МИКРОСХЕМА ИНТЕГРАЛЬНАЯ

1867ВЦ8Ф1

Руководство пользователя

от 09.09.2021

Содержание

1 Обзор ИС 1867ВЦ8Ф1

Микросхема 1867ВЦ8Ф1 – это высокопроизводительная двухпроцессорная система на кристалле, содержащая два ядра 32-разрядного процессора цифровой обработки сигналов (ПЦОС) с плавающей запятой и четыре периферийных устройства – Ethernet 10/100, USB 2.0 Device (режимы приёма/передачи High/Full Speed со скоростями 480/12 Мбит/с), UART с архитектурой UART 16550 со скоростями приёма/передачи от 150 бит/с до 5 Мбит/с и MIL-STD-1553 со скоростью приёма/передачи 1 Мбит/с. Процессорные ядра функционально совместимы с 32-разрядным ПЦОС 1867ВЦ3Ф (функциональный аналог TMS320C40 фирмы Texas Instruments). Периферийные устройства могут подключаться к любому из процессоров в любое время через коммутатор и, соответственно, могут управляться из любого процессора.

Процессорные ядра соединены через коммуникационные порты COMM3 и COMM0, которые обеспечивают приём/передачу данных со скоростью до 480 Мбайт/с. Это даёт возможность реализовать эффективную мультипроцессорную обработку данных.

Каждое процессорное ядро может адресовать до восьми Мбайт асинхронной или синхронной памяти с произвольным доступом через глобальную шину.

Средства разработки и отладки ИС 1867ВЦ8Ф1 включают:

- отладочный порт JTAG с внутрисхемным эмулятором, который обеспечивает мультипроцессорную отладку;
- интегрированную среду разработки и отладки Code Composer, которые поддерживают процессор TMS320C40 (от фирмы Texas Instruments);
- эмулятор XDS510 (от фирмы Texas Instruments или от третьих фирм);
- компиляторы с языков C/C++, Assembler, Linker, DSP/BIOS, API, анализатор программ (от фирмы Texas Instruments или от третьих фирм).

1.1 Краткие характеристики ИС 1867ВЦ8Ф1

В таблице 1.1 приведены краткие функциональные характеристики ИС 1867ВЦ8Ф1.

Таблица 1.1 – Краткие функциональные характеристики ИС 1867ВЦ8Ф1

Наименование параметра, единица измерения	Значение параметра
1	2
Общие характеристики	
Число ПЦОС	2
Разрядность ПЦОС	32
Устройство доступа к периферии	1
Число устройств UART	1
Число устройств Ethernet 10/100 (IEEE 802.3x)	1
Число устройств USB 2.0	1
Число устройств MIL-STD-1553	1
Внутрикристальная схема отладки с интерфейсом JTAG (IEEE 1149.1)	
Блок внутрисхемной мультипроцессорной эмуляции	
Характеристики процессорного ядра	
Разрядность АЛУ, бит: - плавающая запятая (ПЗ) - фиксированная запятая (ФЗ)	 40 32
Умножитель, бит: - плавающая запятая - фиксированная запятая	 40 × 40 32 × 32
Объем ПЗУ, бит	4К × 32
Объем ОЗУ, бит	2К × 32
Объем кэш-ОЗУ, бит	128 × 32
Объем адресуемой памяти, бит	4Г × 32
Таймеры	2
8-разрядные коммутационные порты	6

Продолжение таблицы 1.1

1	2
Сопроцессор ПДП, количество каналов	6
Тактовая частота процессора, МГц, не более	100
Производительность: - фиксированная запятая, MIPS, не менее - плавающая запятая, MFLOPS, не менее	 50 100
Характеристики UART	
Архитектура UART (регистровый уровень)	NS16550A
Скорость передачи/приёма	от 150 бит/с до 5 Мбит/с
Режим работы	8/32-разрядный
Режим передачи/приёма	асинхронный
Контроль чётности/нечётности	есть
Управление контролем по чётности	есть
Управление модемом	есть
Длина передаваемых/принимаемых данных, бит	5/6/7/8
Число стоп, бит	1/1,5/2
Управление прерыванием	есть
Характеристики Ethernet 10/100	
Управление потоком и автоматическая генерация управляющих фреймов в полнодуплексном режиме	есть
Размер буфера передачи/приём, бит	16K × 32
Стандарт	IEEE 802.3x
Обнаружение коллизии	есть
Автопередача после коллизии в полнодуплексном режиме	есть
Протокол обнаружения несущей и разрешения коллизии	CSMA/CD
Автоматическая генерация и проверка 32-битный CRC	есть
Автоматическая генерация преамбулы и её удаление	есть
Интерфейс	MII
Скорость передачи/приёма данных, Мбит/с	10/100
Ширина обмениваемых данных, бит	32
Управление прерыванием	есть
Характеристики USB 2.0	
Число EndPoint	16
Размер буфера передачи/приёма, бит	16K × 32
Длина слова, бит	32
Размер пакета, байт	до 512
Формирование сигнала прерывания	есть
Управление прерыванием	есть
Ширина обмениваемых данных, бит	32
Характеристики MIL-STD-1553	
Стандарт	ГОСТ Р 52070-2003
Число каналов передачи/приёма	2
Число адресуемых оконечных устройств	31
Режим работы	монитор шины, контроллер шины, оконечное устройство
Размер буфера приёма/передачи данных, байт	6K
Контроль по чётности	есть
Режим обмена с буфером приёма/передачи	прямой доступ
Ширина обмениваемых данных, бит	16
Управление прерыванием	есть

Окончание таблицы 1.1

1	2
Характеристики устройства доступа к периферии (коммутатора)	
Функции	бесконфликтный доступ к периферийным устройствам из ПЦОС
Характеристики блока внутрисхемной мультипроцессорной эмуляции	
Интерфейс	в соответствии со стандартом IEEE 1149.1 (JTAG)
Количество поддерживаемых процессоров	не менее 2

В таблице 1.2 приведены электрические характеристики ИС 1867ВЦ8Ф1, в таблице 1.3 представлены значения предельно допустимых режимов эксплуатации микросхем в диапазоне рабочих температур.

Номинальное значение напряжения питания ИС 1867ВЦ8Ф1:

- буферов ввода-вывода U_{CC1} должно быть 3,3 В;
- ядра U_{CC2} должно быть 1,8 В.

Допустимые отклонения значения напряжения питания буферов ввода-вывода от номинального должны быть не более ± 0,3 В, ядра – не более ± 0,18 В.

Таблица 1.2 – Электрические параметры ИС 1867ВЦ8Ф1 при приёмке и поставке в диапазоне рабочих температур окружающей среды от минус 60 ºС до плюс 85 ºС

Наименование параметра, единица измерения, режим измерения		Буквенное обозначение параметра	Норма параметра		Температура среды, ºС
			не менее	не более	
1 Выходное напряжение низкого уровня буферов ввода-вывода, В, U_{CC1} = 3,0 В, U_{CC2} = 1,62 В, I_{OL} = 1,5 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} = 1,62 В, I_{OH} = –0,3 мА		U_{OH}	2,4	–	
3 Входной ток низкого уровня, мкА, U_{CC1} = 3,6 В, U_{CC2} = 1,98 В, U_{IL} = 0 В	Входы «pulldown»	I_{IL}	–10	10	
	Входы «pullup»		–400	–10	
	Вход X2/CLKIN_COMM		–50	50	
	Все остальные входы		–10	10	
4 Входной ток высокого уровня, мкА, U_{CC1} = 3,6 В, U_{CC2} = 1,98 В, $U_{IH} = U_{CC1}$	Входы «pulldown»	I_{IH}	10	800	
	Входы «pullup»		–10	10	
	Вход X2/CLKIN_COMM		–50	50	
	Все остальные входы		–10	10	
5 Выходной ток низкого уровня буфера с третьим состоянием в состоянии «Выключено», мкА, U_{CC1} = 3,6 В, U_{CC2} = 1,98 В, U_{OZL} = 0 В		I_{OZL}	–10	10	
6 Выходной ток высокого уровня буфера с третьим состоянием в состоянии «Выключено», мкА, U_{CC1} = 3,6 В, U_{CC2} = 1,98 В, $U_{OZH} = U_{CC1}$		I_{OZH}	–10	10	
7 Динамический ток потребления ядра, мА, U_{CC1} = 3,6 В, U_{CC2} = 1,98 В, f_{CI} = 100 МГц		I_{OCC}	–	1000	

Примечания

1 Входы «pulldown» – CntrlPLL, TRST_COMM#, ROMEN_1, ROMEN_2.

2 Входы «pullup» – RESET_COMM#, NMI_1#, NMI_2#.

3 $U_{CC1} = U_{\#VCC\_DR}$, $U_{CC2} = U_{\#VCC\_CL} = U_{\cap VCC\_A\_CPUPLL} = U_{\cap VCC\_A\_UARTPLL}$.

4 Параметры I_{IL}, I_{IH}, I_{OZL}, I_{OZH} при температуре минус 60 ºС не измеряются, а гарантируются нормами при температуре (25 ± 10) ºС.

5 Функциональный контроль (ФК) проводится при U_{CC1} = (3,0; 3,6) В, U_{CC2} = (1,62; 1,98) В, f_{CI} = 100 МГц.

Таблица 1.3 – Предельно допустимые и предельные режимы эксплуатации микросхем в диапазоне рабочих температур от минус 60 ºС до 85 ºС

Наименование параметра режима, единица измерения	Буквенное обозначение параметра	Предельно допустимый режим		Предельный режим	
		не менее	не более	не менее	не более
1 Напряжение питания буферов ввода-вывода, В	U_{CC1}	3,0	3,6	–0,3	4,4
2 Напряжение питания ядра, В	U_{CC2}	1,62	1,98	–0,3	2,4
3 Входное напряжение низкого уровня на входах, В	U_{IL}	–0,3	0,8	–0,6	–
4 Входное напряжение высокого уровня на входах, В	U_{IH}	2	U_{CC1} + 0,3	–	U_{CC1} + 0,6
5 Выходное напряжение на выходе с третьим состоянием в состоянии «Выключено», В	U_{OZ}	–0,3	U_{CC1} + 0,3	–0,6	U_{CC1} + 0,6
6 Выходной ток низкого уровня, мА	I_{OL}	–	1,5	–	3
7 Выходной ток высокого уровня, мА	I_{OH}	–0,3	–	–1	–
8 Частота следования импульсов тактового сигнала, МГц	f_{CI}	–	100	–	–
9 Ёмкость нагрузки, пФ	C_L	–	60	–	80
Примечания 1 $U_{CC1} = U_{\#VCC\_DR}$, $U_{CC2} = U_{\#VCC\_CL} = U_{\cap VCC\_A\_CPUPLL} = U_{\cap VCC\_A\_UARTPLL}$. 2 Время работы в одном из предельных режимов должно быть не более 5 с (кроме параметра 9).					

1.2 Краткое описание ИС 1867ВЦ8Ф1

1.2.1 Структура ИС

ИС 1867ВЦ8Ф1 содержит следующие основные блоки (см. рисунок 1.1):

- ядро ПЦОС 1;
- ядро ПЦОС 2;
- блок управления сбросом (блок сброса);
- блок PLL ПЦОС;
- блок коммутатора (COMMUTATOR);
- блок преобразователя сигналов локальной шины ПЦОС в сигналы шины Wishbone (LB_WB_MEM 16K × 32 бит);
- блок преобразователя сигналов локальной шины ПЦОС в сигналы шины AMBA AHB (LB_AHB_MEM 16K × 32 бит);
- периферийное устройство UART с архитектурой UART 16550;
- периферийное устройство USB 2.0, FIFO 8 × 32 бит, 16 × 32 бит;
- периферийное устройство Ethernet 10/100 FIFO передачи 2K × 40 бит, FIFO приёма 4K × 36 бит;
- периферийное устройство MIL-STD-1553, ОЗУ 3K × 18 бит;
- блок PLL UART.

Ниже приведено краткое описание и функциональное назначение блоков ИС 1867ВЦ8Ф1.

1.2.1.1 Ядро ПЦОС 1 и ядро ПЦОС 2

Ядра ПЦОС 1 и ПЦОС 2 представляют собой 32-разрядные процессоры цифровой обработки сигналов с плавающей запятой. Архитектура процессорных ядер функционально совместима с цифровым процессором 1867ВЦ3Ф (функциональный аналог 32-разрядного DSP TMS320C40 фирмы Texas Instruments). Функциональная совместимость означает, что программные средства (компиляторы с языков C/C++, отладчики, например, Code Composer и т. п. программные средства), которые поддерживают ИС 1867ВЦ3Ф и её функциональный аналог DSP TMS320C40, будут также поддерживать и оба процессорных ядра 1867ВЦ8Ф1.

Внутри ИС ядра ПЦОС 1 и ПЦОС 2 соединены друг с другом через их коммуникационные порты COMM3 и COMM0, соответственно.

Коммуникационные порты COMM0 – COMM2 и COMM4, COMM5 ядра ПЦОС 1 и COMM1 – COMM5 ядра ПЦОС 2 выведены на внешние выводы ИС 1867ВЦ8Ф1. На внешние выводы ИС 1867ВЦ8Ф1 выведены сигналы глобальной шины ядра ПЦОС 1 и ядра ПЦОС 2 (управляющие сигналы, сигналы шины данных и сигналы адресной шины).

Сигналы входов/выходов общего назначения и прерывания IIOF(0 – 3)_x# подключены к ядру ПЦОС 1 (IIOF0_1# – IIOF3_1#) и ядру ПЦОС 2 (IIOF0_2# – IIOF3_2#), соответственно, через коммутатор.

Ядро ПЦОС 1 и ядро ПЦОС 2 тактируются от блока PLL ПЦОС с частотой до 100 МГц. Сигналы локальной шины (управляющие, адреса и данных) от ядра ПЦОС 1 и от ядра ПЦОС 2 поступают на коммутатор.

Подробное описание ядер ПЦОС 1 и ПЦОС 2 приведено в разделе 5 данного Руководства.

1.2.1.2 Блок управления сбросом

Блок управления сбросом осуществляет формирование внутренних сигналов сброса для всех блоков ИС 1867ВЦ8Ф1 при установлении активного уровня на выводе RS_COMM.

1.2.1.3 Блок PLL ПЦОС

Блок PLL ПЦОС осуществляет формирование тактирующего сигнала, подаваемого на входы тактирования ядер ПЦОС 1 и ПЦОС 2. Блок PLL ПЦОС умножает на 10 частоту, входного сигнала, поступающего на вывод X2/CLKIN_COMM.

Максимальная тактовая частота, подаваемая на вход X2/CLKIN_COMM при включенном блоке PLL ПЦОС (на выводе CntrlPLL уровень логического «0»), не должна превышать 10 МГц.

1.2.1.4 Коммутатор

Коммутатор реализует подключение локальных шин ПЦОС 1 и ПЦОС 2 к периферийным устройствам (UART 16550, USB 2.0, Ethernet 10/100, MIL-STD-1553 и PLL UART). Подключение (коммутация) периферийного устройства к одному из ядер ПЦОС 1 и ПЦОС 2 осуществляется программно и может выполняться в любое время, если разрешён доступ к регистру коммутации. Разрешение доступа к регистру коммутации одного из ядер ПЦОС 1 и ПЦОС 2 выполняется арбитром коммутатора. Подробное описание архитектуры коммутатора приведено в разделе 6 данного Руководства.

1.2.1.5 Блок LB_WB_MEM

Блок LB_WB_MEM осуществляет преобразование сигналов локальной шины ядра ПЦОС 1 и ядра ПЦОС 2 в сигналы шины Wishbone. Этот блок содержит буферную память размером 16K × 32 для хранения принимаемых/передаваемых данных от ядра ПЦОС 1 и ядра ПЦОС 2 и устройства USB 2.0. Приём/передача сигналов в буфер осуществляется в режиме прямого доступа со стороны периферийного устройства USB 2.0. Ядро ПЦОС 1 и ядро ПЦОС 2 читают и записывают данные в буфер программно. Доступ к буферу со стороны ядра ПЦОС 1 и ядра ПЦОС 2 и устройства USB 2.0 осуществляется одновременно без тактов ожидания как со стороны устройства USB 2.0, так и со стороны ядра ПЦОС 1 и ядра ПЦОС 2.

1.2.1.6 Блок LB_AHB_MEM

Блок LB_AHB_MEM осуществляет преобразование сигналов локальной шины ядра ПЦОС 1 и ПЦОС 2 в сигналы шины AMBA AHB. Этот блок содержит буферную память размером 16K × 32 для хранения принимаемых/передаваемых данных от ядра ПЦОС 1 и ядра ПЦОС 2 и устройства Ethernet 10/100. Приём/передача сигналов в буфер осуществляется в режиме прямого доступа со стороны периферийного устройства Ethernet 10/100. Ядро ПЦОС 1 и ядро ПЦОС 2 читают и записывают данные в буфер программно. Доступ к буферу со стороны ядра ПЦОС 1 и ядра ПЦОС 2 и устройства Ethernet 10/100 осуществляется последовательно с тактами ожидания как со стороны устройства Ethernet 10/100, так и со стороны ядра ПЦОС 1 и ядра ПЦОС 2. Если в данный момент времени осуществляется чтение/запись в буфер со стороны устройства Ethernet 10/100, то ядро ПЦОС 1 и ядро ПЦОС 2 будет находиться в режиме ожидания (сигнал готовности LRDY будет пассивным) и наоборот, если ядро ПЦОС 1 и ядро ПЦОС 2 будет осуществлять доступ к буферу, то сигнал шины AHB ahb_pgrant будет пассивным.

Примечание – Для уменьшения времени ожидания со стороны устройств необходимо планирование доступа к буферу.

1.2.1.7 Периферийное устройство UART 16550

Периферийное устройство UART 16550 осуществляет приём/передачу данных по последовательной шине в асинхронном режиме. Устройство также имеет выводы управления и состояния модема. Подробнее периферийное устройство UART 16550 описано в разделе 10.

1.2.1.8 Периферийное устройство USB 2.0

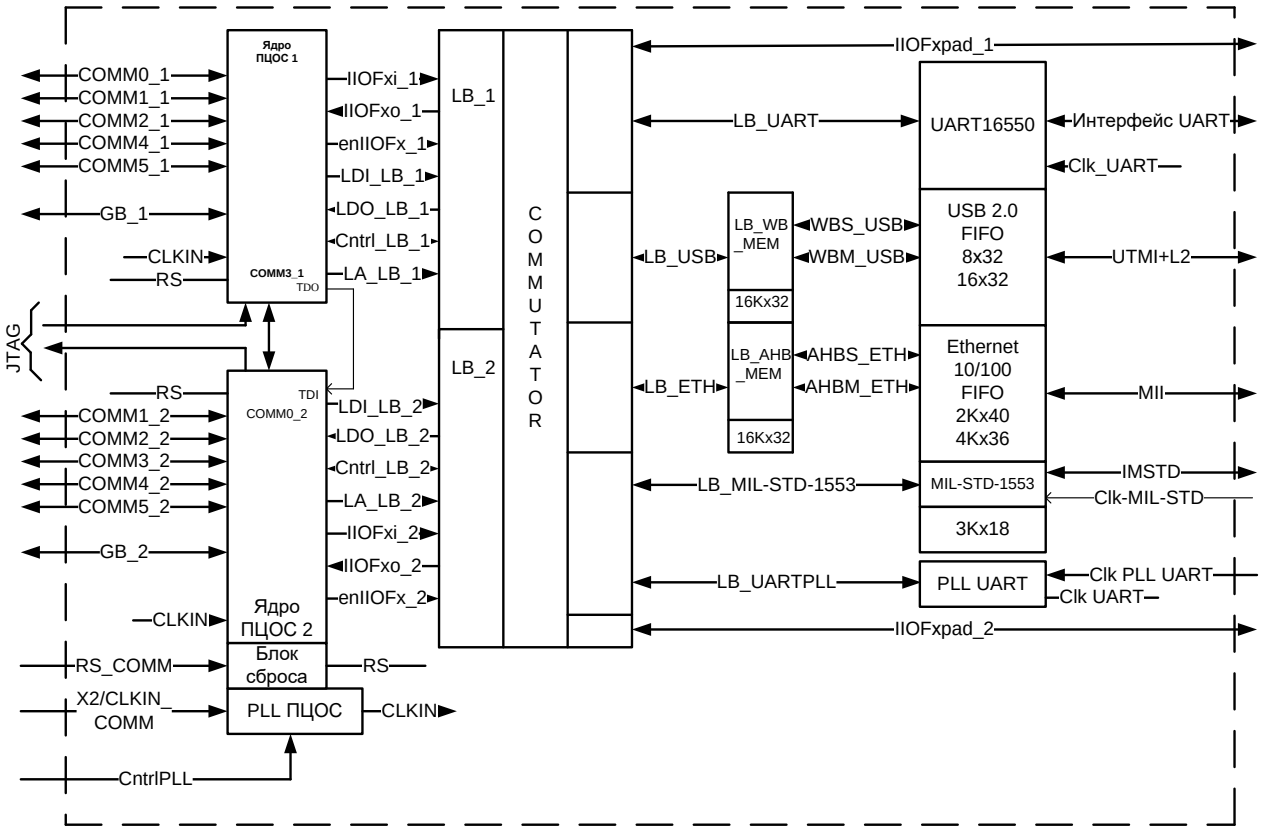
Периферийное устройство USB 2.0 реализует стандарт Universal Serial Bus Specification, Revision 2, Chapter 9. Оно осуществляет приём/передачу данных по интерфейсу UTMI в режимах Full Speed (12 Мбит/с) и High Speed (480 Мбит/с). Приём/передача данных, передаваемых по интерфейсу UTMI, осуществляется в/из буфера блока LB_WB_MEM размером 16K × 32. Устройство USB 2.0 имеет 16 EndPoint. Каждому EndPoint может быть назначен любой адрес в буфере. Длина пакета передаваемых/принимаемых данных не должна превышать 512 байт. EndPoint с номером 0 является управляющим и передающим/принимающим. Остальные EndPoint могут быть запрограммированы как принимающие/передающие EndPoint.

1.2.1.9 Периферийное устройство Ethernet 10/100

Периферийное устройство Ethernet 10/100 реализует стандарт IEEE 802.3. Оно осуществляет приём/передачу данных по интерфейсу MII на скорости 10/100 Мбит/с. Приём/передача данных по интерфейсу MII осуществляется в/из буфера блока LB_AHB_MEM размером 16K × 32.

1.2.2 Структурная схема ИС 1867ВЦ8Ф1

На рисунке 1.1 приведена структурная схема ИС 1867ВЦ8Ф1.



Примечание − x = 0, 1, 2, 3.

Рисунок 1.1 − Структурная схема ИС 1867ВЦ8Ф1

Принятые условные обозначения на рисунке 1.1 приведены в таблице 1.4.

Таблица 1.4 − Условные обозначения на структурной схеме ИС 1867ВЦ8Ф1

Условное обозначение	Описание
1	2
COMM0_1 COMM1_1 COMM2_1 COMM4_1 COMM5_1	Сигналы коммуникационных портов 0, 1, 2, 4, 5 ядра ПЦОС 1
COMM3_1	Сигнал коммуникационного порта ядра ПЦОС 1 для внутреннего соединения с ядром ПЦОС 2
GB_1	Сигналы глобальной шины ядра ПЦОС 1
CLKIN	Сигнал тактирования ядра ПЦОС 1 и ядра ПЦОС 2
RS	Сигнал сброса ядра ПЦОС 1 и ядра ПЦОС 2
COMM1_2 COMM2_2 COMM3_2 COMM4_2 COMM5_2	Сигналы коммуникационных портов 1, 2, 3, 4, 5 ядра ПЦОС 2
COMM0_2	Сигнал коммуникационного порта ядра ПЦОС 2 для внутреннего соединения с ядром ПЦОС 1
GB_2	Сигналы глобальной шины ядра ПЦОС 2

Продолжение таблицы 1.4

1	2
RS_COMM	Сигнал сброса
CLKIN	Сигнал тактирования ядра ПЦОС 1 и ядра ПЦОС 2
X2_CLKIN_COMM	Сигнал тактирования, подаваемый на блок PLL ПЦОС
CntrlPLL	Сигнал включения блока PLL ПЦОС (CntrlPLL = логическому 0) или выключения (CntrlPLL = логической 1)
IIOFxi_1	Входные сигналы общего назначения и сигналы прерывания IIOF3_1# – IIOF0_1# ядра ПЦОС 1
IIOFxo_1	Выходные сигналы общего назначения IIOF3_1# – IIOF0_1# ядра ПЦОС 1
enIIOFx_1	Сигналы разрешения выходных сигналов общего назначения ядра ПЦОС 1
LDI_LB_1	Входные сигналы шины данных локальной шины ядра ПЦОС 1
LDO_LB_1	Выходные сигналы шины данных локальной шины ядра ПЦОС 1
Cntrl_LB_1	Сигналы управления локальной шины ядра ПЦОС 1
LA_LB_1	Сигналы адресной шины локальной шины ядра ПЦОС 1
LDI_LB_2	Входные сигналы шины данных локальной шины ядра ПЦОС 2
LDO_LB_2	Выходные сигналы шины данных локальной шины ядра ПЦОС 2
Cntrl_LB_2	Сигналы управления локальной шины ядра ПЦОС 2
LA_LB_2	Сигналы адресной шины локальной шины ядра ПЦОС 2
IIOFxi_2	Входные сигналы общего назначения и сигналы прерывания IIOF3_2# – IIOF0_2# ядра ПЦОС 2
IIOFxo_2	Выходные сигналы общего назначения IIOF3_2# – IIOF0_2# ядра ПЦОС 2
enIIOFx_2	Сигналы разрешения выходных сигналов общего назначения ядра ПЦОС 2
IIOFxpad_1	Входные/выходные сигналы общего назначения и прерывания, поступающие от внешних выводов на ядро ПЦОС 1
LB_UART	Сигналы локальной шины (управляющие, шина данных и шина адреса), поступающие на периферийное устройство UART 16550
LB_USB	Сигналы локальной шины (управляющие, шина данных и шина адреса), поступающие на периферийное устройство USB 2.0
LB_ETH	Сигналы локальной шины (управляющие, шина данных и шина адреса), поступающие на периферийное устройство Ethernet 10/100
LB_MIL-STD-1553	Сигналы локальной шины (управляющие, шина данных и шина адреса), поступающие на периферийное устройство MilStd-1553
LB_UARTPLL	Сигналы локальной шины (управляющие, шина данных и шина адреса), поступающие на периферийное устройство PLL UART
IIOFxpad_2	Входные/выходные сигналы общего назначения и прерывания, поступающие от внешних выводов на ядро ПЦОС 2
Clk_UART	Сигнал тактирования, поступающий на периферийное устройство UART от блока PLL UART
Clk PLL UART	Сигнал тактирования, поступающий на блок PLL UART
UTMI+L2	Сигналы интерфейса UTMI+L2, поступающие от устройства физического уровня шины USB 2.0
MII	Сигналы интерфейса MII, поступающие от устройства физического уровня шины IEEE 802.3

Окончание таблицы 1.4

1	2
IMSTD	Сигналы интерфейса MIL-STD, поступающие от устройства физического уровня шины MIL-STD-1553
LB_WB_MEM	Блок преобразования сигналов локальной шины в сигналы шины Wishbone
LB_AHB_MEM	Блок преобразования сигналов локальной шины в сигналы шины AHB
WBS_USB	Сигналы шины Wishbone к устройству USB
WBM_USB	Сигналы шины Wishbone к ОЗУ от устройства USB
AHBS_ETH	Сигналы шины AHB к устройству Ethernet
AHBM_ETH	Сигналы шины AHB к ОЗУ от устройства Ethernet
PLL UART	Блок генератора – умножитель частоты для UART
PLL ПЦОС	Блок генератора – умножитель частоты для ПЦОС

Условное графическое обозначение ИС 1867ВЦ8Ф1 приведено на рисунке 1.2.

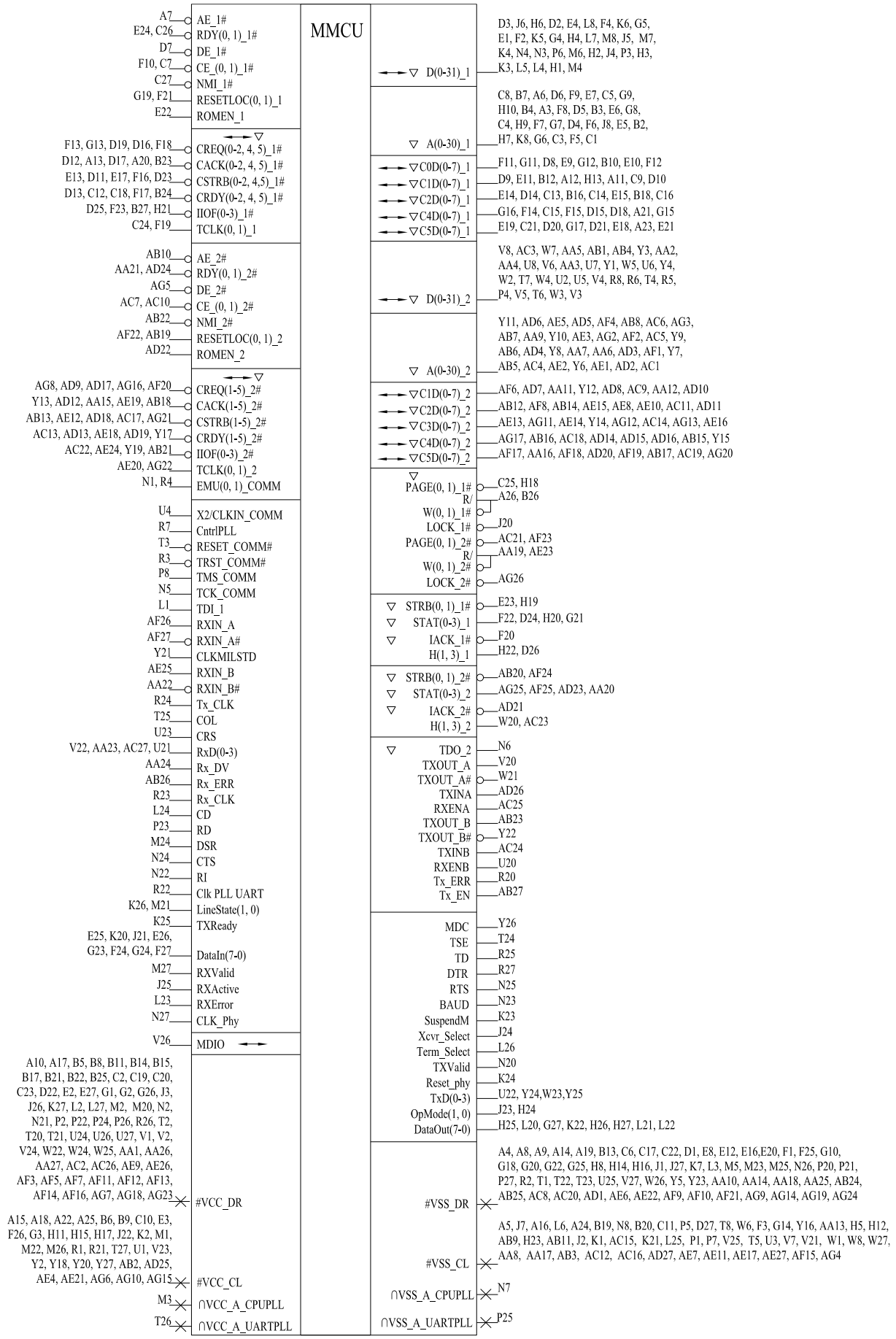
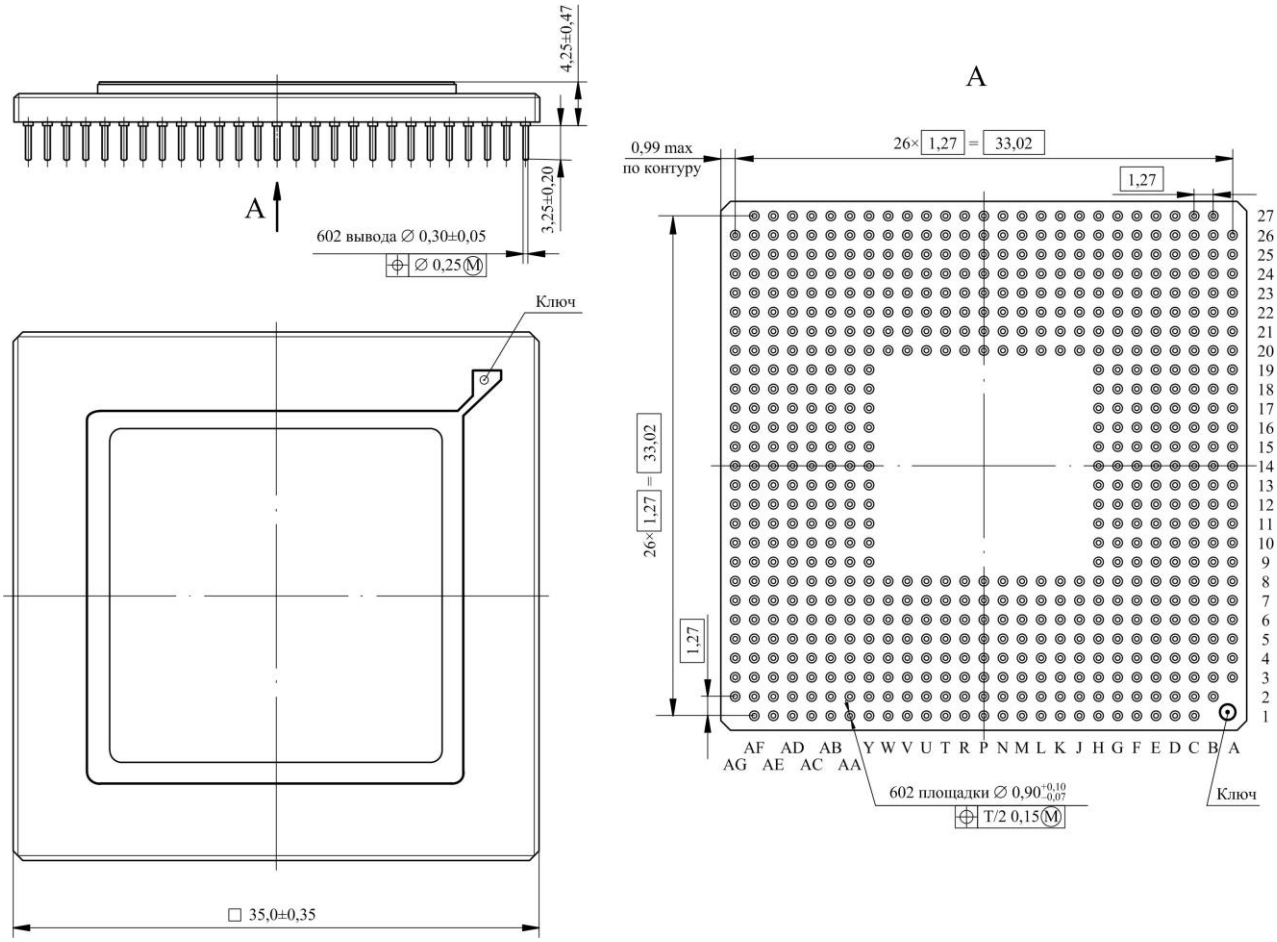


Рисунок 1.2 – Условное графическое обозначение ИС 1867ВЦ8Ф1

2 Конструктивное исполнение микросхемы 1867ВЦ8Ф1

Конструктивное исполнение микросхемы 1867ВЦ8Ф1 приведено на рисунке 2.1.



