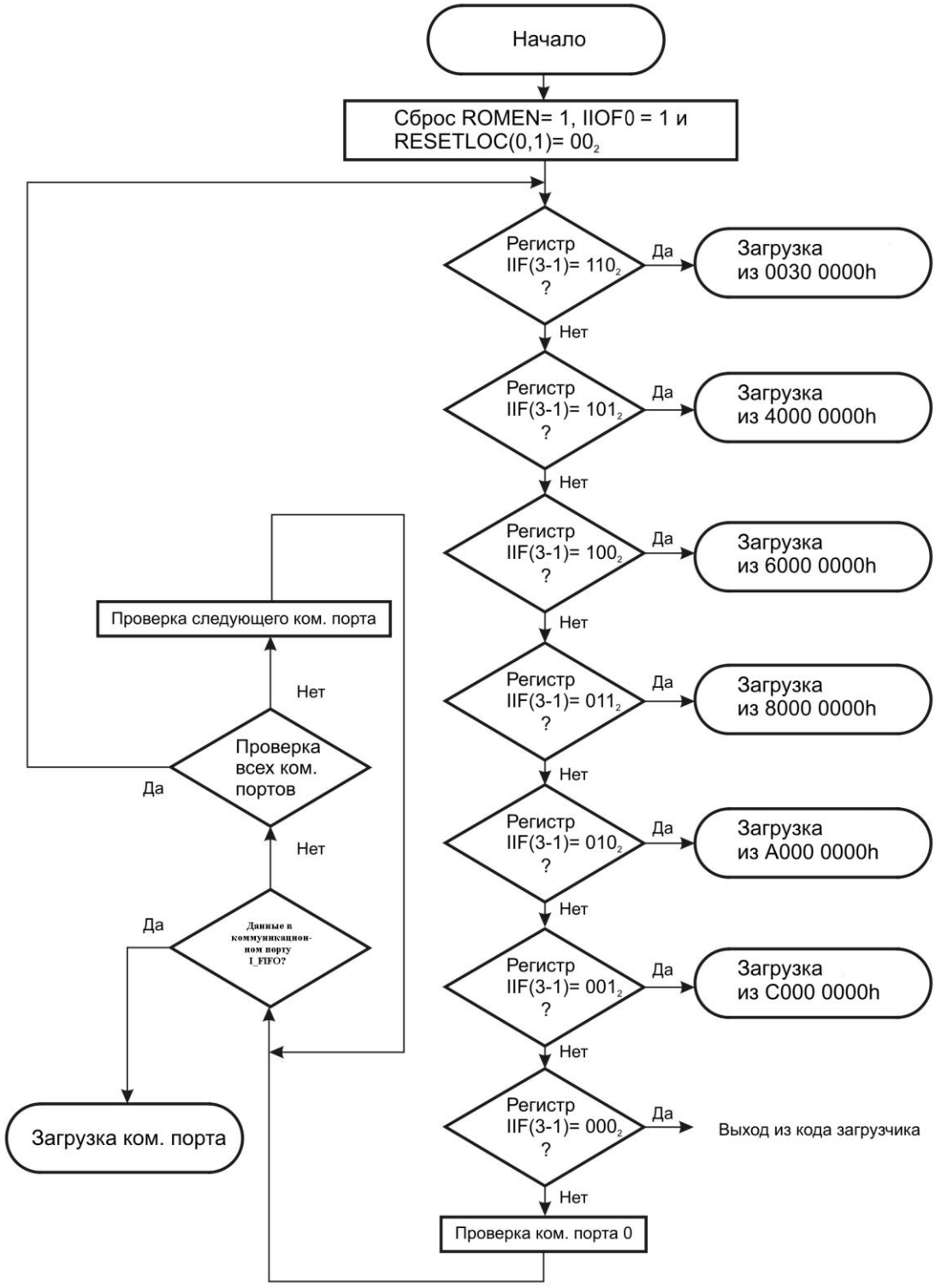
- Если значение битов (3-1) регистра IIF имеет значение от 110_2 до 001_2, то программа загружается из соответствующего адреса памяти, который указан в таблице 10.1, см. рисунок 10.2 для детального рассмотрения работы загрузчика в режиме загрузки из внешней памяти.

- Если значение битов (3-1) регистра IIF имеет значение 000_2, то это режим зарезервирован. Этот режим использовать нельзя.

- Если ни одна из комбинаций 000_2-110_2 не определяется, то загрузчик переходит в режим загрузки через коммуникационные порты и начинает проверять входные каналы коммуникационных портов, начиная с нулевого и заканчивая пятым. Если загрузчик не обнаруживает никаких входных данных от коммуникационных портов, то программа загрузчика возвращается к проверке состояния выводов IIOF3#-IIOF1# снова, см. рисунок 10.3 для детального рассмотрения работы загрузчика в режиме загрузки от коммуникационных портов.

- Когда данные исходной программы найдены, то программа загружается с адреса, используя разрядность, указанную в первом слове (8, 16 или 32 бита). Загрузчик не может загрузить исходную программу с любого адресного пространства адреса, значение которого меньше 0000 1000h, если адрес не находится в карте памяти, то происходит повторное декодирование. Первые пять слов исходной программы определяют условия загрузки и выполнения программы и должны отвечать определённым критериям. Оставшиеся слова служат для вспомогательной установки ИС. Для детального рассмотрения процесса загрузки используется таблица 10.2. Далее выполняется инструкция IACK, указывая завершение работы загрузчика. Это может потребоваться, чтобы переключить режим микрокомпьютера (ROMEN = 1) в режим микропроцессора (ROMEN = 0).

- Далее происходит выполнение загружаемой программы (точкой входа будет являться первое слово загруженной программы).



Примечание – Принятое условное сокращение:
- «ком.» – коммуникационный (порт).

Рисунок 10.1 – Алгоритм выбора режима работы загрузчика в зависимости от значений выводов IIOF3#-IIOF0#

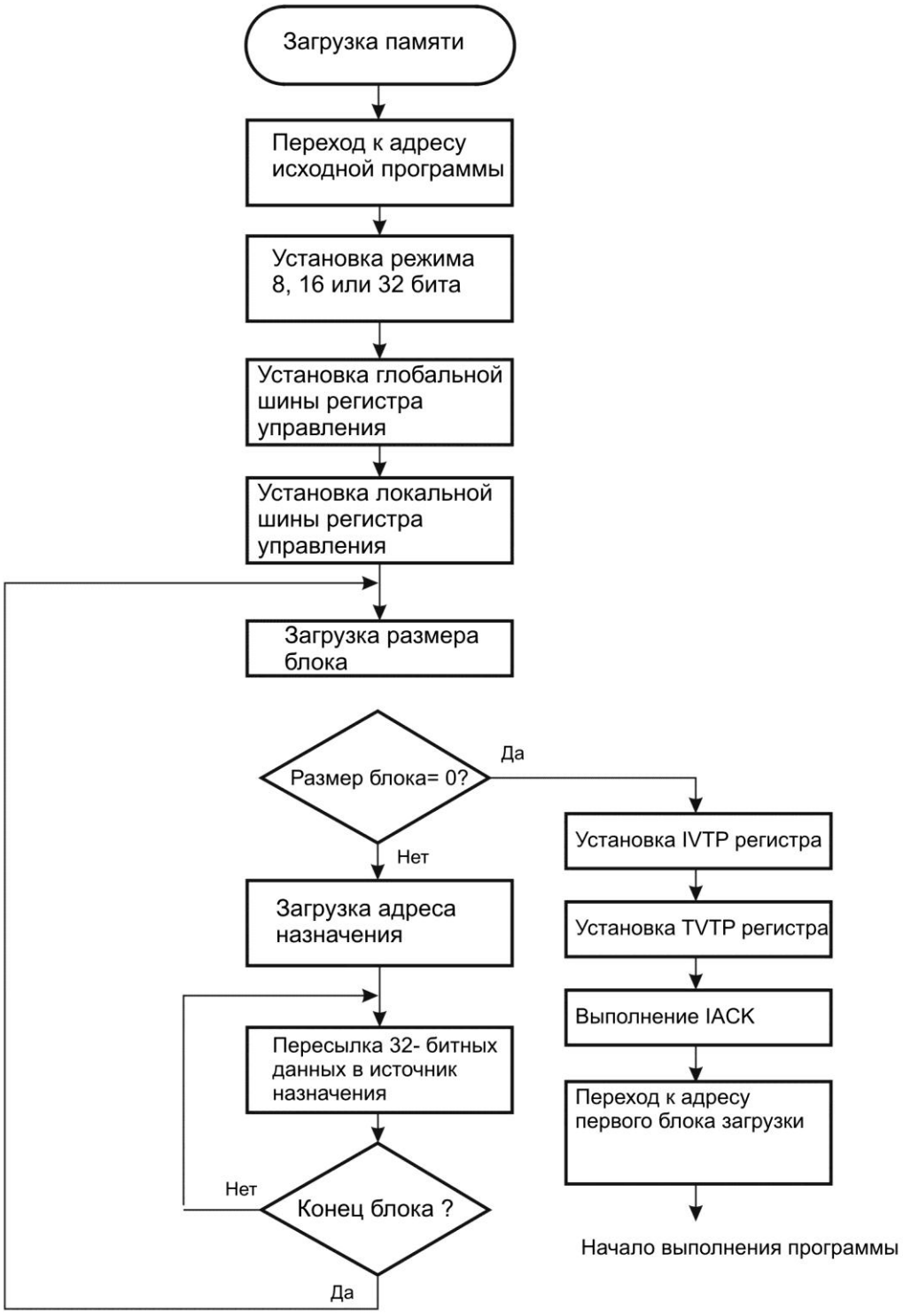
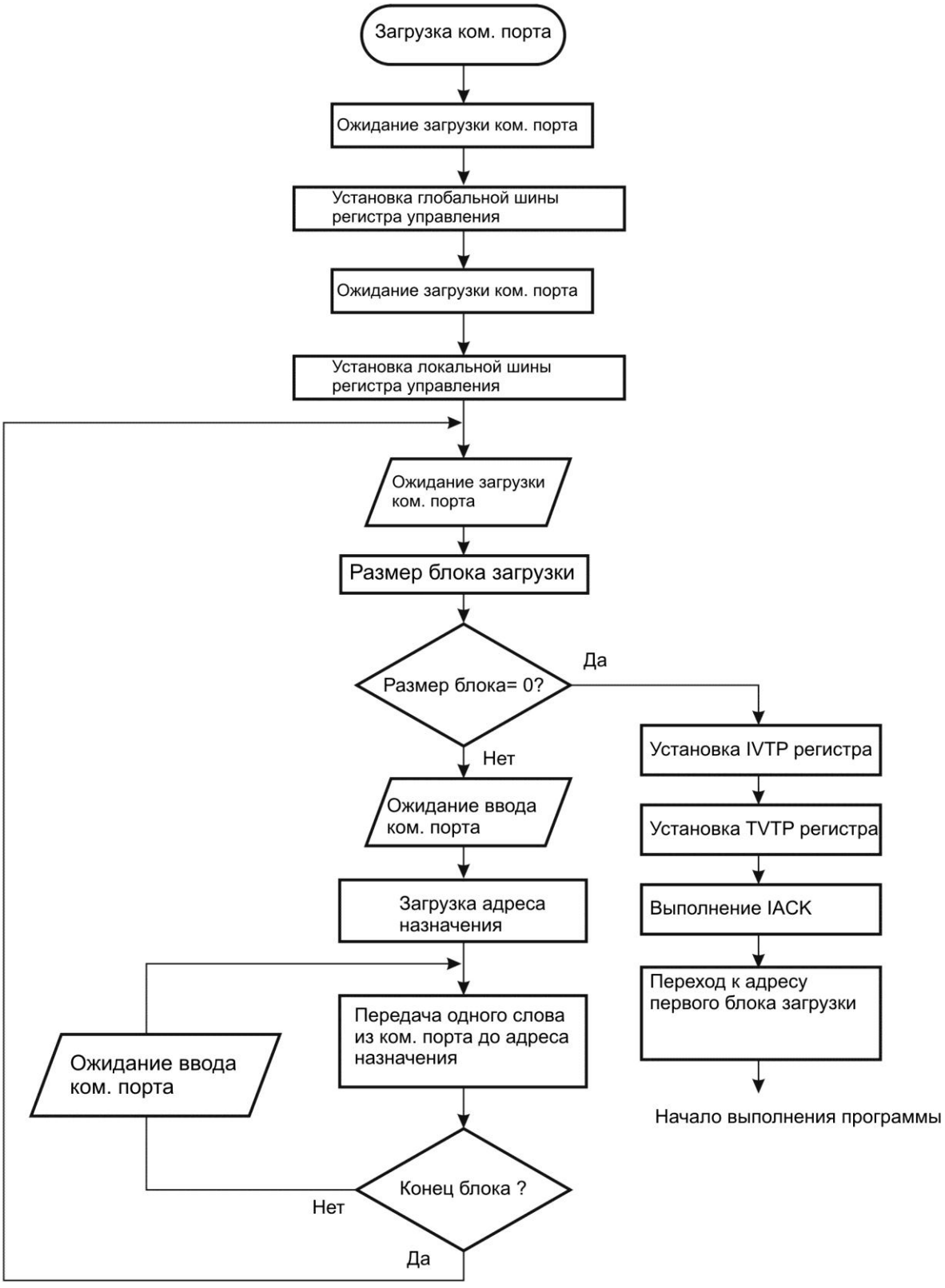


Рисунок 10.2 – Порядок загрузки внешней памяти и исполнения программы



Примечание – Принятое условное сокращение:

- «ком.» – коммуникационный (порт).

Рисунок 10.3 – Порядок загрузки через коммуникационные порты и переход на выполнение загруженной программы

Поток данных с исходной программой должен быть в формате, приведенном в таблице 10.2. Значение данных программы от 4 слова до слова n различно для разных исходных кодов программы. Первые три слова и последние три слова не меняются. Они затрагивают блоки из исходной программы. Восемь наименьших значащих битов (младшие биты) первого слова определяют разрядность внешней памяти. Если выбирается побайтовая загрузка или загрузка полусловами, то последовательность загрузки проходит от младших битов к старшим.

Таблица 10.2 – Структура потока данных исходной программы

Номер слова	Описание обработки слова данных
1	Разрядность внешней памяти, из которой будет производиться загрузка (8, 16 или 32 разряда)
2	Значение установки регистра управления глобальной памяти
3	Значение установки регистра управления локальной памяти
4	Размер блока 32-разрядных слов, который будет загружен в первом блоке программы, после загрузки заданного размера блока должно следовать слово, забитое нулями (это сделано для того, чтобы не произошло сбоя при загрузке следующего блока)
5	Адрес, с которого будет производиться загрузка программы
6	Первое слово загружаемой программы
n	Последнее слово загружаемой программы (программа расположена от 4 слов последовательности до n слов последовательности)
n+1	Слово, состоящее из нулей. Примечание – Если были отправлены несколько блоков исходной программы, то слово n будет последним словом последнего блока исходной программы. Каждый блок исходной программы имеет формат, показанный в четырех словах через n. Это слово, состоящее из всех нулей, следует за последней группой элементов исходной программы.
n+2	Значение регистра IVTP
n+3	Значение регистра TVTP
n+4	Ячейка памяти для инструкции IACK
Примечание – Затененные строки относятся к блоку исходной программы.	

Каждая исходная программа при передаче нескольких блоков программы может быть загружена в различные адресные пространства. Каждый блок программы определяет размер своей программы и адрес назначения. Необходимо закончить весь блок программы нулевым словом 0000 0000h.

В последних словах указываются адреса для установки размещения таблиц системных и программных прерываний. В последнем слове указывается адрес размещения инструкции IACK, которая указывает на завершение работы загрузчика.

10.4 Пример загрузки программы из внешней памяти

Если во время системного сброса входной сигнал ROMEN находится в высоком состоянии и входные сигналы (RESETLOC0, RESETLOC1) = 00_2, то загрузчик будет загружать программы, которые находятся во внешней памяти (8-, 16- или 32-разрядной памяти, находящейся по адресу согласно установки входов IIOF#). Поскольку начальные адреса зарезервированы для работы загрузчика, то они не должны использоваться для вектора системного сброса микросхемы.

8 младших битов первого слова потока данных определяют разрядность внешней памяти (8, 16 или 32 бита), из которой будет загружена программа:

- для 8-разрядной памяти значение равно 08h;
- для 16- разрядной памяти значение равно 0010h;
- для 32- разрядной памяти значение равно 0000 0020h.

Если используется 8-или 16-разрядная память, то последовательность загрузки происходит от младших битов к старшим. При загрузке 16-разрядных слов сначала происходит загрузка младшего слова, затем старшего и далее эти 16-разрядные слова соединяются, образуя полное 32-разрядное слово. Аналогично происходит и с побайтовой загрузкой. Загрузка 32-разрядных слов происходит напрямую без всякой обработки.

При использовании 16-разрядной внешней памяти она должна быть подключена к выводам (L)D15-(L)D0, для 8-разрядной внешней памяти к выводам (L)7-(L)0. Все, не задействованные выводы шин данных, следует подключить к питанию через отдельные резисторы номиналом 22 кОм.

Таблицы 10.3, 10.4 и 10.5 показывают примеры использования 8-, 16- и 32-разрядной внешней памяти.

В данных примерах используются следующие установки:

Если после системного сброса выводы IIOF3#-IIOF0# имеют состояние 110_2, то внешняя память находится по адресу 0030 0000h и имеет разрядность 8, 16 и 32 бита. Память глобальной шины имеет одно программное состояние ожидания внешнего сигнала RDY (SWW=11), размер страницы равен 64К слов для обоих стробов STRB0# и STRB1#, а размер области памяти в 1 Гбайт на каждый строб. Память локальной шины имеет два программных состояния ожидания (SWW=01), размер страницы равен 32К слов для обоих стробов и размер области памяти в 1 Гбайт на каждый строб. Первый блок программы содержит 294 слова и размещается по адресу 002F F840h. Второй блок программы содержит 64 слова и размещается по адресу 002F F800h. Указатели страниц регистров IVTP и TVTP находятся вначале СОЗУ ИС. Команда IACK находится по адресу 0030 0000h.

Таблица 10.3 – Пример загрузки ЦОС с 8-разрядной памятью

Номер слова	Адрес	Значение	Комментарии
1	0030 0000h	08h	Разрядность внешней памяти равна 8 битам
	0030 0001h	00h	
	0030 0002h	00h	
	0030 0003h	00h	
2	0030 0004h	F0h	Значение регистра управления глобальной памятью равно 1D7B C9F0h
	0030 0005h	C9h	
	0030 0006h	7Bh	
	0030 0007h	1Dh	
3	0030 0008h	50h	Значение регистра управления локальной памятью равно 1D73 9250h
	0030 0009h	92h	
	0030 000Ah	73h	
	0030 000Bh	1Dh	
4	0030 000Ch	26h	Размер первого блока программы равен 126h
	0030 000Dh	01h	
	0030 000Eh	00h	
	0030 000Fh	00h	
5	0030 0010h	40h	Стартовый адрес первого блока программы расположен по адресу 002F F840h
	0030 0011h	F8h	
	0030 0012h	2Fh	
	0030 0013h	00h	

Окончание таблицы 10.3

Номер слова	Адрес	Значение данных	Комментарии
с 6 по 299	0030 0014h	…	Инструкции первого блока программы
	…		
	0030 04ABh		
300	0030 04ACh	40h	Размер второго блока программы равен 40h
	0030 04ADh	00h	
	0030 04AEh	00h	
	0030 04AFh	00h	
301	0030 04B0h	00h	Стартовый адрес второго блока программы расположен по адресу 002F F800h
	0030 04B1h	F8h	
	0030 04B2h	2Fh	
	0030 04B3h	00h	
с 302 по 365	0030 04B4h	…	Инструкции второго блока программы
	…		
	0030 05B3h		
366	0030 05B4h	00h	Нулевое слово (0000 0000h), указывающее на конец загрузки блоков программы
	0030 05B5h	00h	
	0030 05B6h	00h	
	0030 05B7h	00h	
367	0030 05B8h	00h	Значение регистра IVTP равно 002F F800h
	0030 05B9h	F8h	
	0030 05BAh	2Fh	
	0030 05BBh	00h	
368	0030 05BCh	00h	Значение регистра TVTP равно 002F F800h
	0030 5BDh	F8h	
	0030 05BEh	2Fh	
	0030 05BFh	00h	
369	0030 05C0h	00h	Инструкция IACK находится по адресу 0030 0000h (и это будет последним словом в загрузке), далее процессор переходит к выполнению загруженной программы

Таблица 10.4 – Пример загрузки ЦОС с 16-разрядной памятью

Номер слова	Адрес	Значение	Комментарии
1	0030 0000h	0010h	Разрядность внешней памяти равна 16 битам
	0030 0001h	0000h	
2	0030 0002h	C9F0h	Значение регистра управления глобальной памятью равно 1D7B C9F0h
	0030 0003h	1D7Bh	
3	0030 0004h	9250h	Значение регистра управления локальной памятью равно 1D73 9250h
	0030 0005h	1D73h	
4	0030 0006h	0126h	Размер первого блока программы равен 126h
	0030 0007h	0000h	
5	0030 0008h	F840h	Стартовый адрес первого блока программы расположен по адресу 002F F840h
	0030 0009h	002Fh	
с 6 по 299	0030 000Ah	. . .	Инструкции первого блока программы
	…		
	0030 0255h		

Окончание таблицы 10.4

Номер слова	Адрес	Значение данных	Комментарии
300	0030 0256h	0040h	Размер второго блока программы равен 40h
	0030 0257h	0000h	
301	00300258h	F800h	Стартовый адрес второго блока программы расположен по адресу 002F F800h
	0030 0259h	002Fh	
с 302 по 365	0030 025Ah	. . .	Инструкции второго блока программы
	…		
	0030 02D9h		
366	0030 02DAh	0000h	Нулевое слово (0000 0000h), указывающее на конец загрузки блоков программы
	0030 02DBh	0000h	
367	0030 02DCh	F800h	Значение регистра IVTP равно 002F F800h
	0030 02DDh	002Fh	
368	003002DEh	F800h	Значение регистра TVTP равно 002F F800h
	0030 02DFh	002Fh	
369	0030 02E0h	0000h	Инструкция IACK находится по адресу 0030 0000h (и это будет последним словом в нашей загрузке), далее процессор переходит к выполнению загруженной программы
	0030 02E1h	0030h	

Таблица 10.5 – Пример загрузки ЦОС с 32-разрядной памятью

Номер слова	Адрес	Значение данных	Комментарии
1	0030 0000h	0000 0020h	Разрядность внешней памяти равна 16 битам
2	0030 0001h	1D7B C9F0h	Значение регистра управления глобальной памятью равно 1D7B C9F0h
3	0030 0002h	1D73 9250h	Значение регистра управления локальной памятью равно 1D73 9250h
4	0030 0003h	0000 0126h	Размер первого блока программы равен 126h
5	0030 0004h	002F F840h	Стартовый адрес первого блока программы расположен по адресу 002F F840h
6	0030 0005h		
по	…	. . .	Инструкции первого блока программы
299	0030 012Ah		
300	0030 012Bh	0000 0040h	Размер второго блока программы равен 40h
301	0030 012Ch	002F F800h	Стартовый адрес второго блока программы расположен по адресу 002F F800h
302	0030 012Dh		
по	…	. . .	Инструкции второго блока программы
365	0030 016Ch		
366	0030 016Dh	0000 0000h	Нулевое слово (0000 0000h), указывающее на конец загрузки блоков программы
367	0030016Eh	002F F800h	Значение регистра IVTP равно 002F F800h
368	0030 016Fh	002F F800h	Значение регистра TVTP равно 002F F800h
369	0030 0170h	0030 0000h	Инструкция IACK находится по адресу 0030 0000h (и это будет последним словом в нашей загрузке), далее процессор переходит к выполнению загруженной программы

10.5 Пример загрузки программы через коммуникационный порт

Если во время системного сброса входной сигнал ROMEN находится в высоком состоянии и входные сигналы IIOF3#-IIOF0# находятся в высоком состоянии, то загрузчик переходит в режим загрузки через коммуникационные порты. Процессор начинает опрос коммуникационных портов, начиная с нулевого и заканчивая пятым, как только на одном из портов будут обнаружены входные данные, то загрузчик начинает выполнять загрузку. Загрузка через коммуникационный порт проходит аналогично загрузке из внешней памяти, за исключением этапа выбора разрядности внешней памяти, так как к коммуникационному порту возможно подключение только 8-разрядной памяти. В примере 10.1 приведена программа загрузки через коммуникационные порты в мультипроцессорной системе, в которой имеются один ведущий и, по крайней мере, один ведомый процессор.

Пример 10.1 – Программа загрузки в мультипроцессорной системе через коммуникационные порты

```
*_____________________________________________________
* Ведущий процессор (МАСТЕР) – Таблица загрузки
*_____________________________________________________
.text
.word   32                      ; разрядность памяти
.word   3003c000h               ; управляющий регистр глобальной памяти
                                ; МАСТЕРА
                                ; (зависит от системы)
.word   3d79c210h                   ; управляющий регистр локальной памяти
                                ; МАСТЕРА;
                                ; ( зависит от системы)
*_____________________________________________________
* Программный блок МАСТЕРА
*_____________________________________________________
.word   10                          ; размер блока
.word   2ff800h                     ; адрес загрузки блока
* Код процессора МАСТЕРА: этот код посылает таблицу загрузки
*в подчиненный процессор
ldi 8, rc                           ; цикл 9 раз: размер таблицы загрузки ведомого
                                    ; процессора
rptbd endb1
ldp src                             ; src во внешней памяти
ldi @src, ar0
ldi @dst, ar1
ldi *ar0++(1), r0                   ; блок начала
endb1: sti r0,*ar1
bu $                                ; процессор МАСТЕР циклится
src .word BOOT_TABLE2               ; адрес таблицы загрузки подчиненного
                                    ; процессора
dst .word 100042h                   ; адрес выходного FIFO (OFIFO),
                                    ; подсоединенного к подчиненному процессору
*_____________________________________________________
* Конец всех блоков
*_____________________________________________________
.word 0                             ; конец последовательности загрузки МАСТЕРА
.word   2ffd00h                     ; значение IVTP МАСТЕРА
```

Окончание примера 10.1

```
.word   2ffd00h                 ; значение TVTP МАСТЕРА
.word   40000000h               ; адрес инструкции IACK МАСТЕРА
*_____________________________________________________
* Конец таблицы загрузки МАСТЕРА: размер = 9 слов
*_____________________________________________________
*_____________________________________________________
* Таблица загрузки подчиненного процессора
*_____________________________________________________
BOOT_TABLE2:                    ; Таблица загрузки подчиненного процессора
.word   3003c000h               ; регистр управления глобальной памятью
                                ; подчиненного процессора (зависит от системы)
.word   3d79c210h               ; регистр управления локальной памятью
                                ; подчиненного процессора (зависит от системы)
.word   1                       ; размер блока
.word   2ff800h                 ; dst адрес загрузки
bu  $                           ; цикл подчиненного процессора
.word   0                       ; конец последовательности загрузчика
                                ; подчиненного процессора
.word   2ffd00h                 ; значение IVTP подчиненного процессора
.word   2ffd00h                 ; значение TVTP подчиненного процессора
.word   40000000h               ; адрес инструкции IACK подчиненного процессора
*_____________________________________________________
* Конец кода EPROM
*_____________________________________________________
```

10.6 Изменение состояния выводов IIOF3#-IIOF0# после завершения работы загрузчика

Так как во время работы загрузчика необходимо, чтобы значение выводов IIOFn# было постоянно задано в нужной комбинации для обеспечения требуемого режима загрузки, то для дальнейшего использования этих выводов (после завершения работы загрузчика), необходимо использовать внешние схемотехнические решения. На рисунке 10.4 приведена схема, которая выставляет вывод IIOFn# в низкий уровень и удерживает его до тех пор, пока не придёт сигнал подтверждения прерывания IACK#. Далее вывод IIOFn# доступен для использования внешними источниками прерываний.

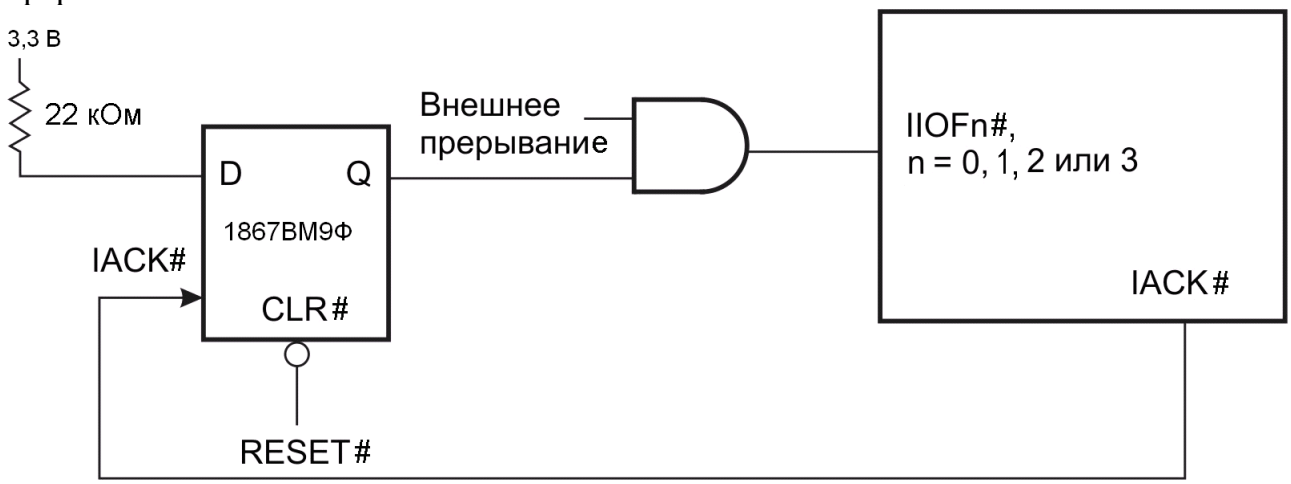


Рисунок 10.4 – Схема управления выводом IIOFn#, где n = 0, 1, 2 или 3

11 Сопроцессор прямого доступа к памяти

Сопроцессор прямого доступа к памяти ПДП – встроенное программируемое устройство, позволяющее осуществлять передачу данных памяти одновременно с выполнением операций ЦПУ при наименьших потерях производительности. В данном разделе описывается сопроцессор ПДП и даются советы по его программированию.

11.1 Сопроцессор ПДП – программируемое периферийное устройство

Сопроцессор ПДП – программируемое периферийное устройство, которое передает блоки данных на вне процессорном уровне для повышения производительности ИС и облегчения обмена пакетами входных/выходных данных. Сопроцессор ПДП осуществляет передачу данных в память и из памяти во всем пространстве адресов карты памяти ИС. Например, передача может осуществляться во внутреннюю и внешнюю память и обратно, а также через любой из шести коммуникационных портов.

Сопроцессор ПДП обеспечивает:

- передачу/прием данных по шести каналам из памяти и в память в основном режиме;
- обмен данными по двенадцати каналам между памятью и коммуникационными портами в специальном режиме – «режиме с разделением»;
- автоматическую инициализацию регистров с помощью связанных списков, хранящихся в памяти, которые позволяют продолжать работу сопроцессора ПДП без вмешательства ЦПУ;
- параллельную работу ЦПУ и сопроцессора ПДП с одинаковой скоростью, (поддерживается отдельным внутренним адресом DMA и шинами данных);
- перемещения по матрице данных по рядам и колонкам с использованием регистров исходного и конечного адреса с индексными переменными;
- бит-реверсивную адресацию для реализации алгоритма БПФ (FFT);
- синхронизацию передачи данных с помощью внешних и внутренних прерываний.

11.2 Функциональное описание сопроцессора ПДП

Сопроцессор ПДП поддерживает шесть каналов ПДП, которые выполняют передачу данных из любого места карты памяти ИС 1867ВМ9Ф.

Каждый канал ПДП управляется девятью регистрами, которые расположены в адресном пространстве периферии, см. рисунок 11.1. Основные регистры ПДП описаны в подразделе 11.3.

Сопроцессор ПДП имеет специальные встроенные шины адреса и данных, см. рисунок 2.8. Все обращения осуществляются шестью каналами ПДП, которые арбитрируются в сопроцессоре ПДП. Шесть каналов ПДП передают данные в режиме с последовательным разделением времени, а не одновременно, так как они делят между собой одни и те же шины.

Каналы ПДП могут работать постоянно или могут быть запущены внешними IIOF3#-IIOF0# или внутренними (внутренние таймеры и коммуникационные порты) прерываниями.

Сопроцессор ПДП может передавать данные в бит-реверсивной форме (для приложений БПФ) или в линейной форме. Он также может передавать матричные данные по рядам и столбцам.

Сопроцессор ПДП имеет два базовых операционных режима:

- основной режим: используется для передачи данных из памяти в память. Основной режим описан в подразделе 11.4. Последовательность передачи в основном режиме приведена в 12.2.1.

- режим с разделением. Используется для двухканальных передач между коммуникационным портом и памятью. Режим разделения описан в подразделе 11.5.

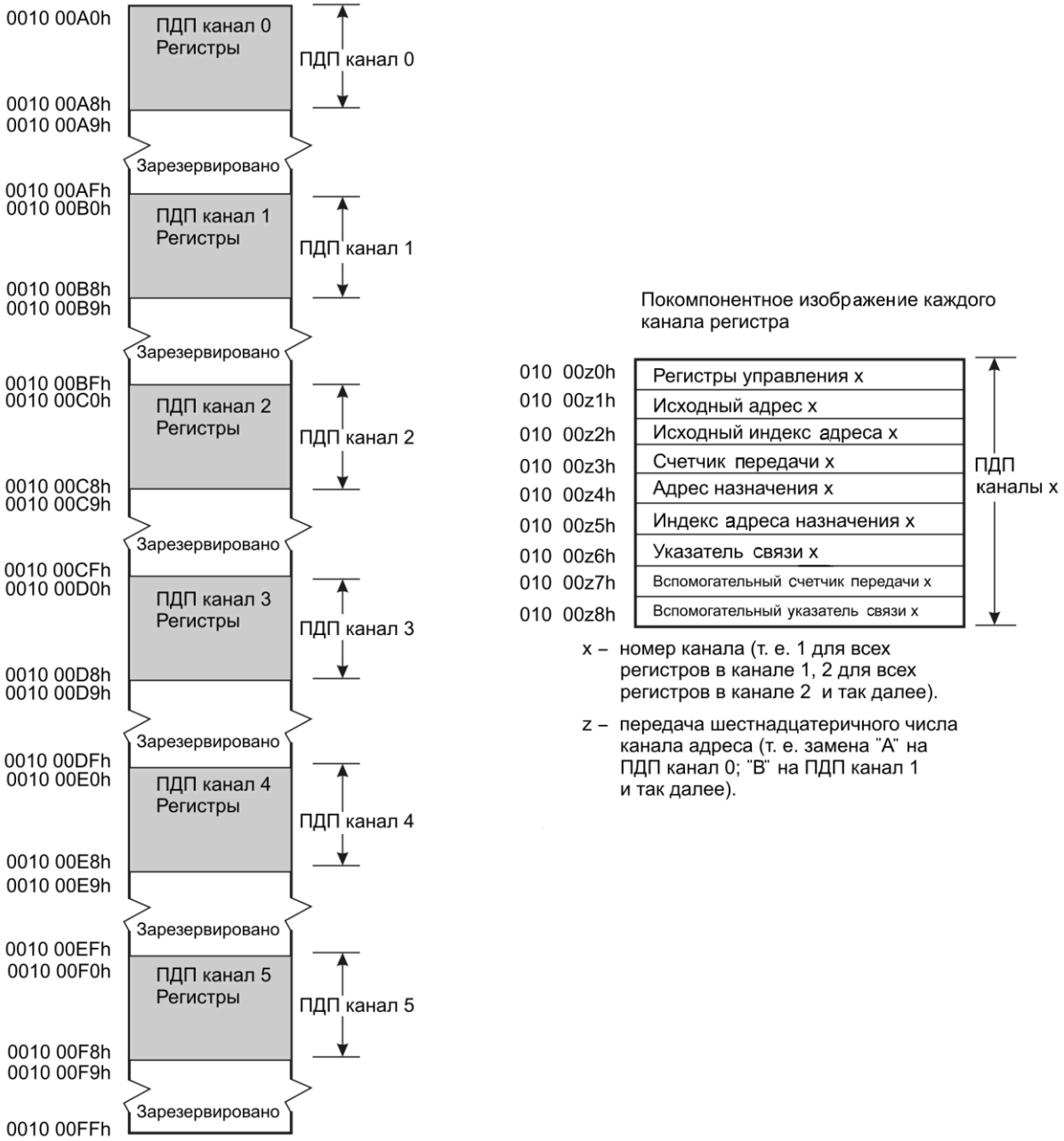


Рисунок 11.1 – Карта памяти сопроцессора ПДП

11.2.1 Базовые операции ПДП

Если блок данных передается из одной области памяти в другую (основной режим), то нужно выполнить инициализацию регистров:

- В регистр начального адреса канала ПДП загружается адрес, откуда будет производиться чтение.

- В регистр конечного адреса того же канала ПДП загружается адрес, куда будет производиться запись.

- В счетчик передачи загружается число передаваемых слов.

- В индексный регистр начального/конечного адреса загружается шаг модификации регистра начального/конечного адреса.

- В регистр управления каналом ПДП загружаются установки соответствующего режима для синхронизации чтения и записи сопроцессора ПДП с прерываниями. Регистр DIE определяет, какое прерывание используется для синхронизации.

Далее необходимо запустить сопроцессор ПДП через поле DMA START в регистре управления каналом ПДП.

Передача слов – канал ПДП считывает слово по адресу, указанному в регистре начального адреса и записывает его во временный регистр канала ПДП.

После чтения каналом ПДП – к значению регистра начального адреса добавляется значение индексного регистра начального адреса.

После завершения операции чтения канал ПДП записывает значение временного регистра по адресу назначения, указанному регистром конечного адреса.

После выборки конечного адреса счетчик передачи декрементируется, и значение индексного регистра конечного адреса добавляется к значению регистра конечного адреса.

Примечание – Оба индексных регистра (начального и конечного адреса) содержат значения со знаком. Допускается переменная величина шага или чтение из памяти и/или запись в память последовательно с единичным индексом. Если индексный регистр равен «0», то сопроцессор ПДП передает данные между определенными фиксированными адресами.

В течение каждого цикла записи счетчик передачи декрементируется. Блок передаваемых данных заканчивается, когда счетчик передачи достигает «0» и завершается последняя операция записи передаваемых данных. Канал ПДП устанавливает флаг прерывания счетчика передачи TCINT в регистре управления каналом ПДП.

После завершения передачи данных сопроцессор ПДП может быть запрограммирован для следующего:

- остановки для перепрограммирования (биты TRANSFER MODE=01_2);
- продолжения передачи данных (биты TRANSFER MODE=00_2);
- генерации прерывания для сигнализации ЦПУ, что передача данных завершена (бит TCC=1_2);
- автоинициализации для запуска передачи следующего блока данных (биты TRANSFER MODE=10_2 или TRANSFER MODE=11_2).

Каждый канал ПДП считывает новое значение регистра ПДП из памяти, загружает его в соответствующий регистровый файл и в соответствии с загруженным значением начинает новую передачу. ЦПУ, в любом случае, должен инициализировать передачи, определяемые значениями битов режима передачи:

Автоинициализация выполняется без вмешательства ЦПУ. Биты TRANSFER MODE=10_2.

Автоинициализация требует вмешательства ЦПУ для запуска ПДП. Биты TRANSFER MODE=11_2.

11.3 Регистры сопроцессора ПДП

Каждый канал ПДП имеет девять регистров:

- Регистр управления: содержит информацию о состоянии и режиме работы канала ПДП.
- Регистр адреса источника содержит адрес памяти, с которого начнут считываться данные.
- Индексный регистр адреса источника: содержит значение шага (знаковое 32-разрядное число), используемое для инкрементирования или декрементирования регистра адреса источника.
- Регистр адреса назначения содержит адрес памяти, куда будут записываться данные.

- Индексный регистр адреса назначения: содержит значение шага (знаковое 32-разрядное число), используемое для инкрементирования или декрементирования регистра адреса назначения.

- Регистр-счетчик передачи: содержит размер блока данных, передаваемых в основном или раздельном режиме (основной канал).

- Вспомогательный регистр-счетчик передачи содержит размер блока данных, передаваемых в режиме разделения (вспомогательный канал).

- Регистр-указатель содержит адрес памяти с данными для автоинициализации регистров канала ПДП. Используется в основном режиме или режиме разделения для основного канала.

- Вспомогательный регистр-указатель содержит адрес памяти с данными для автоинициализации регистров канала ПДП. Используется в режиме разделения для вспомогательного канала.

После сброса регистр управления, счетчик передачи, вспомогательный счетчик передачи устанавливаются в «0», а остальные регистры находятся в неопределенном состоянии.

11.3.1 Регистр управления

Формат регистра управления каналом ПДП показан на рисунке 11.2. Следующий за рисунком текст описывает каждое поле регистра.

При сбросе каждый регистр управления каналом ПДП устанавливается в «0». Это делает канал ПДП ниже по приоритету, чем ЦПУ, устанавливает адреса источника и приемника для линейной (последовательной) адресации, конфигурирует канал ПДП для работы в основном режиме.

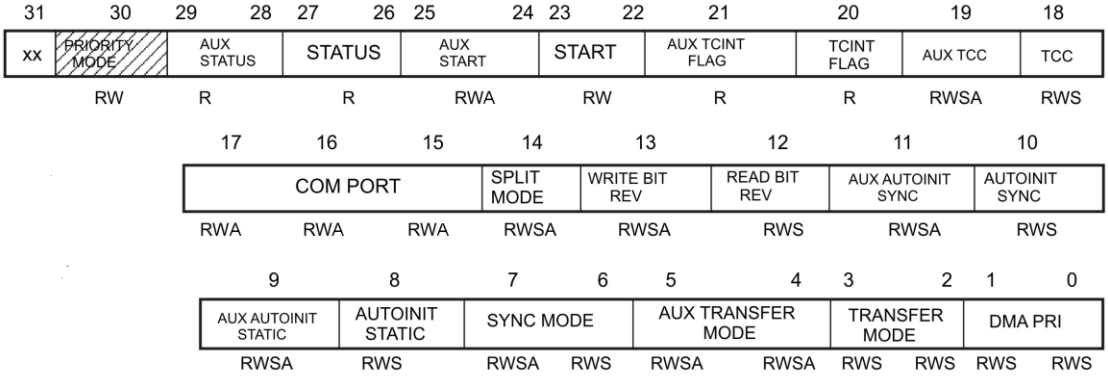


Рисунок 11.2 – Регистр управления каналом ПДП

Функциональное описание разрядов регистра управления каналом ПДП

DMA PRI — Устанавливает приоритет сопроцессора ПДП. Определяет правила арбитража, которые будут использоваться, когда канал ПДП и ЦПУ обращаются к одному и тому же ресурсу. Влияет во всех режимах работы ПДП. Правила предоставления приоритета изложены в таблице 11.1.

TRANSFER MODE — Определяет режим передачи канала ПДП. Влияет на основной режим и на основной канал в режиме разделения. Биты определены в таблице 11.2.

AUX TRANSFER MODE	Определяет режим передачи канала ПДП. Влияет на вспомогательный канал в режиме разделения. Биты определены в таблице 11.2.
SYNC MODE	Определяет режим синхронизации для выполнения передачи данных. Эти биты работают по-разному в основном и раздельном режимах. В таблицах 11.3 и 11.4 приведено описание для основного и раздельного режимов. Примечание – Если канал ПДП управляется прерываниями и для чтения, и для записи, и прерывание для записи приходит перед прерыванием для чтения, то прерывание для записи фиксируется каналом ПДП. После завершения чтения может быть выполнена запись.
AUTOINIT STATIC	Этот бит влияет на основной режим и на основной канал в режиме разделения. Он удерживает вспомогательный указатель постоянным в течение внутренней автоинициализации от коммуникационных портов или других устройств (таких как буферы FIFO). Если бит = 0, то указатель инкрементируется в течение автоинициализации. Если бит = 1, то указатель не инкрементируется (он статичен) в течение автоинициализации.
AUX AUTOINIT STATIC	Этот бит работает как и AUTOINIT STATIC, но только применяется к вспомогательному каналу в режиме разделения.
AUTOINIT SYNC	Бит работает только в соответствии с режимом синхронизации ПДП, определяемом битами 6, 7. Он влияет на прерывание, которое разрешается регистром разрешения прерываний ПДП, смотри рисунок 11.25, используемым для чтений ПДП: если бит = 0, то прерывание игнорируется, автоини-циализированные чтения не синхронизируются с сигналами прерываний; если бит = 1, то прерывание определяется и используется для синхронизации автоинициализированных чтений. Этот бит влияет на основной режим и на основной канал в режиме разделения, см. бит SPLIT MODE. Влияние битов AUTOINIT SYNC и SYNC MODE на автоинициализацию приведено в таблице 11.9.
AUX AUTOINIT SYNC	Бит работает как и AUTOINIT SYNC, но только применительно к вспомогательному каналу в режиме разделения. Влияние битов AUX AUTOINIT SYNC и SYNC MODE на автоинициализацию приведено в таблице 11.9.
READ BIT REV	Бит выбирает тип адресации для изменения адреса источника. Если бит = 0, то адрес источника изменяется посредством 32-разрядной прямой адресации. Если бит = 1, то адрес источника изменяется посредством 24-разрядной бит-реверсивной адресации. Бит влияет на основной режим и на чтения (адреса источника) основного канала в режиме разделения.
WRITE BIT REV	Бит выбирает тип адресации для изменения адреса назначения. Если бит = 0, адрес назначения изменяется посредством 32-разрядной прямой адресации. Если бит = 1, адрес назначения изменяется посредством 24-разрядной бит-реверсной адресации. Этот бит влияет на основной режим и на записи (адреса назначения) вспомогательного канала в режиме разделения.

SPLIT MODE — Данный бит управляет режимами сопроцессора ПДП. Если бит = 0, то ПДП осуществляет передачу из памяти в память. Этот режим называется основным. Если бит = 1, то ПДП переходит в режим разделения, при котором каждый канал ПДП разделен на два канала, разрешая одному каналу выполнять передачи из памяти в коммуникационный порт и наоборот. Режим разделения может изменяться посредством автоинициализации в основном режиме или посредством автоинициализации вспомогательного канала в режиме разделения. Режим разделения описан в подразделе 11.4.

COM PORT — Эти биты определяют, какой коммуникационный порт (от 000_2 до 101_2) будет использоваться для передачи данных ПДП. Если SPLIT MODE = 0, то бит COM PORT не влияет на операции канала ПДП. Если SPLIT MODE = 1, то COM PORT определяет, какой из шести коммуникационных портов будет использоваться с каналом ПДП. COM PORT может быть изменен через автоинициализацию в основном режиме или посредством автоинициализации вспомогательного канала в режиме разделения.

TCC — Бит управляет прерыванием счетчика передачи. Если TCC=1, то импульс прерывания канала ПДП посылается к ЦПУ после перехода счетчика передачи через «0» и записи последних переданных данных. Если бит включен, то соответствующее прерывание ПДП (ПДП INT0, INT1) происходит в соответствии с вектором, показанным на рисунке 7.2. Если TCC=0, то импульс прерывания канала ПДП не посылается к ЦПУ, когда счетчик передачи переходит через «0». Этот бит влияет на основной режим и на основной канал в режиме разделения.

AUX TCC — Бит управляет прерыванием вспомогательного счетчика передачи. Если бит=1, то импульс прерывания канала ПДП посылается к ЦПУ после того, как вспомогательный счетчик передачи перейдет через «0» и будет завершена запись последних переданных данных. Если бит включен, то соответствующее прерывание ПДП (ПДП INT0, INT1) происходит, как показано на рисунке 7.2. Если бит = 0, то импульс прерывания канала ПДП не посылается к ЦПУ, когда вспомогательный счетчик передачи переходит через «0». Этот бит влияет только на вспомогательный канал в режиме разделения.

TCINT FLAG — Флаг прерывания счетчика передачи. Этот флаг устанавливается в «1», если счетчик передачи переходит через «0» и завершается запись последних переданных данных. Когда считывается регистр управления каналом ПДП, то этот флаг сбрасывается, пока не будет установлен посредством ПДП в том же цикле, что и чтение. TCINT FLAG управляется в основном режиме и основным каналом в режиме разделения.

AUX TCINT FLAG — Флаг прерывания вспомогательного счетчика передачи. Этот флаг устанавливается в «1», если вспомогательный счетчик передачи переходит через «0» и завершается запись последних переданных данных. Когда считывается регистр управления каналом ПДП, то этот флаг сбрасывается, пока не будет установлен ПДП в том же цикле, что и чтение. AUX TCINT FLAG управляется вспомогательным каналом в режиме разделения. Так как для канала ПДП

доступно только одно прерывание, то можно определить, какое событие устанавливает прерывание TCINT FLAG или AUX TCINT FLAG.

START — Бит запускает и останавливает канал ПДП разными способами (описаны в таблице 11.5). START влияет на основной режим и на основной канал в режиме разделения. Если эти биты используются для удержания канала между последовательными автоинициализациями, то биты START и AUX START должны удерживать последовательность автоинициализации. Если биты START и AUX START изменяются каналом ПДП (например, подавая код останова 10_2 на счетчик передачи для остановки передачи) и запись выполняется из внешнего источника в регистр управления каналом ПДП, то внутреннее изменение биты START и AUX START каналом ПДП имеет приоритет. Смотри значения битов TRANSFER MODE 01_2 в таблице 11.2 для более подробной информации.

AUX START — Бит запускает и останавливает канал ПДП разными способами, описанными в таблице 11.5. AUX START влияет только на вспомогательный канал в режиме разделения.

STATUS — Отображает состояние канала ПДП как показано в таблице 11.6. STATUS обновляется в основном режиме и основным каналом в режиме разделения. Обновления происходят в каждом цикле. Биты STATUS и AUX STATUS также определяются, если канал ПДП был остановлен или был сброшен после записи в биты START и AUX START.

AUX STATUS — Отображает состояние канала ПДП как показано в таблице 11.6. AUX STATUS обновляется только вспомогательным каналом в режиме разделения. Модификация происходит в каждом цикле.

PRIORITY MODE — Режим приоритета доступа канала ПДП: если бит = 0, то приоритет определяется в соответствии с циклом, описанным в подразделе 11.6. Если бит = 1, то приоритет фиксирован, как описано в подразделе 11.6. Этот бит доступен только для нулевого канала ПДП.

Таблица 11.1 – Биты DMA PRI и правила арбитража ЦПУ/ПДП

Биты DMA PRI 1, 0	Описание
1	2
0 0	Сопроцессор ПДП имеет более низкий приоритет, чем ЦПУ. Если канал ПДП и ЦПУ обращаются к одному ресурсу памяти, то доступ получает ЦПУ. Это значение устанавливается при сбросе.
0 1	Эта установка выбирает циклический арбитраж, который устанавливает приоритеты между ЦПУ и каналом ПДП посредством их альтернативного доступа, но не одинаково равнозначного. Приоритет циклически изменяется между ЦПУ и каналом ПДП, когда они конфликтуют в течение последовательных циклов инструкций. Если с самого начала канал ПДП и ЦПУ запрашивают один и тот же ресурс памяти, то ЦПУ имеет приоритет. Если в следующем цикле инструкций сопроцессор ПДП и ЦПУ снова обращаются к одному и тому же ресурсу памяти, то ПДП имеет приоритет. Альтернативный доступ продолжается до тех пор, пока запросы ЦПУ и ПДП конфликтуют в течение последовательных циклов инструкций. Если в предыдущем цикле инструкций конфликта не было, то ЦПУ имеет приоритет.

Окончание таблицы 11.1

1	2
1 0	Зарезервировано
1 1	Доступ сопроцессора ПДП имеет более высокий приоритет, чем приоритет доступа ЦПУ. Если канал ПДП и ЦПУ обращаются к одному ресурсу памяти, то работает ПДП.
0 0	Передача данных не прерывается счетчиком передачи и не выполняется автоинициализация. TCINT (прерывание от счетчика передачи) и AUX TCINT при этом могут использоваться для вызова прерывания, когда счетчик передачи переходит через «0». Канал ПДП продолжает работу. Адрес продолжает инкрементироваться, пока счетчик передачи не достигнет максимального значения 0FFF FFFFh.
0 1	Передача данных отменяется счетчиком передачи. Автоинициализация не выполняется. Код останова 10_2 помещается в поле START (или AUX START), когда передача данных завершена.

Таблица 11.2 – Описание поля TRANSFER MODE (AUX TRANSFER MODE)

Биты TRANSFER MODE 3, 2 (5, 4)	Описание
1 0	Автоинициализация выполняется, когда счетчик передачи переходит в «0», без ожидания вмешательства ЦПУ.
1 1	Канал ПДП автоинициализируется, когда ЦПУ повторно запускает сопроцессор ПДП, используя регистр ПДП в ЦПУ. Когда счетчик передачи переходит в «0», операция ПДП останавливается до тех пор, пока ЦПУ не запустит сопроцессор ПДП, используя поле START (AUX START) в регистре управления каналом ПДП (биты 22, 23 и 24, 25, см. таблицу 11.5). Код останова 10_2 помещается в поле START (или AUX START) сопроцессором ПДП.

Таблица 11.3 – Описание поля SYNC MODE в основном режиме

Биты SYNC MODE 7, 6	Описание
0 0	Синхронизация отсутствует. Прерывания игнорируются, смотри рисунок 11.27
0 1	Синхронизация источником данных. Чтение не выполняется до тех пор, пока не произойдет разрешение прерывания (смотри рисунок 11.28а). Прерывание определяется полем DMAx READ регистра разрешения прерываний ПДП (DIE), смотри 11.10.1.
1 0	Синхронизация приемником данных. Запись не выполняется до тех пор, пока произойдет разрешение прерывания (смотри рисунок 11.29а). Прерывание определяется полем DMAx WRITE регистра разрешения прерываний ПДП (DIE), смотри 11.10.1.
1 1	Синхронизация источником и приемником данных. Чтение выполняется, когда происходит разрешение прерывания (определяется полем DMAx READ). Затем выполняется запись, когда происходит разрешение прерывания (определяется полем DMAx WRITE), как показано на рисунке 11.30.
0 0	Синхронизация отсутствует. Прерывания игнорируются, смотри рисунок 11.27

Окончание таблицы 11.3

1	2
0 1	Синхронизация приемником данных. Запись из основного канала в выходной буфер FIFO коммуникационного порта не выполняется до тех пор, пока не произойдет разрешение прерывания (смотри рисунок 11.28б). Прерывание определяется полем DMAx PRIMARY WRITE регистра разрешения прерываний ПДП (DIE), смотри 11.10.1.
1 0	Синхронизация источником данных. Чтение во вспомогательный канал из входного буфера FIFO коммуникационного порта не выполняется до тех пор, пока не произойдет разрешение прерывания, смотри рисунок 11.28б). Прерывание определяется полем DMAx AUXILIARY READ регистра разрешения прерываний ПДП (DIE), смотри 11.10.1.

Таблица 11.4 – Описание поля SYNC MODE в режиме разделения

Биты SYNC MODE 7, 6	Описание
1 1	Синхронизация источником и приемником данных. Чтение из входного буфера FIFO коммуникационного порта выполняется, когда происходит разрешение прерывания (определяется полем DMAx AUXILIARY READ). Запись в выходной буфер FIFO коммуникационного порта выполняется, когда происходит разрешение прерывания (определяется полем DMAx PRIMARY WRITE). Эти поля являются частью регистра разрешение прерываний ПДП (DIE), смотри 11.10.1.

Таблица 11.5 – Описание поля START (AUX START)

Биты START (AUX START) 23, 22 (25, 24)	Описание
0 0	Сброс канала ПДП. Циклы чтения или записи канала ПДП завершаются (не прерываются); любые прочитанные данные игнорируется. Любые ожидающие (не запущенные) чтения и записи отменяются. Вспомогательный (AUX START=00_2) и основной (START=00_2) счетчики передачи сбрасываются в «0». Канал ПДП сбрасывается, так что когда он стартует после сброса, то начинается новая транзакция, поэтому и выполняется чтение. В этом режиме производится непосредственная остановка без загрузки каких-либо других регистров.
0 1	Остановка ПДП на допустимой границе при чтении и записи. Канал ПДП останавливается на первой допустимой границе при чтении или записи. Если чтение или запись уже начались, то чтение или запись завершается перед остановом. Если чтение или запись еще не начались, то чтение или запись не начинается. В этом режиме производится непосредственная остановка без загрузки каких-либо других регистров.
1 0	Остановка ПДП на границе при передаче. Останавливает канал ПДП на первой возможной границе при передаче данных. Если передача данных ПДП уже началась, то полная передача данных продолжается до своего завершения, включая циклы чтения и записи, до остановки. Если же передача данных не началась, то чтение или запись не запускаются. В этом режиме производится непосредственная остановка без загрузки каких-либо других регистров. Это значение относится также после завершения передачи данных ПДП.
1 1	Запуск ПДП. Запись 11_2 в это поле запускает выполнение операций ПДП, используя значения регистров канала ПДП разных каналов, см. рисунок 11.1. Если ПДП в режиме автоинициализации, то все регистры ПДП загружаются значениями перед началом операции. Сопроцессор ПДП запускается после сброса, если до этого был сброс (биты START или AUX START равны 00_2) или повторный запуск из предыдущего состояния, если был остановлен (биты START или AUX START равны 01_2 или 10_2).

Таблица 11.6 – Описание поля STATUS (AUX STATUS)

Биты STATUS (AUX STATUS) 27, 26 (29, 28)	Описание
0 0	Канал ПДП удерживается на границе при передаче данных (запись завершена, чтение не начато). Это значение устанавливается при сбросе после останова на границе при передаче данных или после передачи блока данных.
0 1	Канал ПДП удерживается на границе передачи данных (чтение завершено, запись не начата). Это возможно, только если поле START (или AUX START) = 01_2.
1 0	Зарезервировано.
1 1	Канал ПДП не удерживается и не сброшен.

11.3.2 Адресные и индексные регистры

Как показано на рисунке 11.3, регистры адреса источника и адреса назначения канала ПДП имеют соответствующие им индексные регистры. После каждого цикла чтения канала ПДП (из адреса источника) или записи (по адресу назначения) соответствующий генератор адреса (источника или приемника) добавляет значение индексного регистра к значению в соответствующем регистре адреса и помещает результат в этот же регистр. Таким образом, регистр адреса работает в качестве аккумулятора, так как возвращает значение суммы своего предыдущего значения и значения индексного регистра, как показано ниже:

Регистр адреса + Индексный регистр → Регистр адреса.

Значения этих регистров не определены при сбросе.

В зависимости от разрядов 12 и 13 (READ BIT REV и WRITE BIT REV) регистра управления ПДП, сложение может быть:

- линейное (обычное сложение): READ BIT REV=0 или WRITE BIT REV=0;
- бит-реверсивное (с обратным переносом):

READ BIT REV=1 или WRITE BIT REV=1.

Значения индексных регистров (адреса источника и приемника) являются знаковыми.

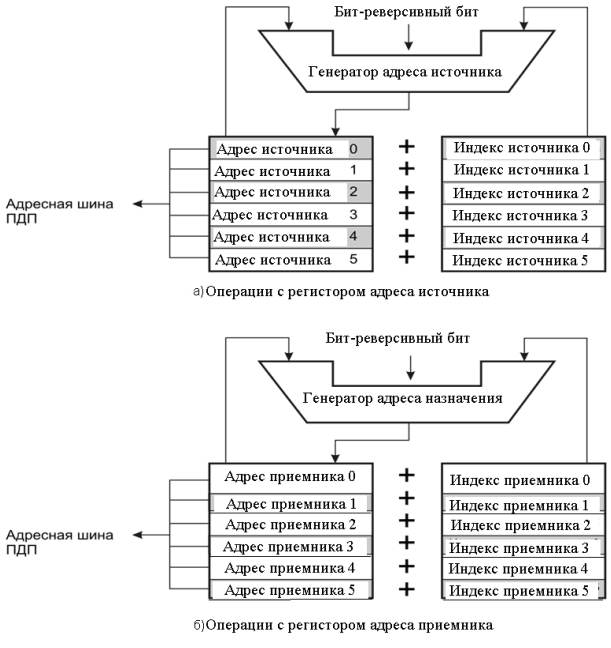


Рисунок 11.3 – Генерация адресов сопроцессора ПДП: операции с регистром адреса источника; б) операции с регистром адреса приемника

11.3.3 Регистры счетчика передачи и вспомогательного счетчика передачи

Регистры счетчика передачи и вспомогательного счетчика передачи содержат значение числа передаваемых слов.

На рисунке 11.4 показано шесть счетчиков передачи и шесть вспомогательных счетчиков передачи. Канал ПДП в режиме разделения использует вспомогательный счетчик передачи для вспомогательного канала и основной счетчик передачи для основного канала. При сбросе значения этих регистров сбрасываются в «0». Счетчики декрементируются после завершения выборки адреса для записи порции передаваемых данных. Флаги TCINT FLAG и AUX TCINT FLAG (биты 20 и 21 регистра управления каналом ПДП, см. рисунок 11.2), не устанавливаются до тех пор, пока счетчик не декрементируется и не будет завершена передача последней порции данных. Соответственно, контроллер прерываний ЦПУ не будет «видеть» прерывание до тех пор, пока счетчик не декрементируется и не будет завершена передача последней порции данных.

Блок декрементирования проверяет, перешел или нет счетчик передачи в «0» после того, как был произведен декремент. В результате, если счетчик имеет значение, равное «1», то канал ПДП будет остановлен после выполнения одной передачи. Таким образом, устанавливая счетчик передачи в «1», канал ПДП передает минимально возможное число слов (один раз). Результат вычислений счетчиком интерпретируется как беззнаковое целое число. Передача может быть остановлена, если после декремента счетчик равен «0». Если канал ПДП не останавливается после того, как счетчик стал равен «0», то счетчик продолжает декрементировать значением меньше «0». Таким образом, устанавливая счетчик передачи в «0», канал ПДП передает максимально возможное число слов (10000 0000h раз).

* x – номер канала ПДП принимает значения от 0 до 5.

Рисунок 11.4 – Счетчик передачи

11.3.4 Регистр-указатель и вспомогательный регистр-указатель

Указатели определяют адрес, из которого будут загружаться новые значения регистров канала ПДП, когда выполняется автоинициализация. Когда канал очищен (счетчик передачи = 0), он может (если он соответственно сконфигурирован) использовать регистр указателя для повторной самозагрузки. На рисунке 11.5 приведены регистры-указатели адреса сопроцессора ПДП. Значения этих регистров при сбросе не определены.

Например, при автоинициализации для загрузки регистров ПДП канала 0 необходимо следующее:

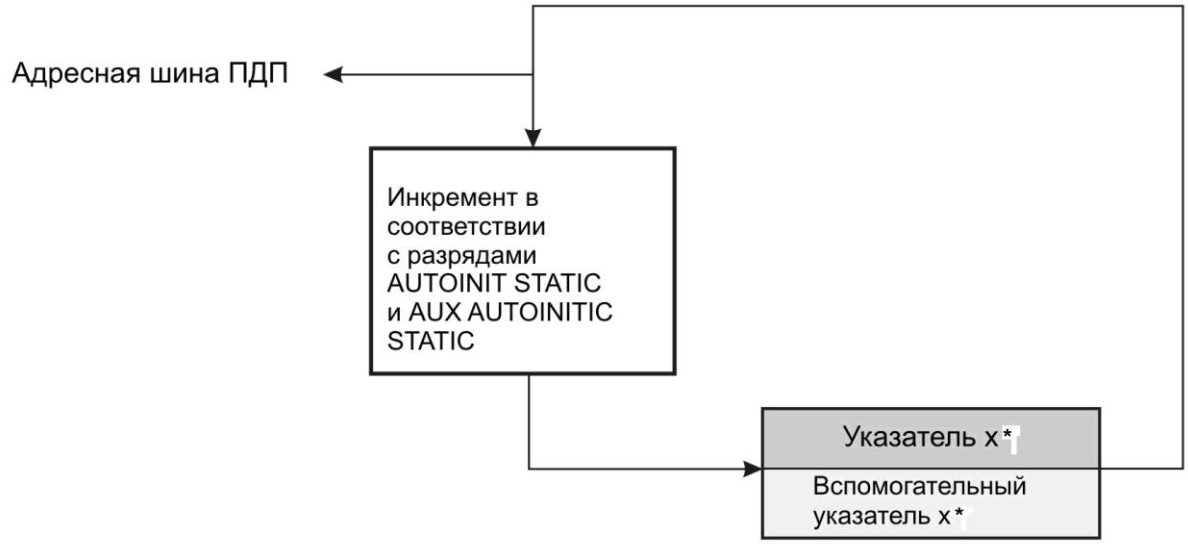
- Получить значение указателя для следующей операции ПДП. Оно является адресом памяти, по которому хранится значение первого регистра канала 0 ПДП (регистр управления каналом показан на рисунке 11.1).

- Взять данные из этого адреса и записать по адресу 0010 00A0h (первое слово регистра канала 0 ПДП, см. рисунок 11.1).

- Инкрементировать регистр-указатель. Если бит AUTOINIT STATIC = 1, то этот этап пропускается.

- Взять следующее слово и записать его по адресу 0010 00A1h.

- Повторять до тех пор, пока весь блок регистров канала 0 ПДП не будет загружен (7 регистров в основном режиме, 5 регистров в режиме разделения).



_____

* x – номер канала ПДП принимает значения от 0 до 5.

Рисунок 11.5 – Регистры-указатели

11.4 Основной режим работы сопроцессора ПДП

Основной режим – это режим сопроцессора ПДП, устанавливаемый по умолчанию. Он используется для передачи данных из памяти в память. Для выбора основного режима необходимо сбросить бит SPLIT MODE (разряд 14 регистра управления каналом ПДП, см. рисунок 11.2), таким образом, просто пишется в него «0» (0 – значение, устанавливаемое при сбросе).

Последовательность передачи блока данных в основном режиме описана в 11.2.1. Арбитраж канала ПДП в основном режиме описан в подразделе 11.6.

Синхронизация ПДП с прерываниями описана в подразделе 11.10. Автоинициализация в основном режиме описана в 11.9.1.

Основной режим ПДП передачи данных включает 2 этапа, как показано на рисунке 11.6:

- канал ПДП считывает данные из адреса, указанного регистром адреса источника, а затем сохраняет их во временном регистре;

- канал ПДП считывает значение временного регистра и записывает его по адресу, указанному регистром адреса назначения.

Можно использовать основной режим для обмена данными с коммуникационным портом, в основном, при однонаправленных передачах. Для двунаправленной передачи данных следует использовать режим разделения.

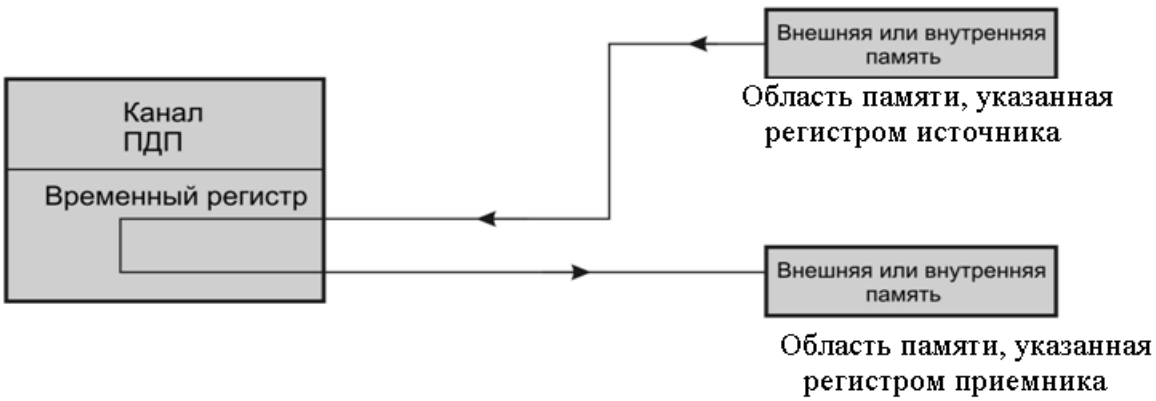


Рисунок 11.6 – Типичная конфигурация канала ПДП в основном режиме

11.5 Режим работы сопроцессора ПДП с разделением

Режим работы ПДП с разделением, см. рисунок 11.7, позволяет использовать один канал ПДП и для чтения, и для записи коммуникационного порта. Режим с разделением преобразует один канал ПДП в два канала ПДП:

- основной канал предназначен для чтения данных из области памяти (внешней/внутренней) и записи их в выходной буфер FIFO коммуникационного порта;

- вспомогательный канал предназначен для чтения данных из входного буфера FIFO коммуникационного порта и записи их в область памяти.

Для выбора режима с разделением необходимо установить бит SPLIT MODE (разряд 14 регистра управления каналом ПДП, рисунок 11.2) в «1».

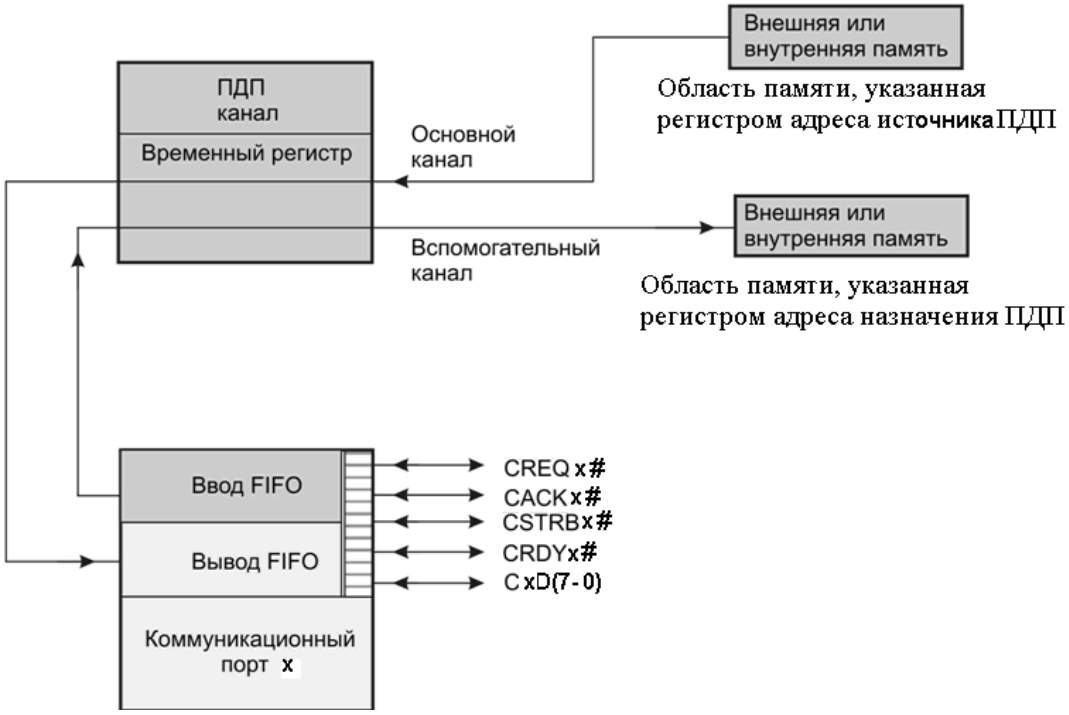
Все шесть каналов ПДП поддерживают режим разделения для всех коммуникационных портов. Поле COM PORT (разряды 15-17, как показано на рисунке 11.2) регистра управления каналом ПДП определяет, какой коммуникационный порт будет использован (порт 0 – порт 5). Канал ПДП в режиме разделения может использоваться с любым коммуникационным портом; однако, синхронизация чтения/записи ограничена условием, что коммуникационный порт должен иметь тот же номер, что и канал ПДП; другими словами, ПДПi синхронизируется только с сигналами, приходящими с коммуникационного порта i, см. подраздел 11.10. На рисунке 11.7 приведена типичная операция в режиме с разделением с одним коммуникационным портом.

Передача данных в режиме с разделением такая же, как и в основном режиме, за исключением следующего:

- Основной канал считывает данные из адреса, указанного регистром адреса источника и записывает их во временный регистр в пределах сопроцессора ПДП. Затем записывает значение временного регистра в выходной буфер FIFO коммуникационного порта, определенного полем COM PORT. Регистры, которые управляют основным каналом ПДП – это регистр управления каналом ПДП, регистр адреса источника, индексный регистр адреса источника (его значение добавляется к значению регистра адреса источника), регистр-счетчик передачи и регистр-указатель.

- Вспомогательный канал считывает слово из входного буфера FIFO коммуникационного порта, определенного полем COM PORT и записывает его во временный регистр в пределах сопроцессора ПДП. Затем записывает значение временного регистра по адресу, указанному регистром адреса назначения. Регистры, управляющие вспомогательным каналом, – это регистр управления каналом ПДП, регистр адреса приемника, индексный регистр адреса приемника (его значение добавляется к значению регистра адреса назначения), вспомогательный регистр-счетчик передачи и вспомогательный регистр-указатель.

Арбитраж канала ПДП в режиме разделения описан в 11.6.3. Синхронизация ПДП с прерываниями описана в подразделе 11.10. Автоинициализация в режиме разделения описана в 11.9.2.



Примечание – «x» принимает значения от 0 до 5.

Рисунок 11.7 – Типичная конфигурация режима разделения ПДП

Примечание – Существует только один временный регистр в каждом канале ПДП. Таким образом, сначала должна завершиться операция основного канала, прежде чем начнет работу вспомогательный канал и наоборот.

Основной и вспомогательный канал разделяют между собой некоторые регистры управления каналом ПДП, а некоторые используют по отдельности:

Поля PRIORITY MODE, COM PORT, SPLIT MODE, DMA PRI используются как основным, так и вспомогательным каналом.

Биты AUX STATUS, AUX START, AUX TCINT флаг, AUX TCC, WRITE BIT REV, SYNC MODE (разряд 7), AUX TRANSFER MODE используются только вспомогательным каналом.

Биты STATUS, START, TCINT флаг, READ BIT REV, SYNC MODE (разряд 6), TRANSFER MODE используются только основным каналом.

11.6 Схемы внутренних приоритетов ПДП

Так как все обращения шести каналов ПДП относятся к общим внутренним шинам данных и адресов ПДП, то для арбитража шин требуется схема приоритетов. В пределах сопроцессора ПДП используется две схемы приоритетов для определения, какой канал будет обслуживаться следующим:

- схема с фиксированными приоритетами, где канал 0 имеет высший приоритет, а канал 5 имеет низший приоритет;

- схема циклических приоритетов, где только что обслуженный канал помещается в конец списка приоритетов (устанавливается при сбросе по умолчанию).