1 Нумерация выводов показана условно.
2 Конфигурация ключа не регламентируется.

Рисунок 2.1 − Конструктивное исполнение микросхемы 1867ВЦ8Ф1
в корпусе МК 6117.602-D (УКВД.430109.577ГЧ)

3 Функциональное назначение выводов ИС 1867ВЦ8Ф1

В таблице 3.1 приведено функциональное назначение выводов ИС.

Таблица 3.1 − Функциональное назначение выводов ИС 1867ВЦ8Ф1

Координаты вывода	Обозначение вывода	Описание	Тип вывода
1	2	3	4
Выводы ядра ПЦОС 1			
Внешний интерфейс глобальной шины			
A7	AE_1#	Сигнал разрешения адресной шины для внешнего интерфейса глобальной шины	I
E23	STRB0_1#	Сигнал строба 0 для внешнего интерфейса глобальной шины, выход с третьим состоянием	O/Z
G21	STAT3_1	Сигнал состояния 3 для внешнего интерфейса глобальной шины, выход с третьим состоянием	O/Z
H20	STAT2_1	Сигнал состояния 2 для внешнего интерфейса глобальной шины, выход с третьим состоянием	O/Z
D24	STAT1_1	Сигнал состояния 1 для внешнего интерфейса глобальной шины, выход с третьим состоянием	O/Z
F22	STAT0_1	Сигнал состояния 0 для внешнего интерфейса глобальной шины, выход с третьим состоянием	O/Z
H19	STRB1_1#	Сигнал строба 1 для внешнего интерфейса глобальной шины, выход с третьим состоянием	O/Z
E24	RDY0_1#	Сигнал готовности для сигнала STRB0	I
C26	RDY1_1#	Сигнал готовности для сигнала STRB1	I
D7	DE_1#	Сигнал разрешения шины данных для внешнего интерфейса глобальной шины	I
F10	CE_0_1#	Сигнал разрешения для управляющих сигналов STRB0, PAGE0 и R/W0	I
C7	CE_1_1#	Сигнал разрешения для управляющих сигналов STRB1, PAGE1 и R/W1	I
C25	PAGE0_1#	Сигнал страницы для сигнала STRB0, выход с третьим состоянием	O/Z
H18	PAGE1_1#	Сигнал страницы для сигнала STRB1, выход с третьим состоянием	O/Z
A26	R/W0_1#	Сигнал чтения/записи для сигнала STRB0, выход с третьим состоянием	O/Z
B26	R/W1_1#	Сигнал чтения/записи для сигнала STRB1, выход с третьим состоянием	O/Z
J20	LOCK_1#	Сигнал блокировки для внешнего интерфейса глобальной шины, выход с третьим состоянием	O/Z
D3	D0_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 0	I/O/Z
J6	D1_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 1	I/O/Z
H6	D2_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 2	I/O/Z
D2	D3_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 3	I/O/Z
E4	D4_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 4	I/O/Z
L8	D5_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 5	I/O/Z

Продолжение таблицы 3.1

1	2	3	4
F4	D6_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 6	I/O/Z
K6	D7_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 7	I/O/Z
G5	D8_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 8	I/O/Z
E1	D9_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 9	I/O/Z
F2	D10_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 10	I/O/Z
K5	D11_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 11	I/O/Z
G4	D12_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 12	I/O/Z
H4	D13_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 13	I/O/Z
L7	D14_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 14	I/O/Z
M8	D15_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 15	I/O/Z
J5	D16_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 16	I/O/Z
M7	D17_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 17	I/O/Z
K4	D18_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 18	I/O/Z
N4	D19_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 19	I/O/Z
N3	D20_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 20	I/O/Z
P6	D21_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 21	I/O/Z
M6	D22_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 22	I/O/Z
H2	D23_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 23	I/O/Z
J4	D24_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 24	I/O/Z
P3	D25_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 25	I/O/Z
H3	D26_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 26	I/O/Z
K3	D27_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 27	I/O/Z
L5	D28_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 28	I/O/Z
L4	D29_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 29	I/O/Z
H1	D30_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 30	I/O/Z
M4	D31_1	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 31	I/O/Z
C8	A0_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 0 бит	O/Z
B7	A1_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 1 бит	O/Z
A6	A2_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 2 бит	O/Z

Продолжение таблицы 3.1

1	2	3	4
D6	A3_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 3 бит	O/Z
F9	A4_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 4 бит	O/Z
E7	A5_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 5 бит	O/Z
C5	A6_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 6 бит	O/Z
G9	A7_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 7 бит	O/Z
H10	A8_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 8 бит	O/Z
B4	A9_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 9 бит	O/Z
A3	A10_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 10 бит	O/Z
F8	A11_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 11 бит	O/Z
D5	A12_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 12 бит	O/Z
B3	A13_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 13 бит	O/Z
E6	A14_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 14 бит	O/Z
G8	A15_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 15 бит	O/Z
C4	A16_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 16 бит	O/Z
H9	A17_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 17 бит	O/Z
F7	A18_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 18 бит	O/Z
G7	A19_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 19 бит	O/Z
D4	A20_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 20 бит	O/Z
F6	A21_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 21 бит	O/Z
J8	A22_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 22 бит	O/Z
E5	A23_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 23 бит	O/Z
B2	A24_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 24 бит	O/Z
H7	A25_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 25 бит	O/Z
K8	A26_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 26 бит	O/Z
G6	A27_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 27 бит	O/Z
C3	A28_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 28 бит	O/Z
F5	A29_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 29 бит	O/Z
C1	A30_1	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 30 бит	O/Z

Продолжение таблицы 3.1

1	2	3	4
Интерфейс коммуникационного порта 0			
F11	C0D0_1	Шина данных коммуникационного порта 0, вход/выход с третьим состоянием, бит 0	I/O/Z
G11	C0D1_1	Шина данных коммуникационного порта 0, вход/выход с третьим состоянием, бит 1	I/O/Z
D8	C0D2_1	Шина данных коммуникационного порта 0, вход/выход с третьим состоянием, бит 2	I/O/Z
E9	C0D3_1	Шина данных коммуникационного порта 0, вход/выход с третьим состоянием, бит 3	I/O/Z
G12	C0D4_1	Шина данных коммуникационного порта 0, вход/выход с третьим состоянием, бит 4	I/O/Z
B10	C0D5_1	Шина данных коммуникационного порта 0, вход/выход с третьим состоянием, бит 5	I/O/Z
E10	C0D6_1	Шина данных коммуникационного порта 0, вход/выход с третьим состоянием, бит 6	I/O/Z
F12	C0D7_1	Шина данных коммуникационного порта 0, вход/выход с третьим состоянием, бит 7	I/O/Z
F13	CREQ0_1#	Сигнал запроса коммуникационного порта 0, вход/выход с третьим состоянием	I/O/Z
D12	CACK0_1#	Сигнал подтверждения на сигнал запроса коммуникационного порта 0, вход/выход с третьим состоянием	I/O/Z
E13	CSTRB0_1#	Сигнал стробирования данных коммуникационного порта 0, вход/выход с третьим состоянием	I/O/Z
D13	CRDY0_1#	Сигнал готовности данных коммуникационного порта 0, вход/выход с третьим состоянием	I/O/Z
Интерфейс коммуникационного порта 1			
D9	C1D0_1	Шина данных коммуникационного порта 1, вход/выход с третьим состоянием, бит 0	I/O/Z
E11	C1D1_1	Шина данных коммуникационного порта 1, вход/выход с третьим состоянием, бит 1	I/O/Z
B12	C1D2_1	Шина данных коммуникационного порта 1, вход/выход с третьим состоянием, бит 2	I/O/Z
A12	C1D3_1	Шина данных коммуникационного порта 1, вход/выход с третьим состоянием, бит 3	I/O/Z
H13	C1D4_1	Шина данных коммуникационного порта 1, вход/выход с третьим состоянием, бит 4	I/O/Z
A11	C1D5_1	Шина данных коммуникационного порта 1, вход/выход с третьим состоянием, бит 5	I/O/Z
C9	C1D6_1	Шина данных коммуникационного порта 1, вход/выход с третьим состоянием, бит 6	I/O/Z
D10	C1D7_1	Шина данных коммуникационного порта 1, вход/выход с третьим состоянием, бит 7	I/O/Z
G13	CREQ1_1#	Сигнал запроса коммуникационного порта 1, вход/выход с третьим состоянием	I/O/Z
A13	CACK1_1#	Сигнал подтверждения на сигнал запроса коммуникационного порта 1, вход/выход с третьим состоянием	I/O/Z
D11	CSTRB1_1#	Сигнал стробирования данных коммуникационного порта 1, вход/выход с третьим состоянием	I/O/Z
C12	CRDY1_1#	Сигнал готовности данных коммуникационного порта 1, вход/выход с третьим состоянием	I/O/Z

Продолжение таблицы 3.1

1	2	3	4
Интерфейс коммуникационного порта 2			
E14	C2D0_1	Шина данных коммуникационного порта 2, вход/выход с третьим состоянием, бит 0	I/O/Z
D14	C2D1_1	Шина данных коммуникационного порта 2, вход/выход с третьим состоянием, бит 1	I/O/Z
C13	C2D2_1	Шина данных коммуникационного порта 2, вход/выход с третьим состоянием, бит 2	I/O/Z
B16	C2D3_1	Шина данных коммуникационного порта 2, вход/выход с третьим состоянием, бит 3	I/O/Z
C14	C2D4_1	Шина данных коммуникационного порта 2, вход/выход с третьим состоянием, бит 4	I/O/Z
E15	C2D5_1	Шина данных коммуникационного порта 2, вход/выход с третьим состоянием, бит 5	I/O/Z
B18	C2D6_1	Шина данных коммуникационного порта 2, вход/выход с третьим состоянием, бит 6	I/O/Z
C16	C2D7_1	Шина данных коммуникационного порта 2, вход/выход с третьим состоянием, бит 7	I/O/Z
D19	CREQ2_1#	Сигнал запроса коммуникационного порта 2, вход/выход с третьим состоянием	I/O/Z
D17	CACK2_1#	Сигнал подтверждения на сигнал запроса коммуникационного порта 2, вход/выход с третьим состоянием	I/O/Z
E17	CSTRB2_1#	Сигнал стробирования данных коммуникационного порта 2, вход/выход с третьим состоянием	I/O/Z
C18	CRDY2_1#	Сигнал готовности данных коммуникационного порта 2, вход/выход с третьим состоянием	I/O/Z
Интерфейс коммуникационного порта 4			
G16	C4D0_1	Шина данных коммуникационного порта 4, вход/выход с третьим состоянием, бит 0	I/O/Z
F14	C4D1_1	Шина данных коммуникационного порта 4, вход/выход с третьим состоянием, бит 1	I/O/Z
C15	C4D2_1	Шина данных коммуникационного порта 4, вход/выход с третьим состоянием, бит 2	I/O/Z
F15	C4D3_1	Шина данных коммуникационного порта 4, вход/выход с третьим состоянием, бит 3	I/O/Z
D15	C4D4_1	Шина данных коммуникационного порта 4, вход/выход с третьим состоянием, бит 4	I/O/Z
D18	C4D5_1	Шина данных коммуникационного порта 4, вход/выход с третьим состоянием, бит 5	I/O/Z
A21	C4D6_1	Шина данных коммуникационного порта 4, вход/выход с третьим состоянием, бит 6	I/O/Z
G15	C4D7_1	Шина данных коммуникационного порта 4, вход/выход с третьим состоянием, бит 7	I/O/Z
D16	CREQ4_1#	Сигнал запроса коммуникационного порта 4, вход/выход с третьим состоянием	I/O/Z
A20	CACK4_1#	Сигнал подтверждения на сигнал запроса коммуникационного порта 4, вход/выход с третьим состоянием	I/O/Z
F16	CSTRB4_1#	Сигнал стробирования данных коммуникационного порта 4, вход/выход с третьим состоянием	I/O/Z
F17	CRDY4_1#	Сигнал готовности данных коммуникационного порта 4, вход/выход с третьим состоянием	I/O/Z

Продолжение таблицы 3.1

1	2	3	4
Интерфейс коммуникационного порта 5			
E19	C5D0_1	Шина данных коммуникационного порта 5, вход/выход с третьим состоянием, бит 0	I/O/Z
C21	C5D1_1	Шина данных коммуникационного порта 5, вход/выход с третьим состоянием, бит 1	I/O/Z
D20	C5D2_1	Шина данных коммуникационного порта 5, вход/выход с третьим состоянием, бит 2	I/O/Z
G17	C5D3_1	Шина данных коммуникационного порта 5, вход/выход с третьим состоянием, бит 3	I/O/Z
D21	C5D4_1	Шина данных коммуникационного порта 5, вход/выход с третьим состоянием, бит 4	I/O/Z
E18	C5D5_1	Шина данных коммуникационного порта 5, вход/выход с третьим состоянием, бит 5	I/O/Z
A23	C5D6_1	Шина данных коммуникационного порта 5, вход/выход с третьим состоянием, бит 6	I/O/Z
E21	C5D7_1	Шина данных коммуникационного порта 5, вход/выход с третьим состоянием, бит 7	I/O/Z
F18	CREQ5_1#	Сигнал запроса коммуникационного порта 5, вход/выход с третьим состоянием	I/O/Z
B23	CACK5_1#	Сигнал подтверждения на сигнал запроса коммуникационного порта 5, вход/выход с третьим состоянием	I/O/Z
D23	CSTRB5_1#	Сигнал стробирования данных коммуникационного порта 5, вход/выход с третьим состоянием	I/O/Z
B24	CRDY5_1#	Сигнал готовности данных коммуникационного порта 5, вход/выход с третьим состоянием	I/O/Z
Прерывания, флаги ввода/вывода, вектор сброса, таймер			
D25	IIOF0_1#	Прерывание и флаг ввода/вывода 0, вход/выход с третьим состоянием	I/O/Z
F23	IIOF1_1#	Прерывание и флаг ввода/вывода 1, вход/выход с третьим состоянием	I/O/Z
B27	IIOF2_1#	Прерывание и флаг ввода/вывода 2, вход/выход с третьим состоянием	I/O/Z
H21	IIOF3_1#	Прерывание и флаг ввода/вывода 3, вход/выход с третьим состоянием	I/O/Z
F20	IACK_1#	Подтверждение прерывания, выход с третьим состоянием	O/Z
C27	NMI_1#	Немаскируемое прерывание. Вывод “подтянут” к 1	I
G19	RESETLOC0_1	Вывод 0 вектора сброса	I
F21	RESETLOCl_1	Вывод 1 вектора сброса	I
E22	ROMEN_1	Разрешение внутрикристального ПЗУ (0 = запрещено, 1 = разрешено). Вывод “подтянут” к 0	I
C24	TCLK0_1	Вывод таймера 0, вход/выход с третьим состоянием	I/O/Z
F19	TCLK1_1	Вывод таймера 1, вход/выход с третьим состоянием	I/O/Z
Выводы ядра ПЦОС 2			
Внешний интерфейс глобальной шины			
AB10	AE_2#	Сигнал разрешения адресной шины для внешнего интерфейса глобальной шины	I
AB20	STRB0_2#	Сигнал строба 0 для внешнего интерфейса глобальной шины, выход с третьим состоянием	O/Z
AA20	STAT3_2	Сигнал состояния 3 для внешнего интерфейса глобальной шины, выход с третьим состоянием	O/Z
AD23	STAT2_2	Сигнал состояния 2 для внешнего интерфейса глобальной шины, выход с третьим состоянием	O/Z
AF25	STAT1_2	Сигнал состояния 1 для внешнего интерфейса глобальной шины, выход с третьим состоянием	O/Z
AG25	STAT0_2	Сигнал состояния 0 для внешнего интерфейса глобальной шины, выход с третьим состоянием	O/Z
AF24	STRB1_2#	Сигнал строба 1 для внешнего интерфейса глобальной шины, выход с третьим состоянием	O/Z

Продолжение таблицы 3.1

1	2	3	4
AA21	RDY0_2#	Сигнал готовности для сигнала STRB0	I
AD24	RDY1_2#	Сигнал готовности для сигнала STRB1	I
AG5	DE_2#	Сигнал разрешения шины данных для внешнего интерфейса глобальной шины	I
AC7	CE_0_2#	Сигнал разрешения для управляющих сигналов STRB0, PAGE0 и R/W0	I
AC10	CE_1_2#	Сигнал разрешения для управляющих сигналов STRB1, PAGE1 и R/W1	I
AC21	PAGE0_2#	Сигнал страницы для сигнала STRB0, выход с третьим состоянием	O/Z
AF23	PAGE1_2#	Сигнал страницы для сигнала STRB1, выход с третьим состоянием	O/Z
AA19	R/ W0_2#	Сигнал чтения для сигнала STRB0, выход с третьим состоянием	O/Z
		Сигнал записи для сигнала STRB0, выход с третьим состоянием	
AE23	R/ W1_2#	Сигнал чтения для сигнала STRB1, выход с третьим состоянием	O/Z
		Сигнал записи для сигнала STRB1, выход с третьим состоянием	
AG26	LOCK_2#	Сигнал блокировки для внешнего интерфейса глобальной шины, выход с третьим состоянием	O/Z
V8	D0_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 0	I/O/Z
AC3	D1_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 1	I/O/Z
W7	D2_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 2	I/O/Z
AA5	D3_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 3	I/O/Z
AB1	D4_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 4	I/O/Z
AB4	D5_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 5	I/O/Z
Y3	D6_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 6	I/O/Z
AA2	D7_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 7	I/O/Z
AA4	D8_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 8	I/O/Z
U8	D9_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 9	I/O/Z
V6	D10_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 10	I/O/Z
AA3	D11_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 11	I/O/Z
U7	D12_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 12	I/O/Z
Y1	D13_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 13	I/O/Z
W5	D14_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 14	I/O/Z
U6	D15_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 15	I/O/Z
Y4	D16_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 16	I/O/Z
W2	D17_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 17	I/O/Z
T7	D18_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 18	I/O/Z
W4	D19_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 19	I/O/Z

Продолжение таблицы 3.1

1	2	3	4
U2	D20_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 20	I/O/Z
U5	D21_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 21	I/O/Z
V4	D22_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 22	I/O/Z
R8	D23_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 23	I/O/Z
R6	D24_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 24	I/O/Z
T4	D25_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 25	I/O/Z
R5	D26_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 26	I/O/Z
P4	D27_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 27	I/O/Z
V5	D28_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 28	I/O/Z
T6	D29_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 29	I/O/Z
W3	D30_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 30	I/O/Z
V3	D31_2	32-разрядный порт данных глобальной шины, вход/выход с третьим состоянием, бит 31	I/O/Z
Y11	A0_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 0 бит	O/Z
AD6	A1_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 1 бит	O/Z
AE5	A2_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 2 бит	O/Z
AD5	A3_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 3 бит	O/Z
AF4	A4_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 4 бит	O/Z
AB8	A5_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 5 бит	O/Z
AC6	A6_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 6 бит	O/Z
AG3	A7_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 7 бит	O/Z
AB7	A8_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 8 бит	O/Z
AA9	A9_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 9 бит	O/Z
Y10	A10_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 10 бит	O/Z
AE3	A11_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 11 бит	O/Z
AG2	A12_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 12 бит	O/Z
AF2	A13_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 13 бит	O/Z
AC5	A14_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 14 бит	O/Z

Продолжение таблицы 3.1

1	2	3	4
Y9	A15_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 15 бит	O/Z
AB6	A16_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 16 бит	O/Z
AD4	A17_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 17 бит	O/Z
Y8	A18_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 18 бит	O/Z
AA7	A19_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 19 бит	O/Z
AA6	A20_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 20 бит	O/Z
AD3	A21_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 21 бит	O/Z
AF1	A22_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 22 бит	O/Z
Y7	A23_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 23 бит	O/Z
AB5	A24_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 24 бит	O/Z
AC4	A25_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 25 бит	O/Z
AE2	A26_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 26 бит	O/Z
Y6	A27_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 27 бит	O/Z
AE1	A28_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 28 бит	O/Z
AD2	A29_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 29 бит	O/Z
AC1	A30_2	31-разрядный адресный порт глобальной шины, выход с третьим состоянием, 30 бит	O/Z
Интерфейс коммуникационного порта 1			
AF6	C1D0_2	Шина данных коммуникационного порта 1, вход/выход с третьим состоянием, бит 0	I/O/Z
AD7	C1D1_2	Шина данных коммуникационного порта 1, вход/выход с третьим состоянием, бит 1	I/O/Z
AA11	C1D2_2	Шина данных коммуникационного порта 1, вход/выход с третьим состоянием, бит 2	I/O/Z
Y12	C1D3_2	Шина данных коммуникационного порта 1, вход/выход с третьим состоянием, бит 3	I/O/Z
AD8	C1D4_2	Шина данных коммуникационного порта 1, вход/выход с третьим состоянием, бит 4	I/O/Z
AC9	C1D5_2	Шина данных коммуникационного порта 1, вход/выход с третьим состоянием, бит 5	I/O/Z
AA12	C1D6_2	Шина данных коммуникационного порта 1, вход/выход с третьим состоянием, бит 6	I/O/Z
AD10	C1D7_2	Шина данных коммуникационного порта 1, вход/выход с третьим состоянием, бит 7	I/O/Z
AG8	CREQ1_2#	Сигнал запроса коммуникационного порта 1, вход/выход с третьим состоянием	I/O/Z
Y13	CACK1_2#	Сигнал подтверждения на сигнал запроса коммуникационного порта 1, вход/выход с третьим состоянием	I/O/Z
AB13	CSTRB1_2#	Сигнал стробирования данных коммуникационного порта 1, вход/выход с третьим состоянием	I/O/Z

Продолжение таблицы 3.1

1	2	3	4
AC13	CRDY1_2#	Сигнал готовности данных коммуникационного порта 1, вход/выход с третьим состоянием	I/O/Z
Интерфейс коммуникационного порта 2			
AB12	C2D0_2	Шина данных коммуникационного порта 2, вход/выход с третьим состоянием, бит 0	I/O/Z
AF8	C2D1_2	Шина данных коммуникационного порта 2, вход/выход с третьим состоянием, бит 1	I/O/Z
AB14	C2D2_2	Шина данных коммуникационного порта 2, вход/выход с третьим состоянием, бит 2	I/O/Z
AE15	C2D3_2	Шина данных коммуникационного порта 2, вход/выход с третьим состоянием, бит 3	I/O/Z
AE8	C2D4_2	Шина данных коммуникационного порта 2, вход/выход с третьим состоянием, бит 4	I/O/Z
AE10	C2D5_2	Шина данных коммуникационного порта 2, вход/выход с третьим состоянием, бит 5	I/O/Z
AC11	C2D6_2	Шина данных коммуникационного порта 2, вход/выход с третьим состоянием, бит 6	I/O/Z
AD11	C2D7_2	Шина данных коммуникационного порта 2, вход/выход с третьим состоянием, бит 7	I/O/Z
AD9	CREQ2_2#	Сигнал запроса коммуникационного порта 2, вход/выход с третьим состоянием	I/O/Z
AD12	CACK2_2#	Сигнал подтверждения на сигнал запроса коммуникационного порта 2, вход/выход с третьим состоянием	I/O/Z
AE12	CSTRB2_2#	Сигнал стробирования данных коммуникационного порта 2, вход/выход с третьим состоянием	I/O/Z
AD13	CRDY2_2#	Сигнал готовности данных коммуникационного порта 2, вход/выход с третьим состоянием	I/O/Z
Интерфейс коммуникационного порта 3			
AE13	C3D0_2	Шина данных коммуникационного порта 3, вход/выход с третьим состоянием, бит 0	I/O/Z
AG11	C3D1_2	Шина данных коммуникационного порта 3, вход/выход с третьим состоянием, бит 1	I/O/Z
AE14	C3D2_2	Шина данных коммуникационного порта 3, вход/выход с третьим состоянием, бит 2	I/O/Z
Y14	C3D3_2	Шина данных коммуникационного порта 3, вход/выход с третьим состоянием, бит 3	I/O/Z
AG12	C3D4_2	Шина данных коммуникационного порта 3, вход/выход с третьим состоянием, бит 4	I/O/Z
AC14	C3D5_2	Шина данных коммуникационного порта 3, вход/выход с третьим состоянием, бит 5	I/O/Z
AG13	C3D6_2	Шина данных коммуникационного порта 3, вход/выход с третьим состоянием, бит 6	I/O/Z
AE16	C3D7_2	Шина данных коммуникационного порта 3, вход/выход с третьим состоянием, бит 7	I/O/Z
AD17	CREQ3_2#	Сигнал запроса коммуникационного порта 3, вход/выход с третьим состоянием	I/O/Z
AA15	CACK3_2#	Сигнал подтверждения на сигнал запроса коммуникационного порта 3, вход/выход с третьим состоянием	I/O/Z
AD18	CSTRB3_2#	Сигнал стробирования данных коммуникационного порта 3, вход/выход с третьим состоянием	I/O/Z
AE18	CRDY3_2#	Сигнал готовности данных коммуникационного порта 3, вход/выход с третьим состоянием	I/O/Z
Интерфейс коммуникационного порта 4			
AG17	C4D0_2	Шина данных коммуникационного порта 4, вход/выход с третьим состоянием, бит 0	I/O/Z
AB16	C4D1_2	Шина данных коммуникационного порта 4, вход/выход с третьим состоянием, бит 1	I/O/Z

Продолжение таблицы 3.1

1	2	3	4
AC18	C4D2_2	Шина данных коммуникационного порта 4, вход/выход с третьим состоянием, бит 2	I/O/Z
AD14	C4D3_2	Шина данных коммуникационного порта 4, вход/выход с третьим состоянием, бит 3	I/O/Z
AD15	C4D4_2	Шина данных коммуникационного порта 4, вход/выход с третьим состоянием, бит 4	I/O/Z
AD16	C4D5_2	Шина данных коммуникационного порта 4, вход/выход с третьим состоянием, бит 5	I/O/Z
AB15	C4D6_2	Шина данных коммуникационного порта 4, вход/выход с третьим состоянием, бит 6	I/O/Z
Y15	C4D7_2	Шина данных коммуникационного порта 4, вход/выход с третьим состоянием, бит 7	I/O/Z
AG16	CREQ4_2#	Сигнал запроса коммуникационного порта 4, вход/выход с третьим состоянием	I/O/Z
AE19	CACK4_2#	Сигнал подтверждения на сигнал запроса коммуникационного порта 4, вход/выход с третьим состоянием	I/O/Z
AC17	CSTRB4_2#	Сигнал стробирования данных коммуникационного порта 4, вход/выход с третьим состоянием	I/O/Z
AD19	CRDY4_2#	Сигнал готовности данных коммуникационного порта 4, вход/выход с третьим состоянием	I/O/Z
Интерфейс коммуникационного порта 5			
AF17	C5D0_2	Шина данных коммуникационного порта 5, вход/выход с третьим состоянием, бит 0	I/O/Z
AA16	C5D1_2	Шина данных коммуникационного порта 5, вход/выход с третьим состоянием, бит 1	I/O/Z
AF18	C5D2_2	Шина данных коммуникационного порта 5, вход/выход с третьим состоянием, бит 2	I/O/Z
AD20	C5D3_2	Шина данных коммуникационного порта 5, вход/выход с третьим состоянием, бит 3	I/O/Z
AF19	C5D4_2	Шина данных коммуникационного порта 5, вход/выход с третьим состоянием, бит 4	I/O/Z
AB17	C5D5_2	Шина данных коммуникационного порта 5, вход/выход с третьим состоянием, бит 5	I/O/Z
AC19	C5D6_2	Шина данных коммуникационного порта 5, вход/выход с третьим состоянием, бит 6	I/O/Z
AG20	C5D7_2	Шина данных коммуникационного порта 5, вход/выход с третьим состоянием, бит 7	I/O/Z
AF20	CREQ5_2#	Сигнал запроса коммуникационного порта 5, вход/выход с третьим состоянием	I/O/Z
AB18	CACK5_2#	Сигнал подтверждения на сигнал запроса коммуникационного порта 5, вход/выход с третьим состоянием	I/O/Z
AG21	CSTRB5_2#	Сигнал стробирования данных коммуникационного порта 5, вход/выход с третьим состоянием	I/O/Z
Y17	CRDY5_2#	Сигнал готовности данных коммуникационного порта 5, вход/выход с третьим состоянием	I/O/Z
Прерывания, флаги ввода/вывода, вектор сброса, таймер			
AC22	IIOF0_2#	Прерывание и флаг ввода/вывода 0, вход/выход с третьим состоянием	I/O/Z
AE24	IIOF1_2#	Прерывание и флаг ввода/вывода 1, вход/выход с третьим состоянием	I/O/Z
Y19	IIOF2_2#	Прерывание и флаг ввода/вывода 2, вход/выход с третьим состоянием	I/O/Z
AB21	IIOF3_2#	Прерывание и флаг ввода/вывода 3, вход/выход с третьим состоянием	I/O/Z
AD21	IACK_2#	Подтверждение прерывания, выход с третьим состоянием	O/Z
AB22	NMI_2#	Немаскируемое прерывание. Вывод “подтянут” к 1	I

Продолжение таблицы 3.1

1	2	3	4
AF22	RESETLOC0_2	Вывод 0 вектора сброса	I
AB19	RESETLOCl_2	Вывод 1 вектора сброса	I
AD22	ROMEN_2	Разрешение внутрикристального ПЗУ (0 – запрещено, 1 – разрешено). Вывод “подтянут” к 0	I
AE20	TCLK0_2	Вывод таймера 0, вход/выход с третьим состоянием	I/O/Z
AG22	TCLK1_2	Вывод таймера 1, вход/выход с третьим состоянием	I/O/Z
Тактирующие сигналы			
U4	X2/CLKIN_COMM	Тактирующий сигнал с частотой от пяти до 10 МГц.	I
R7	CntrlPLL	Управление PLL. Вывод подтянут к 0 CntrlPLL – низкий уровень – PLL включен CntrlPLL – высокий уровень – PLL отключен.	I
H22	H1_1	H1 выходной тактирующий сигнал процессора 1	O
D26	H3_1	H3 выходной тактирующий сигнал процессора 1	O
W20	H1_2	H1 выходной тактирующий сигнал процессора 2	O
AC23	H3_2	H3 выходной тактирующий сигнал процессора 2	O
Сброс			
T3	RESET_COMM#	Общий сигнал сброса. Вывод “подтянут” к 1	I
Эмулятор			
R3	TRST_COMM#	Сигнал сброса тестового порта IEEE 1149.1. Вывод “подтянут” к 0.	I
P8	TMS_COMM	Сигнал выбора режима работы тестового порта IEEE 1149.1	I
N5	TCK_COMM	Тактовый сигнал тестового порта IEEE 1149.1	I
N1	EMU0_COMM	Вывод 0 эмулятора, вход/выход с третьим состоянием	I/O/Z
R4	EMU1_COMM	Вывод 1 эмулятора, вход/выход с третьим состоянием	I/O/Z
L1	TDI_1	Входные данные тестового порта IEEE 1149.1	I
N6	TDO_2	Выходные данные тестового порта IEEE 1149.1, выход с третьим состоянием	O/Z
Выводы устройства MilStd 1553			
AF27	RXIN_A#	Входной сигнал канала А, инверсный	I
AF26	RXIN_A	Входной сигнал канала А, неинверсный	I
V20	TXOUT_A	Выходной сигнал канала А, неинверсный	O
W21	TXOUT_A#	Выходной сигнал канала А, инверсный	O
AD26	TXINA	Сигнал блокировки передатчика канала А	O
AC25	RXENA	Сигнал разрешения приёма информации канала А	O
Y21	CLKMilSTD	Сигнал тактирования с частотой 24 МГц	I
AB23	TXOUT_B	Выходной сигнал канала В, неинверсный	O
Y22	TXOUT_B#	Выходной сигнал канала В, инверсный	O
AC24	TXINB	Сигнал блокировки передатчика канала В	O
AE25	RXIN_B	Входной сигнал канала В, неинверсный	I
AA22	RXIN_B#	Входной сигнал канала В, инверсный	I
U20	RXENB	Сигнал разрешения приёма информации канала В	O
Выводы устройства Ethernet 10/100			
U22	TxD0	Передаваемые данные, бит 0	O
Y24	TxD1	Передаваемые данные, бит 1	O
W23	TxD2	Передаваемые данные, бит 2	O
Y25	TxD3	Передаваемые данные, бит 3	O
R20	Tx_ERR	Ошибка передачи	O
AB27	Tx_EN	Разрешение передачи	O
R24	Tx_CLK	Сигнал тактирования передачи	I
T25	COL	Сигнал коллизии	I
U23	CRS	Сигнал обнаружения несущей	I