11.6.1 Схема с фиксированными приоритетами

Эта схема обеспечивает фиксированные (неизменяемые) приоритеты для каждого канала:

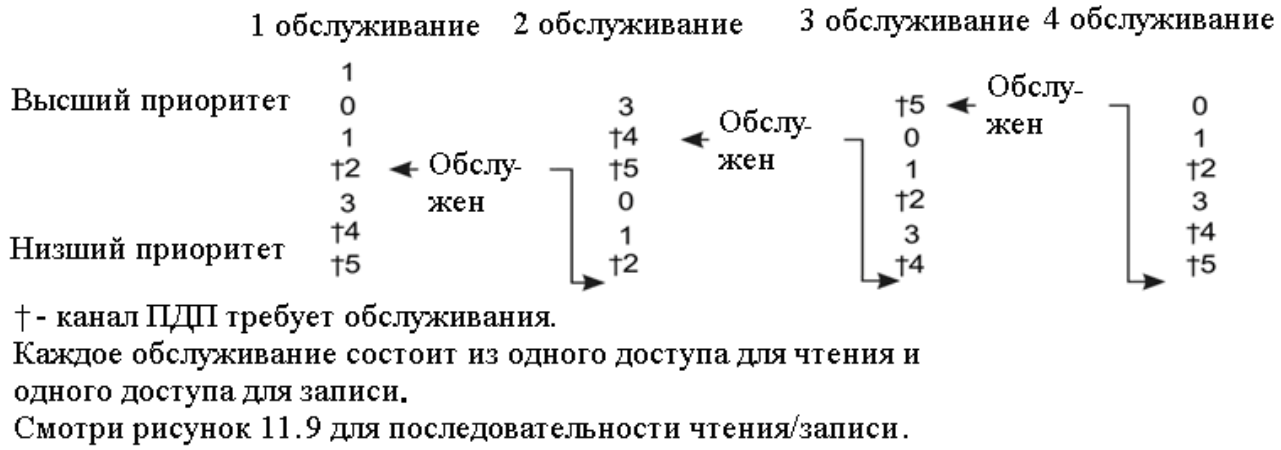
	Номер канала
Высший приоритет	0
	1
	2
	3
	4
Низший приоритет	5

Для выбора фиксированных приоритетов необходимо установить бит PRIORITY MODE (бит 30) регистра управления каналом ПДП канала 0 в «0».

11.6.2 Схема циклических приоритетов

В схеме с циклическими приоритетами каналу, который получил доступ к шинам, присваивается низший приоритет. Остальная последовательность каналов циклически сдвигается, т.е. каналу, следующему за каналом, который получил доступ к шинам последним, присваивается высший приоритет при следующем запросе. Приоритеты циклически изменяются по завершении каждой передачи. На рисунках 11.8 и 11.10 представлена цикличность приоритетов для нескольких обращений сопроцессора ПДП. При системном сбросе приоритеты каналов устанавливаются в порядке от высшего к низшему (0, 1, 2, 3, 4, 5).

Для выбора этой схемы необходимо установить бит PRIORITY MODE (разряд 30) регистра управления каналом ПДП канала 0 в «1».



†- канал ПДП требует обслуживания.
Каждое обслуживание состоит из одного доступа для чтения и одного доступа для записи.
Смотри рисунок 11.9 для последовательности чтения/записи.

Рисунок 11.8 – Пример схемы циклических приоритетов сопроцессора ПДП

При запуске на рисунке 11.8 каналы 2, 4, 5 запрашивают обслуживание (сервис). Так как канал 2 имеет высший приоритет, он обслуживается первым. Затем его приоритет становится низшим. Наивысший приоритет получает канал 3. В следующем обслуживании приоритеты каналов 4 и 5 изменяются аналогично. На рисунке 11.9 приведена полная последовательность чтения и записи.

Примечание – Каждое обслуживание означает один доступ чтения или один доступ записи. Сопроцессор ПДП устанавливает арбитраж каналов на основе «доступ за доступом», поэтому канал ПДП должен конкурировать между доступами чтения и записи и для общего режима и для режима разделения.

R – чтение; W – запись.

Рисунок 11.9 – Пример последовательности циклических приоритетов для чтения и записи (основной режим)

Рисунок 11.10 показывает такую же схему циклических приоритетов, как и на рисунке 11.8, но по-другому. Приоритеты уменьшаются от высшего к низшему по часовой стрелке. Приоритет канала, обслуженного последним, переходит по часовой стрелке и становится низшим.

Рисунок 11.10 – Пример циклических приоритетов

В схеме циклических приоритетов запрос любого канала ПДП будет обслужен после обслуживания всех запросов с более высоким приоритетом.

Максимальное число запросов: 5 – в основном режиме; 11 – в режиме с разделением.

Описанные выше схемы арбитража позволяют избежать монополизации системы одним каналом. Каналы ПДП, которые запущены и не синхронизированы посредством прерываний, всегда запрашивают обслуживание (сервис).

11.6.3 Арбитраж канала ПДП в режиме с разделением

Если канал ПДП запущен в режиме с разделением, то арбитраж между каналами осуществляется по схеме циклических приоритетов. Канал ПДП в режиме с разделением имеет такой же приоритет, как и канал ПДП в основном режиме. Единственный конфликт, который может произойти, это при арбитраже между основным и вспомогательным разделенными каналами. Изменение приоритетов разделенных каналов происходит по схеме с циклическими приоритетами.

Если канал ПДП в режиме с разделением одновременно запускается установкой битов START и AUX START, то выходной (основной) канал имеет более высокий приоритет, чем входной (вспомогательный) канал. Оба бита START и AUX START должны записываться в одно время, для того чтобы достичь условий сброса.

Схема приоритетов каналов для режима с разделением немного отличается от схемы приоритетов каналов для основного режима:

- Для каналов в основном режиме приоритеты изменяются после чтения или записи.
- Для основного и вспомогательного каналов в режиме с разделением приоритеты изменяются после завершения чтения и записи. Это является следствием того, что существует только один временный регистр для обоих каналов ПДП (основного и вспомогательного) для сохранения и чтения данных.

На рисунке 11.11 представлены 2 канала, конкурирующие за шину ПДП, это канал 2 (разделенный канал) и канал 4.

Высший приоритет 0
1
‡[2pri
2aux]
3
†4
Низший приоритет 5

Условные обозначения: † – запрос доступа каналом ПДП; ‡ – запрос доступа каналами ПДП в режиме разделения; 2pri – основной канал ПДП канала 2;
2aux – вспомогательный канал ПДП канала 2.

Рисунок 11.11 – Пример схемы приоритетов каналов в режиме разделения

Схема приоритетов канала, показанная на рисунке 11.11, последовательно представлена на рисунке 11.12. На схеме показано 8 этапов:

- Первое обслуживание является обслуживанием запроса основного канала ПДП канала 2 (2pri), который разделен. 2pri осуществляет чтение, а затем приоритет канала 2 становится низшим, но 2pri остается каналом с более высоким приоритетом в канале 2.
- Во втором обслуживании – канал 4 получает более высокий приоритет, считывает свой адрес источника и получает низший приоритет.
- При третьем обслуживании – значение, считанное посредством 2pri, записывается по адресу назначения и канал 2 получает низший приоритет. Также основной канал 2pri получает более низкий приоритет, чем вспомогательный канал 2aux.

Примечание – Разделенный канал, который только завершил чтение, остается с более высоким приоритетом, чем другой разделенный канал до тех пор, пока данные не будут записаны по адресу назначения.

- При четвертом обслуживании значение, считанное каналом 4 во втором обслуживании, теперь записывается по его адресу назначения, и канал получает низший приоритет.
- В пятом обслуживании подканал 2aux выполняет чтение, и канал 2 получает низший приоритет.
- В шестом обслуживании канал 4 снова выполняет чтение и получает низший приоритет.
- В седьмом и восьмом обслуживаниях значения подканала 2aux и канала 4, которые были прочитаны в обслуживаниях 5 и 6, теперь записываются по их адресу назначения. После записи – канал получает низший приоритет.

- При девятом обслуживании подканал 2pri снова выполняет чтение, как и в пятом обслуживании и циклы чтения/записи продолжаются, начиная с первого обслуживания.

R – чтение; W – запись.

Рисунок 11.12 – Пример последовательности смены приоритетов при обслуживании в режиме с разделением

11.7 Арбитраж ЦПУ и сопроцессора ПДП

Сопроцессор ПДП передает данные по своим внутренним шинам. Арбитраж необходим только тогда, когда возникает конфликт при одновременном обращении к источнику данных со стороны ЦПУ и сопроцессором ПДП. Арбитраж не должен иметь задержки. Если конфликт отсутствует, то ЦПУ и сопроцессор ПДП осуществляют доступ к источникам данных параллельно.

Все варианты арбитража между ЦПУ и сопроцессором ПДП построены на основе доступа, таким образом, что сопроцессор ПДП должен конкурировать за доступ для чтения/записи в основном режиме и режиме с разделением. Доступ сопроцессора ПДП к внутренней памяти начинается в течение H3, см. подраздел 8.4 «Разрешение конфликтов памяти» для более подробной информации.

Если ЦПУ и сопроцессор ПДП обращаются к одному и тому же источнику данных, то биты DMA PRI (биты 0 и 1 регистра управления каналом) определяют правила арбитража, как показано в таблице 11.7.

ЦПУ имеет более высокий приоритет, чем ПДП, если DMA PRI = 00_2; ЦПУ имеет более низкий приоритет, чем ПДП, если DMA PRI = 11_2. Приоритеты изменяются циклически, если DMA PRI = 01_2.

Таблица 11.7 – Значение битов DMA PRI и правила арбитража ЦПУ/ПДП

Биты DMA PRI 1, 0	Описание
0 0	Доступ сопроцессора ПДП имеет приоритет ниже приоритета доступа ЦПУ. Если каналы ПДП и ЦПУ обращаются к одному ресурсу памяти, то работает ЦПУ. Это значение устанавливается при сбросе
0 1	Это значение выбирает циклический арбитраж, который устанавливает приоритеты между ЦПУ и каналом ПДП посредством их альтернативного доступа, но не одинаково равнозначного. Приоритет циклически изменяется между доступами ЦПУ и каналом ПДП, когда они конфликтуют в течение последовательных циклов инструкций. С самого начала, если канал ПДП и ЦПУ запрашивают один и тот же ресурс памяти, то ЦПУ имеет приоритет перед сопроцессором ПДП. Если в следующем цикле сопроцессор ПДП и ЦПУ снова обращаются к одному и тому ресурсу памяти, то ПДП имеет более высокий приоритет, чем ЦПУ. Альтернативный доступ продолжается до тех пор, пока запросы ЦПУ и ПДП конфликтуют в течение последовательных циклов. Если в предыдущем цикле конфликта не было, то ЦПУ имеет более высокий приоритет.
1 0	Зарезервировано
1 1	Доступ сопроцессора ПДП имеет более высокий приоритет, чем приоритет доступа ЦПУ. Если каналы ПДП и ЦПУ обращаются к одному ресурсу памяти, работает ПДП.

11.8 Режимы передачи данных

Каждый канал ПДП может работать в 4 режимах передачи данных. Они отличаются друг от друга следующим:

- используется ли автоинициализация или нет;
- каким образом работает канал, если автоинициализация включена или нет.

Таблица 11.8 и информация в 11.8.1 – 11.8.4 описывают режимы передачи данных.

Таблица 11.8 – Описание поля TRANSFER MODE (AUX TRANSFER MODE)

Биты TRANSFER MODE 3, 2 (5, 4)	Описание
0 0	Передача данных не прерывается счетчиком передачи и не выполняется автоинициализация. TCINT (прерывание счетчика передачи) и AUX TCINT при этом могут использоваться для сообщения о прерывании, когда счетчик передачи переходит через «0». Канал ПДП продолжает работу.
0 1	Передача данных отменяется счетчиком передачи. Автоинициализация не выполняется. Код останова 10_2 помещается в поле START (или AUX START), когда передача данных будет завершена.
1 0	Автоинициализация № 1. Автоинициализация выполняется, когда счетчик передачи переходит в «0», без ожидания вмешательства ЦПУ.
1 1	Автоинициализация № 2. Канал ПДП автоинициализируется, когда ЦПУ повторно запускает сопроцессор ПДП, используя регистр ПДП в ЦПУ. Когда счетчик передачи обращается в «0», то операция останавливается до тех пор, пока ЦПУ не запустит сопроцессор ПДП, используя поле START (AUX START) в регистре управления каналом ПДП (биты 22, 23 и 24, 25 регистра управления каналом ПДП). Код останова 10_2 помещается в поле START (или AUX START) сопроцессором ПДП.

11.8.1 Работа в режиме TRANSFER MODE = 00_2

Когда TRANSFER MODE = 00_2, то передача данных не прерывается, если счетчик передачи переходит в «0» и автоинициализация не выполняется. Хотя счетчик передачи не останавливает передачу данных, но прерывание может быть сгенерировано при переходе счетчика передачи через «0», установив бит TCINT FLAG в «1». Если канал сопроцессора ПДП не был остановлен после достижения счетчиком передачи «0», счетчик продолжает декрементироваться.

11.8.2 Работа в режиме TRANSFER MODE = 01_2

Когда TRANSFER MODE = 01_2, то передача данных прерывается, если счетчик передачи переходит в «0» и автоинициализация не выполняется. Когда счетчик передачи переходит в «0», то канал ПДП останавливается записью 10_2 в поле START или AUX START.

11.8.3 Работа в режиме TRANSFER MODE = 10_2 (автоинициализация № 1)

Этот режим передачи позволяет каналу ПДП работать непрерывно, изменять указатели и синхронизацию с помощью процедуры автоинициализации и самостоятельно отключать ее. Поддерживаются 2 различных метода автоинициализации.

Метод автоинициализации № 1а всегда запускается после системного сброса, после сброса канала ПДП (00_2 записывается в биты START или AUX START), после останова канала ПДП (01_2 или 10_2 записывается в биты START или AUX START). Для выбора режима передачи 10_2 (метод автоинициализации № 1а) необходимо следующее (смотри рисунок 11.13):

- установить регистр управления каналом ПДП в режим передачи 10_2 и сбросить или остановить канал ПДП для автоинициализации;
- установить счетчик передачи в «0» (это выполняется при сбросе ПДП);
- в указатель канала ПДП записать адрес, по которому расположены значения автоинициализации. Инициализация других регистров канала ПДП не требуется, так как они автоматически устанавливаются в нужные значения в процессе автоинициализации;
- запустить канал ПДП, записав в биты START или AUX START 11_2;
- канал ПДП выполняет последовательность автоинициализации и передачу блока данных.

Рисунок 11.13 – Работа канала ПДП в режиме TRANSFER MODE = 10_2 (метод автоинициализации № 1а)

Метод автоинициализации № 1б) запускается, когда счетчик передачи не равен нулю. ПДП запускает передачу данных и автоинициализируется по завершении операции передачи, когда счетчик передачи переходит в «0». Для выбора режима передачи 10_2 (метод автоинициализации 1б) необходимо следующее (смотри рисунок 11.14):

- установить регистр управления каналом ПДП в режим передачи 10_2 и сбросить или остановить канал ПДП для первой операции передачи;
- установить остальные ПДП регистры (начального адреса, конечного адреса, счетчик передачи и т. д.) в соответствии с желаемой операцией передачи.

Примечание – Счетчик передачи теперь отражает число слов для передачи (обычное ненулевое значение) до процесса автоинициализации;

- в указатель канала ПДП записать адрес, по которому расположены значения автоинициализации для последовательных операций передачи данных;
- запустить канал ПДП, записав в биты START или AUX START 11_2.

Канал ПДП выполняет последовательность передачи блока данных и автоинициализации (обратный порядок в отличие от метода 1а).

Примечание – Если канал ПДП запрограммирован для выполнения передачи n блоков данных, то метод автоинициализации № 1а) требует n значений автоинициализации ПДП. Метод автоинициализации № 1б) требует n-1 значений автоинициализации ПДП, так как первая передача данных выполняется в течение инициализации передачи ПДП. Это означает уменьшение некоторого объема памяти, но последующие аналогичные операции ПДП требуют дополнительных (экстра) циклов ЦПУ для установки значений регистров ПДП снова.

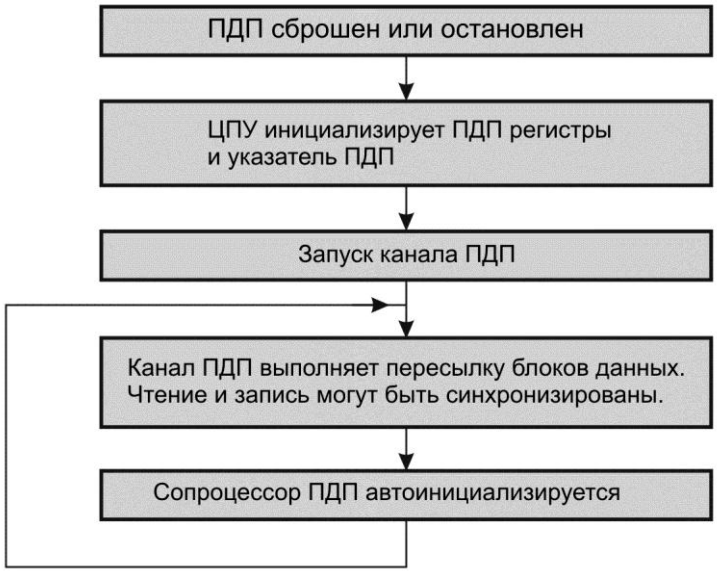


Рисунок 11.14 – Работа канала ПДП в режиме TRANSFER MODE = 10_2 (метод автоинициализации 1б)

11.8.4 Работа в режиме TRANSFER MODE = 11_2 (автоинициализация № 2)

Этот режим, помимо всех достоинств автоинициализации, позволяет ЦПУ легко управлять операциями канала ПДП. Поддерживается два разных метода автоинициализации.

Метод автоинициализации 2а) всегда запускается после системного сброса, после сброса канала ПДП (00_2 записывается в биты START или AUX START), после останова канала ПДП (01_2 или 10_2 записывается в биты START или AUX START). Для выбора

режима передачи 11_2 (метод автоинициализации 2а) необходимо следующее, см. рисунок 11.15:

- установить регистр управления каналом ПДП в режим передачи 11_2 и сбросить или остановить канал ПДП для автоинициализации;
- установить счетчик передачи в «0» (это выполняется при сбросе ПДП);
- в указатель канала ПДП записать адрес, по которому расположены значения автоинициализации. Инициализация других регистров канала ПДП не требуется, так как они автоматически устанавливаются в нужные значения в процессе автоинициализации;
- запустить канал ПДП, записав в биты START или AUX START 11_2;
- канал ПДП автоинициализируется и выполняет передачу блока данных;
- когда счетчик передачи обращается в «0», сопроцессор ПДП ожидает, пока ЦПУ запишет 11_2 в поле START (или AUX START) регистра управления каналом ПДП и автоинициализируется;
- повторение последовательности «автоинициализация-передача-ожидание»;
- когда счетчик передачи обращается в «0», то можно остановить канал ПДП записью 10_2 в поле START (или AUX START).

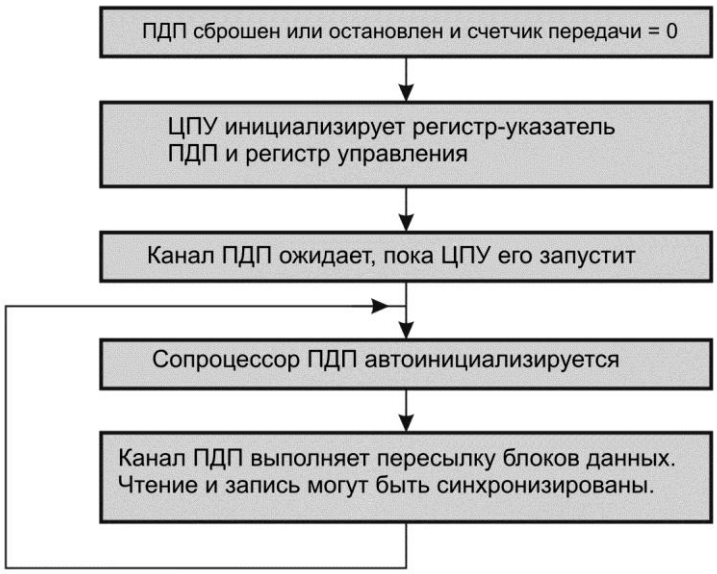


Рисунок 11.15 – Работа канала ПДП в режиме TRANSFER MODE = 11_2 (метод автоинициализации № 2а)

Метод автоинициализации № 2б) запускается, когда счетчик передачи не равен нулю. ПДП запускает передачу данных и автоинициализируется по завершении операции передачи (когда счетчик передачи переходит в «0»). Для выбора режима передачи 11_2 (метод автоинициализации № 2б) необходимо следующее (см. рисунок 11.16):

- установить регистр управления каналом ПДП в режим передачи 11_2 и сбросить или остановить канал ПДП для первой операции передачи;
- установить остальные ПДП регистры (начального адреса, конечного адреса, счетчик передачи и т.д.) в соответствии с желаемой операцией передачи.

Примечание – Счетчик передачи теперь отражает число слов для передачи (обычное ненулевое значение) до процесса автоинициализации;

- в указатель канала ПДП записать адрес, по которому расположены значения автоинициализации для последовательных операций передачи данных;
- запустить канал ПДП, записав в биты START или AUX START 11_2;

- канал ПДП выполняет инициализацию передачи данных. Когда счетчик передачи переходит в «0», ПДП ожидает, пока ЦПУ запишет 11_2 в поле START (или AUX START) регистра управления каналом ПДП и автоинициализируется;

- повторение последовательности «передача-ожидание-автоинициализация».

Примечание – Если канал ПДП запрограммирован для выполнения передачи n блоков данных, то метод автоинициализации № 2а) требует n значений автоинициализации ПДП. Метод автоинициализации № 2б) требует n-1 значений автоинициализации ПДП, так как первая передача данных выполняется в течение инициализации передачи ПДП. Это означает уменьшение некоторого объема памяти, но последующие аналогичные операции ПДП требуют дополнительных (экстра) циклов для установки значений регистров ПДП снова.

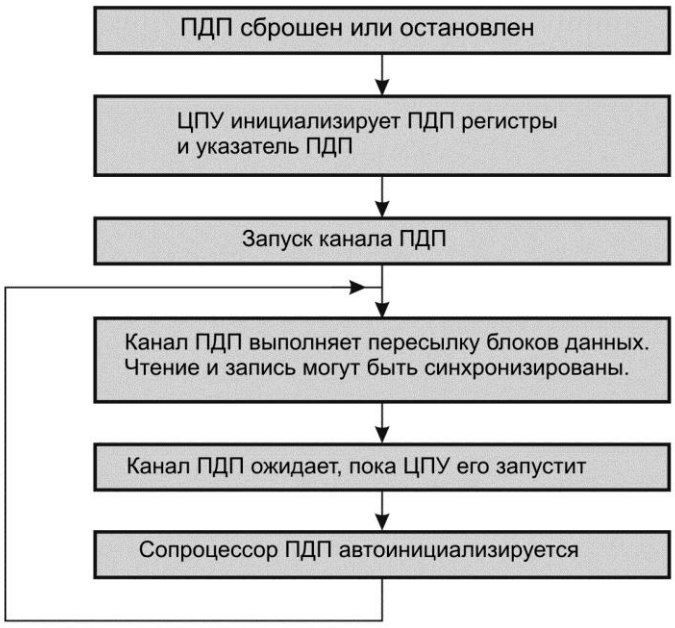


Рисунок 11.16 – Работа канала ПДП в режиме TRANSFER MODE = 11_2 (метод автоинициализации № 2б)

11.9 Автоинициализация

Автоинициализация – это метод перезагрузки регистрового файла канала ПДП, когда счетчик передачи переходит в «0». Когда канал ПДП оперирует в режиме автоинициализации, то регистр-указатель и вспомогательный регистр-указатель используются для инициализации регистров, которые управляют операциями канала ПДП. Эти указатели содержат начальный адрес блока данных, которые должны загружаться в регистровый файл ПДП, как показано на рисунке 11.1. Регистры-указатели описаны в 11.3.4.

Автоинициализация – это обычная операция передачи блока данных, при которой адресом назначения является регистровый файл сопроцессора ПДП. Сопроцессор ПДП считывает значение по адресу, указанному регистром-указателем и записывает его в регистр ПДП через периферийную шину в следующем доступном цикле. Соответственно, автоинициализация доступов чтения/записи также является источником конфликтов доступов ЦПУ/ПДП.

Автоинициализация может произойти:

- без вмешательства ЦПУ, если биты TRANSFER MODE=10_2 (автоинициализация № 1), смотри 11.8.3;

- с вмешательством ЦПУ, если биты TRANSFER MODE=11_2 (автоинициализация № 2). В этом случае ЦПУ должен перезапустить канал ПДП до выполнения автоинициализации, смотри 11.8.4;

- до передачи блока данных (метод автоинициализации № 1а). ПДП запускается, когда счетчик передачи равен «0», затем инициализируется и выполняет передачу блока данных;

- после передачи блока данных (метод автоинициализации № 1б). ПДП запускается, начиная с обычного блока передачи, а затем, когда счетчик передачи перейдет в «0», автоинициализируется.

Автоинициализации № 1 или № 2 могут использовать методы а) или б).

Автоинициализация зависит от текущего режима работы канала ПДП: основного режима или режима разделения. Режим операции управляется битом SPLIT MODE (бит 14 на рисунке 11.2).

Примечание – Во время автоинициализации сопроцессора ПДП не следует изменять бит SPLIT MODE. Этот бит следует изменять только тогда, когда сопроцессор ПДП сброшен или остановлен, смотри описание бита ПДП START в таблице 11.5 для более подробной информации.

11.9.1 Основной режим

Если канал ПДП запущен в основном режиме (SPLIT MODE = 0), то используется регистр-указатель, а регистры канала ПДП загружаются в следующем порядке:

- регистр управления каналом ПДП;
- регистр начального адреса;
- индексный регистр начального адреса;
- регистр-счетчик повторения;
- регистр конечного адреса;
- индексный регистр конечного адреса;
- регистр-указатель.

Размещение новых значений этих регистров в памяти показано на рисунке 11.17.

Рисунок 11.17 – Размещение новых значений регистров канала ПДП в памяти (SPLIT MODE=0)

11.9.2 Режим с разделением

Если канал ПДП запущен в режиме разделения (SPLIT MODE = 1), то последовательность автоинициализации зависит от того, какой счетчик передачи будет остановлен.

Если счетчик передачи переходит в «0» при SPLIT MODE = 1, то используется регистр-указатель для автоинициализации. В этом случае регистры ПДП загружаются в следующем порядке:

- регистр управления каналом ПДП;
- регистр начального адреса;
- индексный регистр начального адреса;
- регистр-счетчик передачи;
- регистр-указатель.

Размещение новых значений этих регистров в памяти показано на рисунке 11.18.

Рисунок 11.18 – Размещение новых значений регистров канала ПДП в памяти (SPLIT MODE = 1 и счетчик передачи = 0)

Если вспомогательный счетчик передачи переходит в «0» при SPLIT MODE =1, то используется вспомогательный регистр-указатель для автоинициализации. В этом случае регистры ПДП загружаются в следующем порядке:

- регистр управления каналом ПДП;
- регистр конечного адреса;
- индексный регистр конечного адреса;
- вспомогательный счетчик передачи;
- вспомогательный регистр-указатель.

Размещение новых значений этих регистров в памяти показано на рисунке 11.19.

Рисунок 11.19 – Размещение новых значений регистров канала ПДП в памяти (SPLIT MODE = 1 и вспомогательный счетчик передачи = 0)

11.9.3 Инкрементирование регистра-указателя

Во время автоинициализации регистр-указатель может инкрементироваться или оставаться постоянным:

- если указатель инкрементируется, то значения автоинициализации сохраняются последовательно в области памяти, а регистр-указатель и вспомогательный регистр-указатель инкрементируются в порядке доступа к областям памяти;

- если канал ПДП автоинициализируется с помощью устройства с пакетной передачей данных, например, внутреннего коммуникационного порта или внешних FIFO, то следует удерживать значение регистра-указателя постоянным.

Инкрементирование управляется битами AUTOINIT STATIC и AUX AUTOINIT STATIC регистра управления каналом ПДП:

- в основном режиме бит AUTOINIT STATIC управляет указателем;

- в режиме разделения AUTOINIT STATIC управляет указателем (основного канала) и AUX AUTOINIT STATIC управляет вспомогательным указателем. Если бит AUTOINIT STATIC (AUX AUTOINIT STATIC) равен «0», то регистр-указатель инкрементируется; если равен «1», то значение регистра-указателя не изменяется.

11.9.4 Синхронизация

Обычно, данные автоинициализации хранятся в памяти, а автоинициализация не нужна. В некоторых случаях требуется передать данные автоинициализации похожим образом, как и синхронизованные данные чтения/записи.

Синхронизация автоинициализации – это функция:

- битов SYNC MODE (биты 6 и 7 регистра управления каналом ПДП), которые управляют синхронизацией передачи данных;

- битов AUTOINIT SYNC (разряды 10 и 11 регистра управления каналом ПДП), которые влияют только на синхронизацию автоинициализации.

Если биты SYNC MODE не установлены для синхронизации передачи данных (т. е. если предшествующая передача данных не синхронизирована прерываниями), то последовательность автоинициализации канала ПДП также не синхронизируется. Если биты SYNC MODE устанавливаются для синхронной передачи данных (если предшествующая передача данных синхронизирована), тогда новая последовательность автоинициализации может быть синхронизирована для чтения или для записи, или и для того и другого одновременно (в зависимости от того, находится ли ПДП в основном режиме или в режиме разделения), как показано в таблице 11.9.

Примечание – Когда оба режима устанавливают «нет синхронизации», то последовательность автоинициализации канала ПДП не синхронизируется прерываниями.

В основном режиме синхронизация записи для операции автоинициализации отсутствует, так как адресом назначения является регистр ПДП, который уже готов.

В режиме с разделением бит 6 регистра управления каналом ПДП управляет синхронизацией автоинициализации основного канала ПДП, а бит 7 управляет синхронизацией автоинициализации вспомогательного канала ПДП.

Если используется синхронизация автоинициализации основного канала, то чтение значений автоинициализации ПДП, хранящихся в памяти, не производится до тех пор, пока не придет прерывание, определенное полем записи основного канала в регистре DIE.

Если используется синхронизация автоинициализации вспомогательного канала, то чтение значений автоинициализации ПДП, хранящихся в памяти, не производится до тех пор, пока не придет прерывание, определенное полем чтения вспомогательного канала в регистре DIE.

Таблица 11.9 – Влияние битов SYNC MODE и AUTOINIT MODE при автоинициализации

SYNC MODE	AUTOINIT SYNC	Основной режим	Режим разделения
Биты 7, 6	Биты 11, 10		
0 0	0 0	Нет синхронизации	Нет синхронизации
0 0	0 1	Нет синхронизации	Нет синхронизации
0 0	1 0	Нет синхронизации	Нет синхронизации
0 0	1 1	Нет синхронизации	Нет синхронизации
0 1	0 0	Нет синхронизации	Нет синхронизации
0 1	0 1	Чтение	Основной канал
0 1	1 0	Нет синхронизации	Нет синхронизации
0 1	1 1	Чтение	Основной канал
1 0	0 0	Нет синхронизации	Нет синхронизации
1 0	0 1	Нет синхронизации	Нет синхронизации
1 0	1 0	Нет синхронизации	Вспомогательный канал
1 0	1 1	Нет синхронизации	Вспомогательный канал
1 1	0 0	Нет синхронизации	Нет синхронизации
1 1	0 1	Чтение	Основной канал
1 1	1 0	Нет синхронизации	Вспомогательный канал
1 1	1 1	Чтение	Вспомогательный и основной каналы

11.9.5 Влияние битов регистра управления ПДП

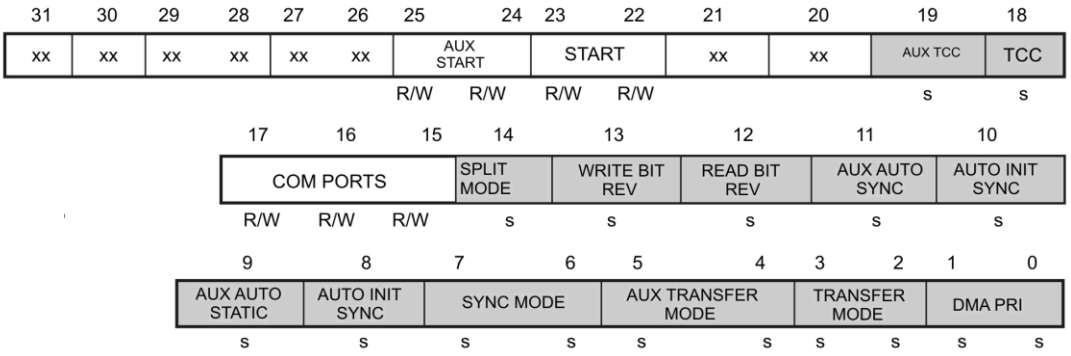
В основном режиме все записываемые биты регистра управления управляются автоинициализацией. Эти биты приведены на рисунке 11.20.

В режиме с разделением во время автоинициализации основного канала ПДП могут быть изменены только записываемые биты основного канала, а биты вспомогательного канала защищены от записи, смотри рисунок 11.21. Другими словами, с помощью ЦПУ или сопроцессора ПДП изменяются только биты основного канала. Также если автоинициализирован вспомогательный канал ПДП, то могут быть изменены только записываемые биты вспомогательного канала, а биты основного канала защищены от записи. Эти биты приведены на рисунке 11.22.

Хотя затененные биты (определенные символом «s» на рисунке 11.20) изменяются в течение автоинициализации, изменения вступают в силу только после завершения автоинициализации. Незатененные разряды вступают в силу немедленно, воздействуя на последовательность автоинициализации. Другими словами, при автоинициализации новые значения затененных разрядов вводятся последними после загрузки всех регистров (определенных регистром-указателем).

Несмотря на то, запущен ли канал ПДП в основном режиме или в режиме с разделением, то если ЦПУ или любой другой внешний источник данных выполняет запись в регистр управления каналом ПДП – это влияет на все записываемые разряды, в том числе и на затененные.

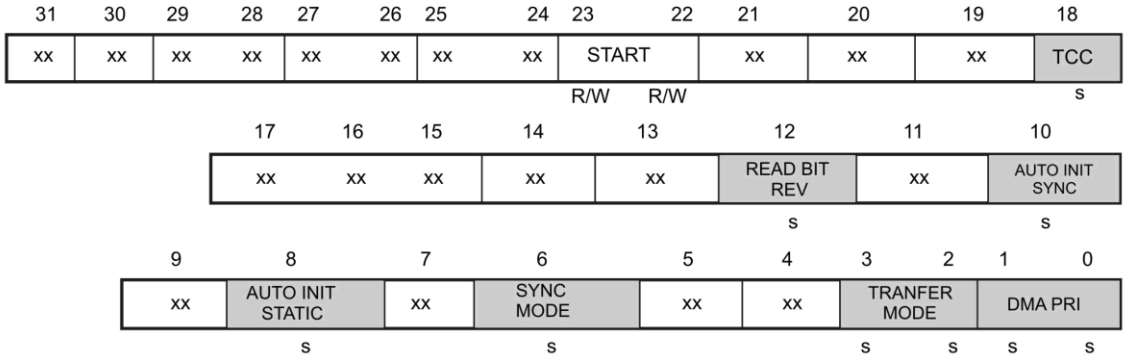
Примечание – Если ЦПУ пишет в регистр управления ПДП в течение автоинициализации, то новое значение вступает в силу только после завершения последовательности автоинициализации. Хотя операция автоинициализации регистров ПДП не подвергается воздействию, последовательность передачи данных может быть затронута.



s - Затененные биты не вступают в силу, пока автоинициализация не завершена.
xx - Защищенный от записи в течение автоинициализации.

R – чтение, R/W – чтение/запись.

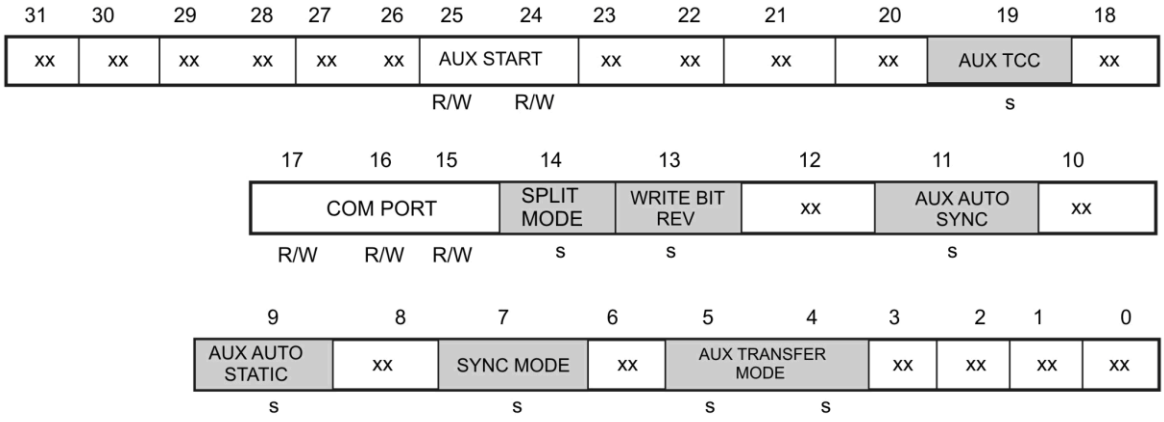
Рисунок 11.20 – Биты регистра управления каналом ПДП, изменяемые с помощью автоинициализации в основном режиме



s - Затененные биты не вступают в силу, пока автоинициализация не завершена.
xx - Защищенный от записи в течение автоинициализации.

R – чтение, R/W – чтение/запись.

Рисунок 11.21 – Биты регистра управления каналом ПДП, изменяемые с помощью автоинициализации основного канала в режиме с разделением



s - Затененные биты не вступают в силу, пока автоинициализация не завершена.

xx – Защищенный от записи в течение автоинициализации вспомогательного канала.

R – чтение, R/W – чтение/запись.

Рисунок 11.22 – Биты регистра управления каналом ПДП, изменяемые с помощью автоинициализации вспомогательного канала в режиме с разделением

11.9.6 Последовательные автоинициализации

Для многих приложений каждый раз необходимо инициализировать канал ПДП одними и теми же данными. В этом случае новое значение регистра-указателя указывает на начальный адрес одного и того же блока данных, содержащего новое значение указателя, как показано на рисунке 11.23. В этом отдельном примере предполагается, что канал не запущен в режиме с разделением.

Если это необходимо, то можно указать с помощью регистра-указателя на новые значения регистров, как показано на рисунке 11.24.

Рисунок 11.23 – Регистр-указатель, указывающий сам на себя

Рисунок 11.24 – Ссылка на новый регистр-указатель

11.10 Прямой доступ к памяти и прерывания

Сопроцессор ПДП использует прерывания следующим образом:

- Сопроцессор ПДП может посылать сигналы прерываний к ЦПУ, когда завершается передача блока данных, смотри биты TCC и AUX TCC на рисунке 11.2.

- Сопроцессор ПДП может принимать сигналы прерывания от внешних сигналов IIOF3#-IIOF0#, таймеров, коммуникационных портов (ICRDY, OCRDY).

В данном подразделе описано, как сопроцессор ПДП принимает прерывания. Этот процесс называется синхронизацией.

Все прерывания сопроцессора ПДП, которые он может «видеть», в первую очередь, приходят на контроллер прерываний ЦПУ. Прерывания по фронту фиксируются ЦПУ установкой соответствующего флага (прерывания по уровню не фиксируются).

Когда внешние прерывания IIOF3#-IIOF0# использованы для синхронизации передачи данных сопроцессором ПДП, то ЦПУ отвечает за конфигурирование внешних прерываний как по фронту, так и по уровню (в зависимости от битов FUNCx и TYPEx регистра флагов прерываний, описанного в 3.1.10).

Прерывания по фронту – это прерывания таймеров, прерывания от ПДП и внешние прерывания, которые сконфигурированы как прерывания по фронту. Подробнее описано в подразделах 7.4 «Прерывания», 7.6 «Прерывания сопроцессора ПДП». Когда контроллер прерываний определил, что прерывание по фронту и которое ожидает канал ПДП (разряды регистра DIE установлены в «1»), и которое зафиксировано регистром флагов, то ЦПУ сбрасывает флаг прерывания и посылает импульс прерывания к каналу ПДП. Канал ПДП локально фиксирует прерывание до тех пор, пока оно не будет обслужено. В то же время, зафиксированное прерывание сбрасывается сопроцессором ПДП за 2 цикла.

Прерывания по уровню, генерируемые коммуникационными портами и внешними прерываниями, которые сконфигурированы как прерывания по уровню, управляются контроллером прерываний ЦПУ по-разному. Когда контроллер прерываний определяет, что прерывание по уровню, которое ожидает канал ПДП (биты регистра DIE установлены в «1») получено, то ЦПУ посылает импульс прерывания к каналу ПДП. Канал ПДП локально фиксирует прерывание до тех пор, пока оно не будет обслужено. В то же время, зафиксированное прерывание сбрасывается сопроцессором ПДП за 2 цикла.

Сигнал сброса прерывания, генерируемый сопроцессором ПДП после того, как ПДП обслужит прерывание, имеет более высокий приоритет, чем сигнал установки прерывания. Поэтому, сигнал прерывания не будет последовательно устанавливаться, даже если ЦПУ последовательно посылает импульсы установки прерывания. Таким образом, когда используется схема приоритетов и канал ПДП с более высоким приоритетом управляется последовательностью сигналов прерывания, то канал ПДП с более низким приоритетом будет обслуживаться в промежутках времени между обслуживаниями высокоприоритетного канала.

Когда выборка конвейера остановлена, то это на процесс обработки прерываний не влияет. Когда прерывания разрешены в регистре DIE, то прерывания автоматически фиксируются контроллером прерываний ПДП и сохраняются для дальнейшего использования с помощью сопроцессора ПДП. Когда флаг прерывания (таймера, внешнего прерывания) зафиксирован, то IIF флаг сбрасывается.

Примечание – IIF флаги сбрасываются только тогда, когда контроллер прерываний ЦПУ фиксирует прерывание, но не когда ПДП отвечает на него. Даже если ПДП не был запущен, то фиксирование прерываний происходит, за исключением случая, когда биты запуска в регистре управления ПДП имеют значения сброса (START или AUX START установлены в 00_2).

Сброс ПДП сбрасывает зафиксированные прерывания. Во избежание потери ранее полученных прерываний рекомендуется инициализировать регистр DIE после запуска ПДП, когда биты запуска ПДП имеют значение 11_2.

Примечание – Когда ПДП завершает передачу данных, то биты запуска (AUX START) устанавливаются в 10_2. Поэтому ПДП не пропустит какое-либо прерывание между передачами данных.

Сопроцессор ПДП и ЦПУ могут отвечать на одно и то же прерывание, если ЦПУ не связан с конфликтом конвейера или с какой-либо инструкцией, которая останавливает выборку инструкций, см. 7.4.1 для более подробной информации. Разные каналы ПДП, включая вспомогательные и основные каналы, также могут отвечать на одно и то же прерывание. Если одно и то же прерывание выбрано для синхронизации адреса источника и приемника, то циклы чтения и записи разрешаются одним сигналом прерывания.

Внутренний конвейер ЦПУ гарантирует корректное взаимодействие между коммуникационным портом, который генерирует прерывания по уровню и каналом ПДП, который синхронизируется теми же прерываниями по уровню.

Примечание – Когда синхронизируются каналы ПДП по внешним прерываниям, то лучше конфигурировать сигналы прерываний в качестве сигналов прерываний по фронту, чтобы быть уверенным, что прерывание будет распознано только как одно.

11.10.1 Прерывания и синхронизация каналов ПДП

Прерывания можно использовать для синхронизации передачи каналом ПДП. Для установки синхронного режима передачи требуется следующее:

- установить биты SYNC MODE (биты 6, 7) регистра управления каналом ПДП в соответствии с требуемой схемой синхронизации источника данных и приемника данных, смотри 11.10.2 для получения более подробной информации;

- установить регистр DIE для разрешения соответствующего прерывания для выбранной синхронной передачи. На рисунках 11.25 и 11.26 показан регистр DIE для основного режима и режима с разделением, соответственно. В таблицах 11.10 и 11.11 представлены различные прерывания для синхронизации в основном режиме, а в таблицах 11.12 и 11.13 – для режима разделения.

Рекомендуется инициализировать регистр DIE после запуска ПДП, когда биты запуска имеют значение 11_2. Это предотвратит потерю ранее полученных прерываний, которые могут произойти, если включите регистр DIE, когда биты запуска сброшены в 00_2 (значение при сбросе).

R– чтение; W– запись.

Рисунок 11.25 – Функции битов регистра DIE для основного режима ПДП

Таблица 11.10 – Синхронизация прерываний ПДП каналов 0 и 1 (ПДП0 и ПДП1) для основного режима

Значение битов (в ПДП0 или ПДП1)	Прерывание разрешено для ПДП0 или ПДП1				Источник прерываний для синхронизации ПДП
	ПДП0		ПДП1		
	чтение	запись	чтение	запись	
0 0†	Нет	Нет	Нет	Нет	--
0 1	ICRDY0	OCRDY0	ICRDY1	OCRDY1	От коммуникационного порта
1 0	IIOF0#	IIOF1#	IIOF2#	IIOF3#	От внешних сигналов IIOF3#-IIOF0#
1 1	TIM0	TIM0	TIM0	TIM0	От таймера TIM0
Примечание – † – Канал ПДП остановлен (операции чтения или записи не выполняются), если используется синхронная передача данных.					

Таблица 11.11 – Синхронизация прерываний ПДП каналов от 2 до 5 (от ПДП2 до ПДП5) для основного режима

Значение битов (от ПДП2 до ПДП5)	Прерывание разрешено для каналов от ПДП2 до ПДП5†		Источник прерываний для синхронизации ПДП
	ПДПх чтение †	ПДПх запись †	
0 0 0‡	Нет	Нет	--
0 0 1	ICRDYx †	OCRDYx †	От коммуникационного порта
0 1 0	IIOF0#	IIOF0#	От внешних сигналов IIOF3#-IIOF0#
0 1 1	IIOF1#	IIOF1#	
1 0 0	IIOF2#	IIOF2#	
1 0 1	IIOF3#	IIOF3#	
1 1 0	TIM0	TIM0	От таймеров TIM0 и TIM1
1 1 1	TIM1	TIM1	
Примечания 1 † – «х» в ПДПх обозначает номер канала ПДП, а также номер соответствующего прерывания ICRDYx и OCRDYx. Например, если и в ПДП2 чтение и в ПДП5 запись записано 001_2, то разрешаются прерывания ICRDY2 и OCRDY2, соответственно. Остальные значения разрядов (010_2 и 111_2) такие же (как показано в таблице 11.11) для каналов от ПДП2 до ПДП5. 2 ‡ – Канал ПДП остановлен (операции чтения или записи не выполняются), если используется синхронная передача данных.			

R– чтение; W– запись.

Рисунок 11.26 – Функции битов регистра DIE для режима разделения ПДП

Таблица 11.12 – Синхронизация прерываний ПДП каналов 0 и 1 (ПДП0 и ПДП1) для режима с разделением

Значение разрядов (в ПДП0 или ПДП1)	Прерывание разрешено для ПДП0 или ПДП1				Источник прерываний для синхронизации ПДП
	Вспомогательный канал ПДП0 чтение	Основной канал ПДП0 запись	Вспомогательный канал ПДП1 чтение	Основной канал ПДП1 запись	
0 0†	Нет	Нет	Нет	Нет	-
0 1	ICRDY0	OCRDY0	ICRDY1	OCRDY1	От коммуникационного порта
1 0	IIOF0#	IIOF1#	IIOF2#	IIOF3#	От внешних сигналов IIOF3#-IOF0#
1 1	TIM0	TIM0	TIM0	TIM0	От таймера TIM0
Примечание – † – Канал ПДП остановлен (операции чтения или записи не выполняются), если используется синхронная передача данных.					

Таблица 11.13 – Синхронизация прерываний ПДП каналов от 2 до 5 (от ПДП2 до ПДП5) для режима разделения

Значение разрядов (от ПДП2 до ПДП5)	Прерывание разрешено для каналов от ПДП2 до ПДП5†		Источник прерываний для синхронизации ПДП
	Вспомогательный канал ПДПх чтение †	Основной канал ПДПх запись †	
0 0 0‡	Нет	Нет	-
0 0 1	ICRDYx †	OCRDYx †	От коммуникационного порта
0 1 0	IIOF0#	IIOF0#	От внешних сигналов IIOF3#-IIOF0#
0 1 1	IIOF1#	IIOF1#	
1 0 0	IIOF2#	IIOF2#	
1 0 1	IIOF3#	IIOF3#	
1 1 0	TIM0	TIM0	От таймеров TIM0 и TIM1
1 1 1	TIM1	TIM1	
Примечания 1 † – «x» в ПДПх обозначает номер канала ПДП, а также номер соответствующего прерывания ICRDYx и OCRDYx. Например, если и в ПДП2 чтение и в ПДП5 записано 001_2, то разрешаются прерывания ICRDY2 и OCRDY2, соответственно. Остальные значения битов (010_2 и 111_2) такие же (как показано в таблице) для каналов от ПДП2 до ПДП5. 2 ‡ – Канал ПДП остановлен (операции чтения или записи не выполняются), если используется синхронная передача данных.			

11.10.2 Биты режима синхронизации

Таблицы 11.3 и 11.4 показывают, как значение поля SYNC MODE регистра управления каналом ПДП определяет синхронизацию в основном режиме и режиме с разделением соответственно:

- нет синхронизации (SYNC MODE=00_2);
- синхронизация источника данных;
- для основного режима (SYNC MODE=01_2);
- для режима разделения (SYNC MODE=10_2);
- синхронизация приемника данных;

- для основного режима (SYNC MODE=10_2);
- для режима разделения (SYNC MODE=01_2);
- синхронизация источника и приемника данных (SYNC MODE=11_2).

Когда сопроцессор ПДП находится в режиме с разделением, то основной канал поддерживает только синхронизацию записи (приемника) передаваемых данных, вспомогательный канал поддерживает только синхронизацию чтения (источника) передаваемых данных. В режиме с разделением биты 6 и 7 регистра управления каналом ПДП, смотри таблицу 11.3, используются для управления синхронизацией канала:

- бит 6 управляет синхронизацией записи основного канала (синхронизация приемника);
- бит 7 управляет синхронизацией чтения вспомогательного канала (синхронизация источника).

Скорость передачи данных ПДП в режиме с синхронизацией описана в 11.11.2.

11.10.2.1 Биты режима без синхронизации

Когда SYNC MODE=00_2, то синхронизация не выполняется. ПДП выполняет циклы чтения и записи, пока имеет приоритет для использования шины ПДП. Все прерывания игнорируются.

Примечание – Есть разница между этим режимом и режимом, когда в поля чтения или записи регистра DIE записаны нули. Если в полях чтения/записи регистра DIE записаны нули, то происходит полная остановка ПДП, если используется синхронизация, однако, при состоянии SYNC MODE = 00_2 канал ПДП свободно работает. Рисунок 11.27 показывает используемый при SYNC MODE = 00_2 механизм.

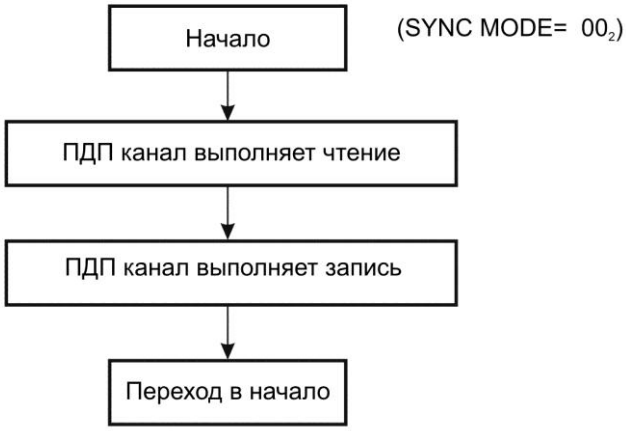


Рисунок 11.27 – Нет синхронизации ПДП (SYNC MODE = 00_2)

11.10.2.2 Синхронизация с источником данных

Когда SYNC MODE=01_2 (для основного режима) или когда SYNC MODE=10_2 (для вспомогательного канала в режиме с разделением), сопроцессор ПДП синхронизируется с источником данных, см. рисунок 11.28. Чтение не выполняется, пока канал ПДП не получит сигнал прерывания. Затем все прерывания ПДП глобально запрещаются. Однако в регистре разрешения прерываний ПДП разряды не изменяются.

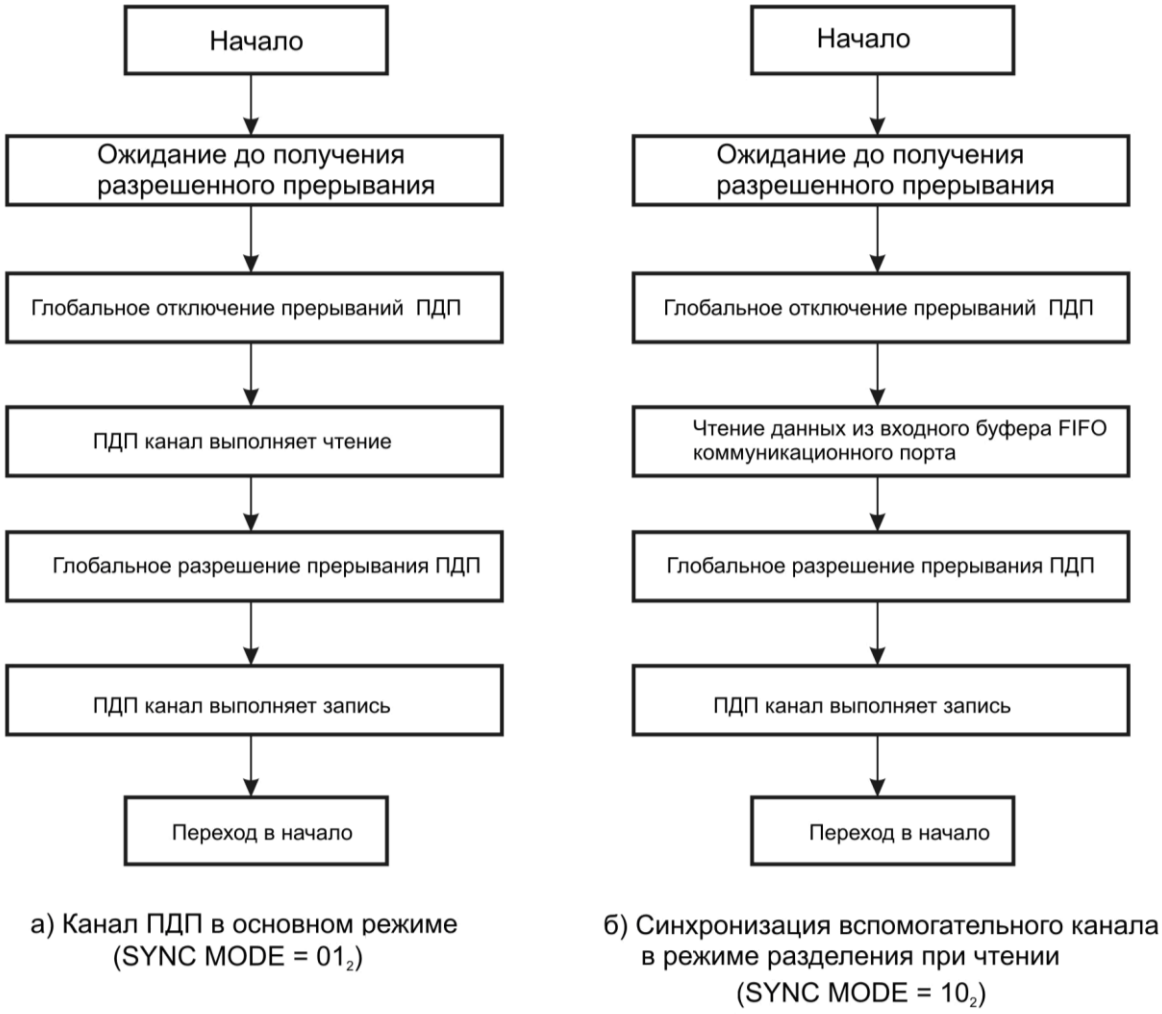


Рисунок 11.28 – Синхронизация источника ПДП

а) канал ПДП в основном режиме SYNC MODE = 01_2;

б) синхронизация вспомогательного канала в режиме разделения при чтении SYNC MODE = 10_2

11.10.2.3 Синхронизация с приемником данных

Когда SYNC MODE=10_2 (для основного режима) или когда SYNC MODE=01_2 (для основного канала в режиме с разделением), сопроцессор ПДП синхронизируется с приемником данных (смотри рисунок 11.29). Запись не выполняется, пока канал ПДП не получит сигнал прерывания.

В основном режиме чтение выполняется без ожидания прерывания. Однако в режиме с разделением чтение выполняется только тогда, когда получено прерывание, которое разрешает запись. Во избежание ситуации блокировки, которая может произойти, так как основной канал выполняет чтение, но никогда не пишет из временного регистра, потому что он не получает прерывание записи. В этом случае вспомогательный канал не сможет работать, так как внутренний временный регистр ПДП занят.

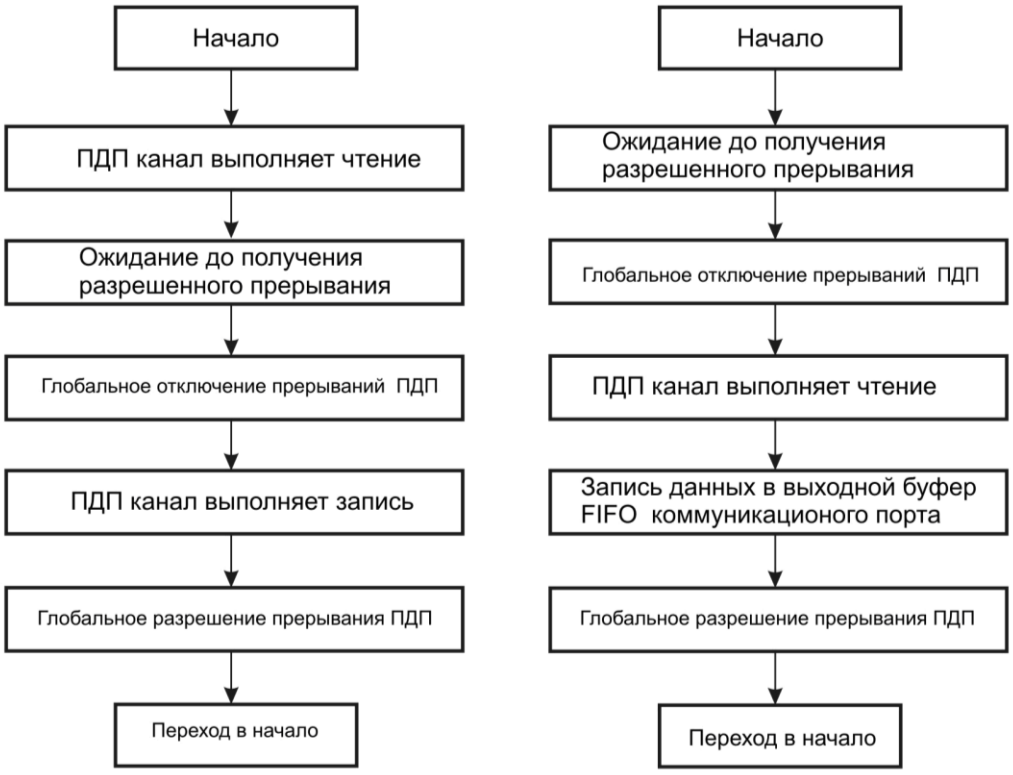


Рисунок 11.29 – Синхронизация приемника ПДП

а) канал ПДП в основном режиме SYNC MODE = 10_2;

б) синхронизация вспомогательного канала в режиме разделения при чтении SYNC MODE = 01_2

11.10.2.4 Синхронизация с источником и приемником данных

Когда SYNC MODE=11_2, то чтение выполняется, когда приходит прерывание для чтения, запись выполняется, когда приходит прерывание для записи. Если прерывание для записи получено до прерывания для чтения, то прерывание для записи фиксируется, и запись данных ПДП не выполняется, пока не закончится чтение. Синхронизация источника и приемника данных (SYNC MODE=11_2) показана на рисунке 11.30.

Если выбран режим с разделением, он представляется как 2 независимые синхронизации для основного (синхронизация записи) и вспомогательного (синхронизация чтения) каналов, см. рисунки 11.28б) и 11.29б).

Если выбрано одно и то же прерывание и для чтения, и для записи (как в основном режиме, так и в режиме разделения), то только одно единственное прерывание разрешает операции чтения и записи.

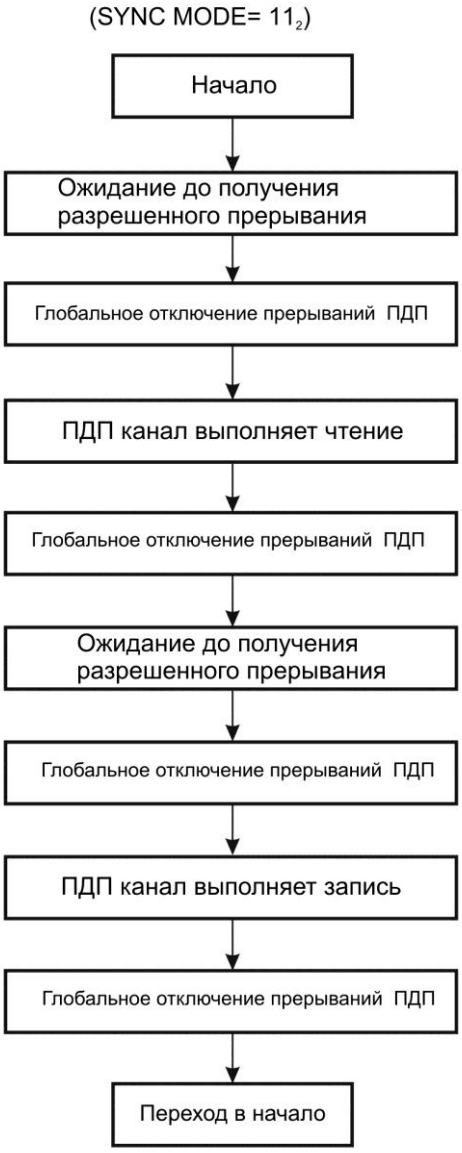


Рисунок 11.30 – Синхронизация источника и приемника ПДП в основном режиме **(**SYNC MODE = 11_2)

11.11 Временные диаграммы передачи данных памяти ПДП

ИС 1867ВМ9Ф поддерживает шесть каналов ПДП (12 каналов ПДП, если они все в режиме с разделением) со схемами арбитража с фиксированными/циклическими приоритетами и конфигурируемыми схемами приоритетов ЦПУ/ПДП, см. 11.6 и 11.7.

Максимальная скорость передачи данных возможна, если ИС 1867ВМ9Ф передает одно слово каждые 2 цикла. Шесть каналов ПДП передают данные в последовательности, разделенной по времени, а не одновременно, так как они используют общие шины.

Временные диаграммы передачи данных памяти сопроцессором ПДП могут быть очень сложными, особенно если возникает конфликт источника. Однако некоторые правила помогут посчитать время передачи данных для определенных установок ПДП. Для упрощения 11.11.1 и 11.11.2 описывают одноканальную передачу данных без конфликтов с ЦПУ или другими каналами ПДП. Можно также получить реальные времена передачи данных ПДП с помощью совокупного подсчета для одноканальных передач с учетом конфликтных ситуаций.

11.11.1 Диаграмма одноканальной передачи данных памяти ПДП

Если передача данных памяти ПДП не имеет конфликтов с ЦПУ или другими каналами ПДП, то число циклов передачи зависит от того, находятся ли адреса источника и приемника во внутренней памяти, периферии или во внешних портах. Если используется внешний порт, то скорость передачи данных зависит от двух факторов: состояния ожидания внешней шины и конфликта чтения/записи (например, если запись следует за чтением, чтение занимает 2 цикла). На рисунках 11.31 – 11.33 показано требуемое число циклов передачи данных ПДП от различных источников к разным приемникам. Содержимое рисунка 11.31 показывает число циклов, необходимое для Т передач, предполагая, что нет конфликтов конвейера.

Циклы	T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Источник и приемник во внутренней памяти	9	R		R		R		R		R		R		R		R		R	
			W		W		W		W		W		W		W		W		W
Источник на локальной шине Приемник во внутренней памяти	4	R	R	R		R	R	R		R	R	R		R	R	R			
			Cr				Cr				Cr				Cr				
					W				W				W				W		
Источник на глобальной шине Приемник во внутренней памяти	4	R	R	R		R	R	R		R	R	R		R	R	R			
			Cr				Cr				Cr				Cr				
					W				W				W				W		

Адрес источника Адрес приемника: внутренний
Внутренний (1+1)T
Локальная шина [(1+Cr)+1]T
Глобальная шина [(1+Cr)+1]T

Примечание – Принятые условные обозначения:
T – число передач;
Cr – состояние ожидания чтения источника;
R – одноцикловые чтения;
W – одноцикловые записи;
R R – многоцикловые чтения.

Рисунок 11.31 – Диаграмма и число циклов передачи данных ПДП во внутренний адрес

Циклы	T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Источник во внутренней памяти	4	R		R				R				R							
Приемник на локальной шине			W	W	W	W	W	W	W		W	W	W	W	W	W	W		
					Cw			Cw				Cw				Cw			
Источник на локальной шине	2	R	R	R					R	R	R	R							
Приемник на локальной шине			Cr								Cr								
					W	W	W	W					W	W	W	W			
							Cw								Cw				
Источник на глобальной шине	3	R	R	R		R	R		R	R	R								
Приемник на локальной шине			Cr				Cr				Cr								
					W	W	W	W	W	W	W	W	W	W	W	W			
							Cw				Cw				Cw				

Адрес источника	Адрес приемника: локальная шина
Внутренний	1+(2+Cw)T
Локальная шина	[(2+Cr)+(2+Cw)]T-1
Глобальная шина	[(1+Cr)+(2+Cw)]+[2+max(Cr,Cw)](T-1)

Примечание – Принятые условные обозначения:

T – число передач;

Cr – состояние ожидания чтения источника;

Cw – состояние ожидания записи в приемник;

R – одноцикловые чтения;

R R – многоцикловые чтения;

W – одноцикловые записи;

W W – многоцикловые записи.

Рисунок 11.32 – Диаграмма и число циклов передачи данных ПДП на локальную шину

Запись на локальную и глобальную шину занимает минимум 2 цикла, но внутри кристалла для ПДП/ЦПУ требуется один цикл для выполнения записи во внешнюю память. Таким образом, ПДП/ЦПУ могут передавать данные в следующем цикле, если не на ту же самую внешнюю шину. Например, ПДП передает 1024 слова из внутренней памяти блока СОЗУ 1 в память с одним состоянием ожидания на глобальную шину, пока ЦПУ выбирает инструкции из области локальной шины и выбирает операнды из блока СОЗУ 0, то время передачи ПДП, в соответствии с рисунком 11.32:

Тп=1+(2+1)1024 = 1 + 3072 = 3073 циклов.

Циклы	T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Источник во внутренней памяти	4	R		R				R				R							
			W W W W				W W W W				W W W W				W W W W			
Приемник на глобальной шине					Cw				Cw				Cw				Cw		
Источник на локальной шине	3	R R R				R R R				R R R									
			Cr				Cr				Cr								
					W W W W				W W W W				W W W W						
Приемник на глобальной шине							Cw				Cw				Cw				
Источник на глобальной шине	2	R R R							R R R R										
			Cr								Cr								
					W W W W								W W W W						
Приемник на глобальной шине							Cw								Cw				

Адрес источника — Адрес приемника: глобальная шина

Внутренний: $1+(2+Cw)T$

Локальная шина: $[(1+Cr)+(2+Cw)]+[2+\max(Cr,Cw)](T-1)$

Глобальная шина: $[(2+Cr)+(2+Cw)]T-1$

Примечание – Принятые условные обозначения:
T – число передач;
Cr – состояние ожидания чтения источника;
Cw – состояние ожидания записи в приемник;
R – одноцикловые чтения;
R R – многоцикловые чтения;
W – одноцикловые записи;
W W – многоцикловые записи.

Рисунок 11.33 – Диаграмма и число циклов передачи данных ПДП на глобальную шину

11.11.2 Скорость передачи данных ПДП в режиме синхронизации

Режим синхронизации, используемый для передачи данных, влияет на скорость передачи ПДП. Скорость передачи меньше, если синхронизация используется, так как она занимает 2 цикла для сброса запроса от прерывания. Однако эти два дополнительных цикла могут поглощаться, если несколько каналов ПДП запущены одновременно.

В основном режиме при использовании синхронизации максимальная скорость передачи – это одно слово каждые 3 цикла. Рисунок 11.34 показывает число циклов передач ПДП, требуемых в основном режиме при различных синхронизациях. Для упрощения описывается одноканальная передача данных памяти ПДП без конфликтов с ЦПУ или с другими каналами, без состояний ожидания памяти со всегда включенными прерываниями.

Циклы	T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Нет синхронизации	9	R		R		R		R		R		R		R		R		R	
			W		W		W		W		W		W		W		W		W
Синхронизация чтения	6	R			R			R			R			R			R		
			Rr			Rr			Rr			Rr			Rr			Rr		
			W			W			W			W			W			W	
Синхронизация записи	5	R		R			R			R			R						
			W			W			W			W			W				
				Wr			Wr			Wr			Wr			Wr			
Синхронизация чтения и записи	5	R			R			R			R			R					
			Rr			Rr			Rr			Rr			Rr				
			W			W			W			W			W				
				Wr			Wr			Wr			Wr			Wr			

Синхронизация	Длительность
Нет синхронизации	2T
Синхронизация чтения	3T
Синхронизация записи	1+3T
Синхронизация чтения и записи	1+3T

Примечание – Принятые условные обозначения:
T – число передач;
R – одноцикловые чтения;
W – одноцикловые записи;
Rr – сброс флага чтения (2 цикла);
Wr – сброс флага записи (2 цикла).

Рисунок 11.34 – Диаграмма основного режима ПДП при различных синхронизациях

В режиме с разделением при использовании синхронизации максимальная скорость передачи и для основного, и для вспомогательного канала – одно слово каждые 4 цикла. Когда вспомогательный и основной каналы запущены одновременно, то дополнительные два цикла, необходимые для сброса прерываний, поглощаются, и максимальная скорость может быть равна одному слову за каждые 2 цикла.

Рисунок 11.35 показывает число циклов передач ПДП, требуемых в режиме разделения при различных синхронизациях. Для упрощения описывается одноканальная передача данных памяти ПДП без конфликтов с ЦПУ или с другими каналами, без состояний ожидания памяти и со всегда включенными прерываниями.

Циклы	Т	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Нет синхронизации (оба канала запущены)	8	R				R				R				R					
			W				W				W				W				
				R				R				R				R			
					W				W				W				W		
Синхронизация основного канала (вспомогательный канал не запущен)	4	R				R				R				R					
			W				W				W				W				
				Pr				Pr				Pr				Pr			
Синхронизация вспомогательного канала (основной канал не запущен)	4	R				R				R				R					
			W				W				W				W				
				Ar				Ar				Ar				Ar			
Синхронизация основного и вспомогательного каналов (оба канала запущены)	8	R				R				R				R					
			W				W				W				W				
				Pr				Pr				Pr				Pr			
				R				R				R				R			
					W				W				W				W		
						Ar				Ar				Ar					Ar

Синхронизация	Длительность
Нет синхронизации (оба канала запущены)	2Т
Синхронизация основного канала (вспомогательный канал не запущен)	4Т
Синхронизация вспомогательного канала (основной канал не запущен)	4Т
Синхронизация основного и вспомогательного каналов (оба канала запущены)	2Т+2

Примечание – Принятые условные обозначения:
Т – число передач;
R – одноцикловое чтение основного или вспомогательного каналов;
W – одноцикловая запись основного или вспомогательного каналов;
Pr – сброс флага основного канала (2 цикла);
Ar – сброс флага дополнительного канала (2 цикла).

Рисунок 11.35 – Диаграмма основного режима ПДП при различных синхронизациях

12 Коммуникационные порты

ИС 1867ВМ9Ф имеет шесть коммуникационных портов (далее – порт) для связи с другими микросхемами, имеющими аналогичный интерфейс или высокоскоростные внешние устройства. Одна важная функция портов состоит в том, что они могут работать с сопроцессором прямого доступа к памяти, чтобы передать либо принять данные без вмешательства центрального процессора, что позволяет центральному процессору выполнять другие задачи.

Раздел 12 описывает главные особенности карты памяти, регистры и операции, выполняемые коммуникационными портами ИС.

12.1 Основные функции коммуникационных портов

Каждый коммуникационный порт ЦОС имеет несколько основных характеристик:

- Максимальная скорость двунаправленной передачи данных 512 Мбайт в секунду (при 12,5 нс процессорном цикле).
- Простая связь между микропроцессорами через восьмиразрядные шины данных и четыре сигнала управления передачей.
- Буферизация памяти FIFO всех передаваемых/принимаемых данных.
- Автоматический арбитраж и процедура установления связи, чтобы гарантировать синхронизацию связи.
- Синхронизация коммуникационных портов между центральным процессором или сопроцессором ПДП через внутренние прерывания и внутренние сигналы готовности.
- Поддержка широкого разнообразия многопроцессорных архитектур, включая кольца, гиперкубы, двунаправленные конвейеры, двумерные Евклидовые сети, гексагональные сетки и трехмерные сети.
- Программный сброс коммуникационного порта.

12.2 Краткий обзор коммуникационных портов

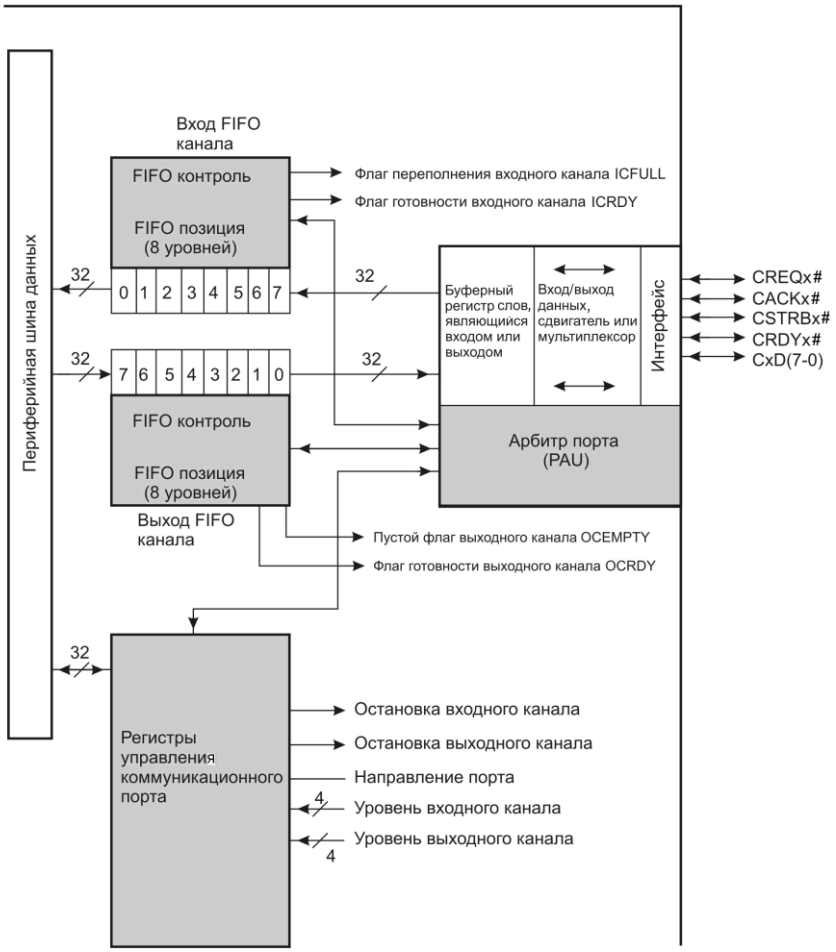
ИС 1867ВМ9Ф имеет шесть идентичных быстродействующих портов, каждый из которых обеспечивает двунаправленную связь с подобными ИС или другими внешними устройствами. Рисунок 12.1 показывает конструкцию одного коммуникационного порта.