Продолжение таблицы 3.1

1	2	3	4
V22	RxD0	Принимаемые данные, бит 0	I
AA23	RxD1	Принимаемые данные, бит 1	I
AC27	RxD2	Принимаемые данные, бит 2	I
U21	RxD3	Принимаемые данные, бит 3	I
AA24	Rx_DV	Принимаемые данные правильные	I
AB26	Rx_ERR	Ошибка принимаемых данных	I
R23	Rx_CLK	Сигнал тактирования приёма	I
Y26	MDC	Управление сигналом тактирования данных	O
T24	TSE	Разрешение последовательной передачи	O
V26	MDIO	Управление входом/выходом данных	I/O
Выводы устройства UART			
L24	CD	Обнаружение несущей	I
R25	TD	Передаваемые данные к внешнему устройству	O
P23	RD	Принимаемые данные из внешнего устройства	I
R27	DTR	Готовность терминала	O
M24	DSR	Готовность к передаче	I
N25	RTS	Запрос передачи	O
N24	CTS	Готовность внешнего устройства к приёму	I
N22	RI	Индикатор вызова	I
R22	Clk PLL UART	Сигнал тактирования PLL UART	I
N23	BAUD	Выходная тактовая частота приёма/передачи	O
Выводы устройства USB 2.0			
K23	SuspendM	Установка трансивера в режим ожидания	O
J24	Xcvr_Select	1: Выбор режима трансивера Full speed 0: Выбор режима трансивера High speed	O
L26	Term_Select	1: Разрешение окончания режима Full speed 0: Разрешение окончания режима High speed	O
K26	LineState_1	Сигнал 1 состояния линии	I
M21	LineState_0	Сигнал 0 состояния линии	I
J23	OpMode_1	Сигнал 1 выбора режима работы	O
H24	OpMode_0	Сигнал 0 выбора режима работы	O
H25	DataOut7	Выходные данные, бит 7	O
L20	DataOut6	Выходные данные, бит 6	O
G27	DataOut5	Выходные данные, бит 5	O
K22	DataOut4	Выходные данные, бит 4	O
H26	DataOut3	Выходные данные, бит 3	O
H27	DataOut2	Выходные данные, бит 2	O
L21	DataOut1	Выходные данные, бит 1	O
L22	DataOut0	Выходные данные, бит 0	O
N20	TXValid	Передаваемые данные правильные	O
K25	TXReady	Готовность передачи	I
E25	DataIn7	Входные данные, бит 7	I
K20	DataIn6	Входные данные, бит 6	I
J21	DataIn5	Входные данные, бит 5	I
E26	DataIn4	Входные данные, бит 4	I
G23	DataIn3	Входные данные, бит 3	I
F24	DataIn2	Входные данные, бит 2	I
G24	DataIn1	Входные данные, бит 1	I
F27	DataIn0	Входные данные, бит 0	I
M27	RXValid	Принимаемые данные правильные	I
J25	RXActive	Готовность приёма данных	I
L23	RXError	Ошибка приёма	I
N27	CLK_Phy	Тактирующий сигнал от трансивера	I
K24	Reset_phy	Сигнал сброса трансивера	O

Продолжение таблицы 3.1

1	2	3	4
Питание			
N7	∩VSS_A_CPUPLL	Аналоговый вывод земли PLL ядер процессоров	–
M3	∩VCC_A_ CPUPLL	Аналоговый вывод питания PLL ядер процессоров	–
P25	∩VSS_A_UARTPLL	Аналоговый вывод земли PLL UART	–
T26	∩VCC_A_ UARTPLL	Аналоговый вывод питания PLL UART	–
A4	#VSS_DR	Земляной вывод буферов ввода-вывода	–
A8	#VSS_DR	Земляной вывод буферов ввода-вывода	–
A9	#VSS_DR	Земляной вывод буферов ввода-вывода	–
A14	#VSS_DR	Земляной вывод буферов ввода-вывода	–
A19	#VSS_DR	Земляной вывод буферов ввода-вывода	–
B13	#VSS_DR	Земляной вывод буферов ввода-вывода	–
C6	#VSS_DR	Земляной вывод буферов ввода-вывода	–
C17	#VSS_DR	Земляной вывод буферов ввода-вывода	–
C22	#VSS_DR	Земляной вывод буферов ввода-вывода	–
D1	#VSS_DR	Земляной вывод буферов ввода-вывода	–
E8	#VSS_DR	Земляной вывод буферов ввода-вывода	–
E12	#VSS_DR	Земляной вывод буферов ввода-вывода	–
E16	#VSS_DR	Земляной вывод буферов ввода-вывода	–
E20	#VSS_DR	Земляной вывод буферов ввода-вывода	–
F1	#VSS_DR	Земляной вывод буферов ввода-вывода	–
F25	#VSS_DR	Земляной вывод буферов ввода-вывода	–
G10	#VSS_DR	Земляной вывод буферов ввода-вывода	–
G18	#VSS_DR	Земляной вывод буферов ввода-вывода	–
G20	#VSS_DR	Земляной вывод буферов ввода-вывода	–
G22	#VSS_DR	Земляной вывод буферов ввода-вывода	–
G25	#VSS_DR	Земляной вывод буферов ввода-вывода	–
H8	#VSS_DR	Земляной вывод буферов ввода-вывода	–
H14	#VSS_DR	Земляной вывод буферов ввода-вывода	–
H16	#VSS_DR	Земляной вывод буферов ввода-вывода	–
J1	#VSS_DR	Земляной вывод буферов ввода-вывода	–
J27	#VSS_DR	Земляной вывод буферов ввода-вывода	–
K7	#VSS_DR	Земляной вывод буферов ввода-вывода	–
L3	#VSS_DR	Земляной вывод буферов ввода-вывода	–
M5	#VSS_DR	Земляной вывод буферов ввода-вывода	–
M23	#VSS_DR	Земляной вывод буферов ввода-вывода	–
M25	#VSS_DR	Земляной вывод буферов ввода-вывода	–
N26	#VSS_DR	Земляной вывод буферов ввода-вывода	–
P20	#VSS_DR	Земляной вывод буферов ввода-вывода	–
P21	#VSS_DR	Земляной вывод буферов ввода-вывода	–
P27	#VSS_DR	Земляной вывод буферов ввода-вывода	–
R2	#VSS_DR	Земляной вывод буферов ввода-вывода	–
T1	#VSS_DR	Земляной вывод буферов ввода-вывода	–
T22	#VSS_DR	Земляной вывод буферов ввода-вывода	–
T23	#VSS_DR	Земляной вывод буферов ввода-вывода	–
U25	#VSS_DR	Земляной вывод буферов ввода-вывода	–
V27	#VSS_DR	Земляной вывод буферов ввода-вывода	–
W26	#VSS_DR	Земляной вывод буферов ввода-вывода	–
Y5	#VSS_DR	Земляной вывод буферов ввода-вывода	–
Y23	#VSS_DR	Земляной вывод буферов ввода-вывода	–
AA10	#VSS_DR	Земляной вывод буферов ввода-вывода	–
AA14	#VSS_DR	Земляной вывод буферов ввода-вывода	–
AA18	#VSS_DR	Земляной вывод буферов ввода-вывода	–
AA25	#VSS_DR	Земляной вывод буферов ввода-вывода	–
AB24	#VSS_DR	Земляной вывод буферов ввода-вывода	–
AB25	#VSS_DR	Земляной вывод буферов ввода-вывода	–
AC8	#VSS_DR	Земляной вывод буферов ввода-вывода	–

Продолжение таблицы 3.1

1	2	3	4
AC20	#VSS_DR	Земляной вывод буферов ввода-вывода	–
AD1	#VSS_DR	Земляной вывод буферов ввода-вывода	–
AE6	#VSS_DR	Земляной вывод буферов ввода-вывода	–
AE22	#VSS_DR	Земляной вывод буферов ввода-вывода	–
AF9	#VSS_DR	Земляной вывод буферов ввода-вывода	–
AF10	#VSS_DR	Земляной вывод буферов ввода-вывода	–
AF21	#VSS_DR	Земляной вывод буферов ввода-вывода	–
AG9	#VSS_DR	Земляной вывод буферов ввода-вывода	–
AG14	#VSS_DR	Земляной вывод буферов ввода-вывода	–
AG19	#VSS_DR	Земляной вывод буферов ввода-вывода	–
AG24	#VSS_DR	Земляной вывод буферов ввода-вывода	–
A5	#VSS_CL	Земляной вывод ядра	–
J7	#VSS_CL	Земляной вывод ядра	–
A16	#VSS_CL	Земляной вывод ядра	–
L6	#VSS_CL	Земляной вывод ядра	–
A24	#VSS_CL	Земляной вывод ядра	–
B19	#VSS_CL	Земляной вывод ядра	–
N8	#VSS_CL	Земляной вывод ядра	–
B20	#VSS_CL	Земляной вывод ядра	–
C11	#VSS_CL	Земляной вывод ядра	–
P5	#VSS_CL	Земляной вывод ядра	–
D27	#VSS_CL	Земляной вывод ядра	–
T8	#VSS_CL	Земляной вывод ядра	–
W6	#VSS_CL	Земляной вывод ядра	–
F3	#VSS_CL	Земляной вывод ядра	–
G14	#VSS_CL	Земляной вывод ядра	–
Y16	#VSS_CL	Земляной вывод ядра	–
AA13	#VSS_CL	Земляной вывод ядра	–
H5	#VSS_CL	Земляной вывод ядра	–
H12	#VSS_CL	Земляной вывод ядра	–
AB9	#VSS_CL	Земляной вывод ядра	–
H23	#VSS_CL	Земляной вывод ядра	–
AB11	#VSS_CL	Земляной вывод ядра	–
J2	#VSS_CL	Земляной вывод ядра	–
K1	#VSS_CL	Земляной вывод ядра	–
AC15	#VSS_CL	Земляной вывод ядра	–
K21	#VSS_CL	Земляной вывод ядра	–
L25	#VSS_CL	Земляной вывод ядра	–
P1	#VSS_CL	Земляной вывод ядра	–
P7	#VSS_CL	Земляной вывод ядра	–
V25	#VSS_CL	Земляной вывод ядра	–
T5	#VSS_CL	Земляной вывод ядра	–
U3	#VSS_CL	Земляной вывод ядра	–
V7	#VSS_CL	Земляной вывод ядра	–
V21	#VSS_CL	Земляной вывод ядра	–
W1	#VSS_CL	Земляной вывод ядра	–
W8	#VSS_CL	Земляной вывод ядра	–
W27	#VSS_CL	Земляной вывод ядра	–
AA8	#VSS_CL	Земляной вывод ядра	–
AA17	#VSS_CL	Земляной вывод ядра	–
AB3	#VSS_CL	Земляной вывод ядра	–
AC12	#VSS_CL	Земляной вывод ядра	–
AC16	#VSS_CL	Земляной вывод ядра	–
AD27	#VSS_CL	Земляной вывод ядра	–
AE7	#VSS_CL	Земляной вывод ядра	–
AE11	#VSS_CL	Земляной вывод ядра	–

Продолжение таблицы 3.1

1	2	3	4
AE17	#VSS_CL	Земляной вывод ядра	–
AE27	#VSS_CL	Земляной вывод ядра	–
AF15	#VSS_CL	Земляной вывод ядра	–
AG4	#VSS_CL	Земляной вывод ядра	–
A10	#VCC_DR	Вывод питания буферов ввода-вывода	–
A17	#VCC_DR	Вывод питания буферов ввода-вывода	–
B5	#VCC_DR	Вывод питания буферов ввода-вывода	–
B8	#VCC_DR	Вывод питания буферов ввода-вывода	–
B11	#VCC_DR	Вывод питания буферов ввода-вывода	–
B14	#VCC_DR	Вывод питания буферов ввода-вывода	–
B15	#VCC_DR	Вывод питания буферов ввода-вывода	–
B17	#VCC_DR	Вывод питания буферов ввода-вывода	–
B21	#VCC_DR	Вывод питания буферов ввода-вывода	–
B22	#VCC_DR	Вывод питания буферов ввода-вывода	–
B25	#VCC_DR	Вывод питания буферов ввода-вывода	–
C2	#VCC_DR	Вывод питания буферов ввода-вывода	–
C19	#VCC_DR	Вывод питания буферов ввода-вывода	–
C20	#VCC_DR	Вывод питания буферов ввода-вывода	–
C23	#VCC_DR	Вывод питания буферов ввода-вывода	–
D22	#VCC_DR	Вывод питания буферов ввода-вывода	–
E2	#VCC_DR	Вывод питания буферов ввода-вывода	–
E27	#VCC_DR	Вывод питания буферов ввода-вывода	–
G1	#VCC_DR	Вывод питания буферов ввода-вывода	–
G2	#VCC_DR	Вывод питания буферов ввода-вывода	–
G26	#VCC_DR	Вывод питания буферов ввода-вывода	–
J3	#VCC_DR	Вывод питания буферов ввода-вывода	–
J26	#VCC_DR	Вывод питания буферов ввода-вывода	–
K27	#VCC_DR	Вывод питания буферов ввода-вывода	–
L2	#VCC_DR	Вывод питания буферов ввода-вывода	–
L27	#VCC_DR	Вывод питания буферов ввода-вывода	–
M2	#VCC_DR	Вывод питания буферов ввода-вывода	–
M20	#VCC_DR	Вывод питания буферов ввода-вывода	–
N2	#VCC_DR	Вывод питания буферов ввода-вывода	–
N21	#VCC_DR	Вывод питания буферов ввода-вывода	–
P2	#VCC_DR	Вывод питания буферов ввода-вывода	–
P22	#VCC_DR	Вывод питания буферов ввода-вывода	–
P24	#VCC_DR	Вывод питания буферов ввода-вывода	–
P26	#VCC_DR	Вывод питания буферов ввода-вывода	–
R26	#VCC_DR	Вывод питания буферов ввода-вывода	–
T2	#VCC_DR	Вывод питания буферов ввода-вывода	–
T20	#VCC_DR	Вывод питания буферов ввода-вывода	–
T21	#VCC_DR	Вывод питания буферов ввода-вывода	–
U24	#VCC_DR	Вывод питания буферов ввода-вывода	–
U26	#VCC_DR	Вывод питания буферов ввода-вывода	–
U27	#VCC_DR	Вывод питания буферов ввода-вывода	–
V1	#VCC_DR	Вывод питания буферов ввода-вывода	–
V2	#VCC_DR	Вывод питания буферов ввода-вывода	–
V24	#VCC_DR	Вывод питания буферов ввода-вывода	–
W22	#VCC_DR	Вывод питания буферов ввода-вывода	–
W24	#VCC_DR	Вывод питания буферов ввода-вывода	–
W25	#VCC_DR	Вывод питания буферов ввода-вывода	–
AA1	#VCC_DR	Вывод питания буферов ввода-вывода	–
AA26	#VCC_DR	Вывод питания буферов ввода-вывода	–
AA27	#VCC_DR	Вывод питания буферов ввода-вывода	–
AC2	#VCC_DR	Вывод питания буферов ввода-вывода	–
AC26	#VCC_DR	Вывод питания буферов ввода-вывода	–

Окончание таблицы 3.1

1	2	3	4
AE9	#VCC_DR	Вывод питания буферов ввода-вывода	–
AE26	#VCC_DR	Вывод питания буферов ввода-вывода	–
AF3	#VCC_DR	Вывод питания буферов ввода-вывода	–
AF5	#VCC_DR	Вывод питания буферов ввода-вывода	–
AF7	#VCC_DR	Вывод питания буферов ввода-вывода	–
AF11	#VCC_DR	Вывод питания буферов ввода-вывода	–
AF12	#VCC_DR	Вывод питания буферов ввода-вывода	–
AF13	#VCC_DR	Вывод питания буферов ввода-вывода	–
AF14	#VCC_DR	Вывод питания буферов ввода-вывода	–
AF16	#VCC_DR	Вывод питания буферов ввода-вывода	–
AG7	#VCC_DR	Вывод питания буферов ввода-вывода	–
AG18	#VCC_DR	Вывод питания буферов ввода-вывода	–
AG23	#VCC_DR	Вывод питания буферов ввода-вывода	–
A15	#VCC_CL	Вывод питания ядра	–
A18	#VCC_CL	Вывод питания ядра	–
A22	#VCC_CL	Вывод питания ядра	–
A25	#VCC_CL	Вывод питания ядра	–
B6	#VCC_CL	Вывод питания ядра	–
B9	#VCC_CL	Вывод питания ядра	–
C10	#VCC_CL	Вывод питания ядра	–
E3	#VCC_CL	Вывод питания ядра	–
F26	#VCC_CL	Вывод питания ядра	–
G3	#VCC_CL	Вывод питания ядра	–
H11	#VCC_CL	Вывод питания ядра	–
H15	#VCC_CL	Вывод питания ядра	–
H17	#VCC_CL	Вывод питания ядра	–
J22	#VCC_CL	Вывод питания ядра	–
K2	#VCC_CL	Вывод питания ядра	–
M1	#VCC_CL	Вывод питания ядра	–
M22	#VCC_CL	Вывод питания ядра	–
M26	#VCC_CL	Вывод питания ядра	–
R1	#VCC_CL	Вывод питания ядра	–
R21	#VCC_CL	Вывод питания ядра	–
T27	#VCC_CL	Вывод питания ядра	–
U1	#VCC_CL	Вывод питания ядра	–
V23	#VCC_CL	Вывод питания ядра	–
Y2	#VCC_CL	Вывод питания ядра	–
Y18	#VCC_CL	Вывод питания ядра	–
Y20	#VCC_CL	Вывод питания ядра	–
Y27	#VCC_CL	Вывод питания ядра	–
AB2	#VCC_CL	Вывод питания ядра	–
AD25	#VCC_CL	Вывод питания ядра	–
AE4	#VCC_CL	Вывод питания ядра	–
AE21	#VCC_CL	Вывод питания ядра	–
AG6	#VCC_CL	Вывод питания ядра	–
AG10	#VCC_CL	Вывод питания ядра	–
AG15	#VCC_CL	Вывод питания ядра	–

Примечание – Принятые условные обозначения:
- I – вход,
- O – выход,
- I/O – вход/выход,
- I/O/Z – вход/выход с третьим состоянием,
- O/Z – выход с третьим состоянием.

4 Описание архитектуры ИС 1867ВЦ8Ф1

Архитектура ИС 1867ВЦ8Ф1 основана на архитектуре ядра 32-разрядного ПЦОС. Ядро ПЦОС функционально и архитектурно совместимо с ИС 1867ВЦ3Ф.

ИС 1867ВЦ8Ф1 содержит (см. подраздел 1.2.1) два ядра: ПЦОС 1 и ПЦОС 2, а также внешние, по отношению к ядрам, периферийные устройства (UART, USB 2.0, MIL-STD-1553 и Ethernet 10/100). Ядра ПЦОС обеспечивают доступ к области адресов глобальной памяти, а также к области адресов внутриядерных периферийных устройств, таких как таймеры, устройства ПДП и т.п. и области адресов внешних периферийных устройств.

Карты памяти ПЦОС 1 и ПЦОС 2 идентичны. Область адресов локальной памяти отведена для доступа к регистрам и памяти внешних периферийных устройств.

Карты памяти ИС 1867ВЦ8Ф1 приведены в таблицах 4.1 – 4.4. Они включают области адресов ядер ПЦОС 1, ПЦОС 2 (карта памяти периферийных устройств ядра ПЦОС и области памяти приведена в подразделе 5.4), области адресов внешних периферийных устройств, см. разделы 7, 8, 9, 10 и 11 данного Руководства.

Периферийные устройства ядер ПЦОС картируются с адреса 0010 0000h по адрес 0010 00FFh. Внешние периферийные устройства картируются с адреса 0030 0000h по адрес 0071 3FFFh. Карта памяти периферийных устройств ядер ПЦОС приведена в таблице 4.3.

Все внешние, по отношению к ядрам ПЦОС, периферийные устройства ИС 1867ВЦ8Ф1 подключаются к локальной шине ПЦОС через коммутатор и картируются в область памяти локальных шин ПЦОС 1 и ПЦОС 2 с адреса 0030 0000h до адреса 0071 3FFFh, см. таблицу 4.4. Подключение внешних периферийных устройств к одному из процессоров осуществляется через регистр коммутации RC, который является разделяемым ресурсом между ПЦОС 1 и ПЦОС 2. Описание коммутатора приведено в разделе 6.

Примечание – При программировании внешних периферийных устройств необходимо установить режим ожидания сигнала готовности на локальной шине – биты 5, 4 (поле SWW) регистра контроля интерфейса локальной памяти (адрес 0010 0004h) должны быть равны 00_2. Стробирование записи/чтения в/из регистров/памяти внешних периферийных устройств осуществляется по сигналу LSTRB0 для ПЦОС 1 и ПЦОС 2, поэтому биты 28 – 24 (поле STRB ACTIVE) регистра контроля интерфейса локальной памяти (адрес 0010 0004h) должны быть равны 11110_2 (значение, устанавливаемое после сброса).

ПЦОС соединены друг с другом через коммуникационные порты: COMM0 со стороны ПЦОС 2 и COMM3 со стороны ПЦОС 1. Важно отметить, что после сброса порт COMM0 ПЦОС 2 конфигурируется как выходной порт, а порт COMM3 ПЦОС 1 конфигурируется как входной порт.

В таблицах 4.1 и 4.2 приведены карты памяти процессорных ядер ИС 1867ВЦ8Ф1 ПЦОС 1 и ПЦОС 2, соответственно.

Таблица 4.1 – Карта памяти ПЦОС 1

Диапазон адресов	Наименование области памяти ПЦОС 1	
	ROMEN_1 = 0	ROMEN_1 = 1
0000 0000h – 0000 0FFFh	Область адресов локальной шины Резерв	Загрузчик ядра ПЦОС из ПЗУ
0000 1000h – 000F FFFFh	Резерв	
0010 0000h – 0010 00FFh	Периферийные устройства ядра	
0010 0100h – 001F FFFFh	Резерв	
0020 0000h – 002F F7FFh	Резерв	
002F F800h – 002F FBFFh	ОЗУ 1K × 32 блок 1	
002F FC00h – 002F FFFFh	ОЗУ 1K × 32 блок 2	
0030 0000h – 0071 3FFFh	Область адресов внешних периферийных устройств	
0071 4000h – 7FFF FFFFh	Резерв	
8000 0000h – FFFF FFFFh	Область адресов глобальной шины	

Таблица 4.2 – Карта памяти ПЦОС 2

Диапазон адресов	Наименование области памяти ПЦОС 2	
	ROMEN_2= 0	ROMEN_2 = 1
0000 0000h – 0000 0FFFh	Область адресов локальной шины Резерв	Загрузчик ядра ПЦОС из ПЗУ
0000 1000h – 000F FFFFh	Резерв	
0010 0000h – 0010 00FFh	Периферийные устройства ядра	
0010 0100h – 001F FFFFh	Резерв	
0020 0000h – 002F F7FFh	Резерв	
002F F800h – 002F FBFFh	ОЗУ 1К × 32 блок 1	
002F FC00h – 002F FFFFh	ОЗУ 1К × 32 блок 2	
0030 0000h – 0071 3FFFh	Область адресов внешних периферийных устройств	
0071 4000h – 7FFF FFFFh	Резерв	
8000 0000h – FFFF FFFFh	Область адресов глобальной шины	

4.1 Карта памяти периферийных устройств ядер ПЦОС

В таблице 4.3 приведена карта памяти периферийных устройств ядер ПЦОС 1 и ПЦОС 2.

Таблица 4.3 – Карта памяти ядер ПЦОС 1 и ПЦОС 2

Адрес	Регистр	Описание
1	2	3
0010 0000h 0010 000Fh	Регистры управления локальной и глобальной шинами (16 слов)	5.4.2.1, рисунок 5.14
0010 0010h 0010 001Fh	Регистры управления работой ПЦОС (16 слов)	5.4.2.2
0010 0020h 0010 002Fh	Регистры таймера 0 (16 слов)	5.4.2.3, рисунок 5.15
0010 0030h 0010 003Fh	Регистры таймера 1 (16 слов)	
0010 0040h 0010 004Fh	Регистры коммуникационного порта 0 (16 слов)	5.4.2.4, рисунок 5.16
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)	

Окончание таблицы 4.3

1	2	3
0010 00A0h 0010 00AFh	Регистры 0 канала сопроцессора ПДП (16 слов)	5.4.2.5, рисунок 5.17
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 Карта памяти внешних периферийных устройств

В таблице 4.4 приведена карта памяти внешних, по отношению к ядрам ПЦОС, периферийных устройств ИС 1867ВЦ8Ф1. Функциональное назначение и описание регистров, их битов и полей для конкретных периферийных устройств приведено в разделах 7, 8, 9, 10 и 11 данного Руководства.

Таблица 4.4 – Карта памяти внешних периферийных устройств ИС 1867ВЦ8Ф1

Диапазон адресов	Устройство	Описание
0030 0000h – 0030 0001h	Коммутатор	Раздел 6
0040 0000h – 0040 0012h	UART 16550	Раздел 10
0040 01FFh – 0040 01FFh	PLL UART	Раздел 11
0050 0000h – 0050 000Dh 0050 1000h – 0050 1BFFh	Регистры MIL-STD-1553 ОЗУ MIL-STD-1553	Раздел 9
0060 0000h – 0060 013Ch 0061 0000h – 0061 3FFFh	Регистры USB 2.0 ОЗУ USB 2.0	Раздел 8
0070 0000h – 0070 0067h 00710 000h – 0071 3FFFh	Регистры Ethernet 10/100 ОЗУ Ethernet 10/100	Раздел 7

5 Описание ядра процессора цифровой обработки сигналов

5.1 Общее описание процессорного ядра ИС 1867ВЦ8Ф1

5.1.1 Введение

Ядро процессора 1867ВЦ8Ф1 – это 32-разрядный цифровой сигнальный процессор с плавающей запятой, оптимизированный для параллельных процессов. Ядро процессора 1867ВЦ8Ф1 совмещает высокую производительность ЦПУ и ПДП контроллера с шестью коммуникационными портами для работы с мультипроцессорной системой и периферийными устройствами. Ядро процессора 1867ВЦ8Ф1 имеет встроенный модуль анализа, который поддерживает точки останова при разработке и отладке параллельных процессов. Коды команд ядра процессора 1867ВЦ8Ф1 совместимы с кодами команд ПЦОС 1867ВЦ3Ф.

5.1.2 Общий обзор возможностей ЦПУ ИС 1867ВЦ8Ф1

Возможности ЦПУ – производительность 100 миллионов операций в секунду при работе с фиксированной запятой и 200 миллионов операций в секунду при работе с плавающей запятой с максимальной пропускной способностью ввода-вывода 500 Мбайт/с. Ядро процессора 1867ВЦ8Ф1 имеет 2К слов встроенного ОЗУ, 128 слов программного кэша и загрузчик программы. Две внешние шины обеспечивают адресацию к 4Г словам единого адресного пространства.

Ключевые возможности ядра процессора 1867ВЦ8Ф1:

- 50 миллионов операций в секунду при работе с фиксированной запятой и 100 миллионов операций в секунду при работе с плавающей запятой
- максимальная пропускная способность ввода/вывода 500 Мбайт/с;
- IEEE преобразование плавающей запятой для удобства использования;
- однотактный цикл выполнения команд при операциях с байтами, полусловами и словами;
- регистры основного назначения;
- поддержка деления и извлечения квадратного корня;
- встроенная память, включающая 2К слов статического ОЗУ, 128 слов программного кэша и загрузчик программы;
- две внешние шины, обеспечивающие адресацию к 4Г словам единого адресного пространства;
- два 32-битных таймера;
- 6 и 12 каналов ПДП;
- 6 коммуникационных портов для мультипроцессорных систем;
- IDLE режим пониженного энергопотребления.

5.2 Архитектурный обзор

Высокая производительность ИС 1867ВЦ8Ф1 достигнута за счёт большой точности и широкого динамического диапазона вычислительного блока с плавающей запятой, двухпортового СОЗУ, большой внутрикристальной памяти (до 448К слов – 224К слов на глобальной и 224К слов на локальной шинах), высокой степени параллелизма работы коммуникационных портов, сопроцессора ПДП и ЦПУ.

В данном подразделе даётся описание архитектуры ИС 1867ВЦ8Ф1.

На рисунке 5.1 приведена структурная блок-схема процессорного ядра ИС.