Каждый порт содержит следующие компоненты:

- Входной буфер FIFO имеет восемь уровней 32-разрядных слов. Входной буфер FIFO изолирует ЦОС от случайных внешних данных на шине данных порта и буферизирует синхронизированные полученные данные.
- Выходной буфер FIFO имеет восемь уровней 32-разрядных слов. Выходной буфер FIFO изолирует от случайных внутренних данных на шине данных порта и буферизирует синхронизированные передаваемые данные.
- Модуль арбитра порта PAU – координирует работу коммуникационных портов в соответствии с поставленной задачей. Подробно арбитр описывается в подразделе 12.4.
- Регистр инструкций коммуникационного порта CPCR позволяет управлять и контролировать функциями коммуникационного порта и операциями передачи данных между ИС и внешними устройствами.
- Регистр программного сброса коммуникационного порта позволяет очистить входной буфер FIFO и выходной буфер FIFO порта. Это объясняется в 12.3.4.

Коммуникационный порт передает 32-разрядное слово, сохраненное в его выходном байтовом буфере FIFO. Поскольку сигналы управления передачей данных и шины данных портов двунаправлены, то каждый порт должен получить право монопольно использовать информационные шины коммуникационного порта, прежде чем начать передачу данных (назовем передачу права монопольного владения шиной данных – передача «жетона»). Коммуникационный порт, получивший «право» имеет монопольное право на передачу данных. Таким образом «жетон» используется, чтобы определять владеет ли коммуникационный порт монопольно шиной данных.

Примечание – Далее фраза «получение/передача жетона» имеет тот же смысл, что и фраза «передача/прием права».



Примечание – «x» принимает значения от 0 до 5.

Рисунок 12.1 – Блок-схема коммуникационного порта

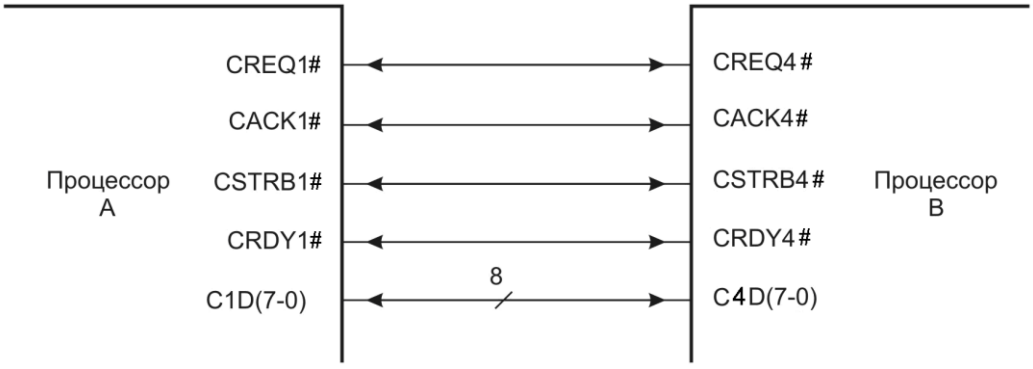


Рисунок 12.2 – Пример соединения двух ИС 1867ВМ9Ф через коммуникационные порты

Рисунок 12.2 является примером соединения двух ИС 1867ВМ9Ф через коммуникационные порты. Это соединение происходит через соединение сигналов управления передачи данных и шины передачи данных:

- CREQx# – это сигнал запроса передачи «жетона» коммуникационным портом. ИС активирует этот сигнал для запроса использования шины данных коммуникационного порта.

- CACKx# – это сигнал подтверждения передачи «жетона» коммуникационным портом, монопольно владеющим шиной. ИС активирует этот сигнал, когда передает право на монопольное использование информационной шины, этот сигнал является откликом на сигнал запроса указателя передачи CREQx# от другого ЦОС.

- CSTRBx# – это сигнал строба шины данных коммуникационного порта. ИС активирует этот сигнал, указывая на то, что он «разместил» байт достоверных данных на информационной шине порта.

- CRDYx# – это сигнал готовности коммуникационного порта. ИС активирует этот сигнал, указывая на то, что байт данных получен через информационную шину коммуникационного порта.

- CxD7-CxD0 – это сигналы шины данных коммуникационного порта. Эта шина служит для двунаправленной передачи данных между ЦОС или другими внешними устройствами.

12.2.1 Операция передачи «жетона»

Чтобы передавать «жетон», арбитры двух ИС «договариваются» между собой, чтобы сгенерировать сигналы и управляющие последовательности, необходимые для гарантированной передачи данных. Чтобы избежать конфликтов на шине данных, арбитры выносят решение о том, какой ИС передать монопольное владение шиной данных.

С помощью сигналов CREQx# и CACKx# арбитры двух ИС реализуют процедуру установления связи между двумя ИС.

Взаимодействие происходит следующим образом:

- Арбитр, который не имеет монопольного доступа к информационной шине CxD7-CxD0, активирует сигнал CREQx# для запроса на монопольное использование шины.

- Арбитр, который имеет монопольный доступ к информационной шине, в ответ активирует сигнал CACKx#, чтобы подтвердить запрос и передать монопольное использование шины запрашивающему арбитру.

Таким способом эти сигналы передают «жетон» передачи (или приоритет) от одного арбитра к другому, смотри подраздел 12.4.1 для детального рассмотрения данного вопроса.

12.2.2 Операция передачи данных

Операция передачи данных происходит в четыре основных этапа.

Этап 1. ЦПУ или сопроцессор ПДП записывает 32-разрядные данные в выходной буфер FIFO (коммуникационного порта) через адрес коммуникационного порта, находящегося в карте памяти. Карта памяти представлена на рисунке 12.3.

Этап 2. Коммуникационный порт по байту передает 32-разрядное информационное слово на шине CxD7-CxD0, стробируя передачу каждого байта сигналом CSTRBx#, сообщая тем самым, что байт выставлен на шине достоверно.

Этап 3. После получения байта данных принимающий коммуникационный порт генерирует сигнал CRDYx#, указывая на то, что байт данных «получен» и порт готов «принять» следующий байт.

Этап 4. После получения 4 байтов 32-разрядного слова ЦПУ или сопроцессор ПДП может начать считывание данных из входного буфера FIFO по адресу, расположенному в карте памяти коммуникационных портов, представленной на рисунке 12.3.

Каждый входной и выходной буфер FIFO может буферизовать максимум восемь 32-разрядных слов.

12.2.3 Карта памяти и регистры коммуникационных портов

На рисунке 12.3 приведена карта памяти коммуникационных портов ИС. На карте памяти представлены адреса регистров CPCRs, входных и выходных буферов FIFO коммуникационных портов. Назначение битов регистра CPCR представлено на рисунке 12.4.

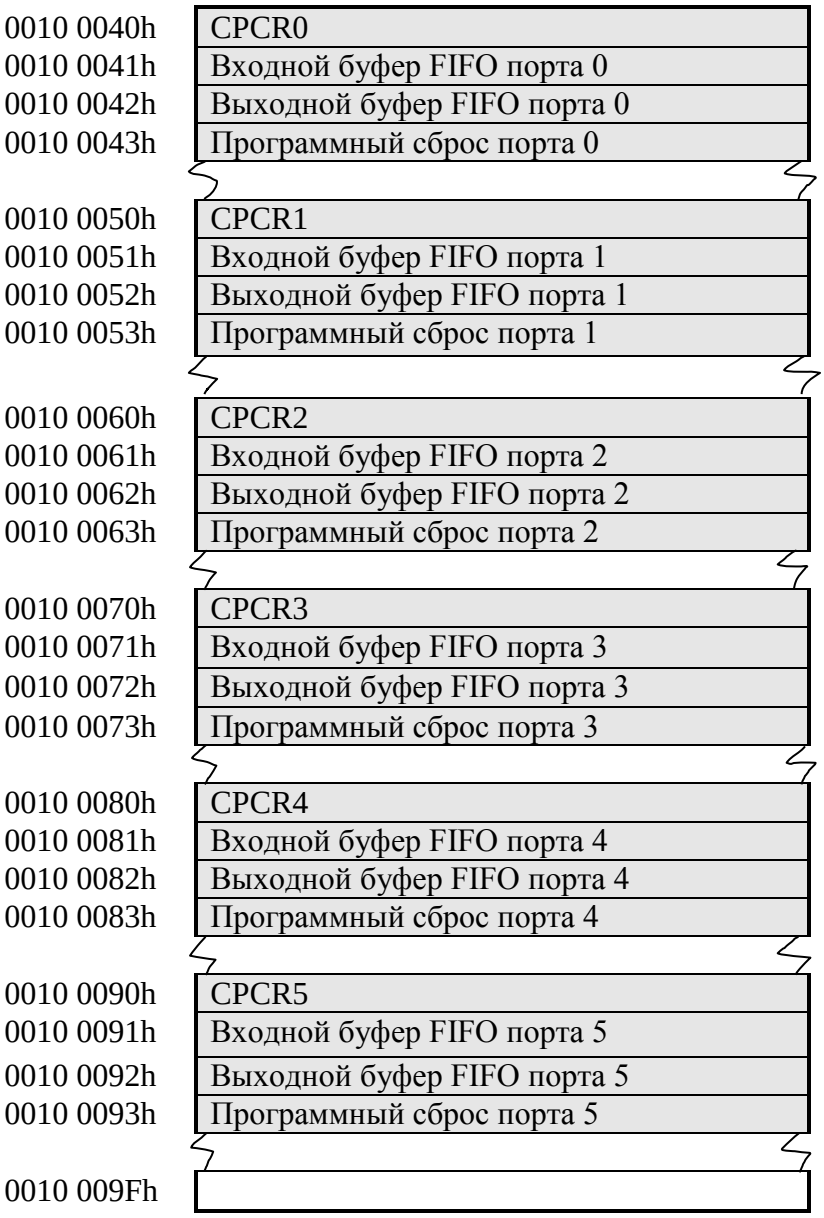


Рисунок 12.3 – Карта памяти коммуникационных портов

12.2.4 Регистр управления коммуникационного порта CPCR

На рисунке 12.4 показано расположение и мнемоника битов регистра управления коммуникационного порта CPCR.

Рисунок 12.4 – Регистр управления коммуникационного порта CPCR

Описание битов регистра управления CPCR

Бит PORT DIR – направление работы порта. Этот бит определяет направление операций передачи данных для коммуникационного порта.

Если PORT DIR = 0, то порт находится в режиме передачи (выходной канал).

Если PORT DIR = 1, то порт находится в режиме приема (входной канал).

Этот бит разрешён только для чтения. Изменить режим направления порта программным способом не возможно.

Бит ICH – удержание входного канала. При записи «1» в бит ICH входной канал порта останавливается. При записи «0» в бит ICH входной канал возобновляет свою работу.

Бит OCH – удержание выходного канала. При записи «1» в бит OCH выходной канал порта останавливается. При записи «0» в бит OCH выходной канал возобновляет свою работу.

Бит OUTPUT LEVEL – уровень выходного буфера FIFO.

Содержание этого 4-разрядного поля:

- Если значение равно 0000_2(0), то оно указывает на то, что выходной буфер FIFO пуст.

- Если значение равно от 0001_2(1) до 0111_2(7), то это указывает на количество записанных байт в выходном буфере FIFO.

- Если значение равно 1111_2(15), то это указывает на то, что выходной буфер FIFO заполнен.

Если выходной буфер пуст OUTPUT LEVEL = 0000_2, то срабатывает прерывание OCEMPTY = 1. Когда ЦПУ или сопроцессор ПДП записывает данные в пустой выходной буфер FIFO, то прерывание OCEMPTY очищается (OCEMPTY = 0) и находится в этом состоянии, пока буфер снова не очистится. Выходной буфер FIFO с одним или более свободными позициями также вырабатывает прерывание OCRDY = 1 на ЦПУ и сопроцессоре ПДП. По этому условию ЦПУ или сопроцессор ПДП могут записать данные в выходной буфер FIFO, см. подраздел 12.6 для более детальной информации.

Бит INPUT LEVEL – уровень входного буфера FIFO.

Содержание этого 4-разрядного поля:

- Если значение равно 0000_2(0), то это указывает на то, что входной буфер FIFO пуст.

- Если значение равно от 0001_2(1) до 0111_2(7), то это указывает количество заполненных позиций во входном буфере FIFO.

- Если значение равно 1111_2(15), то это указывает на то, что входной буфер FIFO заполнен.

Если входной буфер FIFO заполнен (INPUT LEVEL = 1111_2), то срабатывает прерывание ICFULL = 1. Когда ЦПУ или сопроцессор ПДП считывают все данные из входного буфера, то прерывание ICFULL очищается (ICFULL = 0) и находится в этом состоянии, пока буфер снова не заполнится. Входной буфер FIFO с одним или более свободными позициями также вырабатывает прерывание ICRDY = 1 на ЦПУ и сопроцессоре ПДП. По этому условию ЦПУ или сопроцессор ПДП могут считывать данные из входного буфера FIFO.

Бит зарезервированный – эти разряды не определены.

12.2.5 Регистр входного буфера FIFO

Этот регистр разрешён только для чтения и используется для чтения данных из входного буфера FIFO.

12.2.6 Регистр выходного буфера FIFO

Этот регистр используется для записи данных в выходной буфер FIFO и дальнейшей выдачи их на шину данных порта.

12.2.7 Регистр программного сброса коммуникационного порта

Заполненные позиции входного и выходного буферов FIFO коммуникационного порта могут быть очищены при записи данных по адресу регистра программного сброса коммуникационного порта. В таблице 12.1 указаны адреса регистров программного сброса коммуникационных портов. Программный сброс портов не влияет на состояние внешних выводов порта.

Таблица 12.1 – Адреса регистров программного сброса коммуникационных портов

Номер порта	Адрес регистра программного сброса порта
0	0100 043h
1	0100 053h
2	0100 063h
3	0100 073h
4	0100 083h
5	0100 093h

В примере 12.1 приведена программа, выполняющая программный сброс коммуникационного порта.

Пример 12.1

```
; ------------------------------------------------;
; RESET1: Сбрасывает данные коммуникационного порта     1;
; ------------------------------------------------;
RESET1 push   AR0          ; Сохранение регистров
       push   R0           ;
       push   RC           ;
       ldhi   010h,AR0     ;Запись в AR0 базовый адрес COM 1
       or     050h,AR0     ;
FLUSH: rpts   1            ;Сброс данных в FIFO
       sti    R0,*+AR0(3)  ;
       rpts   10           ;Ждать
       nop                 ;
       ldi    *+AR0(0),R0  ;Проверить наличие данных в других портах
       and    01FE0h,R0    ;
       bnz    FLUSH        ;
       pop    RC           ;Восстановление регистров
       pop    R0           ;
       pop    AR0          ;
       rets                ;Возврат
```

12.2.8 Арбитр коммуникационного порта

Арбитры определяют, какой коммуникационный порт имеет право на монопольное владение информационной шиной данных коммуникационного порта. Для своей работы арбитр использует сигналы CREQ# и CACK#, которые сообщают ему какому из коммуникационных портов передать право монопольного использования шины. Операция передачи права подробно описывается в подразделе 12.7.

После системного сброса половина портов ИС имеет право монопольного использования шины данных (коммуникационные порты 0, 1, 2), а другая половина (коммуникационные порты 3, 4, 5) не имеет.

Арбитр является синхронным конечным автоматом с четырьмя состояниями. Это показано в таблице 12.2. Изменение этих состояний не возможно программным способом.

Таблица 12.2 – Состояния конечного автомата арбитра

Номер состояния	Краткое описание	Состояние арбитра
Состояние «0» Ожидание с «жетоном»	1 Коммуникационный порт имеет «жетон» (PORT DIR = 0). 2 Канал не занят.	Коммуникационный порт в настоящее время имеет «жетон» и право на монопольное использование шины данных, но шина в данный момент не используется. При этом условии бит PORT DIR регистра CPCR равен нулю (выход). После системного сброса это состояние имеют коммуникационные порты с номерами 0, 1 и 2.
Состояние «1» Ожидание без «жетона»	1 Коммуникационный порт имеет «жетон» (PORT DIR = 1). 2 Арбитр не производит запрос указателя передачи.	Коммуникационный порт в настоящее время не имеет «жетон» и право на монопольное использование шины данных и не производит запрос указателя передачи. При этом условии бит PORT DIR регистра CPCR равен единице (вход). После системного сброса это состояние имеют коммуникационные порты с номерами 3, 4 и 5.
Состояние «2» Активизация	1 Коммуникационный порт имеет «жетон» (PORT DIR = 0). 2 Канал занят (OUTPUT LEVE $\neq$ 0).	Коммуникационный порт в настоящее время имеет «жетон» и производит использование шины данных. При этом условии, бит PORT DIR равен нулю (выход), а поле OUTPUT LEVEL не равно нулю.
Состояние «3» Ожидание «жетона»	1 Коммуникационный порт не имеет «жетон» (PORT DIR = 1). 2 Арбитр производит запрос «жетона» (OUTPUT LEVE $\neq$ 0).	Коммуникационный порт в настоящее время не имеет «жетона» и права на монопольное использование шины данных, но производит запрос «жетона». При этом условии бит PORT DIR регистра CPCR равен единице (вход), и поле OUTPUT LEVEL не равно нулю.

Рисунок 12.5 показывает диаграмму состояний и управление переходом из одного состояния в другое.

Чтобы передать данные по информационной шине коммуникационного порта, арбитр должен вынести решение между двумя типами запросов:

- внутренние запросы к выходным данным в выходном буфере FIFO, показанном как BUSRQ = 1 на рисунке 12.5;
- внешние запросы, полученные через сигнал CREQ#, показанный как TOKRQ = 1 на рисунке 12.5.

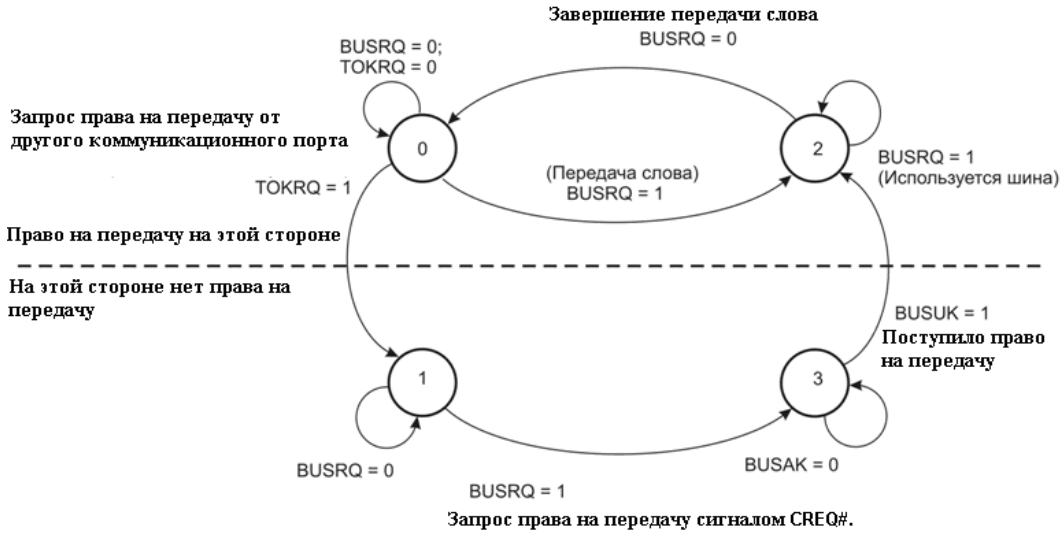


Рисунок 12.5 – Диаграмма состояний и управления переходами

Описание работы блока арбитра порта, диаграмма которого представлена на рисунке 12.5. Рассмотрим операцию передачи данных от ИС А к ИС В. Передача начинается с нулевого состояния арбитра ИС А (ожидание с «жетоном». Арбитр ИС В находится в состоянии 1 (ожидание без «жетона»).

Если арбитр А получает запрос на передачу данных из выходного буфера (BUSRQ = 1), то арбитр переходит в состояние 2 (активный). После того, как буфер вывода передает одно слово, арбитр удаляет запрос шины (BUSRQ = 0) и арбитр возвращается к состоянию «0» (ожидание с «жетоном»).

Если арбитр ИС В получает запрос на передачу данных из выходного буфера, то чтобы использовать шину данных арбитр ИС В должен подать сигнал CREQ# для запроса шины от арбитра ИС А. Арбитр ИС А обнаруживает этот запрос через переменную TOKRQ=1 и затем активирует сигнал CACKx#, чтобы передать «жетон» монопольного использования шины данных к арбитру ИС В. После этого арбитр ИС В генерирует внутренний сигнал подтверждения (BUSAK = 1), чтобы указать, что он получил монопольное использование шины. В результате операции передачи «жетона», арбитр ИС А входит в состояние «1» (простой без «жетона»), а арбитр ИС В переходит в состояние «2» (активный).

Чтобы препятствовать монополизации шины данных любому коммуникационному порту, арбитр всегда возвращается к состоянию «0» (ожидание с «жетоном») и производит запрос указателя передачи (активный CREQ#) от внешнего устройства после каждой передачи слова.

Если запрос «жетона» передачи активен, то «жетон» передачи переходит к «запрашивающей» ИС для передачи слова. Если ИС А и ИС В имеют информацию к

отправке в своих выходных буферах FIFO, то они чередуют использование информационной шины, чтобы обеспечить двунаправленный информационный канал.

Если запрос «жетона» получен в конце передачи слова и «передающая» ИС имеет другое слово для отправки в своем выходном буфере FIFO, то могут произойти две следующие ситуации:

- если сигнал CREQ# переходит в низкий уровень, перед тем как сигнал CRDY# перешёл в низкий уровень, для последнего байта слова, то ИС прекращает передачу данных и начинает передачу «жетона»;

- если сигнал CREQ# переходит в низкий уровень после или одновременно с переходом сигнала CRDY# в низкий уровень для последнего байта слова, то ИС, «имеющая жетон», продолжает передавать следующее слово и передает «жетон» только после окончания передачи всего слова.

Таким образом, ИС не будет передавать «жетон» до тех пор, пока не закончится передача четырех байтов.

12.3 Остановка работы буферов ввода и вывода порта

ИС может остановить входной буфер FIFO или выходной буфер FIFO или оба в период между передачей слов.

Чтобы остановить входной буфер FIFO, необходимо записать «1» в бит ICH = 1 регистра управления коммуникационного порта CPCR. Этот бит также может читаться, чтобы определить, что порт остановился или находится в состоянии готовности получить данные. Чтобы не останавливать работу входного буфера FIFO нужно записать «0» в бит ICH.

Чтобы остановить выходной буфер FIFO, необходимо записать «1» в бит OCH = 1 регистра управления коммуникационного порта CPCR. Этот бит также может читаться, чтобы решить, что порт останавливается или находится в состоянии готовности передать данные. Чтобы не останавливать работу выходного буфера FIFO, необходимо записать «0» в бит OCH. Операции останова/пуска буферов FIFO обсуждаются в следующих подразделах. Итоговые сведения представлены в таблице 12.3.

Таблица 12.3 – Операции останова/пуска буферов FIFO

Комбинации состояний	Если порт имеет «жетон»	Если порт не имеет указатель передачи
Входной остановлен	А. Не выполняет передачу указателя передачи.	А. Если останов происходит после принятия слова, то порт не выдает сигнал готовности на принятие нового слова. Если останов происходит во время принятия слова, то порт принимает одно слово и останавливается
Выходной не остановлен	Б. Передает данные.	Б. Если останов происходит после принятия первого байта слова, то порт принимает конец слова и останавливается.
Входной не остановлен	А. Не передает данные.	А. Принимает данные
Выходной остановлен	Б. Если останов происходит после передачи первого байта слова, то порт передает слово и останавливается.	Б. Не отвечает на запросы передачи «жетона»
	В. Передает «жетон».	

Окончание таблицы 12.3

Комбинации состояний	Если порт имеет «жетон»	Если порт не имеет указатель передачи
Входной остановлен Выходной остановлен	А. Не выполняет передачу «жетона». Б. Не передает данные.	А. Если останов происходит после принятия слова, то порт не выдает сигнал готовности на принятие нового слова. Если останов происходит во время принятия слова, то порт принимает одно слово и останавливается Б. Если останов происходит после принятия первого байта слова, то порт принимает слово и останавливается
	В. Если останов происходит после передачи первого байта слова, то порт передает конец слова и останавливается.	В. Не отвечает на запросы передачи указателя передачи.

12.3.1 Описание останова входного буфера FIFO

Цель останова входного буфера FIFO состоит в том, чтобы остановить работу входного буфера FIFO как можно скорее, не теряя входные данные.

Если коммуникационный порт, с входным буфером FIFO, который или останавливается или полон, не реагирует на низкий уровень сигнала CSTRB# низким уровнем сигнала CRDY# или подтверждает запрос «передачи права» низким уровнем сигнала CACK# на низкий уровень сигнала CREQ#, то это указывает на то, что порт находится в процессе передачи слова.

Логическая схема коммуникационного порта проверяет бит останова входного буфера FIFO в регистре CPCR только после окончательного получения слова.

Это подразумевает, что:

- Если коммуникационный порт получает бит останова входного буфера, когда не происходит прием слова, то входной буфер FIFO не останавливается немедленно, а ждет, чтобы получить одно слово и затем остановиться. Это пример останова входного буфера FIFO после сброса.

- Если коммуникационный порт получает бит останова входного буфера, когда происходит прием слова, то входной буфер FIFO не останавливается немедленно, а ждет, получения всего слова и затем остановится. Это состояние сохраняется до тех пор, пока не произойдёт сброс бита останова либо сброс порта. При сбросе бита останова прием данных продолжается без каких либо потерь.

- Если входной буфер FIFO коммуникационного порта останавливается в течение запроса «жетона» от другого коммуникационного порта, с которым он соединён, тогда запрос «жетона» подтверждается перед остановкой входного буфера FIFO.

12.3.2 Описание останова выходного буфера FIFO

Останов выходного буфера FIFO аналогичен останову входного буфера FIFO. Если выходной буфер остановлен, то возможны две ситуации, в зависимости от того, имеет или не имеет порт право передачи.

Если порт не имеет право передачи, то выходной буфер FIFO останавливается немедленно, и он не реагирует на запрос указателя передачи.

- Если коммуникационный порт, запрашивающий право передачи, останавливается после отправки низкого уровня сигнала CREQ#, то коммуникационный порт принимает указатель передачи и останавливается немедленно после этого.

Если порт имеет указатель передачи, то:

- Если в настоящее время порт передает слово, то только после передачи слова произойдёт останов выходного буфера и никакие новые передачи не произойдут.

- Если в настоящее время порт не передает слово, то происходит немедленный останов буфера и никакие новые передачи не произойдут.

- Если входной буфер FIFO не останавливается, а выходной буфер FIFO останавливается, то порт отвечает на запрос права передачи от другого порта.

- Если входной буфер FIFO останавливается и выходной буфер FIFO останавливается, тогда порт не отвечает на запрос указателя передачи от другого порта.

- Если коммуникационный порт имеет право передачи, а в выходном буфере имеются данные для передачи, то происходит очищение бита останова выходного буфера и порт возобновляет передачу данных.

12.4 Координация работы коммуникационных портов с ЦПУ и сопроцессором ПДП

Коммуникационные порты поддерживают два типа синхронизации по внутренним сигналам:

- Сигнал готовности готов/не готов, который может остановить работу ЦПУ или сопроцессора ПДП.

- Сигналы, внутренних системных прерываний, которые могут управлять работой ЦПУ или сопроцессора ПДП.

Самый простой вид синхронизации основан на сигнале готовности готов/не готов. Если сопроцессор ПДП или ЦПУ пытаются читать пустой входной буфер FIFO или записывать данные в полный выходной буфер FIFO, но сигнал готовности не возвращается в ЦПУ или ПДП, то они продолжают производить чтение или запись (происходит удержание периферийной шины данных) до тех пор, пока сигнал готовности не будет получен.

Сигнал готовности, для выходного канала, является OCRDY (выходной канал готов), этот сигнал также является сигналом прерывания. Сигнал готовности для входного канала является ICRDY (входной канал готов), который также является сигналом прерывания.

При синхронизации по сигналам прерываний каждый коммуникационный порт генерирует четыре сигнала прерывания, описание которых представлено ниже:

- ICFULL (полный входной канал) – указывает на то, что входной буфер FIFO имеет восемь слов;

- ICRDY (входной канал готов) – указывает на то, что, по крайней мере, одно слово находится во входном буфере FIFO;

- OCRDY (выходной канал готов) – указывает на то, что, по крайней мере, одно слово находится в выходном буфере FIFO;

- OCEMPTY (выходной канал пуст) – указывает на то, что выходной буфер FIFO пустой.

Центральный процессор может ответить на все четыре сигнала прерывания.

Сопроцессор прямого доступа в память может ответить только на сигналы прерывания ICRDY и OCRDY. Каждый канал сопроцессора ПДП может ответить

только на сигналы ICRDY и OCRDY, исходящие из одного из коммуникационных портов, от какого именно порта зависит от конфигурации.

Примечание – Ни один из четырех сигналов прерывания коммуникационного порта не имеет флаги в регистре IIF. Эти четыре сигнала состояния коммуникационного порта ICFULL, ICRDY, OCRDY, OCEMPTY могут быть получены проверкой состояний соответствующих битов в регистре управления коммуникационного порта CPCR. Например, чтобы определить есть ли прерывание от ICFULL, достаточно проверить 12 бит регистра CPCR.

Максимальная устойчивая скорость передачи данных для любого одиночного коммуникационного порта при входной тактовой частоте процессора, равной 160 МГц, равна 64 Мбайт/с. Тем не менее, когда несколько коммуникационных портов одновременно работают, то эта скорость может быть не достигнута. Сопроцессор ПДП может передавать данные память-память с максимальной скоростью 160 Мбайт/с (одна последовательность «чтение-запись» за два цикла). Сопроцессор ПДП может управлять двумя коммуникационными портами с максимальной скоростью. Для управления большим числом каналов сопроцессор ПДП становится узким местом независимо от того как он используется. ЦПУ может выполнять два чтения и две записи за два цикла, используя параллельные инструкции с максимальной скоростью 320 Мбайт/с. Для более пяти коммуникационных портов ЦПУ становится узким местом.

12.4.1 Операция передачи права монопольного владения шиной (передача «жетона»)

Операция передачи «жетона» требует процедуры «рукопожатия» с помощью сигналов через выводы CREQ# и CACK#. Это поясняется на рисунке 12.6. Для ясности, суффикс идентифицирует сигналы, выходящие из соответствующей ИС. Например, CREQb# обозначает сигнал CREQ#, выходящий из ИС. В таблице 12.4 приведен список событий, а рисунок 12.6 графически показывает процедуру установления связи.

Таблица 12.4 – Операция передачи указателя передачи

Номер события	Описание
0	Первоначально ИС А имеет право управления шиной и простаивает.
1	ИС В имеет данные для передачи ИС А и запрашивает право передачи, устанавливая сигнал CREQb# в низкий уровень.
2	После задержки ИС А обнаруживает запрос права передачи, когда сигнал CREQa# переходит в низкий уровень.
3	После задержки низкого фронта сигнала CREQa# ИС А подтверждает запрос сбросом сигнала CACKa# в низкий уровень.
4	После задержки ИС В получает подтверждение, когда сигнал CACKb# переходит в низкий уровень.
5	Сигнал CRDYa# переходит из высокоимпедансного состояния в высокий уровень после падения CACKa# в низкий уровень.
6	ИС А переводит шину CaD7-CaD0 в высокоимпедансное состояние после перевода сигнала CACKa# в низкий уровень.
7	ИС В переводит CSTRBb# из высокоимпедансного состояния в высокий уровень после падения CACKb# в низкий уровень.
8	ИС В переводит CREQb# в высокий уровень после задержки низкого фронта сигнала CACKb#.
9	После задержки ИС А получает высокий уровень сигнала CREQa#.
10	Сигнал CREQa# переходит из высокоимпеданого состояния в высокий уровень после получения высокого уровня на CREQa#.

Окончание таблицы 12.4

Номер события	Описание
11	ИС А переводит CACKa# в высокий уровень после того, как CREQa# приходит в высокий уровень.
12	ИС А переводит CACKa# в высокоимпедансное состояние после того, как сигнал CREQa# переходит в высокий уровень и после того, как сигнал CACKa# переходит в высокий уровень.
13	ИС А переводит сигнал CSTRBa# в высокоимпедансное состояние после того, как сигнал CREQa# переходит в высокий уровень.
14	ИС В переводит CREQb# в высокоимпедансное состояние после того, как сигнал CREQb# переходит в высокий уровень.
15	ИС В переводит сигнал CACKb# из высокоимпедансного состояния в высокий уровень после того как CREQb# переходит в высокий уровень.
16	ИС В переводит CRDYb# в высокоимпедансное состояние после того, как сигнал CREQb# переходит в высокий уровень.
17	ИС В переводит шину CDb из входного в выходное состояние после того как сигнал CREQb# переходит в высокий уровень.
18	ИС В управляет передачей первого байта на шине CbD7-CbD0 по срезу фронта H1 после того, как CREQb# переходит в высокий уровень.
19	ИС В переводит CSTRBb# в низкий уровень на втором фронте H1 после фронта CREQb#.
20	После задержки ИС А получает первый байт на CaD7-CaD0.
21	После задержки ИС А получает низкий уровень сигнала CSTRBa#, сообщающий о достоверности данных на шине.
22	ИС А читает данные и устанавливает сигнал CRDYa# низким уровнем.

12.5 Операция передачи слова

Коммуникационные порты ИС передают слова байтно, начиная с младшего байта. Операция передачи байта происходит с использованием сигналов CSTRB# и CRDY#. Пример передачи одного слова представлен на рисунке 12.7. Для ясности суффикс идентифицирует сигналы, выходящие из соответствующей ИС. Передача слова происходит следующим образом:

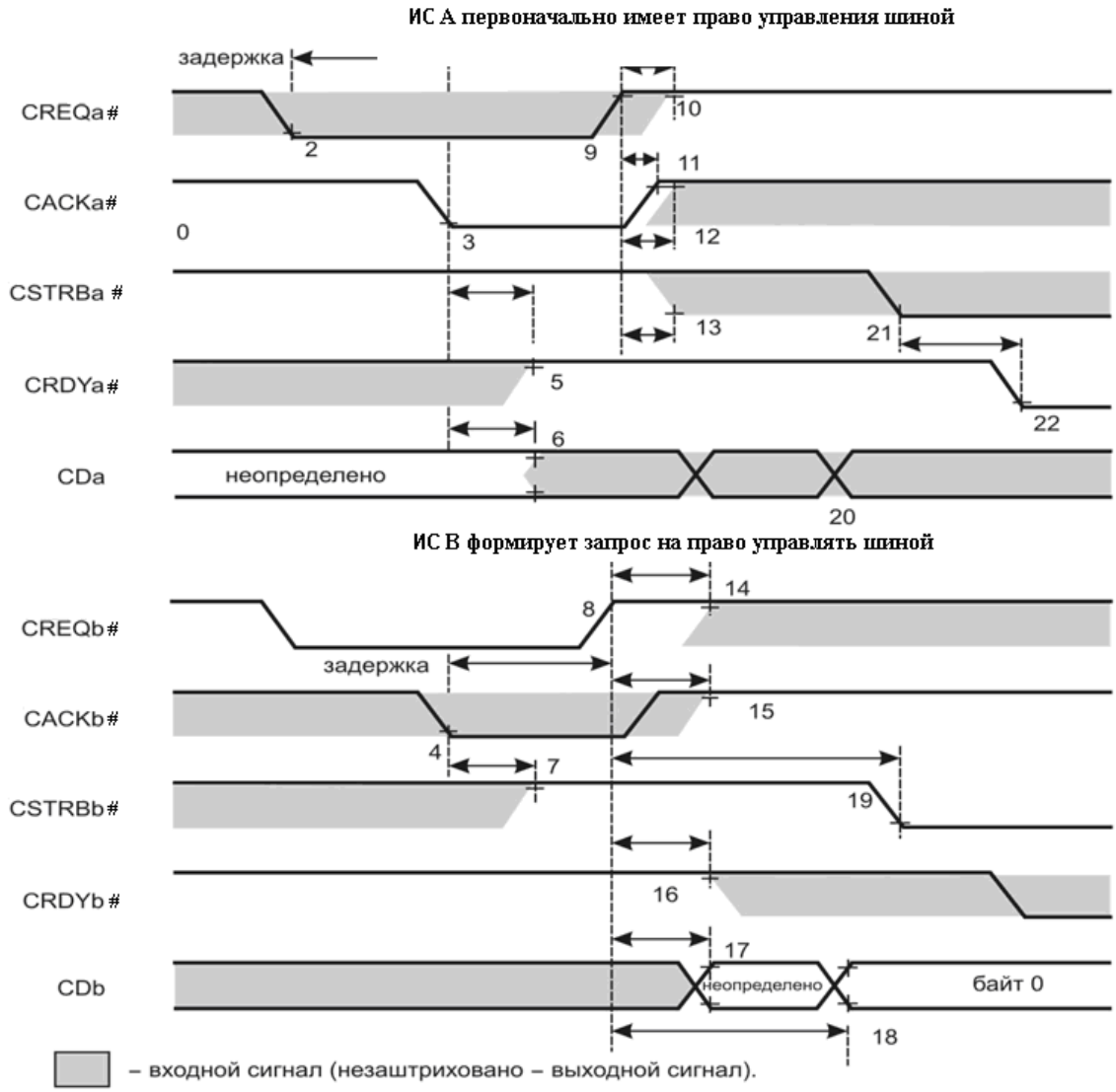
- ИС В выставляет младший байт на шине данных порта и через задержку выставляет сигнал CSTRBb# в низкий уровень, сообщая этим, что на шине выставлены достоверные данные;

- ИС А принимает низкий уровень сигнала CSTRBa# и через задержку выставляет сигнал CRDYa# в низкий уровень и принимает данные.

После принятия ИС В сигнала CRDYb# низким уровнем, он через задержку восстанавливает CSTRBb# в высокий уровень.

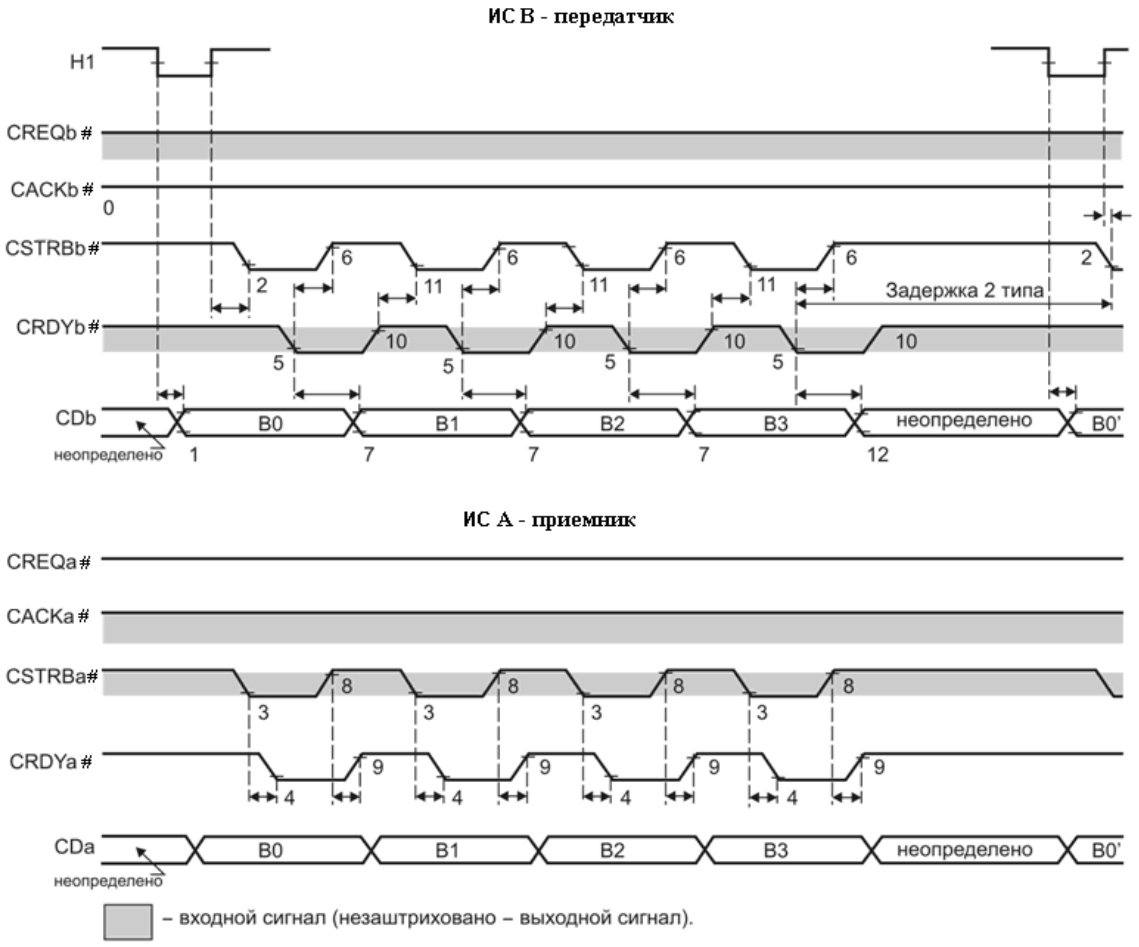
После принятия данных ИС А восстанавливает CRDYa# в высокий уровень.

После принятия процессором ИС В сигнала CRDYb# высоким уровнем, он начинает передачу следующего байта.



Примечание – Пояснение задержки см. в подразделе 12.6 «Синхронизация».

Рисунок 12.6 – Операция передачи права владения шиной



Примечание – B0' – 0 байт нового слова.

Рисунок 12.7 – Операция передачи слова

12.6 Синхронизация

В течение времени на границе передачи слов и в течение передачи «жетона» требуется синхронизация H1/H3. Арбитр порта включает три типа синхронизации:

- Синхронизация первого типа обозначает задержки, которые изменяются от 1 до 2 машинных циклов от получения сигнала на входе до ответа на выходе (игнорируя аналоговые задержки). Входной сигнал распознается, когда H1 находится в высоком уровне, а затем он проходит через серию задержек H3/H1 высокого уровня. Ответ происходит при запуске следующего H3 в высоком уровне. Минимальная задержка синхронизации первого типа (1 машинный цикл) может происходить только тогда, когда входной сигнал изменяется перед тем, как H1 перейдет в низкий уровень. Задержка показана на рисунке 12.8.

Максимальная задержка синхронизации первого типа (2 машинных цикла) может происходить только тогда, когда входной сигнал изменяется после того, как сигнал H1 перейдет в низкий уровень. Задержка показана на рисунке 12.9.

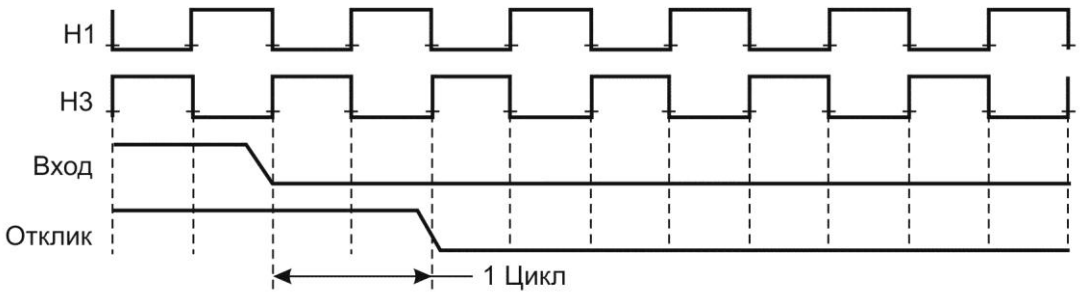


Рисунок 12.8 – Минимальная задержка синхронизации первого типа

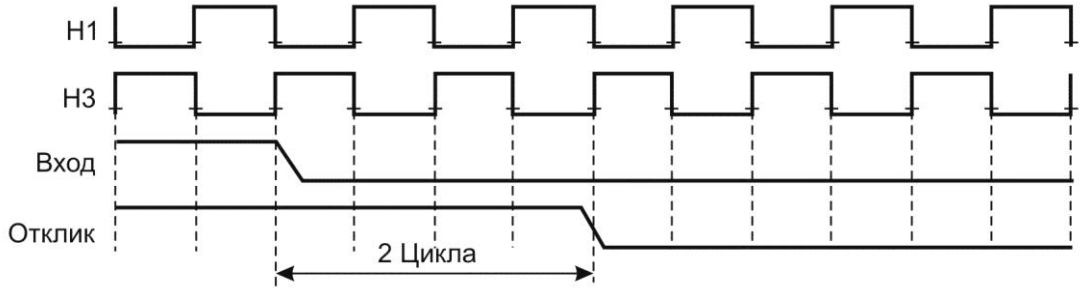


Рисунок 12.9 – Максимальная задержка синхронизации первого типа

- Синхронизация второго типа определяет задержки, которые изменяются от 1,5 до 2,5 машинных циклов от получения сигнала на входе до ответа на выходе (игнорируя аналоговые задержки). Входной сигнал распознается, когда сигнал H1 находится в высоком уровне, а затем он проходит через серию задержек H3/H1/H3 высокого уровня. Ответ происходит при переходе следующего H1 в высокий уровень. Минимальная задержка синхронизации второго типа (1,5 машинного цикла) может происходить только тогда, когда входной сигнал изменяется перед тем, как H1 перейдет в низкий уровень. Задержка показана на рисунке 12.10. Максимальная задержка синхронизации второго типа (2,5 машинных цикла) может происходить только тогда, когда входной сигнал изменяется после того, как сигнал H1 перейдет в низкий уровень.

Задержка показана на рисунке 12.11.

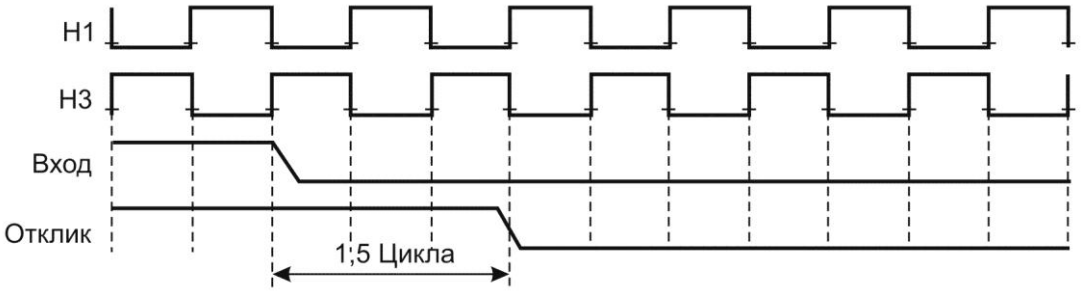


Рисунок 12.10 – Минимальная задержка синхронизации второго типа

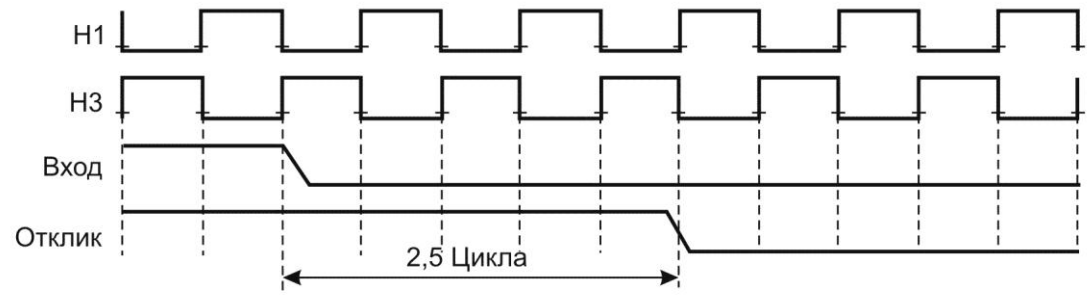


Рисунок 12.11 – Максимальная задержка синхронизации второго типа

- Синхронизация третьего типа определяет задержки, которые изменяются от 0,5 до 1,5 машинного цикла от получения сигнала на входе до ответа на выходе (игнорируя аналоговые задержки). Входной сигнал определяется, когда сигнал H1 находится в высоком уровне, а затем он проходит через серию задержек H3 высокого уровня. Ответ происходит при переходе следующего H1 в высокий уровень. Минимальная задержка синхронизации третьего типа (0,5 машинного цикла) может происходить только тогда, когда входной сигнал изменяется перед тем, как сигнал H1 перейдет в низкий уровень. Задержка показана на рисунке 12.12. Максимальная задержка синхронизации третьего типа (1,5 машинного цикла) может происходить только тогда, когда входной сигнал изменяется после того, как сигнал H1 перейдет в низкий уровень. Задержка показана на рисунке 12.13.

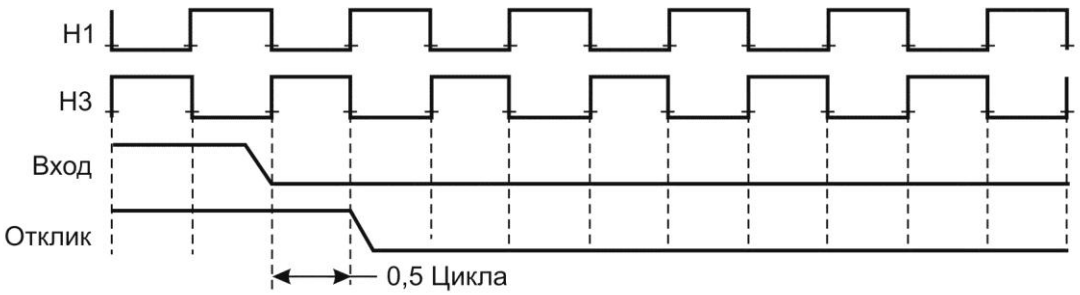


Рисунок 12.12 – Минимальная задержка синхронизации третьего типа

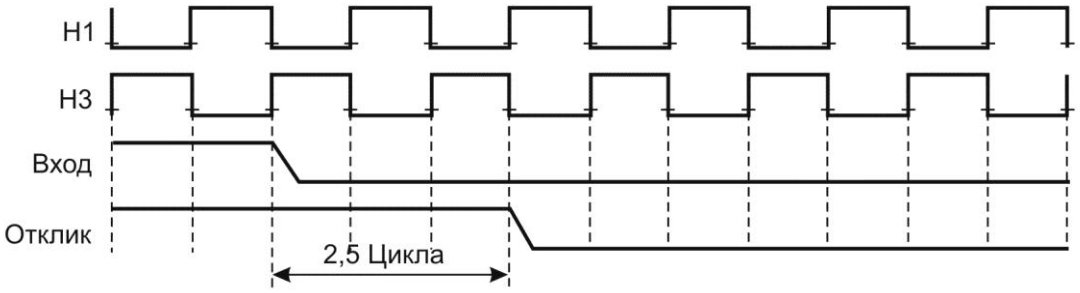


Рисунок 12.13 – Максимальная задержка синхронизации третьего типа

В таблице 12.6 приведены типы задержек синхронизации для сигналов коммуникационного порта.

Таблица 12.6 – Сигналы коммуникационного порта и задержки синхронизации

Входной (↓) - выходной (↑) сигнал	Тип задержки	Минимальная задержка, машинные циклы	Максимальная задержка, машинные циклы
CREQ#↓ - CACK#↓	1	1	2
CACK#↓ - CREQ#↑	1	1	2
CRDY#↓ - CD верны между передачей слов от конца до конца передачи	1	1	2
CRDY#↓ - CSTRB#↓ между передачей слов от конца до конца передачи	2	1,5	2,5
CACK#↓ - CSTRB# переключается от входного к выходному сигналу высокого уровня	3	0,5	1,5

12.6.1 Сброс коммуникационного модуля

В этом разделе описывается состояние коммуникационных портов ИС 1867ВМ9Ф во время и после подачи питания, а также после системного сброса.

После включения питания состояние модуля зависит от сигнала RESET#:

- если RESET# в низком уровне, то ИС сбрасывается немедленно, при этом применимо описание «при сбросе» (смотри ниже);
- если RESET# не в низком уровне, то ИС находится в неопределенном состоянии.

Сигналы коммуникационного порта могут иметь комбинацию состояний. При сбросе (пока RESET#=0), все выводы коммуникационного порта устанавливаются в состояние высокого импеданса. Входные и выходные каналы предполагаются пустыми, так как все значения во входных и выходных буферах теряются. Дополнительные резисторы должны использоваться на всех управляющих сигналах для гарантии того, что они находятся в высоком уровне, если сброс одновременно не выполняется для всех процессоров многопроцессорной системы с использованием ИС 1867ВМ9Ф.

После сброса по фронту RESET# коммуникационные порты с номерами 0, 1, 2 конфигурируются как выходные и имеют следующие состояния:

- Арбитр сбрасывается в «0» – арбитр имеет право передачи и находится в состоянии ожидания.

Состояния выводов, смотри рисунок 12.14, устанавливаются как:

- сигналы CxD7-CxD0 в неопределенное значение;
- сигналы CACK# и CSTRB# в «1» (пассивны);
- сигналы CREQ# и CRDY# остаются в состоянии высокого импеданса.

Примечание – Программный сброс одного коммуникационного порта сбрасывает только буферы FIFO, но не оказывает влияния на внешние выводы.

- Регистр управления коммуникационным портом сбрасывается в «0»:
- PORT DIR = 0 – коммуникационный порт конфигурируется для операций выхода,
- INPUT LEVEL = 0 – входной буфер FIFO пустой;
- OUTPUT LEVEL = 0 – выходной буфер FIFO пустой;
- ICH = 0 – входной буфер FIFO не остановлен;
- OCH = 0 – выходной буфер FIFO не остановлен;
- ICRDY = 0 – входной буфер FIFO пустой и не готов для чтения из него;
- OCRDY = 0 – выходной буфер FIFO не полный и готов для записи в него.

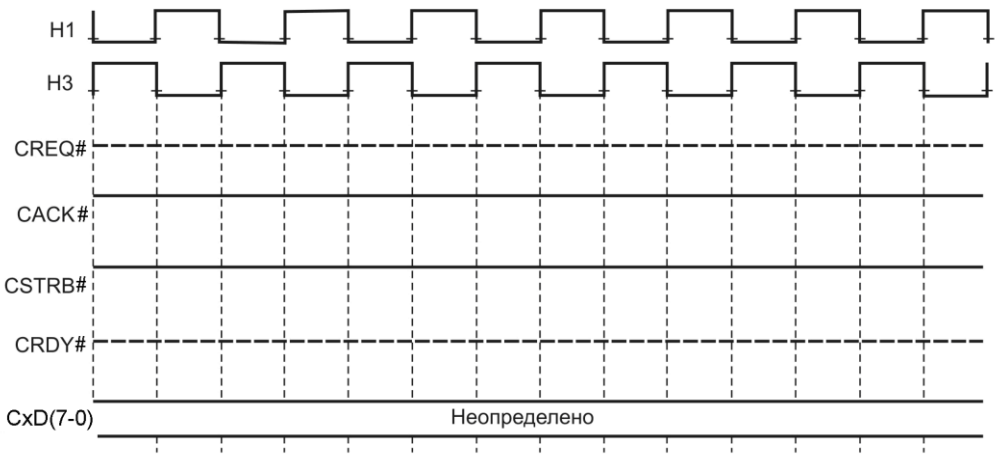


Рисунок 12.14 – Состояние сигналов коммуникационного порта после сброса выходного порта

После сброса (по фронту RESET#) коммуникационные порты с номерами 3, 4, 5 конфигурируются как входные и имеют следующие состояния:

- Арбитр устанавливается в состояние «1», т. е. арбитр не имеет право передачи и не находится в состоянии ожидания.

Состояния выводов, см. рисунок 12.15, устанавливаются как:

- сигналы CxD7-CxD0 остаются в состоянии высокого импеданса;
- сигналы CREQ# и CRDY# в 1 (пассивны);
- сигналы CACK# и CSTRB# остаются в состоянии высокого импеданса.

Примечание – Программный сброс одного коммуникационного порта сбрасывает только буферы FIFO, но не оказывает влияния на внешние выводы.

- Регистр управления коммуникационным портом устанавливается в значение 04h:
- PORT DIR = 1 – коммуникационный порт конфигурируется для операций входа;
- INPUT LEVEL = 0 – входной буфер FIFO пустой;
- OUTPUT LEVEL = 0 – выходной буфер FIFO пустой;
- ICH = 0 – входной буфер FIFO не остановлен;
- OCH = 0 – выходной буфер FIFO не остановлен;
- ICRDY = 0 – входной буфер FIFO пустой и не готов для чтения из него;
- OCRDY = 0 – выходной буфер FIFO не полный и готов для записи в него.

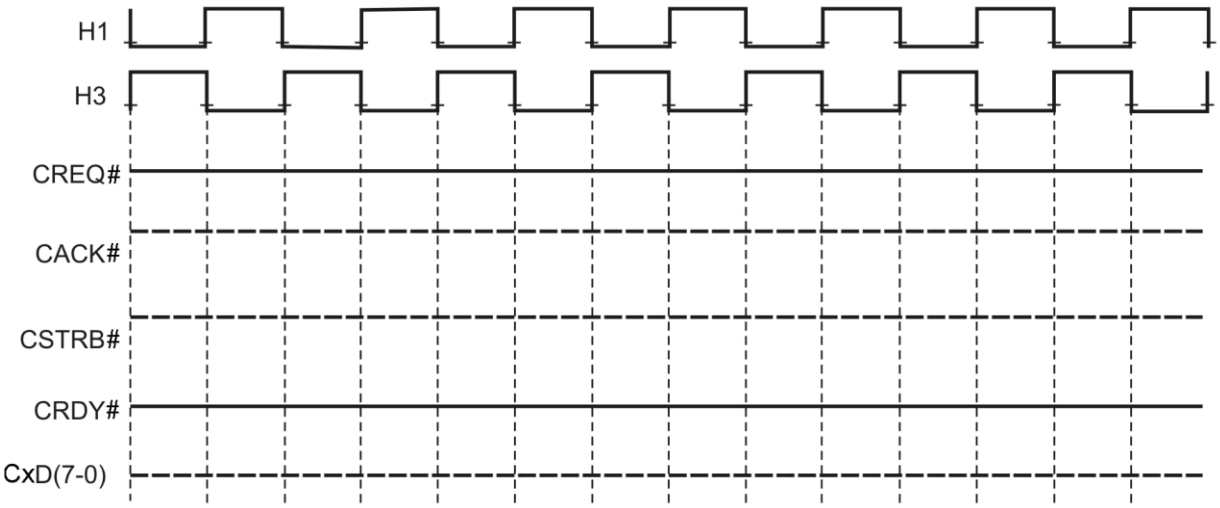


Рисунок 12.15 – Состояние после сброса входного порта

12.6.2 Рекомендации по использованию портов

При сбросе порты с номерами 0, 1, 2 конфигурируются как выходные (PORT DIR = 0), а порты с номерами 3, 4, 5 – как входные (PORT DIR = 1). При соединении двух ИС 1867ВМ9Ф порты соединяются в противоположных друг другу направлениях передачи данных, то есть любой порт с номером 0, 1, 2 соединяется с любым портом с номером 3, 4, 5.

Качество сигналов коммуникационных портов очень важно. Необходимо максимально уменьшить шумы на плате для надежной работы коммуникационных портов.

Не нужно читать из пустого входного FIFO. Это может привести к останову ЦПУ или сопроцессора ПДП и периферийной шины.

Не нужно писать в не подсоединенный коммуникационный порт. Если выходное FIFO полное и будет записан очередной байт, то это может привести к останову периферийной шины.

Тактирующие сигналы двух соединенных ИС должны быть в пределах соотношения 2 : 1. Если не следовать этой рекомендации, то это может привести к столкновению активных уровней сигналов и тем самым вызвать повреждение линий связи. Это ограничение не действует на другие устройства, подключаемые к коммуникационному порту.

13 Таймеры

13.1 Обзор таймеров

ИС 1867ВМ9Ф содержит два таймера общего назначения, которые могут отсчитывать время, генерировать импульсы и прерывать ЦПУ и сопроцессор ПДП.

Это 32-разрядные счетчики с двумя режимами тактирования – внутренним или внешним тактированием, см. рисунок 13.1. Модули таймера могут быть использованы для формирования периодических сигналов, поступающих внутрь ИС или на внешние устройства.

Доступный для каждого таймера вывод входа/выхода может быть использован как тактовый вход таймера, как выходной сигнал синхронизации или вывод входа/выхода общего назначения.

С внутренним тактированием таймер может быть использован в качестве указателя внешнему ЦАП/АЦП начать преобразование или прервать сопроцессор ПДП для начала пересылки данных.

Каждый таймер состоит из 32-разрядного счетчика, компаратора, селектора входных тактов, импульсного генератора и дополнительной аппаратной части.

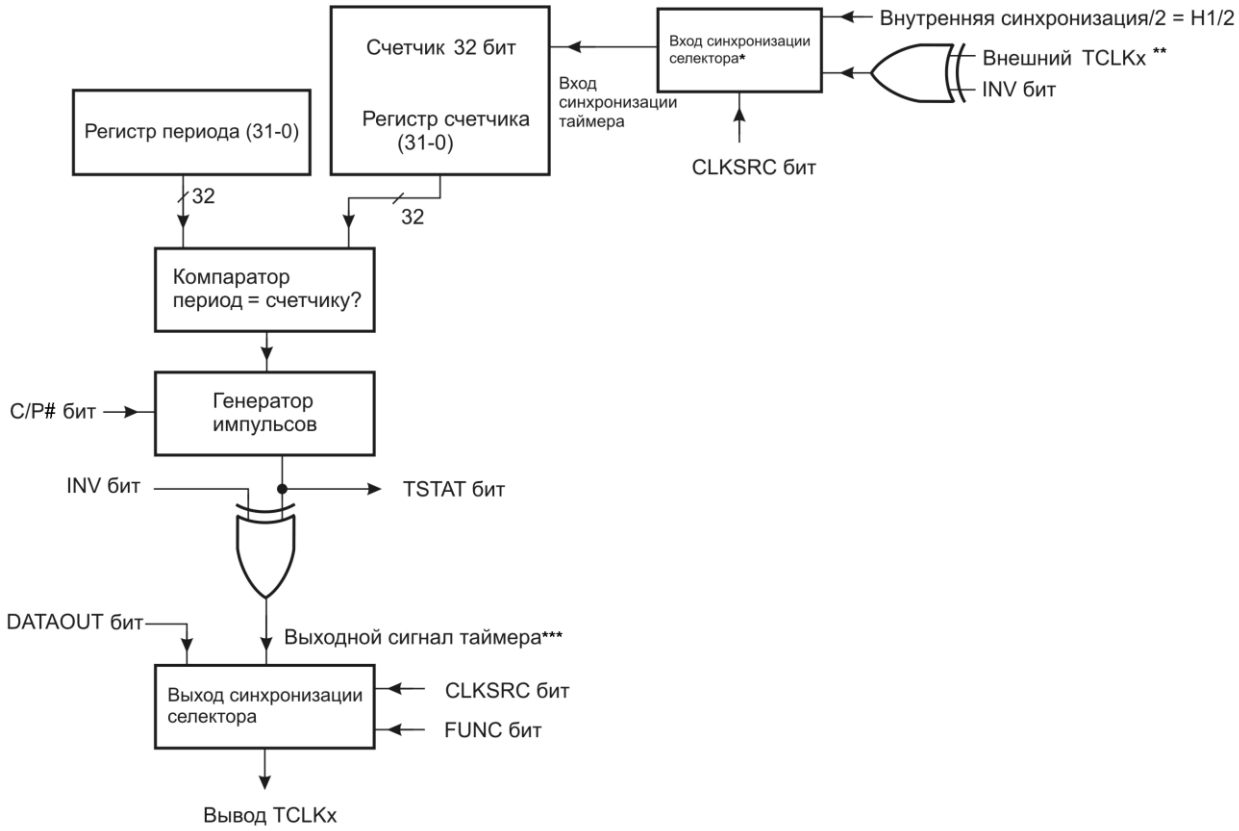
В режиме отсчета тактов входного сигнала, если счетчик таймера отсчитал количество тактов, которое равно значению, записанному в регистре периода, счетчик сбрасывается в «0» и генерирует выходной сигнал.

Входной тактовый сигнал таймера может быть равен половине внутренней тактовой частоты ИС или частоте на входе TCLKx. Это определяется битом CLKSRC в регистре управления таймером. Если используется внешнее тактирование, то таймер может отсчитывать сигналы при переходе входного сигнала как из «0» в «1» (переход сигнала из низкого в высокий уровень или передний фронт), так и от «1» к «0» в зависимости от значения бита INV.

Выходной сигнал таймера зависит от режима тактирования, выбираемого битом C/P# (режим тактов или импульсов), см. подраздел 13.1.4.

Выход таймера может быть направлен через выводы TCLKx, которые, в свою очередь, могут использоваться как входы/выходы общего назначения.

На рисунке 13.1 изображена блок-схема модуля таймера ИС 1867ВМ9Ф.



x = 0, 1.

_____

* Селектор управляется разрядом CLKSRC.

** Максимальная частота = f(H1)/2,6.

*** Если CLKSRC = 1 и FUNC = 1, этот сигнал передается на TCLKx, где x = 0, 1.

Рисунок 13.1 – Блок-схема модуля таймера ИС 1867ВМ9Ф

13.1.1 Выводы таймеров

Каждый таймер имеет один вывод, связанный с выводом тактового сигнала таймера TCLK. Вывод TCLK может использоваться как вход/выход общего назначения, как выход таймера или как вход внешней синхронизации для таймера. Каждый таймер имеет свой вывод TCLK: TCLK0 относится к таймеру «0» и TCLK1 относится к таймеру «1».

13.1.2 Регистры таймера

Каждый таймер содержит три регистра:

- регистр управления. Этот регистр определяет режим работы таймера, отражает его состояние и управляет функциональностью выводов входа/выхода TCLK;

- регистр периода. Этот регистр определяет частоту формирования выходного сигнала таймером;

- регистр-счетчик. Этот регистр содержит текущее значение инкрементируемого счетчика.

На рисунке 13.2 приведена карта памяти этих регистров.

Регистр	Адрес периферии	
	Таймер 0	Таймер 1
Регистр управления таймером	10 0020h	10 0030h
Зарезервировано	10 0021h	10 0031h
Зарезервировано	10 0022h	10 0032h
Зарезервировано	10 0023h	10 0033h
Регистр-счетчик таймера	10 0024h	10 0034h
Зарезервировано	10 0025h	100035h
Зарезервировано	10 0026h	10 0036h
Зарезервировано	10 0027h	10 0037h
Регистр периода таймера	100028h	10 0038h
Зарезервировано	10 0029h	10 0039h
Зарезервировано	10 002Ah	10 003Ah
Зарезервировано	10002Bh	10 003Bh
Зарезервировано	10 002Ch	10 003Ch
Зарезервировано	10 002Dh	10 003Dh
Зарезервировано	10 002Eh	10 003Eh
Зарезервировано	10 002Fh	10 003Fh

Рисунок 13.2 – Адреса регистров таймеров 0 и 1

13.1.2.1 Регистр управления таймером

Регистр управления таймером расположен по адресу 10 0020h для таймера «0» и 10 0030h для таймера «1».

Регистр управления таймером – это 32-разрядный регистр, содержащий биты общего управления и управления входом/выходом общего назначения для модуля таймера. На рисунке 13.3 показаны биты регистра, их имена и функции. Разряды 3-0 – это биты управления входом/выходом общего назначения. Разряды 11-6 – это биты общего управления таймером. Необходимо обратить внимание, что при сбросе все разряды устанавливаются в «0», за исключением бита DATIN, который устанавливается в соответствии со значением на TCLK.

Рисунок 13.3 – Регистр управления таймером

Описание битов регистра управления таймером:

- FUNC – бит функции. Бит FUNC управляет функцией вывода TCLK. Если FUNC=0, то TCLK конфигурируется как цифровой вход/выход общего назначения. Если FUNC=1, то TCLK конфигурируется как сигнал таймера.

- I/O – бит вход/выход. Если I/O=1 и FUNC=0, то TCLK конфигурируется как вход. Если I/O=0 и FUNC=0, то TCLK конфигурируется как выход.

- DATAOUT – бит выхода данных. Бит DATAOUT управляет TCLK, когда таймер ИС находится в режиме ввода-вывода общего назначения, то DATAOUT также может использоваться как вход счетчика таймера.

- DATAIN – бит входа данных. Отражает состояние данных на TCLK или DATOUT. Запись в этот бит ни на что не влияет.

- GO – бит начала работы. Бит GO сбрасывает и запускает счетчик таймера. Когда GO=1 и таймер не удерживается, то счетчик обнуляется и начинает инкрементироваться по следующему возрастающему фронту тактового входа таймера. Бит GO очищается по тому же возрастающему фронту. Запись «0» в бит GO не влияет на таймер.

- HLD – бит удержания счетчика. Когда бит HLD равен нулю, то счет запрещен и счетчик удерживается в его текущем состоянии. Когда таймер управляется TCLK, то состояние TCLK тоже удерживается. Внутренний счетчик (с делением на два) тоже удерживается таким образом, что счетчик может продолжиться с состояния останова при установке HLD в «1». Регистры таймера могут считываться и модифицироваться, пока счетчик удерживается. Сигнал RESET имеет более высокий приоритет, чем HLD. Далее приведено описание различных состояний битов GO и HLD.

GO	HLD	Результат
0	0	Останов всех операций таймера. Сброс не происходит (значение сброса).
0	1	Таймер обрабатывается из состояния, предшествующего записи.
1	0	Все операции таймера остановлены, включая обнуление счетчика. Бит GO не очищается до того, как таймер выйдет из состояния удержания.
1	1	Таймер сбрасывается и запускается.

- C/P# – бит управления режимом такт/импульс. Когда C/P# =1, то выбирается режим такт, что указывает на то, что флаг состояния и внешний выход будут иметь цикл со скважностью 50 %. Когда C/P# =0, то флаг состояния и внешний выход будут активными для одного цикла H1 в течение каждого периода таймера, см. рисунок 13.4.

- CLKSRC – бит определения источника тактирования таймера. Когда CLKSRC=1, то для инкрементирования счетчика будет использоваться внутренний такт с частотой в половину частоты H1. Бит INV не влияет на внутренний источник тактирования. Когда CLKSRC=0, то для инкрементирования счетчика используется внешний сигнал с вывода TCLK. Внешний такт синхронизирован внутри, обеспечивая, таким образом, возможность работы с внешними асинхронными источниками тактирования, которые не превышают максимально поддерживаемой частоты внешнего такта f(H1)/2,6.

- INV – бит управления инверсией. Если использован внешний источник тактирования и INV=1, то внешний такт инвертируется при входе в счетчик. Если выход внешнего генератора импульсов подключен к TCLK и INV=1, то выходной сигнал инвертируется. Если INV=0, то инверсии сигнала не происходит ни на входе, ни на выходе таймера. При использовании TCLK как входа/вывода порта состояние INV, вне зависимости от его значения, не влияет на функционирование.

- TSTAT – бит указывает на состояние таймера. Он отслеживает выход не инвертированного вывода TCLK. Этот флаг выставляет прерывание ЦПУ при переходе из «0» в «1». Запись в этот бит ни на что не влияет.

13.1.2.2 Регистр периода таймера

Регистр периода таймера размещен по адресу 10 0028h для таймера «0» и – 10 0038h для таймера «1».

32-разрядный регистр периода таймера используется для определения частоты формирования сигнала таймером.

Частота сигнала таймера определяется частотой входной синхронизации таймера и регистром периода. Следующие уравнения верны для внутренней и внешней синхронизации:

- f(режим импульса) = f(синхронизация таймера)/(содержимое регистра периода).
- f(режим такта) = f(синхронизация таймера)/(2x содержимое регистра периода).

При сбросе этот регистр сбрасывается в «0».

13.1.2.3 Регистр-счетчик таймера

Регистр-счетчик размещается по адресу 10 0024h для таймера «0» и – 10 0034h для таймера «1».

32-разрядный регистр-счетчик инкрементируется с каждым тактом входной синхронизации таймера. Счетчик таймера может быть инкрементирован по переднему фронту (INV=0) или по заднему фронту (INV=1) внешней входной синхронизации (CLKSRC=0).

Для внутренней синхронизации (CLKSRC=1) счетчик таймера инкрементируется только по переднему фронту. Счетчик обнуляется всякий раз, когда достигает значения регистра периода. При сбросе этот регистр сбрасывается в «0».

13.1.3 Граничные условия в регистрах управления

Определенные условия, такие как ноль, в регистре периода и переполнение счетчика, влияют на работу таймера. Эти условия перечислены ниже:

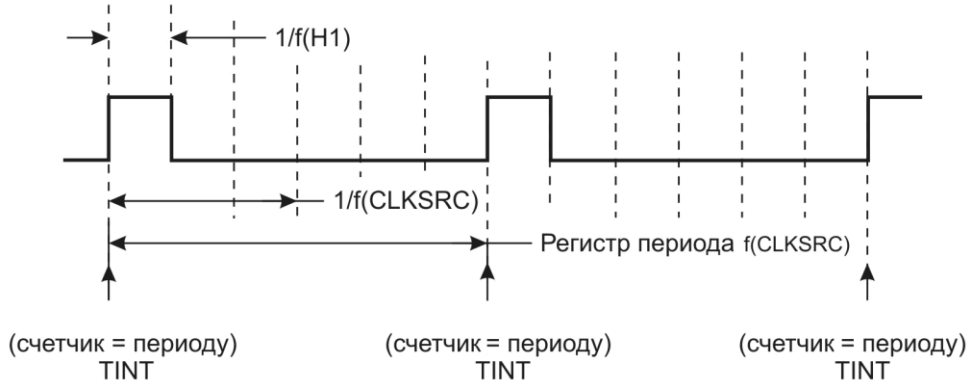
- Когда регистры периода и счетчика установлены в «0», то работа таймера зависит от режима, выбранного битом C/P#. При выборе импульсного режима (C/P# = 0) вывод TSTAT устанавливается и остается в этом состоянии. При тактовом режиме (C/P# = 1) длительность цикла составляет 2/f(H1) и внешние такты игнорируются.
- Когда регистр счетчика не равен «0», а регистр периода =0, то счетчик будет считать до «0» и далее будет вести себя так же, как описано выше.
- Когда регистр счетчика установился в значение, большее значения регистра периода, то счетчик может переполниться при инкрементировании. Когда счетчик достигнет максимального 32-разрядного значения (0FFF FFFFh), то он просто перейдет через «0» и продолжит счет.

Примечание – Запись в счетчик с периферийной шины имеет приоритет перед инкрементированием регистра счетчика, она модифицирует его и изменит значение регистра управления.

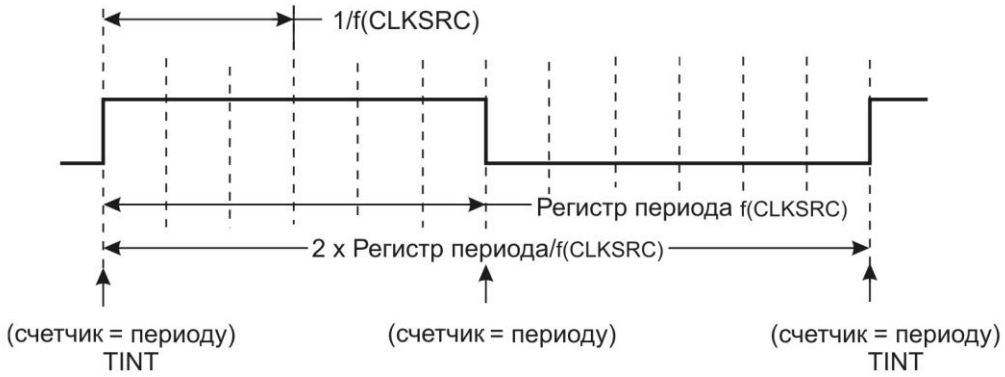
13.1.4 Генерация импульсов таймером

Генератор импульсов таймера, см. рисунок 13.1, может генерировать несколько различных внешних сигналов. Эти сигналы могут быть инвертированы битом INV. На рисунке 13.4 приведены два основных режима – импульсный и тактовый. Источник внутреннего такта f(такт таймера) в обоих случаях имеет частоты f(H1)/2, при этом генерируемый извне источник такта f(такт таймера) может иметь макси-

мальную частоту f(H1)/2,6. В импульсном режиме (C/P# = 0) ширина импульса составляет 1/f(H1). В режиме такта (C/P# = 1) ширина импульса равна значению регистра периода, деленному на частоту входной синхронизации.