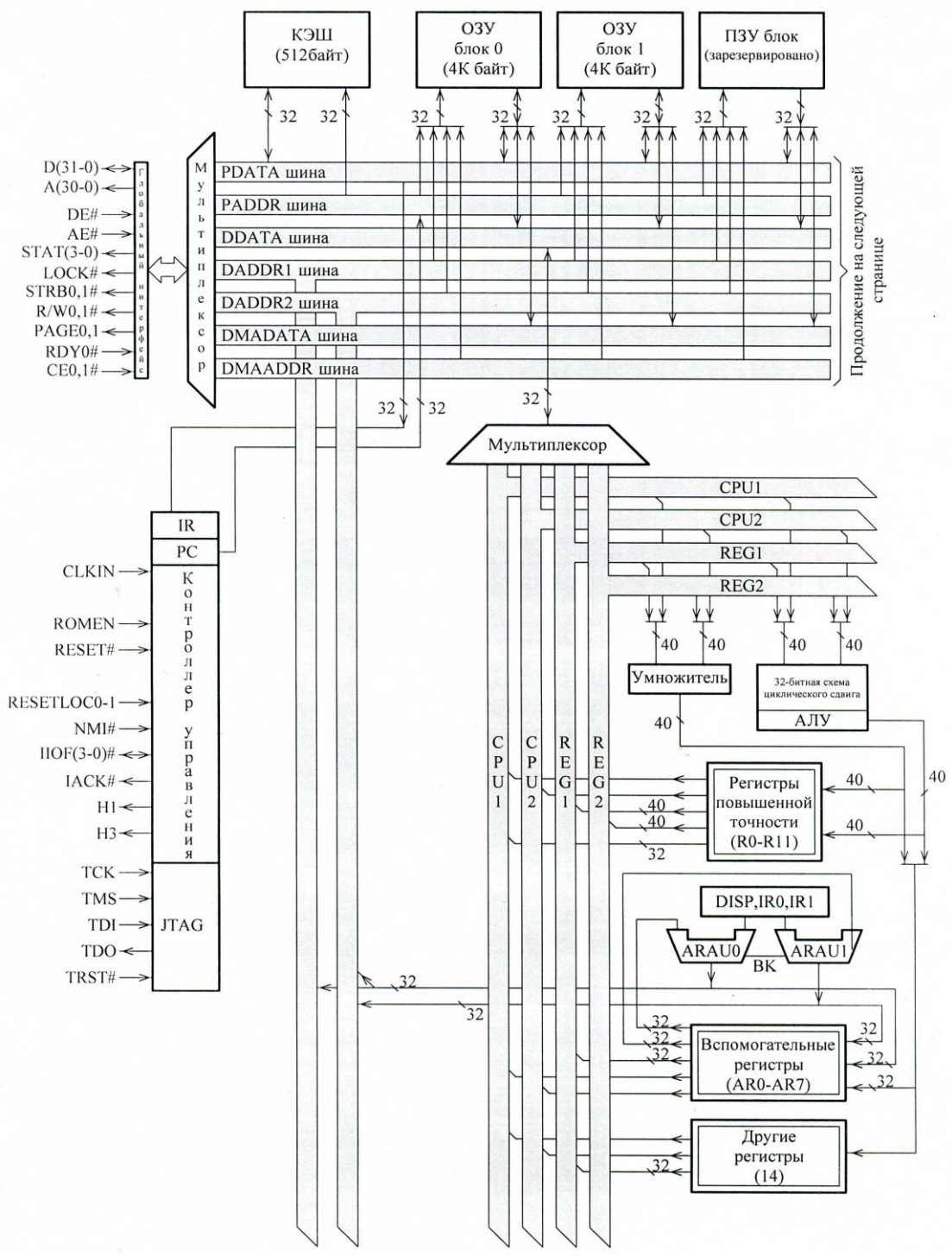


Рисунок 5.1, лист 1 – Блок-схема процессорного ядра 1867ВЦ8Ф1

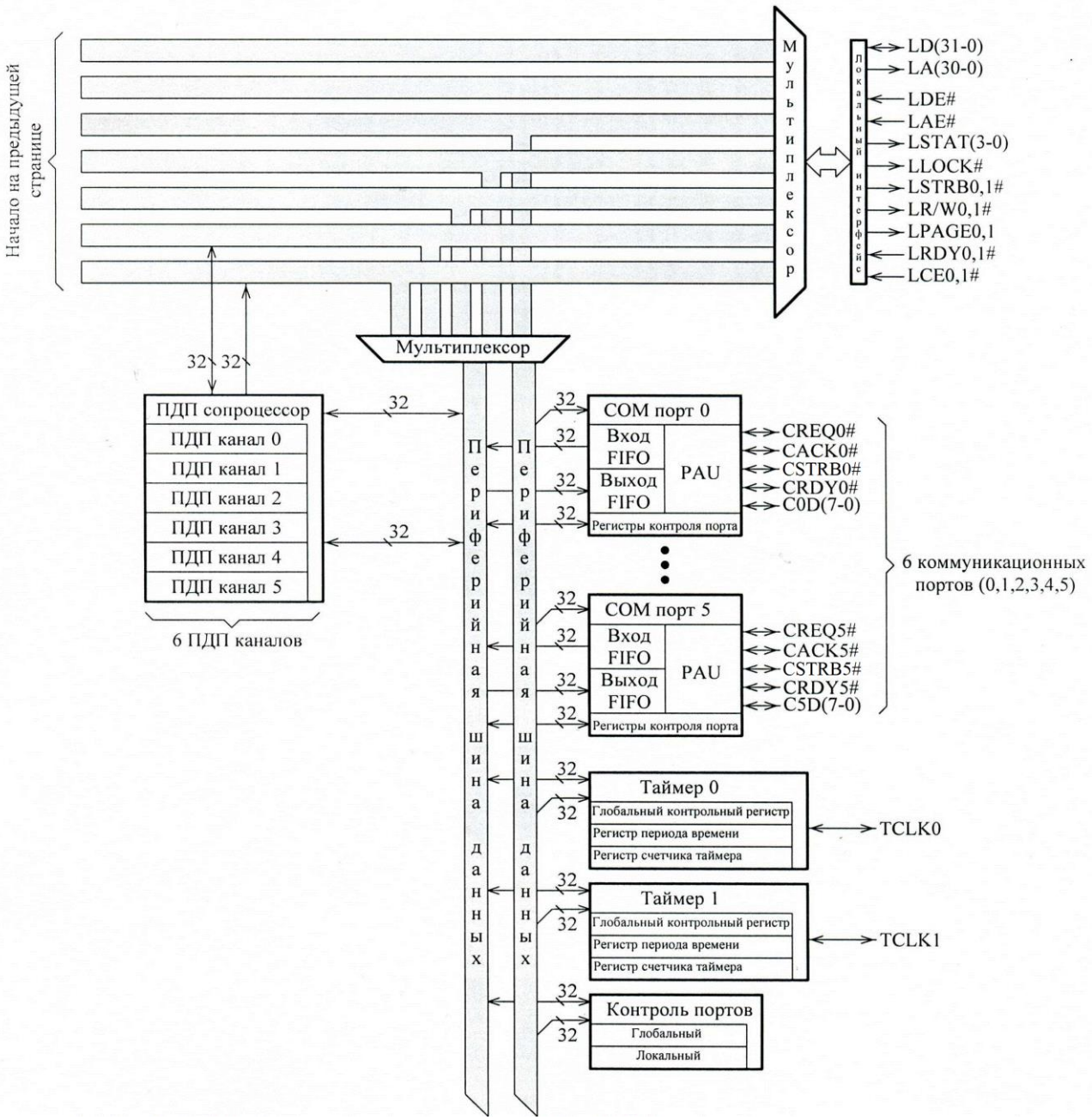


Рисунок 5.1, лист 2

5.2.1 Центральное процессорное устройство ЦПУ

Архитектура ЦПУ процессорных ядер микросхемы 1867ВЦ8Ф1 основана на работе с регистрами. ЦПУ можно разделить на несколько функционально законченных блоков:

- аппаратный умножитель;
- арифметико-логическое устройство АЛУ;
- наборы внутренних шин CPU1/CPU2 и REG1/REG2;
- арифметические устройства вспомогательных регистров ARAU*n*;
- наборы регистров ЦПУ, представленные в таблице 5.1.

На рисунке 5.2 приведены основные функциональные блоки ЦПУ.

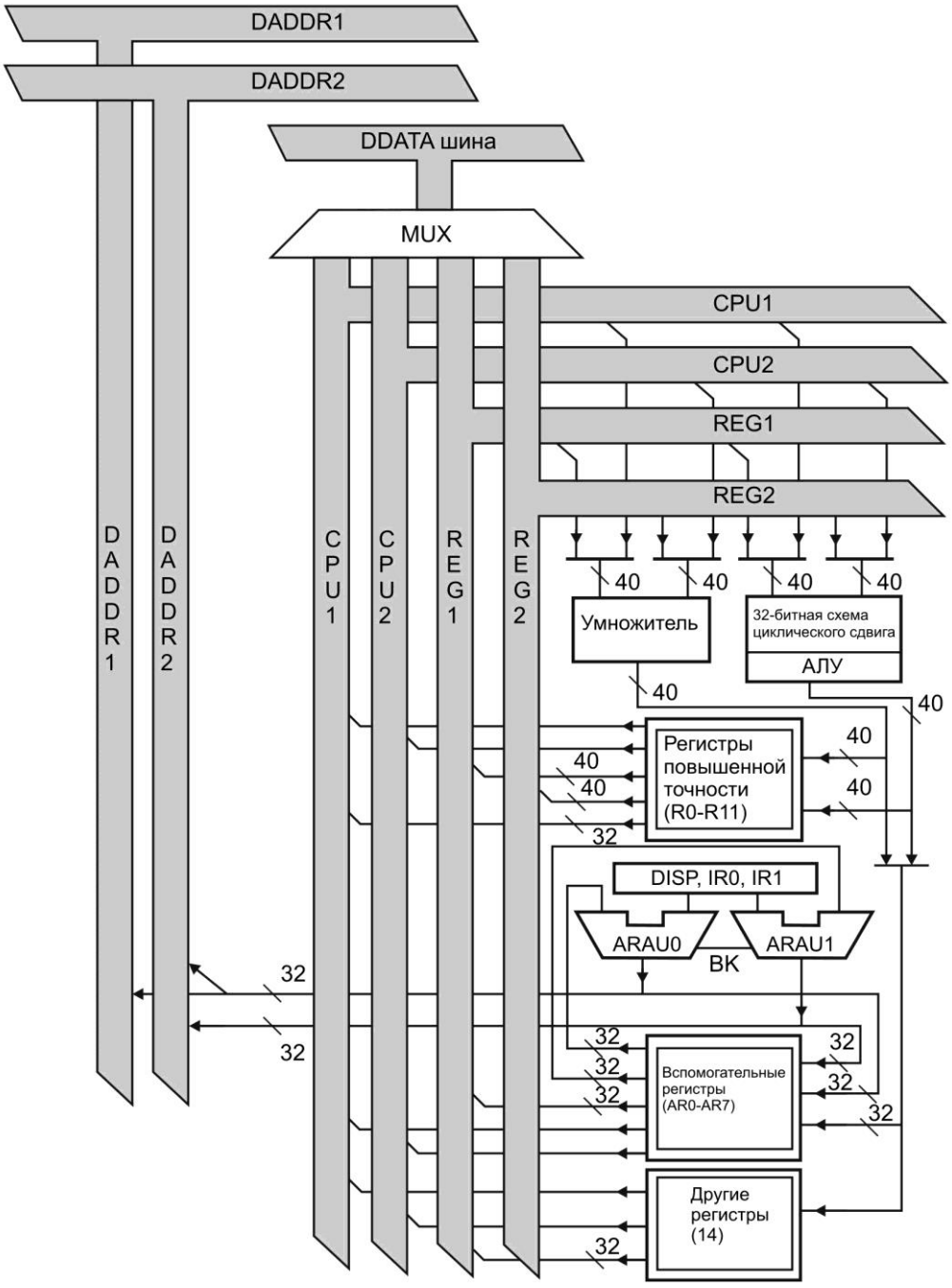


Рисунок 5.2 − Компоненты ЦПУ

5.2.1.1 Умножитель

Умножитель выполняет операции 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.

5.2.1.2 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.

5.2.1.3 Арифметические устройства вспомогательных регистров ARAU*n*

Два арифметических устройства вспомогательных регистров ARAU0 и ARAU1 могут генерировать два адреса в одном цикле. ARAU*n* функционируют параллельно с умножителем и АЛУ. Они обеспечивают вычисление адреса операнда со смещением, индексацию с использованием регистров IR0 и IR1, циклическую и бит-реверсную адресацию операндов.

5.2.1.4 Регистры ЦПУ

ПЦОС 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 подразделе 5.3 представлена более детальная информация, примеры управления стеком и использования регистров.

Регистры ЦПУ, выполняемые ими функции, и ссылки на более детальную информацию приведены в таблице 5.1.

Таблица 5.1 – Основные регистры ЦПУ

Синтаксис ассемблера	Соответствующее имя функции	Ссылка в Руководстве пользователя
R0	Регистр повышенной точности 0	5.3.1.1
R1	Регистр повышенной точности 1	5.3.1.1
R2	Регистр повышенной точности 2	5.3.1.1
R3	Регистр повышенной точности 3	5.3.1.1
R4	Регистр повышенной точности 4	5.3.1.1
R5	Регистр повышенной точности 5	5.3.1.1
R6	Регистр повышенной точности 6	5.3.1.1
R7	Регистр повышенной точности 7	5.3.1.1
R8	Регистр повышенной точности 8	5.3.1.1
R9	Регистр повышенной точности 9	5.3.1.1
R10	Регистр повышенной точности 10	5.3.1.1
R11	Регистр повышенной точности 11	5.3.1.1
AR0	Вспомогательный регистр 0	5.3.1.2
AR1	Вспомогательный регистр 1	5.3.1.2
AR2	Вспомогательный регистр 2	5.3.1.2
AR3	Вспомогательный регистр 3	5.3.1.2
AR4	Вспомогательный регистр 4	5.3.1.2
AR5	Вспомогательный регистр 5	5.3.1.2
AR6	Вспомогательный регистр 6	5.3.1.2
AR7	Вспомогательный регистр 7	5.3.1.2
DP	Указатель страницы данных	5.3.1.3
IR0	Индексный регистр 0	5.3.1.4
IR1	Индексный регистр 1	5.3.1.4
BK	Регистр размера блока	5.3.1.5
SP	Указатель системного стека	5.3.1.6
ST	Регистр состояния	5.3.1.7
DIE	Разрешение прерывания ПДП процессора	5.3.1.8
IIE	Регистр разрешения внутренних прерываний	5.3.1.9
IIF	Флаги ввода-вывода	5.3.1.10
RS	Регистр адреса начала повторений	5.3.1.11
RE	Регистр адреса конца повторений	5.3.1.11
RC	Счётчик повторений	5.3.1.11

Регистры повышенной точности R0 – R11 имеют возможность хранения и выполнения операций с 32-разрядными целыми и 40-разрядными значениями с плавающей запятой. Любая команда, содержащая значения с плавающей запятой, использует разряды 39 – 0. Если операнды являются целыми со знаком или беззнаковыми, используются только разряды 31 – 0, разряды 39 – 32 остаются без изменений. Это верно для всех операций сдвига. В подразделе 5.5 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. Они также могут использоваться для реализации различных функций, таких как циклические счётчики или как 32-разрядные регистры общего назначения, которые могут быть модифицированы умножителем и АЛУ. В подразделе 5.6 приведена подробная информация и примеры использования вспомогательных регистров для адресации.

Указатель страницы данных DP является 32-разрядным регистром. Шестнадцать младших разрядов указателя страницы данных используются в режиме прямой адресации как указатели на адресуемую страницу данных. Длина страницы данных 64K слов, всего имеется 256 страниц.

32-разрядные индексные регистры IR0 и IR1 используются арифметическим устройством вспомогательных регистров ARAU для вычисления индексированного адреса. В подразделе 5.6 приведены примеры использования индексных регистров для адресации.

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

Указатель системного стека SP – это 32-разрядный регистр, содержащий адрес вершины системного стека. SP всегда указывает на последний элемент, загруженный в стек. При вталкивании данных в стек происходит прединкремент, при выталкивании данных из стека – постдекремент указателя системного стека. SP манипулируется прерываниями, переходами, вызовами, возвратами и командами PUSH и POP.

Регистр состояния ST содержит глобальную информацию, относящуюся к состоянию ЦПУ. Обычно операции выставляют флаги условий регистра состояния в соответствии с результатом − нулевым, отрицательным и т. д., включая как операции загрузки и сохранения регистров, так и арифметические и логические функции. Когда регистр состояния загружен, то замена разряда на разряд выполняется над текущим содержимым операнда источника, несмотря на состояние любых разрядов в операнде источника. Следовательно, при следующей загрузке содержимое регистра состояния тождественно равно содержимому операнда источника. Это позволяет легко сохранять и восстанавливать регистр состояния. Подробнее статусный регистр ST описан в 5.3.1.7.

Регистр разрешения прерываний ПДП процессора DIE – это 32-битный регистр, состоящий из 2- и 3-битных полей для указания прерываний, которые синхронизируют работу каждого из шести каналов ПДП. Это позволяет каждому каналу ПДП работать независимо от работы основного ЦПУ. Также каждый ПДП канал может быть синхронизирован с внешним прерыванием или встроенными в кристалл таймерами. Подробнее этот регистр описан в 5.3.1.8.

Регистр разрешения прерываний ЦПУ (IIE) – это 32-битный регистр, который разрешает либо запрещает прерывания ЦПУ от шести коммуникационных портов, двух таймеров и шести каналов ПДП. Подробнее этот регистр описан в 5.3.1.9.

Регистр флага прерываний ЦПУ IF является также 32-разрядным регистром. Единица в бите регистра флага прерывания показывает, что соответствующие прерывания установлены. Ноль показывает, что соответствующие прерывания не установлены.

Регистр флагов ввода-вывода IOF управляет функцией специализированных внешних выводов IIOF(0 – 3)_1#, IIOF(0 – 3)_2# ядер ПЦОС 1 и ПЦОС 2, соответственно. Эти выводы могут быть установлены в качестве входа или выхода, а также в качестве входов внешних прерываний, через них также может быть считана и записана информация. Смотри 5.3.1.10 для получения детальной информации.

Счётчик повторений RC – это 32-разрядный регистр, используемый для точного определения того, сколько раз должен быть повторен блок команд при выполнении повторений 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.

Счётчик команд PC – это 32-разрядный регистр, содержащий адреса следующих выбираемых инструкций. Хотя PC не является частью регистрового файла ЦПУ, он может быть изменён командами, которые изменяют процесс выполнения программы.

5.2.1.5 Расширенный регистровый файл ЦПУ

Кроме основного регистрового файла ЦПУ, расширенный регистровый файл содержит в себе два специальных регистра, которые действуют как указатели:

- IVTP указатель системной векторной таблицы прерываний, которая определяет векторы для всех системных прерываний;
- TVTP указатель программной векторной таблицы прерываний, которая определяет вектора всех программных прерываний.

Эти два регистра полностью описаны в 5.3.2 «Расширенный регистровый файл ЦПУ».

5.2.2 Организация памяти

Общее адресное пространство ИС 1867ВЦ8Ф1 составляет 4096K (четыре Гигабайта) 32-разрядных слов. Программа, данные и область ввода-вывода содержатся внутри этого 4 Гбайт (пословно адресуемого) адресного пространства, обеспечивая, таким образом, хранение таблиц, коэффициентов, кода программы или данных либо и/или в СОЗУ, либо и/или во внутрикристальном ПЗУ, либо и/или во внутрикристальном ОЗУ, подключенными к локальной/глобальной шине, либо и/или во внешнем ОЗУ/ПЗУ, подключенными к локальной/глобальной шине. В этом случае объем памяти может быть максимальным, и область памяти распределяется так, как это необходимо. Управляя одним выводом ROMEN_x, можно выбирать место расположения области памяти от 0000 0000h до 000F FFFFh либо во внешнем ОЗУ (ROMEN_x = 0), либо во внутрикристальном ПЗУ (ROMEN_x = 1).

5.2.2.1 ОЗУ, ПЗУ и кэш

На рисунке 5.3 показана организация памяти ИС 1867ВЦ8Ф1. Микросхема имеет два блока сверхоперативной памяти СОЗУ 0 и СОЗУ 1, ПЗУ и внутрикристальную память ВОЗУ 0 и ВОЗУ 1. Объем каждого блока СОЗУ составляет 1K × 32 бит, объем блока ПЗУ составляет 4K × 32 бит.

Блок внутрикристальной памяти ВОЗУ 0 подключен к локальной шине, а блок внутрикристальной памяти ВОЗУ 1 подключен к глобальной шине. Максимальный объем каждого блока памяти ВОЗУ 0 и ВОЗУ 1 составляет 224K × 32 бит. Блоки СОЗУ и ПЗУ поддерживают два обращения ЦПУ в одном цикле.

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 и выполнять выборку программы из блоков ВОЗУ 0/ВОЗУ 1 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 и обращений к блокам ВОЗУ 0/ВОЗУ 1. Это позволяет увеличить производительность, так как кэш команд поддерживает два обращения за цикл, в то время как внешний интерфейс и ВОЗУ 0/ВОЗУ 1 поддерживают одно обращение за цикл. При этом шины внешнего интерфейса свободны для использования сопроцессором ПДП для выборки из внешней внекристальной памяти и блоков ВОЗУ 0/ВОЗУ 1, а также для использования другими устройствами в системе.

В разделе 5.4 приведена более подробная информация о памяти и командах кэш.

Организация памяти изображена на рисунке 5.3.

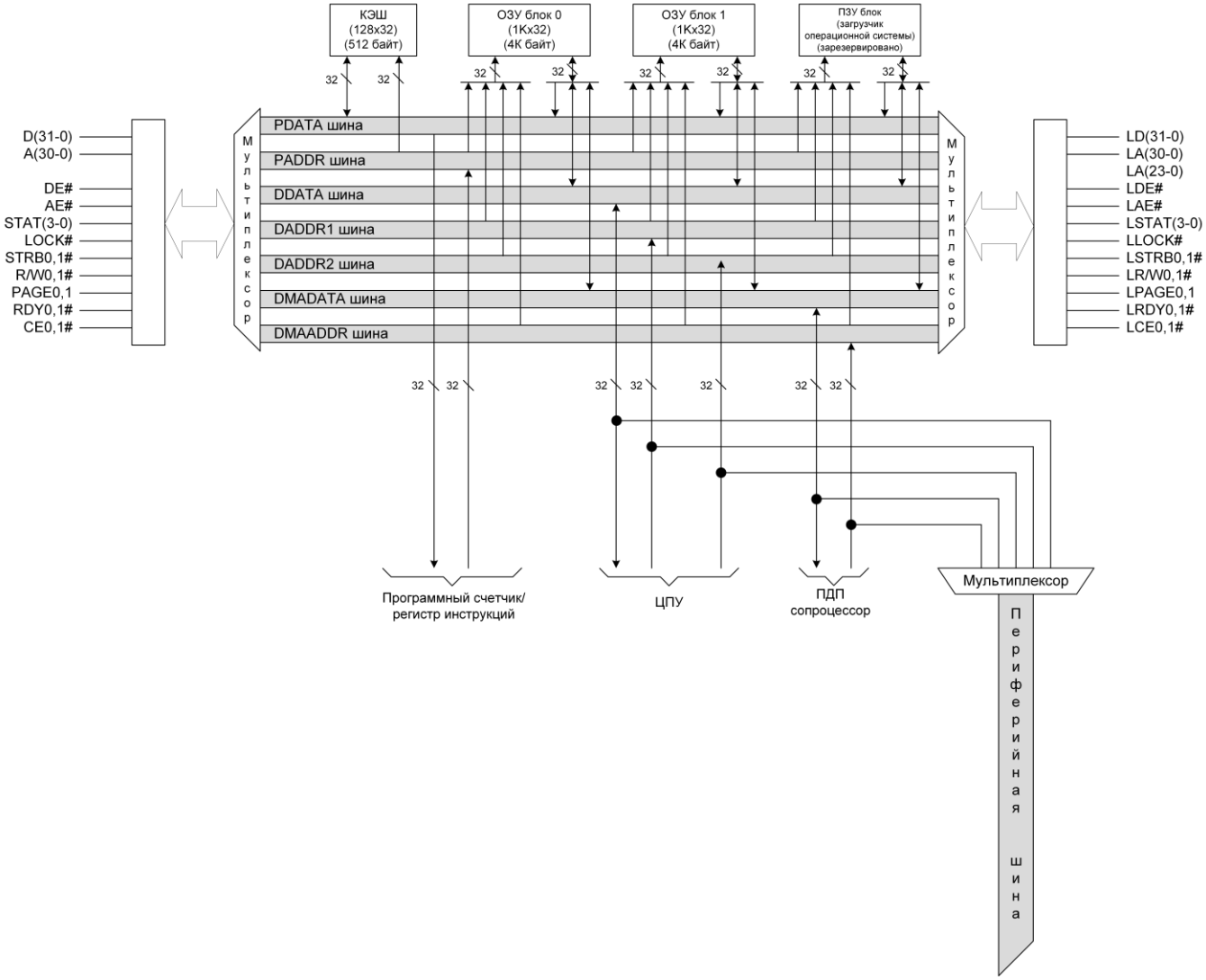


Рисунок 5.3 − Организация памяти

5.2.2.2 Карта памяти

Карта памяти каждого из процессорных ядер ИС приведена на рисунке 5.4. В зависимости от состояния внешнего вывода ROMEN_x карта памяти может иметь разные конфигурации:

- если на ROMEN_x установлен высокий уровень, то область адресного пространства с 0000 0000h по 000F FFFFh зарезервирована для работы внутреннего ПЗУ, и соответствующее процессорное ядро функционирует в режиме микрокомпьютера (правая часть рисунка 5.4);
- если на ROMEN_x установлен низкий уровень, то область адресного пространства с 0000 0000h по 000F FFFFh доступна локальной шине, и соответствующее процессорное ядро функционирует в режиме микропроцессора. (левая часть рисунка 5.4).

Остальная часть карты памяти ядра ПЦОС остаётся неизменной и не зависит от значения сигнала ROMEN_x.

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

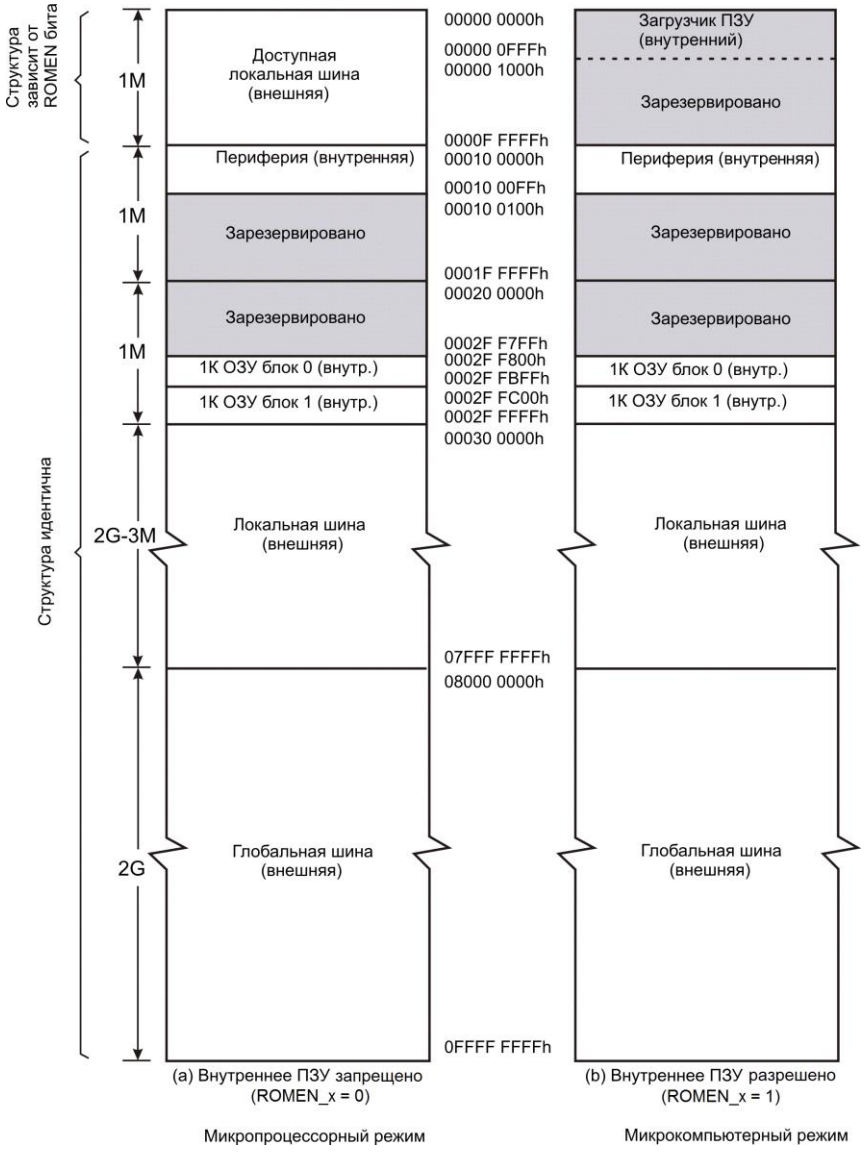


Рисунок 5.4 – Карта памяти процессорных ядер ИС

В таблице 5.2 представлена периферийная карта памяти ПЦОС.

Таблица 5.2 – Периферийная карта памяти

Адрес	Периферийное устройство	Примечание
0010 0000h 0010 000Fh	Регистры управления локальной и глобальной шинами (16 слов)	См. 5.4.2.1, рисунок 5.14
0010 0010h 0010 001Fh	Регистры управления работой ПЦОС (16 слов)	См. 4.2.2
0010 0020h 0010 002Fh	Регистры таймера 0 (16 слов)	См. 5.4.2.3, рисунок 5.15
0010 0030h 0010 003Fh	Регистры таймера 1 (16 слов)	
0010 0040h 0010 004Fh	Регистры коммуникационного порта 0 (16 слов)	См. 5.4.2.4, рисунок 5.16
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)	См. 5.4.2.5, рисунок 5.17
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)	

5.2.2.3 Режимы адресации памяти

ПЦОС реализует базовый набор инструкций общего назначения, такие как арифметические инструкции, которые ориентированы на цифровую обработку сигналов и другие приложения, требующие объёмных числовых вычислений. В разделе 5.6 приведена 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 трёхоперандные 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 счётчика:
- регистровый режим, такой как для основного режима адресации;
- относительно счётчика инструкций PC – 16 разрядов со знаком или 24 разряда смещения прибавляется к содержимому PC.

5.2.3 Внутренние шины ПЦОС

Высокое быстродействие ИС 1867ВЦ8Ф1 достигается за счёт 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, внешнее ОЗУ и регистры внутренних периферийных устройств). На рисунке 5.3 показаны эти внутренние шины и их соединение с блоками процессора.

Программный счётчик PC соединён с 32-битной программной адресной шиной PADDR. Регистр инструкций IR подключен к 32-битной программной шине данных PDATA. Такая шинная структура даёт 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 передаёт 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 5.2 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.

5.2.4 Внешние шины ПЦОС

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 подразделе 5.9.

5.2.5 Прерывания

Ядро процессора 1867ВЦ8Ф1 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 5.7.4.

5.2.6 Периферийные устройства

Все периферийные устройства процессора ИС 1867ВЦ8Ф1 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 1867ВЦ8Ф1 состоят из двух таймеров, шести коммуникационных портов и устройства управления PLL. На рисунке 5.5 представлены периферийные устройства ИС.

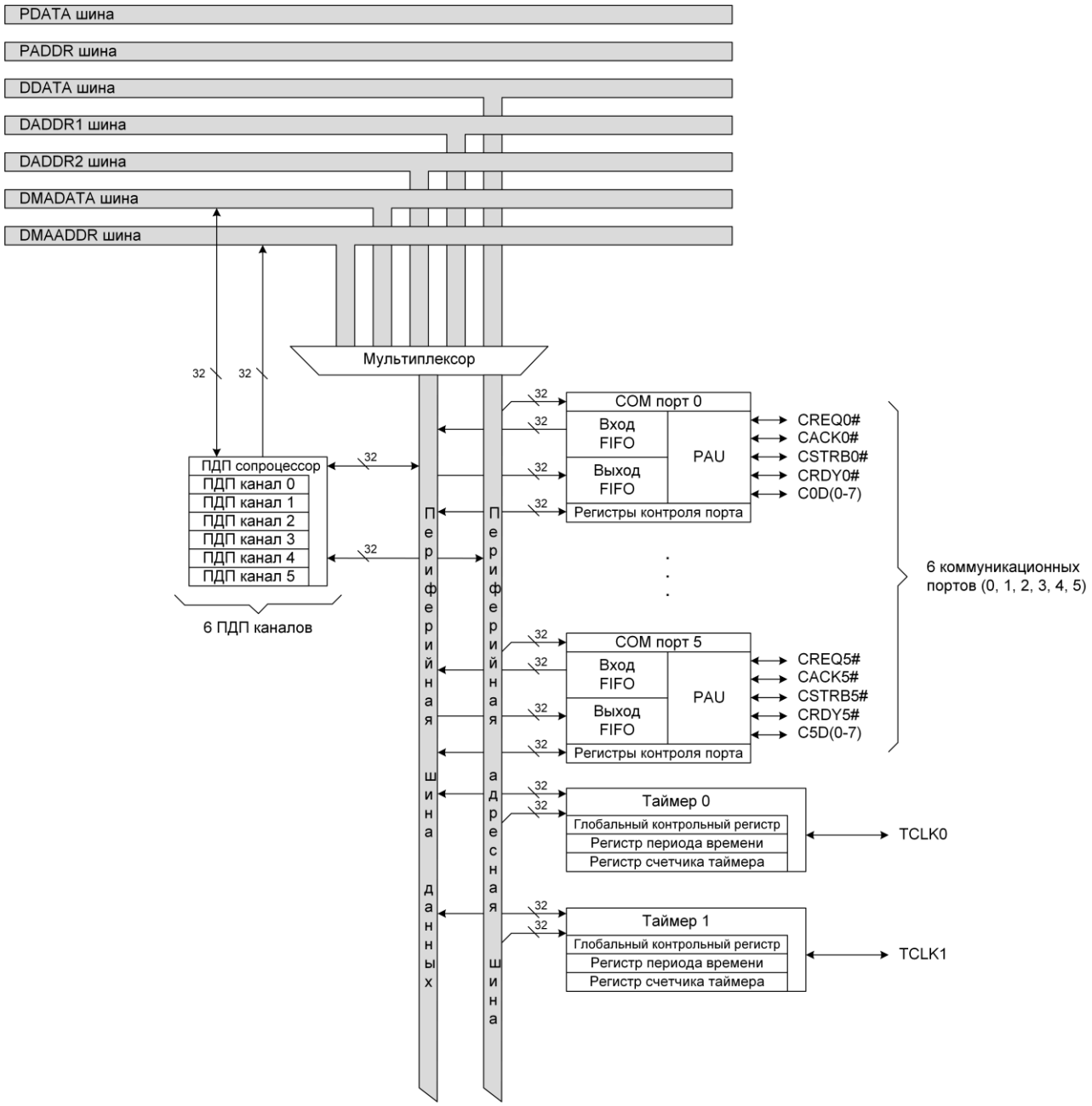


Рисунок 5.5 – Периферийные устройства ПЦОС

5.2.6.1 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 ресурсоёмких вычислительных задач, в которой достигается оптимальная системная производительность за счёт 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 подразделе 5.12.

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.

5.2.6.2 Сопроцессор ПДП

Шесть каналов встроенного сопроцессора ПДП могут, независимо от работы основного ЦПУ, читать/записывать данные в любой адрес, который определён 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 счётчик 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 подразделе 5.11.

5.2.6.3 Таймеры

Каждый таймер является 32-разрядным универсальным счётчиком времени или числа событий. Таймер имеет 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 рассматриваются в подразделе 5.13.

5.3 Регистры ЦПУ

Главный регистровый файл ЦПУ объединяет 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, управление стеком и состоянием процессора, блоковые повторения, ветвления и прерывания.

Расширенный регистровый файл ЦПУ объединяет два регистра – указатель системной векторной таблицы IVTP и указатель программной таблицы прерывания TVTP.

В этом подразделе описывается каждый регистр ЦПУ.

5.3.1 Главный регистровый файл ЦПУ

Ядро процессора 1867ВЦ8Ф1 имеет 32 регистра в мультипортовом регистровом файле, который тесно связан с ЦПУ. Счётчик команд PC не включен в эти 32 регистра. Состав регистрового файла представлен в таблице 5.3.