а) TSTAT и выход таймера INV = 0, когда C/P# = 0 (импульсный режим).



б) TSTAT и выход таймера INV = 0, когда C/P# = 1 (тактовый режим).

Рисунок 13.4 – Временные диаграммы импульсного и тактового режимов таймера

Период выдачи сигнала таймером определяется входной частотой тактирования таймера и регистром периода. Следующие выражения определены либо для внутреннего, либо для внешнего тактирования таймера:

f(импульсный режим) = f(такта таймера) регистр периода.

f(тактовый режим) = f(такта таймера)/(2 × регистр периода).

Если регистр периода равен «0», то см. 13.1.2.

На рисунке 13.5 приведено несколько примеров выхода TCLK при установке различных значений регистра периода и выборе импульсного или тактового режима, зависимость тактирования таймера от внутренней частоты f(H1), f(H2).

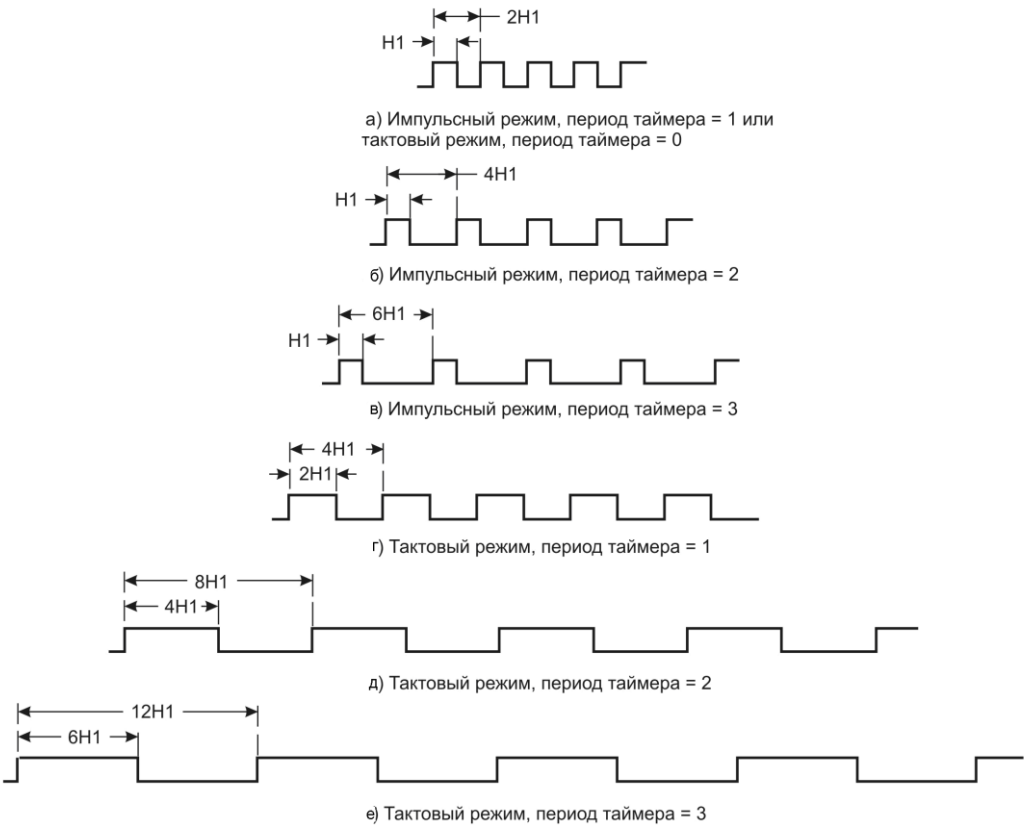


Рисунок 13.5 – Примеры генерации выхода таймера

13.1.5 Прерывания от таймера

Каждый таймер может послать сигнал прерывания ЦПУ, когда сигнал TSTAT изменяется из «0» в «1». Таймер 0 посылает сигнал TINT0, а таймер 1 посылает сигнал TINT1.

Прерывание по TINT0 использует вектор прерывания по адресу IVTP+002h. Оно имеет приоритет второго уровня, т. е. после NMI и RESET.

Прерывание по TINT1 использует вектор прерывания по адресу IVTP+02Bh. Оно имеет низший приоритет из всех прерываний.

13.1.6 Прерывания от таймеров и их векторы прерываний

Прерывание TINT0 относится к таймеру 0. Это прерывание использует вектор прерывания по адресу IVTP+002h. Оно имеет приоритет второго уровня, т.е. после прерываний от NMI и RESET.

Прерывание TINT1 относится к таймеру 1. Это прерывание использует вектор прерывания по адресу IVTP+02Bh. Оно имеет самый низкий приоритет из всех прерываний.

13.1.7 Операция прерывания от таймера

Прерывание от таймера возникает, когда TSTAT изменяется из «0» в «1». Частота прерываний таймера зависит от установленного режима (режим такта или режим импульса).

В режиме импульса частота прерывания равна

f(прерывания) = f(такта таймера)/регистр периода.

В режиме такта частота прерывания равна

f(прерывания) = f(такта таймера)/(2 × регистр периода).

Если регистр периода равен «0», то смотри 13.1.4.

Прерывания от таймера могут использоваться для прерываний ЦПУ или сопроцессора ПДП.

Биты разрешения прерываний от таймера для ЦПУ находятся в регистре IIE. Разряд 0 в IIE относится к TINT0, а разряд 1 относится к TINT1. Для более подробной информации о IIE регистре смотри 3.1.9 «Регистр разрешения внутренних прерываний ЦПУ IIE».

Биты разрешения прерываний от таймера для сопроцессора ПДП находятся в регистре DIE. Некоторые биты в этом регистре управляют ответами канала ПДП на сигналы от таймеров. Для более подробной информации о регистре DIE смотри 3.1.8 «Регистр разрешения прерывания DIE сопроцессора ПДП».

13.1.8 Соглашения по использованию прерываний от таймеров

Использование прерываний от таймеров зависит от требуемого приоритета операций с таймерами. Если операция с таймером имеет низкий приоритет по сравнению с другими устройствами, то необходимо использовать таймер 1, так как прерывание этого таймера имеет самый низкий приоритет относительно других прерываний. Если операция с таймером требует высокий приоритет по сравнению с другими устройствами, то необходимо использовать таймер 0, так как прерывание этого таймера имеет второй уровень приоритета после NMI.

13.1.9 Конфигурирование выводов таймера

Выводы каждого таймера могут быть сконфигурированы различным способом в зависимости от состояния битов CLKSRC, FUNC и I/O. Четыре конфигурации таймера определяются значениями CLKSRC и FUNC в регистре управления.

13.1.9.1 Значение битов CLKSRC = 1 и FUNC = 0

Если CLKSRC = 1 и FUNC = 0, см. рисунок 13.6, то на вход таймера подается внутренняя синхронизация. Прерывания также генерируются при переходе TSTAT из «0» в «1». Бит INV регистра управления не влияет на внутреннюю частоту. В этом режиме TCLK подключается к I/O порту и может использоваться как вход/выход общего назначения. Если I/O = 0, то TCLK конфигурируется как вход общего назначения, при этом его состояние может быть считано из DATIN. Вывод DATOUT не влияет на TCLK или DATIN. Если I/O = 1, то TCLK конфигурируется как выход общего назначения, то DATOUT устанавливает значение на TCLK и может быть считан из DATIN.

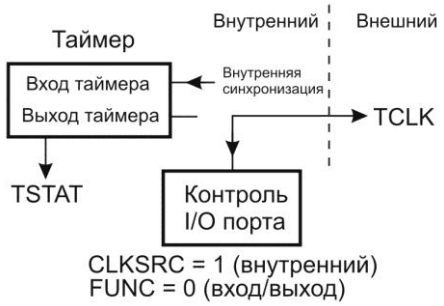


Рисунок 13.6 – Конфигурация выводов таймера при CLKSRC = 1 и FUNC = 0

13.1.9.2 Значение битов CLKSRC = 1 и FUNC = 1

Если CLKSRC = 1 и FUNC = 1, см. рисунок 13.7, то на вход таймера приходит внутренняя синхронизация, а выход таймера подключается к TCLK. Значение TCLK можно инвертировать, устанавливая INV в «1». Также значение TCLK может быть считано из DATIN.

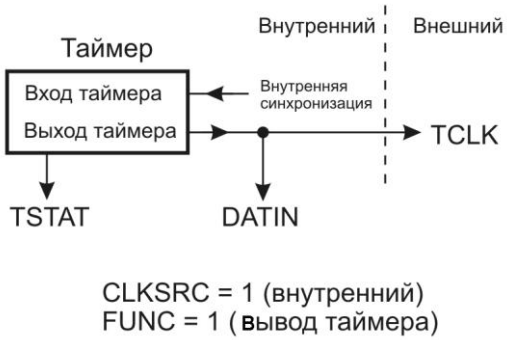


Рисунок 13.7 – Конфигурация выводов таймера при CLKSRC = 1 и FUNC = 1

13.1.9.3 Значение битов CLKSRC = 0 и FUNC = 0

Если CLKSRC = 0 и FUNC = 0, см. рисунок 13.8, то таймер продолжает генерировать сигналы прерывания и управляется в соответствии с битом I/O:

- Если I/O = 0, то таймер синхронизируется от TCLK. Значение TCLK можно инвертировать, устанавливая INV в «1». Также значение TCLK может быть считано из DATIN.

- Если I/O = 1, то TCLK становится выходом, при этом и TCLK, и таймер управляются DATOUT. Все переходы DATOUT из «0» в «1» инкрементируют счетчик. Бит INV не влияет на DATOUT. Значение DATOUT может быть считано из DATIN.

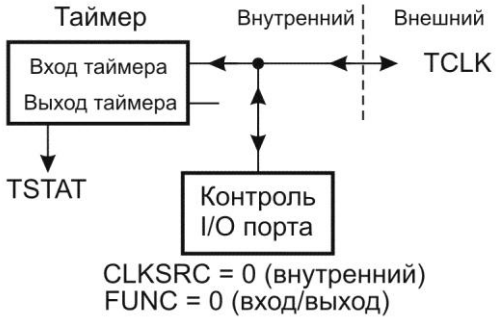


Рисунок 13.8 – Конфигурация выводов таймера при CLKSRC = 0 и FUNC = 0

13.1.9.4 Значение битов CLKSRC = 0 и FUNC = 1

Если CLKSRC = 0 и FUNC = 1, см. рисунок 13.9, то TCLK управляет таймером. Если INV = 0, то все переходы TCLK из «0» в «1» инкрементируют счетчик. Если INV = 1, то все переходы TCLK из «1» в «0» инкрементируют счетчик. Значение TCLK может быть считано из DATIN.

13.1.10 Использование выводов TCLKx как входов/выходов общего назначения

Если FUNC = 0, то TCLKx могут использоваться в качестве входов/выходов общего назначения. На рисунках 13.10 и 13.11 изображено подключение TCLKx в случае использования их в качестве входов/выходов общего назначения. На рисунке 13.10 бит I/O = 0 и TCLK конфигурируется как вход, значение которого может быть считано из бита DATIN.

На рисунке 13.11 бит I/O = 1 и TCLK конфигурируется как выход, значение которого определяется битом DATOUT.

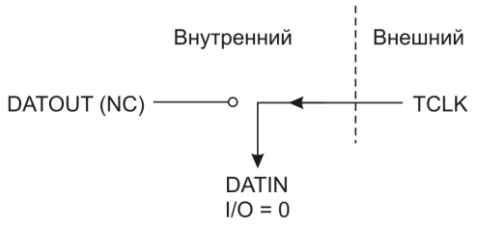


Рисунок 13.10 – TCLK конфигурируется как вход (I/O = 0)

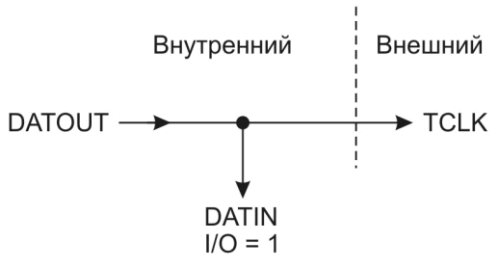


Рисунок 13.11 – TCLK конфигурируется как выход (I/O = 1)

13.1.11 Программирование таймера

Для установки режима работы (конфигурирования) таймера нужно выполнить следующие основные шаги:

- остановить таймер, очистив биты GO/HLD в регистре управления таймером.

Примечание – Таймер останавливается также по RESET;

- сконфигурировать таймер при GO = 0 и HLD = 0, установив в нужное значение регистр счетчика таймера и регистр периода таймера;

- если необходимо, то запустить таймер установкой битов GO/HLD в «1» в регистре управления таймера.

Пример 13.1 показывает, как установить таймер микросхемы для генерации максимальной частоты выходного сигнала на выводах TCLKx.

Пример 13.1 – Фрагмент программы установки максимальной тактовой частоты таймера

```
* Установка максимальной тактовой частоты таймера.
* Пример показывает, как установить таймер для генерации таймером сигнала
* с максимальной частотой, используя внутреннюю синхронизацию.
* Секция “TIMR_REGISTER” располагается, начиная с 10 0020h
TIM0_CTL_REG    .usect “TIMR_REGISTER”, 4
TIM0_CNT_REG    .usect “TIMR_REGISTER”, 4
TIM0_PRD_REG    .usect “TIMR_REGISTER”, 8
                .text
                .
                .
                .
                LDI 0, R0
                STI R0,@TIM0_PRD_REG
                LDI 3C1H, R0
                STI R0,@TIM0_CTL_REG
                .
                .
                .
                .end
```

Приложение А
(обязательное)

Инструкции языка Ассемблер

Множество инструкций ИС 1867ВМ9Ф эффективно выполняют высокопроизводительные числовые вычисления, цифровую обработку сигналов и приложения общего характера.

Для разработки приложений для ИС разработан язык Ассемблер и множество утилит, которые облегчают создание и отладку приложений.

Система инструкций языка Ассемблер ИС может также использовать один из 20 кодов условий с одной из 10 условных команд, таких как LDFcond. Раздел А.2 описывает коды условий и флаги.

Все инструкции языка Ассемблер объединены в основные группы, которые включают инструкции загрузки и сохранения, двух или трехоперандные арифметические/логические инструкции, параллельные инструкции, а также инструкции программного управления межпроцессорным взаимодействием и блокировки доступа к шине. Методы адресации, используемые в данных инструкциях, описаны в разделе А.5 «Инструкции ИС 1867ВМ9Ф».

Ассемблер имеет дополнительные синтаксические формы для упрощения записи текста программы на языке Ассемблер при использовании специальных инструкций. Список дополнительных форм с их описанием приводится.

Каждая отдельная инструкция описана и приведена в алфавитном порядке. В примере описания инструкций приводится использованный формат и приводится его объяснение.

А.1 Система инструкций языка Ассемблер

Система инструкций ИС исключительно хорошо подходит для решения задач ЦОС и других приложений, требующих высокопроизводительных вычислений. Все инструкции имеют длину одно машинное слово, и большинство инструкций выполняются за один такт. В дополнение к инструкциям умножения и накопления ИС имеет полный набор инструкций общего назначения.

Система инструкций состоит из 113 инструкций, организованных в следующие функциональные группы:

- инструкции загрузки и сохранения;
- двухоперандные арифметико-логические инструкции;
- трехоперандные арифметико-логические инструкции;
- инструкции программного управления;
- инструкции управления блокировкой;
- инструкции параллельных операций.

Каждая из этих групп обсуждается в соответствующих подразделах.

А.1.1 Инструкции загрузки и сохранения

ИС поддерживают 24 инструкции загрузки и сохранения, представленные в таблице А.1. Инструкции загрузки и сохранения могут:

- загружать слово из памяти в регистр;
- сохранять слово из регистра в память;
- управлять данными в системном стеке;
- передавать данные между основными и расширенными регистрами.

Две из этих инструкций могут загружать данные по условию. Эти инструкции используются для нахождения минимального или максимального значения из набора данных, смотри раздел А.2 «Коды условий и флаги» для детального описания кодов условий.

Таблица А.1 – Инструкции загрузки и сохранения

Инструкция	Описание	Инструкция	Описание
LBb	Загрузка байта (со знаком)	LDPK	Непосредственная загрузка DP регистра
LBUb	Загрузка байта (без знака)	LHw	Загрузка полуслова со знаком
LDA	Загрузка адресного регистра	LHUw	Загрузка беззнакового полуслова
LDE	Загружает экспоненту с ПЗ	LWLct	Загрузка слова с левым смещением
LDEP	Загрузка целого, расширенного файла регистра в основной регистр	LWRct	Загрузка слова с правым смещением
LDF	Загружает значение с ПЗ	POP	Выталкивает целое слово из стека
LDFcond	Загружает значение с ПЗ по условию	POPF	Выталкивает значение с ПЗ из стека
LDHI	Непосредственная загрузка 16 разрядов без знака в 16 старших разрядов	PUSH	Загружает целое в стек
LDI	Загружает целое	PUSHF	Загрузка в стек значения с ПЗ
LDIcond	Загружает целое По условию	STF	Сохраняет значение с ПЗ
LDM	Загружает мантиссу с ПЗ	STI	Сохраняет целое
LDPE	Загрузка целого, основного регистра в расширенный файловый регистр	STIK	Непосредственное сохранение целого
Примечание – Принятое условное обозначение: ПЗ – плавающая запятая.			

А.1.2 Инструкции с двумя операндами

ИС имеют набор из 43-х двухоперандных арифметических и логических инструкций. Два операнда являются исходным операндом и операндом результата. Операндами-источниками могут быть слово в памяти, содержимое регистра или константа. Операнд результата – всегда регистр.

Двухоперандные инструкции обеспечивают целочисленную арифметику, арифметику с ПЗ или логические операции и арифметику повышенной точности.

В таблице А.2 приведены двухоперандные арифметические и логические инструкции ИС 1867ВМ9Ф.

Таблица А.2 – Двухоперандные инструкции

Инструкция	Описание	Инструкция	Описание
ABSF	Абсолютное значение числа с ПЗ	NEGB	Отрицание целого с заемом
ABSI	Абсолютное значение целого	NEGF	Отрицание числа в формате с ПЗ
ADDC*	Сложить целое с переносом	NEGI	Отрицание целого
ADDF*	Сложить значение с ПЗ	NORM	Нормализовать значение с ПЗ
ADDI*	Сложить целые	NOT	Поразрядное логическое дополнение

Окончание таблицы А.2

Инструкция	Описание	Инструкция	Описание
AND*	Поразрядное логическое «И»	OR*	Поразрядное логическое «ИЛИ»
ANDN*	Поразрядное логическое «И» с дополнением	RCPF*	Обратная величина числа с ПЗ
ASH*	Арифметический сдвиг	RND	Округлить значение с ПЗ
CMPF*	Сравнить значения с ПЗ	ROL	Циклический сдвиг влево
CMPI*	Сравнить целые	ROLC	Левый циклический сдвиг с переносом
FIX	Преобразовать число с ПЗ в целое	ROR	Циклический сдвиг вправо
FLOAT	Преобразовать целое в число с ПЗ	RORC	Циклический сдвиг вправо с переносом
FRIEEE	Преобразует IEEE формат с ПЗ в формат с плавающей запятой в дополнительном коде	RSQRF*	Обратная величина квадратного корня
LSH*	Логический сдвиг	SUBB*	Вычитание целых с заемом
MBct	Слияние байта, левый сдвиг	SUBC	Вычитание целых с условием
MHct	Слияние полуслова, левый сдвиг	SUBF	Вычитание значений с ПЗ
MPYF*	Умножить значения с ПЗ	SUBI	Вычесть целое
MPYI*	Умножить целые	SUBRB	Вычесть обратное целое с заемом
MPYSHI*	Умножение целого со знаком, результат – 32 старших разряда	SUBRF	Вычесть обратное число с ПЗ с заемом
MPYUHI*	Умножение беззнакового целого, результат – 32 старших разряда	SUBRI	Вычесть обратное целое
TOIEEE	Преобразует формат с плавающей запятой в дополнительном коде в IEEE формат	XOR*	Поразрядное «исключающее ИЛИ»
TSTB*	Тестировать разрядные поля		

Примечание – Принятое условное обозначение: ПЗ – плавающая запятая.

* Двух- и трехоперандные версии.

А.1.3 Инструкции с тремя операндами

Большинство инструкций имеет только два операнда, однако некоторые арифметические и логические инструкции имеют трехоперандные версии. 19 трехоперандных инструкций позволяют ИС считывать два операнда из памяти или регистрового файла ЦПУ в один цикл и сохранять результаты в регистре. Ниже описываются различия между двух- и трехоперандными инструкциями:

- двухоперандные инструкции имеют один операнд источника (или сдвиг счетчика) и один операнд результата;
- трехоперандные инструкции могут иметь два операнда источника (или один исходный операнд и операнд счетчика) и один операнд результата. Операндом источника является слово памяти, регистр или константа. Операнд результата в трехоперандных инструкциях – всегда регистр.

В таблице А.3 приведен список трехоперандных инструкций. Необходимо обратить внимание, что число 3 может быть опущено в мнемонике трехоперандной инструкции, смотри подраздел А.3.2.

Таблица А.3 – Трехоперандные инструкции

Инструкция	Описание	Инструкция	Описание
ADDC3	Сложение с переносом	MPYI3	Умножить целые
ADDF3	Сложить значения с ПЗ	MPYSHI3	Умножение целых чисел со знаком
ADDI3	Сложить целые	MPYUHI3	Умножение целых чисел без знака
AND3	Поразрядное логическое «И»	OR3	Поразрядное логическое «ИЛИ»
ANDN3	Поразрядное логическое «И» с дополнением	SUBB3	Вычитание целых с заемом
ASH3	Арифметический сдвиг	SUBF3	Вычитание значений с ПЗ
CMPF3	Сравнить значения с ПЗ	SUBI3	Вычитание целых
CMPI3	Сравнить целые	TSTB3	Тестирование разрядных полей
LSH3	Логический сдвиг	XOR3	Поразрядное «ИСКЛЮЧАЮЩЕЕ ИЛИ»
MPYF3	Умножить значения с ПЗ		

А.1.4 Инструкции программного управления

Группа инструкций программного управления состоит из 24 инструкций, определяющих ход выполнения программы. Инструкции повторения обеспечивают повторение блока инструкций RPTB или RPTBD или отдельной инструкции RPTS. Поддерживаются как стандартные, так и задержанные (одноцикловые) переходы. Некоторые из инструкций программного управления могут зависеть от кодов условий, указанных в разделе А.2 для получения более детальной информации о кодах условий.

В таблице А.4 приведены инструкции программного управления.

Таблица А.4 – Инструкции программного управления

Инструкция	Описание	Инструкция	Описание
Bcond	Переход по условию (стандартный)	LAJ	Скачок
BcondAF	Переход по условию, задержанный с аннулированием, если «ложь»	LAJcond	Скачок по условию
BcondAT	Переход по условию, задержанный с аннулированием, если «истина»	LATcond	Скачок и программное прерывание по условию
BcondD	Переход по условию (задержанный)	NOP	Нет операции
BR	Безусловный переход (стандартный)	RETIcond	Возврат из прерывания по условию
BRD	Безусловный переход (задержанный)	RETIcondD	Возврат из программного прерывания по условию, задержанный
CALL	Вызов подпрограммы	RETScond	Возврат из подпрограммы по условию
CALLcond	Вызов подпрограммы по условию	RPTB	Повтор блока команд
DBcond	Декремент и переход по условию (стандартный)	RPTBD	Повтор блока команд, задержанный
DBcondD	Декремент и переход по условию (задержанный)	RPTS	Повтор отдельной команды
IACK	Подтверждение прерывания	SWI	Программное прерывание
IDLE	Холостая работа до прерывания	TRAPcond	Условное программное прерывание

А.1.5 Инструкции операций блокировки

Инструкции операций блокировки поддерживают мультипроцессорные коммуникации и используют внешние сигналы для обеспечения мощного механизма синхронизации. Они также гарантируют целостность коммуникаций и результатов в высокоскоростных операциях, смотри в разделе А.5 примеры использования инструкций блокировки.

Таблица А.5 – Инструкции операций блокировки

Инструкция	Описание	Инструкция	Описание
LDFI	Загрузить значение с ПЗ, с блокировкой	STFI	Сохранить значение с ПЗ, с блокировкой
LDII	Загрузить целое с блокировкой	STII	Сохранить целое, с блокировкой
SIGI	Сигнализация с блокировкой		

А.1.6 Инструкции параллельных операций

Группа инструкций параллельных операций обеспечивают высокую степень параллелизма. Некоторые инструкции могут объединяться парами, которые выполняются параллельно.

Инструкции параллельных операций обеспечивают следующие возможности:

- параллельную загрузку регистров;
- параллельное сохранение;
- параллельное выполнение арифметических операций;

- арифметико-логические вычисления, выполняемые параллельно с сохранением.

Каждая инструкция в паре записывается как отдельная инструкция. Вторая инструкция в паре должна быть отделена двумя вертикальными черточками (||).

В таблице А.6 приведен список пар команд.

Таблица А.6 – Параллельные инструкции

Мнемоника	Описание
а) Инструкции параллельного выполнения арифметических операций и сохранения	
ABSF||STF	Абсолютное значение числа в формате с ПЗ и сохранение значения с ПЗ
ABSI||STI	Абсолютное значение целого числа и сохранение целого значения
ADDF3||STF	Сложить значения в формате с ПЗ и сохранить значение с ПЗ
ADDI3||STI	Сложить целые и сохранить целое значение
AND3||STI	Поразрядное логическое «И» и сохранение целого значения
ASH3||STI	Арифметический сдвиг и сохранение целого значения
FIX||STI	Преобразовать значение числа в формате с ПЗ в целое и сохранить целое
FLOAT||STF	Преобразовать целое в значение числа в формате с ПЗ и сохранить в формате с ПЗ
FRIEEE||STF	Преобразовать из IEEE формата в формат с ПЗ в дополнительном коде и сохранить в формате с ПЗ
LDF||STF	Загрузить значение в формате с ПЗ и сохранить в формате с ПЗ
LDI||STI	Загрузить целое и сохранить целое значение
LSH3||STI	Логический сдвиг и сохранение целого значения
MPYF3||STF	Умножить значения в формате с ПЗ и сохранить значение с ПЗ
MPYI3||STI	Умножить целое и сохранить целое значение
NEGF||STF	Обратное значение в формате с ПЗ и сохранить значение с ПЗ
NEGI||STI	Обратное целое и сохранить целое значение
NOT||STI	Логическое дополнение (поразрядная инверсия) значения и сохранить целое значение
OR3||STI	Поразрядное логическое «ИЛИ» и сохранение целого значения
STF||STF	Сохранить значения в формате с ПЗ
STI||STI	Сохранить целые значения
SUBF3||STF	Вычесть значение в формате с ПЗ и сохранить значение в формате с ПЗ
TOIEEE||STF	Преобразование в IEEE формат и сохранение
SUBI3||STI	Вычесть целое и сохранить целое значение
XOR3||STI	Поразрядное «ИСКЛЮЧАЮЩЕЕ ИЛИ» значений и сохранить целое значение
б) Инструкции параллельной загрузки	
LDF||LDF	Загрузить значение в формате с ПЗ
LDI||LDI	Загрузить целое значение
в) Инструкции параллельного умножения и сложения/вычитания	
MPYF3||ADDF3	Умножить и прибавить значение в формате с ПЗ
MPYF3||SUBF3	Умножить и вычесть значение в формате с ПЗ
MPYI3||ADDI3	Умножить и прибавить целое значение
MPYI3||SUBI3	Умножить и вычесть целое значение

А.1.7 Недопустимые инструкции

ИС 1867ВМ9Ф не может распознавать недопустимые инструкции. Результатом выполнения недопустимого кода может быть неопределенность. Недопустимый код операции может быть сгенерирован только неправильным использованием программного обеспечения, ошибкой в коде ПЗУ или дефектами ОЗУ.

А.2 Коды условий и флаги

ИС 1867ВМ9Ф имеет 20 кодов условий с 00000_2 по 10100_2, которые могут быть использованы с условными командами, такими как RETScond или LDFcond, за исключением кода 01011_2.Условиями являются знаковые и беззнаковые сравнения, сравнение с нулем и сравнения, основанные на состоянии отдельных флагов условий. Необходимо обратить внимание, что все условные инструкции могут включать суффикс U для обозначения безусловной операции.

Семь флагов условий содержат информацию о статусе арифметических или логических операций. Флаги условий хранятся в регистре состояния ST. Влияние инструкции на флаг условия зависит от значения поля SET COND (бит 15 регистра состояния). Значение SET COND 0 или 1 не влияет на инструкции сравнения CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3.

Если SET COND = 0, то флаги условий регистра ST устанавливаются, если объектом операции является один из регистров с повышенной точностью (R0-R11).

Если SET COND = 1, то флаги условий регистра ST устанавливаются, если объектом операции является один из регистров основного регистрового файла, за исключением регистра состояния.

Флаги условий могут изменяться многими инструкциями или в следующих случаях:

- результат получен при выполнении определенной операции с бесконечной точностью. Это возможно при выполнении операций сравнения и тестирования, которые не сохраняют результат в регистре или при выполнении арифметических операций, которые приводят к переполнению или отрицательному переполнению (потере значимости);
- результат записан в регистр назначения, как показано в таблице А.7. Этот случай подходит и для других инструкций, изменяющих флаги условий.

Таблица А.7 – Форматы выходных значений

Тип операции	Выходной формат
С плавающей запятой	8-разрядная экспонента, 1-знаковый разряд, 31-разрядная дробная часть
Целая	32-разрядное целое
Логическая	32-разрядное беззнаковое целое

На рисунке А.1 приведены флаги условий в младших разрядах регистра состояния. Далее следует список флагов условий регистра состояния и описание того, как данные флаги устанавливаются различными командами. Для более детального описания влияния отдельных команд на флаги условий необходимо смотреть описание команд в подразделе А.3.3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	Analysis
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SET COND	PGIE	GIE	CC	CE	CF	PCF	RM	OVM	LUF	LV	UF	N	Z	V	C
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Примечание – Принятые условные обозначения: xx – резервные разряды;

R = Чтение, W = Запись.

Рисунок А.1 – Регистр состояния

Описание флагов

LUF – Флаг, фиксирующий отрицательное переполнение при работе с ПЗ. Устанавливается всякий раз при установке флага UF (флаг отрицательного переполнения при работе в формате с ПЗ). Флаг LUF может быть очищен только при сбросе или при изменении регистра состояния ST.

LV – Флаг, фиксирующий переполнение. Устанавливается всякий раз при установке флага V (флаг условия переполнения). Флаг LV может быть очищен только при сбросе или при изменении регистра состояния ST.

UF – Флаг отрицательного переполнения при работе в формате с ПЗ. Отрицательное переполнение при работе с ПЗ возникает всякий раз, когда экспонента результата меньше или равна минус 128. При возникновении отрицательного переполнения флаг UF устанавливается, и выходное значение устанавливается в «0». Если не происходит отрицательного переполнения, то флаг UF очищается.

N – Флаг отрицательного значения. Логические операции присваивают флагу N значение старшего значащего разряда выходного значения. Для целочисленных операций и операций с ПЗ флаг N устанавливается, если результат отрицательный и очищается в противном случае. Ноль означает положительное значение.

Z – Флаг нулевого значения. Для логических, целочисленных и операций с ПЗ флаг Z устанавливается при равенстве выходного результата «0» и очищается в противном случае.

V – Флаг переполнения. Для целочисленных операций флаг V устанавливается, если результат не соотвествует формату, определенному для приемника (т. е. $-2^{32} \le$ результат $\le 2^{32} - 1$), в противном случае флаг V очищается. Для операций в формате с ПЗ флаг V устанавливается, если экспонента результата больше 127, в противном случае флаг V очищается. Логические операции всегда очищают V.

C – Флаг C при выполнении целочисленного сложения очищается в случае возникновения переноса относительно старшего значащего разряда результата. При выполнении целочисленного вычитания флаг C устанавливается при возникновении заема в разряде, относящемся к старшему значащему разряду результата. В противном случае при целочисленных операциях флаг C очищается. Для инструкций сдвига этот флаг устанавливается по значению последнего сдвигаемого вовне разряда; при нулевом значении сдвига флаг устанавливается в ноль.

Таблица А.8 содержит мнемонику кодов условий, описание кода условия и устанавливаемые флаги для каждого из 20 кодов условий.

Таблица А.8 – Коды условий и флаги

Условие	Двоичный код	Описание	Флаг*
а) Безусловные сравнения			
U	00000	Безусловный	Безусловное
б) Беззнаковые сравнения			
LO	00001	Меньше чем	C
LS	00010	Меньше или равно	C ИЛИ Z
HI	00011	Больше чем	~C И ~Z
HS	00100	Больше или равно	~C
EQ	00101	Равно	Z
NE	00110	Не равно	~Z
в) Знаковые сравнения			
LT	00111	Меньше чем	N
LE	01000	Меньше или равно	N ИЛИ Z
GT	01001	Больше чем	~N И ~Z
GE	01010	Больше или равно	~N
EQ	00101	Равно	Z
NE	00110	Не равно	~Z
г) Сравнение с нулем			
Z	00101	Ноль	Z
NZ	00110	Не ноль	~Z
P	01001	Положительное	~N И ~Z
N	00111	Отрицательное	N
NN	01010	Не отрицательное	~N
д) Сравнение с флагами условий			
NN	01010	Не отрицательное	~N
N	00111	Отрицательное	N
NZ	00110	Не ноль	~Z
Z	00101	Ноль	Z
NV	01100	Нет переполнения	~V
V	01101	Переполнение	V
NUF	01110	Нет отрицательного переполнения	~UF
UF	01111	Отрицательное переполнение	UF
NC	00100	Нет переноса	~C
C	00001	Перенос	C
NLV	10000	Нет фиксируемого переполнения	~LV
LV	10001	Фиксируемое переполнение	LV
NLUF	10010	Нет фиксируемого отрицательного переполнения с ПЗ	~LUF
LUF	10011	Фиксируемое отрицательное переполнение с ПЗ	LUF
ZUF	10100	Ноль или отрицательное переполнение с ПЗ	Z ИЛИ UF

Примечание – Принято условное обозначение инверсии «~».

* Означает логическое дополнение (условие «не истина»).

A.3 Описание отдельных инструкций

Раздел A.3 содержит описание отдельных инструкций языка Ассемблер для ИС 1867ВМ9Ф. Инструкции приведены в алфавитном порядке. Информация о каждой инструкции включает синтаксис языка Ассемблер, выполняемую операцию, операнды, код, описание, количество циклов, биты состояния, бит режима и примеры.

Определение символов и аббревиатур, также как и дополнительных синтаксических форм, поддерживаемых языком Ассемблер, находится в начале раздела описания отдельных инструкций. Также приводится пример инструкции с описанием используемого специального формата и объясняется ее содержание.

Группировка инструкций по выполняемым функциям и полный список инструкций приведен в разделе A.1.

A.3.1 Символы и аббревиатуры

В таблице A.9 приведены символы и аббревиатуры, использованные в описании отдельных инструкций.

Таблица A.9 – Символы инструкций

Символ	Назначение
src	Операнд источника
src1	Операнд источника 1
src2	Операнд источника 2
src3	Операнд источника 3
src4	Операнд источника 4
dst	Операнд назначения (результата)
dst1	Операнд назначения (результата)
dst2	Операнд назначения (результата)
disp	Смещение
cond	Условие
count	Счетчик сдвига
G	Основные методы адресации
T	Трехоперандные методы адресации
P	Параллельные методы адресации
B	Методы адресации с условным переходом
ARn	Вспомогательный регистр n
IRn	Индексный регистр n
Rn	Регистр повышенной точности n
RC	Регистр счетчика повторений
RE	Регистр конечного адреса повторений
RS	Регистр начального адреса повторений
ST	Регистр состояния
C	Бит переноса
GIE	Бит разрешения глобальных прерываний
N	Вектор программного прерывания (trap)
PC	Счетчик команд
RM	Флаг режима повторения
SP	Указатель системного стека
|x|	Абсолютное значение x
x →y	Присваивает значение x приемнику y
x(man)	Поле мантиссы (знак + дробная часть) от x
x(exp)	Поле экспоненты
op1||op2	Операция 1 выполняется параллельно с операцией 2

Окончание таблицы А.9

Символ	Назначение
x AND y	Поразрядное логическое «И» значений x и y
x OR y	Поразрядное логическое «ИЛИ» значений x и y
x XOR y	Поразрядное логическое «ИСКЛЮЧАЮЩЕЕ ИЛИ» значений x и y
~x	Поразрядное логическое дополнение (инверсия) значения x
x << y	Сдвиг числа x влево на y разрядов
x >> y	Сдвиг числа x справа на y разрядов
*++SP	Инкрементировать SP и использовать инкрементированный SP как адрес операнда
*SP--	Использовать SP как адрес операнда и декрементировать SP
Примечание – Принято условное обозначение инверсии «~».	

А.3.2 Дополнительный синтаксис языка Ассемблер

Язык Ассемблер позволяет использовать упрощенную форму записи некоторых команд. Эти дополнительные формы упрощают синтаксис языка Ассемблер, например:

Регистр назначения может быть опущен в унарных (с одним операндом) арифметических и логических операциях, когда один и тот же регистр используется в качестве источника и приемника, например,

ABSI R0, R0 может быть записан как

ABSI R0

Описанный выше пример используется в инструкциях: ABSI, ABSF, FIX, FLOAT, NEGB, NEGF, NEGI, NORM, NOT, RND.

Все трехоперандные инструкции могут быть записаны без суффикса 3, например,

ADDI3 R0, R1, R2

можно записать как

ADDI R0, R1, R2

и использовать в инструкциях: ADDC3, ADDF3, ADDI3, AND3, ANDN3, ASH3, LSH3, MPYF3, MPYI3, OR3, SUBB3, SUBF3, SUBI3, XOR3, MPYSHI3, MPYUHI3.

Все трехоперандные инструкции сравнения можно записать без суффикса 3, например,

CMPI3 R0,*AR0

можно записать как

CMPI R0,*AR0

и использовать в инструкциях: CMPI3, CMPF3, TSTB3.

Разрешена запись операндов с косвенной адресацией и с явно заданным нулевым смещением. В трехоперандных или параллельных инструкциях операнды с нулевым смещением автоматически преобразуются в режим с отсутствием смещения, например, LDI *+AR0 (0), R1 – такая запись разрешена.

Также

ADDI3 *+AR0 (0), R1, R2

эквивалентно записи

ADDI3 *+AR0, R1, R2

Операнды с косвенной адресацией могут быть записаны без смещения, в этом случае предполагается единичное смещение, например,

LDI *AR0++(1), R0

может быть записано как

LDI *AR0++, R0

Все условные инструкции могут включать в себя суффикс U для обозначения безусловной операции, также суффикс U может быть исключен из команды короткого безусловного перехода, например,

BU метка

может быть записана как

В метка

Метки могут быть записаны как с последующим символом (:), так и без него. Например,

label0: NOP
label1 NOP
label2: (метка относится к следующей строке)

Пустые выражения для указания смещения (выражение в скобках) в косвенном методе адресации запрещены:

LDI *+AR0(), R0 – запрещено

Операнды длинного непосредственного метода адресации (назначение операндов инструкций BR и CALL) могут быть написаны со знаком @, например,

BR метка

может быть записана как

BR @метка

Псевдооперация LDP может быть использована для загрузки регистра (обычно DP) 16 старшими разрядами перемещаемого адреса следующим образом, например,

LDP addr, REG или LDP @addr, REG или LDP addr

Знак @ является дополнительным. Инструкция LDP генерирует инструкцию LDIU с непосредственным операндом и специальным типом смещения.

Параллельные инструкции могут быть написаны в другом порядке, например,

ADDI
|| STI

может быть записана как

STI
|| ADDI

Символы (||), определяющие вторую часть параллельной инструкции, могут быть написаны в любом месте строки, начиная с нулевого столбца, например,

ADDI
|| STI

может быть записана как

ADDI || STI

Если второй операнд параллельной инструкции такой же, как и третий (регистр назначения), то третий операнд может быть опущен. Т.е. это дает возможность написания трехоперандной команды, которая выглядит как нормальная двухоперандная инструкция, например,

ADDI *AR0, R2, R2
|| MPYI *AR1, R0, R0

может быть записана как

ADDI *AR0, R2, R2
|| MPYI *AR1, R0

Инструкции, которые могут использовать эту запись (применимо для всех параллельных инструкций, имеющих в качестве второго операнда регистр): ADDI, ADDF, AND, MPYI, MPYF, OR, SUBI, SUBF, XOR.

Все коммутативные операции в параллельных инструкциях могут быть записаны в другом порядке, например, часть параллельной инструкции ADDI может быть записана одним из двух способов:

ADDI *AR0 R1, R2

или

ADDI R1,*AR0, R2

Описанное выше относится к параллельным инструкциям – ADDI, ADDF, MPYI, MPYF, AND, OR, XOR.

Для описания регистров ЦПУ в операндах используется синтаксис в таблице А.10.

А.3.3 Описание инструкций

В подразделе А.3.3 описана каждая инструкция Ассемблера ИС в алфавитном порядке. Информация об инструкции включает синтаксис языка Ассемблер, операнды, код операции, производимую операцию, описание, статусные биты, количество циклов, биты состояния, биты модификации и пример (примеры).

Таблица А.10 – Синтаксис регистров ЦПУ

Синтаксис Ассемблера	Адрес регистра (hex)	Назначение регистра
R0	00	Регистр повышенной точности 0
R1	01	Регистр повышенной точности 1
R2	02	Регистр повышенной точности 2
R3	03	Регистр повышенной точности 3
R4	04	Регистр повышенной точности 4
R5	05	Регистр повышенной точности 5
R6	06	Регистр повышенной точности 6
R7	07	Регистр повышенной точности 7
R8	1C	Регистр превышенной точности 8
R9	1D	Регистр повышенной точности 9
R10	1E	Регистр повышенной точности 10
R11	1F	Регистр повышенной точности 11
AR0	08	Вспомогательный регистр 0
AR1	09	Вспомогательный регистр 1
AR2	0A	Вспомогательный регистр 2
AR3	0B	Вспомогательный регистр 3
AR4	0C	Вспомогательный регистр 4
AR5	0D	Вспомогательный регистр 5
AR6	0E	Вспомогательный регистр 6
AR7	0F	Вспомогательный регистр 7
DP	10	Указатель страницы данных
IR0	11	Индексный регистр 0
IR1	12	Индексный регистр 1
BK	13	Регистр размера блока
SP	14	Указатель системного стека
ST	15	Регистр состояния
DIE	16	Регистр разрешения прерывания ПДП
IIE	17	Регистр разрешения внутреннего прерывания
IIF	18	IIOF выводы и регистр флага прерывания
RS	19	Регистр адреса начала повторения
RE	1A	Регистр адреса конца повторения
RC	1B	Счетчик повторений
IVTP	00	Указатель на таблицу векторов системных прерываний
TVTP	01	Указатель на таблицу векторов программных прерываний

А.4 Описание примеров инструкций

Для примера, более подробно рассмотрим описание инструкции INST, далее описание остальных инструкций будет менее подробным.

Пример – Инструкция INST

Синтаксис:

INST src, dst

или

INST1 src2, dst1

|| INST2 src3, dst2

Каждая инструкция начинается с синтаксического выражения на языке Ассемблер. Метки могут размещаться либо до команды (мнемонического выражения) на той же строке, либо на предыдущей строке в первом столбце. Дополнительное поле комментария, которое завершает синтаксис, не включается в синтаксическое выражение. Пробелы требуются между всеми полями (метка, команда, операнд и поля комментария).

Примеры синтаксиса иллюстрируют общий одностроковый и двухстроковый синтаксис, используемый в параллельной адресации. Необходимо обратить внимание, что символ (||), обозначающий параллельную адресную пару, может быть размещен где угодно перед мнемоникой во второй строке. Первая команда в паре может иметь метку, но вторая иметь метки не может.

Операнды:

src: основные методы адресации (G)

dst: регистровая адресация (R0-R11)

Здесь представлен список типов операндов, которые используются в инструкции.

Код операции:

31 24	23 16	15 8	7 0
0 0 0 INST	G dst	src	

31	24 23	16	15	8	7 0
1 1 INST1 || INST2	dst1	0 0 0	src3	dst2	src2

Примеры кодов приведены с использованием основной и параллельной адресации. Пара команд для примера с параллельной адресацией состоит из операций INST1 и INST2.

Примечание – Оба отдельных кода операции в этом случае – это 32-разрядные инструкции.

Поле слова инструкции:

G	Методы адресации источника
00	Регистровая (R0-R11)
01	Прямая
10	Косвенная
11	Непосредственная

Поле G слова инструкции описывает метод адресации, который соответствует каждому полю операнда в слове инструкции.

Операция:

|src| → dst

или

|src2| → dst1

|| src3 → dst2

Последовательность выполнения инструкции описывает порядок выполнения инструкции. Для параллельных инструкций последовательность выполняется параллельно. Флаги условий в регистре состояния определяют режимы для инструкций с условиями, например, Bcond.

dst: регистровая адресация (любой регистр из основного регистрового файла в ЦПУ)
или
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst1: регистровая адресация (R0-R7)
src3: регистровая адресация (R0-R7)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)
Операнды определены в соответствии с методом адресации и/или типом использованной адресации. Необходимо обратить внимание, что косвенная адресация использует смещения и индексные регистры. Смотри раздел 6 «Методы адресации» КФДЛ.431282.006ТО для получения более полной информации по режимам адресации.
Описание:
Описано выполнение инструкции, ее влияние на состояние регистров процессора и содержимое памяти. Описаны также ограничения, накладываемые на операнды процессором или языком Ассемблер. Описание идет параллельно и дополняет информацию, приведенную в описании работы.

Статусные биты

LUF – флаг фиксирующий условие отрицательного переполнения при работе с ПЗ. Устанавливается в «1» всякий раз при установке флага UF (флаг отрицательного переполнения при работе в формате с ПЗ), в противном случае не изменяется.

LV – флаг фиксирующий условие переполнения. Устанавливается в «1» всякий раз при переполнении, в противном случае не изменяется.

UF – флаг условия отрицательного переполнения при работе в формате с ПЗ.

При потере значимости результата с ПЗ флаг UF устанавливается в «1», в противном случае устанавливается в «0».

N – флаг отрицательного условия. Для некоторых операций флаг N имеем значение старшего значащего разряда выходного значения «1», если результат отрицательный и «0» в противном случае.

Z – флаг нулевого условия. Флаг равен «1» при равенстве выходного результата равного нулю и «0» в противном случае. Для логических команд и команд сдвига при получении нулевого результата флаг равен «1» и «0» в противном случае.

V – флаг условия переполнения. Устанавливается в «1» при переполнении и в «0» в противном случае.

C – флаг переноса устанавливается в «1» при возникновении заема или переноса и в «0» в противном случае. Для команд сдвига этот флаг устанавливается по значению последнего сдвигаемого вовне разряда. При нулевом сдвиге устанавливается в ноль.

Семь флагов условий (статусных флагов), сохраняемые в регистре состояния (ST), изменяются большинством инструкций, только если регистр назначения представляет собой один из (R7-R0). Они дают информацию о статусе результата или статусе результата арифметических или логических операций.

Бит режима:
Флаг режима переполнения OVM. В общем случае, на выполнение целочисленных операций влияет значение бита OVM.
В основном, целые операции влияют на значение бита OVM.
Циклы:
1
Цифра определяет число циклов, необходимых для осуществления инструкции.

Пример:

```
INST @98AEh, R5
```

Перед инструкцией		После инструкции	
DP	80h	80h	
R5	07 6690 0000h (2.30562500e + 02)	R5	00 6690 0000h (1.80126593e + 00)
Память в 0080 98AEh		Память в 0080 98AEh	
5CDFh (1.00001107e + 00)		5CDFh (1.00001107e + 00)	
LUF	0	LUF	0
LV	0	LV	0
UF	0	UF	0
N	0	N	0
Z	0	Z	0
V	0	V	0
C	0	C	0

Код, представленный в вышеописанном формате, показывает влияние инструкции на системные указатели (DP или SP), регистры (R1 или R5), значение в памяти по определенному адресу и семь статусных битов. Значения, заданные в регистрах, включают ноль для указания экспоненты в операциях с ПЗ. Для регистров и адресов памяти представлен перевод шестнадцатиричных значений в число с основанием десять. Более подробная информация о статусных битах приведена в таблице А.8.

А.5 Инструкции ИС 1867ВМ9Ф

А.5.1 Инструкция ABSF

Синтаксис:
ABSF src, dst
Операнды:
src: основные методы адресации
dst: регистры (R0-R11)
Код:

31 29	23 21	16	0
0 0 0	0 0 0 0 0 0	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (R0-R11)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
|src| → dst
Описание:
Абсолютное значение операнда src загружается в регистр dst. Операнды src и dst являются числами в формате с ПЗ.
Переполнение происходит, если src(man) = 8000 0000h и src(exp) = 7Fh. Результат – dst(man)=7FFF FFFFh и dst(exp)=7Fh
Статусные биты:

LUF – не изменяется.
LV = 1 при возникновении переполнения с ПЗ,
в противном случае не меняется.
UF = 0
N = 0
Z = 1 при генерации нулевого результата, иначе =0
V = 1 при возникновении ПЗ переполнения, иначе = 0.
C – не изменяется.

Бит режима:
Операция OVM зависит от значения бита OVM.

Циклы:
1

Пример:

ABSF R4, R7

	Перед инструкцией			После инструкции	
R4	05C 8000 F971h	-9.90337307e + 27	R4	05C 8000 F971h	-9.9033737e + 27
R7	07D 2511 00AEh	5.48527255e + 37	R7		
LV	0		LV	0	
Z	0		Z	0	
V	0		V	0	

A.5.2 Инструкция ABSF||STF

Синтаксис:
ABSF src2, dst1
|| STF src3, dst2
Операнды:
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst1: регистровая адресация (R0-R7)
src3: регистровая адресация (R0-R7)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)
Код:

31 29	24 23	16	15 8	7 0
1 1	0 0 1 0 0	dst1	0 0 0	src3	dst2	src2

Поле слова инструкции:
Нет

Операция:

|src2| → dst1
|| src3 → dst2

Описание:

Вычисление абсолютного значения числа в формате с ПЗ и параллельно сохранение значения в формате с ПЗ. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций STF считывает из регистра, и операция, которая производится параллельно ABSF, записывает в тот же регистр, то STF имеет на входе содержимое регистра до того, как оно было изменено инструкцией ABSF.
Если src2 и dst2 указывают на один и тот же адрес, то src2 считывается до записи в dst2.
Если src3 и dst1 указывают на один и тот же адрес, то src3 считывается до записи в dst1.
Переполнение возникает, если src(man) = 8000 0000h и src(exp) = 7Fh. Результат – dst(man) = 7FFF FFFFh и dst(exp)=7Fh.

Циклы:

1

Статусные биты:

LUF – не изменяется.
LV = 1 при возникновении ПЗ переполнения, в противном случае не меняется.
UF = 0
N = 0
Z = 1 при генерации нулевого результата, иначе = 0.
V = 1 при возникновении ПЗ переполнения, иначе = 0.
C – не изменяется.

Пример:

ABSF *++AR3 (IR1), R4
|| STF R4,*-AR7 (1)

Перед инструкцией

AR3	80 9800h	
IR1	0AEFh	
R4	733C0 0000h	1.79750e+02
AR7	80 98C5h	

Данные в 80 98AFh

	58B 4000h	-6.118750e+01

Данные в 80 98C4h

	0h	
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR3	80 98AFh	
IR1	0AFh	6.118750e+01
R4	0	
AR7	80 98C5h	

Данные в 80 98AFh

	58B 4000h	-6.118750e+01

Данные в 80 98C4h

	733 C000h	1.79750e+02
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.3 Инструкция ABSI

Синтаксис:
ABSI src, dst

Операнды:

src: основные методы адресации
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

31 24	23 16	15 8	7 0
0 0 0	0 0 0 0 0 1	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (R0-R11)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
|src| → dst

Описание:

Абсолютное значение операнда src загружается в регистр dst. Операнды src и dst являются целыми со знаком.
Переполнение возникает, если src=8000 0000h. Если ST(OVM)=1, то результатом будет dst=7FFF FFFFh. Если ST(OVM)=0, то результатом будет dst=8000 0000h.
Статусные биты:
Флаги состояния изменяются только, если регистр назначения – один из (R7-R0) и если ST(SET COND)=0. Если ST(SET COND)=1, они изменяются для всех вспомогательных регистров.

LUF – не изменяется.
LV = 1 при возникновении целочисленного переполнения,
= 0 в противном случае не меняется.
UF = 0
N = 0
Z = 1 при генерации нулевого результата, иначе = 0.
V = 1 при возникновении целочисленного переполнения, иначе = 0.

C – не изменяется.

Бит режима:

Операция OVM зависит от значения бита OVM.

Циклы:

1

Пример 1:

```
ABSI  R0, R0
или
ABSI  R0
```

	Перед инструкцией			После инструкции	
R0	0FFFF FFCBh	-53	R0	035h	53

Пример 2:

```
ABSI  *AR1, R3
```

	Перед инструкцией			После инструкции	
AR1	20h		AR1	20h	
R3	0h		R3	35h	53
Данные в 20h			Данные в 20h		
	0FFFF FFCBh	-53		0FFFF FFCBh	-53

A.5.4 Инструкция ABSI ||STI

Синтаксис:

```
ABSI src2, dst1
|| STI src3, dst2
```

Операнды:

src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst1: регистровая адресация (R0-R7)
src3: регистровая адресация (R0-R7)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)

Код:

31 24	23 16	15 8	7 0
1 1 | 0 0 1 0 1	dst1 | 0 0 0 | src3	dst2	src2

Поле слова инструкции:

Нет

Операция:

|src2| → dst1
|| src3 → dst2

Описание:

Вычислить абсолютное значение целого числа и параллельно сохранить значение целого. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций STI считывает из регистра, и операция, которая выполняется параллельно (ABSI), записывает в тот же регистр, то STI имеет на входе содержимое регистра, которое было до того, как оно было изменено командой ABSI.

Если src2 и dst2 указывают на один и тот же адрес, то src2 считывается до записи в dst2.

Переполнение возникает, если src = 8000 0000h. Если ST(OVM)=1, то результат – dst = 7FFF FFFFh. Если ST(OVM)=0, то результат – dst = 8000 0000h.

Статусные биты:

LUF – не изменяется.
LV = 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF = 0
N = 0

Z = 1 при генерации нулевого результата, иначе = 0.
V = 1 при возникновении целочисленного переполнения, иначе = 0.
C – не изменяется.

Бит режима:
Операция OVM зависит от значения бита OVM.

Циклы:
1

Пример:

```
ABSI *-AR5(1), R5
|| STI R1,*-AR2--(IR1)
```

Перед инструкцией

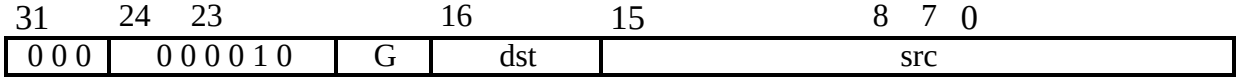
AR5	80 99E2h	
R5	0h	
R1	42h	66
AR2	80 98FFh	
Данные в 80 99E1h		
	58B 4000h	-53
Данные в 80 98FFh		
	0	2
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR5	80 99E2h	
R5	35h	53
R1	42h	66
AR1	80 98F0h	
Данные в 80 99E1h		
	58B 4000h	-53
Данные в 80 98FFh		
	733 C000h	66
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.5 Инструкция ADDC

Синтаксис:
ADDC src, dst
Операнды:
src: основные методы адресации
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:



Поле слова инструкции:

G	Методы адресации src
00	Регистровая (R0-R11)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
dst + src + C → dst
Описание:
Сумма операндов dst и src и флага C (перенос) загружается в регистр dst. Операнды src и dst являются целыми со знаком.
Статусные биты:
Флаги состояния изменяются только, если регистр назначения – один из (R7-R0).
LUF – не изменяется.
LV = 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF = 0
N = 1 при генерации отрицательного результата, иначе = 0.

Z = 1 при генерации нулевого результата, иначе = 0.
V = 1 при возникновении целочисленного переполнения, иначе = 0.
C = 1 при возникновении переноса, иначе = 0.

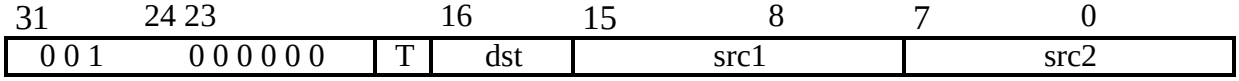
Бит режима:
Операция OVM зависит от значения бита OVM.
Циклы:
1

Пример:
ADDC R1, R5

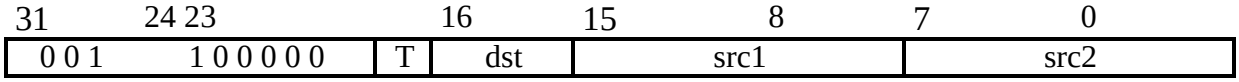
	Перед инструкцией			После инструкции	
R1	00 FFFF 5C25h	-41 947	R1	00 FFFF 5C25h	-41 947
R5	00 FFFF 019Eh	-65 122	R5	00 FFFE 5DC4h	-107 068
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	0		C	0	

A.5.6 Инструкция ADDC3

Синтаксис:
ADDC3 src2, src1, dst
Операнды:
src1, src2: тип 1 или тип 2 трехоперандного метода адресации
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:
Тип 1



Тип 2



Поля слова инструкции:

Тип 1

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	Регистровая (любой регистр ЦПУ)
01	Косвенная (смещение=0, 1, IR0, IR1)	Регистровая (любой регистр ЦПУ)
10	Регистровая (любой регистр ЦПУ)	Косвенная (смещение=0, 1, IR0, IR1)
11	Косвенная (смещение=0, 1, IR0, IR1)	Косвенная (смещение=0, 1, IR0, IR1)

Тип 2

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	8-разрядная знаковая непосредственная
01	Регистровая (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-разрядная знаковая непосредственная
11	Косвенная *+ARn1 (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:
src1 + src2 + C → dst
Описание:
Сумма операндов src1 и src2 и флага C (перенос) загружается в регистр dst. Операнды src1, src2 и dst являются целыми со знаком.

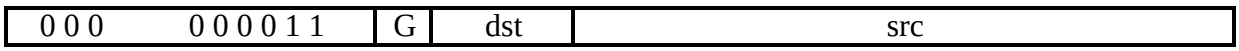
Статусные биты:
Если ST(SET COND)=0, то флаги состояния изменяются только в том случае, если регистр назначения – один из (R0-R11). Если ST(SET COND)=1, то флаги изменяются для всех регистров.

LUF	– не изменяется.
LV	=1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF	=0
N	=1 при генерации отрицательного результата, иначе = 0.
Z	=1 при генерации нулевого результата, иначе = 0.
V	=1 при возникновении целочисленного переполнения, иначе = 0.
C	=1 при возникновении переноса, иначе = 0.

Бит режима:
Операция OVM зависит от значения бита OVM.
Циклы:
1
Пример:
Нет

A.5.7 Инструкция ADDF

Синтаксис:
ADDF src, dst
Операнды:
src: основные методы адресации
dst: регистровая адресация (R0-R11)
Код:



Поле слова инструкции:

G	Методы адресации src
00	Регистровая (R0-R11)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
dst + src → dst
Описание:
Сумма операндов dst и src загружается в регистр dst. Операнды src и dst – числа в формате с плавающей запятой.
Статусные биты:

LUF	=1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	=1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF	=1 при потере значимости результата с ПЗ, иначе = 0.
N	=1 при генерации отрицательного результата, иначе = 0.
Z	=1 при генерации нулевого результата, иначе = 0.
V	=1 при возникновении ПЗ-переполнения, иначе = 0.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:
ADDF*AR4++(IR1), R5

	Перед инструкцией		После инструкции	
AR4	80 9800h		80 992Bh	
R1	12Bhh	66	12Bh	
R5	057980 0000h	6.23750e+01	09052C 0000h	5.3268750e+02
Данные в 80 9800h			Данные в 80 9800h	
	86B 2800h	4.7031250e+02	86B 2800h	4.7031250e+02
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		0	
V	0		0	
C	0		0	

A.5.8 Инструкция ADDF3

Синтаксис:
ADDF3 src2, src1, dst
Операнды:
src1, src2: тип 1 или тип 2 трехоперандного метода адресации
dst: режим регистра (R0-R11)
Код:
Тип 1

31	24 23		16	15	8	7	0
0 0 1	0 0 0 0 0 1	T	dst	src1		src2	

Тип 2

31	24 23		16	15	8	7	0
0 0 1	1 0 0 0 0 1	T	dst	src1		src2	

Поля слова инструкции:

Тип 1

T	Методы адресации src1	Методы адресации src2
00	Регистровая (R0-R11)	Регистровая (R0-R11)
01	Косвенная (смещение=0, 1, IR0, IR1)	Регистровая (R0-R11)
10	Регистровая (R0-R11)	Косвенная (смещение=0, 1 ,IR0, IR1)
11	Косвенная (смещение=0, 1, IR0, IR1)	Косвенная (смещение=0, 1, IR0, IR1)

Тип 2

T	Методы адресации src1	Методы адресации src2
01	Регистровая (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
11	Косвенная *+ARn1 (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:
src1 + src2 $\rightarrow$ dst
Описание:

Сумма операндов src1 и src2 загружается в регистр dst. Операнды src1, src2 и dst являются числами в формате с ПЗ.
Статусные биты:

LUF	= 1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	= 1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF	= 1 при потере значимости результата с ПЗ, иначе = 0.
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении ПЗ-переполнения, иначе = 0.
C	– не изменяется.

Циклы:
1
Бит режима:
Операция OVM не зависит от значения бита OVM.
Пример:
ADDF3 *+AR1 (2),*+AR1 (8),

Перед инструкцией

AR1	2FF820h	
R4	0h	
Данные в 22F F822h		
	700 F000h	4.7031250e+02
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR1	2FF820h	
R4	070DB2 0000h	1.41695313e+02
Данные в 22F F828h		
	34C 2000h	1.27590e+01
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.9 Инструкция ADDF3||STF

Синтаксис:
ADDF3 src2, src1, dst1
|| STF src3, dst2
Операнды:
src1: регистровая адресация (R0-R7)
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst1: регистровая адресация (R0-R7)
src3: регистровая адресация (R0-R7)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)
Код:

31 24	23 16	15 8	7 0
1 1 0 0 1 1 0	dst	src1	src2

Операция:
src1 + src2 → dst1
|| src3 → dst2
Описание:
Сложение в формате с ПЗ и сохранение числа в формате с ПЗ производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STF) считывает из регистра, и операция, которая производится параллельно (ADDF3), записывает в тот же регистр, то STF имеет на входе содержимое регистра до того, как оно было изменено командой ADDF3.

Если src2 и dst2 указывают на один и тот же адрес, то src2 считывается до записи в dst2.

Статусные биты:

LUF	=1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	=1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF	=1 при потере значимости результата с ПЗ, иначе = 0.
N	=1 при генерации отрицательного результата, иначе = 0.
Z	=1 при генерации нулевого результата, иначе = 0.
V	=1 при возникновении ПЗ-переполнения, иначе = 0.
C	– не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Циклы:

1

Пример:

```
ADDF3 *+AR3 (IR1), R2, R5
|| STF  R4,*AR2
```

Перед инструкцией

AR3	80 9800h	
IR1	0A5h	
R2	070C80 0000h	1.4050e+02
R5	0h	
R4	057B40 0000h	6.2081250e+01
AR2	80 98F3h	

Данные в 80 98A5h

	733 C000h	1.79750e+02

Данные в 80 98F3h

	0h	2
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR3	80 9800h	
IR1	0A5h	
R2	070C80 0000h	1.4050e+02
R5	082020 0000h	3.20250e+02
R4	057B40 0000h	6.2081250e+01
AR2	80 98F3h	

Данные в 80 98A5h

	58B 4000h	1.79750+02

Данные в 80 98F3h

	54B 4000h	6.28125e+01
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.10 Инструкция ADDI

Синтаксис:

```
ADDI src, dst
```

Операнды:

src: основные методы адресации

dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

31 24	23 16	15 8	7 0
0 0 0 0 0 0 0 1 0 0	G	dst	src

Операция:

dst + src → dst

Поле слова инструкции:

G	методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Описание:
Сумма операндов dst и src загружается в регистр dst. Операнды dst и src являются целыми со знаком.
Статусные биты:
Если ST(SET COND)=0, флаги состояния изменяются, только если регистр назначения – один из (R0-R11). Если ST(SET COND)=1, они изменяются для всех регистров назначения.

LUF – не изменяется.
LV = 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF = 0
N = 1 при генерации отрицательного результата, иначе = 0.
Z = 1 при генерации нулевого результата, иначе = 0.
V = 1 при возникновении целочисленного переполнения, иначе = 0.
C = 1 при возникновении переноса, иначе = 0.

Бит режима:
Операция OVM зависит от значения бита OVM.
Циклы:
Нет
Пример:
ADDI R3, R7

Перед инструкцией

R3	0FFFF FFCBh	-53
R7	35h	53
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

R3	0FFFF FFCBh	-53
R7	0h	
LUF	0	
LV	0	
UF	0	
N	0	
Z	1	
V	0	
C	0	

A.5.11 Инструкция ADDI3

Синтаксис:
ADDI3 <src2>, <src1>, <dst>
Операнды:
src1, src2: тип 1 или тип 2 трехоперандного метода адресации
dst: режим регистра (любой регистр из основного регистрового файла ЦПУ)
Код:
Тип 1

31	24 23		16	15	8	7	0
0 0 1	0 0 0 0 1 0	T	dst	src1		src2	

Тип 2

31	24 23		16	15	8	7	0
0 0 1	1 0 0 0 1 0	T	dst	src1		src2	

Поля слова инструкции:

Тип 1

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	Регистровая (любой регистр ЦПУ)
01	Косвенная (смещение=0, 1, IR0, IR1)	Регистровая (любой регистр ЦПУ)
10	Регистровая (любой регистр ЦПУ)	Косвенная (смещение=0, 1, IR0, IR1)
11	Косвенная (смещение=0, 1, IR0, IR1)	Косвенная (смещение=0, 1, IR0, IR1)

Тип 2

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	8-битное знаковое прямое
01	Регистровая (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-битное знаковое прямое
11	Косвенная *+ARn1 (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:

src1 + src2 → dst

Описание:

Сумма операндов src1 и src2 загружается в регистр dst. Операнды src1, src2 и dst являются целыми со знаком.

Статусные биты:

Если ST(SET COND)=0, флаги состояния изменяются, только если регистр назначения – один из (R0-R11). Если ST(SET COND)=1, они изменяются для всех регистров назначения.

LUF	– не изменяется.
LV	= 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF	= 0
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении целочисленного переполнения, иначе = 0.
C	= 1 при возникновении переноса, иначе = 0.

Бит режима:

Операция OVM зависит от значения бита OVM.

Циклы:

1

Пример:

Нет

A.5.12 Инструкция ADDI3 || STI

Синтаксис:

ADDI3 src2, src1, dst1

|| STI src3, dst2

Операнды:

src1: регистровая адресация (R0-R7)

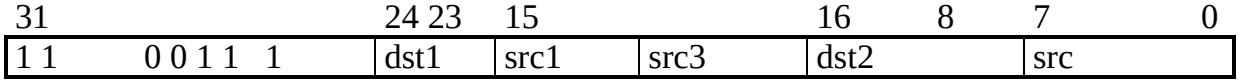
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)

dst1: регистровая адресация (R0-R7)

src3: регистровая адресация (R0-R7)

dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)

Код:



Поле слова инструкции:

Нет

Операция:

src1 + src2 → dst1
|| src3 → dst2
Описание:
Целочисленное сложение и сохранение целого производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (ADDI3), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено командой ADDI3.
Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.
Статусные биты:

LUF	– не изменяется.
LV	= 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF	= 0
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении целочисленного переполнения, иначе = 0.
C	= 1 при возникновении переноса, иначе = 0.

Бит режима:
Операция OVM зависит от значения бита OVM.
Циклы:
1

Пример:

```
ADDI3       *AR0--(IR0), R5, R0
|| STI  R3,*AR7
```

Перед инструкцией

AR0	80 992Ch	
IR0	0Ch	
R5	0DCh	220
R0	0h	
R3	35h	53
AR7	80 983Bh	

Данные в 80 992Ch

	12Ch	300

Данные в 80 983Bh

	0h	
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR0	80 9920h	
IR0	0Ch	
R5	0DCh	220
R0	208h	520
R3	35h	53
AR7	80 983Bh	

Данные в 80 992Ch

	12Ch	300

Данные в 80 983Bh

	35h	53
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.13 Инструкция AND

Синтаксис:
AND src, dst

Операнды:
src: основные методы адресации
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

31 24	23 16	15 8	7 0
0 0 0 0 0 0 1 0 1	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
dst AND src $\rightarrow$ dst

Описание:
Результат поразрядного логического «И» между операндами dst и src записывается в регистр dst. Предполагается, что src и dst являются целыми без знака.

Статусные биты:
Если ST(SET COND)=0, флаги состояния изменяются, только если регистр назначения – один из (R0-R11). Если ST(SET COND)=1, они изменяются для всех регистров назначения.

LUF	– не изменяется.
LV	– не изменяется.
UF	= 0
N	– старший значащий разряд выходного значения.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 0
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.

Циклы:
1

Пример:
AND R1, R2

Перед инструкцией		После инструкции	
R1	80h	R1	80h
R2	0AFFh	R2	80h
LUF	0	LUF	0
LV	0	LV	0
UF	0	UF	0
N	0	N	0
Z	0	Z	0
V	0	V	0
C	1	C	1

A.5.14 Инструкция AND3

Синтаксис:
AND3 src2, src1, dst

Операнды:
src1, src2: тип 1 или тип 2 трехоперандного метода адресации
dst: режим регистра (любой регистр из основного регистрового файла ЦПУ)
Код:
Тип 1

31 24	23	16	15 8	7 0
0 0 1 0 0 0 0 0 1 0	T	dst	src1	src2

Тип 2

31 24	23	16	15 8	7 0
0 0 1 1 0 0 0 1 0	T	dst	src1	src2

Поля слова инструкции:
Тип 1

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	Регистровая (любой регистр ЦПУ)
01	Косвенная (смещение=0, 1, IR0, IR1)	Регистровая (любой регистр ЦПУ)
10	Регистровая (любой регистр ЦПУ)	Косвенная (смещение=0, 1, IR0, IR1)
11	Косвенная (смещение=0, 1, IR0, IR1)	Косвенная (смещение=0, 1, IR0, IR1)

Тип 2

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	3-битное знаковое прямое
01	Регистровая (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	3-битное знаковое прямое
11	Косвенная *+ARn1 (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:
src1 @src2 → dst

Описание:
Результат поразрядного логического «И» операндов src1 и src2 загружается в регистр dst.

Статусные биты:
Если ST(SET COND)=0, флаги состояния изменяются, только если регистр назначения – один из (R0-R11). Если ST(SET COND)=1, они изменяются для всех регистров назначения.

LUF – не изменяется.
LV – не изменяется.
UF = 0
N – старший значащий разряд выходного значения.
Z = 1 при генерации нулевого результата, иначе = 0.
V = 0
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.

Циклы:
1

Пример:
Различие между AND и AND3 в этих примерах:

```
AND3 80h, R0, R0  R0=FFFF FFFFh    R0=FFFF FF80h
AND  80h, R0      R0=FFFF FFFFh    R0=0000 0080h
```

A.5.15 Инструкция AND3 || STI

Синтаксис:
AND src2, src1, dst1
||STI src3, dst2

Операнды:

src1: регистровая адресация (R0-R7)
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst: регистровая адресация (R0-R7)
src3: регистровая адресация (R0-R7)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)

Код:

31	24 23	15		16 8	7 0
1 1 0 1 0 0 0	dst1	src1	src3	dst2	src2

Поле слова инструкции:
Нет

Операция:

src1 AND src2 → dst1
|| src3 → dst2

Описание:

Поразрядное логическое «И» и сохранение целого параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (AND3), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено командой AND3.
Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы:

1

Статусные биты:

LUF	– не изменяется.
LV	– не изменяется.
UF	= 0
N	– старший значащий разряд выходного результата.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 0
C	– не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Пример:

AND3 *+AR1(IR0), R4, R7
|| STI R3,*AR2

Перед инструкцией

AR1	80 99F1h	
IR0	8h	
R4	0DCh	
R7	0h	
R3	35h	53
AR2	80 983Fh	

Данные в 80 99F9h

	5C53h

Данные в 80 983Fh

	0h	
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	

После инструкции

AR1	80 99F1h	
IR0	8h	
R4	0A323h	
R7	03h	
R3	35h	53
AR2	80 983Fh	

Данные в 80 99F9h

	5C53h

Данные в 80 983Fh

	35h	53
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	

C	0	C	0

A.5.16 Инструкция ANDN

Синтаксис:
ANDN src, dst

Операнды:

src: основные методы адресации
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

31 24	23 16	15 8	7 0
0 0 0 0 0 0 1 1 0	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:

Поразрядное логическое «И» между операндом dst и поразрядным логическим дополнением (~) операнда src загружается в регистр dst. Предполагается, что операнды dst и src являются беззнаковыми целыми.

Статусные биты:

Если ST(SET COND)=0, флаги состояния изменяются, только если регистр назначения – один из (R0-R11). Если ST(SET COND)=1, они изменяются для всех регистров назначения.

LUF	– не изменяется.
LV	– не изменяется.
UF	= 0
N	– старший значащий разряд выходного результата.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 0
C	– не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Циклы:

1

Пример:

ANDN @980Ch, R2

Перед инструкцией

DP	80h
R2	0C2Fh

Данные в 80 980Ch

	0A02h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

DP	80h	
R2	042Dh	1.41695313e+02

Данные в 80 980Ch

	0A02h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

A.5.17 Инструкция ANDN3

Синтаксис:
ANDN3 src2, src1, dst
Операнды:
src1, src2: тип 1 или тип 2 трехоперандного метода адресации
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:
Тип 1

31	24 23		16	15	8	7	0
0 0 1	0 0 0 1 0 0	T	dst	src1		src2	

Тип 2

31	24 23		16	15	8	7	0
0 0 1	1 0 0 1 0 0	T	dst	src1		src2	

Поля слова инструкции:
Тип 1

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	Регистровая (любой регистр ЦПУ)
01	Косвенная (смещение=0, 1, IR0, IR1)	Регистровая (любой регистр ЦПУ)
10	Регистровая (любой регистр ЦПУ)	Косвенная (смещение=0, 1, IR0, IR1)
11	Косвенная (смещение=0, 1, IR0, IR1)	Косвенная (смещение=0, 1, IR0, IR1)

Тип 2

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	8-битное знаковое прямое
01	Регистровая (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn(5-битное беззнаковое смещение)	8-битное знаковое прямое
11	Косвенная *+ARn1(5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:
src1 AND ~src2 → dst
Описание:
Поразрядное логическое «И» между операндом src1 и поразрядным логическим дополнением (~) операнда src загружается в регистр dst. Предполагается, что операнды dst, src1 и src2 являются беззнаковыми целыми.
Статусные биты:

LUF	– не изменяется.
LV	– не изменяется.
UF	= 0
N	– старший значащий разряд выходного результата.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 0
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:
Нет

A.5.18 ASH инструкция

Синтаксис:
ASH src_count, dst
Операнды:
src_count: основные методы адресации
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23 16	15 8	7 0
0 0 0 0 0 0 1 1 1	G dst	src_count	

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
Если count ≥0, то
dst << count → dst
В противном случае:
dst >> |count| → dst
Описание:
Младшие восемь значащих разрядов оператора count используются для генерации двоичного дополнения счетчика сдвига до 32. Если операнд count больше нуля, операнд dst сдвигается влево на величину операнда count. При сдвиге младшие разряды заполняются нулями, а старшие сдвигаются вовне через разряд переноса C.
Арифметический левый сдвиг:

$$C \leftarrow dst \leftarrow 0$$

Если операнд count меньше 0, операнд dst сдвигается вправо на абсолютное значение операнда count. Старшие разряды числа сдвигаются с расширением знака вправо. Младшие разряды сдвигаются вовне через разряд переноса C.
Арифметический сдвиг право:

$$Знак\ dst \rightarrow dst \rightarrow C$$

Если операнд count равен 0, сдвиг не происходит, и разряд C (перенос) устанавливается в 0. Операнды count и dst являются целыми со знаком.
Статусные биты:
Если ST(SET COND)=0, флаги состояния изменяются, только если регистр назначения – один из (R0-R11). Если ST(SET COND)=1, они изменяются для всех регистров назначения.