Таблица 5.3 – Главный регистровый файл ЦПУ

Регистровый символ	Регистровое машинное значение (шестнадцатеричное)	Назначенное имя функции	Ссылка
1	2	3	4
R0	00	Регистр повышенной точности 0	5.3.1.1
R1	01	Регистр повышенной точности 1	5.3.1.1
R2	02	Регистр повышенной точности 2	5.3.1.1
R3	03	Регистр повышенной точности 3	5.3.1.1
R4	04	Регистр повышенной точности 4	5.3.1.1
R5	05	Регистр повышенной точности 5	5.3.1.1
R6	06	Регистр повышенной точности 6	5.3.1.1
R7	07	Регистр повышенной точности 7	5.3.1.1
R8	1C	Регистр повышенной точности 8	5.3.1.1
R9	1D	Регистр повышенной точности 9	5.3.1.1
R10	1E	Регистр повышенной точности 10	5.3.1.1
R11	1F	Регистр повышенной точности 11	5.3.1.1
AR0	08	Вспомогательный регистр 0	5.3.1.2
AR1	09	Вспомогательный регистр 1	5.3.1.2
AR2	0A	Вспомогательный регистр 2	5.3.1.2
AR3	0B	Вспомогательный регистр 3	5.3.1.2
AR4	0C	Вспомогательный регистр 4	5.3.1.2
AR5	0D	Вспомогательный регистр 5	5.3.1.2
AR6	0E	Вспомогательный регистр 6	5.3.1.2
AR7	0F	Вспомогательный регистр 7	5.3.1.2
DP	10	Указатель страницы данных	5.3.1.3
IR0	11	Индексный регистр 0	5.3.1.4
IR1	12	Индексный регистр 1	5.3.1.4
BK	13	Регистр размера блока	5.3.1.4
SP	14	Указатель системного стека	5.3.1.6
ST	15	Регистр состояния	5.3.1.7
DIE	16	Регистр разрешения прерывания ПДП сопроцессора	5.3.1.8
IIE	17	Регистр разрешения внутреннего прерывания	5.3.1.9
IIF	18	Регистр флагов IIOF (IIOF(3-0)_x#, таймеры, ПДП)	5.3.1.10
RS	19	Регистр адреса начала повторений	5.3.1.11
RE	1A	Регистр адреса конца повторений	5.3.1.11
RC	1B	Счётчик повторений	5.3.1.11

Все регистры, представленные в таблице 5.3, 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 режимов косвенной адресации и могут быть использованы как универсальные 32-битные целочисленные и логические регистры. Оставшиеся регистры поддерживают системные функции, такие как адресация, управление стеком, состоянием процессора, прерываниями и блоковыми повторениями. В подразделе 5.6 приведена более подробная информация и примеры использования регистров ЦПУ для адресации.

5.3.1.1 Регистры повышенной точности R0 − R11

12 регистров повышенной точности R0 − R11 могут хранить и оперировать с 32-разрядными целыми и 40-разрядными значениями с плавающей запятой.

Эти регистры состоят из двух различных областей:

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

Любая команда, которая содержит операнд с плавающей запятой, использует разряды 39 − 0. Рисунок 5.6 иллюстрирует хранение 40-разрядных значений с плавающей запятой в регистрах повышенной точности.

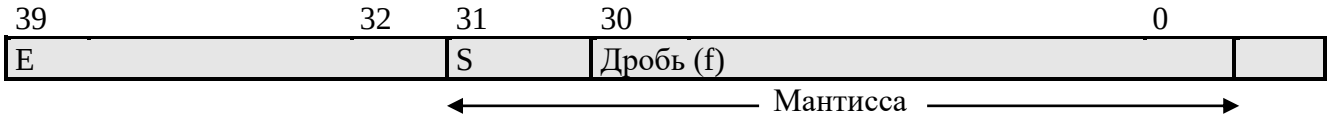


Рисунок 5.6 − Формат числа с плавающей запятой в регистре повышенной точности

Для работы с целыми значениями разряды 31 − 0 регистров повышенной точности содержат целые значения (со знаком или без знака). Любая команда, которая содержит операнды с целыми значениями со знаком или без знака используют 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 5.7.

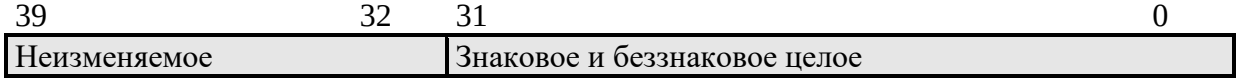


Рисунок 5.7 − Формат целого числа в регистре повышенной точности

5.3.1.2 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 32-разрядных адресов. Однако они также могут быть использованы для выполнения различных функций, таких как циклический счётчик, для косвенной адресации или 32-разрядные регистры общего назначения, которые могут быть изменены умножителем и АЛУ. В подразделе 5.6 приведена более подробная информация и примеры использования вспомогательных регистров для адресации.

5.3.1.3 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. Страница данных имеет длину 64K слов (65536) страниц. Биты 31 − 16 зарезервированы, они всегда читаются, как нули, и не могут быть изменены записью в регистр. Указатель страницы данных DP может быть загружен с использованием LDP или LDI инструкции.

На рисунке 5.42 показаны эти регистровые функции.

5.3.1.4 Индексные регистры IR0, IR1

32-разрядные индексные регистры IR0, IR1 используются арифметическим устройством вспомогательных регистров ARAU для индексирования адресов. Индексный регистр IR0 также используется для бит-реверсной адресации. В разделе 6 приведена подробная информация и примеры использования индексных регистров при адресации.

5.3.1.5 Регистр размера блока BK

32-разрядный регистр размера блока BK используется ARAU при циклической адресации для точного определения размера блока данных. В 5.6.8 приведена подробная информация об использовании регистра размера блока.

5.3.1.6 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 манипулируется программными прерываниями, системными прерываниями, вызовами, возвратами и командами PUSH, PUSHF, POP, POPF. При выталкивании из стека и вталкивании данных в стек выполняется прединкремент и постдекремент указателя стека на всех 32 разрядах.

5.3.1.7 Регистр состояния ST

Регистр состояния ST содержит глобальную информацию о состоянии ЦПУ. Обычно операции устанавливают флаги в регистре состояния в соответствии с наличием нулевого, отрицательного результата и т. д. Сюда включаются операции загрузки и хранения регистра, а также арифметические и логические операции. Однако если регистр состояния загружен, содержимое операнда источника заменяет текущее содержимое разряд в разряд, независимо от состояния любого разряда в операнде источника. Поэтому следующая загрузка содержит регистр состояния, идентичный содержимому операнда источника. Это позволяет достаточно просто сохранять и восстанавливать регистр состояния. При системном сбросе в этот регистр записывается 0, после сброса CF устанавливается в 1. Формат регистра состояния показан на рисунке 5.8. Далее представлено описание каждого поля регистра состояния.

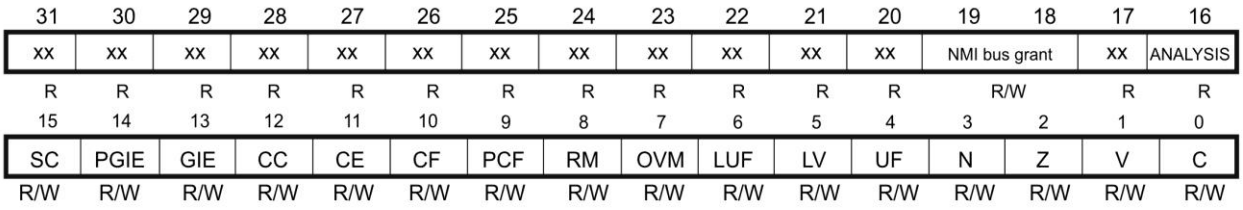


Рисунок 5.8 − Регистр состояния ST

Принятые условные обозначения на рисунке 5.8:

- xx – резервный разряд;
- R – чтение;
- W – запись;
- C – флаг переноса;
- V – флаг переполнения;
- Z – флаг нулевого значения;
- N – флаг отрицательного значения;
- UF – флаг потери значимости разрядов числа с плавающей запятой;
- LV – фиксируемый флаг переполнения;
- LUF – фиксируемый флаг потери значимости разрядов числа с плавающей запятой;
- OVM – флаг режима переполнения. Этот флаг действует только при целочисленных операциях. Если OVM = 0, то режим переполнения отключен; целые результаты с переполнением не обрабатываются специальным методом. Если OVM = 1:

а) целые положительные результаты с переполнением устанавливаются в значение наибольшего положительного 32-разрядного числа в дополнительном коде (7FFFFFFFh);

б) целые отрицательные результаты устанавливаются в наименьшее отрицательное 32-разрядное число в дополнительном коде (80000000h);

- RM – флаг режима повторения. Если RM = 1, то PC модифицируется или при повторе блока команд, или в режиме одиночного повторения;
- PCF – предыдущее состояние бита CF. Когда происходит программное или системное прерывание, то CF бит устанавливается в 1, а PCF принимает предыдущее значение бита CF.

RETI и RETID инструкции копируют PCF в CF бит;

- CF – «замораживание» кэш, см. таблицу 5.4. Если CF = 1, то кэш заморожен. Если кэш разрешён CE = 1, то выборка из кэш разрешена, но состояние кэш не изменяется. Эта функция может быть использована для сохранения часто используемых кодов резидентно в кэш.

При сбросе в этот разряд заносится 0. Очистка кэш CC = 1 разрешается, если CF = 0;

- CE – разрешение кэш, см. таблицу 5.4. CE = 1 разрешает кэш, позволяя использовать кэш в соответствии с алгоритмом кэш LRU (наименее использованный). CE = 0 запрещает кэш; кэш не изменяется. Выборка из кэш не выполняется. Эта функция используется для отладки системы.

При сбросе в этот разряд заносится 0. Очистка кэш CC = 1 разрешается, если CE = 0;

- CC – очистка кэш. CC = 1 запрещает все содержимое кэш. Этот разряд всегда очищается после того, как он записан, и всегда читается как 0. При сбросе в этот разряд записывается 0;

Таблица 5.4 − Описание CE и CF битов

CE	CF	Эффект
0	0	Кэш не разрешён
0	1	Кэш не разрешён
1	0	Кэш разрешён и не заморожен
1	1	Кэш разрешён, но заморожен (кэш только читается)

- GIE – глобальное разрешение прерываний. Если GIE = 1, то ЦПУ отвечает на разрешённое прерывание. Если GIE = 0, то ЦПУ не отвечает на разрешённое прерывание. NMI# не оказывает влияния на состояние GIE бита. IDLE, LAT, RETI, RETID и TRAP инструкции влияют на значение GIE бита. GIE устанавливается в 0, когда возникает программное или системное прерывание;

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

- SET COUND (SC) – этот бит определяет установки флагов условий (ST биты 0 – 6).

Если SET COUND = 0, флаги условий устанавливаются, если конечным операндом является какой-либо регистр повышенной точности R0 – R11. Эта установка ядра процессора 1867ВЦ8Ф1 аналогична установке флагов условий в ПЦОС 1867ВЦ3Ф. При сбросе в этот разряд заносится 0.

Если SET COUND = 1, флаги условий устанавливаются, если конечным операндом является некоторый регистр в главном регистровом файле, исключая регистр состояния. Флаги условий всегда устанавливаются при выполнении CMPF, CMPI, CMPF3, CMPI3, TSTB или TSTB3 инструкций, несмотря на значение SET COUND;

- ANALYSIS – бит только для чтения, используется в режиме анализа для обеспечения информации состояния для эмуляции;

- NMI# – разрешение передачи по шине, NMI# используется при корректировке ошибок работы коммуникационного порта при использовании программного сброса. Если бит 19 = 1 и бит 18 = 0, то внутренний сигнал разрешения передачи по периферийной шине устанавливается по заднему фронту NMI#. Если NMI# определён, когда периферийная шина находится в состоянии останова, то NMI# прерывает незаконченный цикл, и происходит переход к обслуживающей программе NMI#. Состояние останова может возникнуть при записи в заполненный буфер FIFO или при чтении из пустого буфера FIFO;

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

5.3.1.8 Регистр разрешения прерывания сопроцессора ПДП (DIE). Основной и раздельный режимы

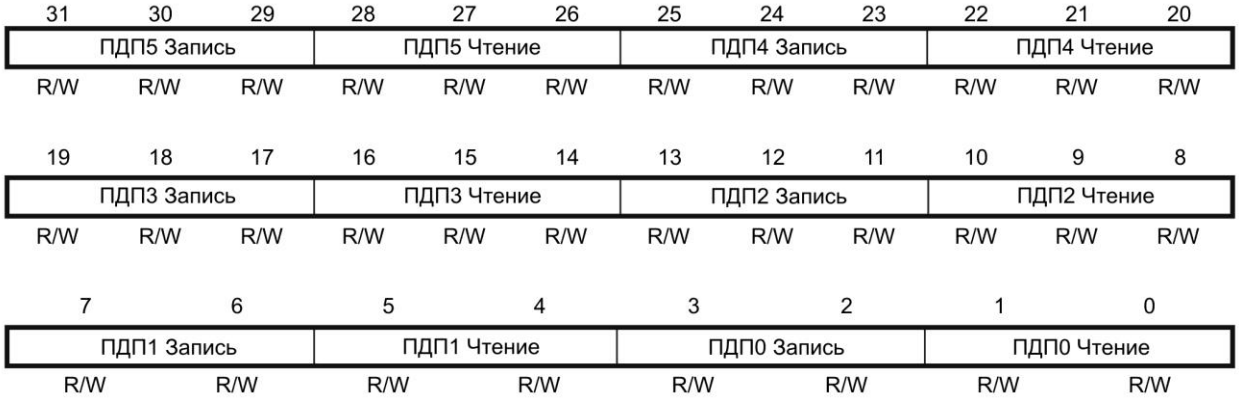
32-битный регистр разрешения прерывания ПДП (DIE) разделён на шесть частей, которые определяют, какие прерывания могут быть использованы при управлении синхронизацией для каждого из шести каналов сопроцессора ПДП. Управление синхронизацией происходит при чтении или записи в канале ПДП. При сбросе во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

Каждый канал ПДП проверяет не только определённые прерывания синхронизации ПДП, но также в режиме синхронизации – какой канал в настоящее время используется, см. таблицу 5.38. Режим синхронизации определяется полем SYNC MODE в регистрах управления каналами ПДП, которые находятся в сопроцессоре ПДП.

Используя синхронизационные прерывания, каждый канал ПДП может, например, обслуживать соответствующий коммуникационный порт. Необходимо обратить внимание, что ПДПi может быть синхронизирован только с сигналами, приходящими от коммуникационного порта i (где $0 \le i \le 5$). Также, каждый канал ПДП может быть синхронизирован с внешними прерываниями и встроенными таймерами.

Основной режим

На рисунке 5.9 показан регистр разрешения прерываний ПДП для основного режима. В таблице 5.5 представлены все возможные комбинации ПДП0 и ПДП1 для основного режима. В таблице 5.6 представлены все трёхбитные комбинации ПДП2 – ПДП5 для основного режима.



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

Рисунок 5.9 − Регистр разрешения прерываний ПДП для основного режима ПДП

Таблица 5.5 − Синхронизационные прерывания основного режима каналов ПДП0 и ПДП1

Значение бит (в ПДП0 или в ПДП1)	Разрешение прерывания ПДП0 или ПДП1				Источник прерывания для синхронизации ПДП
	ПДП0 Чтение	ПДП0 Запись	ПДП1 Чтение	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 выводов IIOF0# – IIOF3#
1 1	TIM0	TIM0	TIM0	TIM0	От таймера TIM0

* При использовании синхронной передачи канал ПДП остановлен (операции чтения или записи не выполняются).

Таблица 5.6 − Синхронизационные прерывания основного режима каналов ПДП2 – ПДП5

Значение бит (в ПДП2 – ПДП5)	Разрешение прерываний ПДП2 – ПДП5*		Источник прерывания для синхронизации ПДП
	ПДПх Чтение	ПДПх Запись	
0 0 0**	нет	нет	– –
0 0 1	ICRDYx*	ICRDYx*	От коммуникационного порта
0 1 0	IIOF0#	IIOF0#	От внешних выводов IIOF0# – IIOF3#
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	

Примечание – Сопроцессор ПДП использует сигналы для синхронизации.

Прерывания в таблице 5.5 и таблице 5.6 (ICRDYx, OCRDYx, TIM0 и т. д.) не векторные. Сопроцессор ПДП использует их как сигналы для синхронизации передачи. Описание в 5.11.10.

_____

* x в ПДПх – это номер канала, который также является номером для соответствующих прерываний ICRDYx и OCRDYx. Например, 001_2 в ПДП2 READ и ПДП5 WRITE разрешит прерывания ICRDY2 и OCRDY5 соответственно. Все остальные допустимые битовые значения (от 010_2 до 111^2) одинаковы для ПДП2 – ПДП5.

** При использовании синхронной передачи канал ПДП остановлен (операции чтения или записи не выполняются).

Раздельный режим

На рисунке 5.10 приведён формат регистра разрешения прерываний ПДП для раздельного режима. В таблице 5.7 представлены все возможные комбинации ПДП0 и ПДП1 для раздельного режима. В таблице 5.8 представлены все трёхбитные комбинации ПДП2 – ПДП5 для раздельного режима.

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

Рисунок 5.10 − Регистр разрешения прерываний ПДП для раздельного режима ПДП

Таблица 5.7 − Синхронизационные прерывания раздельного режима каналов ПДП0 и ПДП1

Значение бит (в ПДП0 или в ПДП1)	Разрешение прерывания ПДП0 или ПДП1				Источник прерывания синхронизации ПДП
	ПДП0 Вспомога-тельный Чтение	ПДП0 Основной Запись	ПДП1 Вспомога-тельный Чтение	ПДП1 Основной 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 IIOF0# − IIOF3#
1 1	TIM0	TIM0	TIM0	TIM0	От таймера TIM0

_____

* При использовании синхронной передачи канал ПДП остановлен (операции чтения или записи не выполняются).

Таблица 5.8 − Синхронизационные прерывания раздельного режима каналов ПДП2 – ПДП5

Значение бита (в ПДП2 − ПДП5)	Разрешение прерываний ПДП2 − ПДП5*		Источник прерывания синхронизации ПДП
	ПДПx Вспомогательный Чтение*	ПДПx Основной Запись*	
0 0 0**	нет	нет	– –
0 0 1	ICRDYx*	OCRDYx*	От коммуникационного порта
0 1 0	IIOF0#	IIOF0 #	От внешних сигналов IIOF0# − IIOF3#
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	

_____

* x в ПДПx – это номер канала, который также является номером для соответствующих прерываний ICRDYx и OCRDYx. Например, 001_2 в ПДП2 READ и ПДП5 WRITE разрешит прерывания ICRDY2 и OCRDY5 соответственно. Все остальные допустимые битовые значения (от 010_2 до 111^2) одинаковы для ПДП2 – ПДП5.

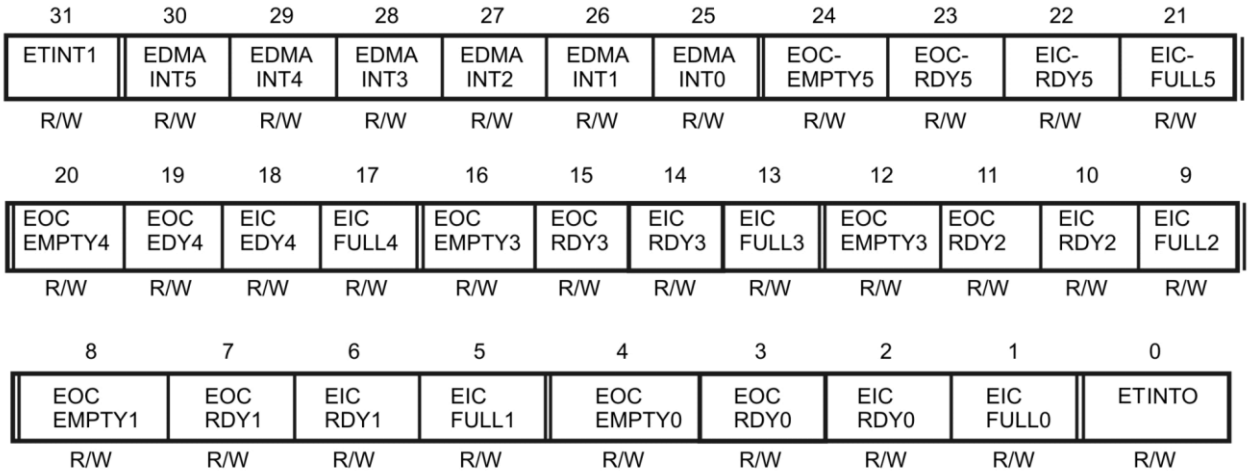
** При использовании синхронной передачи канал ПДП остановлен (операции чтения или записи не выполняются).

5.3.1.9 Регистр разрешения внутренних прерываний ЦПУ IIE

Формат 32-разрядного регистра разрешения внутренних прерываний IIE приведён на рисунке 5.11. Регистр IIE разрешает/запрещает следующие прерывания ЦПУ:

- от таймеров 0 и 1;
- от коммуникационных портов 0 – 5:
  - входной буфер заполнен;
  - входной буфер готов;
  - выходной буфер готов;
  - выходной буфер пуст;
- от сопроцессора ПДП, каналы 0 – 5.

На рисунке 5.11 показаны разряды регистра IIE. 1 означает разрешение прерывания, 0 – запрет. При сбросе все разряды обнуляются.



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

Рисунок 5.11 – Регистр разрешения внутренних прерываний IIE

Примечание – Поля, отделённые двойными линиями, относятся к разным блокам.

Ниже приведены описания для каждого бита IIE:

EICFULLx – прерывание, если входной буфер коммуникационного порта x заполнен;

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 пуст;

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 разряде номер пять означает прерывание по заполнению входного буфера первого коммуникационного порта. Установка в 1 соответствующего разряда разрешает прерывание, 0 – запрещает прерывание.

5.3.1.10 Регистр флагов IIOF (IIF)

Регистр IIF, формат которого приведён на рисунке 5.12, управляет внешними выводами IIOF(3 – 0)_x#. Программируя соответствующим образом этот регистр можно задать:

- какие выводы IIOF(3 – 0)_x# будут сконфигурированы в качестве входов/выходов общего назначения и какие будут задействованы как входы прерываний;
- является ли вывод входом (только чтение) или выходом (чтение/запись);
- вход прерывания активируется фронтом или уровнем;
- разрешено ли внешнее прерывание или нет.

Регистр IIF также включает в себя флаги прерываний таймера, ПДП и NMI#. На рисунке 5.12 представлены разряды регистра IIF. Ниже приведено описание каждого разряда.

Разряды регистра IIF программно доступны на чтение и запись, что позволяет конфигурировать выводы IIOF(3 – 0)_x# либо как входы/выходы общего назначения, либо как входы прерывания. Например, если в IIF регистре FUNCx = 0 (вход/выход) и TYPEx = 1 (выход), тогда путём программирования битов FLAGx, можно также задавать уровень на внешних выводах IIOF(3 – 0)_x#. Если FUNCx = 1 (прерывание), запись 1 в разряд FLAGx регистра IIF будет равнозначна появлению соответствующего сигнала прерывания. Следовательно, все прерывания могут быть установлены и/или сброшены программно. Так как биты прерываний могут быть прочитаны, они могут быть опрошены программно, когда интерфейс управления прерываниями не задействован.

Внутренние прерывания обрабатываются аналогично. Биты в IIF регистре отвечающие за внутренние прерывания (например, TINT0, TINT1) могут быть считаны и записаны программно. Запись 1 активирует прерывание, запись 0 – сбрасывает. Все внутренние прерывания выполняются за один цикл H1/H3. Для модификации содержимого IIF используются логические операции (AND, OR и т. д.) как показано далее:

Корректно	Некорректно
LDI @MASK, R0	LDI IIF, R1
AND R0, IIF	AND @MASK, R1
	LDI R1, IIF

Программные и системные прерывания кратко описаны в 5.3.2 и более детально в 5.7.4 и 5.7.5.

31	30	29	28	27	26	25	24
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

Принятые условные обозначения: R − чтение, W − запись, R/W − чтение/запись.

Рисунок 5.12 − Регистр флагов прерываний IIF

FUNCx – определяет режим вывода IIOF(3–0)_x#:
- если FUNCx = 0, вывод IIOF(3–0)_x# – это вход/выход (R/W) общего назначения;
- если FUNCx = 1, IIOF(3–0)_x# − вход прерывания.

TYPEx − задаёт тип функции для IIOF(3–0)_x#:
- если сигнал IIOF(3–0)_x# − вход/выход общего назначения (FUNCx = 0):
  - TYPEx = 0 конфигурирует IIOF(3–0)_x# входом;
  - TYPEx = 1 конфигурирует IIOF(3–0)_x# выходом.
- если вывод IIOF(3–0)_x# − вход прерывания (FUNCx = 1):
  - TYPEx = 0 – IIOFx прерывание будет фиксироваться по фронту;
  - TYPEx = 1 – IIOF(3–0)_x# прерывание будет активироваться по уровню.

FLAGx − флаг вывода IIOF(3–0)_x#:
- если вывод IIOF(3–0)_x# − вход сигнала общего назначения (FUNCx = 0, TYPEx = 0):
  - FLAGx равен значению IIOF(3–0)_x# и только для чтения;
- если вывод IIOF(3–0)_x# − выход сигнала общего назначения (FUNCx = 0, TYPEx = 1):
  - FLAGx равен значению IIOF(3–0)_x# и для чтения, и для записи.
- если вывод IIOF(3–0)_x# − вход прерывания (FUNCx = 1):
  - FLAGx = 0, если прерывание не установлено;
  - FLAGx = 1, если прерывание установлено.

Если в FLAGx записан 0 (ноль), соответствующее прерывание сбрасывается.

- EIIOFx − запрет/разрешение внешнего прерывания:
    - EIIOFx = 0 запрещает внешние прерывания по сигналу IIOF(3–0)_x#;
    - EIIOFx = 1 разрешает внешние прерывания по сигналу IIOF(3–0)_x#.

NMI − флаг немаскированного прерывания NMI#.

NMI# прерывание (по внешнему сигналу NMI#) ведёт себя подобно другим прерываниям, исключая те, которые не могут быть маскированы (запрещены) посредством бита GIE (ST бит 13) или посредством записи бита NMI. Это временное маскирование во время задержанных переходов и многоцикловых операций ЦПУ. При сбросе этот бит равен 0. Установленное прерывание очищается только посредством обслуживания прерывания. NMI# – прерывание, фиксируемое по заднему фронту и по уровню. Бит NMI предназначен только для чтения.

Если при чтении NMI# = 0, прерывание не установлено.

Если при чтении NMI# = 1, прерывание установлено.

Reserved − зарезервировано; читается как ноль.

TINT0 − флаги прерывания таймеров 0 и 1.

TINT1 − если при чтении TINTx = 0, прерывание таймера не установлено.

Если при чтении TINTx = 1, прерывание таймера установлено.

Запись нуля в этот бит очищает прерывание.

DMAINTx − флаг прерывания каналов 0 − 5 сопроцессора ПДП.

Если при чтении DMAINTx = 0, прерывание канала не установлено.

Если при чтении DMAINTx = 1, прерывание канала установлено.

Запись нуля в этот бит очищает прерывание.

Примечания

1. Затенённые IIF разряды 0, 1, 2, 3 используются IIOF0_x#; затенённые IIF разряды 4, 5, 6, 7 используются IIOF1_x# и т. д.

2. x означает соответствующий IIOF# вход прерывания IIOF(0 – 3)#.

5.3.1.11 Регистры блоков повторений RS, RE, счётчик повторений RC, счётчик команд PC

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

Регистр конечного адреса повторений RE − 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 возврат. Однако, ST(RM) бит остаётся установленным в 1.

Счётчик повторений RC − это 32-разрядный регистр, используемый для точного определения количества повторений блока кода при выполнении. Если в RC записано число n, цикл осуществляется (n+1) раз.

Счётчик команд PC − это 32-разрядный регистр, содержащий адрес следующей выполняемой команды. Счётчик команд не входит в регистровый файл, он является регистром, который может изменяться командами в процессе выполнения программы.

5.3.1.12 Скрытые биты и совместимость

Чтобы добиться совместимости с будущими ИС семейства микропроцессоров ПЦОС, резервные разряды, читающиеся как 0, должны быть записаны как 0. Резервные разряды, имеющие неопределённые значения, не должны изменять текущее значение. В остальных случаях пользователь должен поддерживать резервные разряды точно определёнными.

5.3.2 Расширенный регистровый файл ЦПУ

Расширенный регистровый файл ЦПУ включает в себя два специализированных управляющих регистра:

- указатель системной векторной таблицы прерывания IVTP;
- указатель программной таблицы прерывания TVTP.

Таблица 5.9 − Расширенные регистры ЦПУ

Ассемблерный синтаксис	Регистровое машинное значение	Функциональное имя
IVTP	00	Указатель системной векторной таблицы прерывания. Точки начала системной векторной таблицы прерывания.
TVTP	01	Указатель программной таблицы прерывания. Точки начала программной таблицы прерывания.

Для загрузки (копирования) расширенного регистра в основной регистр (например, в какой-нибудь вспомогательный регистр AR0 − AR7; см. таблицу 5.1), необходимо использовать инструкцию LDEP.

Например:

LDEP IVTP, AR5; содержимое IVTP в AR5

Подобным образом необходимо использовать инструкцию LDPE для загрузки (копирования) основного регистра в расширенный регистр. Ни та, ни другая из этих инструкций не влияют на флаги условий регистра состояния.

LDPE AR5, IVTP; содержимое AR5 в IVTP.

Заметим, что таблицы вектора системных прерываний и вектора программных прерываний не должны превышать 512 слов, таким образом, девять наименьших знаковых разрядов этих указателей являются нулями (т. е. $1\ 000\ 000\ 000_2 = 512 = 200h$). В эти разряды допускается записывать только нули.

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

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

Векторные таблицы системных и программных прерываний могут разделять одно 512 байтное пространство в памяти. При такой конфигурации вы можете расположить вектора программных прерываний там, где нет векторов системных прерываний. Например, если вектор системного прерывания 02Ch не используется, вы можете поместить вектор программного прерывания в IVTP+02Ch (который также является TVTP+02Ch, если таблицы перекрываются) и после вызвать это программное прерывание посредством определения 02Ch в инструкции TRAP.

При сбросе IVTP и TVTP устанавливаются в 0.

5.4 Память и кэш инструкций

Общее адресное пространство ПЦОС – 4Г 32-разрядных слов. Программа, данные и область ввода/вывода содержатся внутри этого 4Г слов адресного пространства, обеспечивая, таким образом, хранение таблиц, коэффициентов, программы или данных либо в ОЗУ, либо в ПЗУ. В этом случае использование памяти может быть максимальным, и область памяти распределяется так, как это необходимо.

Ядро процессора 1867ВЦ8Ф1 имеет два блока ОЗУ 0 и 1, блоки содержат каждый по 1К × 32 бит. Блок ПЗУ – 4К × 32 бит. Каждый из блоков ОЗУ и ПЗУ поддерживает два обращения к ЦПУ в одном цикле. Наличие раздельных шин команд, шин данных и шин ПДП обеспечивает параллельно: выборку программы, 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 – все внутри одного цикла.

128 × 32-разрядный кэш команд обеспечивает хранение часто повторяющихся фрагментов кода, что значительно уменьшает число необходимых внекристальных обращений. Эта операция допускается для кодов, сохраняемых вне кристалла в медленной памяти. Внешние шины в этом случае также свободны для использования ПДП, выборки внешней памяти или для использования другими устройствами в системе. В данном подразделе приведена более подробная информация о памяти и кэш инструкций.

5.4.1 Карта памяти

Карта памяти каждого из процессорных ядер ИС приведена на рисунке 5.13. В зависимости от состояния внешнего вывода ROMEN_x карта памяти может иметь разные конфигурации:

- если на ROMEN_x установлен высокий уровень, то область адресного пространства с 0000 0000h по 000F FFFFh зарезервирована для работы внутреннего ПЗУ, и соответствующее процессорное ядро функционирует в режиме микрокомпьютера (правая часть рисунка 5.13);
- если на ROMEN_x установлен низкий уровень, то область адресного пространства с 0000 0000h по 000F FFFFh доступна локальной шине, и соответствующее процессорное ядро функционирует в режиме микропроцессора. (левая часть рисунка 5.13).

Остальная часть карты памяти ядра ПЦОС остаётся неизменной и не зависит от значения сигнала ROMEN_x.

Доступ к области памяти, начинающейся с адреса 0010 0000h, не зависит от ROMEN_x. Далее приведён обзор основных адресных диапазонов:

- 00000000h-000FFFFFh: локальная шина или встроенное ПЗУ, в зависимости от значения ROMEN_x;
- 00100000h-001000FFh: внутренние периферийные устройства (сопроцессор ПДП, коммуникационные порты, таймеры и т. д.);
- 00100100h-002FF7FFh: зарезервированы;
- 002FF800h-002FFBFFh: 1K ОЗУ, блок 0;
- 002FFC00h-002FFFFFh: 1K ОЗУ, блок 1;
- 00300000h-7FFFFFFFh: Локальная шина;
- 800000000h-0FFFFFFFFh: Глобальная шина.

ВНИМАНИЕ: Обращение к зарезервированной области памяти может привести к непредсказуемым результатам.