LUF	– не изменяется.
LV	= 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF	= 0
N	– старший значащий разряд выходного значения.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении целочисленного переполнения, иначе = 0.
C	– устанавливается по значению последнего сдвигаемого вовне разряда. Равен «0», если счетчик сдвига 0.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример 1:

ASH R1, R3

Перед инструкцией

R1	10h	16
R3	0A E000h	
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

R1	10h
R3	0E000 0000h
LUF	0
LV	1
UF	0
N	1
Z	0
V	1
C	0

Пример 2:

ASH @98C3h, R5

Перед инструкцией

DP	80h	16
R5	0AEC0 0001h	

Данные в 80 98C3h

	0FFE8h	-24
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

DP	80h
R5	0FFFF FFAEh

Данные в 80 98C3h

	0FFE8h	-24
LUF	0	
LV	0	
UF	0	
N	1	
Z	0	
V	0	
C	1	

A.5.19 Инструкция ASH3

Синтаксис:

ASH3 src_count, src, dst

Операнды:

src, src_count: тип 1 или тип 2 трехоперандного метода адресации

dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

Тип 1

31 24	23	16	15 8	7 0
0 0 1 0 0 0 1 0 1	T	dst	src	src_count

Тип 2

31 24	23	16	15 8	7 0
0 0 1 1 0 0 1 0 1	T	dst	src	src_count

Поля слова инструкции:

Тип 1

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	Регистровая (любой регистр ЦПУ)
01	Косвенная (смещение=0, 1, IR0, IR1)	Регистровая (любой регистр ЦПУ)
10	Регистровая (любой регистр ЦПУ)	Косвенная (смещение=0, 1, IR0, IR1)
11	Косвенная (смещение=0, 1, IR0, IR1)	Косвенная (смещение=0, 1, IR0, IR1)

Тип 2

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	8-битное знаковое прямое
01	Регистровая (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-битное знаковое прямое
11	Косвенная *+ARn1 (5-битное беззнако-	Косвенная *+ARn2 (5-битное беззнако-

вое смещение) вое смещение)

Операция:

Если count ≥ 0, то

src << count → dst

В противном случае:

src >> |count| → dst

Описание:

Семь младших значащих разрядов операнда count используются для генерации двоичного дополнения счетчика сдвига до 32 разрядов.

Если операнд count больше 0, операнд src сдвигается влево на число разрядов, определенных операндом count. Младшие разряды при сдвиге заполняются нулями, старшие сдвигаются вовне через разряд переноса регистра состояния C.

Арифметический левый сдвиг:

C ← src2 ← 0

Если операнд count меньше 0, операнд src сдвигается вправо на число разрядов, определенных абсолютным значением операнда count. Старшие разряды операнда src сдвигаются вправо с расширением знака. Младшие разряды сдвигаются вовне через разряд переноса C регистра состояния.

Арифметический сдвиг вправо:

знак dst → src2 → C

Если операнд src_count равен 0, сдвиг не происходит, и разряд C (перенос) устанавливается в 0. Операнды src_count, src и dst являются целыми со знаком.

Статусные биты:

LUF	– не изменяется.
LV	= 1 при возникновении целочисленного переполнения, иначе не изменяется.
UF	= 0
N	– старший значащий разряд выходного результата
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении целочисленного переполнения, иначе = 0.
C	– устанавливается по значению последнего сдвигаемого вовне разряда. Равен «0», если count = 0.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Циклы:

1

Пример:

Нет

A.5.20 Инструкция ASH3 || STI

Синтаксис:

ASH3 src_count, src2, dst1

|| STI src3, dst2

Операнды:

src_count: регистровая адресация (R0-R7)

src2: косвенная адресация (смещение = 0, 1, IR0, IR1)

dst1: регистровая адресация (R0-R7)

src3: регистровая адресация (R0-R7)

dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)

Код:

31 24	23 16	15 8	7 0
0 0 0 0 0 1 0 0 1	dst1	src_count src3	dst2 src2

Поле слова инструкции:

Нет

Операция:

Count = 7 (семи) наименее значащим битам src_count
Если count ≥ 0, то
src2 << count → dst1
В противном случае:
src2 >> |count| → dst1
|| src3 → dst2

Описание:

Семь младших значащих разрядов операнда count используются для генерации двоичного дополнения счетчика сдвига до 32 разрядов.
Если операнд count больше 0, то операнд src сдвигается влево на число разрядов, определенных операндом count. Младшие разряды при сдвиге заполняются нулями, старшие сдвигаются вовне через бит переноса регистра состояния C.
Арифметический левый сдвиг:

C ← src2 ← 0

Если операнд count меньше 0, то операнд src сдвигается вправо на число разрядов, определенных абсолютным значением операнда count. Старшие разряды операнда src сдвигаются вправо с расширением знака.
Младшие разряды сдвигаются вовне через бит переноса C регистра состояния.
Арифметический сдвиг вправо:

Знак src2 -> src2 -> C

Если операнд count равен 0, то сдвиг не происходит, и бит C (перенос) устанавливается в 0. Операнды count и dst являются целыми со знаком.
Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (ASH3), записывает в тот же регистр, то STI имеет на входе содержимое регистра до того, как оно было изменено командой ASH3.
Если src2 и dst2 указывают на один и тот же адрес, то src2 считывается до записи в dst2.
Статусные биты:

LUF	– не изменяется.
LV	= 1 при возникновении целочисленного переполнения, иначе не изменяется.
UF	= 0
N	– старший значащий разряд выходного результата.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении целочисленного переполнения, иначе = 0.
C	– устанавливается по значению последнего сдвигаемого вовне разряда. Равен «0», если count = 0.

Бит режима:

Операция OVM зависит от значения бита OVM.

Циклы:

1

Пример:

```
ASH3 R1,*AR6++(IR1), R0
|| STI  R5,*AR2
```

Перед инструкцией

AR6	80 9900h	
IR1	8Ch	
R1	0FFE8h	-24
R0	0h	
R5	35h	53
AR2	80 98A2h	

Данные в 80 9900h

0AE00 0000h

Данные в 80 98A2h

0h

После инструкции

AR6	80 998Ch	
IR1	8Ch	
R1	0FFE8h	-24
R0	0FFFF FFAEh	
R5	35h	53
AR2	80 98A2h	

Данные в 80 9900h

0AE00 0000h

Данные в 80 98A2h

35h	53

LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

LUF	0
LV	0
UF	0
N	1
Z	0
V	0
C	0

A.5.21 Инструкция Bcond

Синтаксис:

Bcond src

Операнды:

src: методы адресации условного перехода

Код:

31		24 23		16	15 8	7 0
0 1 1 0 1 0	B	0 0 0	0	cond	регистр или смещение	

Поле слова инструкции:

B	Методы адресации src
0	Регистровая
1	Относительно PC

Операция:

Если cond – истина, то если src с регистровым методом адресации (любой регистр из основного регистрового файла ЦПУ), то

src $\rightarrow$ PC,

иначе если src с методом адресации относительно PC (метка или адрес), то

смещение + PC + 1 $\rightarrow$ PC.

В противном случае продолжение.

Описание:

Bcond означает стандартный переход, который выполняется за четыре цикла. Переход осуществляется, если условие истинно, т. к. конвейер при этом должен быть очищен, смотри подраздел 8.2 «Конфликты конвейера». Если операнд src установлен в регистровый метод адресации, то содержимое указанного регистра загружается в PC. Если операнд src установлен в метод адресации относительно PC, то Ассемблер генерирует смещение; смещение = метка - (PC команды перехода + 1). Это смещение сохраняется как 16-разрядное целое со знаком в 16 младших значащих разрядах слова команды перехода. Смещение добавляется к PC команды перехода + 1 для генерации нового PC.

Микросхема 1867ВМ9Ф поддерживает 20 кодов условий, которые могут использоваться этой инструкцией.

Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Циклы:

4 (несмотря на то, произошел переход или нет)

Пример:

BZ R0

Перед инструкцией

PC	2B00h
R0	0003 FF00h
LUF	0
LV	0
UF	0
N	0
Z	1
V	0
C	0

16

После инструкции

PC	3FF00h
R0	0003 FF00h
LUF	0
LV	0
UF	0
N	0
Z	1
V	0
C	0

Примечание – Если инструкция BZ вызвана непосредственно после инструкции RND с нулевым операндом, то ветвление не произойдет, потому что нулевой флаг не установлен. Для разрешения этой проблемы необходимо использовать инструкцию BZYF вместо BZ.

A.5.22 Инструкция BcondAF

Синтаксис:

BcondAF src

Операнды:

src: методы адресации условного перехода

Код:

31		24 23	16	15 8	7 0
0 1 1 0 1 0	B	0 1 0 1	cond	регистр или смещение	

Поле слова инструкции:

B	Методы адресации src
0	Регистровая
1	Относительно PC

Операция:

Если cond – истина, то если src с регистровым методом адресации (любой регистр из основного регистрового файла ЦПУ), то

src → PC,

иначе если src с методом адресации относительно PC (метка или адрес), то

смещение + PC перехода + 3 → PC

В противном случае, если

cond – ложь,

то аннулируется результат фазы выполнения первой следующей инструкции и результат фазы чтения и фазы выполнения второй и третьей последующих инструкций и продолжение выполнение.

Описание:

Если условие истинно, то инструкция BcondAF выполняет три инструкции, следующие за переходом, и потом выполняет переход. Если условие ложно, то переход не происходит, отменяются фаза выполнения первой инструкции, следующей за переходом, а также фазы чтения и фазы выполнения следующих второй и третьей инструкций. Три следующих за BcondAF инструкции не влияют на cond. Если src регистр, то содержимое указанного регистра загружается в PC. Если src операнд в относительном PC-методе адресации, то сумма PC инструкции перехода + 3 заносится в PC. В относительном PC-методе поле смещения интерпретируется как 16-разрядное знаковое целое.

Ни одна из трех инструкций, следующих за BcondAF, не должна изменять ход программы. В течение выполнения BcondAF прерывания отключены.

Инструкция BcondAF особенно полезна для управления выходом в конце из цикла. Следует быть осторожным, когда используется PUSH/POP, LDPK, LDA, которые могут изменить регистры ARn, SP, DP в фазе декодирования и/или чтения. Это условие также применимо при использовании инструкций для выполнения косвенной адресации с изменением ARn.

Статусные биты:

LUF	– не изменяется.
LV	– не изменяется.
UF	– не изменяется.
N	– не изменяется.
Z	– не изменяется.
V	– не изменяется.
C	– не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Циклы:

1

Пример:

Нет

A.5.23 Инструкция BcondAT

Синтаксис:

BcondAT src

Операнды:

src: методы адресации условного перехода

Код:

31		24 23	16	15 8	7 0
0 1 1 0 1 0	B	0 0 1 1	cond	регистр или смещение	

Поле слова инструкции:

B	Методы адресации src
0	Регистровая
1	Относительно PC

Операция:

Если cond – истина, то если src в регистровом методе адресации (любой регистр из основного регистрового файла ЦПУ), то

src → PC,

иначе аннулирование результата фазы выполнения первой следующей инструкции и результата фазы чтения и фазы выполнения второй и третьей последующих инструкций и продолжение выполнения, иначе если src в методе адресации относительно PC (метка или адрес), то

смещение + PC перехода + 3 → PC и

аннулирование результата фазы выполнения первой следующей инструкции и результата фазы чтения и фазы выполнения второй и третьей последующих инструкций и продолжение выполнения, в противном случае продолжение выполнения.

Описание:

Если условие истинно, то происходит переход, отменяются фаза выполнения первой инструкции, следующей за переходом, а также фазы чтения и выполнения следующих второй и третьей инструкций. Три следующих за BcondAT инструкции не влияют на cond. Если src операнд в регистровом методе, то содержимое указанного регистра заносится в PC. Если src операнд в относительном PC-методе адресации, то сумма PC инструкции перехода + 3 заносится в PC. В относительном PC-методе поле смещения интерпретируется как 16-разрядное знаковое целое.

Ни одна из трех инструкций, следующих за BcondAT не должна изменять линейность программы. В течение выполнения BcondAT прерывания отключены.
Инструкция BcondAT не аннулирует сигналы состояния внешнего интерфейса. Будьте осторожны, когда используете PUSH/POP, LDPK, LDA, которые могут изменить регистры ARn, SP, DP в фазе декодирования и/или чтения. Это условие также применимо при использовании инструкций для выполнения косвенной адресации с изменением ARn. BcondAT полезно использовать для управления входом в начале цикла.
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:

Нет

A.5.24 Инструкция BcondD

Синтаксис:
BcondD src
Операнды:
src: методы адресации условного перехода (B)
Код:

31		24 23		16	15 8	7 0
0 1 1 0 1 0	B	0 0 0	1	cond	src (регистр или смещение)	

Поле слова инструкции:

B	Методы адресации src
0	Регистровая
1	Относительно PC

Операция:
Если cond = истина и если src с регистровым методом адресации (любой регистр из основного регистрового файла ЦПУ), то

src → PC (Program Counter).

Если src с методом адресации относительно PC (метка или адрес), то

смещение + PC + 3 → PC

В противном случае продолжить.
Описание:
BcondD означает задержанный переход, при котором выполняются три инструкции до изменения PC. В результате получается одноцикловый переход, и три инструкции, следующие за BcondD, не изменяют cond.
Три инструкции, следующие за BcondD, не должны менять линейность выполнения программы. Прерывания отключены на все время выполнения BcondD.
Переход осуществляется, если условие истинно. Если операнд src установлен в режим регистровой адресации, то содержимое указанного регистра загружается в PC. Если операнд src установлен в режим адресации относительно PC, то Ассемблер генерирует смещение; смещение = метка - (Program инструкции перехода + 3). Это смещение сохраняется как 16-разрядное целое со знаком в 16 младших значащих разрядах слова инструкции перехода. Смещение добавляется к Program Counter инструкции перехода + 3 для генерации нового PC.

ИС 1867ВМ9Ф имеет 20 кодов условий, которые могут быть использованы с этой инструкцией, смотри раздел А.2, где приведен список мнемоник кодов условий, описание кодов условий и устанавливаемые флаги.
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Циклы:
1
Бит режима:
Операция OVM не зависит от значения бита OVM.

Пример:

BNZD 36 (36 = 24h)

Перед инструкцией

PC	50h
R0	0
LUF	0
LV	0
UF	0
N	0
Z	1
V	0
C	0

16

После инструкции

PC	77h
R0	0
LUF	0
LV	0
UF	0
N	0
Z	1
V	0
C	0

А.5.25 Инструкция BR

Синтаксис:
BR src
Операнды:
src: режим относительной PC-адресации
Код:

31 24	23 16	15 8	7 0
0 1 1 0 0 0 0	0	src (смещение)	

Поле слова инструкции:
Нет
Операция:
PC + 1 + смещение → PC
Описание:
Обеспечивает безусловный переход. Ассемблер генерирует смещение:
смещение = src - (PC инструкции ветвления +1). Это смещение является 24-разрядным целым в 24 младших битах инструкции ветвления. Это смещение добавляется к PC инструкции плюс 1 для генерации нового PC.

Примечание – Для стандартного перехода разряд 24 равен «0».

Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Циклы:

4

Пример:

Нет

A.5.26 Инструкция BRD

Синтаксис:

BRD src

Операнды:

src: режим относительной PC адресации

Код:

31 24	23 16	15 8	7 0
0 1 1 0 0 0 0	1	src (смещение)	

Поле слова инструкции:

Нет

Операция:

PC + 3 + смещение → PC

Описание:

Обеспечивает безусловный задержанный переход. Ассемблер генерирует смещение: смещение = src - (PC инструкции ветвления +3). Это смещение является 24-разрядным целым в 24 наименее значащих битах инструкции ветвления. Это смещение добавляется к PC инструкции плюс 3 для генерации нового PC. Три инструкции, следующие за BRD, выполняются. Никакие из этих инструкций не могут изменить выполнение программы, т. е. повлиять на значение PC.

Примечание – Для задержанного перехода разряд 24 равен 1.

Циклы:

1

Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Пример:

Нет

A.5.27 Инструкция CALL

Синтаксис:

CALL src

Операнды:

src: режим относительной PC адресации

Код:

31 24	23 16	15 8	7 0
0 1 1 0 0 0 1	0	src (смещение)	

Поле слова инструкции:
Нет
Операция:
Следующий PC → *(++SP)
PC + 1 + смещение → PC
Описание:
Осуществляется вызов подпрограммы. Следующее значение PC записывается в системный стек. Ассемблер генерирует смещение: смещение = src - (PC инструкции ветвления +1). Операнд src загружается в PC. Это смещение является 24-разрядным целым в 24 наименее значащих битах инструкции ветвления. Это смещение добавляется к PC инструкции смещения, плюс «1» для генерации нового PC.
Статусные биты:

LUF	– не изменяется.
LV	– не изменяется.
UF	– не изменяется.
N	– не изменяется.
Z	– не изменяется.
V	– не изменяется.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
4
Пример:
Нет

A.5.28 Инструкция CALLcond

Синтаксис:
CALLcond src
Операнды:
src: режим адресации условного перехода
Код:

31	24 23	16	15 8	7 0
0 1 1 1 0 0	B	0 0 0 0	cond	src (регистр или смещение)

Поле слова инструкции:

B	Методы адресации src
0	Регистровая
1	Относительно PC

Операция:
Если cond – истина:
Следующий PC → *++SP
Если src с регистровым методом адресации (любой регистровый файл ЦПУ), то
src → PC
Если src в методе адресации относительно PC (метка или адрес), то
смещение + PC + 1 → PC
В противном случае продолжить.
Описание:
Вызов осуществляется, если условие истинно. Если условие истинно, то следующее значение PC загружается в системный стек. Если операнд src установлен в метод регистровой адресации, то содержимое указанного регистра загружается в PC. Если операнд src установлен в метод адресации относительно PC, то Ассемблер генерирует смещение; смещение = метка - (PC команды перехода + 1). Это смещение сохраняется как 16-разрядное целое со знаком в 16 младших значащих разрядах слова инструкции вызова. Смещение добавляется к PC команды перехода + 1 для генерации нового PC.

В микросхеме 1867ВМ9Ф имеется 20 кодов условий, которые могут быть использованы с этой командой, смотри раздел А.2, где приведен список мнемоник кодов условий, описание кодов условий и устанавливаемые флаги.
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
5 (в не зависимости от того, что какое-либо из условий истинно или нет)
Пример:
CALLNZ R5

Перед инструкцией

PC	123h
SP	80 9865h
R5	789h

LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

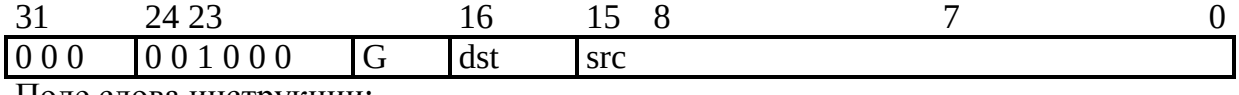
AR1	789h
SP	80 9836h
R5	789h

Данные в 9836h

	124h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

А.5.29 Инструкция CMPF

Синтаксис:
CMPF src, dst
Операнды:
src: основные методы адресации
dst: регистровая адресация (R0-R11)
Код:



Поле слова инструкции:

G	Методы адресации src
00	Регистровая (R0-R11)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
dst - src
Описание:
Операнд src вычитается из операнда dst. Результат не заносится в какой-либо регистр, обеспечивая, таким образом, возможность сравнения без изменения значения каких бы то ни было операндов. Предполагается, что операнды dst и src являются числами в формате с ПЗ.

Статусные биты:

LUF	= 1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	= 1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF	= 1 при потере значимости результата с ПЗ, иначе = 0.
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении ПЗ-переполнения, иначе = 0.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM

Циклы:
1

Пример:

CMPF *+AR4, R6

Перед инструкцией

AR4	80 98F2h	
R6	070C80 0000h	1.4050e+02
Данные в 80 98F3h		
	070C 8000h	1.4050e+02
LUF	0	
LV	0	
UF	0	
N	1	
Z	0	
V	0	
C	0	

После инструкции

AR4	80 98AFh	
R6	0AFh	1.4050e+02
Данные в 80 98F3h		
	070C 8000h	1.4050e+02
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.30 Инструкция CMPF3

Синтаксис:
CMPF3 src2, src1

Операнды:
src1 - src2: тип 1 или тип 2 трехоперандного адресного режима

Код:

Тип 1

31	24 23		16	15	8	7	0
0 0 1	0 0 0 1 1 0	T	0 0 0 0 0	src1		src2	

Тип 2

31	24 23		16	15	8	7	0
0 0 1	1 0 0 1 1 0	T	0 0 0 0 0	src1		src2	

Поля слова инструкции:
Тип 1

T	Методы адресации src1	Методы адресации src2
00	Регистровая (R0-R11)	Регистровая (R0-R11)
01	Косвенная (смещение=0, 1, IR0, IR1)	Регистровая (R0-R11)
10	Регистровая (R0-R11)	Косвенная (смещение=0, 1, IR0, IR1)
11	Косвенная (смещение=0, 1, IR0, IR1)	Косвенная (смещение=0, 1, IR0, IR1)

Тип 2

T	Методы адресации src1	Методы адресации src2
01	Регистровая (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
11	Косвенная *+ARn1 (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:
src1 - src2
Описание:
Операнд src2 вычитается из операнда src1. Результат никуда не заносится, обеспечивая возможность сравнения без изменения значения каких бы то ни было операндов и предполагается, что операнды dst и src являются числами в формате с ПЗ.
Циклы:
1
Статусные биты:

LUF	= 1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	= 1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF	= 1 при потере значимости результата с ПЗ, иначе = 0.
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении ПЗ-переполнения, иначе = 0.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.

Пример:
CMPF3 *AR2,*AR3--(1)

Перед инструкцией

AR2	80 9831h	
AR3	80 9852h	
Данные в 80 9831h		
	58B 4000h	2.5044e+02
Данные в 80 9852h		
	57A2000h	6.253125e+01
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR2	80 9831h	
AR3	809851h (decrement)	
Данные в 80 9831h		
	58B 4000h	2.5044e+02
Данные в 80 9852h		
	733 C000h	6.253125e+01
LUF	0	
LV	0	
UF	0	
N	1	
Z	0	
V	0	
C	0	

A.5.31 Инструкция CMPI
Синтаксис:
CMPI src, dst
Операнды:
src: основные методы адресации
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

31 23	24 16	15 8	7 0
0 0 0 0 0 1 0 0 1	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
dst - src
Описание:
Операнд src вычитается из операнда dst. Результат никуда не заносится, обеспечивая, таким образом, возможность сравнения без изменения значения каких бы то ни было операндов. Предполагается, что операнды dst и src являются целыми без знака.
Статусные биты:

LUF – не изменяется.
LV = 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF = 0
N = 1 при генерации отрицательного результата, иначе = 0.
Z = 1 при генерации нулевого результата, иначе = 0.
V = 1 при возникновении целочисленного переполнения, иначе = 0.
C = 1 при возникновении заема, иначе = 0.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:
CMPI R3, R7

Перед инструкцией

R3	898h	2200
R7	3E8h	1000
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

R3	80 98AFh	2200
R7	0AFh	1000
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.32 Инструкция CMPI3
Синтаксис:
CMPI3 src2, src1
Операнды:
src1- src2: тип 1 или тип 2 трехоперандного адресного режима
Код:
Тип 1

31 24	23	16	15 8	7 0
0 0 1 0 0 0 1 1 1	T	00000	src1	src2

Тип 2

31 24	23	16	15 8	7 0
0 0 1 1 0 0 1 1 1	T	00000	src1	src2

Поля слова инструкции:

Тип 1

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	Регистровая (любой регистр ЦПУ)
01	Косвенная (смещение=0, 1, IR0, IR1)	Регистровая (любой регистр ЦПУ)
10	Регистровая (любой регистр ЦПУ)	Косвенная (смещение=0, 1, IR0, IR1)
11	Косвенная (смещение=0, 1, IR0, IR1)	Косвенная (смещение=0, 1, IR0, IR1)

Тип 2

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	8-разрядная знаковая непосредственная
01	Регистровая (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-разрядная знаковая непосредственная
11	Косвенная *+ARn1 (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:

src1 - src2

Описание:

Операнд src2 вычитается из операнда src1. Результат никуда не заносится, обеспечивая, таким образом, возможность сравнения без изменения значения каких бы то ни было операндов. Предполагается, что операнды dst и src являются целыми со знаком. Хотя инструкция имеет только два операнда, но она имеет вид трехоперандной, т. к. операнды определены в трехоперандном формате.

Статусные биты:

LUF – не изменяется.
LV = 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF = 0
N = 1 при генерации отрицательного результата, иначе = 0.
Z = 1 при генерации нулевого результата, иначе = 0.
V = 1 при возникновении целочисленного переполнения, иначе = 0.
C = 1 при возникновении заема, иначе = 0.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Циклы:

1

П р и м е р:

Нет

A.5.33 Инструкция DBcond

Синтаксис:

DBcond ARn, src

Операнды:

src: методы адресации условного перехода (B)
ARn: вспомогательный регистр

Код:

31	24	23	16	15 8	7 0
0 1 1 0 1 1	B	ARn	0	cond	регистр или смещение

Поле слова инструкции:

B	Методы адресации src
0	Регистровая
1	Относительно PC

Операция:

ARn - 1 → ARn

Если cond – истина и ARn ≥0, то src в регистровом методе адресации (любой регистр из основного регистрового файла ЦПУ)

src → PC

Если src в методе адресации относительно PC (метка или адрес), то

смещение + PC + 1 → PC

В противном случае продолжить.

Описание:

DBcond означает стандартный переход, который выполняется за четыре цикла, т. к. конвейер должен быть очищен, если cond – "истина". Если условие истинно и определенный вспомогательный регистр больше или равен «0», то определенный вспомогательный регистр уменьшен и происходит ветвление.

Вспомогательный регистр интерпретируется как 32-разрядное знаковое целое. Обратите внимание, что условие перехода не зависит от декремента вспомогательного регистра.

Если операнд src установлен в режим регистровой адресации, то содержимое указанного регистра загружается в PC. Если операнд src установлен в режим адресации относительно PC, то Ассемблер генерирует смещение; смещение = метка - (PC инструкции перехода + 1). Смещение добавляется к PC инструкции перехода + 1 для генерации нового PC.

ИС 1867ВМ9Ф имеется 20 кодов условий, которые могут быть использованы с этой инструкцией, смотри раздел А.2, где приведен список мнемоник кодов условий, описание кодов условий и устанавливаемые флаги.

Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Циклы:

4

Пример:

DBLT AR3, R2

Перед инструкцией

PC	5Fh
AR3	67
R2	9Fh
LUF	0
LV	0
UF	0
N	1
Z	0
V	0
C	0

После инструкции

PC	9Fh
AR3	11h
R2	9Fh
LUF	0
LV	0
UF	0
N	1
Z	0
V	0
C	0

А.5.34 Инструкция DBcondD

Синтаксис:
DBcondD ARn, src

Операнды:

src: методы адресации условного перехода (B)
ARn: вспомогательный регистр

Код:

31 24	23	16	15 8	7 0
0 1 1 0 1 1	B	ARn	1	cond	src (регистр или смещение)

Поле слова инструкции:

B	Методы адресации src
0	Регистровая
1	Относительно PC

Операция:

ARn - 1 → ARn
Если cond – истина и ARn ≥0:
Если src в регистровом методе адресации (Rn: 0 ≤ n ≤ 27),
src → PC
Если src в методе адресации относительно PC (метка или адрес),
смещение + PC + 3 → PC
В противном случае продолжение.

Описание:

DBcondD означает задержанный переход, который выполняется за один цикл. Перед изменением PC выполняются три инструкции. Определенный вспомогательный регистр декрементируется, и выполняется переход, если cond – «истина», и если значение в определенном вспомогательном регистре больше или равно «0». Три инструкции, следующие за DBcondD, не влияют на статусные биты.
Вспомогательный регистр интерпретируется как 32-разрядное знаковое целое. Никакие три инструкции, следующие за DBcondD, не должны менять линейность выполнения программы. Прерывания неактивны для задержанной DBcondD инструкции.
Обратите внимание, что условие перехода не зависит от декремента вспомогательного регистра.
Если операнд src установлен в режим регистровой адресации, содержимое указанного регистра загружается в PC. Если операнд src установлен в режим адресации относительно PC, Ассемблер генерирует смещение; смещение = метка - (PC инструкции перехода + 3). Смещение добавляется к PC инструкции перехода + 3 для генерации нового PC. Заметьте, что разряд 21=1 для задержанного перехода.
В микросхеме 1867ВМ9Ф имеется 20 кодов условий, которые могут быть использованы с этой инструкцией, смотри раздел А.2, где приведен список мнемоник кодов условий, описание кодов условий и устанавливаемые флаги.
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.

Циклы:

1

Пример:
DBZDAR5, $+110h ($ – текущее значение Program Counter)

Перед инструкцией		После инструкции	
PC	0h	PC	110h
AR5	67	AR5	66h

LUF	0
LV	0
UF	0
N	0
Z	1
V	0
C	0

LUF	0
LV	0
UF	0
N	0
Z	1
V	0
C	0

A.5.35 Инструкция FIX

Синтаксис:
FIX src, dst

Операнды:
src: основные методы адресации
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

31 24	23 16	15 8	7 0
0 0 0	0 0 1 0 1 0	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (R0-R11)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
fix(src) → dst

Описание:
Операнд src в формате с ПЗ преобразуется к ближайшему целому, меньшему или равному по значению, и результат записывается в регистр dst. Операнд src имеет значение в формате с ПЗ, операнд dst – целое со знаком.
Поле экспоненты регистра результата (если он один) не изменяется.
Целочисленное переполнение возникает, когда число в формате с ПЗ слишком велико, чтобы быть представлено как 32-разрядное в дополнительном коде. В случае целочисленного переполнения результат будет записан в направлении переполнения.

Статусные биты:

LUF	– не изменяется.
LV	= 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF	= 0
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении целочисленного переполнения, иначе = 0.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.

Циклы:
1

Пример:

FIX R1, R2

Перед инструкцией

R1	0A 2820 0000h	1.3454e+3
R2	0h	
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

R1	0A 2820 0000h	1.3454e+3
R2	541h	1345
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.36 Инструкция FIX || STI

Синтаксис:

FIX src2, dst1

|| STI src3, dst2

Операнды:

src2: косвенная адресация (смещение = 0, 1, IR0, IR1)

dst1: регистровая адресация (R0-R7)

src3: регистровая адресация (R0-R7)

dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)

Код:

31 24	23 16	15 8	7 0
1 1 | 0 1 0 1 0	dst1 | 0 0 0 | src3	dst2	src2

Поле слова инструкции:

Нет

Операция:

fix(src2) → dst1

|| src3 → dst2

Описание:

Преобразование числа в формате с ПЗ в целое. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (FIX), записывает в тот же регистр, то STI имеет на входе содержимое регистра до того, как оно было изменено командой FIX. Если src2 и dst2 указывают на один и тот же адрес, то src2 считывается до записи в dst2.

Целочисленное переполнение возникает, когда число в формате с ПЗ слишком велико, чтобы быть представлено как 32-разрядное в дополнительном коде. В случае целочисленного переполнения результат будет записан в направлении переполнения.

Статусные биты:

LUF – не изменяется.

LV = 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF = 0

N = 1 при генерации отрицательного результата, иначе = 0.

Z = 1 при генерации нулевого результата, иначе = 0.

V = 1 при возникновении целочисленного переполнения, иначе = 0.

C – не изменяется.

Бит режима:

Операция OVM зависит от значения бита OVM.

Циклы:

1

Пример:

```
FIX  *++AR4(1),
|| STI  R0,*AR2
```

Перед инструкцией

AR4	80 98A2h	
R1	0h	66
R0	0DCh	220
AR2	80 983Ch	

Данные в 80 98A3h

	733 C000h	1.79750e+02

Данные в 80 983Ch

	0h	2
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR4	80 98A3h	
IR1	0B3h	179
R4	0	220
AR7	80 98C5h	

Данные в 80 98A3h

	733 C000h	1.79750e+02

Данные в 80 98C4h

	0DCh	220
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.37 Инструкция FLOAT

Синтаксис:

```
FLOAT src, dst
```

Операнды:

src: основные методы адресации (G)

dst: регистровая адресация (R0-R11)

Код:

31 24	23	16	15 8	7 0
0 0 0	0 0 1 0 1 1	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (R0-R11)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:

float(src) → dst

Описание:

Целочисленный операнд src преобразуется в число в формате с ПЗ, величина которого равна величине числа в целочисленном формате и записывается в регистр dst. Операнд src является целым числом со знаком, операнд dst – число в формате с ПЗ.

Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF = 0
N = 1 при генерации отрицательного результата, иначе = 0.
Z = 1 при генерации нулевого результата, иначе = 0.
V = 0
C – не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Циклы:

1

Пример:

FLOAT *++AR2 (2), R5

Перед инструкцией

AR2	80 9800h	
R5	034C 2000h	1.27578125e+01

Данные в 80 98AFh

	58B 4000h	174
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR2	80 98AFh	
R5	072E0 0000h	1.74e+02

Данные в 80 98AFh

	58B 4000h	174
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.38 Инструкция FLOAT || STF

Синтаксис:
FLOAT src2, dst1
|| STF src3, dst2
Операнды:
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst1: регистровая адресация (R0-R7)
src3: регистровая адресация (R0-R7)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)
Код:

31 24	23 16	15 8	7 0
1 1 | 0 1 0 1 1	dst1 | 0 0 0 | src3	dst2	src2

Операция:
float(src2) → dst1
|| src3 → dst2
Описание:
Выполняется преобразование из целого формата в формат с ПЗ. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STF) считывает из регистра, и операция, которая производится параллельно (FLOAT), записывает в тот же регистр, то STF имеет на входе содержимое регистра до того, как оно было изменено командой FLOAT.
Если src2 и dst2 указывают на один и тот же адрес, то src2 считывается до записи в dst2.
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF = 0
N = 1 при генерации отрицательного результата, иначе = 0.
Z = 1 при генерации нулевого результата, иначе = 0.
V = 0
C – не изменяется.

Бит режима:
Операция OVM зависит от значения бита OVM.
Циклы:
1

Пример:

FLOAT *+AR2(IR0), R6
|| STF R7,*AR1
Перед инструкцией После инструкции

AR2	80 98C5h	
IR0	8h	
R6	0h	
R7	733C0 0000h	1.27578125e+01
AR1	80 9933h	

Данные в 80 98CDh

	0AEh	174

Данные в 80 9933h

	0h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

AR3	80 98C5h	
IR1	8h	
R6	07 2E00 0000h	1.740e+02
R7	03 4C20 0000	1.27578125e+01
AR1	80 9933h	

Данные в 80 98CDh

	0AEh	174

Данные в 80 9933h

	034C 2000h	1.79750e+02
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.39 Инструкция FRIEE

Синтаксис:
FRIEE src, dst

Операнды:
src: прямая или косвенная адресация (G)
dst: регистр с повышенной точностью (R0-R11)

Код:

31 24	23 16	15 8	7 0
0 0 0	1 1 1 0 0 0	G	dst	src

Поле слова инструкции:

G	Методы адресации src
01	Прямая
10	Косвенная

Операция:
Преобразование src из IEEE формата → dst

Описание:
Операнд src преобразовывается из IEEE формата с плавающей запятой в расширенный формат с плавающей запятой в дополнительном коде.
Операнд src берется из памяти. Результат преобразования переносится в регистр с повышенной точностью, как число с плавающей запятой одинарной точности.

Статусные биты:

LUF	– не изменяется.
LV	– не изменяется.
UF	= 0
N	– знак результата.
Z	= 1 если результат равен 0, иначе = 0.
V	= 1 при переполнении, иначе = 0.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.

Циклы:
1

Пример:
Нет

A.5.40 Инструкция FRIEEE||STF

Синтаксис:
FRIEEE src2, dst1
|| STF src3, dst2
Операнды:
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst1: регистровая адресация (R0-R7)
src3: регистровая адресация (R0-R7)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)
Код:

31 24	23 16	15 8	7 0
1 1 1 1 0 0 1	dst1 | 0 0 0 | src3	dst2	src2

Операция:
Преобразование src2 из IEEE формата → dst1
параллельно с src3 → dst2
Описание:
Операнд src2 преобразуется из формата с плавающей точкой в формате IEEE в формат в дополнительном коде. Результат преобразования переносится в регистр с повышенной точностью dst1, как число с плавающей запятой одинарной точности.
Параллельно происходит сохранение результата в формате с плавающей точкой.
Если src2 и dst2 находятся в одном месте, то чтение src2 предшествует записи в dst2.
Статусные биты:

LUF	– не изменяется.
LV	– устанавливается, если есть переполнение, иначе не изменяется.
UF	= 0
N	– знак результата.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при переполнении, иначе = 0.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:
Нет

A.5.41 Инструкция IACK

Синтаксис:
IACK src
Операнды:
основные методы адресации src (G)
Код:

31 24	23 16	15 8	7 0
0 0 0 | 1 1 0 1 1 0	G	0 0 0 0 0	src

Поле слова инструкции:

G	Методы адресации src
01	Прямая
10	Косвенная

Операция:
Выполняет фиктивную операцию чтения, устанавливая сигнал IACK# в низкий уровень.
По завершении фиктивного чтения устанавливает IACK# в высокий уровень.
Описание:

Выполняется фиктивная операция чтения с установкой сигнала IACK#. По завершении фиктивного чтения сигнал IACK# устанавливается в «1». Эта инструкция может быть использована для подтверждения внешнего прерывания. Если указанный адрес относится к внекристальной памяти, то осуществляется операция чтения по данному адресу. Сигнал IACK# и адрес могут использоваться для сигнализации подтверждения прерывания внешним устройствам. Чтение данных процессором не используется.

Примечание – Сигнал IACK# расширяется многоцикловым чтением.

Циклы:

1

Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Пример:

IACK *AR5

Перед инструкцией

IACK#	1
PC	300h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

IACK#	1
PC	301h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

6.118750e+01

А.5.42 Инструкция IDLE

Синтаксис:

IDLE

Операнды:

Нет

Код:

31	24 23	16	15 8	7 0
0 0 0	0 0 1 1 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		

Поле слова инструкции:

Нет

Операция:

$1 \rightarrow$ ST(GIE)

Следующий PC $\rightarrow$ PC

Ожидание до прерывания

Описание:

Устанавливается глобальное разрешение прерывания, следующее значение PC загружается в PC и ЦПУ ожидает прерывания. Когда прерывание получено, то содержимое PC загружается в активный системный стек, и процессор переходит к программе обработки прерываний.

Статусные биты:

LUF – не изменяется.
LV – не изменяется.

UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Циклы:

1

Пример:

Нет

A.5.43 Инструкция IDLE2

Синтаксис:

IDLE

Операнды:

Нет

Код:

31 24	23 16	15 8 7 0
0 0 0	0 0 1 1 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Поле слова инструкции:

Нет

Операция:

1 → ST(GIE)
Следующий PC → PC
Ожидание до прерывания

Описание:

Инструкция IDLE2 действует так же как IDLE, за исключением того, что еще она останавливает функциональную синхронизацию внутри ИС. Это позволяет перевести ИС в режим пониженного потребления питания. PC инкрементируется один раз, и процессор остается в состоянии ожидания, пока не произойдет внешнее прерывание от NMI# или IIOFx#.

В режиме IDLE2 микросхема 1867ВМ9Ф ведет себя следующим образом: регистры ЦПУ, переферии и память сохраняют предыдущее состояние. Когда ИС находится в функциональном (не в режиме эмулятора) режиме, то синхронизация останавливается на высоком уровне сигнала H1 и на низком уровне сигнала H3.

ИС 1867ВМ9Ф остается в IDLE2 до тех пор, пока не произойдет одно из внешних прерываний NMI# или IIOFx# в течение, как минимум, последних двух циклов сигнала H1. Затем синхронизация стартует после задержки на 1 цикл сигнала H1. Такты начинают идти в фазе противоположной той, в которой они были остановлены (сигнал H1 может стартовать в высоком уровне, если H3 был перед этим остановлен в высоком уровне и наоборот). Однако сигналы H1 и H3 остаются в противофазе по отношению друг к другу.

В течение режима IDLE2 для обнаружения внешнего прерывания ЦПУ, оно должно произойти в течение последних двух циклов сигнала H1. Для того чтобы процессор распознал только одно прерывание после рестарта операции, сигнал прерывания должен быть сконфигурирован в режим определения по фронту или должен присутствовать не менее трех циклов в режиме его определения по уровню.

Когда ИС 1867ВМ9Ф находится в режиме эмуляции, то сигналы H1 и H3 продолжают работать в нормальном режиме, и ЦПУ работает как и при выполнении инструкции IDLE. Синхронизация продолжается для правильной работы эмулятора.

Любой сигнал внешнего прерывания может вывести устройство из IDLE2, но для того, чтобы ЦПУ определил это прерывание, оно должно быть разрешено. Если преры-

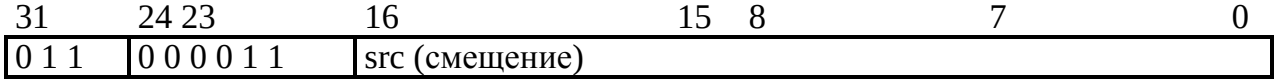
вание определено и обрабатывается ЦПУ, то инструкция, следующая за инструкцией IDLE2, не выполняется до того, пока не произойдет возврата из режима IDLE.
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:
Нет

A.5.44 Инструкция LAJ

Синтаксис:
LAJ src
Операнды:
src: метод адресации относительно PC
Код:



Поле слова инструкции:
Нет
Операция:
PC LAJ+4 → регистр с повышенной точностью R11
смещение + 3 + PC LAJ → PC
Описание:
LAJ выполняет одноцикловый отложенный вызов подпрограммы, что позволяет трем инструкциям, следующим за LAJ инструкцией, быть выполненными перед ветвлением. Возвращенный адрес (адрес LAJ инструкции + 4) размещен в регистре повышенной точности R11. Ассемблер гнерирует смещение: смещение = src - (PC инструкции ветвления + 1). Это смещение сохраняется как 24-разрядное знаковое целое в 24 наименее значащих битах инструкции перехода. Это смещение добавляется к PC инструкции перехода плюс 1 для генерации нового PC. Смотри подраздел 6.6 «Относительная адресация (относительно счетчика инструкций PC)» для получения более детальной информации.
Никакие три инструкции, следующие за LAJ инструкцией, не должны изменять R11 или линейность выполнения программы. Прерывания запрещены в течение LAJ инструкции.
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:

1

Пример:

Нет

A.5.45 Инструкция LAJcond

Синтаксис:

LAJcond src

Операнды:

src: режим адресации условного перехода

Код:

31	24 23	16	15 8	7 0
0 1 1 1 0 0	B	0 0 0 1	cond	src (регистр или смещение)

Поле слова инструкции:

B	Методы адресации src
0	Регистровая
1	Относительно PC

Операция:

Если условие – «истина» и если (src – регистр), то

PC LAJcond + 4 → регистр повышенной точности R11

src → PC

Если src в методе относительной PC адресации

PC LAJcond + 4 → регистр повышенной точности R11

смещение = src – (PC LAJ + 3)

смещение + PC LAJ + 3 → PC

Иначе – продолжение.

Описание:

LAJcond выполняет одноцикловый задержанный вызов подпрограммы по условию, что позволяет трем инструкциям, следующим за LAJcond инструкцией, быть выполненными перед переходом без изменения cond. Адрес возврата (адрес LAJ инструкции + 4) размещен в регистре повышенной точности R11. Адрес перехода формируется с регистровым методом адресации или адресацией относительно PC.

Никакие три инструкции, следующие за LAJcond инструкцией, не должны изменять R11 или менять линейность выполнения программы. Прерывания запрещены в течение LAJcond инструкции.

Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Циклы:
1

Пример:

Нет

A.5.46 Инструкция LATcond

Синтаксис:
LATcond N
Операнды:
N: режим непосредственной адресации по номеру TVT таблицы ($0 \le N \le 511$)
Код:

31 24	23 16	15 8	7 0
0 1 1 1 0 1 0 0 1	0 0 cond	0 0 0 0 0 0 0	N

Поле слова инструкции:
Нет
Операция:
Если условие – «истина», то
ST(GIE) → ST(PGIE)
ST(CF) → ST(PCF)
0 → ST(GIE)
1 → ST(CF)
PC LAcond + 4 → регистр повышенной точности R11
вектор N таблицы TVT → PC
Иначе – продолжение.
Описание:
LATcond выполняет задержанное одноцикловое программное прерывание по условию. Если условие – «истина», то разряды GIE и CF регистра ST сохранены в PGIE и PCF регистра состояния. Все прерывания отключаются (0 → GIE), и кэш «заблокирован» (1 → CF). Содержимое PC LATcond + 4 помещается в R11, и в PC загружается адрес, указанный вектором N из таблицы веторов программных прерываний. Если содержимое – «ложь», то продолжается выполнение программы. Если программные прерывания имеют вложенность, то может возникнуть необходимость сохранить регистр состояния перед выполнением инструкции LATcond N.
Три инструкции, следующие за LATcond, выбираются и выполняются, но они не влияют на cond. Они не должны изменять линейность выполнения программы или непосредственно изменять регистр состояния. Прерывания запрещены в течение инструкции LATcond N.
Статусные биты:

LUF	– не изменяется.
LV	– не изменяется.
UF	– не изменяется.
N	– не изменяется.
Z	– не изменяется.
V	– не изменяется.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:

Нет

A.5.47 Инструкция LBb

Синтаксис:
LBb src, dst
Операнды:
src: регистровая, прямая, 16-разрядная непосредственная или косвенная адресация
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23 16	15 8	7 0
1 0 1 1 0 0 0	B	G	dst	src

Поле слова инструкции:

G	Методы адресации src	B	Методы адресации src2
00	Регистровая	00	Байт 0 младшего байта
01	Прямая	01	Байт 1
10	Косвенная	10	Байт 2
11	Непосредственная (16 бит)	11	Байт 3 старшего байта

Операция:
Знаково-расширенный байт (3, 2, 1, 0) src → dst
b = загружаемый байт (3, 2, 1, 0)

3	2	1	0

= b (разрядный указатель 3-0)

Описание:
Заданный байт src операнда дополняется знаком и смещается вправо на восемь младших разрядов dst регистра. Байт src – знаковый. Когда установлен режим непосредственной адресации и байт 2 (B=10) или байт 3 (B=10) выбраны, то инструкция LBb производит расширение со знаком 16-разрядного значения. Значения 00h или FFh сохраняются в восьми младших разрядах dst регистра.
Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF – не изменяется.
LV – не изменяется.
UF = 0
N = 1, если сгенерирован отрицательный результат, иначе = 0
Z = 1, если сгенерирован нулевой результат, иначе = 0
V = 0
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:
LB2 R1, R2; знаковое расширение байта 2 R1 → R2

Перед инструкцией

R1	00AB 0000h
R2	0000 0000h

После инструкции

R1	00AB 0000h
R2	FFFF FFABh

A.5.48 Инструкция LBUb

Синтаксис:
LBUb src, dst
Операнды:
src: регистровая, прямая, 16-разрядная непосредственная, косвенная адресация
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

31 24	23 16	15 8	7 0
1 0 1 1 0 0 1	B	G	dst	src

Поле слова инструкции:

G	Методы адресации src	B	Методы адресации src2
00	Регистровая	00	Байт 0 LS байта
01	Прямая	01	Байт 1
10	Косвенная	10	Байт 2
11	Непосредственная (16 бит)	11	Байт 3 MS байта

Операция:

Байт (3, 2, 1, 0) src → dst

b = загружаемый байт (3, 2, 1, 0)

3	2	1	0

= b (разрядный указатель 3-0)

Описание:

Заданный байт src операнда смещается вправо на восемь младших разрядов dst регистра без дополнения знаком. Байт src – незнаковый. Когда установлен режим непосредственной адресации и байт 2 (B=10) или байт 3 (B=11) выбраны, то инструкция LBUb производит расширение со знаком 16-разрядного значения. Значения 00h или FFh сохраняются в восьми младших разрядах dst регистра.

Статусные биты:

Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF – не изменяется.
LV – не изменяется.
UF = 0
N = 0
Z = 1, если сгенерирован нулевой результат, иначе = 0.
V = 0
C – не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Циклы:

1

Пример:

LB2 R1, R2

Перед инструкцией

R1	00AB 0000h
R2	0000 0000h

После инструкции

R1	00AB 0000h
R2	0000 00ABh

A.5.49 Инструкция LDA

Синтаксис:

LDA src, dst

Операнды:

src: основные методы адресации (G)
dst: режим регистра (только адресные регистры)

Код:

31 24	23 16	15 7	8 0
0 0 0 1 1 1 1 0 1	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная

11 Непосредственная

Операция:
src → dst
Описание:
Операнд src загружается в регистр dst. Регистр dst может быть любым адресным регистром: AR0-AR7, IR0, IR1, DP, BK или SP. Загрузка завершается в конце фазы чтения конвейера, как результат: LDA – одноцикловая и быстрее, чем LDI для загрузки этих регистров. Все операнды интерпретируются как знаковые целые.

Предупреждение – Внимание: src и dst не могут быть одним и тем же регистром.

Статусные биты:

LUF	– не изменяется.
LV	– не изменяется.
UF	– не изменяется
N	– не изменяется
Z	– не изменяется.
V	– не изменяется
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:
Нет

A.5.50 Инструкция LDE

Синтаксис:
LDE src, dst
Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (R0-R11)

31 24	23 16	15 8	7 0
0 0 0 0 0 1 1 0 1	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (R0-R11)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
src(exp) → dst(exp)
Описание:
Поле экспоненты операнда src загружается в поле экспоненты регистра dst. Поле мантиссы регистра dst не изменяется, если только значение загруженной экспоненты не является зарезервированным для нуля значением экспоненты, что определяется точностью операнда src, то поле мантиссы операнда dst устанавливается в ноль. Операнды src и dst являются числами в формате с ПЗ. Непосредственные значения выражаются в коротком формате с ПЗ.
Статусные биты:

LUF	– не изменяется.
LV	– не изменяется.
UF	– не изменяется.
N	– не изменяется.
Z	– не изменяется.

V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM

Циклы:
1

Пример:

LDE R0, R5

Перед инструкцией

R0	020005 6F30h	4.00066337e+00
R5	0A056F E332h	1.06749648e+03
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

R0	020005 6F30h	4.00066337e+00
R5	02056F E332h	4.16990814e+00
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.51 Инструкция LDEP

Синтаксис:
LDEP src, dst

Операнды:
src: дополнительный регистровый файл IVTP или TVTP
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

31 24	23	16	15 8	7 0
0 1 1 1 0 1 1 0 0	G	dst	0 0 0 0 0 0 0 0 0 0 0 0	src

Поле слова инструкции:
Нет

Операция:
src(exp) → dst(exp)

Описание:
Инструкция LDEP загружает регистр ЦПУ содержимым регистра IVTP (указатель таблицы векторов системных прерываний) или регистра TVTP. Эти регистры описаны в подразделе 3.2 «Дополнительный регистровый файл ЦПУ» КФДЛ.431282.006ТО. Регистровый операнд src дополнительного регистрового файла загружается в dst регистр в основном регистровом файле. Содержимое регистра dst должно быть целым.

Статусные биты:
LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM

Циклы:
1

Пример:
Нет

A.5.52 Инструкция LDF

Синтаксис:
LDF src, dst
Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (R0-R11)
Код:

31 24	23 16	15 8	7 0
0 0 0 0 0 1 1 1 0	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (R0-R11)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
src → dst
Описание:
Операнд src загружается в регистр dst. Операнды src и dst являются числами в формате с плавающей запятой.
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF = 0
N = 1 при генерации отрицательного результата, иначе = 0.
Z = 1 при генерации нулевого результата, иначе = 0.
V = 0
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:
LDF @9800h, R2

Перед инструкцией

DP	80h	
R2	0h	

Данные в 80 9800h

	10C5 2A00h	2.19254303e+00
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

DP	80h	
R2	010C5 2A00h	2.19254303e+00

Данные в 80 9800h

	733 C000h	2.19254303e+00
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.53 Инструкция LDFcond

Синтаксис:
LDFcond src, dst

Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (R0-R11)

Код:

31	23 24		16	15 8	7 0
0 1 0 0	cond	G	dst	src	

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (R0-R11)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
Если cond – "истина":
src → dst.
Иначе:
dst не изменяется

Описание:
Если cond – "истина", то операнд src загружается в регистр dst. В противном случае регистр dst не изменяется. Операнды src и dst являются числами в формате с ПЗ.
В микросхеме 1867ВМ9Ф имеется 20 кодов условий, которые могут быть использованы с этой командой, смотри раздел А.2, где приведен список мнемоник кодов условий, описание кодов условий и устанавливаемые флаги.
Обратите внимание, что команда LDFU (загрузить число в формате ПЗ безусловно) используется для зарузки (R0-R11) без влияния флагов условий.
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.

Циклы:
1

Пример:
LDFZ R3, R5

Перед инструкцией

R3	2CFF2C D500h	1.77055560e+13
R5	5F0000 003Eh	3.96140824e+28
LUF	0	
LV	0	
UF	0	
N	0	
Z	1	
V	0	
C	0	

После инструкции

R3	2CFF2C D500h	1.77055560e+13
R5	2CFF2C D500h	1.77055560e+13
LUF	0	
LV	0	
UF	0	
N	0	
Z	1	
V	0	
C	0	

A.5.54 Инструкция LDFI

Синтаксис:

LDFI src, dst

Операнды:

src: основные методы адресации src (G)

dst: регистровая адресация (R0-R11)

Код:

31 24	23 16	15 8	7 0
0 0 0 0 0 1 1 1 1	G	dst	src

Поле слова инструкции:

G	Методы адресации src
01	Прямая
10	Косвенная

Операция:

Формирует сигнал блокировки.

src → dst

Описание:

Операнд src загружается в регистр dst. Операция блокировки выполняется с помощью сигналов LOCK# или LLOCK#. Операнды src и dst являются числами в формате с ПЗ. Обратите внимение, что допускается только прямая и косвенная адресация. Детальное описание смотри в подразделе 6.4 «Косвенная адресация» КФДЛ.431282.006ТО.

Циклы:

1

Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF = 0
N = 1 при генерации отрицательного результата, иначе = 0.
Z = 1 при генерации нулевого результата, иначе = 0.
V = 0
C – не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

П р и м е р:

LDFI *+AR2, R7

Перед инструкцией

AR2	80 98F1h	
R7	0h	

Данные в 80 98F2h

	584 C000h	-6.28125e+01
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR2	80 98F1h	
R7	0584C0 0000h	-6.28125e+01

Данные в 80 98F2h

	584 C000h	-6.28125e+01
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.55 Инструкция LDF || LDF

Синтаксис:
LDF src2, dst2
|| LDF src1, dst1
Операнды:
src1: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst1: регистровая адресация(R0-R7)
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst2: регистровая адресация (R0-R7)
Код:

31	24 23	16	15 8	7 0
1 1 | 0 0 0 1 0	dst2	dst1	0 0 0 | src1	src2

Поле слова инструкции:
Нет
Операция:
src2 → dst2
|| src1 → dst1
Описание:
Загрузка двух чисел выполняется параллельно. Если обе LDF загружают в один и тот же регистр, то Ассемблер выдает предупреждение об ошибке. В результате этого получается LDF src2, dst2
Статусные биты:

LUF	– не изменяется.
LV	– не изменяется.
UF	– не изменяется.
N	– не изменяется.
Z	– не изменяется.
V	– не изменяется.
C	– не изменяется.

Циклы:
1
Бит режима:
Операция OVM не зависит от значения бита OVM.
Пример:
LDF *--AR1(IR0), R7
|| LDF *AR7++(1), R3

Перед инструкцией

AR1	80 985Fh	
IR0	8h	
R7	0h	
AR7	80 989Ah	
R3	0h	
Данные в 80 9857h		
	70C 4000h	1.4050e+02
Данные в 80 988Ah		
	57B 4000h	6.281250e+01
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR1	80 9857h	
IR0	8h	
R7	070C80 0000h	1.4050e+02
AR7	80 988Bh	
R3	057B40 0000h	6.281250e+01
Данные в 80 9857h		
	70C 4000h	1.4050e+02
Данные в 80 988Ah		
	57B 4000h	6.281250e+01
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.56 Инструкция LDF || STF

Синтаксис:
LDF src2, dst1
|| STF src3, dst2

Операнды:
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst1: регистровая адресация (R0-R7)
src3: регистровая адресация (R0-R7)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)

Код:

31	24	23	16	15	8	7	0
1 1	0 1 1 0 0	dst1	0 0 0	src3	dst2		src2

Поле слова инструкции:
Нет

Операция:
src2 → dst1
|| src3 → dst2

Описание:
Параллельно выполняются операции загрузки и сохранения чисел в формате с ПЗ. Если операнды src2 и dst2 указывают на один и тот же адрес, то src2 считывается до того, как будет записан dst2.

Статусные биты:

LUF	– не изменяется.
LV	– не изменяется.
UF	– не изменяется.
N	– не изменяется.
Z	– не изменяется.
V	– не изменяется.
C	– не изменяется.

Циклы:
1

Бит режима:
Операция OVM не зависит от значения бита OVM.

Пример:

LDF *AR2--(1), R1
|| STF R3,*AR4++(IR1)

Перед инструкцией

AR2	80 98E6h	
R1	0h	
R3	057B40 0000h	6.281250e+01
AR4	80 9900h	
IR1	10h	

Данные в 80 98E7h

	70C 4000h	1.4050e+02

Данные в 80 9900h

	0h	
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR1	80 98E6h	
R1	070C80 0000h	1.4050e+02
R3	057B40 0000h	6.281250e+01
AR4	80 9910h	
IR1	10h	

Данные в 80 9857h

	70C 4000h	1.4050e+02

Данные в 80 988Ah

	57B 4000h	6.281250e+01
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.57 Инструкция LDHI

Синтаксис:
LDHI src, dst
Операнды:
src: 16-разрядная беззнаковая непосредственная адресация
dst: регистровая адресация
Код:

31 24	23	16	15 7	8 0
0 0 0 1 1 1 1 1 1 1	1 1	dst	src (непосредственное значение)	

Поле слова инструкции:
Нет
Операция:
src → 16 наболее значивых битов dst
Описание:
16-битное беззнаковое число src загружается в 16 старших разрядов регистра dst, «0» загружается в 16 младших разрядов. Значение регистра dst является целым.
Статусные биты:

LUF	– не изменяется.
LV	– не изменяется.
UF	– не изменяется.
N	– не изменяется.
Z	– не изменяется.
V	– не изменяется.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:
LDHI 44h, R2

Перед инструкцией		После инструкции	
R2	ABCD EF12hh	R2	0044 0000h

A.5.58 Инструкция LDI

Синтаксис:
LDI src, dst
Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23	16	15 7	8 0
0 0 0 0 1 0 0 0 0 0	G	dst	src	

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (R0-R11)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
src → dst
Описание:
Операнд src загружается в регистр dst. Операнды src и dst являются целыми со знаком.
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF = 0
N = 1 при генерации отрицательного результата, иначе = 0.
Z = 1 при генерации нулевого результата, иначе = 0.
V = 0
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.

Циклы:
1

Пример:

LDI *-AR1 (IR0), R5

Перед инструкцией

AR1	2Ch	
AR0	5h	
R5	3C5h	965
Данные в 27h		
	26h	38
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR1	2Ch	
IR0	5h	
R5	26h	38
Данные в 27h		
	26h	38
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.59 Инструкция LDIcond

Синтаксис:
LDIcond src, dst

Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

31 24	23 16	15 8	7 0
0 1 0 1	cond	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
Если cond – "истина", то
src → dst
Иначе: dst не изменяется.

Описание:
Если cond – "истина", то операнд src загружается в регистр dst. В противном случае, регистр dst не изменяется. Операнды src и dst являются целыми со знаком.
LDP (альтернативная форма LDIU) загружает регистр-указатель страницы даных (DP) или другой регистр с 16 старшими разрядами адреса перемещения.

В микросхеме 1867ВМ9Ф имеется 20 кодов условий, которые могут быть использованы с этой инструкцией, смотри раздел А.2, где приведен список мнемоник кодов условий, описание кодов условий и устанавливаемые флаги.
Обратите внимание, что команда LDIU (безусловная загрузка целого) используется для загрузки выбранного регистра ЦПУ без изменения флагов условий в то время, как инструкция LDI влияет на них.
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM зависит от значения бита OVM.
Циклы:
1

Пример:

LDIZ R4, R6

Перед инструкцией

R4	027Ch	636
R6	0FE2h	4066
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

R4	027Ch	636
R6	0FE2h	4066
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

А.5.60 Инструкция LDII

Синтаксис:
LDII src, dst
Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

31 29	24 23		16	15 8	7 0
0 0 0	0 1 0 0 0 1	G	dst	src	

Поле слова инструкции:

G	Методы адресации src
01	Прямая
10	Косвенная

Операция:
Формирует сигнал блокировки
src → dst
Описание:
Операнд src загружается в регистр dst. Операция блокировки выполняется с помощью сигналов LOCK# или LLOCK#. Операнды src и dst являются целыми со знаком. Необходимо обратить внимание на то, что разрешена только прямая или косвенная адресация. Детальное описание смотри в подразделе 9.7 «Операция блокировки».
Статусные биты:

Если ST(SET COND)=0, то флаги состояния изменяются только в том случае, если регистр назначения – один из (R0-R11). Если ST(SET COND)=1, то они изменяются для всех регистров.

LUF – не изменяется.
LV – не изменяется.
UF = 0
N = 1 при генерации отрицательного результата, иначе = 0.
Z = 1 при генерации нулевого результата, иначе = 0.
V = 0
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:
LDII @985Fh, R3

Перед инструкцией

DP	80
R3	0h

Данные в 80 985Fh

	0DCh
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

DP	80
R3	0DCh

Данные в 80 98F5h

	0DCh
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

A.5.61 Инструкция LDI || LDI

Синтаксис:
LDI src2, dst2
|| LDI src1, dst1
Операнды:
src1: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst1: регистровая адресация (R0-R7)
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst2: регистровая адресация (R0-R7)
Код:

31	24 23	16	15 8	7 0
1 1	0 0 0 1 1	dst2	dst1	0 0 0	src1	src2

Поле слова инструкции:
Нет
Операция:
src2 → dst2
|| src1 → dst1
Описание:
Загрузка двух целых выполняется параллельно. Если обе LDI загружают в один и тот же регистр, то Ассемблер выдает предупреждение об ошибке. В результате этого получается LDI src2, dst2
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.

Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.

Циклы:
1

Пример:

```
LDI  *--AR1(1), R7
|| LDI  *AR7++(IR0), R1
```

Перед инструкцией

AR1	80 9826h	
R7	0h	
AR7	0AEFh	
R4	733C0 0000h	1.79750e+02
AR7	80 98C5h	

Данные в 80 98AFh

	58B 4000h	-6.118750e+01

Данные в 80 98C4h

	0h	
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR3	80 98AFh	
AR7	80 98C5h	
IR1	0AFh	6.118750e+01
R4	0	
AR7	80 98C5h	

Данные в 80 98AFh

	58B 4000h	-6.118750e+01

Данные в 80 98C4h

	733 C000h	1.79750e+02
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.62 Инструкция LDI || STI

Синтаксис:

```
LDI src2, dst1
|| STI src3, dst2
```

Операнды:
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst1: регистровая адресация (R0-R7)
src3: регистровая адресация (R0-R7)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)
Код:

31	24	23	16	15 8	7 0
1 1	0 1 1 0 1	dst1	0 0 0	src3	dst2	src2

Поле слова инструкции:
Нет

Операция:
src2 → dst1
|| src3 → dst2

Описание:
Параллельно выполняются операции загрузки и сохранения целых чисел. Если операнды src2 и dst2 указывают на один и тот же адрес, то src2 считывается до того, как будет записан в dst2.
Статусные биты:
LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.

V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.

Циклы:
1

Пример:

```
LDI  *-AR1 (1), R2
|| STI  R7,*AR5++(IR0)
```

Перед инструкцией

AR1	80 98E7h	
R2	0h	
R7	35h	53
AR5	80 982Ch	
IR0	8h	

Данные в 80 98E6h

	0DCh	220

Данные в 80 982Ch

	0h	
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR1	80 98E7h	
R2	0DCh	220
R7	35h	53
AR5	80 9834h	
AR7	8h	

Данные в 80 98E6h

	0DCh	220

Данные в 80 982Ch

	35h	53
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.63 Инструкция LDM

Синтаксис:
LDM src, dst

Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (R0-R11)

Код:

31 24	23 16	15 8	7 0
0 0 0 | 0 1 0 0 1 0 | G	dst	src	

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (R0-R11)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
src(man) → dst(man)

Описание:
Поле мантиссы операнда src загружается в поле мантиссы регистра dst. Поле экспоненты dst не изменяется. Операнды src и dst являются числами в формате с ПЗ. При использовании непосредственного режима адресации разряды 15-12 слова команды устанавливаются в «0» Ассемблером. Если src в памяти, то 32-разрядные данные загружаются в поле мантиссы.

Статусные биты:
LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.

N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.

Циклы:
1

Пример:
LDM 156.75, R2 (156.75 = 07 1CC0 0000h)

Перед инструкцией

R2	0h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

AR3	00 1CC0 0000h	1.22460938e+00
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.64 Инструкция LDP

Синтаксис:
LDP src [, DP]

Операнды:
src: 16 старших разрядов 32-разрядного адреса источника (src)
dst: опционально (указатель страницы данных, если [, DP] указан)

Код:

31 24	23	16	15 8	7 0
0 0 0	0 1 0 0 0 0	1 1	1 0 0 0 0	src

Поле слова инструкции:
Нет

Операция:
src → указатель страницы данных

Описание:
Эта псевдооперация является альтернативной формой инструкции LDIU, за исключением того, что команда LDP определена только для непосредственного режима адресации (биты 22, $21=11_2$). Эти 16 старших разрядов 32-битного абсолютного значения src (заметим, что src меньше, чем 32 бита, которые заполнены нулями) записываются в младшие 16 разрядов указателя страницы данных.
Младшие 16 разрядов указателя используются при прямой адресации как указатель страницы данных, к которой будет осуществляться адресация.
Всего имеется 64 К страниц, каждая из которых размером 64 К слов. Разряды 31-16 указателя зарезервированы и сохранятся как «0».
Статусные биты:
LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.

Циклы:

1

Пример:

LDP @809900h, DP

или

LDP @809900h

Перед инструкцией		После инструкции		
DP	6465h	DP	0080h	16 наиболее значащих байтов 32 битов

src, расширенный нулями

LUF	0	LUF	0
LV	0	LV	0
UF	0	UF	0
N	0	N	0
Z	0	Z	0
V	0	V	0
C	0	C	0

A.5.65 Инструкция LDPE

Синтаксис:

LDPE src, dst

Операнды:

src: режим регистровой адресации (любой регистр из основного регистрового файла ЦПУ)

dst: расширенный регистровый файл IVTP или TVTP

Код:

31 24	23 16	15 8	7 0
0 1 1 1 0 1 1 0 1	0 0	dst	0 0 0 0 0 0 0 0 0 0 0 0	src

Поле слова инструкции:

Нет

Операция:

src → dst

Описание:

Загрузка регистра указателя таблицы векторнов системных прерываний IVTP или регистра указателя таблицы векторов программных прерываний TVTP. Эти регистры описаны в подразделе 3.2 «Дополнительный регистровый файл ЦПУ».

Регистровый операнд src загружается из основного регистрового файла в dst регистр в дополнительном регистровом файле. Операнд dst должен быть целым.

Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:
LDPE AR0, TVTP; установка указателя векторной таблицы программных прерываний

A.5.66 Инструкция LDPK

Синтаксис:
LDPK src
Операнды:
src: 16-разрядная беззнаковая непосредственная адресация
Код:

31 24	23 16	15 8	7 0
0 0 0	1 1 1 1 1 0 1 1	1 0 0 0 0	src

Поле слова инструкции:
Нет
Операция:
src → DP
Описание:
16-битное беззнаковое непосредственное значение загружается в DP регистр. Эта операция завершается с окончанием фазы декодирования инструкции LDPK; таким образом, загруженное значение готово для следующих инструкций для непосредственной адресации. Необходимо использовать эту инструкцию осторожно, когда используется регистр DP в инструкции, которая предшествует LDPK, например:
PUSH DP
LDPK new_value
Загрузка нового значения DP в стек вместо сохранения предыдущего значения.
Статусные биты:

LUF	– не изменяется.
LV	– не изменяется.
UF	– не изменяется.
N	– не изменяется.
Z	– не изменяется.
V	– не изменяется.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:
Нет

A.5.67 Инструкция LHw

Синтаксис:
LHw src, dst
Операнды:
src: регистровая, прямая, 16-разрядная непосредственная или косвенная адресация.
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23	16	15 8	7 0
1 0 1 1 1 0 1 0	H G	dst	src	

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

H	src половина слова
0	Половина слова 0 (наименее значащая часть слова)
1	Половина слова 1 (наиболее значащая часть слова)

Операция:

Расширенная знаком половина слова (0, 1) src → dst
w=загружаемая половина слова (0, 1)

1	0	= w указатель

Описание:

Указанная половина слова src операнда, дополненная знаком и смещенная вправо на 16 младших разрядов, записывается в регистр dst. Половина слова src расширена знаком. Когда определен режим непосредственной адресации и выбрана половина слова 1 (H=1), то инструкция LHw выполняет знаковое расширение 16-разрядного значения до 32-разрядного значения. Следовательно, соответствующая половина слова (0000h или FFFFh) сохраняется в 16 младших разрядах dst регистра.

Статусные биты:

Если ST(SET COND)=0, то флаги состояния изменяются, только если регистр назначения – один из (R0-R11). Если ST(SET COND)=1, то они изменяются для всех регистров назначения.

LUF – не изменяется.
LV – не изменяется.
UF = 0
N = 1, если получен отрицательный результат, иначе = 0.
Z = 1, если получен нулевой результат, иначе = 0.
V = 0
C – не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Циклы:

1

Пример:

LH0 R1, R2

Перед инструкцией

R1	ABCD EF12h
R2	1234 5678h

После инструкции

R1	ABCD EF12h
R2	FFFF EF12h

A.5.68 Инструкция LHUw

Синтаксис:

LHUw src, dst

Операнды:

src: регистровая, прямая, 16-разрядная непосредственная или косвенная адресация
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

31 24	23 16	15 8	7 0
1 0 1 | 1 1 0 1 0 | H | G	dst	src	

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

H	src половина слова
0	Половина слова 0 (наименее значащая часть слова)
1	Половина слова 1 (наиболее значащая часть слова)

Операция:

Незнаковая половина слова (0, 1) src → dst

w=загружаемая половина слова (0, 1)

1	0	= w указатель

Описание:

Указанная половина слова src операнда – беззнаковое и смещенное вправо на 16 младших разрядов, записывается в dst регистр. Когда определен режим непосредственной адресации и выбрана половина слова 1 (H=1), то LHw инструкция выполняет знаковое расширение 16-разрядного значения до 32-разрядного значения. Следовательно, соответствующая половина слова (0000h или FFFFh) сохраняется в 16 младших разрядах dst регистра.

Статусные биты:

Если ST(SET COND)=0, то флаги состояния изменяются, только если регистр назначения – один из (R0-R11). Если ST(SET COND)=1, то они изменяются для всех регистров назначения.

LUF – не изменяется.
LV – не изменяется.
UF = 0
N = 0
Z = 1, если сгенерирован нулевой результат, иначе = 0.
V = 0
C – не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Циклы:

1

Пример:

LH0 R1, R2

Перед инструкцией

R1	ABCD EF12h
R2	1234 5678h

После инструкции

R1	ABCD EF12h
R2	FFFF EF12h

A.5.69 Инструкция LSH

Синтаксис:

LSH src_count, dst

Операнды:

src_count: основные методы адресации (G)

dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

31 29	24 23	16	15 8	7 0
0 0 0	0 1 0 0 1 1	G	dst	src_count

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:

count = 7 (семи) младших разрядов src_count

Если count ≥0, то

dst << count → dst

Иначе:

dst >> |count| → dst

Описание:

Семь младших разрядов операнда count используются для генерации двоичного дополнения счетчика сдвига. Если операнд count больше «0», то операнд dst сдвигается влево на количество разрядов, определенное значением операнда count. Младшие разряды заполняются нулями, старшие разряды сдвигаются вовне через разряд переноса C (carry).

Логический сдвиг влево:

$$C \leftarrow dst \leftarrow 0$$

Если операнд count меньше «0», то dst сдвигается вправо на число разрядов, определенное абсолютным значением операнда count. Старшие разряды операнда dst заполняются «0» при сдвиге вправо. Младшие разряды сдвигаются вовне через разряд переноса C.

Логический сдвиг вправо:

$$0 \rightarrow dst \rightarrow C$$

Если операнд count = 0, то сдвиг не происходит и разряд переноса C устанавливается в «0».

Если операнд count больше, чем 32, то бит C (перенос) получает значение наименее значащего бита. Если count меньше, чем 32, то бит C устанавливается в «0».

Предполагается, что операнд count является целым со знаком, а операнд dst – беззнаковое целое.

Циклы:

1

Статусные биты:

Если ST(SET COND)=0, то флаги состояния изменяются только в том случае, если регистр назначения – один из (R0-R11). Если ST(SET COND)=1, то они изменяются для всех регистров.

LUF	– не изменяется.
LV	– не изменяется.
UF	= 0
N	– старший разряд выходного значения.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 0
C	– устанавливается по значению последнего сдвинутого вовне разряда.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Пример 1:

LSH R4, R7

Перед инструкцией

R4	018h	24
R7	02ACh	
LUF	0	
LV	0	

После инструкции

R4	018h	24
R7	0AC00 0000h	
LUF	0	
LV	0	

UF	0
N	0
Z	0
V	0
C	0

UF	0
N	1
Z	0
V	0
C	0

Пример 2:

LSH *-AR5(IR0), R5

Перед инструкцией

AR5	80 9908h
IR0	4h
R5	00 12C0 0000h

Данные в 80 9904h

	0FFF FFFF4h	-12
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR5	80 9908h
IR0	4h
R5	00 0001 2C00h

Данные в 80 9904h

	0FFF FFFF4h	-12
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.70 Инструкция LSH3

Синтаксис:

LSH3 src_count, src, dst

Операнды:

src, src_count: тип 1 или тип 2 трехоперандный тип адресации

dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

Тип 1

31 24	23	16	15 8	7 0
0 0 1 0 0 1 0 0 0	T	dst	src	src_count

Тип 2

31 24	23	16	15 8	7 0
0 0 1 1 0 1 0 0 0	T	dst	src	src_count

Поля слова инструкции:

Тип 1

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	Регистровая (R0-R11)
01	Косвенная (смещение=0, 1, IR0, IR1)	Регистровая (R0-R11)
10	Регистровая (любой регистр ЦПУ)	Косвенная (смещение=0, 1, IR0, IR1)
11	Косвенная (смещение=0, 1, IR0, IR1)	Косвенная (смещение=0, 1, IR0, IR1)

Тип 2

T	Методы адресации src1	Методы адресации src2
01	Регистровая (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
01	Регистровая (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)
11	Косвенная *+ARn (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:

Если count ≥ 0, то

src << count → dst.