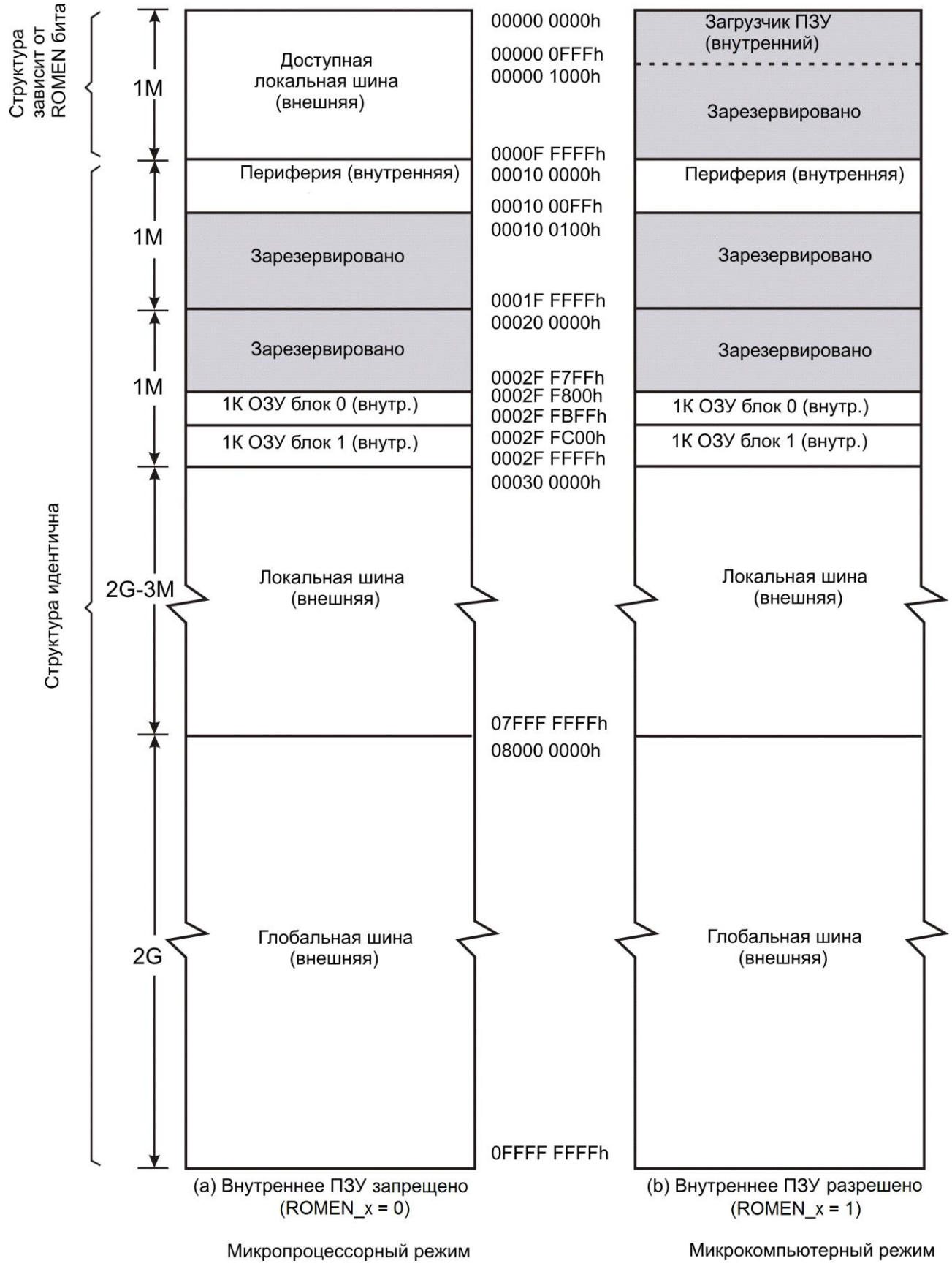


Рисунок 5.13 − Карта памяти ядра ПЦОС

5.4.2 Карта памяти периферийной шины

В таблице 5.10 представлена периферийная карта памяти ПЦОС.

Таблица 5.10 − Периферийная карта памяти

Адрес	Периферийное устройство
0010 0000h 0010 000Fh	Регистры управления локальной и глобальной шинами (16 слов)
0010 0010h 0010 001Fh	Регистры управления работой ПЦОС (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)

5.4.2.1 Регистры управления локальной и глобальной шинами

Эти регистры управляют работой глобальной и локальной шины. В карте памяти эти регистры занимают блок из 16 слов, на рисунке 5.14 показаны адреса этих регистров. Более подробное описание данных регистров представлено в подразделе 5.9.

Эти регистры могут определять следующие установки:

- размеры страниц подключаемой внешней памяти;
- размеры адресных блоков стробируемых разными стробами;
- состояния ожидания;
- другие сходные операции, которые формируют интерфейс памяти.

00100000h	Регистр управления интерфейсом глобальной памяти
0100001h – 00100003h	Зарезервировано
00100004h	Регистр управления интерфейсом локальной памяти
00100005h 0010 000Fh	Зарезервировано

Рисунок 5.14 – Регистры управления интерфейсом памяти

5.4.2.2 Регистры управления работой ПЦОС

Следующий 16-словный блок в карте памяти периферийной шины, как показано на рисунке 5.15, содержит часть регистров управления работой ПЦОС. Эти регистры зарезервированы для функций эмуляции.

5.4.2.3 Регистры таймера

Эта группа регистров занимает адресное пространство с 00100020h по 0010003Fh в карте памяти периферийной шины, и представлена на рисунке 5.15. Таймеры и их регистры детально рассматриваются в подразделе 5.13.

Та	00100020h	Регистр контроля таймера 0
	00100021h – 00100023h	Зарезервировано
	00100024h	Регистр счётчика таймера 0
	00100025h – 00100027h	Зарезервировано
	00100028h	Регистр периода таймера 1
	00100029h	Зарезервировано
Та	00100030h	Регистр контроля таймера 1
	00100031h – 00100033h	Зарезервировано
	00100034h	Регистр счётчика таймера 1
	00100035h – 00100037h	Зарезервировано
	00100038h	Регистр периода таймера 1
	0010003Fh	Зарезервировано

Рисунок 5.15 − Регистры таймера

5.4.2.4 Карта памяти коммуникационного порта

На рисунке 5.16 иллюстрировано расположение адресов регистров глобального управления коммуникационных портов CPCR, входных и выходных буферов FIFO, а также регистр сброса коммуникационных портов. Эти регистры более детально описаны в подразделе 5.12.

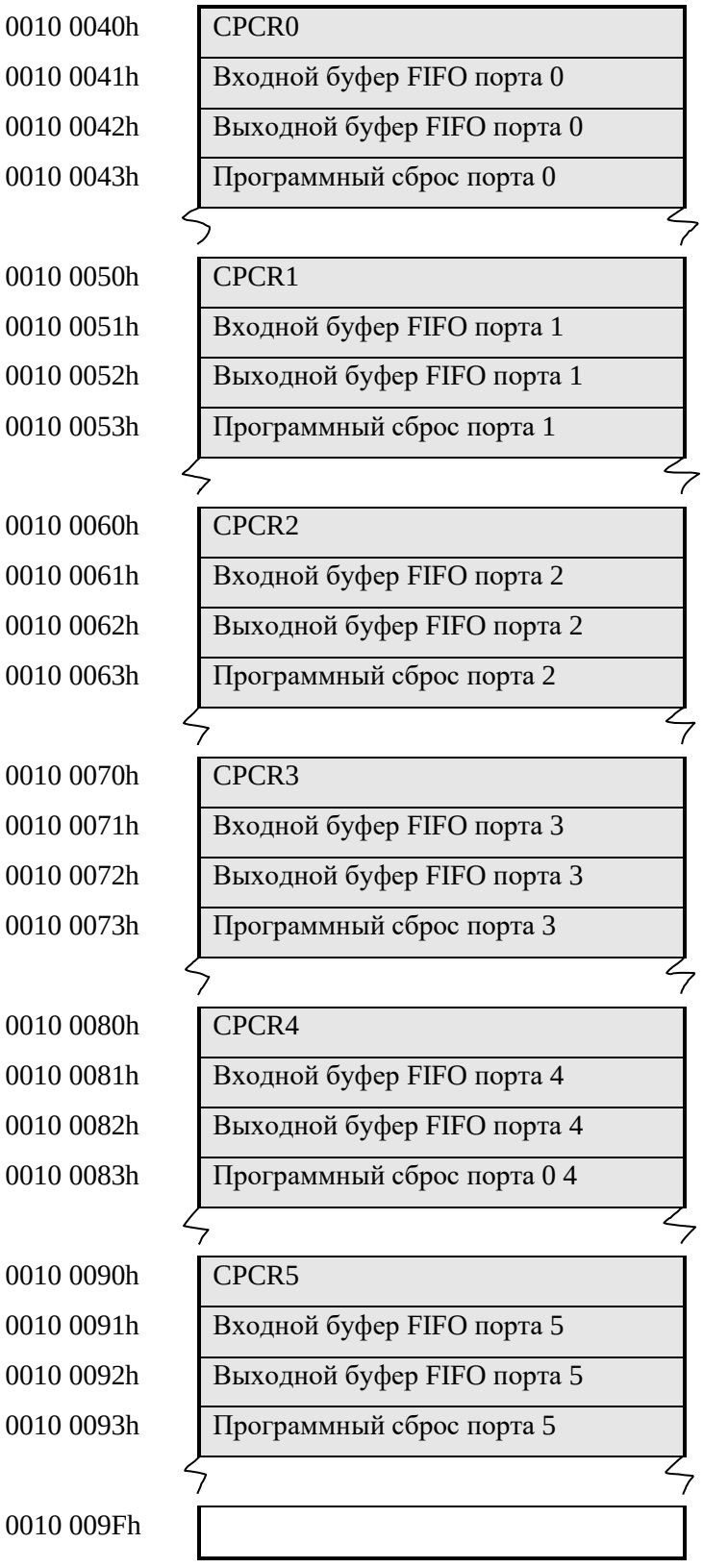


Рисунок 5.16 − Карта памяти коммуникационных портов

5.4.2.5 Регистры сопроцессора ПДП

ПДП регистры, показанные на рисунке 5.17, это нижний блок регистров в карте памяти периферийной шины. Эти регистры описаны в подразделе 5.11.

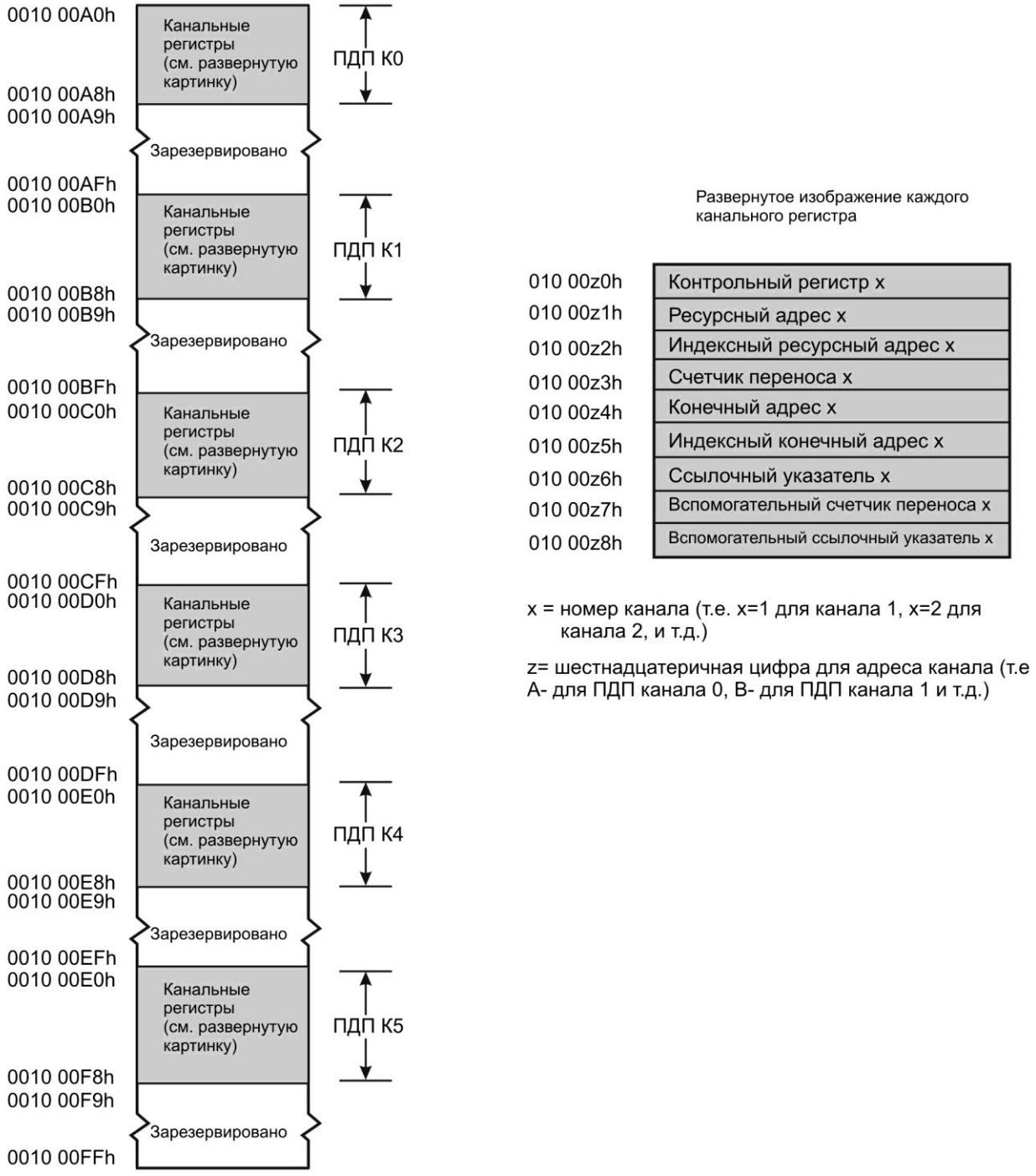


Рисунок 5.17 − Карта памяти сопроцессора ПДП

5.4.3 Кэш команд

128 × 32-разрядный кэш команд позволяет максимально ускорить выполнение программы при минимальных системных затратах, путём сохранения в кэш фрагментов программы, которые могут быть выбраны, когда необходим повторный доступ к этим фрагментам. Это значительно уменьшает число необходимых обращений к памяти вне кристалла и позволяет хранить программу вне кристалла в медленной недорогой памяти. Внешние шины в этом случае свободны от выборки программы, что может быть использовано ПДП и другими элементами системы.

Благодаря использованию алгоритма LRU (Least Recently Used), кэш может работать в полностью автоматическом режиме без вмешательства пользователя. LRU – это алгоритм, при котором из кэш вытесняются дольше всего незапрашиваемые фрагменты программы.

5.4.3.1 Архитектура кэш

Кэш команд, см. рисунок 5.19, включает 128 × 32-разрядных слов ОЗУ, достаточных для хранения 128 слов памяти программ. Он разделён на четыре сегмента по 32 слова. С каждым сегментом связан 27-разрядный регистр стартового адреса сегмента (SSA). Для каждого слова кэш есть соответствующий одноразрядный флаг присутствия P (Present).

Когда ЦПУ запрашивает командное слово из внешней памяти, производится проверка, не содержится ли слово в кэш команд. Разделение адреса команды, используемого алгоритмом управления кэш, показано на рисунке 5.18. 27 старших разрядов адреса команды используются для выбора сегмента, 5 младших разрядов определяют адрес слова команды внутри выбранного сегмента. 27 старших значащих разрядов адреса команды сравниваются с двумя регистрами начального адреса сегмента SSA. Если соответствие найдено, то проверяется флаг P. Флаг P показывает, присутствует ли уже или нет слово внутри соответствующего сегмента памяти кэш.

P = 1: слово уже представлено в памяти кэш.

P = 0: размещение в кэш неправильно (т. е. включает ненужные данные).

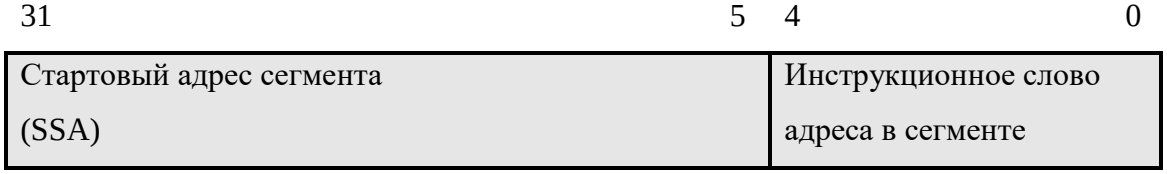


Рисунок 5.18 − Адресное разделение для контрольного алгоритма кэш

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

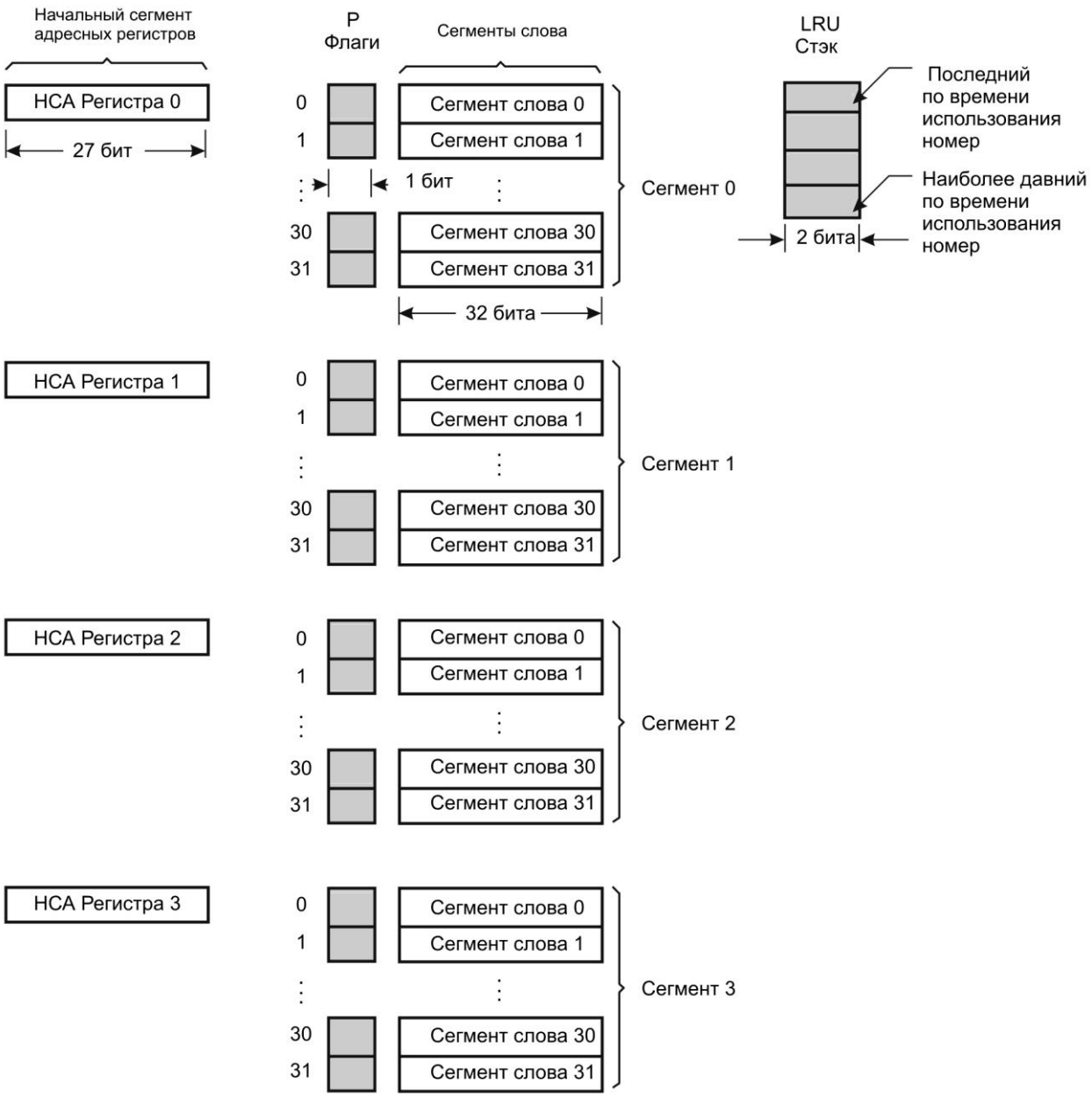


Рисунок 5.19 − Архитектура инструкционного кэш

Стек LRU после каждого доступа к кэш определяет, какой из двух сегментов квалифицируется как неиспользованный дольше; таким образом, стек содержит либо 0, 1, либо 1, 0. Каждый раз при доступе к сегменту, номер сегмента удаляется из стека LRU и вталкивается в вершину стека LRU. Таким образом, число в вершине стека представляет собой номер последнего использованного сегмента, а число на дне стека – номер неиспользованного дольше всех сегмента.

При сбросе системы стек LRU инициализируется с установкой 0 в вершине, с 1 на дне стека, а все P флаги в кэш команд очищены.

Когда необходима замена, для неё выбирается неиспользованный дольше всех сегмент. 32 флага P для заменяемого сегмента устанавливаются в 0, и в сегментный регистр SSA записываются 27 старших разрядов адреса команды.

5.4.3.2 Управляющие разряды кэш

Четыре управляющих бита кэш локализованы в статусном регистре ЦПУ ST: бит очистки кэш CC, бит активизации кэш CE, бит замораживания кэш CF и бит предыдущей заморозки кэш PCF. Статусный регистр показан на рисунке 5.8.

Бит очистки кэш CC. Установка CC = 1 делает недействительными все входы в кэш. Этот бит всегда является очищенным после записи, таким образом, он всегда читается как 0. При сбросе в тот бит записывается 0. Флаг кэш P = 0, когда кэш очищен.

Бит активизации кэш CE. Установка CE = 1 активизирует кэш, позволяя кэш быть использованным соответственно LRU (замещения наиболее давней по использованию страницы) алгоритму кэш. Установка CE = 0 останавливает работу кэш, это не позволяет кэш обновляться или изменяться (таким образом, может быть сделан непереносимый кэш). При сбросе этот бит читается как 0. Очистка кэш (CC = 1) позволена, когда CE = 0.

Бит заморозки кэш CF. Установка CF = 1 замораживает кэш, включая заморозку LRU (замещения наиболее давней по использованию страницы) стековой манипуляции. Если кэш активизирован CE = 1 и кэш заморожен CF = 1, переносы из кэш позволены, но не позволены изменения содержимого кэш. Очистка кэш CC = 1 позволена, когда CF = 1. При сбросе, этот бит очищается в 0 и после сброса устанавливается в 1. Когда CF = 0, очистка кэш позволена CC = 1. CF устанавливается к 1, когда прерывание или системное прерывание захвачено. Также, RETI и RETID инструкции копируют PCF в CF бит.

Таблица 5.11 определяет результат установки различных комбинаций, разрядов CE и CF.

Таблица 5.11 – Результат установки различных комбинаций разрядов CE и CF

CE	CF	Эффект
0	0	Кэш не активизирован
0	1	Кэш не активизирован
1	0	Кэш активизирован и не заморожен
1	1	Кэш активизирован и заморожен

Бит предварительной заморозки кэш PCF. Когда произошло прерывание, значение бита CF скопировано в бит PCF, и CF бит устанавливается в 1. Это очищает кэш в течение процесса прерывания, что особенно полезно, когда петли кода прерваны. Программа, обслуживающая прерывания, может быть опционально использована кэш под программным контролем. Прерывания могут также быть вложенными, обеспечивая сохранение значений статусного регистра перед прерываниями. Когда инструкции RETIcond и RETIcondD осуществляют процесс выхода из прерывания, содержимое PCF бита копируется в CF бит.

5.4.3.3 Алгоритм кэш

Когда ПЦОС запрашивает командное слово из внешней памяти, то возможны два случая: кэш-попадание (команда найдена в кэш) или кэш-отсутствие (команда в кэш не найдена):

- кэш-попадание. Требуемая команда содержится в кэш, и производятся следующие действия:

- командное слово считывается из кэш;

- номер сегмента, внутри которого содержится слово, удаляется из стека LRU и вталкивается на вершину стека LRU, таким образом, перемещая номер другого сегмента на дно стека;

- кэш-отсутствие. Команда не содержится в кэш.

Типы кэш отсутствия:

- слово не найдено. Регистр адреса сегмента сравнивает адрес команды, но подходящий флаг не установлен. Следующие действия производятся параллельно:

- командное слово считывается из памяти и копируется в кэш;

- номер сегмента, внутри которого содержится слово, удаляется из стека LRU и вталкивается в вершину стека LRU, таким образом, перемещая номер другого сегмента на дно стека;

- устанавливается соответствующий флаг P.

- сегмент не найден. Никакой адрес сегмента не соответствует адресу команды. Следующие действия происходят параллельно:

- наиболее редко используемый сегмент выбирается для замены. Флаги P для всех 32 слов очищены;

- регистр SSA для выбранного сегмента загружается 27 старшими разрядами адреса запрошенного командного слова;

- командное слово выбирается и копируется в кэш. Оно направляется в соответствующую позицию наиболее редко используемого сегмента. Флаг P для этого слова устанавливается в 1;

- номер сегмента, содержащего слово, удаляется из стека LRU и вталкивается в вершину стека LRU, таким образом, перемещая номер другого сегмента на дно стека.

Из кэш команд могут быть выбраны только команды. Все операции чтения/записи данных в память происходят без участия кэш. Выборка программы из внутренней памяти не изменяет кэш, и не будет генерировать состояние попадания или отсутствия кэш. Кэш команд – это блок памяти одиночного доступа. Пустая программная выборка (т. е. с последующим ветвлением) обрабатывается кэш, как действительная выборка программы, и может генерировать состояние попадания или отсутствия кэш.

Особое внимание необходимо при использовании самомодифицирующегося кода. Если команда находится в кэш и соответствующая область основной памяти изменена, то копия команды в кэш не изменяется.

Наиболее эффективное использование кэш может быть достигнуто при помощи выравнивания кода программы по границе адреса в 32 слова. Это может быть сделано, если использовать директиву *.align* при написании программ на языке ассемблера.

5.5 Форматы данных и операции с плавающей запятой

В процессорных ядрах ИС 1867ВЦ8Ф1 данные разделены на три основных типа: целые числа, целые числа без знака и числа в формате с плавающей запятой. Отметим, что термины целые и целые со знаком считаются эквивалентными. Процессорные ядра ИС 1867ВЦ8Ф1 поддерживают короткие и с одинарной точностью форматы для целых со знаком и без знака. Поддерживаются также короткие, с одинарной точностью и с повышенной точностью форматы для значений с плавающей запятой.

Операции с плавающей запятой обеспечивают удобные и безошибочные вычисления. Реализация арифметики с плавающей запятой в процессорных ядрах 1867ВЦ8Ф1 позволяет производить операции с плавающей запятой со скоростью целочисленных, 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 процессорным ядром ИС 1867ВЦ8Ф1.

5.5.1 Целочисленный формат со знаком

Ядра ИС 1867ВЦ8Ф1 поддерживают 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.

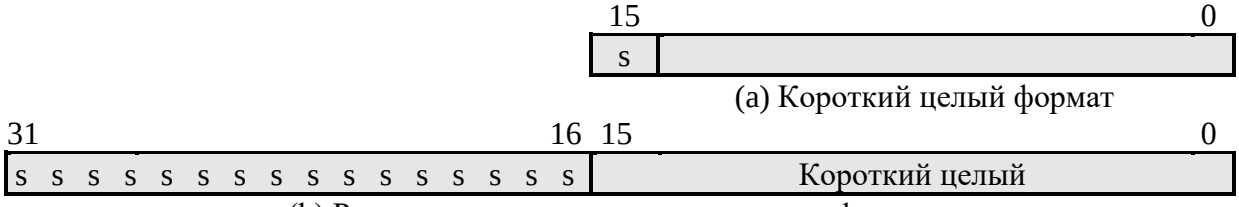
Примечание – При размещении целочисленных операндов в регистрах повышенной точности задействуются только биты 31 – 0; биты 39 – 32 не модифицируются и не используются.

5.5.1.1 Короткий целочисленный формат

16-разрядный формат коротких целых чисел в дополнительном коде используется для непосредственных целочисленных операндов. Для инструкций, содержащих целочисленные операнды, происходит расширение этого формата на знак до 32 разрядов (см. рисунок 5.20). Диапазон целого числа si, представленный в коротком целом формате:

$-2^{15} \le si \le 2^{15} - 1$

На рисунке 5.20 приведён короткий целый формат и расширение знака короткого целого.



s – знаковый разряд.

Рисунок 5.20 − Короткий целый формат и расширение знака короткого целого

5.5.1.2 Целый формат с одинарной точностью

В целом формате с одинарной точностью целые числа представлены в коде с дополнением до двух. Диапазон целого числа, представленного в целом формате с одинарной точностью: $-2^{31} \le sp \le 2^{31} - 1$. На рисунке 5.21 приведён целый формат с одинарной точностью.

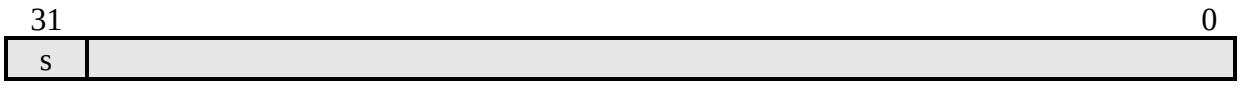


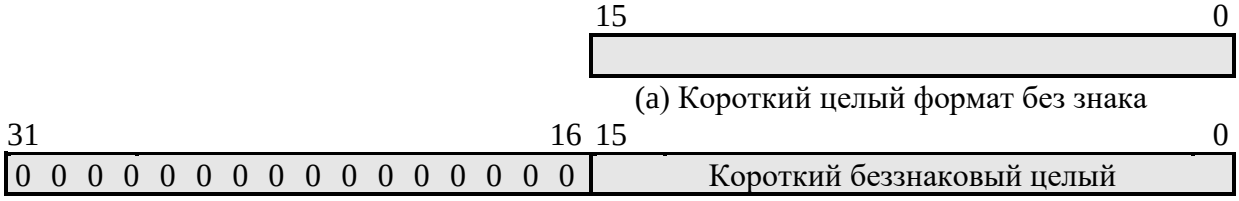
Рисунок 5.21 − Целый формат с одинарной точностью

5.5.2 Форматы целых без знака

Процессорные ядра ИС 1867ВЦ8Ф1 поддерживают два целочисленных формата без знака: 16-разрядный короткий формат и 32-разрядный формат с одинарной точностью. В регистрах повышенной точности в целочисленных беззнаковых операндах используются только разряды 31 – 0; разряды 39 – 32 не изменяются.

5.5.2.1 Короткий целочисленный формат без знака

На рисунке 5.22 приведён 16-разрядный короткий формат без знака, используемый для непосредственных целых беззнаковых операндов. В тех командах, которые содержат целые без знака операнды, этот формат заполняется нулями до 32 разрядов. Диапазон целого числа – $0 \le si \le 2^{16}$.



(a) Короткий целый формат без знака

(b) Заполнение нулями короткого целого формата без знака

Рисунок 5.22 − Короткий целочисленный формат без знака с заполнением нулями

5.5.2.2 Целочисленный формат с одинарной точностью без знака

В целочисленном формате с одинарной точностью без знака число представлено как 32-разрядное значение, как показано на рисунке 5.23. Диапазон целого числа – $0 \le sp \le 2^{32}$.

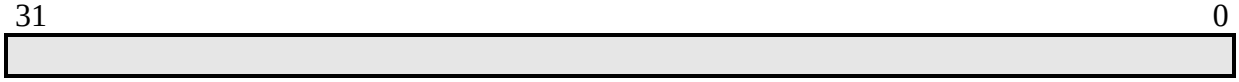


Рисунок 5.23 − Целочисленный формат с одинарной точностью без знака

5.5.3 Форматы с плавающей запятой

Процессорные ядра ИС 1867ВЦ8Ф1 поддерживают три формата с плавающей запятой:

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

Все форматы чисел с плавающей запятой в ядре процессора 1867ВЦ8Ф1 состоят из трёх полей: поля экспоненты (e), однобитного поля знакового разряда (s) и поля дробной части (f). Они представлены на рисунке 5.24. Поле знака и поле дробной части могут рассматриваться как единая часть и определяются как поле мантиссы (man). Каждый формат разделён на эти три поля, как показано на рисунке 5.24.

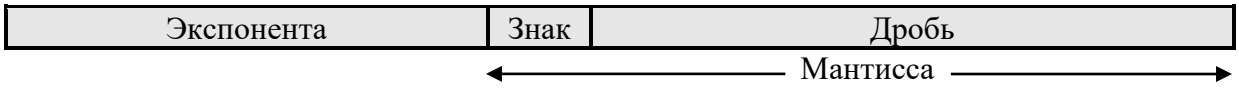


Рисунок 5.24 − Общий формат числа с плавающей запятой

Основная формула для вычисления значения числа с плавающей запятой приведена ниже. В выражении s – это значение одного бита, $\bar{s}$ – это инверсия значения одного бита, f – двоичное значение дробного поля и e – это десятичный эквивалент поля экспоненты.

Значение числа с плавающей запятой

$$x = s\bar{s}.f_2 \times 2^e \quad (5.1)$$

Мантисса представляется нормализованным, дважды дополненным числом. В нормализованном представлении старший незнаковый разряд является неявным, что обеспечивает дополнительный разряд точности. Неявный знаковый разряд используется как:

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

Если единичный бит s равен 0, мантисса становится $01.f_2$, где f – двоичное представление дробного поля. Если s – 1, мантисса становиться равной $10.f_2$, где f – двоичное представление дробного поля.

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

Поле экспоненты – это число в форме дополнения до двух. По существу, поле экспоненты сдвигает двоичную запятую в мантиссе. Если экспонента положительна, двоичная запятая сдвигается вправо. Если экспонента отрицательна, двоичная запятая сдвигается влево.