Иначе:

src >> |count| → dst

Описание:

Семь младших разрядов операнда count используются для генерации двоичного дополнения счетчика сдвига. Если операнд count больше «0», то копия операнда src сдвигается влево на число разрядов, определенное значением операнда count, и результат записывается в dst (src не изменяется). Младшие разряды заполняются нулями, старшие разряды сдвигаются вовне через бит переноса C (Бит переноса).

Логический сдвиг влево:

$$C \leftarrow dst \leftarrow 0$$

Если операнд count меньше 0, то src сдвигается вправо на число разрядов, определенное абсолютным значением операнда count. Старшие разряды операнда dst заполняются «0» при сдвиге вправо. Младшие разряды сдвигаются вовне через разряд переноса C.

Логический сдвиг вправо:

$$0 \rightarrow dst \rightarrow C$$

Если операнд count = 0, то сдвиг не происходит и разряд переноса C устанавливается в «0». Предполагается, что операнд count является целым со знаком, а операнды dst и src – беззнаковые целые.

Циклы:

1

Статусные биты:

Флаги состояния изменяются только в том случае, если регистр назначения – один из (R7-R0).

LUF	– не изменяется.
LV	– не изменяется.
UF	= 0
N	– старший разряд выходного значения.
Z	= 1 при генерации нулевого выхода, иначе = 0.
V	= 0
C	– устанавливается по значению последнего сдвинутого вовне разряда. Равен «0» для счетчика сдвига, равного 0.

Бит режима:

Операция OVM не зависит от значения бита OVM.

П р и м е р:

Нет

A.5.71 Инструкция LSH3||STI

Синтаксис:

LSH3 src_count, src2, dst1

|| STI src3, dst2

Операнды:

src_count: регистр (R0-R7)

src2: косвенная адресация (смещение = 0, 1, IR0, IR1)

dst1: регистровая адресация (R0-R7)

src3: регистровая адресация (R0-R7)

dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)

Код:

31	24 23	16	15	8	7	0
1 1	0 1 1 1 0	dst1	count	src3	dst2	src2

Поле слова инструкции:

Нет

Операция:
count = 7 (семи) младших разрядов src_count
Если count ≥ 0, то
src2 << count → dst1
Иначе:
src2 >> |count| → dst1
|| src3 → dst2
Описание:
Семь младших разрядов операнда count используются для генерации двоичного дополнения счетчика сдвига.
Если операнд count больше «0», то копия операнда src2 сдвигается влево на число разрядов, определенное значением операнда count и результат записывается в dst1 (src2 не изменяется). Младшие разряды заполняются нулями, старшие разряды сдвигаются вовне через разряд переноса C (carry).
Логический сдвиг влево:

$$C \leftarrow dst2 \leftarrow 0$$

Если операнд count меньше «0», то dst сдвигается вправо на число разрядов, определенное абсолютным значением операнда count. Старшие разряды операнда dst заполняются «0» при сдвиге вправо. Младшие разряды сдвигаются вовне через разряд переноса C.
Логический сдвиг вправо:

$$0 \rightarrow dst2 \rightarrow C$$

Если операнд count = 0, то сдвиг не происходит и разряд переноса C устанавливается в 0.
Предполагается, что операнд src_count является 7-разрядным целым со знаком, а операнды dst1 и src2 – беззнаковые целые. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (LSH3), записывает в тот же регистр, то STI имеет на входе содержимое регистра до того, как оно было изменено инструкцией LSH3.
Если src2 и dst2 указывают на один и тот же адрес, то src2 считывается до записи в dst2.
Статусные биты:

LUF	– не изменяется.
LV	– не изменяется.
UF	= 0.
N	– старший разряд выходного значения.
Z	= 1 при генерации нулевого выхода, иначе = 0.
V	= 0
C	– устанавливается по значению последнего сдвинутого вовне разряда. Равен «0» для счетчика сдвига, равного 0.

Бит режима:
Операция OVM зависит от значения бита OVM.
Циклы:
1

Пример1:

```
LSH3  R2,*++AR3 (1), R0
|| STI  R4,*-AR5
```

Перед инструкцией

R2	18h	24
AR3	80 98C2h	
R0	0h	
R4	0DCh	220
AR5	80 98A3h	

Данные в 80 98C3h

AC0h

После инструкции

R2	18h	24
AR3	80 98C3h	
R0	0AC00 0000h	
R4	0DCh	220
AR5	80 98A3h	

Данные в 80 98C3h

0ACh

Данные в 80 98A2h

	0h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

Данные в 80 98A2h

	0DCh	220
LUF	0	
LV	0	
UF	0	
N	1	
Z	0	
V	0	
C	0	

Пример 2:

```
LSH3  R7,*AR2--(1), R2
|| STI   R0,*+AR0 (1)
```

Перед инструкцией

R7	0FFFFF FF4h	-12
AR2	80 9863h	
R2	0h	
R4	0DCh	300
AR0	80 98B7h	

Данные в 80 9863h

2C00 0000h

Данные в 80 98A2h

	0h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

R7	18h	-12
AR2	80 9862h	
R2	2C000h	
R4	0DCh	300
AR0	80 98B7h	

Данные в 80 9863h

2C00 0000h

Данные в 80 98A2h

	12Ch	300
LUF	0	
LV	0	
UF	0	
N	1	
Z	0	
V	0	
C	0	

A.5.72 Инструкция LWLct

Синтаксис:
LWLct src, dst
Операнды:
ct: число байтов {0, 1, 2 или 3} для смещения влево (ct x 8 = смещение в битах)
src: регистровая, прямая, 16-разрядная непосредственная или косвенная адресация
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23 16	15 8	7 0
1 0 1 1 0 1 0 B	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

B	src байт
00	Нет смещения
01	Смещение влево на 1 байт

10	Смещение влево на 2 байта
11	Смещение влево на 3 байта

Операция:
src << {0, 1, 2 или 3} байты и объединение с dst -> dst
Описание:
Операнд src – это смещенный влево на определенное число байтов и объединенный с байтами dst регистра, который равен нижней части влево сдвинутым младшим байтам src операнда. Когда определен метод непосредственной адресации, то эта инструкция выполняет знаковое расширение 16-разрядного значения до 32-разрядного значения, а затем это 32-разрядное значение смещает и объединяет.
Циклы:
1
Статусные биты:
Если ST(SET COND)=0 и конечные регистры – это (R0-R11), то флаги состояний изменяются. Если ST(SET COND)=1, то они изменяются для всех конечных регистров.

LUF	– не изменяется.
LV	– не изменяется.
UF	= 0
N	– старший разряд выходного значения.
Z	= 1 при генерации нулевого выхода, иначе = 0.
V	= 0
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.

Пример:

LWL2 R1, R2

Перед инструкцией

R1	ABCD EF12h
R2	1234 5678h

После инструкции

R1	ABCD EF12h
R2	EF12 5678h

A.5.73 Инструкция LWRct

Синтаксис:
LWRct src, dst
Операнды:
ct: число байтов {0, 1, 2 или 3} для смещения вправо (ct x 8 = смещение в байтах)
src: регистровая, прямая, 16-разрядная непосредственная или косвенная адресация
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23 16	15 8	7 0
1 0 1 1 0 1 1 | B	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

B	src байт
00	Нет смещения
01	Смещение влево на 1 байт
10	Смещение влево на 2 байта
11	Смещение влево на 3 байта

Операция:
rsc >> {0, 1, 2 или 3} байты и объединение с dst -> dst
Описание:
Операнд src смещен вправо на определенное число байтов и объединен с байтами dst регистра, который равен верхней части вправо сдвинутым старшим байтам src операнда. Знак не расширяется. Когда определен режим непосредственной адресации, то эта инструкция выполняет знаковое расширение 16-разрядного значения до 32-разрядного значения, а затем это 32-разрядное значение смещается и объединяется.
Циклы:
1
Статусные биты:
Если ST(SET COND)=0 и конечные регистры – это (R0-R11), флаги состояний изменяются. Если ST(SET COND)=1, они изменяются для всех конечных регистров.

LUF – не изменяется.
LV – не изменяется.
UF = 0
N – старший разряд выходного значения.
Z = 1 при генерации нулевого выхода, иначе = 0.
V = 0
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.

Пример:
LWL2 R1, R2

Перед инструкцией		После инструкции	
R1	ABCD EF12h	R1	ABCD EF12h
R2	1234 5678h	R2	12AB CDEFh

A.5.74 Инструкция MBct

Синтаксис:
MBct src, dst

Операнды:

ct: число байтов {0, 1, 2 или 3} для смещения влево (ct x 8 = смещение в байтах)
src: регистровая, прямая или косвенная адресация
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

31 24	23 16	15 8	7 0
1 0 1 1 1 0 0	B	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

B	src байт
00	Нет смещения
01	Смещение влево на 1 байт
10	Смещение влево на 2 байта
11	Смещение влево на 3 байта

Операция:

Восемь младших разрядов src << {0, 1, 2 или 3} байты и объединенные с dst -> dst

Описание:

Восемь младших разрядов src операнда смещены влево на 0, 1, 2 или 3 байта и объединены с битами dst регистра, которые равны нижней части влево сдвинутых младших байтов src операнда. Когда определен режим непосредственной адресации, то эта инструкция выполняет знаковое расширение 16-разрядного значения до 32-разрядного значения, а затем это 32-разрядное значение смещается и объединяется.

Циклы:

1

Статусные биты:

Если ST(SET COND)=0 и конечные регистры – это (R0-R11), флаги состояний изменяются. Если ST(SET COND)=1, они изменяются для всех конечных регистров.

LUF	– не изменяется.
LV	– не изменяется.
UF	= 0
N	– старший разряд выходного значения.
Z	= 1 при генерации нулевого выхода, иначе = 0.
V	= 0
C	– не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Пример:

MB2AR1, AR2

Перед инструкцией

AR1	ABCD EF12h	(0012 0000h)
AR2	1234 5678h	

После инструкции

AR1	ABCD EF12h
AR2	1212 5678h

A.5.75 Инструкция MHct

Синтаксис:
MHct src, dst

Операнды:

ct: число смещений половины слова (16 бит)
src: регистровая, прямая, 16-разрядная непосредственная или косвенная адресация
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23	16	15 8	7 0
1 0 1 1 1 0 0	H	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

H	src половина слова
0	Половина слова 0 (МБ слова)
1	Половина слова 1 (СБ слова)

Операция:
16 младших разрядов src << {0,1} половина слова и объединение с dst -> dst
Описание:
16 младших разрядов src операнда смещены влево на 0 или 1 половину слова и объединены с битами dst регистра, которые являются нижней частью влево сдвинутым байтам src операнда. Когда определен режим непосредственной адресации, эта инструкция выполняет знаковое расширение 16-разрядного значения до 32-разрядного значения, затем это 32-разрядное значение смещается и объединяется.
Статусные биты:
Если ST(SET COND)=0 и конечные регистры – это (R0-R11), то флаги состояний изменяются. Если ST(SET COND)=1, то они изменяются для всех конечных регистров.

LUF – не изменяется.
LV – не изменяется.
UF = 0
N – старший разряд выходного значения.
Z = 1 при генерации нулевого выхода, иначе = 0.
V = 0
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:
MH1 AR1, AR2

Перед инструкцией (EF12 0000h)

AR1	ABCD EF12h
AR2	1234 5678h

После инструкции

AR1	ABCD EF12h
AR2	EF12 5678h

A.5.76 Инструкция MPYF

Синтаксис:
MPYF src, dst
Операнды:
src: основные методы адресации src (G)
dst: регистровая адресация (R0-R11)
Код:

31 24	23	16	15 7	8 0
0 0 0	0 1 0 1 0 0	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (R0-R11)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
dst × src → dst
Описание:
Результат умножения операндов dst и src загружается в регистр dst. Операнд src, если он имеет регистровую адресацию (R0-R11) и dst являются числами в расширенном формате с ПЗ. Для нерегистрового метода адресации src принимает вид числа с плавающей запятой с одинарной точностью.
Статусные биты:

LUF	= 1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	= 1 при возникновении переполнения с ПЗ, в противном случае не меняется.
UF	= 1 при потере значимости результата с ПЗ, иначе = 0.
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении переполнения с ПЗ, иначе = 0.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
П р и м е р:
MPYFR0, R2

Перед инструкцией			После инструкции		
R0	07 0C80 0000h	1,4050e + 02	R0	07 0C80 0000h	1.4050e + 02
R2	03 4C20 0000h	1,27578125e + 01	R2	0A 600F 2000h	1.79247266e + 03
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	0		C	0	

A.5.77 Инструкция MPYF3

Синтаксис:
MPYF3 src2, src1, dst
Операнды:
src1, src2: 1-го или 2-го типа трехоперандной адресации
Код:
Тип 1

31	23 24		16	15	8 7	0
0 0 1	0 0 1 0 0 1	T	dst	src1	src2	

Тип 2

31	23 24		16	15	8 7	0
0 0 1	1 0 1 0 0 1	T	dst	src1	src2	

Поля слова инструкции:
Тип 1

T	Методы адресации src1	Методы адресации src2
00	Регистровая (R0-R11)	Регистровая (R0-R11)
01	Косвенная (смещение=0, 1, IR0, IR1)	Регистровая (любой регистр ЦПУ)
10	Регистровая (R0-R11)	Косвенная (смещение=0, 1, IR0, IR1)
11	Косвенная (смещение=0, 1, IR0, IR1)	Косвенная (смещение=0, 1, IR0, IR1)

Тип 2

T	Методы адресации src1	Методы адресации src2
01	Регистровая (R0-R11)	Косвенная *+ARn (5-битное беззнаковое смещение)
11	Косвенная *+ARn1 (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:
src1 × src2 → dst
Описание:
Результат умножения операндов src1 и src2 загружается в регистр dst. Операнды src1 и src2 (если src1 и src2 – регистры (R0-R11)) и dst обрабатываются как числа в формате с ПЗ с расширенной точностью. Если src1 и src2 не являются регистрами, то предполагается, что они являются числами с плавающей запятой одинарной точности.
Статусные биты:

LUF	= 1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	= 1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF	= 1 при потере значимости результата с ПЗ, иначе = 0.
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении ПЗ-переполнения, иначе = 0.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
П р и м е р:
Нет.

A.5.78 Инструкция MPYF3||ADDF3

Синтаксис:
MPYF3 srcA, srcB, dst1
|| ADDF3 srcC, srcD, dst2
Операнды:
srcA
srcB } Любые два должны быть с косвенной адресацией (смещение = 0, 1, IR0, IR1) и
srcC } любые два должны быть регистрами (R0-R7)
srcD
dst1 регистр (d1):
- 0 = R0
- 1 = R1

dst2 регистр (d2):
- 0 = R2
- 1 = R3

src1 регистровая адресация (R0-R7)
src2 регистровая адресация (R0-R7)
src3 косвенная адресация (смещение = 0, 1, IR0, IR1)
src4 косвенная адресация (смещение = 0, 1, IR0, IR1)
P параллельные методы адресации ($0 \le P \le 3$)
Операция (поле P)
00 src3 × src4, src1 + src2
01 src3 × src1, src4 + src2
10 src1 × src2, src3 + src4
11 src3 × src1, src2 + src4
Код:

31	24	23	16	15	8	7	0
1 0	0 0 0 0	P	d1	d2	src1	src2	src3	src4

Поле слова инструкции:
Нет
Операция:
srcA × srcB → dst1
|| srcC + srcD → dst2
Описание:
Умножение в формате с ПЗ и сложение в формате с ПЗ производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (MPYF3) считывает из регистра, и операция, которая производится параллельно (ADDF3), записывает в тот же регистр, то MPYF3 имеет на входе содержимое регистра до того, как оно было изменено инструкцией ADDF3.
Для четырех возможных операндов источника могут быть записаны любые комбинации режимов адресации, при этом два записываются как косвенные, а два – как регистры. Присвоение операндов источника srcA - srcD полям src1 - src4 изменяется в зависимости от комбинации использованных режимов адресации, и поле P кодируется соответственно. Ассемблер может, когда нет указания, изменить прядок операндов в коммуникативных операциях для упрощения процесса вычисления.
Если src2 и dst2 указывают на один и тот же адрес, то src2 считывается до записи в dst2.
Статусные биты:

LUF	= 1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	= 1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF	= 1 при потере значимости результата с ПЗ, иначе = 0.
N	= 0

Z = 0
V = 1 при возникновении ПЗ-переполнения, иначе = 0.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.

Циклы:
1

Пример:

```
MPYF3 *AR5++(1), *- -AR1 (IR0), R0
|| ADDF3 R5, R7, R3
```

Перед инструкцией			После инструкции		
AR5	80 98C5h		AR5	80 98C6h	
AR1	80 98A8h		AR1	80 98A4h	
IR0	4h		IR0	4h	
R0	0h		R0	04 6718 000h	2.88867188e+01
R5	07 33C0 0000h	1.79750e+02	R5	07 33C0 0000h	1.79750e+02
R7	07 0C80 0000h	1.4050e+02	R7	07 0C80 0000h	1.4050e+02
R3	0h		R3	08 2020 0000h	3.20250e+02
Данные в 80 98C5h			Данные в 80 98C5h		
	34C 0000h	1.2750e+01		34C 0000h	1.2750e+01
Данные в 80 98A4h			Данные в 80 98A4h		
	111 0000h	2.265625e+0		111 0000h	2.265625e+0
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	0		C	0	

А.5.79 Инструкция MPYF3 || STF

Синтаксис:

```
MPYF3 src2, src1, dst1
|| STF  src3, dst2
```

Операнды:
src1: регистровая адресация (R0-R7)
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst1: регистровая адресация (R0-R7)
src3: регистровая адресация (R0-R7)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)
Код:

31	24 23	16	15	8	7	0
1 1	0 1 1 1 1	dst1	src1	src3	dst2	src2

Поле слова инструкции:
Нет

Операция:
src1 × src2 → dst1
|| src3 → dst2

Описание:
Умножение в формате с ПЗ и сохранение числа в формате с ПЗ производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (MPYF3) записывает в регистр, и операция, которая производится параллельно (STI), считывает из того же ре-

гистра, то STF имеет на входе содержимое регистра до того, как оно было изменено командой MPYF3.
Если src2 и dst2 указывают на один и тот же адрес, то src2 считывается до записи в dst2.

Статусные биты:

LUF = 1 при потере значимости результата с ПЗ, иначе не изменяется.
LV = 1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF = 1 при потере значимости результата с ПЗ, иначе = 0.
N = 1 при генерации отрицательного результата, иначе = 0.
Z = 1 при генерации нулевого результата, иначе = 0.
V = 1 при возникновении ПЗ-переполнения, иначе = 0.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.

Циклы:
1

Пример:

```
MPYF3 *-AR2 (1), R7, R0
|| STF  R3,*AR0--(IR0)
```

Перед инструкцией

AR2	80 982Bh	
R7	05 7B40 0000h	6.281250e+01
R0	0h	
R3	80 9860h	4.7031250e+02
AR0	80 9860h	
IR0	8h	
Данные в 80 982Ah		
	70C 8000h	1.4050e+02
Данные в 80 9860h		
	0h	
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR2	80 982Bh	
R7	80 9862h	6.281250e+01
R0	2C000h	8.82515625e+03
R3	0DCh	4.7031250e+02
AR0	80 98B7h	
IR0	8h	
Данные в 80 982Ah		
	70C 8000h	1.4050e+02
Данные в 80 9860h		
	86B28 0000h	4.7031250e+02
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.80 Инструкция MPYF3 || SUBF3

Синтаксис:

```
MPYF3  srcA, srcB, dst1
|| SUBF3  srcC, srcD, dst2
```

Операнды:

srcA, srcB, srcC, srcD — Любые два должны быть с косвенной адресацией (смещение = 0, 1, IR0, IR1) и любые два должны быть регистры (R0-R7)

dst1 регистр (d1):
- 0 = R0
- 1 = R1

dst2 регистр (d2):
- 0 = R2
- 1 = R3

src1 регистровая адресация (Rn: $0 \le n \le 7$)

src2 регистровая адресация (Rn: 0 ≤ n ≤ 7)
src3 косвенная адресация (смещение = 0, 1, IR0, IR1)
src4 косвенная адресация (смещение = 0, 1, IR0, IR1)
P параллельные методы адресации (0 ≤ P ≤ 3)
Операция (поле P)
00 src3 × src4, src1 - src2
01 src3 × src1, src4 - src2
10 src1 × src2, src3 - src4
11 src3 × src1, src2 - src4
Код:

31		24	23		16	15	8	7	0
1 0	0 0 0 1	P	d1	d2	src1	src2	src3	src4	

Поле слова инструкции:
Нет.
Операция:
srcA × srcB → dst1
|| srcD - srcC → dst2
Описание:
Умножение в формате с ПЗ и вычитание в формате с ПЗ производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (MPYF3) считывает из регистра, и операция, которая производится параллельно (SUBF3), записывает в тот же регистр, то MPYF3 имеет на входе содержимое регистра до того, как оно было изменено командой SUBF3.
Для четырех возможных операндов источника могут быть записаны любые комбинации режимов адресации, при этом два записываются как косвенные, а два – как регистры. Присвоение операндов источника srcA - srcD полям src1 - src4 изменяется в зависимости от комбинации использованных режимов адресации, и поле P кодируется, соответственно. Ассемблер может, когда это не определено, изменить порядок операндов в коммуникативных операциях для упрощения процедуры вычисления.
Статусные биты:

LUF	= 1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	= 1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF	= 1 при потере значимости результата с ПЗ, иначе = 0.
N	= 0
Z	= 0
V	= 1 при возникновении ПЗ-переполнения, иначе = 0.
C	– не изменяется.

Бит режима:
Операция OVM зависит от значения бита OVM.
Циклы:
1
П р и м е р:

```
MPYF3 R5,*++AR7 (IR1), R0
|| SUBF3 R7,*AR3--(1), R2
или
MPYF3 *++AR7 (IR1), R5, R0
|| SUBF3 R7,*AR3--(1), R2
```

Перед инструкцией

R2	32h	50
AR0	80 98E3h	
R0	0h	
AR5	80 99FCh	
IR1	0Ch	

После инструкции

R2	320h	800
AR0	80 98E4h	
R0	01324h	4900
AR5	80 99F0h	
IR1	0Ch	

R4	07D0h	2000
Данные в 80 98E4h		
	62h	98
Данные в 80 99FCh		
	4B0h	1200
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

R4	07D0h	2000
Данные в 80 98E4h		
	62h	98
Данные в 80 99FCh		
	4B0h	1200
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.81 Инструкция MPYI

Синтаксис:

MPYI src, dst

Операнды:

src: основный адресные модели (G)

dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

31 24	23 16	15 8	7 0
0 0 0	0 1 0 1 0 1	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:

dst × src → dst

Описание:

Результат произведения операндов dst и src загружается в регистр dst. При чтении операнды src и dst являются 32-разрядными целыми числами со знаком. Результат является 64-разрядным целым со знаком. В регистр dst помещается 32 младших значащих разряда результата.

Целочисленное переполнение возникает, когда хотя бы один из старших 32 разрядов 64-разрядного результата отличается от старшего разряда 32-разрядного выходного значения.

Статусные биты:

Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF	– не изменяется.
LV	= 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF	= 0
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении целочисленного переполнения, иначе = 0.
C	– не изменяется.

Бит режима:

Операция OVM зависит от значения бита OVM.

Циклы:

1

Пример:

MPYI R1, R5

Перед инструкцией

R1	00 0033 C251h	3 392 081
R5	00 0078 B600h	7 910 912
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

R1	00 0033 C251h	3 392 081
R5	00 E21D 9600h	-501 377 536
LUF	0	
LV	1	
UF	0	
N	1	
Z	0	
V	1	
C	0	

Примечание – Результат с переполнением и R5 содержит 32 младших разряда произведения. Чтобы получить 32 старших разряда, используйте инструкции MPYSHI3 или MPYUHI3.

A.5.82 Инструкция MPYI3

Синтаксис:

MPYI3 src2, src1, dst

Операнды:

src1, src2: трехоперандные методы адресации первого или второго типа

dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

Тип 1

31 23 24		16	15 8	7 0
0 0 1 0 0 0 1 0 1 0	T	dst	src1	src2

Тип 2

31 23 24		16	15 8	7 0
0 0 1 1 0 1 0 1 0	T	dst	src1	src2

Поля слова инструкции:

Тип 1

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	Регистровая (любой регистр ЦПУ)
01	Косвенная (смещение=0, 1, IR0, IR1)	Регистровая (любой регистр ЦПУ)
10	Регистровая (любой регистр ЦПУ)	Косвенная (смещение=0, 1, IR0, IR1)
11	Косвенная (смещение=0, 1, IR0, IR1)	Косвенная (смещение=0, 1, IR0, IR1)

Тип 2

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	8-битная знаковая прямая
01	Регистровая (любой регистр ЦПУ)	Косвенная *+ARn(5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-битная знаковая прямая
11	Косвенная *+ARn1 (5-битное беззнаковое смещение)	Косвенная *+ARn2(5-битное беззнаковое смещение)

Операция:

src1 × src2 → dst

Описание:

Результат произведения операндов src1 и src2 загружается в регистр dst. Операнды src1 и src2 являются 32-разрядными целыми со знаком. Результат является 64-разрядным целым со знаком. В регистр dst помещается 32 младших значащих разряда результата.

Целочисленное переполнение возникает, когда хотя бы один из старших 32 разрядов 64-разрядного результата отличается от старшего разряда 32-разрядного выходного значения.
Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF	– не изменяется.
LV	= 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF	= 0
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении целочисленного переполнения, иначе = 0.
C	– не изменяется.

Бит режима:
Операция OVM зависит от значения бита OVM.
Циклы:
1
Пример:
Нет

A.5.83 Инструкция MPYI3 || ADDI3

Синтаксис:
MPYI3 srcA, srcB, dst1
|| ADDI3 srcC, srcD, dst2
Операнды:

srcA, srcB, srcC, srcD, dst1 } Любые два регистра (Rn: 0 ≤ n ≤ 7)
srcA, srcB, srcC, srcD, dst1 } Любые два операнда с косвенной адресацией (смещение = 0, 1, IR0, IR1)
srcA, srcB, srcC, srcD, dst1 } регистр (d1):
- 0 = R0
- 1 = R1

dst2 регистр (d2):
- 0 = R2
- 1 = R3

src1 регистровая адресация (R0-R7)
src2 регистровая адресация (R0-R7)
src3 косвенная адресация (смещение = 0, 1, IR0, IR1)
src4 косвенная адресация (смещение = 0, 1, IR0, IR1)
P параллельные типы адресации (0 ≤ P ≤ 3)
Операция (поле P):
00 src3 × src4, src1 + src2
01 src3 × src1, src4 + src2
10 src1 × src2, src3 + src4
11 src3 × src1, src2 + src4
Код:

31 24	23		16	15 8	7 0
1 0 | 0 0 1 0 | P	d1	d2 | src1	src2	src3	src4

Поле слова инструкции:
Нет
Операция:

srcA × srcB → dst1
|| srcD + srcC → dst2
Описание:
Целочисленное умножение и целочисленное сложение производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (MPYI3) считывает из регистра, и операция, которая выполняется параллельно (ADDI3), записывает в тот же регистр, то MPYI3 имеет на входе содержимое регистра до того, как оно было изменено командой ADDI3.
Для четырех возможных операндов источника могут быть записаны любые комбинации режимов адресации, при этом два записываются как косвенные, а два – как регистры. Присвоение операндов источника srcA - srcD полям src1 - src4 изменяется в зависимости от комбинации использованных режимов адресации, и поле P кодируется, соответственно. Ассемблер, если это не определено, может изменить порядок операндов в коммуникативных операциях для упрощения процесса вычисления.
Статусные биты:

LUF	– не изменяется.
LV	= 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF	= 0
N	= 0
Z	= 0
V	= 1 при возникновении целочисленного переполнения, иначе = 0.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:

```
MPYI3 R7, R4, R0
|| ADDI3 *-AR3,*AR5--(1), R3
```

Перед инструкцией

R7	14h	20
R4	64h	100
R0	0h	
AR3	80 981Fh	
AR5	80 996Eh	
R3	0h	
Данные в 80 981Eh		
	0FFFF FFCBh	-53
Данные в 80 996Eh		
	35h	53
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

R7	14h	20
R4	64h	100
R0	07D0h	2000
AR3	80 981Fh	
AR5	80 996Dh	
R3	0h	
Данные в 80 981Eh		
	0FFFF FFCBh	-53
Данные в 80 996Eh		
	35h	53
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.84 Инструкция MPYI3 || STI

Синтаксис:

```
MPYI3 src2, src1, dst1
|| STI  src3, dst2
```

Операнды:

src1: регистровая адресация (R0-R7)
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst1: регистровая адресация (R0-R7)
src3: регистровая адресация (R0-R7)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)
Код:

31	24	23	16	15	8	7	0
1 1	1 0 0 0 0	dst1	src1	src3	dst2	src2	

Поле слова инструкции:
Нет
Операция:
src1 × src2 → dst1
|| src3 → dst2
Описание:
Целочисленное умножение и сохранение целого выполняется параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (MPYI3) записывает в регистр, и операция, которая выполняется параллельно (STI), считывает из того же регистра, то STI имеет на входе содержимое регистра до того, как оно было изменено инструкцией MPYI3. Если src2 и dst2 указывают на один и тот же адрес, то src2 считывается до записи в dst2.
Переполнение целого числа происходит, когда какой либо из 32 старших разрядов 64-битного результата отличается от старших разрядов 32-битного значения dst1.
Статусные биты:

LUF	– не изменяется.
LV	= 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF	= 0
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении целочисленного переполнения, иначе = 0.
C	– не изменяется.

Бит режима:
Операция OVM зависит от значения бита OVM.
Циклы:
1
П р и м е р:

```
MPYI3 *-AR0(1), R5, R7
|| STI  R2,*-AP3(1)
```

Перед инструкцией

AR0	80 995Ah	
R5	32h	50
R7	0h	
R2	0DCh	220
AR3	80 982Fh	

Данные в 80 98E4h

	0C8h	200

Данные в 80 99FCh

	0h	
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR0	80 995Bh	
R5	32h	50
R7	2710h	10000
R2	0DCh	220
AR3	80 982Fh	

Данные в 80 98E4h

	0C8h	200

Данные в 80 99FCh

	0DCh	220
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.85 Инструкция MPYI3 || SUBI3

Синтаксис:
MPYI3 srcA, srcB, dst1
|| SUBI3 srcC, srcD, dst2
Операнды:
srcA, srcB, srcC, srcD — Любые два операнда – регистры (R0-R7) и любые два операнда – с косвенной адресацией (смещение = 0, 1, IR0, IR1)
dst1 регистр (d1):
0 = R0
1 = R1
dst2 регистр (d2):
0 = R2
1 = R3
src1 регистровая адресация (R0-R7)
src2 регистровая адресация (R0-R7)
src3 косвенная адресация (смещение = 0, 1, IR0, IR1)
src4 косвенная адресация (смещение = 0, 1, IR0, IR1)
P параллельные типы адресации ($0 \le P \le 3$)
Операция (поле P):
00 src3 × src4, src1 - src2
01 src3 × src1, src4 - src2
10 src1 × src2, src3 - src4
11 src3 × src1, src2 - src4
Код:

31		24	23		16	15	8	7	0
1 0	0 0 1 1	P	d1	d2	src1	src2	src3	src4	

Поле слова инструкции:
Нет
Операция:
srcA × srcB → dst1
|| srcD - srcC → dst2
Описание:
Целочисленное умножение и целочисленное вычитание выполняется параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (MPYI3) считывает из регистра, и операция, которая выполняется параллельно (SUBI3), записывает в тот же регистр, то MPYI3 имеет на входе содержимое регистра до того, как оно было изменено инструкцией SUBI3.
Для четырех возможных операндов источника могут быть записаны любые комбинации режимов адресации, при этом два записываются как косвенные, а два – как регистровые. Присвоение операндов источника srcA - srcD полям src1 - src4 изменяется в зависимости от комбинации использованных режимов адресации. Поле P кодируется, соответственно. Если не указано, то Ассемблер может изменить порядок в коммутативных операциях для упрощения процесса вычисления.
Переполнение целого числа происходит, когда какой-либо из 32 старших разрядов 64-битного результата отличается от старших разрядов 32-битного значения dst1.
Статусные биты:
LUF – не изменяется.
LV = 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF = 1 при возникновении отрицательного целочисленного переполнения, в противном случае не меняется.
N = 0
Z = 0
V = 1 при возникновении целочисленного переполнения, иначе = 0.
C – не изменяется.

Бит режима:
Операция OVM зависит от значения бита OVM.
Циклы:
1

Пример:

```
MPYI3 R2,*++AR0 (1), R0
|| SUBI3 *AR5--(IR1), R4, R2
или
MPYI3 *++AR0 (1), R2, R0
|| SUBI3 *AR5--(IR1), R4, R2
```

Перед инструкцией

R2	32h	50
AR0	80 98E3h	
R0	0h	
AR5	80 99FCh	
IR1	0Ch	
R4	07D0h	2000
Данные в 80 98E4h		
	0C8h	98
Данные в 80 99FCh		
	4B0h	1200
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR0	320h	800
R5	80 98E4h	
R7	01324h	4900
R2	80 99F0h	
AR3	0Ch	
R4	07D0h	2000
Данные в 80 98E4h		
	62h	98
Данные в 80 99FCh		
	4B0h	1200
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.86 Инструкция MPYSHI

Синтаксис:

```
MPYSHI src, dst
```

Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23 22	16	15 8	7 0
0 0 0	1 1 1 0 1 1	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
dst × src → dst
Описание:

32 старших разряда результата произведения операндов dst и src загружается в регистр dst. При чтении операнды src и dst являются 32-разрядными целыми числами со знаком. Результат является 64-разрядным целым со знаком. В регистр dst помещается 32 старших разряда результата. Инструкция MPYI сохраняет 32 младших разряда результата.

Статусные биты:

Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF – не изменяется.
LV – не изменяется.
UF = 0
N = 1 при генерации отрицательного результата, иначе = 0.
Z = 1, если все 64 бита результата =0, иначе = 0.
V = 0
C – не изменяется.

Бит режима:

Операция OVM зависит от значения бита OVM.

Циклы:

1

П р и м е р:

Нет

A.5.87 Инструкция MPYSHI3

Синтаксис:

MPYSHI3 src2, src1, dst

Операнды:

src1: трехоперандные методы адресации первого или второго типа
src2: трехоперандные методы адресации первого или второго типа
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

Тип 1

31 23	24	16	15 8	7 0
0 0 1 0 1 0 0 0 1	T	dst	src1	src2

Тип 2

31 23	24	16	15 8	7 0
0 0 1 1 1 0 0 0 1	T	dst	src1	src2

Поля слова инструкции:

Тип 1

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	Регистровая (любой регистр ЦПУ)
01	Косвенная (смещение=0, 1, IR0, IR1)	Регистровая (любой регистр ЦПУ)
10	Регистровая (любой регистр ЦПУ)	Косвенная (смещение=0, 1, IR0, IR1)
11	Косвенная (смещение=0, 1, IR0, IR1)	Косвенная (смещение=0, 1, IR0, IR1)

Тип 2

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	8-битная знаковая прямая
01	Регистровая (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-битная знаковая прямая
11	Косвенная *+ARn1 (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:

src1 × src2 → dst
Описание:
Результат произведения операндов src1 и src2 загружается в регистр dst. Операнды src1 и src2 являются 32-разрядными целыми со знаком. Результат является 64-разрядным целым со знаком. В регистр dst помещается 32 старших разряда результата. Инструкция MPYSHI3 сохраняет 32 младших разряда результата.
Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF	– не изменяется.
LV	= 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF	= 0
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении целочисленного переполнения, иначе = 0.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:
Нет

A.5.88 Инструкция MPYUHI

Синтаксис:
MPYUHI src, dst
Операнды:
src: основные методы адресации
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23	16	15 8	7 0
0 0 0 1 1 1 1 0 0	G	dst	src	

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
dst × src → dst
Описание:
32 старших разряда результата произведения операндов dst и src загружается в регистр dst. При чтении операнды src и dst являются 32-разрядными целыми числами со знаком. Результат является 64-разрядным целым со знаком. В регистр dst помещается 32 старших разряда результата. Инструкция MPYUHI сохраняет 32 младших разряда результата.
Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются.
Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF	– не изменяется.

LV – не изменяется.
UF = 0
N = 0
Z = 1, если все 64 бита результата = 0, иначе = 0.
V = 0
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

П р и м е р:

Нет

A.5.89 Инструкция MPYUHI3

Синтаксис:
MPYUHI3 src2, src1, dst
Операнды:
src1, src2: трехоперандные методы адресации первого или второго типа
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:
Тип 1

31 23 24		16	15 8	7 0
0 0 1 0 1 0 0 1 0	T	dst	src1	src2

Тип 2

31 23 24		16	15 8	7 0
0 0 1 1 1 0 0 1 0	T	dst	src1	src2

Поля слова инструкции:
Тип 1

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	Регистровая (любой регистр ЦПУ)
01	Косвенная (смещение=0, 1, IR0, IR1)	Регистровая (любой регистр ЦПУ)
10	Регистровая (любой регистр ЦПУ)	Косвенная (смещение=0, 1, IR0, IR1)
11	Косвенная (смещение=0, 1, IR0, IR1)	Косвенная (смещение=0, 1, IR0, IR1)

Тип 2

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	8-битная знаковая прямая
01	Регистровая (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-битная знаковая прямая
11	Косвенная *+ARn1 (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:
src1 × src2 → dst
Описание:
Результат произведения операндов src1 и src2 загружается в регистр dst. Операнды src1 и src2 являются 32-разрядными целыми со знаком. Результат является 64-разрядным целым со знаком. В регистр dst помещается 32 старших разряда результата. Инструкция MPYUHI3 сохраняет 32 младших разряда результата.
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.
Статусные биты:
LUF – не изменяется.

LV – не изменяется.
UF = 0
N = 0
Z = 1, если все 64 бита результата = 0, иначе Z = 0.
V = 0
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:
Нет

A.5.90 Инструкция NEGB

Синтаксис:
NEGBsrc, dst
Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 23	24 16	15 8	7 0
0 0 0 | 0 1 0 1 1 0	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
0 - src - C → dst
Описание:
Результат вычитания из «0» операнда src и C загружаются в регистр dst. Предполагается, src и dst являются целыми со знаком.
Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.
LUF – не изменяется.
LV = 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF = 0
N = 1 при генерации отрицательного результата, иначе = 0.
Z = 1 при генерации нулевого результата, иначе = 0.
V = 1 при возникновении целочисленного переполнения, иначе = 0.
C = 1 при генерации заема, иначе = 0.

Бит режима:
Операция OVM зависит от значения бита OVM.
Циклы:
1

Пример:
NEGBR5, R7

Перед инструкцией			После инструкции		
R5	0FFFF FFCBh	-53	R5	0FFFF FFCBh	-53

R7	0h		R7	34h	52
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	1		C	1	

A.5.91 Инструкция NEGF

Синтаксис:
NEGF src, dst
Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (R0-R11)
Код:

31 23	24	16	15 8	7 0
0 0 0	0 1 0 1 1 1	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
0 - src → dst
Описание:
Результат вычитания из «0» операнда src загружается в регистр dst. Предполагается что, src и dst являются числами в формате с ПЗ.
Статусные биты:

LUF = 1 при потере значимости результата с ПЗ, иначе не изменяется.
LV = 1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF = 1 при потере значимости результата с ПЗ, иначе = 0.
N = 1 при генерации отрицательного результата, иначе = 0.
Z = 1 при генерации нулевого результата, иначе = 0.
V = 1 при возникновении ПЗ-переполнения, иначе = 0.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:
NEGF *++AR3 (2), R1

Перед инструкцией			После инструкции		
AR3	80 9800h		AR3	80 9802h	
R1	05 7B40 0025h	6.28125006e+01	R1	07 F380 000h	-1.4050e+02
Данные в 80 9802h			Данные в 80 9802h		
	70C 8000h	1.4050e+02		70C 8000h	1.4050e+02
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	

Z	0		Z	0	
V	0		V	0	
C	0		C	0	

A.5.92 Инструкция NEGF || STF

Синтаксис:
NEGF src2, dst1
|| STF src3, dst2
Операнды:
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst1: регистровая адресация (R0-R7)
src3: регистровая адресация (R0-R7)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)
Код:

31 24	23 16	15 8	7 0
1 1 | 1 0 0 0 1 | dst1	0 0 0 | src3	dst2	src2

Поле слова инструкции:
Нет
Операция:
0 - src2 → dst1
|| src3 → dst2
Описание:
Число в формате с ПЗ вычитается из «0», сохраняется в dst1 и параллельно сохраняется число из src3 в dst2 в формате с ПЗ. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STF) считывает из регистра, и операция, которая выполняется параллельно (NEGF), записывает в тот же регистр, STF имеет на входе содержимое регистра до того, как оно было изменено командой NEGF.
Если src2 и dst2 указывают на один и тот же адрес, то src2 считывается до записи в dst2.
Статусные биты:

LUF	= 1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	= 1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF	= 1 при потере значимости результата с ПЗ, иначе = 0.
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении ПЗ-переполнения, иначе = 0.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:
NEGF *AR4--(1), R7
|| STF R2,*++AR5 (1)

Перед инструкцией

AR4	80 98E1h	
R7	0h	
R2	07 33C0 0000h	1.79750e+02
AR5	80 9803h	

Данные в 80 98E4h

57 B40 0000h	6.281250e+01

Данные в 80 99FCh

После инструкции

AR4	80 98E0h	
R7	05 84C0 0000h	-6.281250e+01
R2	07 33C0 0000h	1.79750e+02
AR5	80 9804h	

Данные в 80 98E4h

57 B40 0000h	6.281250e+01

Данные в 80 99FCh

	0h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

	733 C000h	1.79750e+02
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.93 Инструкция NEGI

Синтаксис:
NEGI src, dst
Операнды:
src: основные методы адресации src (G)
dst: регистры (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23 16	15 8	7 0
0 0 0	0 1 1 0 0 0	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
0 - src → dst
Описание:
Операнд, представляющий собой вычитание из нуля операнда src, загружается в регистр dst. Предполагается, что src и dst являются целыми со знаком.
Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF	– не изменяется.
LV	= 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF	= 0
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении целочисленного переполнения, иначе = 0.
C	= 1 при возникновении заема, иначе = 0.

Бит режима:
Операция OVM зависит от значения бита OVM.
Циклы:
1
Пример:
NEGI 174, R5 (174 = 0AEh)

Перед инструкцией

R5	0DCh	220
LUF	0	
LV	0	
UF	0	
N	0	

После инструкции

AR4	0FFFF FF52h	-174
LUF	0	
LV	0	
UF	0	
N	0	

Z	0		Z	0
V	0		V	0
C	0		C	0

A.5.94 Инструкция NEGI || STI

Синтаксис:
NEGI src2, dst1
|| STI src3, dst2
Операнды:
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst1: регистровая адресация (R0-R7)
src3: регистровая адресация (R0-R7)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)
Код:

31 24	23 16	15 8	7 0
1 1 | 1 0 0 1 0 | dst1	0 0 1 | src3	dst2	src2

Поле слова инструкции:
Нет
Операция:
0 - src2 → dst1
|| src3 → dst2
Описание:
Вычитание из нуля числа в целочисленном формате и сохранение целого числа производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (NEGI), записывает в тот же регистр, то STI имеет на входе содержимое регистра до того, как оно было изменено командой NEGI.
Если src2 и dst2 указывают на один и тот же адрес, то src2 считывается до записи в dst2.
Статусные биты:

LUF	– не изменяется.
LV	= 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF	= 0
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении целочисленного переполнения, иначе = 0.
C	= 1 при возникновении заема, иначе = 0.

Бит режима:
Операция OVM зависит от значения бита OVM.
Циклы:
1
Пример:
NEGI *-AR3, R2
|| STI R2,*AR1++

Перед инструкцией

AR3	80 982Fh	
R2	19h	25
AR1	80 98A5h	

Данные в 80 982Eh

0DCh	220

Данные в 80 98A5h

После инструкции

AR4	80 982Fh	
R2	0FFFF FF24h	-220
AR1	80 98A6h	

Данные в 80 982Eh

0DCh	220

Данные в 80 98A5h

	0h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

	19h
LUF	0
LV	0
UF	0
N	1
Z	0
V	0
C	1

25

A.5.95 Инструкция NOP

Синтаксис:
NOP src
Операнды:
src: основные методы адресации (G)
Код:

31	24 23		16	15 8 7 0
0 0 0	0 1 1 0 0 1	G	0 0 0 0 0	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (нет операции)
10	Косвенная (видоизменение ARn: 0≤n≥7)

Операция:
Нет никаких операций АЛУ или умножителя.
ARn изменяются, если src определен в косвенной адресации.
Описание:
Если операнд src определен в косвенном методе адресации, то выполняется определенная операция адресации и имеет место пустое чтение памяти. Если операнд src опущен, то не выполняются никакие операции.
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример 1:

NOP

Перед инструкцией		После инструкции	
PC	3Ah	PC	3Bh

Пример 2:

NOP*AR3 --(1)

Перед инструкцией		После инструкции	
PC	5h	PC	6h
AR3	80 9900h	AR3	80 98FFh

A.5.96 Инструкция NORM

Синтаксис:
NORM src, dst
Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (R0-R11)
Код:

31 24	23 16	15 8	7 0
0 0 0 0 1 1 0 1 0	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
norm (src) → dst
Описание:
Предполагается, что операнд src является ненормализованным числом в формате с ПЗ, т. е. неявный (следующий за знаковым) разряд установлен по значению знакового разряда. Операнд dst равен нормализованному операнду src с удаленным неявным разрядом. Экспонента операнда dst устанавливается равной экспоненте операнда src минус величина левого сдвига, необходимого для нормализации src. Операнд dst является нормализованным числом в формате с ПЗ.
Если src(exp) = -128 и src(man) = 0, то dst = 0, Z = 1 и UF = 0.
Если src(exp) = -128 и src(man) ≠ 0, то dst = 0, Z = 0 и UF = 1.
При любых других значениях src и при возникновении отрицательного переполнения (т. е. потере значимости) dst(man) устанавливается в «0» и dst(exp) = -128. Если src(man) = 0, то dst(man) = 0 и dst(exp) = -128.

Для получения более полной информации смотри раздел 5.7 «Нормализация числа».

Статусные биты:

LUF	= 1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	– не изменяется.
UF	= 1 при потере значимости результата с ПЗ, иначе = 0.
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 0
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:
NORM R1, R2

Перед инструкцией

R1	04 0000 3AF5h
R2	07 0C80 0000h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

R1	04 0000 3AF5h	
R2	F2 6BD4 0000h	1.12451613e-04
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.97 Инструкция NOT

Синтаксис:
NOT src, dst
Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23 22	16	15 8	7 0
0 0 0	0 1 1 0 1 1	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
~src → dst
Описание:
Поразрядное дополнение операнда src загружается в регистр dst. Дополнение формируется путем инверсии каждого разряда операнда src. Операнды src и dst являются беззнаковыми целыми.
Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF	– не изменяется.
LV	– не изменяется.
UF	= 0
N	– старший разряд выходного значения.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 0
C	– не изменяется.

Бит режима:
Операция OVM зависит от значения бита OVM.
Циклы:
1

Пример:
NOT@982Ch, R4

Перед инструкцией

DP	80h
R2	0h

Данные в 80 982Ch

	5E2Fh
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

AR4	80h
R2	0FFFF A1D0h

Данные в 80 982Ch

	5E2Fh
LUF	0
LV	0
UF	0
N	1
Z	0
V	0
C	0

A.5.98 Инструкция NOT || STI

Синтаксис:
NOT src2, dst1
|| STI src3, dst2
Операнды:
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst1: регистровая адресация (R0-R7)
src3: регистровая адресация (R0-R7)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)
Код:

31	24 23	16	15	8	7	0
1 1	1 0 0 1 1	dst1	0 0 0	src3	dst2	src2

Поле слова инструкции:
Нет
Операция:
~src2 → dst1
|| src3 → dst2
Описание:
Поразрядное логическое «НЕ» (NOT) операнда в src2 и сохранение целого производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (NOT), записывает в тот же регистр, то STI имеет на входе содержимое регистра до того, как оно было изменено командой NOT.
Если src2 и dst2 указывают на один и тот же адрес, то src2 считывается до записи в dst2.
Статусные биты:

LUF	– не изменяется.
LV	– не изменяется.
UF	= 0
N	– старший разряд выходного значения.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 0
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:

```
NOT*AR2, R3
|| STI  R7,*--AR4 (IR1)
```

Перед инструкцией

AR2	80 99BCh	
R3	0h	
R7	0DCh	220
AR4	80 9850h	
IR1	10h	

Данные в 80 99CCh

	0C2Fh

Данные в 80 9840h

	0h	
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	

После инструкции

AR2	80 99CBh	
R3	0FFFF F3D0h	
R7	0DCh	220
AR4	80 9840h	
IR1	10h	

Данные в 80 99CCh

	0C2Fh

Данные в 80 9840h

	0DCh	220
LUF	0	
LV	0	
UF	1	
N	0	
Z	0	
V	0	

C	0	C	0

A.5.99 Инструкция OR

Синтаксис:
OR src, dst
Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23 16	15 8	7 0
0 0 0	1 0 0 0 0 0	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
dst OR src → dst
Описание:
Поразрядное логическое «ИЛИ» (OR) между операндами src и dst загружается в регистр dst. Операнды src и dst являются беззнаковыми целыми.
Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF	– не изменяется.
LV	– не изменяется.
UF	= 0
N	– старший разряд выходного значения
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 0
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:
OR *++AR1 (IR1), R2

Перед инструкцией

AR1	80 9800h
IR1	4h
R2	01256 0000h

220

Данные в 80 9804h

	2BCDh
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

AR1	80 9804h
IR1	4h
R2	01256 2BCDh

220

Данные в 80 9804h

	2BCDh
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

A.5.100 Инструкция OR3

Синтаксис:
OR3 src2, src1, dst
Операнды:
src1, src2: трехоперандные методы адресации первого или второго типа
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:
Тип 1

31 24	23	16	15 8	7 0
0 0 1 0 0 1 0 1 1	T	dst	src	src_count

Тип 2

31 24	23	16	15 8	7 0
0 0 1 1 0 1 0 1 1	T	dst	src	src_count

Поля слова инструкции:
Тип 1

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	Регистровая (любой регистр ЦПУ)
01	Косвенная (смещение=0, 1, IR0, IR1)	Регистровая (любой регистр ЦПУ)
10	Регистровая (любой регистр ЦПУ)	Косвенная (смещение=0, 1, IR0, IR1)
11	Косвенная (смещение=0, 1, IR0, IR1)	Косвенная (смещение=0, 1, IR0, IR1)

Тип 2

T	Методы адресации src1	Методы адресации src2
01	Регистровая (любой регистр ЦПУ)	8-битное знаковое непосредственное
01	Регистровая (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-битное знаковое непосредственное
11	Косвенная *+ARn (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:
src1 OR src2 → dst
Описание:
Поразрядное логическое «ИЛИ» (OR) между операндами src1 и src2. Результат загружается в регистр dst. Операнды src1, src2 и dst являются беззнаковыми целыми. Операнд src2 с непосредственным методом адресации расширен знаком.
Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF – не изменяется.
LV – не изменяется.
UF = 0
N – старший разряд выходного значения.
Z = 1 при генерации нулевого результата, иначе = 0.
V = 0
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:
Нет

A.5.101 Инструкция OR3 || STI

Синтаксис:
OR3 src2, src1, dst1
|| STI src3, dst2
Операнды:
src1: регистровая адресация (R0-R7)
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst1: регистровая адресация (R0-R7)
src3: регистровая адресация (R0-R7)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)
Код:

31	24 23	16	15 8	7 0
1 1 | 1 0 1 0 0	dst1	src1	src3	dst2 | src2

Поле слова инструкции:
Нет
Операция:
src1 OR src2 → dst1
|| src3 → dst2
Описание:
Поразрядное логическое «ИЛИ» (OR) операндов и сохранение целого выполняется параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (OR3), записывает в тот же регистр, то STI имеет на входе содержимое регистра до того, как оно было изменено командой OR3.
Если src2 и dst2 указывают на один и тот же адрес, то src2 считывается до записи в dst2.
Статусные биты:

LUF	– не изменяется.
LV	– не изменяется.
UF	= 0
N	– старший разряд выходного значения.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 0
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:

```
OR3   *++AR2, R5, R2
|| STI  R6,*AR1--
```

Перед инструкцией

AR2	80 9830h
R5	80 0000h
R2	0h
R6	0DCh
AR1	80 9833h

220

Данные в 80 9831h

9800h

Данные в 80 9883h

	0h
LUF	0
LV	0
UF	0
N	0
Z	0

После инструкции

AR2	80 9831h
R5	80 0000h
R2	80 9800h
R6	0DCh
AR1	80 9882h

220

Данные в 80 9831h

9800h

Данные в 80 9883h

	0DCh
LUF	0
LV	0
UF	0
N	0
Z	0

220

V	0	V	0
C	0	C	0

A.5.102 Инструкция POP

Синтаксис:
POP dst
Операнды:
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23 16	15 8	7 0
0 0 0	0 1 1 1 0 0	0 1	dst	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Поле слова инструкции:
Нет
Операция:
*SP-- → dst
Описание:
Содержимое вершины системного стека выталкивается из стека и загружается в регистр dst (в 32 младших разряда). Вершина стека – целое со знаком. Инструкция POP выполняется с постдекрементом указателя стека. Восемь старших разрядов (экспонента) в регистрах расширенной точности (R0-R11) не изменяются. Если требуется, то они могут быть восстановлены инструкцией POPF.
Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF	– не изменяется.
LV	– не изменяется.
UF	= 0
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 0
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:
POP R3

Перед инструкцией			После инструкции		
SP	80 9856h		SP	80 9855h	
R3	012DAh	4 826	R3	0FFFF 0DA4h	-62 044
Данные в 80 9856h			Данные в 80 9856h		
	0FFFF 0DA4h	-62 044		0FFFF ODA4h	-62 044
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	1	
Z	0		Z	0	
V	0		V	0	
C	0		C	0	

А.5.103 Инструкция POPF

Синтаксис:
POPF dst
Операнды:
dst: регистровая адресация (R0-R7)
Код:

31 24	23 16	15 8	7 0
0 0 0	0 1 1 1 0 1	0 1	dst	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Поле слова инструкции:
Нет
Операция:
*SP-- → dst1
Описание:
Содержимое вершины системного стека выталкивается и загружается в регистр dst (в 32 старших разряда). Вершина стека – число в формате с ПЗ. Инструкция POPF выполняется с постдекрементом указателя стека. Восемь младших разрядов в регистрах расширенной точности (R7-R0) заполняются нулями.
Циклы:
1
Статусные биты:

LUF	– не изменяется.
LV	= 0
UF	– не изменяется.
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 0
C	– не изменяется.

Бит режима:
Операция OVM зависит от значения бита OVM
Пример:
POPF R4

Перед инструкцией			После инструкции		
SP	80 984Ah		SP	80 9849h	
R4	012DAh	6.91186578e+00	R4	5F 2C13 0200h	5.32544007e+28
Данные в 80 984Ah			Данные в 80 984Ah		
	0FFFF 0DA4h	5.32544007e+28		5F2C 1302h	5.32544007e+28
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	0		C	0	

А.5.104 Инструкция PUSH

Синтаксис:
PUSH src
Операнды:
src: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23 16	15 8	7 0
0 0 0	0 1 1 1 1 0	0 1	src	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Поле слова инструкции:

Нет
Операция:
src → *++SP
Описание:
Содержимое регистра src (32 младших значащих разряда) загружается в текущую ячейку системного стека. Целое или мантисса регистра с повышенной точностью (R0-R11) сохраняется этой инструкцией. Восемь старших разрядов (экспонента) могут быть загружены с помощью инструкции PUSHF. Операнд src предполагается знаковым целым. Инструкция PUSH изменяет указателя стека с прединкриментом.
Статусные биты:

LUF	– не изменяется.
LV	– не изменяется.
UF	– не изменяется.
N	– не изменяется.
Z	– не изменяется.
V	– не изменяется.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:
PUSH R6

Перед инструкцией

SP	80 98AEh
R6	815Bh

33 115

Данные в 80 98AFh

	0h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

SP	80 98AFh
R6	815Bh

33 115

Данные в 80 98AFh

	8115Bh
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

33 115

А.5.105 Инструкция PUSHF

Синтаксис:
PUSHF src
Операнды:
src: регистровая адресация (R0-R11)
Код:

31 24	23 16	15 8	7 0
0 0 0 0 1 1 1 1 1	0 1	src	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Поле слова инструкции:
Нет
Операция:
src → *++SP
Описание:
Содержимое регистра src (32 старших значащих разряда) загружается в текущую ячейку системного стека. Предполагается, что src является числом в формате с ПЗ. Команда PUSHF выполняется с прединкрементом указателя стека. Восемь младших значащих разрядов мантиссы не сохраняются (обратите внимание на отличие значений в R2 и в стеке в приведенном ниже примере), но они могут быть сохранены при

использовании инструкции PUSH. Инструкция PUSHF должна выполняться после инструкции PUSH.
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:

PUSHF R2

Перед инструкцией

SP	80 9801h	
R2	02 5C12 8081h	6.87725854e+00

Данные в 80 98AFh

	0h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

SP	80 9802h	
R2	02 5C12 8081h	6.87725854e+00

Данные в 80 98AFh

	025C 1280h	33 115
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.106 Инструкция RCPF

Синтаксис:
RCPF src, dst
Операнды:
src: регистр повышенной точности, прямая или косвенная адресация
dst: (R0-R11)
Код:

31 24	23 16	15 8	7 0
0 0 0 1 1 1 0 1 0	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
16-разрядная обратная величина от src → dst
Описание:
16-разрядная аппроксимация обратной величины от операнда src загружается в регистр dst. Операнды dst и src представляют собой числа с плавающей запятой.
Статусные биты:

LUF =1, если произошла потеря значимости результата в формате с плавающей запятой, иначе не изменяется

LV =1, если произошло переполнение результата в формате с плавающей запятой, иначе не изменяется
UF =1, если произошла потеря значимости результата в формате с плавающей запятой, иначе = 0
N =1, если результат отрицательный, иначе = 0
Z =1, если результат нулевой, иначе = 0
V =1, если произошло переполнение результата в формате с плавающей запятой
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:
Нет

A.5.107 Инструкция RETIcond

Синтаксис:
RETIcond
Операнды:
Нет
Код:

31	24 23	16	15 8	7 0
0 1 1 1 1	0 0 0 0 0 0	cond	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	

Поле слова инструкции:
Нет
Операция:
Если cond – "истина", то
*(SP)-- → PC
ST(PGIE) → ST(GIE)
ST(PCF) → ST(CF)
Иначе продолжить выполнение.
Описание:
Если условие истинно, то содержимое вершины стека выталкивается в PC, PGIE копируется в GIE, и PCF копируется в CF. Если условие ложно, то продолжается обычное выполнение, смотри раздел A.2, где приведен список мнемоник кодов условий, описание кодов условий и устанавливаемые флаги.
Статусные биты:
LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
4

Пример:
Нет

A.5.108 RETIcondD инструкция

Синтаксис:
RETIcondD
Операнды:
Нет

Код:

31	24 23	16	15 8 7 0
0 1 1 1 1	0 0 0 0 0 1	cond	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Поле слова инструкции:
Нет
Операция:
Если cond – "истина", то
*(SP) → PC
ST(PGIE) → ST(GIE)
ST(PCF) → ST(CF)
Иначе – продолжение.
Описание:
Выполняется задержанный возврат из системного или программного прерывания. Так как это задержанный возврат, то три инструкции, следующие за RETIcondD, выбираются и исполняются. Эти три инструкции не должны влиять на выполнение программы, загрузку регистра состояния или изменение регистра указателя стека (SP), смотри раздел A.2, где приведен список мнемоник кодов условий, описание кодов условий и устанавливаемые флаги.
Прерывания запрещены в течение выполнения RETIcondD.
Статусные биты:
LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.
Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:
Нет

А.5.109 Инструкция RETScond

Синтаксис:
RETScond
Операнды:
Нет
Код:

31 24	23 16	15 8	7 0
0 1 1 1 1	0 0 0 1	0 0	cond	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Поле слова инструкции:
Нет
Операция:
Если cond – "истина", то
*SP-- → PC,
иначе – продолжение.
Описание:
Выполняется условный возврат. Если условие истинно, то содержимое вершины стека записывается в PC. В микросхеме 1867ВМ9Ф имеется 20 кодов условий, которые могут быть использованы с этой инструкцией, смотри раздел А.2, где приведен список мнемоник кодов условий, описание кодов условий и устанавливаемые флаги.
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
4

Пример:
RETSGE

Перед инструкцией

PC	123h
SP	80 983Ch

Данные в 80 983Ch

	456h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

PC	456h
SP	80 983Bh

Данные в 80 983Ch

	456h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

A.5.110 Инструкция RND

Синтаксис:
RND src, dst
Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (R0-R11)
Код:

31 24	23 16	15 8	7 0
0 0 0 1 0 0 0 0 1 0	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
rnd (src) → dst
Описание:
Результат округления операнда src загружается в регистр dst. Операнд src округляется до ближайшего значения с одинарной точностью в формате с ПЗ. Если значение операнда src находится точно посредине между двумя значениями с одинарной точностью в формате с ПЗ, то оно округляется к большему положительному из этих двух значений. Заметим, что округление «0» не устанавливает нулевой (Z) статусный бит, в отличие от бита переполнения.
Циклы:
1
Статусные биты:

LUF	= 1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	= 1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF	= 1 при потере значимости результата с ПЗ или src = 0, иначе = 0.
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении ПЗ-переполнения, иначе = 0.
C	– не изменяется.

Бит режима:
Операция OVM зависит от значения бита OVM.

Пример:

RND R5, R2

Перед инструкцией

R5	07 33C1 6EEFh	1.79755599e+02
R2	0h	
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

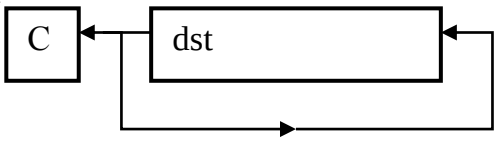
R5	07 33C1 6EEFh	1.79755599e+02
R2	07 33C1 6EEFh	1.79755600e+02
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.111 Инструкция ROL

Синтаксис:
ROL dst
Операнды:
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23 16	15 8	7 0
0 0 0	1 0 0 0 1 1	1 1	dst	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Поля слова инструкции:
Нет
Операция:
dst циклически сдвинутый влево на 1 разряд → dst
Описание:
Содержимое операнда dst циклически сдвигается влево на один разряд и загружается в регистр dst. Это циклический сдвиг с пересылкой старшего значащего разряда в младший.
Циклический левый сдвиг:



Циклы:
1
Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF	– не изменяется.
LV	– не изменяется.
UF	= 0
N	– старший значащий разряд выходного значения.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 0
C	– устанавливается по значению циклически сдвигаемого вовне старшего разряда.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Пример:
ROL R3

Перед инструкцией

R3	8002 5CD4h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

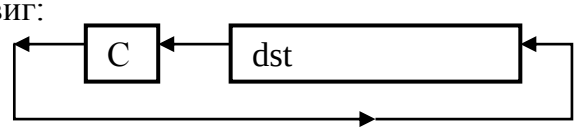
R3	0004 B9A9h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	1

A.5.112 Инструкция ROLC

Синтаксис:
ROLC dst
Операнды:
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23 16	15 8	7 0
0 0 0 1 0 0 1 0 0	1 1	dst	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Операция:
dst циклически сдвинутый влево на 1 разряд через разряд переноса → dst
Описание:
Содержимое операнда dst циклически сдвигается влево на один разряд через бит переноса и загружается в регистр dst. Это циклический сдвиг с пересылкой старшего значащего разряда в разряд переноса, в то время как разряд переноса пересылается в младший значащий разряд.
Циклический левый сдвиг:



Циклы:
1
Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF	– не изменяется.
LV	– не изменяется.
UF	= 0
N	– старший значащий разряд выходного значения.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 0
C	– устанавливается по значению циклически сдвигаемого вовне старшего разряда.

Бит режима:
Операция OVM не зависит от значения бита OVM.

Пример1:
ROLC R3

Перед инструкцией

R3	0000 0420h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	1

После инструкции

R3	00000 0841h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

Приме 2:
ROLC R3

Перед инструкцией

R3	8000 4281h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

R3	0000 8502h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	1

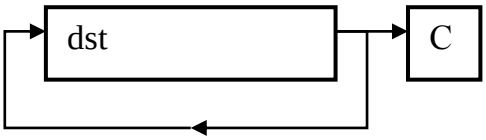
A.5.113 Инструкция ROR

Синтаксис:
ROR dst
Операнды:
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

31 24	23 16	15 8	7 0
0 0 0	1 0 0 1 0 1	1 1	dst	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Поле слова инструкции:
Нет
Операция:
dst циклически сдвинутый вправо на 1 разряд через бит переноса → dst
Описание:
Содержимое операнда dst циклически сдвигается вправо на один разряд и загружается в регистр dst. Младший значащий разряд циклически сдвигается в разряд переноса, а также пересылается в старший значащий разряд.
Циклический правый сдвиг:



Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF	– не изменяется.
LV	– не изменяется.
UF	= 0
N	– старший значащий разряд выходного значения
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 0
C	– устанавливается по значению циклически сдвигаемого вовне старшего разряда.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:
RORR7

Перед инструкцией

R3	0000 00421h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

R3	8000 0210h
LUF	0
LV	0
UF	0
N	1
Z	0
V	0
C	1

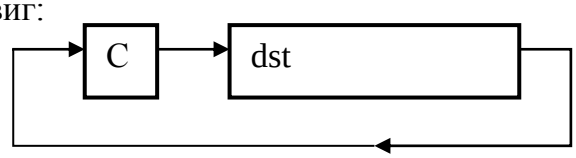
A.5.114 Инструкция RORC

Синтаксис:
RORC dst
Операнды:
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

31 24	23 16	15 8	7 0
0 0 0	1 0 0 1 1 0	1 1	dst	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Поле слова инструкции:
Нет
Операция:
Операнд dst циклически сдвигается вправо на 1 разряд через бит переноса → dst
Описание:
Содержимое операнда dst циклически сдвигается вправо на один разряд через бит переноса регистра состояния и загружается в регистр dst. Это выглядит как 33-разрядный сдвиг. Бит переноса пересылается в старший значащий разряд dst, в то время как младший значащий разряд dst циклически сдвигается в разряд переноса.
Циклический левый сдвиг:



Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF	– не изменяется.
LV	– не изменяется.
UF	= 0
N	– старший значащий разряд выходного значения.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 0
C	– устанавливается по значению циклически сдвигаемого вовне старшего разряда.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:
RORCR4

Перед инструкцией

R4	8000 0081h
LUF	0
LV	0
UF	1
N	0
Z	0
V	0
C	0

После инструкции

R4	4000 0040h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	1

A.5.115 Инструкция RPTB

Синтаксис:
RPTB src
Операнды:
src: 24-разрядная знаковая непосредственная адресация или регистр

Код:
Для 24-битной знаковой непосредственной адресации (смещение):

31 24	23 16	15 8	7 0
0 1 1 0 0 1 0 0	src (смещение)		

Для регистра:

31 24	23 16	15 8	7 0
0 1 1 1 1 0 0 1 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		src

Поле слова инструкции:
Нет
Операция:
src + PC + 1 → RE
1 → ST(RM)
Следующий PC → RS
Описание:
Инструкция RPTB позволяет реализовать многократное повторение блока команд без потерь на установку параметров цикла.
Эта инструкция активизирует режим повторения при изменении PC. Операнд src может быть 32-битным регистром или 24-разрядным непосредственным значением со знаком (смещение). Результирующий адрес src – это конечный адрес блока повторения. Этот адрес загружается в регистр адреса конца повторений (RE). В бит режима повторения регистра состояния [ST(RM)] записывается 1, указывая, что PC будет изменяться в режиме повторения. Адрес следующей инструкции загружается в регистр адреса начала повторения (RS).
RE должно быть больше или равно RS (RE≥RS), иначе блок кода не повторяется, даже если бит RM установлен в 1.
Статусные биты:
LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.
Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
4
Пример:
Нет

A.5.116 Инструкция RPTBD

Синтаксис:
RPTBD src
Операнды:
src: 24-разрядное знаковое непосредственное смещение или регистр

Код:
Для 24-разрядной знаковой непосредственной адресации

31 24	23 16	15 8	7 0
0 1 1 0 0 1 0 1	src (смещение)		

Для регистра:

31 24	23 16	15 8	7 0
0 1 1 1 1 0 0 1 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		src

Поле слова инструкции:
Нет
Операция:
Если src – это непосредственное значение (смещение):

src + PC + 3 → RE,

иначе:

src → RE
1 → ST(RM)
PC RPTBD + 4 → RS

Описание:
Итсрукция RPTBD позволяет блокировать инструкции во время всех повторений без каких либо потерь на многократное повторение и с одноцикловым выполнением RPTBD инструкции. Эта инструкция активизирует режим повторения при изменении PC. Операнд src – 24-разрядное непосредственное значение со знаком. Результирующий адрес загружается в регистр адреса конца повторений (RE). В разряд режима повторения регистра состояния [ST(RM)] записывается 1, указывая, что PC будет изменяться в режиме повторения. Адрес следующей команды +3 загружается в регистр адреса начала повторения (RS).
RE может быть больше или равен RS (RE≥RS), иначе блок кода не повторяется, даже если бит RM установлен в 1.
Инструкция RPTB не переполняет конвейер. Три инструкции не являются частями повторяющегося блока. Регистр RC должет быть загружен перед выполнением инструкции RPTBD. Он не должен загружаться в трех инструкциях после RPTBD.
Прерывания дезактивированы в течение следующих трех инструкций после RPTBD.
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:
Нет

A.5.117 Инструкция RPTS

Синтаксис:

RPTS src

Операнды:

src: основные методы адресации (G)

Код:

31 24	23 16	15 8	7 0
0 0 0 1 0 0 1 1 1 1	G	1 1 0 1 1	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая
01	Прямая
10	Косвенная
11	Непосредственная

Операция:

src → RC

1 → ST(RM)

1 → S

Следующий PC → RS

Следующий PC → RE

Описание:

Инструкция RPTS повторяет отдельную инструкцию несколько раз без потерь на многократное повторение. Выборка инструкции выполняется из регистра инструкций (IR), что предотвращает многократный доступ к памяти.

Операнд src загружается в счетчик повторений (RC). «1» записывается в бит режима повторения (RM) регистра состояния (ST). «1» также записывается в бит повторения одной инструкции (S). Этот бит (S) указывает на то, что выборка инструкции будет выполняться только из регистра инструкций. Следующий адрес PC загружается в регистр адреса конца повторения (RE) и в регистр адреса начала повторения (RS).

Для непосредственного метода адресации операнд src – беззнаковое целое без расширения знака.

Статусные биты:

LUF	– не изменяется.
LV	– не изменяется.
UF	– не изменяется.
N	– не изменяется.
Z	– не изменяется.
V	– не изменяется.
C	– не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Циклы:

4

Пример:

RPTS AR5

Перед инструкцией

PC	123h
ST	0h
RS	0h
RE	0h
RC	0h
AR5	0FFh
LUF	0
LV	0
UF	0
N	0

После инструкции

PC	124h
ST	100h
RS	124h
RE	124h
RC	0FFh
AR5	0FFh
LUF	0
LV	0
UF	0
N	0

Z	0	Z	0
V	0	V	0
C	0	C	0

Примечание – Инструкция RPTS – непрерываемая. Прерывания не произойдут до тех пор, пока не закончится выполнение RPTS. В программах, критичных ко времени, это может привести к неточности их выполнения; таким образом, в программах, критичных ко времени, необходимо использовать инструкцию RPTS с учетом этого факта.

A.5.118 Инструкция RSQRF

Синтаксис:
RSQRF src, dst
Операнды:
src: регистр повышенной точности, прямая или косвенная адресация
dst: регистр повышенной точности
Код:

31 24	23	16	15 8	7 0
0 0 0	1 1 1 0 0 1	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая
01	Прямая
10	Косвенная
	Непосредственная (16 бит)

Операция:
16-разрядная обратная величина квадратного корня от src → dst
Описание:
16-разрядная апроксимированная обратная величина квадратного корня от операнда src загружается в регистр dst. Операнд src предполагается положительным. Операция для отрицательных чисел не определена.
Значения операндов dst и src предполагаются числами с плавающей запятой.
Статусные биты:

LUF – не изменяется.
LV = 1, если введен ноль, иначе не изменяется.
UF = 0
N = 0
Z = 0
V = 1, если введен ноль, иначе = 0
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:

Нет

A.5.119 Инструкция SIGI

Синтаксис:
SIGI src, dst
Операнды:
src: прямая или косвенная адресация (предполагается знаковым целым)
dst: регистровая адресация (предполагается знаковым целым)
Код:

31 24	23	16	15 8	7 0
0 0 0 1 0 1 1 0 0	G	dst	src	

Поле слова инструкции:

G	Методы адресации src
01	Прямая
10	Косвенная

Операция:
LOCK# (или LLOCK#) переходит в низкий уровень
src → dst
LOCK# (или LLOCK#) переходит в высокий уровень
Описание:
Операция блокировки реализуется с помощью сигнала, блокирующего шину (LOCK# или LLOCK#), если и только если выполняется доступ к внешней памяти. Операнды src и dst представляют собой целые со знаком. После чтения сигнал блокировки снимается (сигналы устанавливаются в высокий уровень). Если выполняется доступ к внутренней памяти, то SIGI выполняет чтение, но не устанавливает сигнал блокировки шины в высокий уровень. Смотри раздел 9.7 «Операции блокировки» для получения более детальной информации.
Значения операндов src и dst предполагаются как целые со знаком.
Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF – не изменяется.
LV – не изменяется.
UF = 0
N = 1, если генерируется отрицательный результат, иначе = 0
Z = 1, если генерируется нулевой результат, иначе = 0
V = 0
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:
Нет

A.5.120 Инструкция STF

Синтаксис:
STF src, dst
Операнды:
src: регистровая адресация (Rn: $0 \le n \le 7$)
dst: основные методы адресации

Код:

31 24	23 16	15 8	7 0
0 0 0 1 0 1 0 0 0 0	G	src	dst

Поле слова инструкции:

G	Методы адресации src
01	Прямая
10	Косвенная

Операция:

src → dst

Описание:

Операнд src загружается в память по адресу dst. Операнды src и dst являются числами в формате с ПЗ.

Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Циклы:

1

Пример:

STF R2,@98A1h

Перед инструкцией

DP	80h	
R2	80 983Ch	4.30782204e+01

Данные в 80 98A1h

	0h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

DP	80h	
R2	052 C501 900h	4.30782204e+01

Данные в 80 98A1h

	052 C5019h	4.30782204e+01
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.121 Инструкция STFI

Синтаксис:

STFI src, dst

Операнды:

src: регистровая адресация (R0-R11)
dst: основные методы адресации (G)

Код:

31 24	23 16	15 8	7 0
0 0 0 1 0 1 0 0 0 1	G	src	dst

Поле слова инструкции:

G	Методы адресации src
01	Прямая
10	Косвенная

Операция:

src → dst
сигнализирует об окончании операции блокировки.
Описание:
Операнд src загружается в память по адресу dst. Операция блокировки осуществляется с помощью сигналов LOCK# или LLOCK#. Операнды src и dst являются числами в формате с ПЗ, см. раздел 9.7 «Операции блокировки» для детальной информации.
Статусные биты:

LUF	– не изменяется.
LV	– не изменяется.
UF	– не изменяется.
N	– не изменяется.
Z	– не изменяется.
V	– не изменяется.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:

STFIR3,*-AR4

Перед инструкцией

R3	07 33C0 0000h	1.79750e+02
AR4	80 993Ch	

Данные в 80 993Bh

	0h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

R3	07 33C0 0000h	1.79750e+02
AR4	80 993Ch	

Данные в 80 993Bh

	733 C000h	1.79750e+02
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.122 Инструкция STF || STF

Синтаксис:
STF src2, dst2
|| STF src1, dst1
Операнды:
src1: регистровая адресация (Rn1: 0 ≤ n1 ≤ 7)
dst1: косвенная адресация (смещение = 0, 1, IR0, IR1)
src2: регистровая адресация (Rn2: 0 ≤ n2 ≤ 7)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)
Код:

31	24 23	16	15	8	7	0
1 1	0 0 0 0 0	src2	0 0 0	src1	dst1	dst2

Поле слова инструкции:
Нет
Операция:
src2 → dst2
|| src1 → dst1
Описание:
Две инструкции STF выполняются параллельно. Оба операнда src1 и src2 являются числами в формате с ПЗ.
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:

```
STF  R4, *AR3--
|| STF  R3, *++AR5
```

Перед инструкцией

R4	07 0C80 0000h	1.4050e+02
AR3	80 9835h	
R3	07 33C0 0000h	1.79750e+02
AR5	80 99D2h	

Данные в 80 9835h

	0h

Данные в 80 99D3h

	0h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

AR2	07 0C80 0000h	1.4050e+02
R3	80 9834h	
R7	07 33C0 0000h	1.79750e+02
AR4	80 99D3h	

Данные в 80 9835h

	070C 8000h	1.4050e+02

Данные в 80 98D3h

	0733 C00000h	1.79750e+02
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.123 Инструкция STI

Синтаксис:
STI src, dst
Операнды:
src: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
dst: основные методы адресации (G)
Код:

31 24	23	16	15 8	7 0
0 0 0	1 0 1 0 1 0	G	src	dst

Поле слова инструкции:

G	Методы адресации src
01	Прямая
10	Косвенная

Операция:
src → dst
Описание:
Операнд src загружается в память по адресу dst. Операнды src и dst являются целыми со знаком.
Статусные биты:
LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.

Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:

STI R4,@982Bh

Перед инструкцией

R4	80h	
AR3	42BD7h	273 367
Данные в 80 982Bh		
	0E5FCh	58 876
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR2	80h	
R3	42BD7h	273 367
Данные в 80 982Bh		
	42BD7h	273 367
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.124 Инструкция STII

Синтаксис:
STII src, dst
Операнды:
src: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
dst: основные методы адресации (G)
Код:

31 24	23 16	15 8	7 0
0 0 0	1 0 1 0 1 1	G	src	dst

Поле слова инструкции:

G	Методы адресации src
01	Прямая
10	Косвенная

Операция:
src → dst
Инструкция сообщает об окончании операции блокировки с помощью сигналов LOCK# или LLOCK#.
Описание:
Операнд src загружается в память по адресу dst. Операция блокировки снимается через выводы LOCK# или LLOCK#. Операнды src и dst являются целыми со знаком. Смотри раздел 9.7 «Операции блокировки» для получения более детальной информации.
Статусные биты:
LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.
Бит режима:

Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:

STII R1,@98Aeh

Перед инструкцией

DP	80h	
R1	78Dh	273 367

Данные в 80 98AEh

	25Ch	58 876

После инструкции

DP	80h	
R1	78Dh	

Данные в 80 98AEh

	78Dh	

A.5.125 Инструкция STI || STI

Синтаксис:

STI src2, dst2
|| STI src1, dst1

Операнды:

src1: регистровая адресация (Rn1: $0 \le n1 \le 7$)
dst1: косвенная адресация (смещение = 0, 1, IR0, IR1)
src2: регистровая адресация (Rn2: $0 \le n2 \le 7$)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)

Код:

31	24 23	16	15 8	7 0
1	0 0 0 0 1	src2	0 0 0	src1	dst1	dst2

Поле слова инструкции:

Нет

Операция:

src2 → dst2
|| src1 → dst1

Описание:

Сохранение двух целых выполняется параллельно. Если обе операции сохранения выполняются по одному адресу, то значение записывается как при выполнении операции STI src2, src2.

Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Циклы:

1

Пример:

STI R0,*++AR2(IR0)
|| STI R5,*AR0

Перед инструкцией

R0	0DCh	220
AR2	80 9830h	
IR0	8h	
R5	35h	53
AR0	80 98D3h	

После инструкции

R0	0DCh	220
AR2	80 9838h	
IR0	8h	
R5	35h	53
AR0	80 98D3h	

Данные в 80 9838h

	0h

Данные в 80 99D3h

	0h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

Данные в 80 9838h

	0DCh	220

Данные в 80 98D3h

	35h	53
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.126 Инструкция STIK

Синтаксис:
STIK src, dst
Операнды:
src: 5-разрядное знаковое целое
dst: прямая и косвенная адресация
Код:

31 24	23	16	15 8	7 0
0 0 0 1 0 1 0 1 0	G	src	dst	

Поле слова инструкции:

G	Методы адресации src
00	Прямая
11	Косвенная

Операция:
src → dst
Описание:
Пяти разрядное целое со знаком src загружается в память по адресу dst. Операнды src и dst являются целыми со знаком.
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:

Нет

A.5.127 Инструкция SUBB

Синтаксис:
SUBB src, dst
Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23	16	15 8	7 0
0 0 0 1 0 1 1 0 1	G	dst	src	

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
dst - src - C → dst
Описание:
Разность между операндами dst, src и битом C загружается в регистр dst. Предполагается, что операнды dst являются целыми со знаком.
Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF	– не изменяется.
LV	= 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF	= 0
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении целочисленного переполнения, иначе = 0.
C	= 1 при возникновении заема, иначе = 0.

Бит режима:
Операция OVM зависит от значения бита OVM.
Циклы:
1
Пример:
SUBB *AR5++(4), R5

Перед инструкцией

DP	80 9800h	
R2	0FAh	250

Данные в 80 9800h

	0C7h	199
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	1	

После инструкции

AR5	80 9804h	
R5	032h	50

Данные в 80 9800h

	052 C5019h	100
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.128 Инструкция SUBB3

Синтаксис:
SUBB3 src2, src1, dst
Операнды:
src1, src2: трехоперандные методы адресации первого или второго типа
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:
Тип 1

31 24	23	16	15 8	7 0
0 0 1 0 0 1 1 0 0	T	dst	src1	src2

Тип 2

31 24	23	16	15 8	7 0
0 0 1 1 0 1 1 0 0	T	dst	src1	src2

Поля слова инструкции:
Тип 1

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	Регистровая (любой регистр ЦПУ)
01	Косвенная (смещение=0, 1, IR0, IR1)	Регистровая (любой регистр ЦПУ)
10	Регистровая (любой регистр ЦПУ)	Косвенная (смещение=0, 1, IR0, IR1)
11	Косвенная (смещение=0, 1, IR0, IR1)	Косвенная (смещение=0, 1, IR0, IR1)

Тип 2

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	8-битное знаковое непосредственное
01	Регистровая (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-битное знаковое непосредственное
11	Косвенная *+ARn (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:
src1 - src2 - C → dst
Описание:
Разность между операндами src1 и src2 и битом C (перенос) загружается в регистр dst. Операнды src1, src2 и dst являются целыми со знаком.
Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF	– не изменяется.
LV	= 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF	= 0
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении целочисленного переполнения, иначе = 0.
C	= 1 при возникновении переноса, иначе = 0.

Бит режима:
Операция OVM зависит от значения бита OVM.
Циклы:
1
Пример:
Нет

A.5.129 Инструкция SUBC

Синтаксис:
SUBC src, dst
Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23 16	15 8	7 0
0 0 0 1 0 1 1 1 0	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
Если (dst - src ≥0):
(dst - src << 1) OR 1 → dst
Иначе:
dst << 1 → dst
Описание:
Операнд src вычитается из операнда dst. Операнд dst загружается со значением, зависящим от результата вычитания. Если (dst - src) больше или равно «0», то (dst - src) сдвигается влево на один разряд, младший значащий разряд устанавливается в «1», и результат загружается в регистр dst. Если (dst - src) меньше нуля, то dst сдвигается влево на один разряд и загружается в регистр dst. Операнды dst и src являются целыми без знака.
SUBC может быть использована для выполнения многоразрядного целочисленного деления за один такт.
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример 1:
SUBC @98C5h, R1

Перед инструкцией

DP	80h	
R1	04F6h	1270
Данные в 80 98C5h		
	492h	1170
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

DP	80h	
R1	0C9h	201
Данные в 80 98C5h		
	492h	1170
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

Пример 2:
SUBC 3000, R0 (3000 = 0BB8h)

Перед инструкцией

R0	07D0h	2000

После инструкции

R0	0FA0h	4000

LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	1

LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

A.5.130 Инструкция SUBF

Синтаксис:
SUBF src, dst
Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (R0-R11)
Код:

31 24	23	16	15 8	7 0
0 0 0	1 0 1 1 1 1	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (R0-R11)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
dst - src → dst
Описание:
Разность операнда dst минус операнд src загружается в регистр dst. Операнды dst и src являются числами в формате с ПЗ.
Статусные биты:

LUF	=1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	=1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF	=1 при потере значимости результата с ПЗ, иначе = 0.
N	=1 при генерации отрицательного результата, иначе = 0.
Z	=1 при генерации нулевого результата, иначе = 0.
V	=1 при возникновении ПЗ-переполнения, иначе = 0.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM
Циклы:
1

Пример:
SUBF *AR0--(IR0), R5

Перед инструкцией

AR0	80 9888h	
IR0	80h	
R5	07 33C0 0000h	1.79750000e+02
Данные в 80 9888h		
	70C 8000h	1.4050e+02
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR0	80 9808h	
IR0	80h	
R5	05 1D00 0000h	3.9250e+01
Данные в 80 9888h		
	70C 8000h	1.4050e+02
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.131 Инструкция SUBF3

Синтаксис:
SUBF3 src2, src1, dst
Операнды:
src1, src2: трехоперандные методы адресации первого или второго типа
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:
Тип 1

31 24	23	16	15 8	7 0
0 0 1 0 0 1 1 0 1	T	dst	src1	src2

Тип 2

31 24	23	16	15 8	7 0
0 0 1 1 0 1 1 0 1	T	dst	src1	src2

Поля слова инструкции:
Тип 1

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	Регистровая (любой регистр ЦПУ)
01	Косвенная (смещение=0, 1, IR0, IR1)	Регистровая (любой регистр ЦПУ)
10	Регистровая (любой регистр ЦПУ)	Косвенная (смещение=0, 1, IR0, IR1)
11	Косвенная (смещение=0, 1, IR0, IR1)	Косвенная (смещение=0, 1, IR0, IR1)

Тип 2

T	Методы адресации src1	Методы адресации src2
01	Регистровая (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
11	Косвенная *+ARn (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:
src1 - src2 → dst
Описание:
Разность между операндами src1 и src2 загружается в регистр dst. Операнды dst, src1 и src2 являются числами в формате с ПЗ.
Статусные биты:

LUF = 1 при потере значимости результата с ПЗ, иначе не изменяется.
LV = 1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF = 1 при потере значимости результата с ПЗ, иначе = 0.
N = 1 при генерации отрицательного результата, иначе = 0.
Z = 1 при генерации нулевого результата, иначе = 0.
V = 1 при возникновении ПЗ-переполнения, иначе = 0.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:
Нет

A.5.132 Инструкция SUBF3 || STF

Синтаксис:
SUBF3 src1, src2, dst1
|| STF src3, dst2
Операнды:

src1: регистровая адресация (R0-R7)
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst1: регистровая адресация (R0-R7)
src3: регистровая адресация (R0-R7)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)
Код:

31			24 23		16	15	8	7	0
1 1	1	0 1 0 1	dst1	src1	src3	dst2		src2	

Поле слова инструкции:
Нет
Операция:
src2 - src1 → dst1
|| src3 → dst2
Описание:
Вычитание в формате с ПЗ и сохранение числа в формате с ПЗ выполняется параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STF) считывает из регистра, и операция, которая производится параллельно (SUBF3), записывает в тот же регистр, то STF имеет на входе содержимое регистра до того, как оно было изменено командой SUBF3.
Если src3 и dst1 указывают на один и тот же адрес, то src3 считывается до записи в dst1.
Статусные биты:

LUF	= 1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	= 1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF	= 1 при потере значимости результата с ПЗ, иначе = 0.
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении ПЗ-переполнения, иначе = 0.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:

```
SUBF3 R1,*-AR4 (IR1), R0
|| STF  R7,*+AR5 (IR0)
```

Перед инструкцией

R1	80 9830h	
AR4	80 0000h	
IR1	0h	
R0	0DCh	220
R7	0DCh	220
AR5	80 9833h	
IR0	80 9833h	

Данные в 80 9831h

	9800h

Данные в 80 9883h

	0h	
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	

После инструкции

AR2	80 9831h	
R5	80 0000h	
R2	80 9800h	
R6	0DCh	220
R6	0DCh	220
AR1	80 9882h	
AR1	80 9882h	

Данные в 80 9831h

	9800h

Данные в 80 9883h

	0DCh	220
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	

V	0
C	0

V	0
C	0

A.5.133 Инструкция SUBI

Синтаксис:
SUBI src, dst
Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23	16	15 8	7 0
0 0 0	1 1 0 0 0 0	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
dst - src → dst
Описание:
Разность операнда dst и операнда src згружается в регистр dst. Операнды src и dst являются целыми со знаком.
Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF	– не изменяется.
LV	= 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF	= 0
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	=1 при возникновении целочисленного переполнения, иначе = 0.
C	= 1 при возникновении заема, иначе = 0.

Бит режима:
Операция OVM зависит от значения бита OVM.
Циклы:
1
Пример:
SUBI 220, R7

Перед инструкцией

R7	226h	550
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

R7	14Ah	330
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.134 Инструкция SUBI3

Синтаксис:
SUBI3 src2, src1, dst
Операнды:
src1, src2: трехоперандные методы адресации первого или второго типа
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:
Тип 1

31 24	23	16	15 8	7 0
0 0 1 0 0 1 1 1 0	T	dst	src1	src2

Тип 2

31 24	23	16	15 8	7 0
0 0 1 1 0 1 1 1 0	T	dst	src1	src2

Поля слова инструкции:
Тип 1

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	Регистровая (любой регистр ЦПУ)
01	Косвенная (смещение=0, 1, IR0, IR1)	Регистровая (любой регистр ЦПУ)
10	Регистровая (любой регистр ЦПУ)	Косвенная (смещение=0, 1, IR0, IR1)
11	Косвенная (смещение=0, 1, IR0, IR1)	Косвенная (смещение=0, 1, IR0, IR1)

Тип 2

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	8-битное знаковое непосредственное
01	Регистровая (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-битное знаковое непосредственное
11	Косвенная *+ARn (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:
src1 - src2 → dst
Описание:
Разность операнда src1 и операнда src2 загружается в регистр dst. Операнды src1, src2 и dst являются целыми со знаком.
Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF	– не изменяется.
LV	= 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF	= 0
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении целочисленного переполнения, иначе = 0.
C	= 1 при возникновении заема, иначе = 0.

Циклы:
1
Бит режима:
Операция OVM зависит от значения бита OVM.
Пример:
Нет

A.5.135 Инструкция SUBI3 || STI

Синтаксис:
SUBI3 src1, src2, dst1
|| STI src3, dst2
Операнды:
src1: регистровая адресация (R0-R7)
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst1: регистровая адресация (R0-R7)
src3: регистровая адресация (R0-R7)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)

Код:

31		24 23	16	15 8	7 0
1 1	1 0 1 1 0	dst1	src1	src3	dst2	src2

Поле слова инструкции:
Нет
Операция:
src2 - src1 → dst1
|| src3 → dst2
Описание:
Целочисленное вычитание и сохранение целого производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (SUBI3), записывает в тот же регистр, то STI имеет на входе содержимое регистра до того, как оно было изменено инструкцией SUBI3. Если src3 и dst1 указывают на один и тот же адрес, то src3 считывается до записи в dst1.
Статусные биты:

LUF	– не изменяется.
LV	= 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF	= 0
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении целочисленного переполнения, иначе = 0.
C	= 1 при возникновении заема, иначе = 0.

Бит режима:
Операция OVM зависит от значения бита OVM.
Циклы:
1
Пример:
SUBI3 R7,*+AR2 (IR0), R1
|| STI R3,*++AR7

Перед инструкцией

R0	14h	20
AR2	80 982Fh	
IR0	10h	
R1	0h	
R3	35h	53
AR7	80 983Bh	

Данные в 80 983Fh

	0DCh	220

Данные в 80 993Ch

	0h	
LUF	0	
LV	0	
UF	0	

После инструкции

R7	14h	20
AR2	80 982Fh	
IR0	10h	
R1	0C8h	200
R3	35h	53
AR7	80 983Ch	

Данные в 80 983Fh

	0DCh	220

Данные в 80 983Ch

	35h	53
LUF	0	
LV	0	
UF	0	

N	0		N	0
Z	0		Z	0
V	0		V	0
C	0		C	0

А.5.136 Инструкция SUBRB

Синтаксис:
SUBRB src, dst
Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31	24 23		16	15 8 7 0
0 0 0	1 1 0 0 0 1	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
src - dst - C $\rightarrow$ dst
Описание:
Разность операндов src, dst и C загружается в регистр dst. Операнды dst и src являются целыми со знаком.
Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF	– не изменяется.
LV	= 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF	= 0
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении целочисленного переполнения, иначе = 0.
C	= 1 при возникновении заема, иначе = 0.

Бит режима:
Операция OVM зависит от значения бита OVM.
Циклы:
1

Пример:
SUBRB R4, R6

Перед инструкцией			После инструкции		
R4	03CBh	971	R4	03CBh	971
R6	0258h	600	R6	0172h	370
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	1		C	0	

A.5.137 Инструкция SUBRF

Синтаксис:
SUBRF src, dst
Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (R0-R7)
Код:

31 24	23 16	15 8	7 0
0 0 0 1 1 0 0 1 0	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (R0-R11)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
src - dst → dst
Описание:
Разность операнда src и операнда dst загружается в регистр dst. Операнды dst и src являются числами в формате с ПЗ.
Статусные биты:

LUF	=1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	=1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF	=1 при потере значимости результата с ПЗ, иначе = 0.
N	=1 при генерации отрицательного результата, иначе = 0.
Z	=1 при генерации нулевого результата, иначе = 0.
V	=1 при возникновении ПЗ-переполнения, иначе = 0.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:
SUBRF @9905h, R5

Перед инструкцией

DP	80h	
R5	05 7B40 0000h	6.281250e+01
Данные в 80 9905h		
	733 C000h	1.79750e+02
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

DP	80h	
R5	06 69E0 0000h	1.16937500e+02
Данные в 80 9905h		
	733 C000h	1.79750e+02
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.138 Инструкция SUBRI

Синтаксис:
SUBRI src, dst
Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23	16	15 8	7 0
0 0 0	1 1 0 0 1 1	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
src - dst → dst
Описание:
Разность операнда src и операнда dst загружается в регистр dst. Операнды dst и src являются целыми со знаком.
Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF	– не изменяется.
LV	= 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF	= 0
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении целочисленного переполнения, иначе = 0.
C	= 1 при возникновении заема, иначе = 0.

Циклы:
1
Бит режима:
Операция OVM зависит от значения бита OVM.

Пример:

SUBRI *AR5++(IR0), R3

Перед инструкцией

AR5	80 9900h	
IR0	8h	
R3	0DCh	220
Данные в 80 9905h		
	226h	550
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

DP	80 9908h	
R5	8h	
R5	014Ah	330
Данные в 80 9905h		
	226h	550
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

A.5.139 Инструкция SWI

Синтаксис:
SWI
Операнды:
Нет
Код:

31 24	23 16	15 0
0 1 1 0 0 1 1 0 0	0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Поле слова инструкции:
Нет
Операция:
Выполняет прерывание эмуляции.
Описание:
Инструкция SWI выполняет прерывание эмуляции. Эта инструкция является резервной и не может быть использована при программировании приложений.
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF – не изменяется.
N – не изменяется.
Z – не изменяется.
V – не изменяется.
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
4

Пример:

Нет

A.5.140 Инструкция TOIEEE

Синтаксис:
TOIEEE src, dst
Операнды:
src: регистровая адресация повышенной точности (R0-R11), прямая или косвенная адресация
dst: регистровая адресация повышенной точности
Код:

31 24	23 16	15 8	7 0
0 0 0 0 1 1 0 1 1 1	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая [регистр повышенной точности (R0-R11)]
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
Преобразование src в IEEE формат → dst
Описание:
Операнд src преобразуется из формата с плавающей запятой в дополнительном коде в IEEE формат с плавающей запятой.
Операнд src предполагается числом с плавающей запятой одинарной точности, за исключением режима непосредственной адресации, которое может быть представлено в коротком 16-разрядном формате с плавающей запятой. Преобразованный результат

записывается в 32 старших разряда регистра dst. Иснтрукция STF может быть использована для сохранения результата в памяти.
Статусные биты:

LUF	– не изменяется.
LV	= 1 при возникновении переполнения, в противном случае не меняется.
UF	= 0
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении переполнения, иначе = 0.
C	– не изменяется

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:
Нет

A.5.141 Инструкция TOIEEE||STF

Синтаксис:
TOIEEE src2, dst1
|| STF src3, dst2
Операнды:
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst1: регистровая адресация (R0-R7)
src3: регистровая адресация (R0-R7)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)
Код:

31	24	23	16	15	8	7	0
1 1 1 1 0 0 0	dst1	0 0 0	src3	dst2		src2	

Поля слова инструкции:
Нет
Операция:
Преобразует src2 в IEEE формат → dst1
в параллели с
src3 → dst2
Описание:
Операнд src2 преобразуется из формата с плавающей запятой в дополнительном коде в формат IEEE с плавающей запятой. Операнд src2 предполагается числом с плавающей запятой одинарной точности. Преобразованный результат записывается в 32 старших регистра dst1. Параллельно число с ПЗ сохраняется в памяти.
Если src2 и dst2 указывают на один и тот же адрес, то src2 считывается до записи в dst2.
Статусные биты:

LUF	– не изменяется.
LV	= 1 при возникновении переполнения, в противном случае не меняется.
UF	= 0
N	= 1 при генерации отрицательного результата, иначе = 0.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 1 при возникновении переполнения, иначе = 0.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.

Циклы:
1

Пример:

Нет

A.5.142 Инструкция TRAPcond

Синтаксис:
TRAPcond N
Операнды:
N: метод непосредственной адресации ($0 \le N \le 511$)
Код:

31 24	23 16	15 8	7 0
0 1 1 1 0 1 0	0 0 0 0	cond	0 0 0 0 0 0 0	N

Поля слова инструкции:
Нет
Операция:
Если условие cond – "истина":
ST(GIE) → ST(PGIE)
ST(CF) → ST(PCF)
Следующий PC → *(++SP)
Вектор системного прерывания N → PC
Иначе продолжение выполнения.
Описание:
Если условие истинно, то GIE и CF сохраняются в PGIE и PCF регистра состояния, все прерывания запрещены (0 →GIE), и кэш «заблокирован» (1 → CF). Затем содержимое PC заталкивается в системный стэк, и в PC загружается содержимое определенного вектора программных прерываний (N). Если условие ложно, тогда продолжается выполнение программы.
Если прерывания имеют вложенность, то может понадобиться сохранить регистр состояния перед выполнением TRAPcond.
Статусные биты:

GIE	– устанавливается 0, если TRAP выполняется.
LUF	– не изменяется.
LV	– не изменяется.
UF	– не изменяется.
N	– не изменяется.
Z	– не изменяется.
V	– не изменяется.
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
5

Пример:

Нет

A.5.143 Инструкция TSTB

Синтаксис:
TSTB src, dst
Операнды:

src: основные методы адресации (G)
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23 16	15 8	7 0
0 0 0 1 1 0 1 0 0 G	dst	src	

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр в основном регистровом файле ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
dst AND src
Описание:
Поразрядное логическое «И» операндов dst и src формируется, но результат не загружается ни в один регистр, таким образом, обеспечивается сравнение без потери информации. Предполагается, что операнды являются беззнаковыми целыми.
Статусные биты:
Флаги состояния изменяются для всех конечных регистров.

LUF – не изменяется.
LV – не изменяется.
UF = 0
N – старший значащий разряд выходного значения.
Z = 1 при генерации нулевого результата, иначе = 0.
V = 0
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
П р и м е р:

```
TSTB *-AR4(1),R5
```

Перед инструкцией

AR4	80 99C5h	
R5	898h	2200
Данные в 80 993Ch		
	767h	1895
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR4	80 99C5h	
R5	80 982Fh	2200
Данные в 80 99C4h		
	767h	1895
LUF	0	
LV	0	
UF	0	
N	0	
Z	1	
V	0	
C	0	

A.5.144 Инструкция TSTB3

Синтаксис:
TSTB3 src2, src1
Операнды:
src1, src2: трехоперандные методы адресации первого или второго типа
Тип 1

31 24 23		16	15 8	7 0
0 0 1 0 0 1 1 1 1	T	0 0 0 0 0	src1	src2

Тип 2

31 24 23		16	15 8	7 0
0 0 1 1 0 1 1 1 1	T	0 0 0 0 0	src1	src2

Поля слова инструкции:
Тип 1

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	Регистровая (любой регистр ЦПУ)
01	Косвенная (смещение=0, 1, IR0, IR1)	Регистровая (любой регистр ЦПУ)
10	Регистровая (любой регистр ЦПУ)	Косвенная (смещение=0, 1, IR0, IR1)
11	Косвенная (смещение=0, 1, IR0, IR1)	Косвенная (смещение=0, 1, IR0, IR1)

Тип 2

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	8-битное знаковое непосредственное
01	Регистровая (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-битное знаковое непосредственное
11	Косвенная *+ARn (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:
src1 AND src2
Описание:
Поразрядное логическое «И» операндов src1 и src1 формируется, но результат не загружается ни в один регистр, таким образом, обеспечивается сравнение без потери информации. Предполагается, что операнды являются беззнаковыми целыми. Src2 в методе непосредственной адресации расширен знаком.
Хотя данная инструкция имеет только два операнда, она обозначается как трехоперандная, т. к. операнды определены в трехоперандном формате.
Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF = 0
N – старший значащий разряд выходного значения.
Z = 1 при генерации нулевого результата, иначе = 0.
V = 0
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1
Пример:
Нет

A.5.145 Инструкция XOR

Синтаксис:
XOR src, dst
Операнды:
src: основные методы адресации (G)
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)
Код:

31 24	23 16	15 8	7 0
0 0 0 1 1 0 1 0 1	G	dst	src

Поле слова инструкции:

G	Методы адресации src
00	Регистровая (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
dst XOR src → dst
Описание:
Поразрядное «ИСКЛЮЧАЮЩЕЕ ИЛИ» операндов src и dst загружается в регистр dst. Операнды dst и src являются беззнаковыми целыми.
Статусные биты:
Если ST (SET COND) = 0 и конечным является один из регистров (R0-R11), то флаги состояний изменяются. Если ST (SET COND) = 1, то они изменяются для всех конечных регистров.

LUF – не изменяется.
LV – не изменяется.
UF = 0
N – старший значащий разряд выходного значения.
Z = 1 при генерации нулевого результата, иначе = 0.
V = 0
C – не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:
XOR R1, R2

Перед инструкцией

R1	0F FA32h
R2	0F F5C1h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

R1	0F F412h
R2	00 0FF3h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

A.5.146 Инструкция XOR3

Синтаксис:
XOR3 src2, src1, dst
Операнды:
src1, src2: трехоперандные методы адресации первого или второго типа
dst: регистровая адресация (любой регистр из основного регистрового файла ЦПУ)

Код:

Тип 1

31 24	23	16	15 8	7 0
0 0 1 0 1 0 0 0 0	T	dst	src1	src2

Тип 2

31 24	23	16	15 8	7 0
0 0 1 1 1 0 0 0 0	T	dst	src1	src2

Поля слова инструкции:

Тип 1

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	Регистровая (любой регистр ЦПУ)
01	Косвенная (смещение=0, 1, IR0, IR1)	Регистровая (любой регистр ЦПУ)
10	Регистровая (любой регистр ЦПУ)	Косвенная (смещение=0, 1, IR0, IR1)
11	Косвенная (смещение=0, 1, IR0, IR1)	Косвенная (смещение=0, 1, IR0, IR1)

Тип 2

T	Методы адресации src1	Методы адресации src2
00	Регистровая (любой регистр ЦПУ)	8-битное знаковое непосредственное
01	Регистровая (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-битное знаковое непосредственное
11	Косвенная *+ARn (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:

src1 XOR src2 → dst

Описание:

Поразрядное «ИСКЛЮЧАЮЩЕЕ ИЛИ» между операндами src1 и src2 загружается в регистр dst. Операнды src1, src2 и dst являются беззнаковыми целыми. Src2 в методе непосредственной адресации расширен знаком.

Статусные биты:

LUF – не изменяется.
LV – не изменяется.
UF = 0
N – старший значащий разряд выходного значения.
Z = 1 при генерации нулевого результата, иначе = 0.
V = 0
C – не изменяется.

Бит режима:

Операция OVM не зависит от значения бита OVM.

Циклы:

1

Пример:

Нет

A.5.147 Инструкция XOR3 || STI

Синтаксис:

XOR3 src2, src1, dst1
|| STI src3, dst2

Операнды:

src1: регистровая адресация (R0-R7)
src2: косвенная адресация (смещение = 0, 1, IR0, IR1)
dst1: регистровая адресация (R0-R7)
src3: регистровая адресация (R0-R7)
dst2: косвенная адресация (смещение = 0, 1, IR0, IR1)

Код:

31	24 23	16	15	8 7	0
1 1	1 0 1 1 1	dst1	src1	src3	dst2	src2

Поле слова инструкции:
Нет
Операция:
src1 XOR src2 → dst1
|| src3 → dst2
Описание:
Поразрядное «ИСКЛЮЧАЮЩЕЕ ИЛИ» и сохранение целого выполняется параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (XOR3), записывает в тот же регистр, то STI имеет на входе содержимое регистра до того, как оно было изменено командой XOR3.
Если src2 и dst2 указывают на один и тот же адрес, то src2 считывается до записи в dst2.
Статусные биты:

LUF	– не изменяется.
LV	– не изменяется.
UF	= 0
N	– старший значащий разряд выходного значения.
Z	= 1 при генерации нулевого результата, иначе = 0.
V	= 0
C	– не изменяется.

Бит режима:
Операция OVM не зависит от значения бита OVM.
Циклы:
1

Пример:

```
XOR3 *AR1++, R3, R3
|| STI  R6,*-AR2 (IR0)
```

Перед инструкцией

AR1	80 987Eh	
R3	85h	
R6	0DCh	220
AR2	80 98B4h	
IR0	8h	

Данные в 80 983Fh

	85h

Данные в 80 993Ch

	0h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

AR1	80 987Fh	20
R3	0h	
R6	0DCh	220
AR2	80 98B4h	
IR0	8h	

Данные в 80 983Fh

	85h	220

Данные в 80 983Ch

	0DCh
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

Приложение Б
(обязательное)

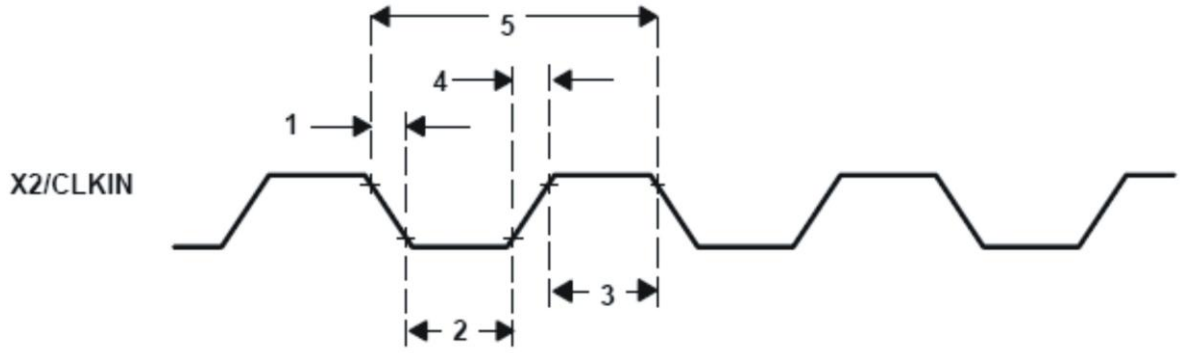
Временные параметры сигналов

Временные параметры сигналов ИС 1867ВМ9Ф представлены в таблицах Б.1 – Б.19 и на рисунках Б.1 – Б.21.

Таблица Б.1 – Временные параметры сигналов X2/CLKIN, H1 и H3 (смотри рисунки Б.1, Б.2)

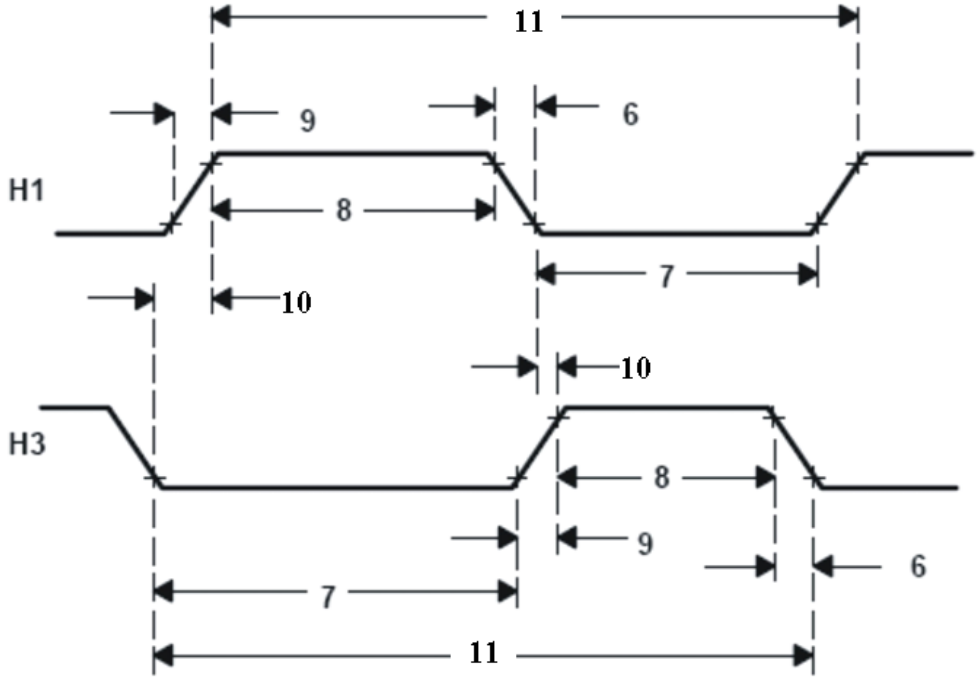
Номер параметра*	Наименование параметра, буквенное обозначение параметра	Значение параметра		Единица измерения
		минимум	максимум	
1	Время спада сигнала X2/CLKIN, $t_{f(CI)}$	1,250		нс
2	Время низкого уровня сигнала X2/CLKIN, $t_{w(CIL)}$ при $t_{c(CI)}$ = min	2,125		нс
3	Время высокого уровня сигнала X2/CLKIN, $t_{w(CIH)}$ при $t_{c(CI)}$= min	2,125		нс
4	Время нарастания фронта сигнала X2/CLKIN, $t_{r(CI)}$	1,250		нс
5	Время цикла сигнала X2/CLKIN, $t_{c(CI)}$	6,250	242,500	нс
6	Время спада сигналов H1, H3, $t_{f(H)}$	3,000		нс
7	Время низкого уровня сигналов H1, H3, $t_{w(HL)}$	$t_{c(CI)}$–1,250	$t_{c(CI)}$+1,250	нс
8	Время высокого уровня сигналов H1, H3, $t_{w(HH)}$	$t_{c(CI)}$–1,250	$t_{c(CI)}$+1,500	нс
9	Время нарастания фронта сигналов H1, H3, $t_{r(H)}$	0,250		нс
10	Время задержки от низкого уровня сигнала H1 до высокого уровня сигнала H3 (или от низкого уровня сигнала H3 до высокого уровня сигнала H1), $t_{d(H1L–H3H)}$ ($t_{d(H3L–H1H)}$)	–0,250	1,000	нс
11	Время цикла сигналов H1, H3, $t_{c(H)}$	12,500	485,000	нс

_____
* Соответствующей цифрой обозначен параметр на рисунках Б.1, Б.2.



Примечание – Цифрами обозначены номера строк таблицы Б.1, содержащих соответствующие параметры.

Рисунок Б.1 – Временная диаграмма сигнала X2/CLKIN



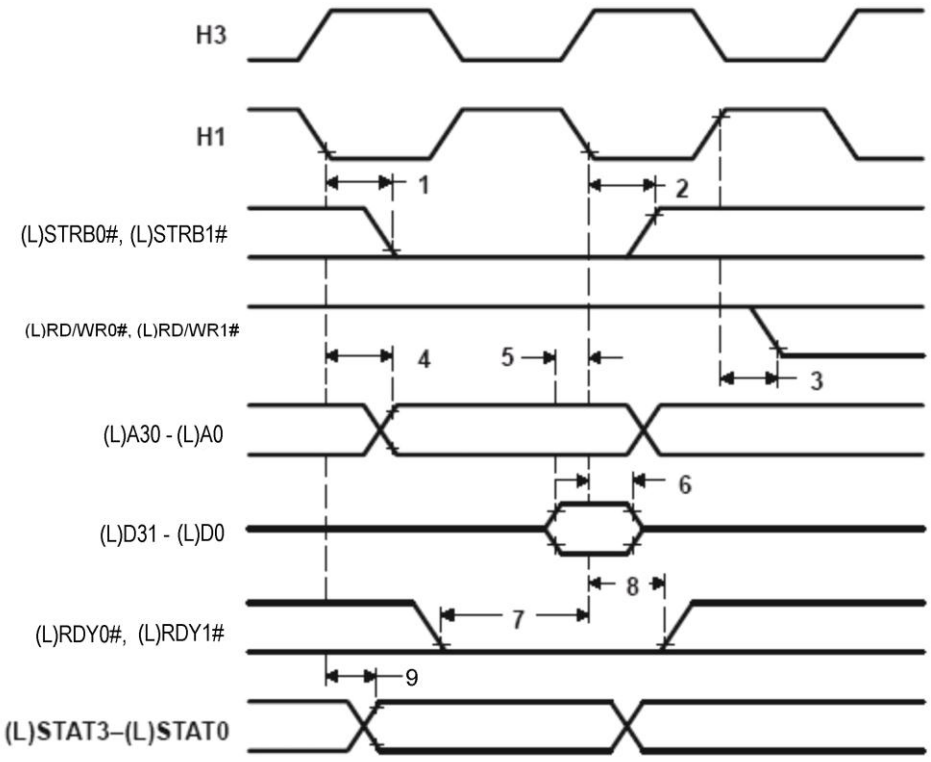
Примечание – Цифрами обозначены номера строк таблицы Б.1, содержащих соответствующие параметры.

Рисунок Б.2 – Временная диаграмма сигналов Н1 и Н3

Таблица Б.2 – Временные параметры для чтения/записи памяти при (L)STRBx# = 0 (смотри рисунки Б.3, Б.4)

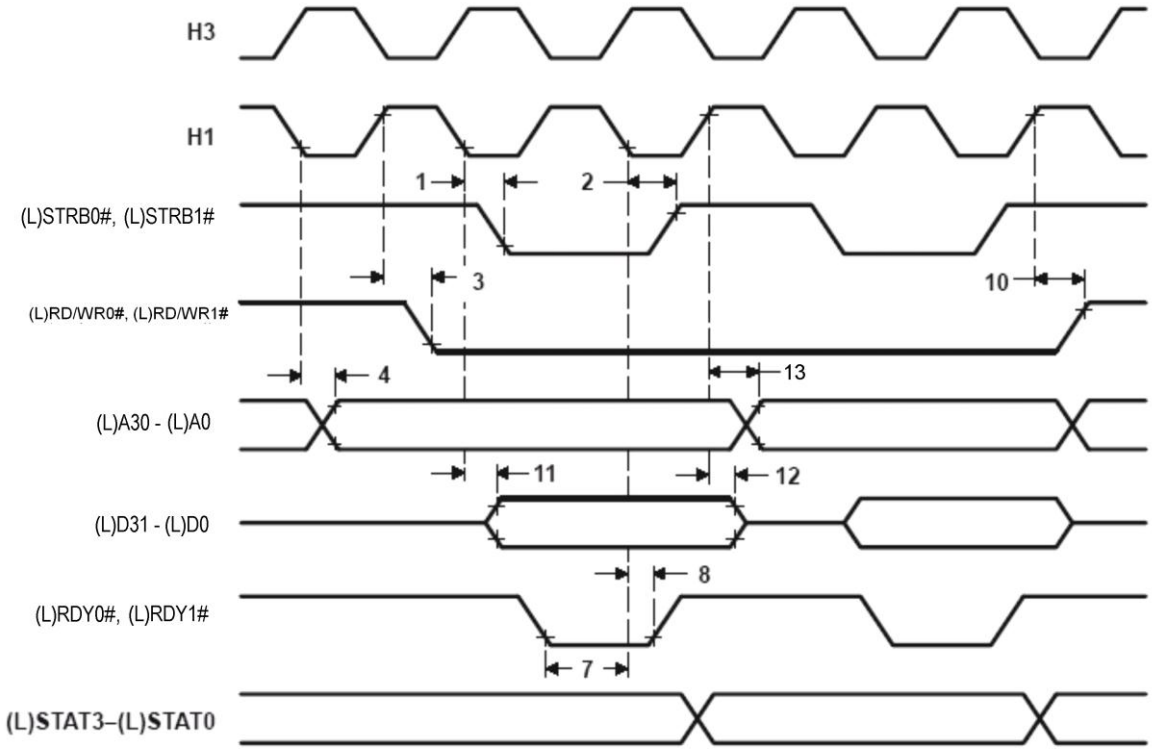
Номер параметра*	Наименование параметра, буквенное обозначение параметра	Значение параметра		Единица измерения
		минимум	максимум	
1	Время задержки от низкого уровня сигнала H1 до низкого уровня сигналов (L)STRB0#, (L)STRB1#, $t_{d[H1L–(L)SL]}$	0,00	2,50	нс
2	Время задержки от низкого уровня сигнала H1 до высокого уровня сигналов (L)STRB0#, (L)STRB1#, $t_{d[H1L–(L)SH]}$	0,00	2,50	нс
3	Время задержки от высокого уровня сигнала H1 до низкого уровня сигналов (L)R/W0#, (L)R/W1#, $t_{d[H1H–(L)R/WL]}$	0,00	2,25	нс
4	Время задержки от низкого уровня сигнала H1 до установившегося значения сигналов (L)A30-(L)A0, $t_{d[H1L–(L)AV]}$	0,00	2,50	нс
5	Время установки значения сигналов (L)D31-(L)D0 перед низким уровнем сигнала H1 (чтение), $t_{su[(L)D–H1L]R}$	3,70		нс
6	Время удержания сигналов (L)D31-(L)D0 после низкого уровня сигнала H1 (чтение), $t_{h[H1L–(L)D]R}$	0,00		нс
7	Время установки значения сигналов (L)RDY0#, (L)RDY1# перед низким уровнем сигнала H1, $t_{su[(L)RDY–H1L]}$	6,25		нс
8	Время удержания сигналов (L)RDY0#, (L)RDY1# после низкого уровня сигнала H1, $t_{h[H1L–(L)RDY]}$	0,00		нс
9	Время задержки от низкого уровня сигнала H1 до установившегося значения сигналов (L)STAT3-(L)STAT0, $t_{d[H1L–(L)STV]}$	2,50		нс
10	Время задержки от высокого уровня сигнала H1 до значения сигналов (L)R/W0#, (L)R/W1# высокого уровня (запись), $t_{d[H1H–(L)R/WH]W}$	2,25		нс
11	Время удержания сигналов (L)D31-(L)D0 после низкого уровня сигнала H1 (запись), $t_{h[H1L–(L)D]W}$	4,00		нс
12	Время удержания сигналов (L)D31-(L)D0 после высокого уровня сигнала H1 (запись), $t_{h[H1H–(L)D]W}$	0,00		нс
13	Время задержки от высокого уровня сигнала H1 до установившегося значения адреса сигналов (L)A30-(L)A0 при циклах записи, $t_{d[H1H–(L)A]W}$	3,25		нс

* Соответствующей цифрой обозначен параметр на рисунках Б.3, Б.4.



Примечание – Цифрами обозначены номера строк таблицы Б.2, содержащих соответствующие параметры.

Рисунок Б.3 – Временная диаграмма для чтения памяти при (L)STRBx# = 0



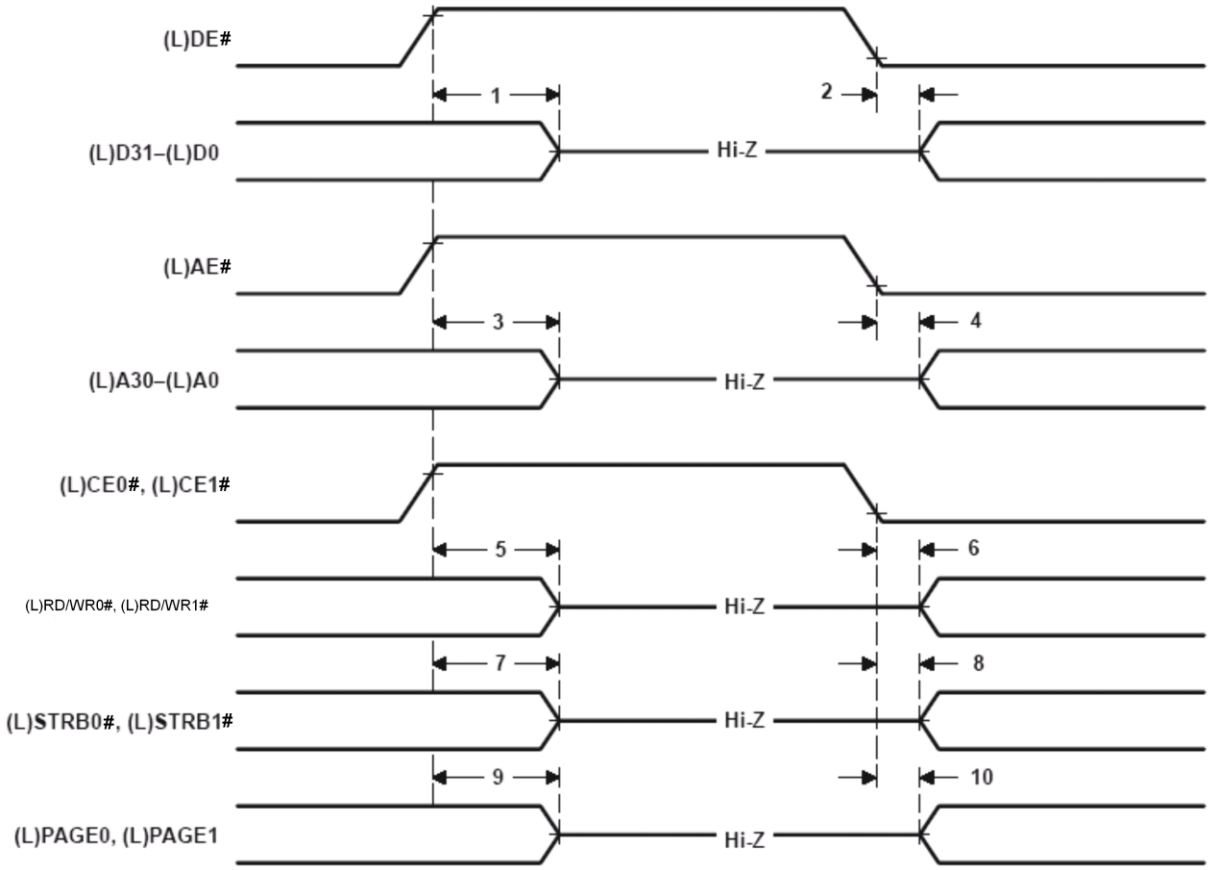
Примечание – Цифрами обозначены номера строк таблицы Б.2, содержащих соответствующие параметры.

Рисунок Б.4 – Временная диаграмма для записи памяти при (L)STRBx# = 0

Таблица Б.3 – Временные параметры сигналов (L)DE#, (L)AE#, (L)CEx# (смотри рисунок Б.5)

Номер параметра*	Наименование параметра, буквенное обозначение параметра	Значение параметра		Единица измерения
		минимум	максимум	
1	Время задержки от высокого уровня сигнала (L)DE# до сигналов (L)D31-(L)D0 в высокоимпедансном состоянии, $t_{d[(L)DEH–(L)DZ]}$	0,00	3,75	нс
2	Время задержки от низкого уровня сигнала (L)DE# до установившегося значения сигналов (L)D31-(L)D0, $t_{d[(L)DEL–(L)DV]}$	0,00	5,5	
3	Время задержки от высокого уровня сигнала (L)AE# до высокоимпедансного состояния сигналов (L)A31-(L)A0, $t_{d[(L)AEH–(L)AZ]}$	0,00	3,75	
4	Время задержки от низкого уровня сигнала (L)AE# до установившегося значения сигналов (L)A31-(L)A0, $t_{d[(L)AEL–(L)AV]}$	0,00	5,25	
5	Время задержки от высокого уровня сигналов (L)CE0#, (L)CE1# до высокоимпедансного состояния сигналов (L)R/W0#, (L)R/W1#, $t_{d[(L)CEH–(L)R/WZ]}$	0,00	3,75	
6	Время задержки от низкого уровня сигналов (L)CE0#, (L)CE1# до установившегося значения сигналов (L)R/W0#, (L)R/W1#, $t_{d[(L)CEL–(L)R/WV]}$	0,00	5,25	
7	Время задержки от высокого уровня сигналов (L)CE0#, (L)CE1# до высокоимпедансного состояния сигналов (L)STRB0#, (L)STRB1#, $t_{d[(L)CEH–(L)SZ]}$	0,00	3,75	
8	Время задержки от низкого уровня сигналов (L)CE0#, (L)CE1# до установившегося значения сигналов (L)STRB0#, (L)STRB1#, $t_{d[(L)CEL–(L)SV]}$	0,00	5,25	
9	Время задержки от высокого уровня сигналов (L)CE0#, (L)CE1# до высокоимпедансного состояния сигналов (L)PAGE0, (L)PAGE1, $t_{d[(L)CEH–(L)PAGEZ]}$	0,00	3,75	
10	Время задержки от низкого уровня сигналов (L)CE0#, (L)CE1# до установившегося значения сигналов (L)PAGE0, (L)PAGE1, $t_{d[(L)CEL–(L)PAGEV]}$	0,00	5,25	

* Соответствующей цифрой обозначен параметр на рисунке Б.5.



Примечания

1 Цифрами обозначены номера строк таблицы Б.3, содержащих соответствующие параметры.

2 Hi – Z – обозначение высокоимпедансного состояния.

Рисунок Б.5 – Временная диаграмма сигналов (L)DE#, (L)AE#, (L)CEx#

Таблица Б.4 – Временной параметр для сигнала (L)LOCK# при выполнении инструкций LDFI или LDII (смотри рисунок Б.6)

Номер параметра*	Наименование параметра, буквенное обозначение параметра	Значение параметра		Единица измерения
		минимум	максимум	
1	Время задержки от сигнала H1 низкого уровня до сигнала (L)LOCK# низкого уровня, $t_{d[H1L–(L)LOCKL]}$	2,75		нс

_____

* Соответствующей цифрой обозначен параметр на рисунке Б.6.

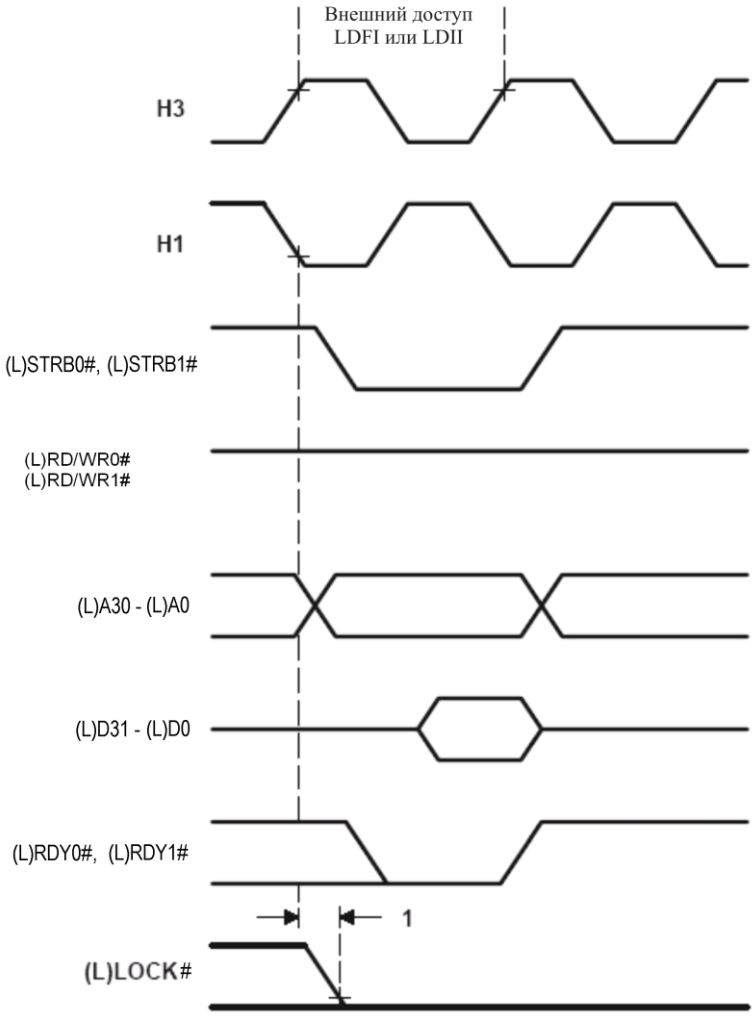


Рисунок Б.6 – Временная диаграмма для сигнала (L)LOCK# при выполнении инструкций LDFI или LDII

Таблица Б.5 – Временной параметр для сигнала (L)LOCK# при выполнении инструкций STFI или STII (смотри рисунок Б.7)

Номер параметра*	Наименование параметра, буквенное обозначение параметра	Значение параметра		Единица измерения
		минимум	максимум	
1	Время задержки от сигнала H1 низкого уровня до сигнала (L)LOCK# высокого уровня, $t_{d(H1L-(L)LOCKH)}$	3,75		нс

* Соответствующей цифрой обозначен параметр на рисунке Б.7.

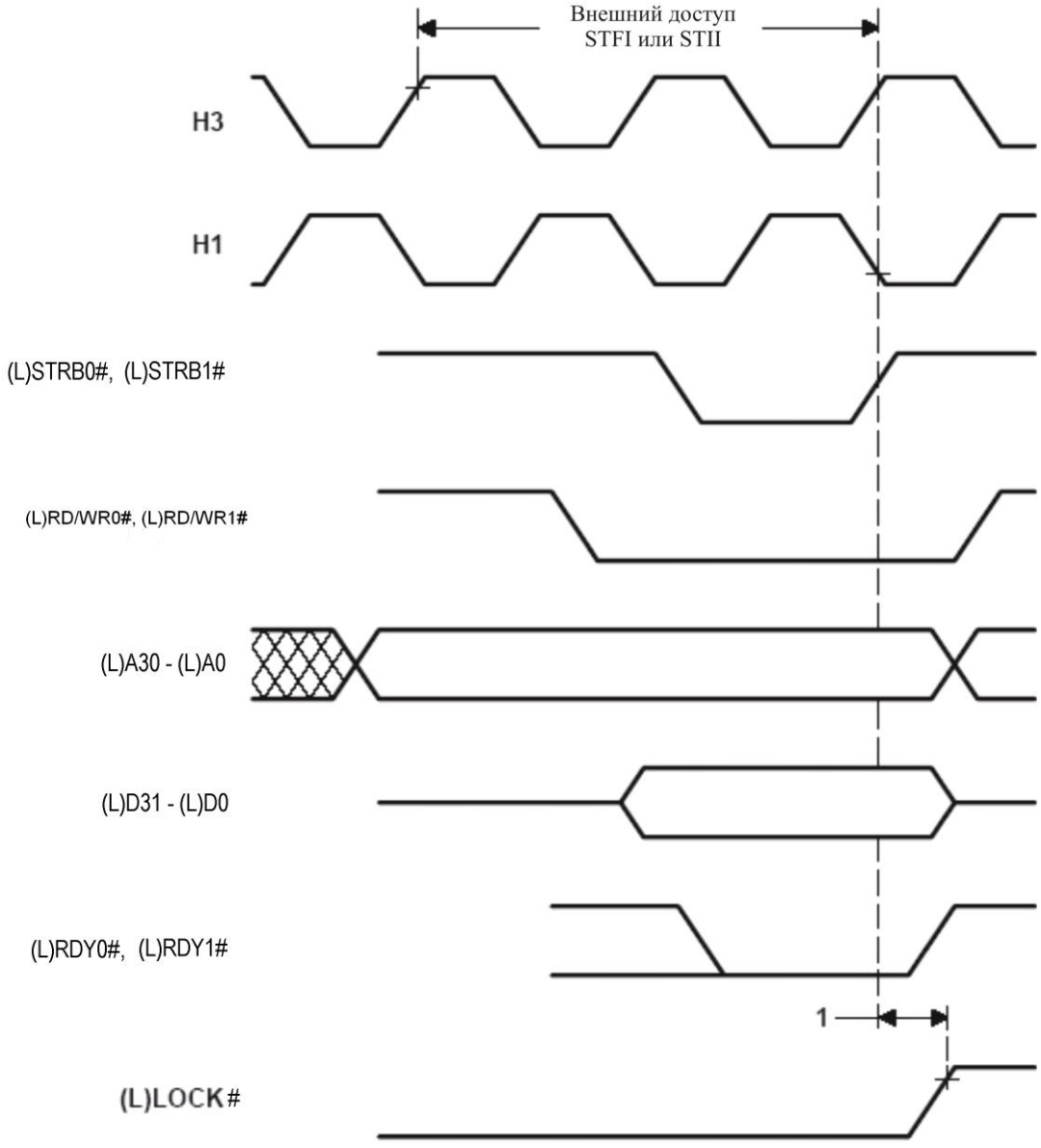


Рисунок Б.7 – Временная диаграмма для сигнала (L)LOCK# при выполнении инструкций STFI или STII

Таблица Б.6 – Временные параметры для сигнала (L)LOCK# при выполнении инструкции SIGI (смотри рисунок Б.8)

Номер параметра*	Наименование параметра, буквенное обозначение параметра	Значение параметра		Единица измерения
		минимум	максимум	
1	Время задержки от низкого уровня сигнала H1 до низкого уровня сигнала (L)LOCK#, $t_{d[H1L–(L)LOCKL]}$	2,75		нс
2	Время задержки от низкого уровня сигнала H1 до высокого уровня сигнала (L)LOCK#, $t_{d[H1L–(L)LOCKH]}$	2,75		нс

_____

* Соответствующей цифрой обозначен параметр на рисунке Б.8.

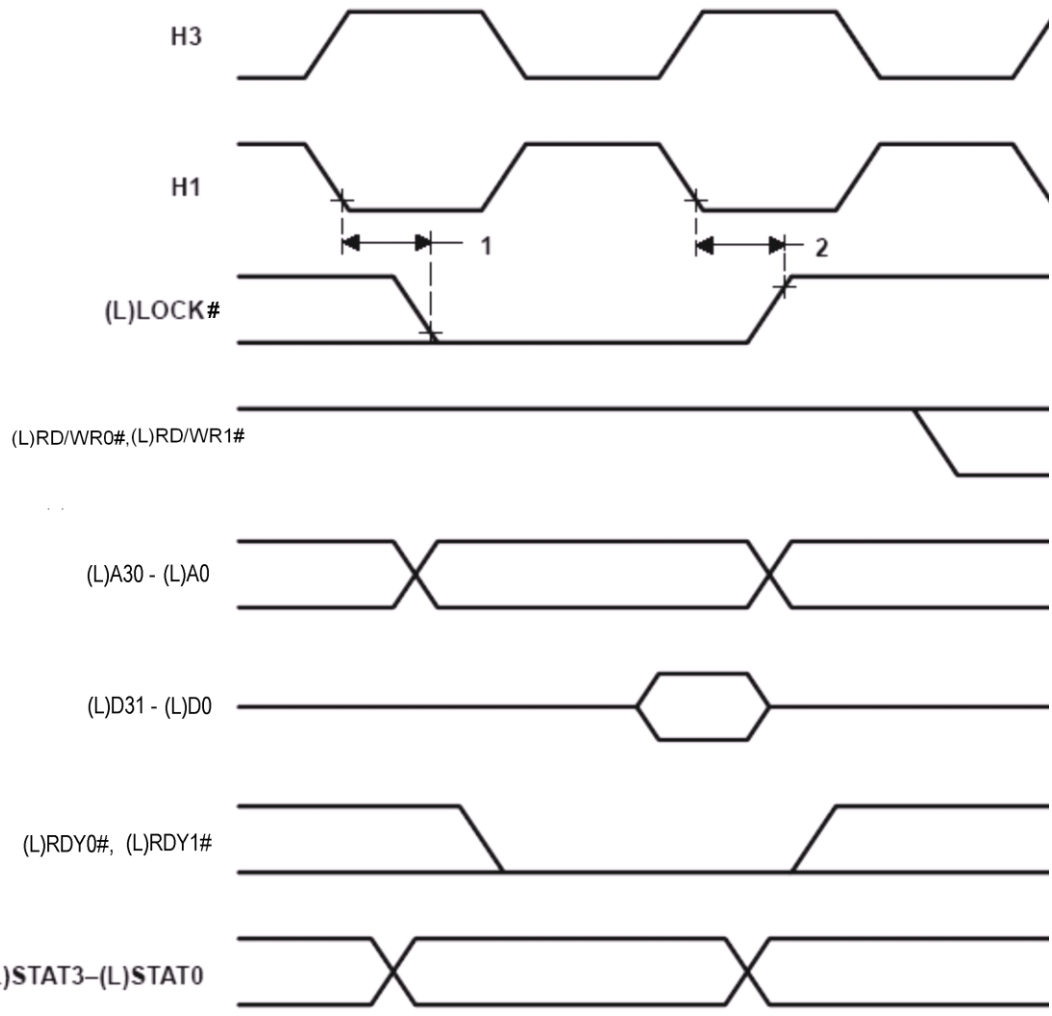


Рисунок Б.8 – Временная диаграмма для сигнала (L)LOCK# при выполнении инструкции SIGI

Таблица Б.7 – Временные параметры для сигналов (L)PAGE0, (L)PAGE1 во время доступа памяти на другую страницу (смотри рисунок Б.9)

Номер параметра*	Наименование параметра, буквенное обозначение параметра	Значение параметра		Единица измерения
		минимум	максимум	
1	Время задержки от низкого уровня сигнала H1 до высокого уровня сигналов (L)PAGE0, (L)PAGE1 при доступе памяти к другой странице, $t_{d[H1L-(L)PAGEH]}$	0,0	2,5	нс
2	Время задержки от низкого уровня сигнала H1 до низкого уровня сигналов (L)PAGE0, (L)PAGE1 при доступе памяти к другой странице, $t_{d[H1L-(L)PAGEL]}$	0,0	2,5	нс

_____

* Соответствующей цифрой обозначен параметр на рисунке Б.9.

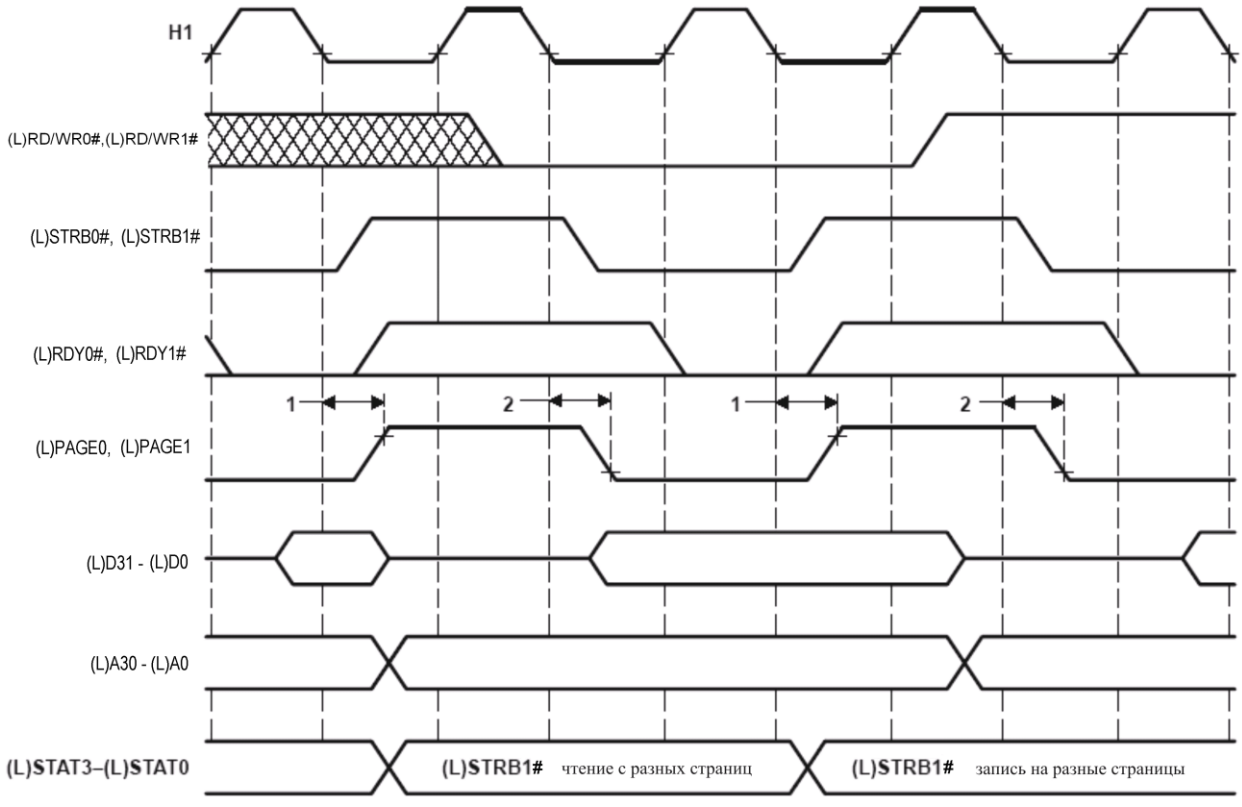


Рисунок Б.9 – Временная диаграмма для сигналов (L)PAGE0, (L)PAGE1 во время доступа памяти на другую страницу

Таблица Б.8 – Временные параметры установки сигналов IIOF3#-IIOF0# (выводы IIOF3#-IIOF0# установлены в качестве выходов) при записи в регистр IIF (смотри рисунок Б.10)

Номер параметра*	Наименование параметра, буквенное обозначение параметра	Значение параметра		Единица измерения
		минимум	максимум	
1	Время установившегося значения выходов IIOF3#-IIOF0# после низкого уровня сигнала H1, $t_{H1L–IIOFV}$	4,5		нс

_____

* Соответствующей цифрой обозначен параметр на рисунке Б.10.

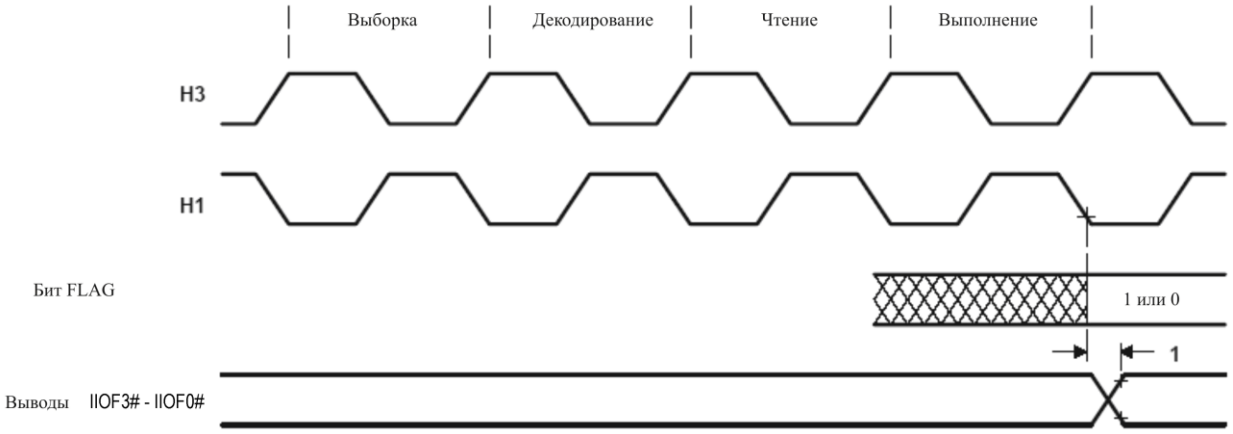


Рисунок Б.10 – Временная диаграмма установки сигналов IIOFx# (выводы IIOFx# установлены в качестве выходов) при записи в регистр IIF

Таблица Б.9 – Временные параметры сигналов IIOF3#-IIOF0# при переходе из режима выхода в режим входа (смотри рисунок Б.11)

Номер параметра*	Наименование параметра, буквенное обозначение параметра	Значение параметра		Единица измерения
		минимум	максимум	
1	Время удержания сигналов IIOF3#-IIOF0# после низкого уровня сигнала H1, $t_{h(H1L–IIOF)}$	3,50		нс
2	Время установки сигналов IIOF3#-IIOF0# перед низким уровнем сигнала H1, $t_{su(IIOF)}$	2,75		нс
3	Время удержания сигналов IOF3#-IIOF0# после низкого уровня сигнала H1, $t_{h(IIOF)}$	0,00		нс

_____

* Соответствующей цифрой обозначен параметр на рисунке Б.11.

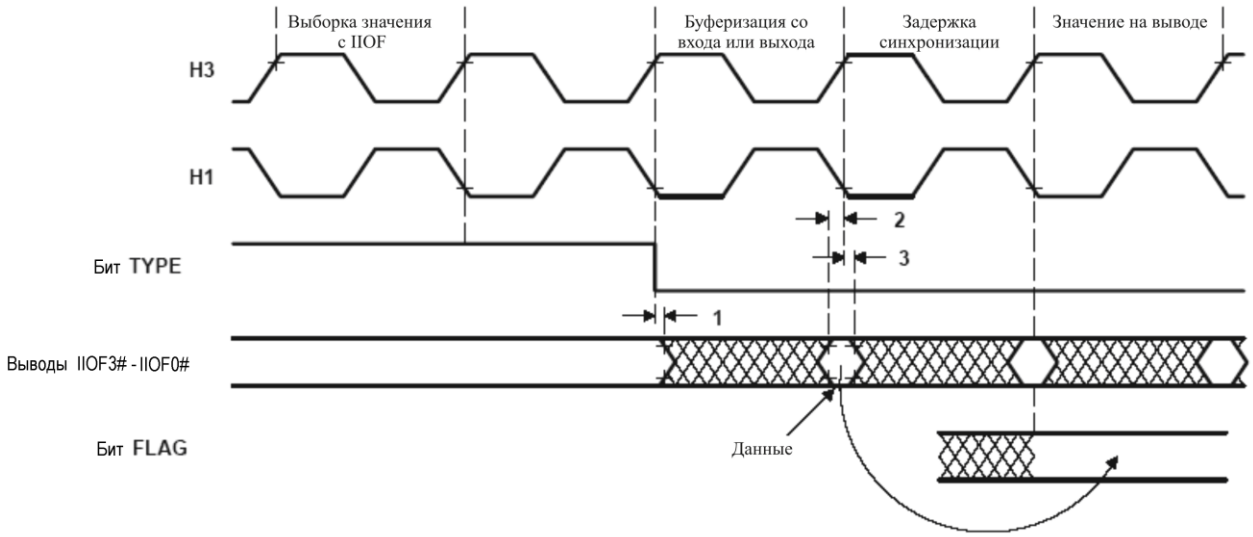


Рисунок Б.11 – Временная диаграмма сигналов IIOF3#-IIOF0# при переходе из режима выхода в режим входа

Таблица Б.10 – Временные параметры сигналов IIOF3#-IIOF0# при переходе из режима входа в режим выхода (смотри рисунок Б.12)

Номер параметра*	Наименование параметра, буквенное обозначение параметра	Значение параметра		Единица измерения
		минимум	максимум	
1	Время задержки от низкого уровня сигнала H1 до переключения сигналов IIOF3#-IIOF0#, $t_{d(H1L–IIOF)}$	4,0		нс

_____

* Соответствующей цифрой обозначен параметр на рисунке Б.12.