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

5.5.3.1 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, см. рисунок 5.25.

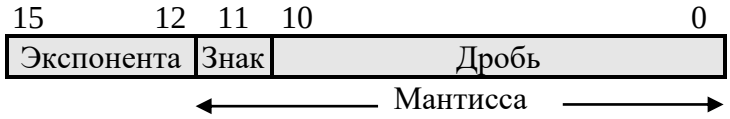


Рисунок 5.25 − Короткий формат числа с плавающей запятой

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

- e = − 8;
- s = 0;
- f = 0.

Операции выполняются с неявной двоичной запятой между 10 и 11 разрядами. 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: $x = (2 - 2^{-11}) \times 2^7 = 2.5594 \times 10^{-2}$;
- наименьшее положительное: $x = 1 \times 2^{-7} = 7.8125 \times 10^{-3}$;
- наибольшее отрицательное: $x = (-1 - 2^{-11}) \times 2^{-7} = -7.8163 \times 10^{-3}$;
- наименьшее отрицательное: $x = -2 \times 2^7 = -2.5600 \times 10^2$.

5.5.3.2 Формат числа с плавающей запятой с одинарной точностью

В формате с одинарной точностью число с плавающей запятой представлено 8-разрядным полем экспоненты (e) и 24-разрядным полем мантиссы (man) в дополнительном коде с неявным старшим незнаковым разрядом, см. рисунок 5.26.

Операции выполняются с неявной двоичной запятой между 23 и 22 разрядами. Когда неявный старший незнаковый разряд определён, 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.

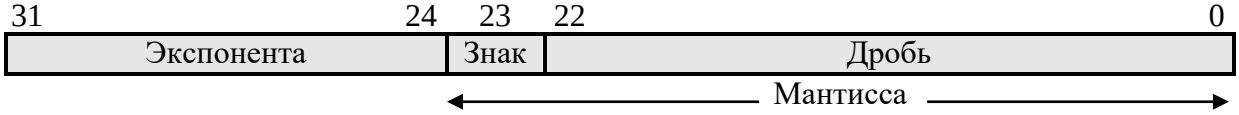


Рисунок 5.26 − Формат числа с плавающей запятой с одинарной точностью

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: $x = (2 - 2^{-23}) \times 2^{127} = 3.4028234 \times 10^{38}$;
- наименьшее положительное: $x = 1 \times 2^{-127} = 5.8774717 \times 10^{-39}$;
- наибольшее отрицательное: $x = (-1 - 2^{-23}) \times 2^{-127} = -5.8774724 \times 10^{-39}$;
- наименьшее отрицательное: $x = -2 \times 2^{127} = -3.4028236 \times 10^{38}$.

5.5.3.3 Формат числа с плавающей запятой с повышенной точностью

В формате с повышенной точностью число с плавающей запятой представляется 8-разрядным полем экспоненты (e) и 32-разрядным полем мантиссы (man) с неявным старшим незнаковым разрядом, см. рисунок 5.27.

Операции выполняются с неявной двоичной запятой между 31 и 30 разрядами. Когда неявный старший незнаковый разряд определён, он размещается непосредственно слева от двоичной запятой. 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.

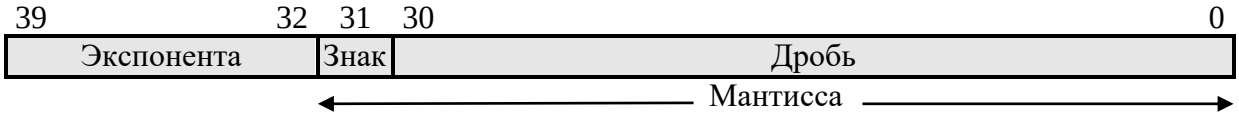


Рисунок 5.27 − Формат числа с плавающей запятой с повышенной точностью

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: $x = (2 - 2^{-31}) \times 2^{127} = 3.4028236683 \times 10^{38}$;
- наименьшее положительное: $x = 1 \times 2^{-127} = 5.8774717541 \times 10^{-39}$;
- наибольшее отрицательное: $x = (-1 - 2^{-31}) \times 2^{-127} = -5.8774717569 \times 10^{-39}$;
- наименьшее отрицательное: $x = -2 \times 2^{127} = -3.4028236691 \times 10^{38}$.

5.5.3.4 Определение десятичного эквивалента числа с плавающей запятой

Определение значения, сохранённого в формате с плавающей запятой, происходит в два этапа:

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

Этап 1. Определение значений экспоненты и мантиссы

Поле экспоненты – число в дополнительном коде, которое зависит от типа преобразуемого числа с плавающей запятой. Десятичный эквивалент экспоненты − e.

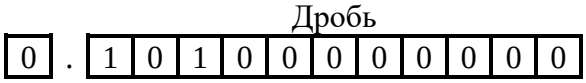
Например, если вы преобразовываете число с плавающей запятой с одинарной точностью и двоичное значение поля экспоненты равно 00000100, тогда десятичное значение экспоненты будет равно 4.

С другой стороны, если двоичное значение поля экспоненты равно 11111100_2, тогда десятичное значение экспоненты будет − 4. Так как первый бит слева равен 1, значит число отрицательное. Значение числа вычисляется путём прибавления 1 к дополнительному коду числа 11111100_2, который равен 00000011_2.

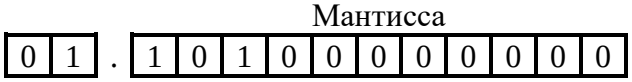
Примечание − Если значение экспоненты совпадает со значением, зарезервированным для нуля, число с плавающей запятой равно нулю. Резервное значение для каждого формата числа с плавающей запятой приведено с типовым описанием в 5.5.3.

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

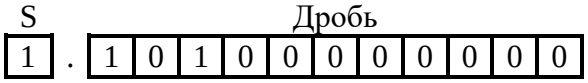
- если s = 0, мантисса формируется посредством записи 01. и дополнением разрядов в поле дробной части после двоичной запятой. Например, если f = 10100000000_2, тогда man = 01.1010000000_2:



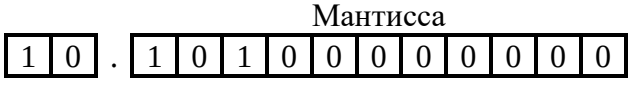
Перезаписывается мантисса как:



- если s = 1, 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:



Перезаписывается мантисса как:



Этап 2. Смещение двоичной запятой в мантиссе в соответствии со значением экспоненты и дальнейшее преобразование числа в десятичное

Если экспонента (e) имеет положительное значение, двоичная запятая e смещается вправо.

Если экспонента (e) имеет отрицательное значение, двоичная запятая e смещается влево.

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

Если e = − 2_{10} и man = 01.10000000000_2, тогда изменённая мантисса становится равной 0.0110000000000_2, что эквивалентно 3/8 в десятичной системе счисления.

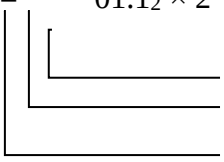
Следующий пример иллюстрирует, как вы можете получить эквивалент числа с плавающей запятой. В каждом примере используется формат числа с плавающей запятой с одинарной точностью.

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

Экспонента = $0000\ 0010_2 = 2$
Знак = 0
Дробь = $.10000_2$

Значение = $01.1_2 \times 2^2 = 0110_2. = 6$

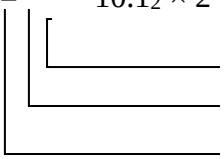


Пример 5.2 – Отрицательное число

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 = $.10000_2$

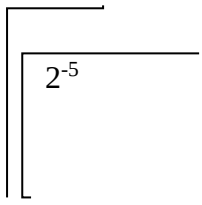
Значение = $10.1_2 \times 2^1 = 101_2. = -3$



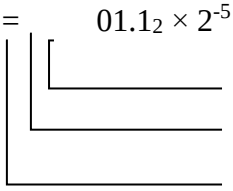
Пример 5.3 – Дробное число

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

Экспонента = $1111\ 1001_2 = -5$
Знак = 0 2^{-6}
Дробь = $.10000_2$

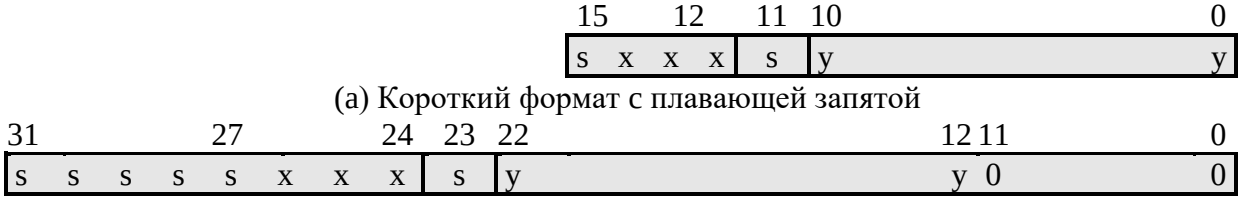


Значение = $01.1_2 \times 2^{-5} = 101_2 = .000011_2 = 3/64$
Дробь
Основание
Знак



5.5.3.5 Преобразование между форматами с плавающей запятой

Операции с плавающей запятой предполагают различные форматы для ввода и вывода. Эти форматы часто требуют преобразования из одного формата с плавающей запятой в другой (т. е. 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 5.28 – 5.31 (s − знаковый разряд экспоненты). Когда число в формате с плавающей запятой нулевого значения преобразуется в формат с большей точностью, оно всегда преобразуется в действительное представление нуля в данном формате.

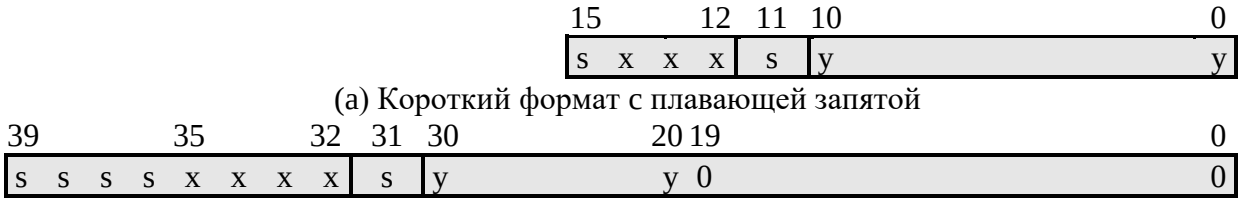


(a) Короткий формат с плавающей запятой

(b) Формат с плавающей запятой одинарной точности

Рисунок 5.28 − Преобразование короткого формата с плавающей запятой в формат с плавающей запятой одинарной точности

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

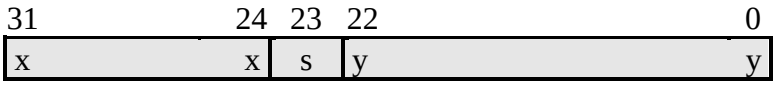


(a) Короткий формат с плавающей запятой

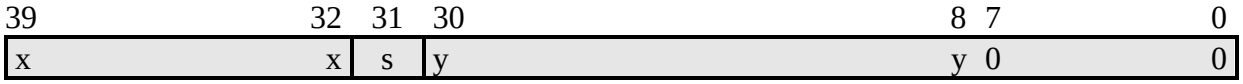
(b) Формат повышенной точности с плавающей запятой

Рисунок 5.29 − Преобразование короткого формата с плавающей запятой в формат с плавающей запятой повышенной точности

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



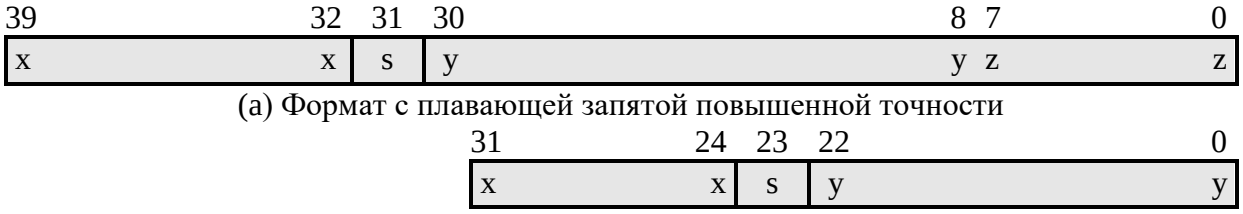
(a) Формат с плавающей запятой одинарной точности



(b) Формат повышенной точности с плавающей запятой

Рисунок 5.30 − 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.



(a) Формат с плавающей запятой повышенной точности

(b) Формат с плавающей запятой одинарной точности

Рисунок 5.31 − Преобразование формата с плавающей запятой повышенной точности в формат с плавающей запятой одинарной точности

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

5.5.4 Преобразование плавающей запятой IEEE Std. 754

Формат с плавающей запятой ядра процессора 1867ВЦ8Ф1 не совместим с форматом IEEE Std. 754. Однако ядро процессора 1867ВЦ8Ф1 поддерживает инструкции для преобразования в IEEE формат и обратно (TOIEEE и FRIEEE, соответственно). Процесс преобразования объясняется в 5.5.4.1 и 5.5.4.2. На рисунке 5.32 показан IEEE формат с плавающей запятой, а на рисунке 5.33 показан формат с плавающей запятой ядра процессора 1867ВЦ8Ф1.

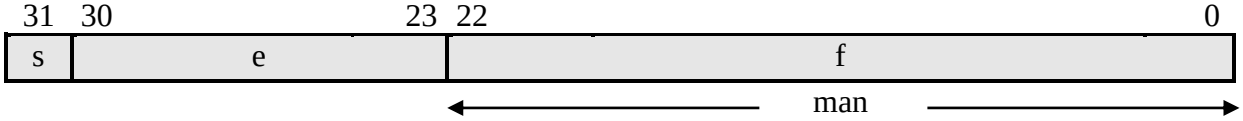


Рисунок 5.32 – IEEE Std.754 формат с плавающей запятой одинарной точности

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

1) если e = 255 и f ≠ 0, тогда v = NaN;
2) если e = 255 и f = 0, тогда $v = (-1)^s$ infinite;
3) если 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 $v = (-1)^s \times 0$ (ноль),

где s = знаковый разряд, e = поле экспоненты, f = поле дроби, NaN = не число.

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

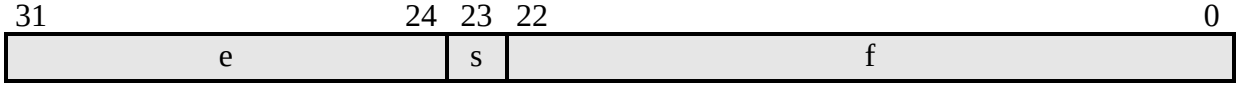


Рисунок 5.33 – Формат с плавающей запятой с одинарной точностью в дополнительном коде ядра процессора 1867ВЦ8Ф1

Рисунок 5.33 показывает формат с плавающей запятой в дополнительном коде ядра процессора 1867ВЦ8Ф1. 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 дроби.

При этом e интерпретируется как целое в дополнительном коде. Дробь и знаковый разряд формируют нормализованную мантиссу в дополнительном коде.

Примечание – Необходимо различать символы для IEEE и форматов ядра процессора 1867ВЦ8Ф1.

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 дополнительном коде индексируются two (т. е. e_{two}, s_{two}, f_{two}).

5.5.4.1 Преобразование IEEE формата в формат числа с плавающей запятой в дополнительном коде ядра процессора 1867ВЦ8Ф1

Преобразование IEEE в дополнительный код является наиболее общим. Это преобразование выполняется в соответствии с правилами, указанными в таблице 5.12.

Таблица 5.12 − Преобразование IEEE формата в формат с плавающей запятой в дополнительном коде

Случай	Если эти значения действительные			Если эти значения одинаковые			
	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 отображает преобразование IEEE положительных NaN и положительной бесконечности в наибольшее положительное число с одинарной точностью в дополнительном коде. При этом появляется сигнал переполнения, что позволяет проверять эти специальные случаи.

Случай 2 отображает преобразование IEEE отрицательных NaN и отрицательной бесконечности в наименьшее отрицательное число с одинарной точностью в дополнительном коде. При этом появляется сигнал переполнения, что позволяет проверять эти специальные случаи.

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

Случай 4 отображает преобразование IEEE отрицательных нормализованных чисел с ненулевой дробью в эквивалентные отрицательные значения в дополнительном коде.

Случай 5 отображает преобразование IEEE отрицательных нормализованных чисел с нулевой дробью в эквивалентные отрицательные значения в дополнительном коде.

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

Ядро процессора 1867ВЦ8Ф1 предполагает, что IEEE числа хранятся как целые в памяти или в регистре. Когда ядро процессора 1867ВЦ8Ф1 преобразовывает IEEE число, он помещает число в регистр повышенной точности, используя поля экспоненты и дроби регистра. 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. 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 IEEE в формат ядра процессора 1867ВЦ8Ф1 при передаче данных через блок памяти

```
        * Преобразование IEEE в формат ядра процессора 1867ВЦ8Ф1 при передаче данных через
блок памяти
        *
        * Программа предполагает, что входной буфер FIFO коммуникационного порта 0
        * Заполнен данными формата IEEE. Восемь слов
        * Передаются из коммуникационного порта 0 в блок 0 внутреннего ОЗУ
        * Формат данных преобразуется из IEEE формата
        * В формат ядра процессора 1867ВЦ8Ф1 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
        .
        .
        .
```

5.5.4.2 Преобразование числа формата с плавающей запятой ядра процессора 1867ВЦ8Ф1 в дополнительном коде в IEEE формат

Это преобразование происходит, как показано в таблице 5.13.

Таблица 5.13 − Преобразование числа формата с плавающей запятой ядра процессора 1867ВЦ8Ф1 в дополнительном коде в IEEE формат

Случай	Если эти значения существуют			Тогда эти значения равны		
	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	$\neq 0$	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 отображает преобразование нуля в дополнительном коде в положительный ноль IEEE.

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

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

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

Случай 5 отображает 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 отображает преобразование наименьшего отрицательного числа в дополнительном коде в отрицательную бесконечность IEEE.

Ядро процессора 1867ВЦ8Ф1 предполагает, что числа в дополнительном коде хранятся в памяти или в регистре повышенной точности в поле экспоненты или дроби регистра (как показано на рисунке 5.33). Если число расположено в регистре повышенной точности, только 24 старших разряда дробной части оперируются как поле дроби и для определения 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 формата процессора 1867ВЦ8Ф1 в IEEE при передаче данных через блок памяти

```
* Преобразование формата процессора 1867ВЦ8Ф1 в IEEE при передаче данных через блок памяти
*
* программа предполагает, что выходной буфер FIFO коммуникационного порта 0 пуст.
* Восемь слов передаются из блока 0 внутреннего ОЗУ в коммуникационный порт 0.
* Формат данных преобразуется из формата ядра процессора 1867ВЦ8Ф1 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
.
.
.
```

5.5.5 Умножение чисел с плавающей запятой

Число с плавающей запятой может быть записано в формате с плавающей запятой согласно формуле:

$$a = a(man) \times 2^{a(exp)}, \quad (5.2)$$

где a(man) – мантисса, a(exp) – экспонента.

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

$$c = a \times b = a(man) \times b(man) \times 2^{(a(exp) + b(exp))} \quad (5.3)$$

$$c(man) = a(man) \times b(man) \quad (5.4)$$

$$c(exp) = a(exp) + b(exp) \quad (5.5)$$

Когда выполняется умножение с плавающей запятой, 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, они преобразуются в формат с плавающей запятой повышенной точности. Эти преобразования производятся автоматически аппаратно, без дополнительных затрат. Все результаты умножений с плавающей запятой предполагают формат с повышенной точностью. Эти умножения происходят в одном цикле.

Блок-схема для умножения с плавающей запятой показана на рисунке 5.34. На этапе 1 32-разрядные мантиссы операндов источника умножаются 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 с 3 по 6 используются для проверки специальных случаев. На этапе 3 проверяется на равенство нулю мантисса 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) сдвигается вправо на два разряда и к c(exp) добавляется два. Этап 6 происходит, 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.

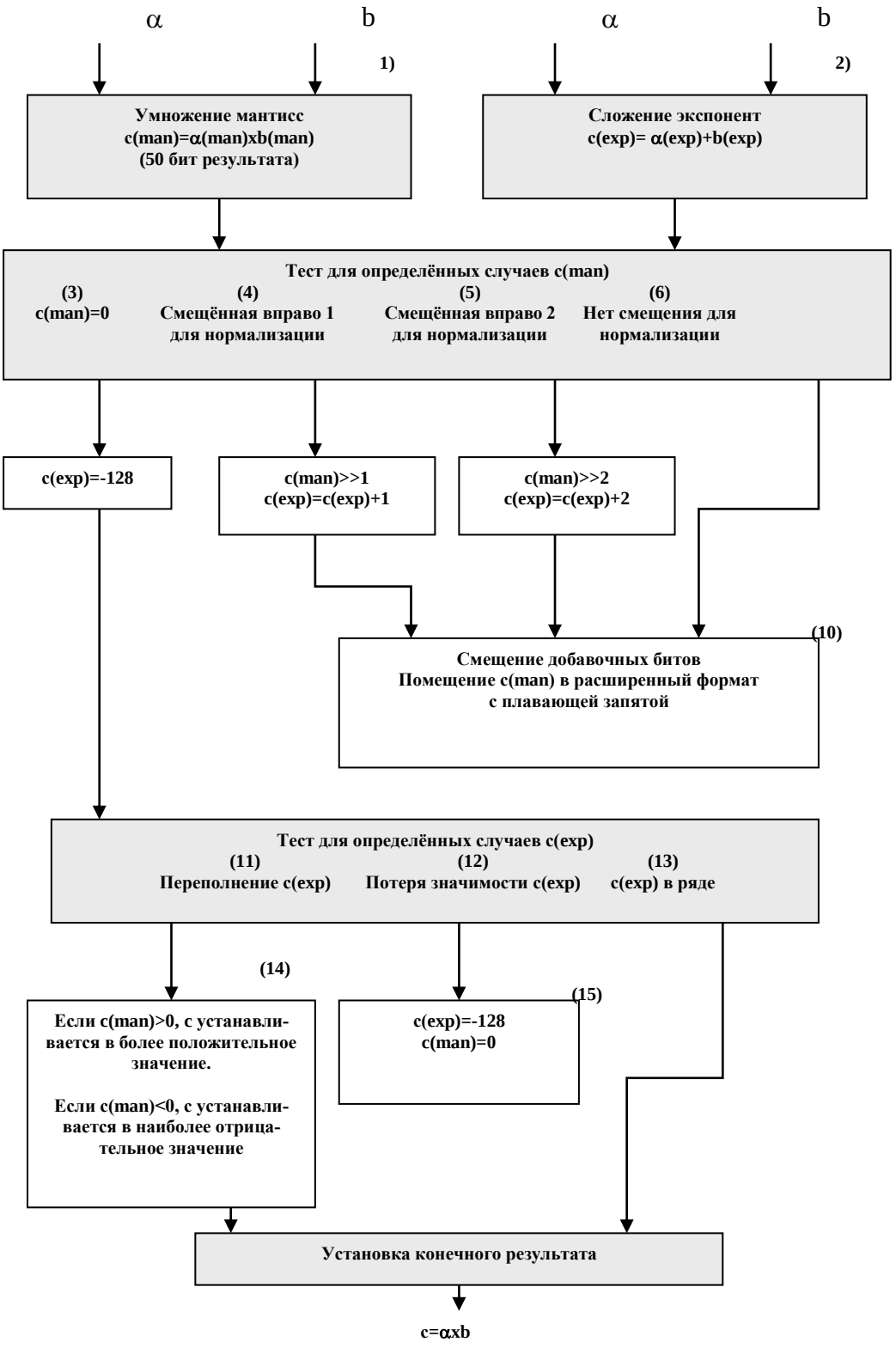


Рисунок 5.34 − Блок-схема для умножения с плавающей запятой

Следующие примеры иллюстрируют, как выполняется умножение с плавающей запятой в ядре процессора 1867ВЦ8Ф1. 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:

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

$$0100.00000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp))}$$

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

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

$$01.00000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp)+2)}$$

При умножении с плавающей запятой экспонента результата может переполняться. Это может произойти, когда экспонента 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:

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

$$0010.00000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp))}$$

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

$$01.10000000000000000000000 \times 2\ a(exp)$$
$$\times\ 01.10000000000000000000000 \times 2\ b(exp)$$

$$01.00100000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp)+1)}$$

Пример 5.8 – Умножение с плавающей запятой (обе мантиссы равны 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:

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

$$0001.00000000000000000000000000000000000000000000 * 2^{(a(exp)+b(exp))}$$

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

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:

$a = 1.0 \times 2^{a(exp)} = 01.0000000000000000000000 \times 2^{a(exp)}$

$b = 2.0 \times 2^{b(exp)} = 10.0000000000000000000000 \times 2^{b(exp)}$

Тогда:

$$\begin{array}{l} \quad 01.0000000000000000000000 \times 2^{a(exp)} \\ \times \quad 10.0000000000000000000000 \times 2^{b(exp)} \\ \hline 1110.0000000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp))} \end{array}$$

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

Умножение с плавающей запятой на ноль.

Любое умножение с плавающей запятой на ноль даёт ноль ($f = 0$, $s = 0$, $exp = -128$).

5.5.6 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:

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

Сумма или разность a и b должна быть определена как:

$$c = a \pm b \tag{5.6}$$
$$= (a(man) \pm (b(man) \times 2^{-(a(exp) - b(exp))})) \times 2^{a(exp)}, \tag{5.7}$$

если $a(exp) \geq b(exp)$

$$= ((a(man) \times 2^{-(b(exp) - a(exp))}) \pm b(man)) \times 2^{b(exp)}, \tag{5.8}$$

если $a(exp) < b(exp)$

Блок-схема для сложения с плавающей запятой приведена на рисунке 5.35. Так как эта блок-схема предполагает данные со знаком, она так же подходит для вычитания с плавающей запятой. Для этого примера предполагается, что $a(exp) \leq b(exp)$. На этапе 1 исходные экспоненты 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 уравнены, 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.

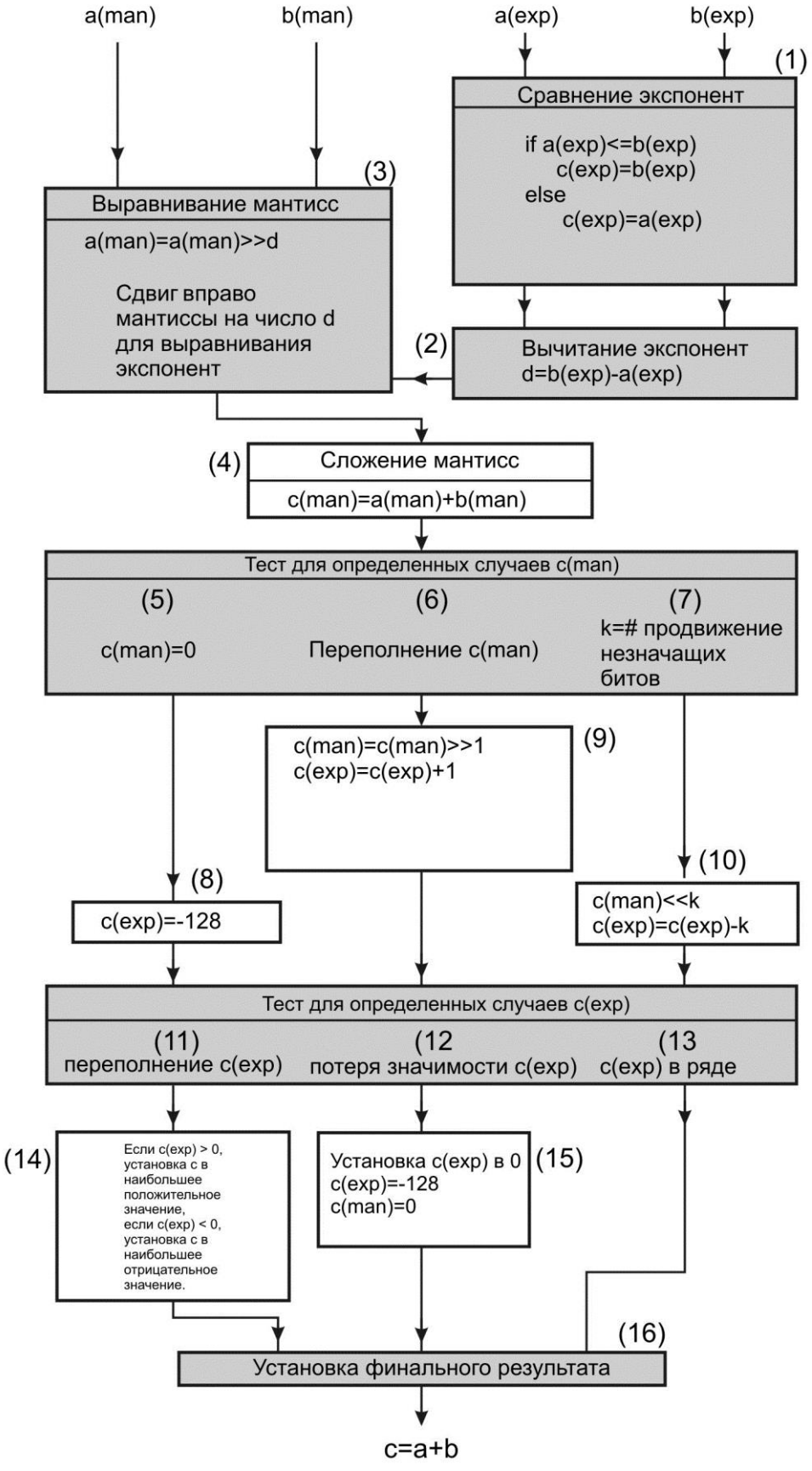


Рисунок 5.35 – Блок-схема для сложения с плавающей запятой

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

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

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

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

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 выровнены, так как два числа имеют одинаковые экспоненты. В результате происходит большая потеря точности старших разрядов, как показано ниже:

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

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

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

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 a + b.

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

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 нулём

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

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

$0 \pm 0 = 0$

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

5.5.7 Нормализация − NORM инструкция

Инструкция NORM берёт число с плавающей запятой с повышенной точностью, если оно не нормализовано, и нормализует его. Так как в предположении число не нормализовано, также предполагается не пустым старший незнаковый разряд. Инструкция NORM выполняет следующие три этапа:

- размещает старший незнаковый разряд числа с плавающей запятой;
- сдвигает влево число для нормализации;
- устанавливает экспоненту.

Данное значение a с плавающей запятой с повышенной точностью не нормализовано, нормализация показана на рисунке 5.36.

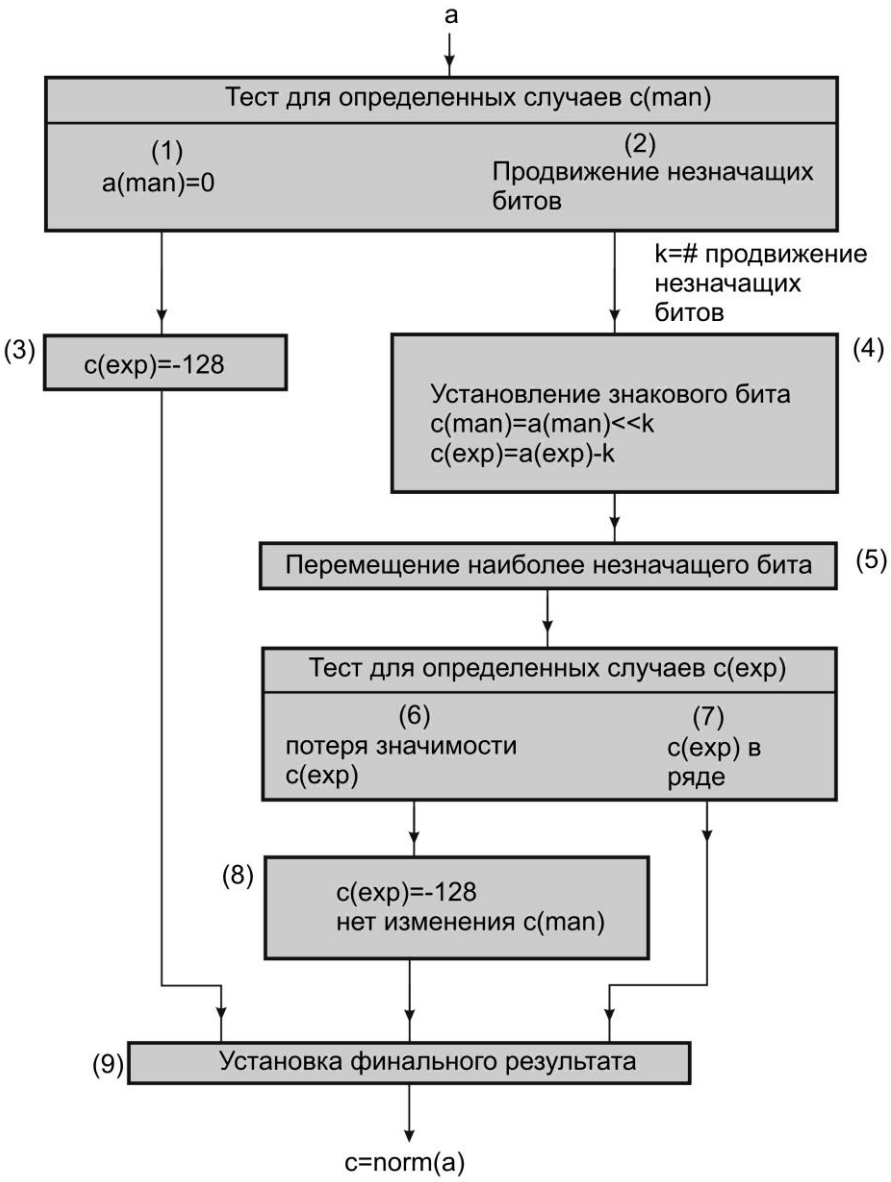


Рисунок 5.36 – Блок-схема выполнения инструкции NORM

Пример 5.14 – NORM инструкция

Предположим, что регистр повышенной точности содержит значение:

man = 00000000000000000001000000000001, exp = 0

Когда выполняется нормализация числа (предполагается что число не нормализовано), то подразумевается, что с двоичной запятой число имеет вид:

man = 0.00000000000000000001000000000001, exp = 0

Знак этого числа расширен на один на один разряд, так что мантисса содержит 33 разряда.

man = 00.00000000000000000001000000000001, exp = 0

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

man = 01.00000000001000000000000000000000, exp = – 19

После удаления резервного разряда формируется окончательное 32-разрядное значение нормализованного числа с плавающей запятой:

man = 00000000001000000000000000000000, exp = – 19

Инструкция NORM применяется для подсчёта количества начальных нулей или начальных единиц в 32-разрядном поле. Если начальное значение экспоненты равно нулю, абсолютное значение конечного результата экспоненты является количеством начальных единиц или нулей. Эта инструкция также используется для обработки ненормализованных чисел с плавающей запятой.