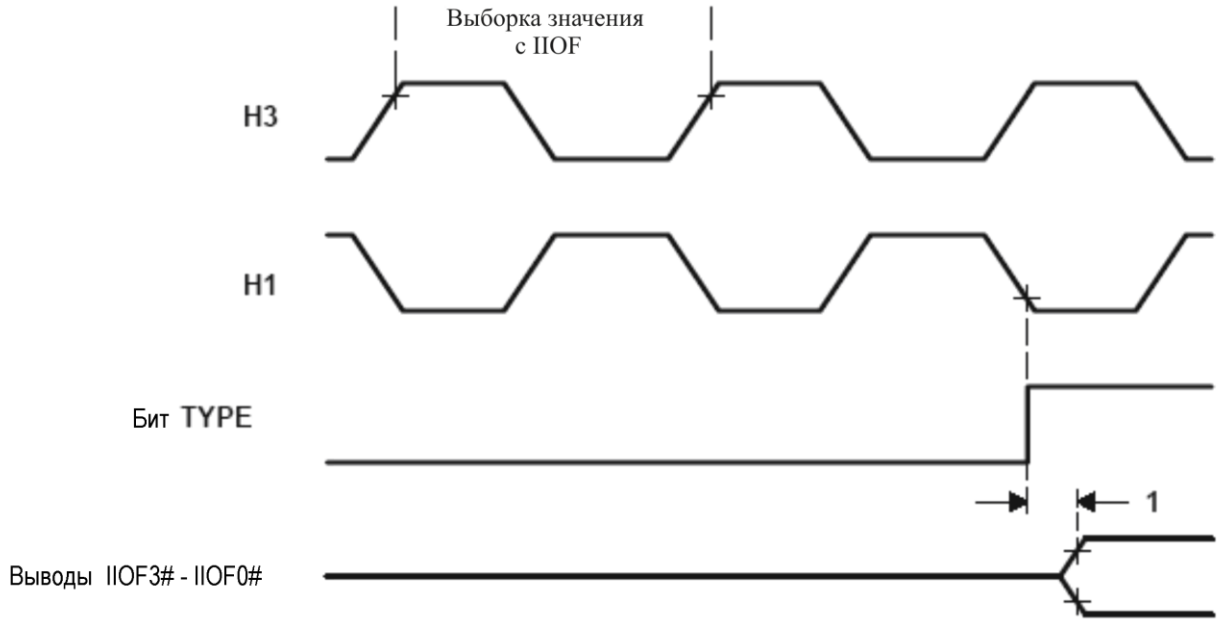
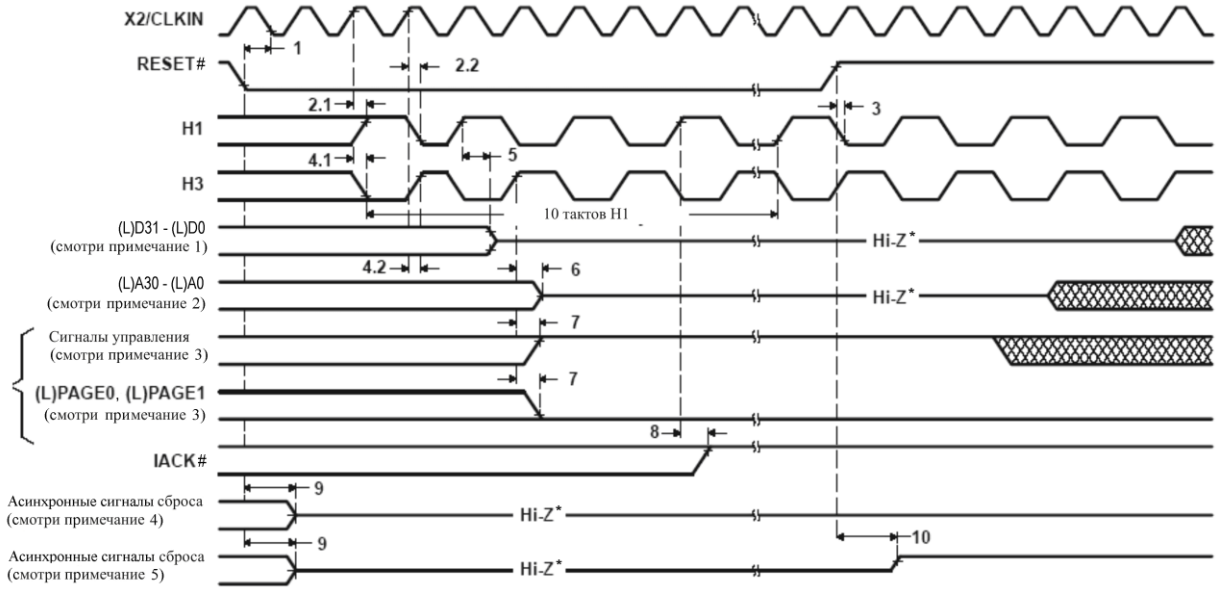


Рисунок Б.12 – Временная диаграмма сигналов IIOF3#-IIOF0# при переходе из режима входа в режим выхода

Таблица Б.11 – Временные параметры сигнала RESET# (смотри рисунок Б.13)

Номер параметра*	Наименование параметра, буквенное обозначение параметра	Значение параметра		Единица измерения
		минимум	максимум	
1	Время установки сигнала RESET# перед низким уровнем сигнала X2/CLKIN, $t_{su(RESET–CIL)}$	2,75	$t_{c(CI)}$	нс
2.1	Время задержки от высокого уровня сигнала X2/CLKIN до высокого уровня сигнала H1, $t_{d(CIH–H1H)}$	0,5	3	нс
2.2	Время задержки от высокого уровня сигнала X2/CLKIN до низкого уровня сигнала H1, $t_{d(CIH–H1L)}$	0,5	3	нс
3	Время установки сигнала RESET# высокого уровня перед низким уровнем сигнала H1 и после 10 циклов сигнала H1, $t_{su(RESETH–H1L)}$	3,25		нс
4.1	Время задержки от высокого уровня сигнала X2/CLKIN до низкого уровня сигнала H3, $t_{d(CIH–H3L)}$	0,5	3	нс
4.2	Время задержки от высокого уровня сигнала X2/CLKIN до высокого уровня сигнала H3, $t_{d(CIH–H3H)}$	0,5	3	нс
5	Время выключения от высокого уровня сигнала H1 до высокоимпедансного состояния сигналов (L)D31-(L)D0, $t_{dis[H1H–(L)DZ]}$	3,25		нс
6	Время выключения от высокого уровня сигнала H3 до высокоимпедансного состояния сигналов (L)A30-(L)A0, $t_{dis[H3H–(L)AZ]}$	2,25		нс
7	Время задержки от высокого уровня сигнала H3 до высокого уровня сигналов управления (низкий уровень для сигналов (L)PAGE0, (L)PAGE1), $t_{d(H3H–CONTROLH)}$	2,25		нс
8	Время задержки от высокого уровня сигнала H1 до высокого уровня сигнала IACK#, $t_{d(H1H-IACKH)}$	2,25		нс
9	Время выключения от низкого уровня сигнала RESET# до высокоимпедансного состояния асинхронных сигналов, $t_{dis(RESETL–ASYNCZ)}$	5,25		нс
10	Время задержки от высокого уровня сигнала RESET# до высокого уровня асинхронных сигналов, $t_{d(RESETH–COMMH)}$	3,75		нс

* Соответствующей цифрой обозначен параметр на рисунке Б.13.



Примечания

1 (L)Dx включает в себя D31-D0, LD31-LD0 и CnD7-CnD0, где n изменяется от 0 до 5.

2 (L)Ax включает в себя A30-A0 и LA30-LA0.

3 Сигналы управления STRB0#, STRB1#, LSTRB0#, LSTRB1#, (L)LOCK#, (L)STAT3-(L)STAT0, (L)R/W0#, (L)R/W1# находятся в высоком уровне пока сигналы (L)PAGE0 и (L)PAGE1 находятся в низком уровне.

4 Сигналы, которые после сброса переходят в высокоимпедансное состояние: TCLK0, TCLK1, IIOF3#-IIOF0# и сигналы коммуникационных портов CREQx#, CACKy#, CSTRBy#, CRDYx#, где x для 0, 1 и 2 портов, y для 3, 4 и 5 портов.

5 Сигналы коммуникационных портов, которые после сброса переходят в высокий уровень: CREQy#, CACKx#, CSTRBx#, CRDYy#, где x для 0, 1 и 2 портов, y для 3, 4 и 5 портов.

6 Цифрами обозначены номера строк таблицы Б.11, содержащих соответствующие параметры.

_____

* Hi-Z – обозначение высокоимпедансного состояния.

Рисунок Б.13 – Временная диаграмма сигнала RESET#

Таблица Б.12 – Временные параметры обработки прерываний от сигналов IIOF3#-IIOF0# при P = $t_{c(H)}$ (смотри рисунок Б.14)

Номер параметра*	Наименование параметра, буквенное обозначение параметра	Значение параметра			Единица измерения
		минимум	норма	максимум	
1	Время установки сигналов IIOF3#-IIOF0# перед низким уровнем сигнала H1, $t_{su(IIOF–H1L)}$	2,75			нс
2	Время импульса прерывания для гарантированного распознавания одного прерывания, $t_{w(IIOF)}$	P	1,5P	<2P	нс

_____

* Соответствующей цифрой обозначен параметр на рисунке Б.14.

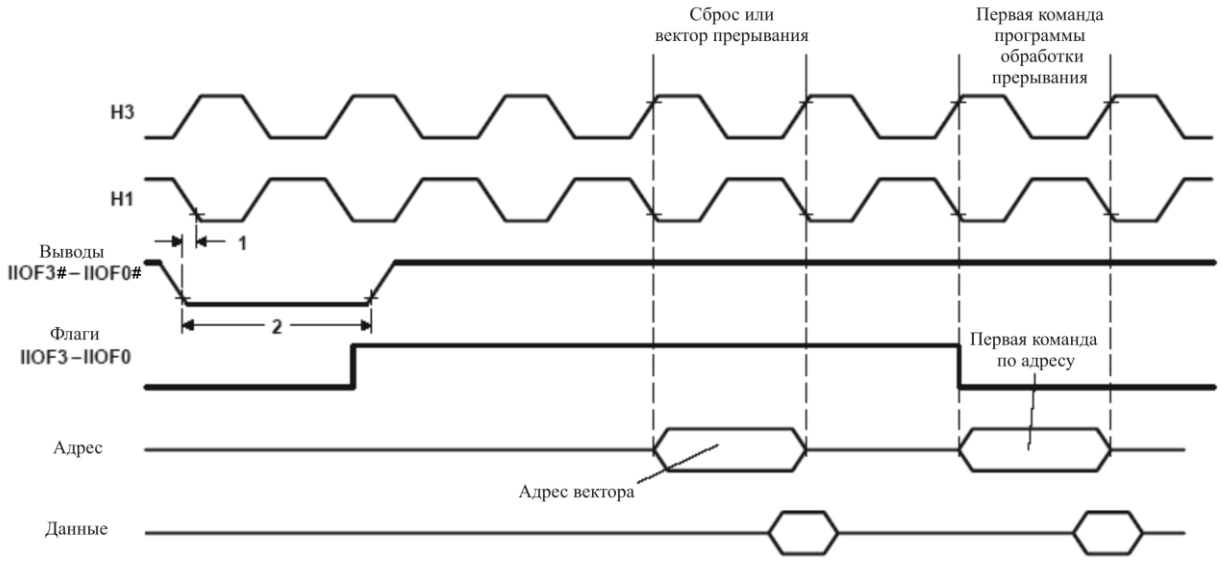


Рисунок Б.14 – Временная диаграмма прерываний от сигналов IIOF3#-IIOF0# при P = $t_{c(H)}$

Таблица Б.13 – Временные параметры сигнала IACK# (смотри рисунок Б.15)

Номер параметра*	Наименование параметра, буквенное обозначение параметра	Значение параметра		Единица измерения
		минимум	максимум	
1	Время задержки от высокого уровня сигнала H1 до низкого уровня сигнала IACK#, $t_{d(H1H–IACKL)}$	2,25		нс
2	Время задержки от низкого уровня сигнала H1 до высокого уровня сигнала IACK# в течение первого цикла инструкции IACK (чтение данных), $t_{d(H1L–IACKH)}$	2,25		нс

_____

* Соответствующей цифрой обозначен параметр на рисунке Б.15.

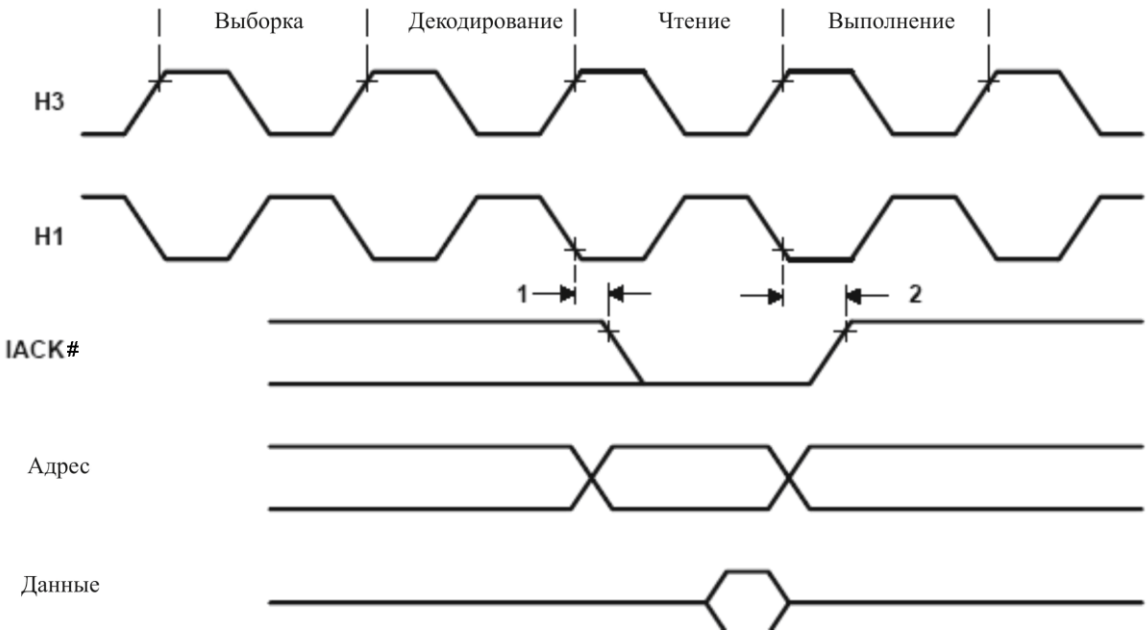


Рисунок Б.15 – Временная диаграмма сигнала IACK#

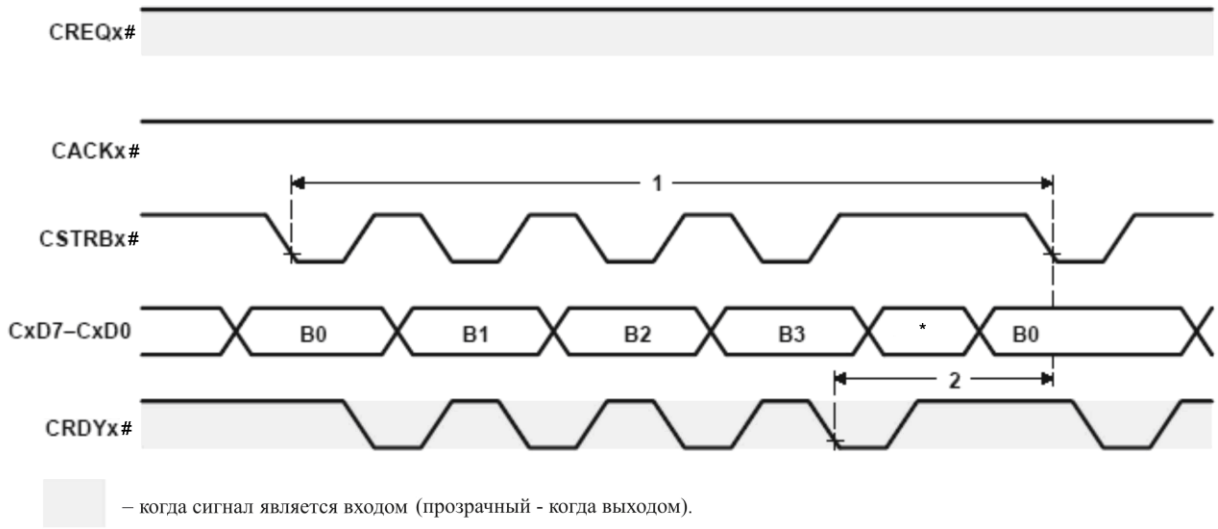
Таблица Б.14 – Временные параметры цикла передачи слов коммуникационным портом при P = $t_{c(H)}$ (смотри рисунок Б.16)

Номер параметра*	Наименование параметра, буквенное обозначение параметра	Значение параметра		Единица измерения
		минимум	максимум	
1	Время цикла передачи слова, $t_{c(WORD)}$ (4 байта = 1 слово)	1,5P+7	2,5P+4,25	нс
2	Время задержки от низкого уровня сигналов CRDYx# до низкого уровня сигналов CSTRBx# между циклами записи, $t_{d(CRDYL–CSL)W}$	1,5P+7	2,5P+7	нс

Примечание – x = 0, …, 5.

_____

* Соответствующей цифрой обозначен параметр на рисунке Б.16.



– когда сигнал является входом (прозрачный - когда выходом).

x = 0, …, 5.

_____

* Неопределенное значение.

Рисунок Б.16 – Временная диаграмма цикла передачи слов коммуникационным портом при P = $t_{c(H)}$

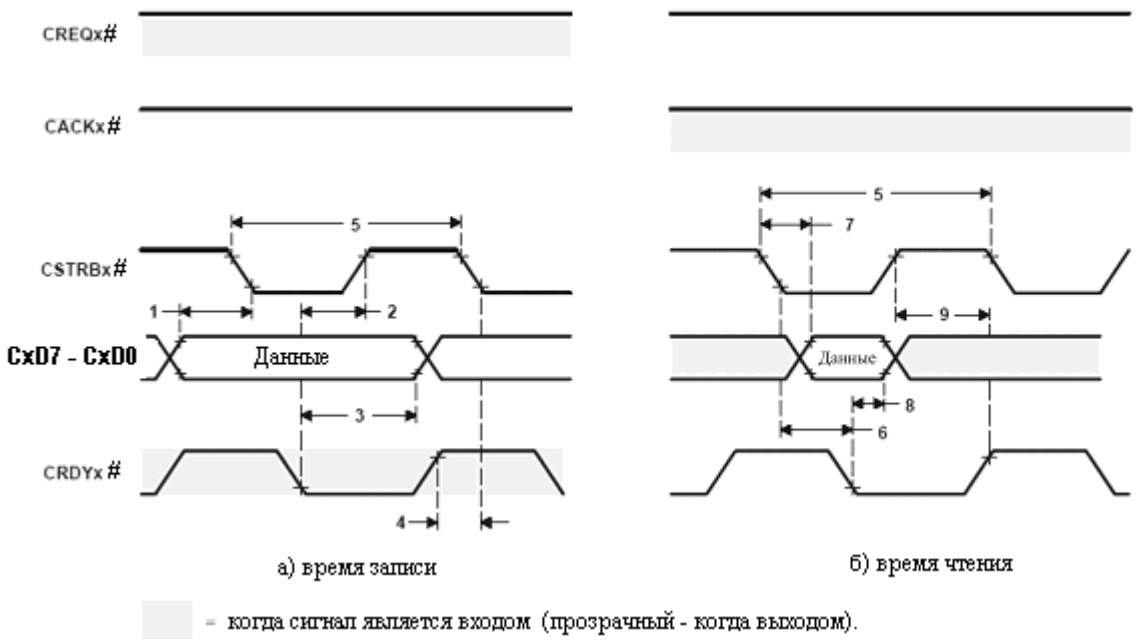
Таблица Б.15 – Временные параметры записи и чтения байта коммуникационного порта (смотри рисунок Б.17)

Номер параметра*	Наименование параметра, буквенное обозначение параметра	Значение параметра		Единица измерения
		минимум	максимум	
1	Время установки значения данных сигналов CxDn перед низким уровнем сигналов CSTRBx# (запись), $t_{su(CD–CSL)W}$	0,50		нс
2	Время задержки от низкого уровня сигналов CRDYx# до высокого уровня сигналов CSTRBx# (запись), $t_{d(CRDYL–CSH)W}$	0,00	3,00	нс
3	Время удержания сигналов CxDn после низкого уровня сигналов CRDYx# (запись), $t_{h(CRDYL–CD)W}$	0,25		нс
4	Время задержки от высокого уровня сигналов CRDYx# до низкого уровня сигналов CSTRBx# для последовательности байтов (запись), $t_{d(CRDYH–CSL)W}$	0,00	3,00	нс
5	Время цикла передачи данных (чтение/запись), $t_{c(BYTE)R/W}$	11,00		нс
6	Время задержки от низкого уровня сигналов CSTRBx# до низкого уровня сигналов CRDYx# (чтение), $t_{d(CSL–CRDYL)R}$	0,00	2,25	нс
7	Время установки значения сигналов CxDn после высокого уровня сигналов CSTRBx# (чтение), $t_{su(CSH–CD)R}$	0,00		нс
8	Время удержания сигналов CxDn после низкого уровня сигнал сигналов CRDYx# (чтение), $t_{h(CRDYL–CD)R}$	0,50		нс
9	Время задержки от высокого уровня сигналов CSTRBx# до высокого уровня сигналов CRDYx# (чтение), $t_{d(CSH–CRDYH)R}$	0,00	2,50	нс

Примечание – x = 0, …, 5; n = 7, …, 0.

_____

* Соответствующей цифрой обозначен параметр на рисунке Б.17.



x = 0, …, 5.

Примечание – Цифрами обозначены номера строк таблицы Б.15, содержащих соответствующие параметры.

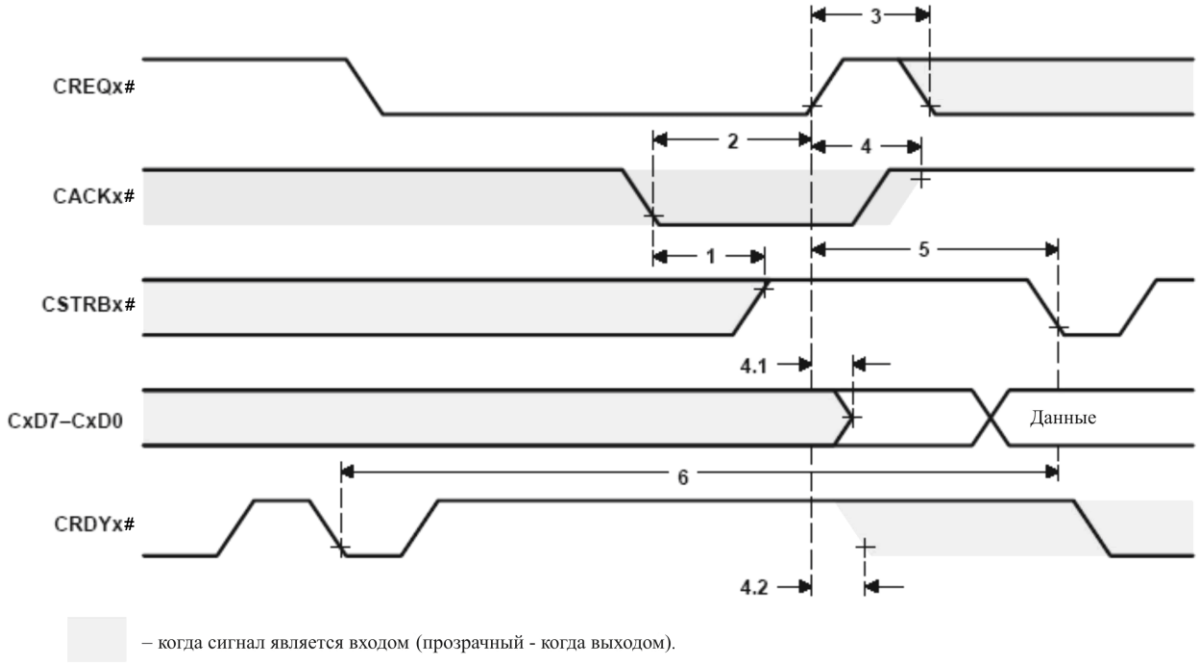
Рисунок Б.17 – Временная диаграмма байта коммуникационного порта (запись и чтение)

Таблица Б.16 – Временные параметры последовательности передачи права от входного к выходному коммуникационному порту при P = $t_{c(H)}$ (см. рисунок Б.18)

Номер параметра*	Наименование параметра, буквенное обозначение параметра	Значение параметра		Единица измерения
		минимум	максимум	
1	Время задержки от низкого уровня сигналов CACKx# до перехода сигналов CSTRBx# из входного состояния в выходное состояние высокого уровня, $t_{d(CAL–CSH)T}$	0,5P–5,0	0,5P+3,25	нс
2	Время задержки от низкого уровня сигналов CACKx# до начала перехода сигналов CREQx# в высокий уровень, $t_{d(CAL–CRQH)T}$	P+5,0	2P+6,5	нс
3	Время задержки от начала перехода сигналов CREQx# в высокий уровень до перехода сигналов CREQx# из выходного состояния во входное, $t_{d(CRQH–CRQ)T}$	0,5P–5,0	0,5P+3,25	нс
4	Время задержки от начала перехода сигналов CREQx# в высокий уровень до перехода сигналов CACKx# из входного состояния в выходное состояние высокого уровня, $t_{d(CRQH–CAH)T}$	0,5P–5,0	0,5P+3,25	нс
4.1	Время задержки от начала перехода сигналов CREQx# в высокий уровень до перехода сигналов CxD7-CxD0 из входного состояния в выходное, $t_{d(CRQH–CD)T}$	0,5P–5,0	0,5P+3,25	нс
4.2	Время задержки от начала перехода сигналов CREQx# в высокий уровень до перехода сигналов CRDYx# из выходного состояния во входное, $t_{d(CRQH–CRDY)T}$	0,5P–5,0	0,5P+3,25	нс
5	Время задержки от начала перехода сигналов CREQx# в высокий уровень до низкого уровня сигналов CSTRBx# для начала внешней передачи слов, $t_{d(CRQH–CSL)T}$	1,5P–8,0	1,5P+2,25	нс
6	Время задержки от низкого уровня сигналов CRDYx# в конце входного слова до низкого уровня сигналов CSTRBx# для выходного слова, $t_{d(CRDYL–CSL)T}$	3,5P+12,0	5,5P+12,0	нс

Примечание – x = 0, …, 5.

* Соответствующей цифрой обозначен параметр на рисунке Б.18.



x = 0, …, 5.

Примечание – Цифрами обозначены номера строк таблицы Б.16, содержащих соответствующие параметры.

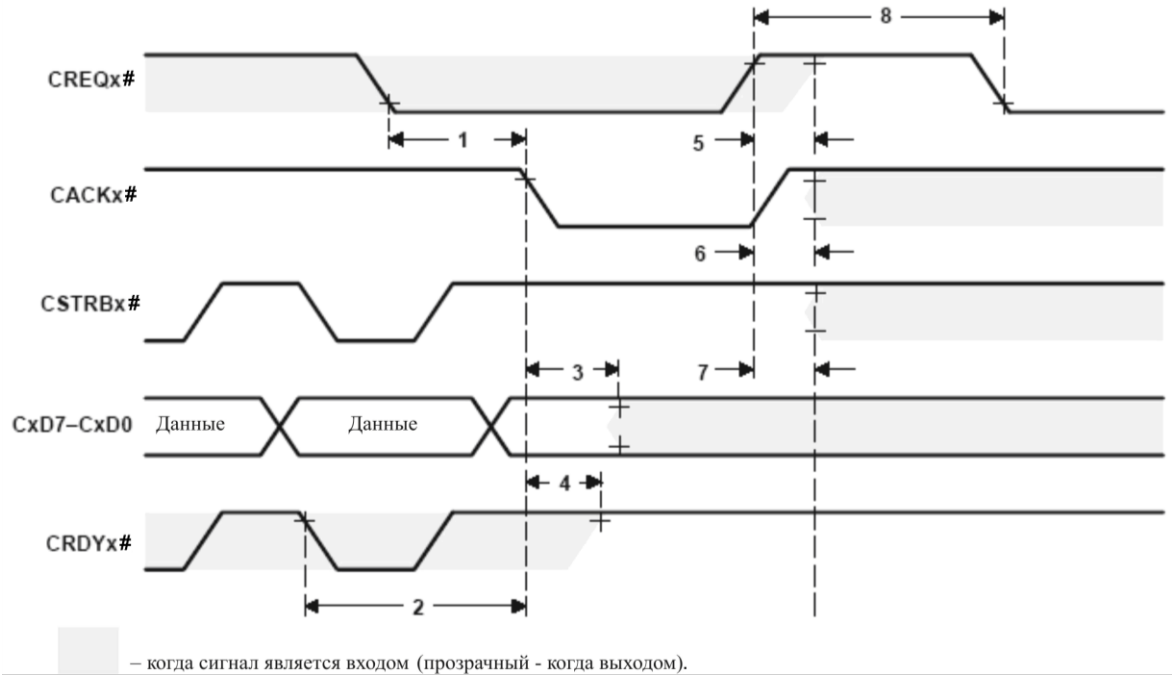
Рисунок Б.18 – Временная диаграмма последовательности передачи права от входного порта к выходному коммуникационному порту при P = $t_{c(H)}$

Таблица Б.17 – Временные параметры последовательности передачи права от выходного порта к входному коммуникационному порту при P= $t_{c(H)}$ (смотри рисунок Б.19)

Номер параметра*	Наименование параметра, буквенное обозначение параметра	Значение параметра		Единица измерения
		минимум	максимум	
1	Время задержки от низкого уровня сигналов CREQx# до начала перехода сигналов CACKx# в низкий уровень для подтверждения запроса «жетона», $t_{d(CRQL–CAL)T}$	P+5	2P+6,5	нс
2	Время задержки от начала перехода сигналов CRDYx# в низкий уровень в конце передачи слова до начала перехода сигналов CACKx# в низкий уровень, $t_{d(CRDYL–CAL)T}$	P+6	2P+6,75	нс
3	Время задержки от начала перехода сигналов CACKx# в низкий уровень до перехода сигналов CxD7-CxD0 из выходного состояния во входное, $t_{d(CAL–CD)T}$	0,5P–8	0,5P+2	нс
4	Время задержки от начала перехода сигналов CACKx# в низкий уровень до перехода сигналов CRDYx# из входного состояния в выходное высокого уровня, $t_{d(CAL–CRDYH)T}$	0,5P–8	0,5P+2	нс
5	Время задержки сигналов CREQx# высокого уровня до перехода сигналов CREQx# из входного состояния в выходное состояние высокого уровня, $t_{d(CRQH–CRQH)T}$	1,0	5,5	нс
6	Время задержки от начала перехода сигналов CREQx# в высокий уровень до перехода сигналов CACKx# из выходного состояния во входное, $t_{d(CRQH–CA)T}$	1,0	5,5	нс
7	Время задержки от начала перехода сигналов CREQx# в высокий уровень до перехода сигналов CSTRBx# из выходного состояния во входное, $t_{d(CRQH–CS)T}$	1,0	5,5	нс
8	Время задержки от высокого уровня сигналов CREQx# до низкого уровня сигналов CREQx# для следующего запроса «жетона», $t_{d(CRQH–CRQL)T}$	P–1	2P+2	нс

Примечание – x = 0, …, 5.

* Соответствующей цифрой обозначен параметр на рисунке Б.19.



x = 0, …, 5.

Рисунок Б.19 – Временная диаграмма последовательности передачи права от выходного порта к входному коммуникационному порту при P = $t_{c(H)}$

Таблица Б.18 – Временные параметры для выводов TCLK0, TCLK1 (смотри рисунок Б.20)

Номер параметра*	Наименование параметра, буквенное обозначение параметра	Значение параметра		Единица измерения
		минимум	максимум	
1	Время установки сигналов TCLK0, TCLK1 перед низким уровнем сигнала H1, $t_{su(TCLK–H1L)}$	2,5		нс
2	Время удержания сигналов TCLK0, TCLK1 после низкого уровня сигнала H1, $t_{h(H1L–TCLK)}$	0		нс
3	Время задержки сигналов TCLK0, TCLK1 после высокого уровня сигнала H1, $t_{d(H1H–TCLK)}$	3,5		нс

_____

* Соответствующей цифрой обозначен параметр на рисунке Б.20.

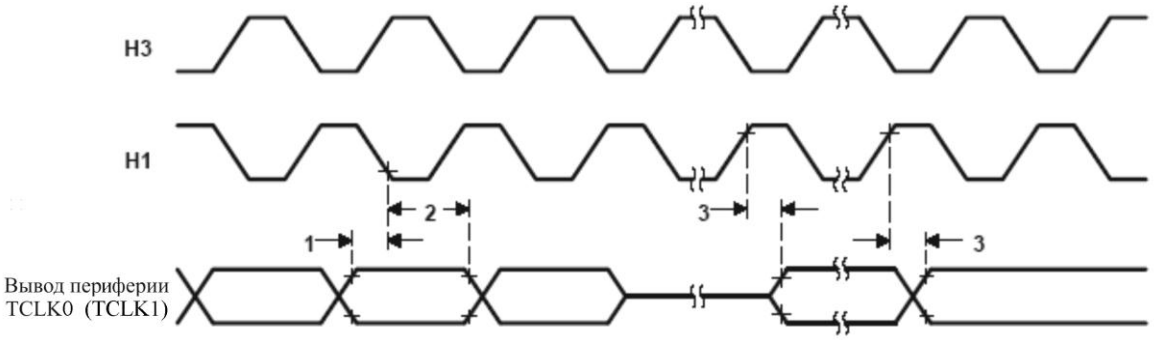


Рисунок Б. 20 – Временная диаграмма для вывода TCLK0 (TCLK1) таймера

Таблица Б.19 – Временные параметры для IEEE 1149.1 тестового порта (смотри рисунок Б.21)

Номер параметра*	Наименование параметра, буквенное обозначение параметра	Значение параметра		Единица измерения
		минимум	максимум	
1	Время установки сигналов TMS/TDI перед высоким уровнем сигнала TCK, $t_{su(TMS/TDI–TCKH)}$	10,0		нс
2	Время удержания сигналов TMS/TDI после высокого уровня сигнала TCK, $t_{h(TCKH–TMS/TDI)}$	5,0		нс
3	Время задержки от сигнала TCK низкого уровня до установившегося значения сигнала TDO, $t_{d(TCKL–TDOV)}$	0,0	15,0	нс

_____

* Соответствующей цифрой обозначен параметр на рисунке Б.21.

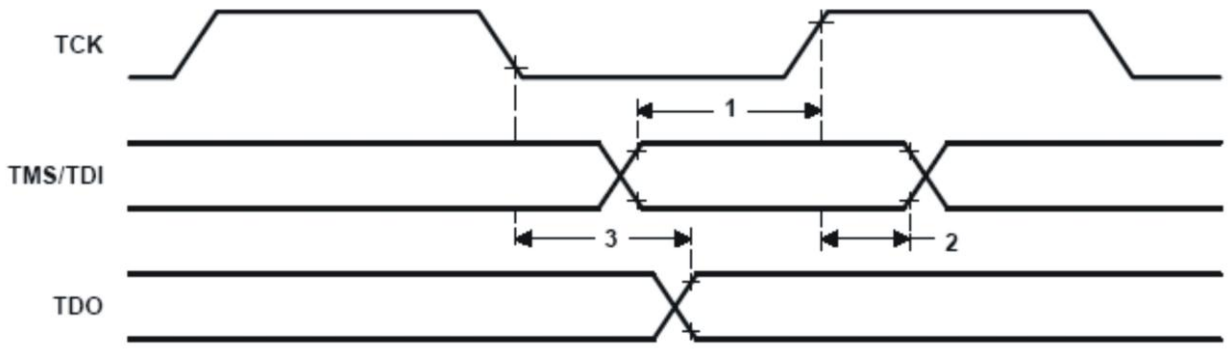


Рисунок Б.21 – Временная диаграмма для IEEE 1149.1 тестового порта

Приложение В
(обязательное)

Электрические параметры микросхемы

Электрические параметры ИС 1867ВМ9Ф при приемке и поставке должны соответствовать нормам, приведенным в таблице В.1.

Таблица В.1

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 низкого уровня буферов ввода-вывода, В, U_{CC1} = 3,0 В, U_{CC2} = 3,0 В, I_{OL} = 2,0 мА		U_{OL}	–	0,4	–60 ± 3 25 ± 10 85 ± 3
2 Выходное напряжение высокого уровня буферов ввода-вывода, В, U_{CC1} = 3,0 В, U_{CC2} = 3,0 В, I_{OH} = –0,3 мА		U_{OH}	U_{CC1}–0,3	–	
3 Входной ток низкого уровня, мкА, U_{CC1} = 3,6 В, U_{CC2} = 3,6 В, U_{IL} = 0 В	Вход «pull-down»	I_{IL}	–15	200	
	Входы «pull-up»		–200	15	
	Вход X2/CLKIN		–55	55	
	Все остальные входы		–15	15	
4 Входной ток высокого уровня, мкА, U_{CC1} = 3,6 В, U_{CC2} = 3,6 В, $U_{IH} = U_{CC1}$	Вход «pull-down»	I_{IH}	–15	200	
	Входы «pull-up»		–200	15	
	Вход X2/CLKIN		–55	55	
	Все остальные входы		–15	15	
5 Выходной ток низкого уровня буфера с третьим состоянием в состоянии «Выключено», мкА, U_{CC1} = 3,6 В, U_{CC2} = 3,6 В, U_{OZL} = 0 В		I_{OZL}	–15	15	
6 Выходной ток высокого уровня буфера с третьим состоянием в состоянии «Выключено», мкА, U_{CC1} = 3,6 В, U_{CC2} = 3,6 В, $U_{OZH} = U_{CC1}$		I_{OZH}	–15	15	
7 Динамический ток потребления буферов ввода-вывода микросхемы, мА, U_{CC1} = 3,6 В, U_{CC2} = 3,6 В, f_{CI} = 40 МГц		I_{OCC1}	–	60	
8 Динамический ток потребления ядра микросхемы, мА, U_{CC1} = 3,6 В, U_{CC2} = 3,6 В, f_{CI} = 40 МГц		I_{OCC2}	–	300	
9 Функциональный контроль, U_{CC1} = (3,0; 3,6) В, U_{CC2} = (3,0; 3,6) В, f_{CI} = (5; 40) МГц		ФК	–	–	

Примечания

1 Параметры I_{IL}, I_{IH}, I_{OZL}, I_{OZH} при температуре минус 60 ºС не измеряются, а гарантируются нормами при температуре (25 ± 10) ºС.

2 Вход «pull-down» – TRST#.

3 Входы «pull-up» – TCK, TDI, TMS.

Значения предельно допустимых и предельных режимов эксплуатации в диапазоне рабочих температур среды должны соответствовать нормам, приведенным в таблице В.2.

Таблица В.2

Наименование параметра режима, единица измерения	Буквенное обозначение параметра	Предельно допустимый режим		Предельный режим	
		не менее	не более	не менее	не более
1	2	3	4	5	6
1 Напряжение питания буферов ввода-вывода, В [1), 2)]	U_{CC1}	3,0	3,6	–0,3	4,5
2 Напряжение питания ядра микросхемы, В [1), 2)]	U_{CC2}	3,0	3,6	–0,3	4,5
3 Входное напряжение низкого уровня, В [2)]	U_{IL}	0	0,8	–0,3	–
4 Входное напряжение высокого уровня, В [2)]	U_{IH}	2,0	U_{CC1}	–	U_{CC1} + 0,3
5 Напряжение на выходе с третьим состоянием в состоянии «Выключено», В [2)]	U_{OZ}	0	U_{CC1}	–0,3	U_{CC1} + 0,3
6 Выходной ток низкого уровня, мА [2)]	I_{OL}	–	2,0	–	2,5
7 Выходной ток высокого уровня, мА [2)]	I_{OH}	–0,3	–	–0,5	–
8 Частота следования импульсов тактовых сигналов, МГц	f_{CI}	5	40	–	–
9 Емкость нагрузки, пФ	C_L	–	20	–	40

[1)] Между напряжениями питания буферов ввода-вывода U_{CC1} и ядра U_{CC2} микросхемы должно сохраняться соотношение $|U_{CC1} - U_{CC2}| \le 0{,}3$ В.

[2)] Время работы в одном из предельных режимов должно быть не более 5 с.

Электрические параметры микросхем, изменяющиеся в процессе и после воздействия специальных факторов, в том числе в диапазоне рабочих температур, должны соответствовать нормам, приведенным в таблице В.3. Остальные параметры должны соответствовать нормам при приемке и поставке, приведенным в таблице В.1.

Таблица В.3

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, ºС
		не менее	не более	
1 Выходное напряжение высокого уровня буферов ввода-вывода, В, U_{CC1} = 3,0 В, U_{CC2} = 3,0 В, I_{OH} = –0,3 мА	U_{OH}	2,0	–	–60 ± 3 25 ± 10 85 ± 3
2 Динамический ток потребления ядра микросхемы, мА, U_{CC1} = 3,6 В, U_{CC2} = 3,6 В, f_{CI} = 40 МГц	I_{OCC2}	–	450	

Микросхемы должны быть стойкими к воздействию специальных факторов 7.И, 7.С, 7.К с характеристиками по группам исполнения ГОСТ РВ 20.39.414.2-98, указанным в таблице В.4.

Таблица В.4

Условное обозначение микросхемы	Группа исполнения для специальных факторов с характеристиками								
	$7.И_1$	$7.И_6$	$7.И_7$	$7.И_{12}$ $7.И_{13}$	$7.С_1$	$7.С_4$	$7.К_1$	$7.К_4$	$7.К_{11}$
1867ВМ9Ф	$5У_с$	$5У_с$[1) 2)]	$0,5\times5У_с$	2×2Р	$5У_с$	$5У_с$	$0,5\times2К$[3)] $2К$[4)]	$0,5\times1К$[3)] $1К$[4)]	60 МэВ·см^2/мг[1) 2)]

_____

1) По катастрофическим отказам.
2) Уровень возникновения тиристорного эффекта.
3) При совместном воздействии факторов с характеристиками $7.К_1$ и $7.К_4$.
4) При независимом воздействии факторов с характеристиками $7.К_1$ и $7.К_4$.

Требования к специальным факторам с характеристиками $7.И_2$, $7.И_3$, $7.И_9$, $7.И_{14}$ – $7.И_{28}$, $7.С_2$, $7.С_5$, $7.К_2$, $7.К_5$ $7.К_7$, $7.К_8$ не предъявляют.

Допускается в процессе и непосредственно после воздействия специальных факторов с характеристиками $7.И_1$, $7.И_6$ временная потеря работоспособности микросхем. По истечении 2 мс от начала воздействия работоспособность восстанавливается.

Уровень бессбойной работы по $7.И_8$ (по характеристике $7.И_6$) должен быть не хуже $1У_с$.

Критериями работоспособности являются U_{OL}, U_{OH} I_{OCC}, ФК.

Приложение Г
(обязательное)
Условное графическое обозначение

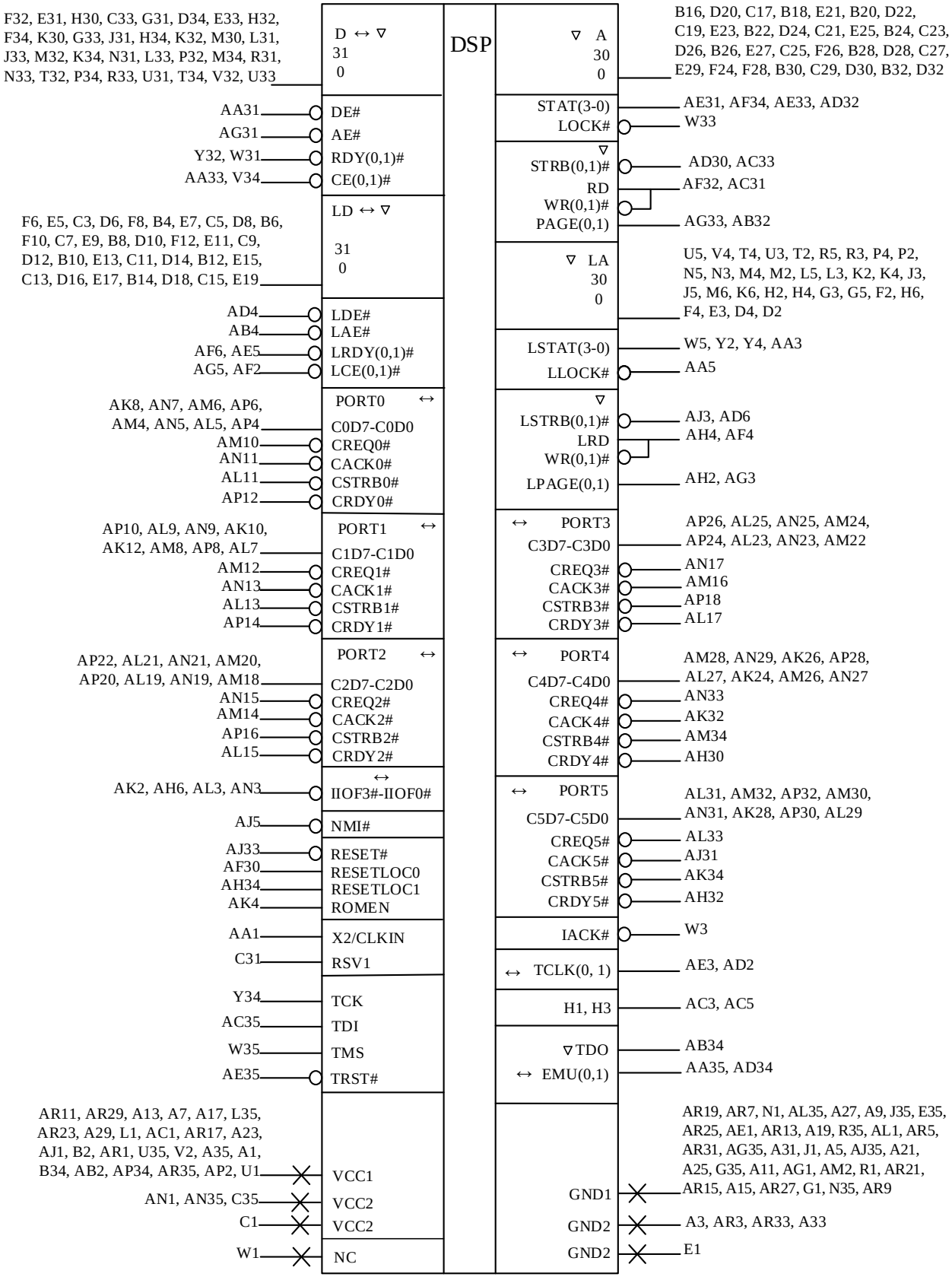


Рисунок Г.1 – Условное графическое обозначение ИС 1867ВМ9Ф

Приложение Д
(обязательное)
Функциональное назначение выводов

В таблице Д.1 приведены обозначения сигналов ИС 1867ВМ9Ф, соответствующие номера выводов корпуса, типы выводов и их функциональное назначение.

Таблица Д.1 – Назначение выводов ИС 1867ВМ9Ф

Обозначение вывода	Вывод корпуса	Направ-ленность вывода*	Описание
1	2	3	4
Интерфейс глобальной шины			
D0	U33	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 0
D1	V32	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 1
D2	T34	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 2
D3	U31	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 3
D4	R33	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 4
D5	P34	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 5
D6	T32	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 6
D7	N33	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 7
D8	R31	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 8
D9	M34	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 9
D10	P32	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 10
D11	L33	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 11
D12	N31	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 12
D13	K34	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 13
D14	M32	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 14
D15	J33	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 15
D16	L31	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 16
D17	M30	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 17
D18	K32	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 18
D19	H34	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 19
D20	J31	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 20
D21	G33	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 21
D22	K30	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 22

Продолжение таблицы Д.1

1	2	3	4
D23	F34	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 23
D24	H32	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 24
D25	E33	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 25
D26	D34	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 26
D27	G31	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 27
D28	C33	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 28
D29	H30	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 29
D30	E31	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 30
D31	F32	IOZ	Вывод 32-разрядного порта данных глобальной шины, вход/выход с 3-им состоянием, бит 31
A0	D32	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 0
A1	B32	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 1
A2	D30	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 2
A3	C29	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 3
A4	B30	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 4
A5	F28	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 5
A6	F24	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 6
A7	E29	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 7
A8	C27	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 8
A9	D28	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 9
A10	B28	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 10
A11	F26	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 11
A12	C25	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 12
A13	E27	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 13
A14	B26	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 14
A15	D26	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 15
A16	C23	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 16
A17	B24	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 17
A18	E25	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 18
A19	C21	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 19
A20	D24	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 20
A21	B22	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 21
A22	E23	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 22
A23	C19	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 23
A24	D22	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 24
A25	B20	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 25
A26	E21	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 26
A27	B18	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 27
A28	C17	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 28
A29	D20	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 29
A30	B16	OZ	Вывод 31-разрядного адресного порта глобальной шины, выход с 3-им состоянием, бит 30
DE#	AA31	I	Вывод «разрешение глобальной шины данных», вход
AE#	AG31	I	Вывод «разрешение глобальной шины адреса», вход
STAT0	AD32	O	Вывод сигнала состояния 0 для внешнего интерфейса глобальной шины, выход
STAT1	AE33	O	Вывод сигнала состояния 1 для внешнего интерфейса глобальной шины, выход
STAT2	AF34	O	Вывод сигнала состояния 2 для внешнего интерфейса глобальной шины, выход
STAT3	AE31	O	Вывод сигнала состояния 3 для внешнего интерфейса глобальной шины, выход
LOCK#	W33	O	Вывод сигнала блокировки интерфейса глобальной шины, выход
STRB0#	AD30	OZ	Вывод строба 0 для внешнего интерфейса глобальной шины, выход с 3-им состоянием
STRB1#	AC33	OZ	Вывод строба 1 для внешнего интерфейса глобальной шины, выход с 3-им состоянием
RD/WR0#	AF32	OZ	Вывод «чтение/запись глобальной шины при доступе по сигналу STRB0#», выход с 3-им состоянием, сигнал 0
RD/WR1#	AC31	OZ	Вывод «чтение/запись глобальной шины при доступе по сигналу STRB1#», выход с 3-им состоянием, сигнал 1
PAGE0	AG33	OZ	Вывод сигнала «выбор страницы глобальной шины при доступе по сигналу STRB0#», выход с 3-им состоянием, страница 0

Продолжение таблицы Д.1

1	2	3	4
PAGE1	AB32	OZ	Вывод сигнала «выбор страницы глобальной шины при доступе по сигналу STRB1#», выход с 3-им состоянием, страница 1
RDY0#	Y32	I	Вывод «готовность глобальной шины при доступе по сигналу STRB0#», вход
RDY1#	W31	I	Вывод «готовность глобальной шины при доступе по сигналу STRB1#», вход
CE0#	AA33	I	Вывод «разрешение выбора сигналов STRB0#, PAGE0, RD/WR0# глобальной шины», вход
CE1#	V34	I	Вывод «разрешение выбора сигналов STRB1#, PAGE1, RD/WR1# глобальной шины», вход
Интерфейс локальной шины			
LD0	E19	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 0
LD1	C15	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 1
LD2	D18	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 2
LD3	B14	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 3
LD4	E17	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 4
LD5	D16	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 5
LD6	C13	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 6
LD7	E15	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 7
LD8	B12	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 8
LD9	D14	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 9
LD10	C11	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 10
LD11	E13	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 11
LD12	B10	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 12
LD13	D12	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 13
LD14	C9	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 14
LD15	E11	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 15
LD16	F12	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 16
LD17	D10	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 17
LD18	B8	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 18
LD19	E9	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 19
LD20	C7	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 20
LD21	F10	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 21
LD22	B6	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 22
LD23	D8	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 23
LD24	C5	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 24
LD25	E7	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 25
LD26	B4	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 26
LD27	F8	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 27
LD28	D6	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 28
LD29	C3	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 29
LD30	E5	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 30
LD31	F6	IOZ	Вывод 32-разрядного порта данных локальной шины, вход/выход с 3-им состоянием, бит 31
LDE#	AD4	I	Вывод «разрешение локальной шины данных», вход
LAE#	AB4	I	Вывод «разрешение локальной шины адресов», вход
LSTAT0	AA3	O	Вывод состояния сигнала 0 для внешнего интерфейса локальной шины, выход
LSTAT1	Y4	O	Вывод состояния сигнала 1 для внешнего интерфейса локальной шины, выход
LSTAT2	Y2	O	Вывод состояния сигнала 2 для внешнего интерфейса локальной шины, выход
LSTAT3	W5	O	Вывод состояния сигнала 3 для внешнего интерфейса локальной шины, выход
LLOCK#	AA5	O	Вывод сигнала блокировки локальной шины, выход
LA0	D2	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 0
LA1	D4	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 1
LA2	E3	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 2
LA3	F4	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 3
LA4	H6	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 4
LA5	F2	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 5
LA6	G5	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 6
LA7	G3	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 7
LA8	H4	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 8
LA9	H2	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 9
LA10	K6	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 10

Продолжение таблицы Д.1

1	2	3	4
LA11	M6	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 11
LA12	J5	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 12
LA13	J3	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 13
LA14	K4	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 14
LA15	K2	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 15
LA16	L3	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 16
LA17	L5	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 17
LA18	M2	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 18
LA19	M4	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 19
LA20	N3	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 20
LA21	N5	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 21
LA22	P2	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 22
LA23	P4	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 23
LA24	R3	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 24
LA25	R5	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 25
LA26	T2	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 26
LA27	U3	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 27
LA28	T4	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 28
LA29	V4	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 29
LA30	U5	OZ	Вывод 31-разрядного адресного порта локальной шины, выход с 3-им состоянием, бит 30
LSTRB0#	AJ3	OZ	Вывод сигнала «локальный строб 0» для внешнего интерфейса локальной шины, выход с 3-им состоянием
LSTRB1#	AD6	OZ	Вывод сигнала «локальный строб 1» для внешнего интерфейса локальной шины, выход с 3-им состоянием
LRD/WR0#	AH4	OZ	Вывод «чтение/запись локальной шины при доступе по сигналу LSTRB0#», выход с 3-им состоянием, сигнал 0
LRD/WR1#	AF4	OZ	Вывод «чтение/запись локальной шины при доступе по сигналу LSTRB1#», выход с 3-им состоянием, сигнал 1
LPAGE0	AH2	OZ	Вывод сигнала «выбор страницы локальной шины при доступе по сигналу LSTRB0#, выход с 3- им состоянием, страница 0
LPAGE1	AG3	OZ	Вывод сигнала «выбор страницы локальной шины при доступе по сигналу LSTRB1#, выход с 3- им состоянием, страница 1
LRDY0#	AF6	I	Вывод «готовность локальной шины при доступе по сигналу LSTRB0#», вход
LRDY1#	AE5	I	Вывод «готовность локальной шины при доступе по сигналу LSTRB1#», вход
LCE0#	AG5	I	Вывод «разрешение выбора сигналов LSTRB0#, LPAGE0, LRD/WR0# локальной шины», вход
LCE1#	AF2	I	Вывод «разрешение сигналов LSTRB1#, LPAGE1, LRD/WR1# локальной шины», вход
Интерфейс коммуникационного порта 0			
C0D0	AP4	IO	Вывод «шина данных коммуникационного порта 0», вход/выход, бит 0
C0D1	AL5	IO	Вывод «шина данных коммуникационного порта 0», вход/выход, бит 1
C0D2	AN5	IO	Вывод «шина данных коммуникационного порта 0», вход/выход, бит 2
C0D3	AM4	IO	Вывод «шина данных коммуникационного порта 0», вход/выход, бит 3
C0D4	AP6	IO	Вывод «шина данных коммуникационного порта 0», вход/выход, бит 4
C0D5	AM6	IO	Вывод «шина данных коммуникационного порта 0», вход/выход, бит 5
C0D6	AN7	IO	Вывод «шина данных коммуникационного порта 0», вход/выход, бит 6
C0D7	AK8	IO	Вывод «шина данных коммуникационного порта 0», вход/выход, бит 7
CREQ0#	AM10	IO	Вывод сигнала «запрос/выбор коммуникационного порта 0», вход/выход
CACK0#	AN11	IO	Вывод сигнала «подтверждение запроса/выбора коммуникационного порта 0», вход/выход
CSTRB0#	AL11	IO	Вывод сигнала «строб данных коммуникационного порта 0», вход/выход
CRDY0#	AP12	IO	Вывод сигнала «готовность данных коммуникационного порта 0», вход/выход
Интерфейс коммуникационного порта 1			
C1D0	AL7	IO	Вывод «шина данных коммуникационного порта 1», вход/выход, бит 0
C1D1	AP8	IO	Вывод «шина данных коммуникационного порта 1», вход/выход, бит 1
C1D2	AM8	IO	Вывод «шина данных коммуникационного порта 1», вход/выход, бит 2
C1D3	AK12	IO	Вывод «шина данных коммуникационного порта 1», вход/выход, бит 3
C1D4	AK10	IO	Вывод «шина данных коммуникационного порта 1», вход/выход, бит 4
C1D5	AN9	IO	Вывод «шина данных коммуникационного порта 1», вход/выход, бит 5
C1D6	AL9	IO	Вывод «шина данных коммуникационного порта 1», вход/выход, бит 6
C1D7	AP10	IO	Вывод «шина данных коммуникационного порта 1», вход/выход, бит 7

Продолжение таблицы Д.1

1	2	3	4
CREQ1#	AM12	IO	Вывод сигнала «запрос/выбор коммуникационного порта 1», вход/выход
CACK1#	AN13	IO	Вывод сигнала «подтверждение запроса/выбора коммуникационного порта 1», вход/выход
CSTRB1#	AL13	IO	Вывод сигнала «строб данных коммуникационного порта 1», вход/выход
CRDY1#	AP14	IO	Вывод сигнала «готовность данных коммуникационного порта 1», вход/выход
Интерфейс коммуникационного порта 2			
C2D0	AM18	IO	Вывод «шина данных коммуникационного порта 2», вход/выход, бит 0
C2D1	AN19	IO	Вывод «шина данных коммуникационного порта 2», вход/выход, бит 1
C2D2	AL19	IO	Вывод «шина данных коммуникационного порта 2», вход/выход, бит 2
C2D3	AP20	IO	Вывод «шина данных коммуникационного порта 2», вход/выход, бит 3
C2D4	AM20	IO	Вывод «шина данных коммуникационного порта 2», вход/выход, бит 4
C2D5	AN21	IO	Вывод «шина данных коммуникационного порта 2», вход/выход, бит 5
C2D6	AL21	IO	Вывод «шина данных коммуникационного порта 2», вход/выход, бит 6
C2D7	AP22	IO	Вывод «шина данных коммуникационного порта 2», вход/выход, бит 7
CREQ2#	AN15	IO	Вывод сигнала «запрос/выбор коммуникационного порта 2», вход/выход
CACK2#	AM14	IO	Вывод сигнала «подтверждение запроса/выбора коммуникационного порта 2», вход/выход
CSTRB2#	AP16	IO	Вывод сигнала «строб данных коммуникационного порта 2», вход/выход
CRDY2#	AL15	IO	Вывод сигнала «готовность данных коммуникационного порта 2», вход/выход
Интерфейс коммуникационного порта 3			
C3D0	AM22	IO	Вывод «шина данных коммуникационного порта 3», вход/выход, бит 0
C3D1	AN23	IO	Вывод «шина данных коммуникационного порта 3», вход/выход, бит 1
C3D2	AL23	IO	Вывод «шина данных коммуникационного порта 3»,вход/выход, бит 2
C3D3	AP24	IO	Вывод «шина данных коммуникационного порта 3», вход/выход, бит 3
C3D4	AM24	IO	Вывод Вывод «шина данных коммуникационного порта 3», вход/выход, бит 4
C3D5	AN25	IO	Вывод «шина данных коммуникационного порта 3»,вход/выход, бит 5
C3D6	AL25	IO	Вывод «шина данных коммуникационного порта 3», вход/выход, бит 6
C3D7	AP26	IO	Вывод «шина данных коммуникационного порта 3», вход/выход, бит 7
CREQ3#	AN17	IO	Вывод сигнала «запрос/выбор коммуникационного порта 3», вход/выход
CACK3#	AM16	IO	Вывод сигнала «подтверждение запроса/выбора коммуникационного порта 3», вход/выход
CSTRB3#	AP18	IO	Вывод сигнала «строб данных коммуникационного порта 3», вход/выход
CRDY3#	AL17	IO	Вывод сигнала «готовность данных коммуникационного порта 3», вход/выход
Интерфейс коммуникационного порта 4			
C4D0	AN27	IO	Вывод «шина данных коммуникационного порта 4», вход/выход, бит 0
C4D1	AM26	IO	Вывод «шина данных коммуникационного порта 4», вход/выход, бит 1
C4D2	AK24	IO	Вывод «шина данных коммуникационного порта 4», вход/выход, бит 2
C4D3	AL27	IO	Вывод «шина данных коммуникационного порта 4», вход/выход, бит 3
C4D4	AP28	IO	Вывод «шина данных коммуникационного порта 4», вход/выход, бит 4
C4D5	AK26	IO	Вывод «шина данных коммуникационного порта 4», вход/выход, бит 5
C4D6	AN29	IO	Вывод «шина данных коммуникационного порта 4», вход/выход, бит 6
C4D7	AM28	IO	Вывод «шина данных коммуникационного порта 4», вход/выход, бит 7
CREQ4#	AN33	IO	Вывод сигнала «запрос/выбор коммуникационного порта 4», вход/выход
CACK4#	AK32	IO	Вывод сигнала «подтверждение запроса/выбора коммуникационного порта 4»», вход/выход
CSTRB4#	AM34	IO	Вывод сигнала «строб данных коммуникационного порта 4», вход/выход
CRDY4#	AH30	IO	Вывод сигнала «готовность данных коммуникационного порта 4», вход/выход
Интерфейс коммуникационного порта 5			
C5D0	AL29	IO	Вывод «шина данных коммуникационного порта 5», вход/выход, бит 0
C5D1	AP30	IO	Вывод «шина данных коммуникационного порта 5», вход/выход, бит 1
C5D2	AK28	IO	Вывод «шина данных коммуникационного порта 5», вход/выход, бит 2
C5D3	AN31	IO	Вывод «шина данных коммуникационного порта 5», вход/выход, бит 3
C5D4	AM30	IO	Вывод «шина данных коммуникационного порта 5», вход/выход, бит 4
C5D5	AP32	IO	Вывод «шина данных коммуникационного порта 5», вход/выход, бит 5
C5D6	AM32	IO	Вывод «шина данных коммуникационного порта 5», вход/выход, бит 6
C5D7	AL31	IO	Вывод «шина данных коммуникационного порта 5», вход/выход, бит 7
CREQ5#	AL33	IO	Вывод сигнала «запрос/выбор коммуникационного порта 5», вход/выход
CACK5#	AJ31	IO	Вывод сигнала «подтверждение запроса/выбора коммуникационного порта 5», вход/выход
CSTRB5#	AK34	IO	Вывод сигнала «строб данных коммуникационного порта 5», вход/выход
CRDY5#	AH32	IO	Вывод сигнала «готовность данных коммуникационного порта 5», вход/выход

Продолжение таблицы Д.1

1	2	3	4
Прерывания, флаги ввода/вывода, сброс, таймеры, синхронизация			
IIOF0#	AN3	IO	Вывод сигнала 0 прерываний и флагов, вход/выход
IIOF1#	AL3	IO	Вывод сигнала 1 прерываний и флагов, вход/выход
IIOF2#	AH6	IO	Вывод сигнала 2 прерываний и флагов, вход/выход
IIOF3#	AK2	IO	Вывод сигнала 3 прерываний и флагов, вход/выход
NMI#	AJ5	I	Вывод «немаскируемое прерывание», вход
IACK#	W3	O	Вывод «подтверждение прерывания», выход
RESET#	AJ33	I	Вывод «общий сброс», вход
RESETLOC0	AF30	I	Вывод сигнала 0 «определение ячейки вектора сброса», вход
RESETLOC1	AH34	I	Вывод сигнала 1 «определение ячейки вектора сброса», вход
ROMEN	AK4	I	Вывод «выбор встроенного ПЗУ» (0 – запрет, 1 – разрешение), вход
TCLK0	AE3	IO	Вывод таймера 0, вход/выход
TCLK1	AD2	IO	Вывод таймера 1, вход/выход
NC	W1	-	Вывод не используется
X2/ CLKIN	AA1	- I	Вывод подключения генератора внешнего синхросигнала/ вывод тактового сигнала, вход
RSV1	C31	I	Вывод «питание подложки». Используется для тестовых целей
H1	AC3	O	Вывод «сигнал системной синхронизации 1», выход
H3	AC5	O	Вывод «сигнал системной синхронизации 3» выход
Порт эмулятора (JTAG интерфейс IEEE 1149.1)			
TCK	Y34	I	Вывод сигнала синхронизации порта эмулятора, вход
TDO	AB34	OZ	Вывод выходных данных порта эмулятора, выход с 3-им состоянием
TDI	AC35	I	Вывод входных данных порта эмулятора, вход
TMS	W35	I	Вывод выбора режима порта эмулятора, вход
TRST#	AE35	I	Вывод сброса порта эмулятора, вход
EMU0	AA35	IO	Вывод порта эмулятора, эмуляция 0, вход/выход
EMU1	AD34	IO	Вывод порта эмулятора, эмуляция 1, вход/выход
Выводы шин питания 3,3 В			
VCC1	AR11	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	AR29	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	A13	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	A7	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	A17	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	L35	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	AR23	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	A29	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	L1	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	AC1	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	AR17	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	A23	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	AJ1	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	B2	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	AR1	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	U35	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	V2	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	A35	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	A1	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	B34	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	AB2	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	AP34	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	AR35	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	AP2	-	Вывод питания периферии (буферов ввода-вывода)
VCC1	U1	-	Вывод питания периферии (буферов ввода-вывода)
VCC2	AN1	-	Вывод питания ядра
VCC2	AN35	-	Вывод питания ядра
VCC2	C35	-	Вывод питания ядра
VCC2	C1	-	Вывод питания ядра

Окончание таблицы Д.1

1	2	3	4
Выводы общих шин			
GND1	AR19	-	Общий вывод периферии (буферов ввода-вывода)
GND1	AR7	-	Общий вывод периферии (буферов ввода-вывода)
GND1	N1	-	Общий вывод периферии (буферов ввода-вывода)
GND1	AL35	-	Общий вывод периферии (буферов ввода-вывода)
GND1	A27	-	Общий вывод периферии (буферов ввода-вывода)
GND1	A9	-	Общий вывод периферии (буферов ввода-вывода)
GND1	J35	-	Общий вывод периферии (буферов ввода-вывода)
GND1	E35	-	Общий вывод периферии (буферов ввода-вывода)
GND1	AR25	-	Общий вывод периферии (буферов ввода-вывода)
GND1	AE1	-	Общий вывод периферии (буферов ввода-вывода)
GND1	AR13	-	Общий вывод периферии (буферов ввода-вывода)
GND1	A19	-	Общий вывод периферии (буферов ввода-вывода)
GND1	R35	-	Общий вывод периферии (буферов ввода-вывода)
GND1	AL1	-	Общий вывод периферии (буферов ввода-вывода)
GND1	AR5	-	Общий вывод периферии (буферов ввода-вывода)
GND1	AR31	-	Общий вывод периферии (буферов ввода-вывода)
GND1	AG35	-	Общий вывод периферии (буферов ввода-вывода)
GND1	A31	-	Общий вывод периферии (буферов ввода-вывода)
GND1	J1	-	Общий вывод периферии (буферов ввода-вывода)
GND1	A5	-	Общий вывод периферии (буферов ввода-вывода)
GND1	AJ35	-	Общий вывод периферии (буферов ввода-вывода)
GND1	A21	-	Общий вывод периферии (буферов ввода-вывода)
GND1	A25	-	Общий вывод периферии (буферов ввода-вывода)
GND1	G35	-	Общий вывод периферии (буферов ввода-вывода)
GND1	A11	-	Общий вывод периферии (буферов ввода-вывода)
GND1	AG1	-	Общий вывод периферии (буферов ввода-вывода)
GND1	AM2	-	Общий вывод периферии (буферов ввода-вывода)
GND1	R1	-	Общий вывод периферии (буферов ввода-вывода)
GND1	AR21	-	Общий вывод периферии (буферов ввода-вывода)
GND1	AR15	-	Общий вывод периферии (буферов ввода-вывода)
GND1	A15	-	Общий вывод периферии (буферов ввода-вывода)
GND1	AR27	-	Общий вывод периферии (буферов ввода-вывода)
GND1	G1	-	Общий вывод периферии (буферов ввода-вывода)
GND1	N35	-	Общий вывод периферии (буферов ввода-вывода)
GND1	AR9	-	Общий вывод периферии (буферов ввода-вывода)
GND2	A3	-	Общий вывод ядра
GND2	AR3	-	Общий вывод ядра
GND2	AR33	-	Общий вывод ядра
GND2	A33	-	Общий вывод ядра
GND2	E1	-	Общий вывод ядра

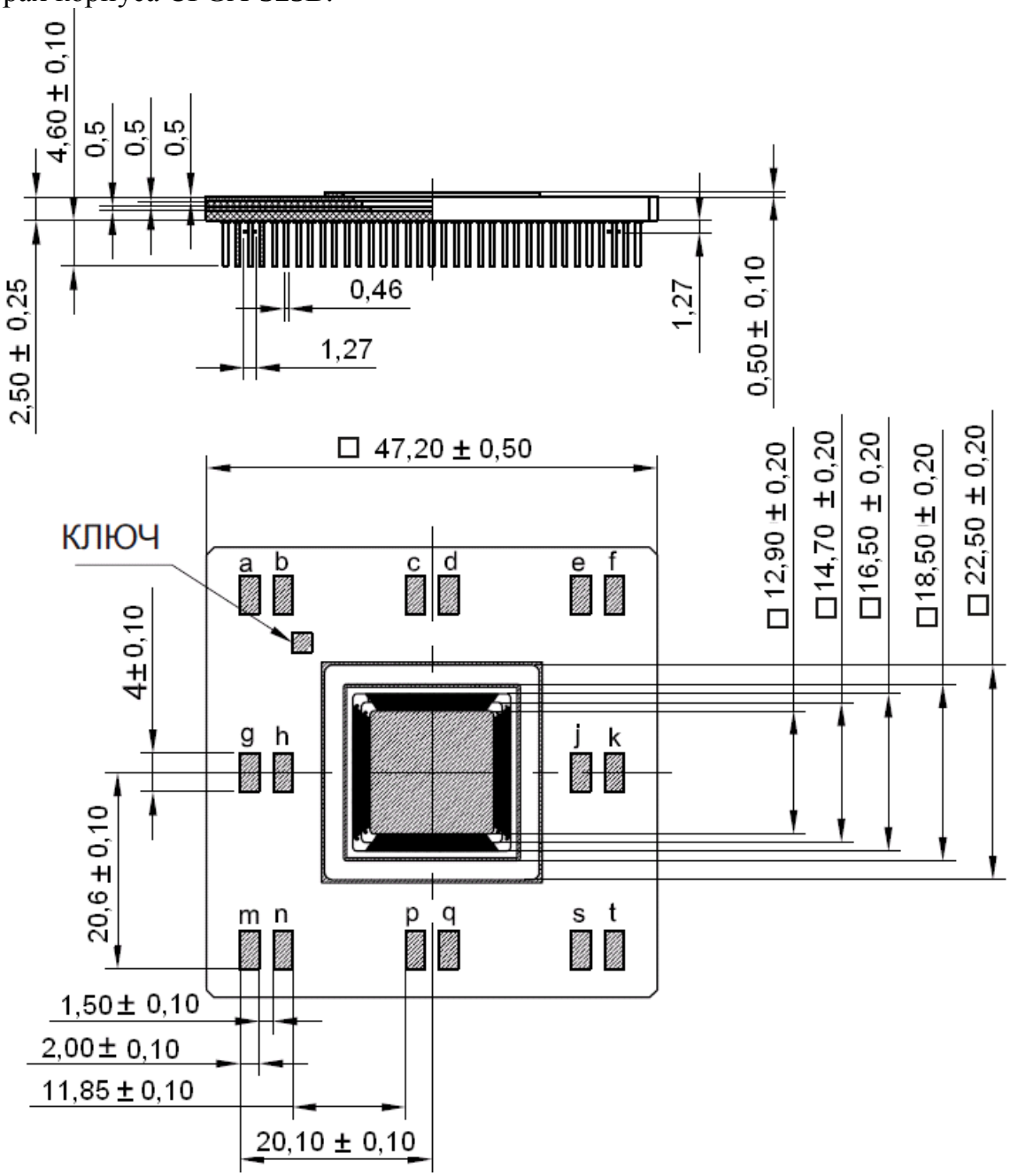
_____

* Принятые условные обозначения:

- I – входной вывод,
- O – выходной вывод,
- IO – входной/выходной вывод,
- IOZ – входной/выходной вывод с третьим состоянием,
- OZ – выходной вывод с третьим состоянием.

Приложение Е
(обязательное)
Корпус CPGA-325B

На рисунке Е.1 приведена информация о присоединительных и габаритных размерах корпуса CPGA-325B.



Примечания

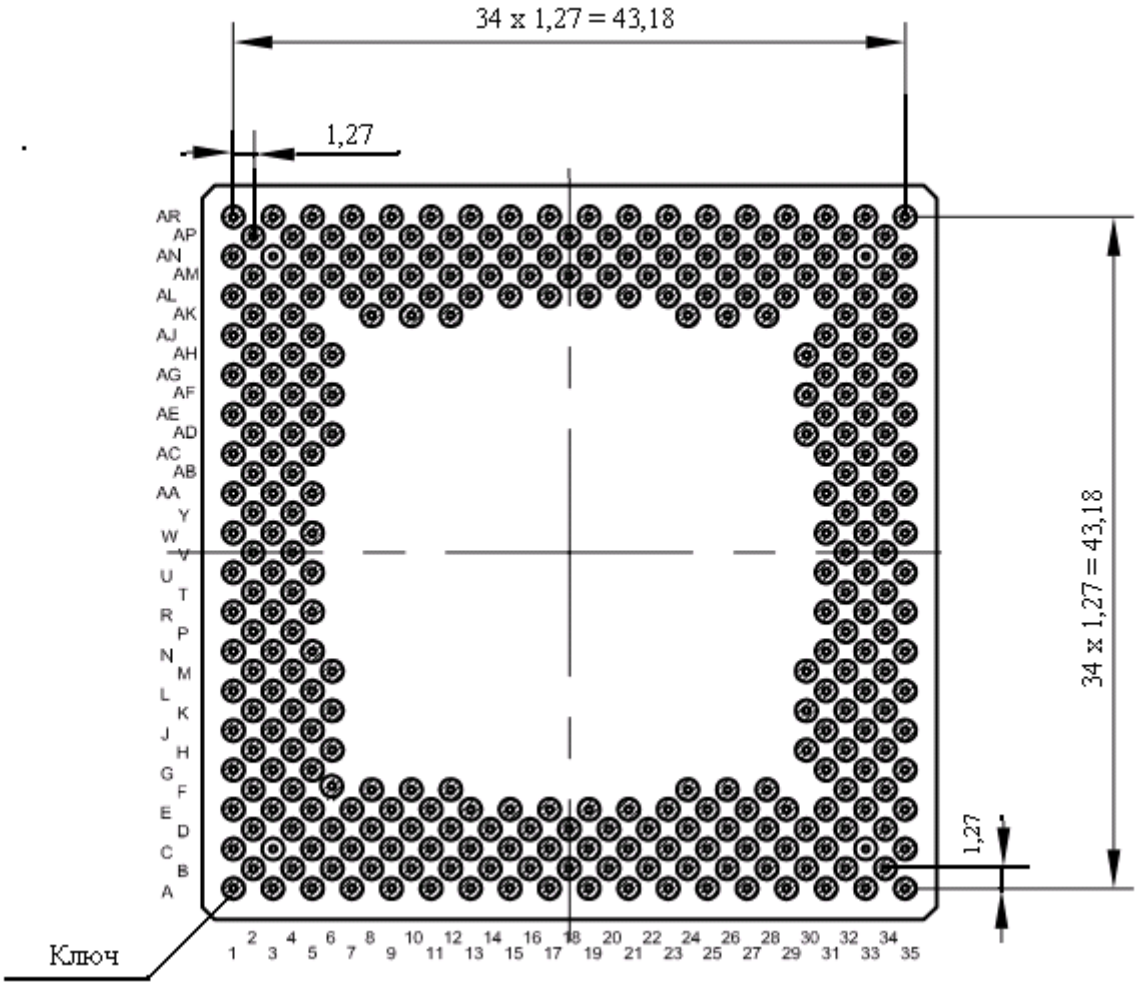
1 Контакты a, e, m, s подключены к выводам GND1 – общий контакт буферов ввода/вывода ИС.

2 Контакты c, g, j, p подключены к выводам GND2 – общий контакт ядра ИС.

3 Контакты b, f, n, t подключены к выводам VCC1 – контакт питания 3,3 В буферов ввода/вывода ИС.

4 Контакты d, h, k, q подключены к выводам VCC2 – контакт питания 3,3 В ядра ИС.

Рисунок Е.1, лист 1 – Корпус CPGA-325B



Примечание – Представлен вид снизу корпуса CPGA-325B.

Рисунок Е.1, лист 2

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ докум.	Подп.	Дата
	изме-ненных	заме-нен-ных	новых	анну-лиро-ванных				
-	-	-	все	-	454	8928		11.11.15
1	1	-	-	-	-	8937		13.11.15
2	-	22, 146, 447,448	-	-	-	8962		14.12.15
3	-	11, 444	-	-	-	9731		30.08.18