5.5.8 Округление − RND инструкция

RND инструкция округляет числа из формата с плавающей запятой с повышенной точностью в формат с плавающей запятой с одинарной точностью. Операция округления аналогична сложению с плавающей запятой. При округлении данного числа a сначала выполняется следующая операция:

$$c = a(man) \times 2^{a(exp)} + (1 \times 2^{(a(exp)-24)}) \qquad (5.9)$$

Затем выполняется преобразование из формата с плавающей запятой с повышенной точностью в формат с плавающей запятой с одинарной точностью. Округление данного числа с плавающей запятой с повышенной точностью показано на рисунке 5.37.

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).

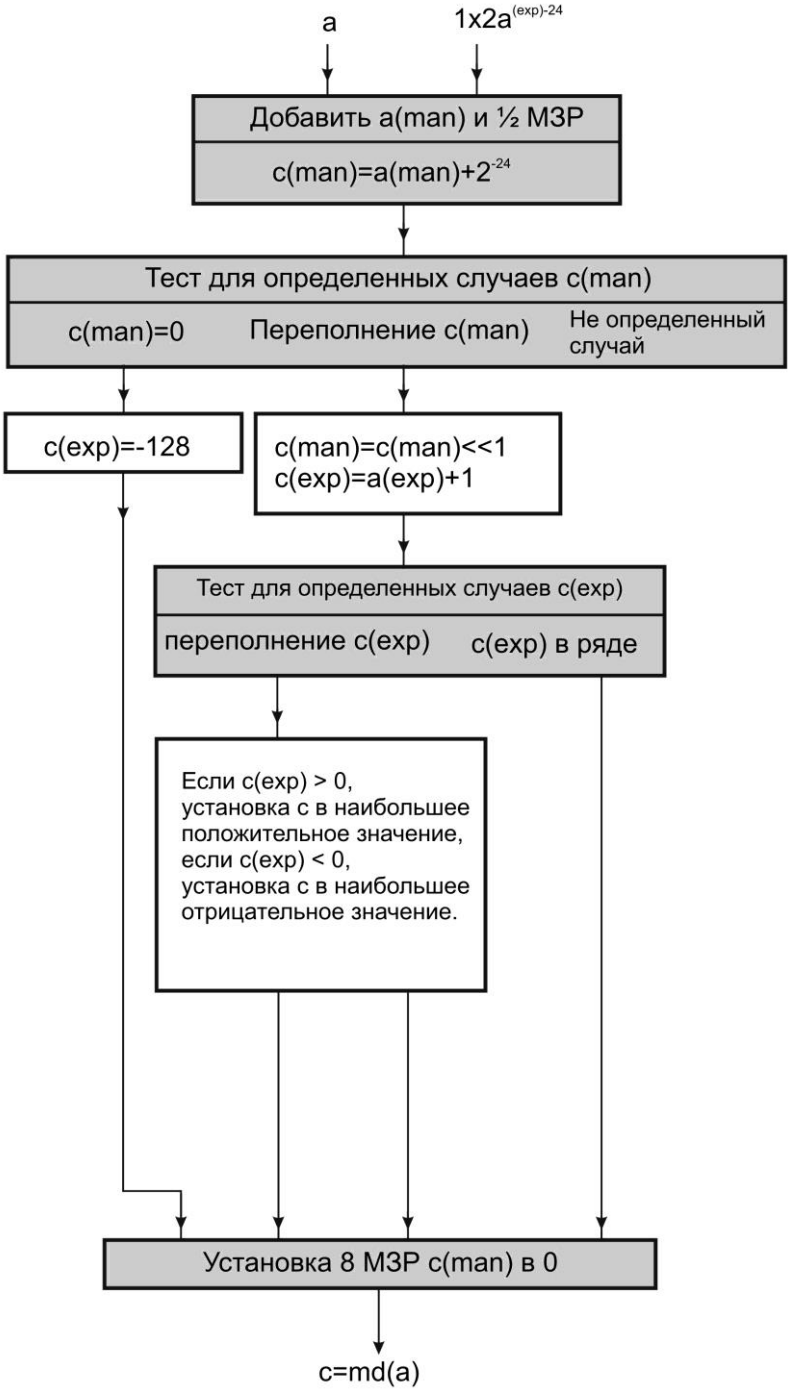


Рисунок 5.37 − Блок-схема для округления числа с плавающей запятой с помощью инструкции RND

5.5.9 Преобразование числа с плавающей запятой в целое − FIX инструкция

Преобразование значения с плавающей запятой в целое, используя инструкцию FIX, позволяет выполнять преобразование чисел в формате с плавающей запятой с одинарной точностью в целые с одинарной точностью в одном цикле. Преобразование значения с плавающей запятой x в целое будет обозначаться fix(x). Преобразование не приведёт к переполнению, если преобразуемое число a находится в диапазоне:

$-2^{31} \le a \le 2^{31} - 1$

Сначала необходимо определить, что:

$a(exp) \leq 30$

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

$rs = 31 - a(exp)$

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

Если $0 \leq x < 1$, тогда $fix(x) = 0$.

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

Блок-схема для преобразования значения с плавающей запятой в целое показана на рисунке 5.38.

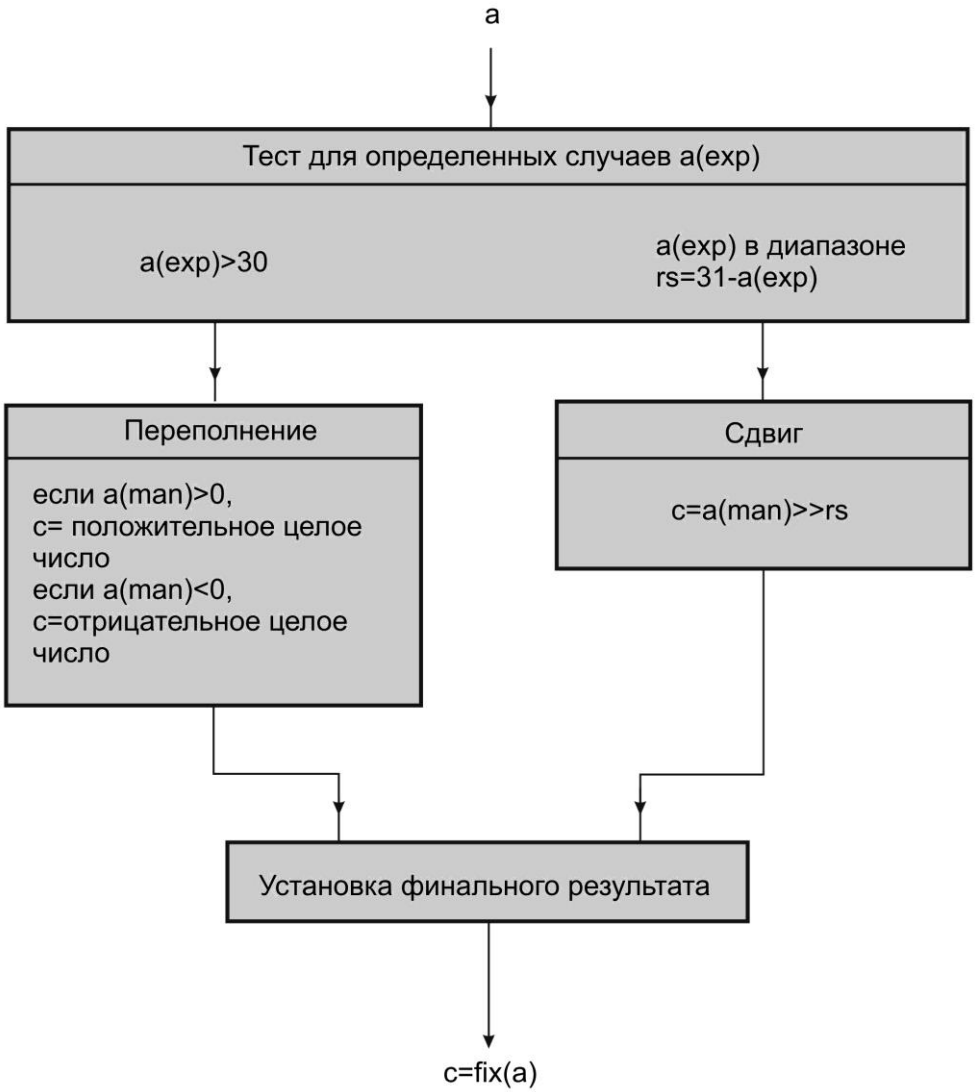


Рисунок 5.38 − Блок-схема преобразования значения с плавающей запятой в целое с помощью инструкции FIX

5.5.10 Преобразование целого числа в число с плавающей запятой − FLOAT инструкция

Преобразование целого значения в значение с плавающей запятой, используя инструкцию FLOAT, позволяет осуществить преобразование целого числа с одинарной точностью в число в формате с плавающей запятой повышенной точности. Блок-схема этого преобразования приведена на рисунке 5.39.

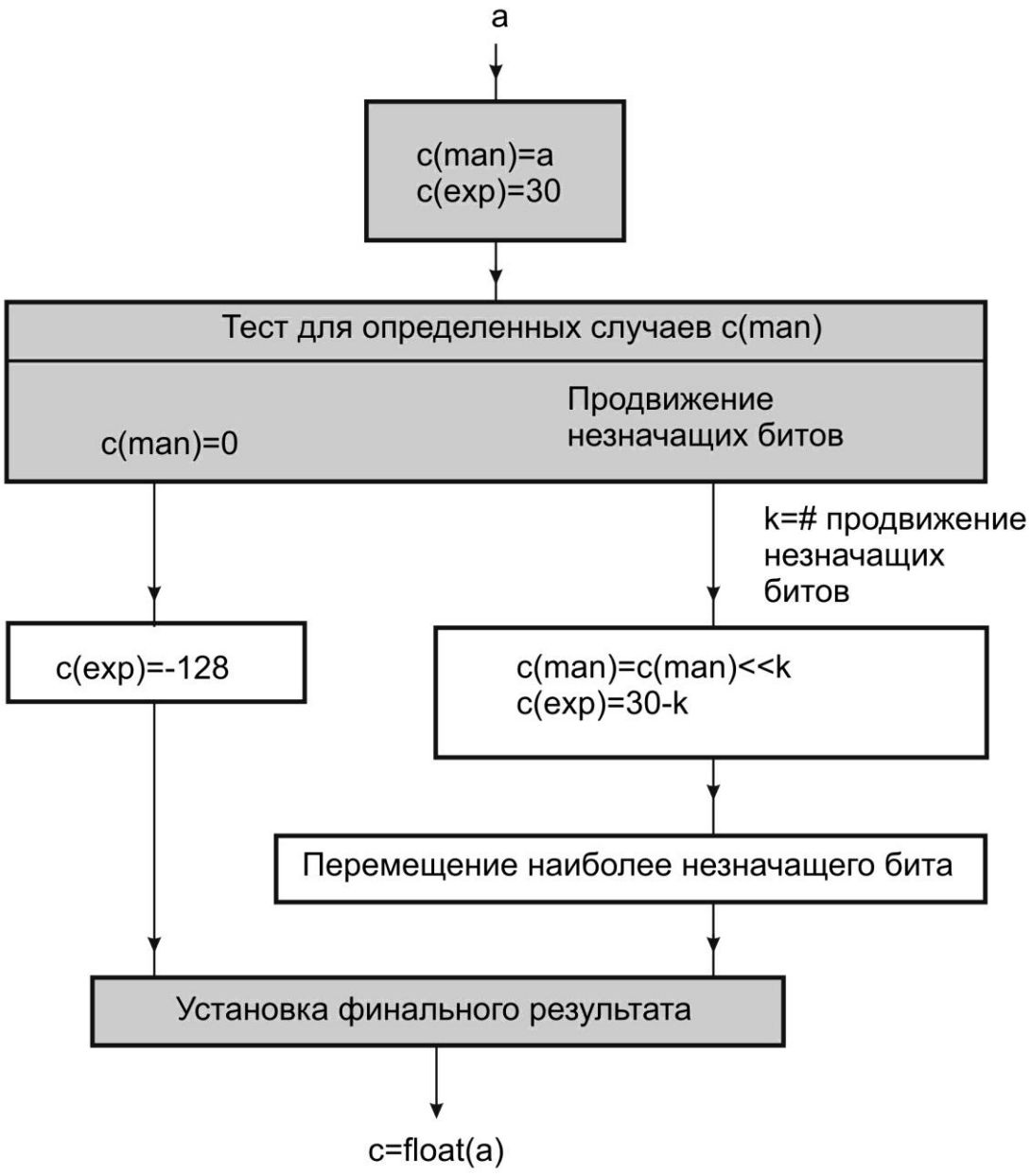


Рисунок 5.39 − Блок-схема преобразования целого числа в число с плавающей запятой с использованием инструкции FLOAT

5.5.11 Обратная величина − RCPF инструкция

Инструкция RCPF генерирует за один цикл приемлемую оценку обратной величины числа с плавающей запятой. Эта оценка имеет правильную экспоненту, а мантисса имеет точность до восьмой двоичной позиции (т. е. погрешность мантиссы составляет $< 2^{-8}$), что даёт 16-разрядное представление результата (восемь разрядов экспоненты плюс восемь разрядов мантиссы). Кроме того, эту оценку можно использовать в качестве начального числа для алгоритма вычисления обратной величины с ещё большей точностью (один из таких случаев – алгоритм Ньютона-Рафсона, описываемый в разделе 5.5.12.1).

На рисунке 5.40 приведён алгоритм, используемый инструкцией RCPF:

- предполагается, что на входные данные равны $v = vman \times 2^{vexp}$;
- предполагается, что на выходе будет $x = xman \times 2^{xexp}$;
- vexp инвертируется;
- если vexp = −128, то результат насыщается до наибольшего положительного числа, и устанавливается флаг переполнения. Флаг условия N имеет тот же знак, что и vsign.

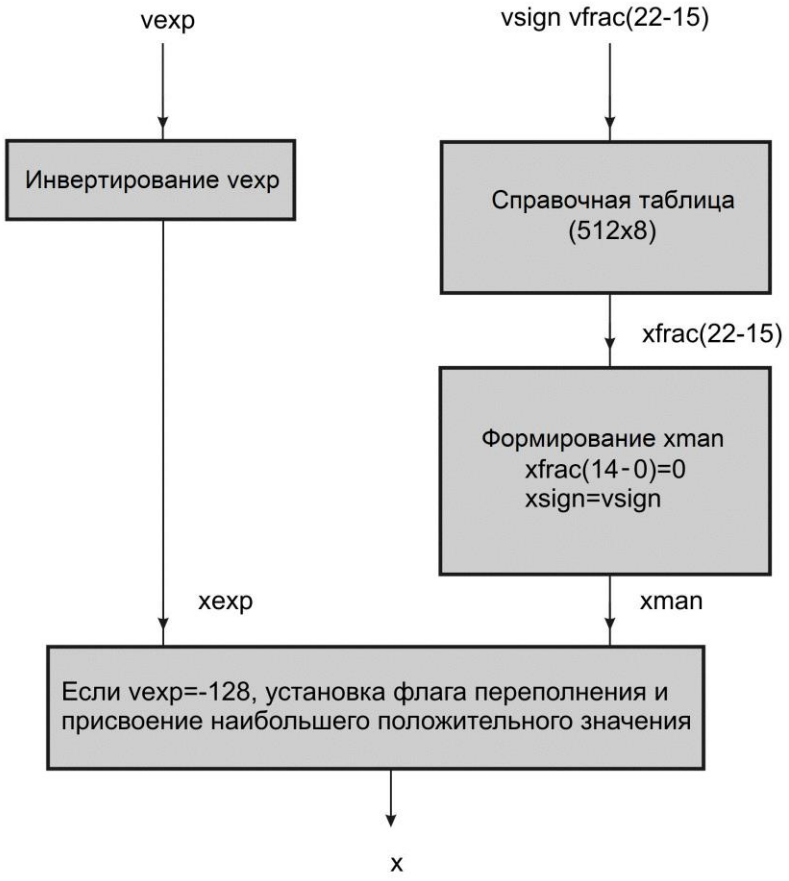


Рисунок 5.40 – Алгоритм использования инструкции RCPF

Таблица значений читается посредством формирования 9-разрядного адреса, конкатенируемого из vsign и разрядов 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 на основе результатов моделирования.

5.5.11.1 Алгоритм получения обратной величины

RCPF инструкция обеспечивает получение обратного значения числа.

Примерная оценка имеет правильную экспоненту, а мантисса имеет точность до восьмой двоичной позиции (т. е. погрешность мантиссы составляет $< 2^{-8}$). Алгоритм Ньютона-Рафсона (показан внизу) может использоваться для дальнейшего повышения точности мантиссы:

$$x[n+1] = x[n](2 - vx[n]), \quad (5.10)$$

где v = число, обратное значение которого ищется;

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

Программа ядра процессора 1867ВЦ8Ф1 для реализации этого алгоритма приведена в примере 5.15. Каждый шаг алгоритма 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; завершение первой итерации (16-разрядная точность)
;
MPYF   R1, R0, R2
SUBRF 2.0, R2
MPYF   R2, R1; завершение второй итерации (32-разрядная точность)
;
;          ; R1 = 1/v
;
```

5.5.12 Обратная величина квадратного корня − RSQRF инструкция

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, путём деления начального вектора на его длину. При этом производится деление на квадратный корень. Инструкция RSQRF даёт простую возможность непосредственного определения этой величины вместо выполнения двухступенчатого приближения отыскания квадратного корня и последующего нахождения обратной величины квадратного корня.

В результате выполнения этого алгоритма квадратный корень находится простым умножением:

$$v=vx[n], \quad (5.11)$$

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

RSQRF инструкция генерирует приемлемую приблизительную оценку обратной величины квадратного корня числа с плавающей запятой в одном цикле. При этом некоторые операционные характеристики инструкции RCPF проходят параллельно:

- RSQRF генерирует оценку (в этом случае, значение обратной величины квадратного корня числа с плавающей запятой);
- мантисса правильная в восьми двоичных позициях (погрешность мантиссы $< 2^{-8}$);

- зачастую это достаточная оценка обратной величины квадратного корня числа; в остальных случаях, она может быть использована как начальное значение для алгоритма вычисления обратной величины квадратного корня с большей точностью.

На рисунке 5.41 изображён алгоритм инструкции RSQRF. В алгоритме:

- предполагается, что на входные данные равны $v = vman \times 2^{vexp}$;
- предполагается, что на выходе будет $x = xman \times 2^{xexp}$;
- vexp инвертируется и сдвигается вправо на один разряд с расширением знака;
- если vexp = −128, то результат насыщается до наибольшего положительного числа, и устанавливается флаг переполнения.

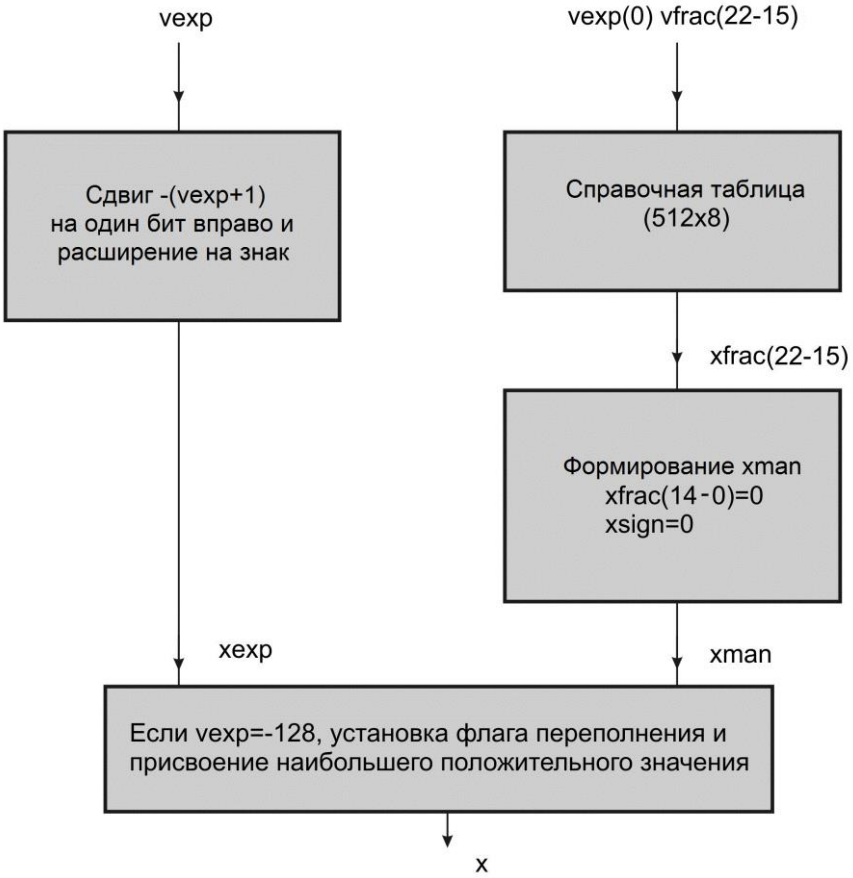


Рисунок 5.41 – Алгоритм инструкции RSQRF

Таблица значений читается посредством формирования 9-разрядного адреса, включающего наименьший значащий разряд vexp и разряды 22 − 15 vfrac. Восьмиразрядный выход таблицы значений формирует разряды 22 − 15 xfrac. Разряды 14 − 0 xfrac сбрасываются в 0. xsign устанавливается в 0. Здесь не выполняется условие для отрицательных значений v.

Значения таблицы генерируются в зависимости от входных чисел.

Результат этого алгоритма: деление – выполнение простого умножения: y/v = yx[n].

В уравнении x[n] – это оценка 1/v, как решения алгоритма Ньютона-Рафсона или другого алгоритма.

5.5.12.1 Алгоритм Ньютона-Рафсона

Инструкция RSQRF позволяет получить обратную величину квадратного корня числа. Оценка имеет правильную экспоненту, а мантисса имеет точность до восьмой двоичной позиции (т. е. погрешность мантиссы составляет $< 2^{-8}$). Алгоритм Ньютона-Рафсона (приведён ниже) может быть использован для уточнения мантиссы:

$$x[n+1] = x[n](1.5 - (v/2)x[n]x[n]), \quad (5.12)$$

где v = число, обратное значение которого ищется.

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

Программа ядра процессора 1867ВЦ8Ф1 для реализации этого алгоритма приведена в примере 5.16. Каждый шаг алгоритма помечен соответствующей точностью, достигнутой в конце шага. Алгоритм занимает десять машинных циклов.

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

```
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 ; завершение первой итерации (16-разрядная точность)
;
MPYF  R1, R1, R2
MPYF  R0, R2
SUBRF 1.5, R2
MPYF  R2, R1 ; завершение второй итерации (32-разрядная точность)
;
;          ; R1 = 1/(v**0.5)
;
```

5.6 Адресация

ПЦОС поддерживает пять групп режимов адресации. Внутри группы могут быть использованы шесть типов адресации, которые обеспечивают доступ к данным в памяти, регистрам и командным словам. В этом разделе подробно рассматривается операции, кодирование и применение режимов адресации. Также здесь описано управление системными стеком, очередями и исключениями из очереди в памяти. В этом подразделе рассмотрены следующие основные темы:

- типы адресации (5.6.1):
    - регистровый;
    - прямой;
    - косвенный;
    - короткий непосредственный;
    - длинный непосредственный;
    - относительный (относительно PC);
- группы режимов адресации (5.6.7):
    - основные режимы адресации;
    - трёхоперандные режимы адресации;
    - параллельные режимы адресации;
    - режим длинной непосредственной адресации;
    - режимы адресации условных переходов;
- циклическая адресация (5.6.8);
- бит-реверсная адресация (5.6.9);
- управление системным стеком (5.6.10).

5.6.1 Типы адресации

Шесть типов адресации делают возможным доступ к данным в памяти, регистрам и командным словам:

- регистровый;
- прямой;
- косвенный;
- короткий непосредственный;
- длинный непосредственный;
- относительный (относительно PC).

Некоторые типы адресации присутствуют в одних инструкциях и не присутствуют в других. По этой причине типы адресации используются в пяти различных группах режимов адресации:

- основные режимы адресации G:
    - регистровый;
    - прямой;
    - косвенный;
    - короткий непосредственный;
- трёхоперандные режимы адресации T:
    - регистровый;
    - косвенный;
- режимы параллельной адресации P:
    - регистровый;
    - косвенный;
- режим длинной непосредственной адресации:
    - длинный непосредственный;
- режимы адресации условных переходов B:
    - регистровый;
    - относительный.

Сначала будут рассмотрены шесть типов адресации, затем пять групп режимов адресации.

5.6.2 Регистровая адресация

В регистровой адресации операнд содержится в регистре ЦПУ, например:

ABSF R1; R1 = | R1 |.

Синтаксис для регистров ЦПУ, синтаксис ассемблера и назначение этих регистров приведены в таблице 5.14.

Таблица 5.14 − Регистры ЦПУ, синтаксис ассемблера и их назначение

Машинный 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 повторения
1Ah	RE	Регистр адреса конца повторения
1Bh	RC	Счётчик повторений
–	IVTP	Указатель таблицы вектора системных прерываний
–	TVTP	Указатель таблицы вектора программных прерываний

5.6.3 Прямая адресация

В прямой адресации адрес данных формируется путём конкатенации восьми младших разрядов указателя страницы данных DP с 16 младшими разрядами командного слова (expr). В результате программа имеет возможность обращаться к большому адресному пространству 65536 страниц (по 64K слов на каждой) без изменения значения DP. Синтаксис и операция прямой адресации приведены ниже.

Синтаксис: @expr

Операция: адрес = DP конкатенируется с expr

На рисунке 5.42 приведено формирование адреса данных. Пример 5.17 иллюстрирует прямую адресацию для команды с данными до и после её выполнения.

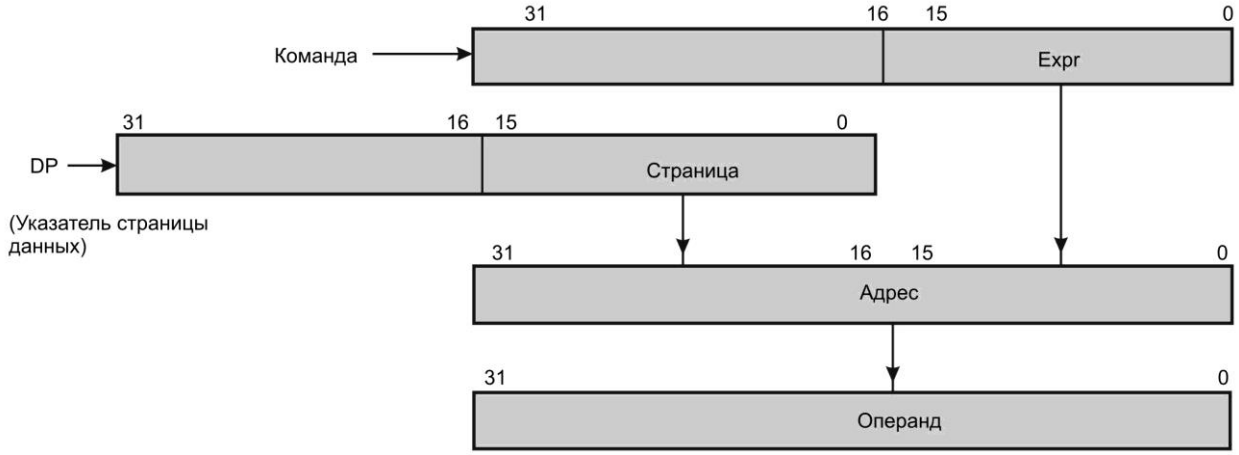


Рисунок 5.42 − Прямая адресация

Пример 5.17 – Прямая адресация

ADDI @0BCDEh, R7

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

5.6.4 Косвенная адресация

При использовании косвенной адресации адрес операнда в памяти определяется, содержимым вспомогательных регистров, а также дополнительным смещением и индексными регистрами. Генерация адреса осуществляется арифметическими устройствами вспомогательных регистров ARAU, реализующими беззнаковую арифметику. При косвенной адресации используются все 32 бита вспомогательных и индексных регистров.

Гибкость косвенной адресации обеспечивается благодаря независимому функционированию блоков ARAU и основной вычислительной секции процессорного ядра 1867ВЦ8Ф1 (вычисление адреса в ARAU осуществляется параллельно с операциями ALU и умножителя). Косвенная адресация определяется пятиразрядным полем в командном слове, именуемым «mod». Смещение − это либо восьмиразрядное целое без знака, содержащееся в командном слове или неявное смещение на единицу. Два индексных регистра IR0 и IR1 также могут быть использованы в косвенной адресации, реализуя 32-битные косвенные смещения (IR0 и IR1 обрабатываются как целые числа со знаком). В некоторых случаях используется также циклическая или бит-реверсная адресация. Механизм формирования адресов в циклической адресации рассмотрен в 5.6.8, бит-реверсной – в 5.6.9.

В таблице 5.15 представлены разные виды косвенной адресации с соответствующим каждому виду полем модификации (mod), синтаксисом ассемблера, операцией и функцией. На рисунке 5.43 показан формат косвенной адресации операнда в инструкционной кодировке. Поле disp не существует для некоторых инструкций.

Рисунок 5.43 − Формат косвенной адресации

Примечание − Вспомогательный регистр ARn может быть закодирован в слове инструкции соответственно его двоичному представлению n (т. е. AR3 кодируется как 11_2), а не его машинным адресом, как показано в таблице 5.14.

Таблица 5.15 − Косвенная адресация

Поле модификации	Синтаксис	Операция	Описание
1	2	3	4
(a) Косвенная адресация со смещением			
00000	* + ARn(disp)	addr = ARn + disp	С предварительным добавлением 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 циркулярной модификацией
00111	*ARn – – (disp)%	addr = ARn ARn = circ(ARn − disp)	С последующим вычитанием смещения и циркулярной модификацией
(b) Косвенная адресация с индексным регистром IR0			
01000	* + ARn(IR0)	addr = ARn + IR0	С предварительным добавлением индекса IR0
01001	* – ARn(IR0)	addr = ARn − IR0	С предварительным вычитанием индекса IR0
01010	* + + ARn(IR0)	addr = ARn + IR0 ARn = ARn + IR0	С предварительным добавлением индекса IR0 и модификацией
01011	* – – ARn(IR0)	addr = ARn − IR0 ARn = ARn − IR0	С предварительным вычитанием индекса IR0 и модификацией

Окончание таблицы 5.15

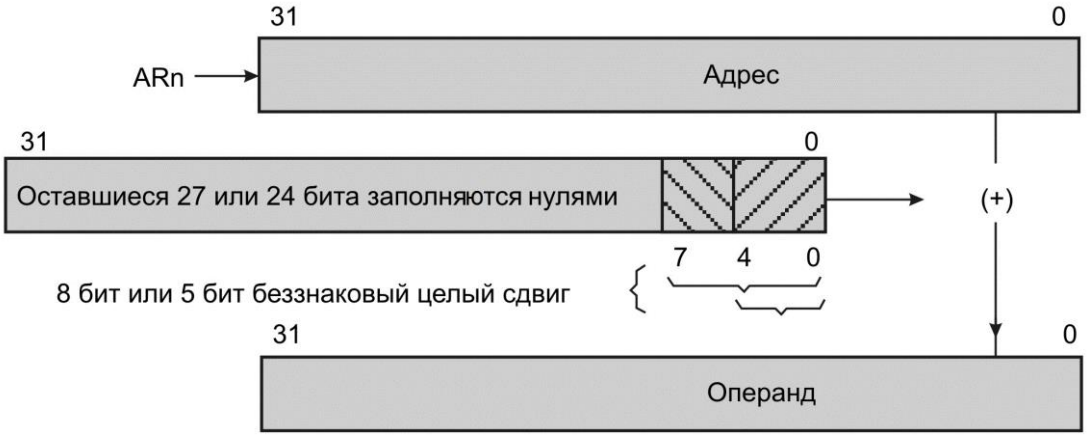
1	2	3	4
01100	*ARn + + (IR0)	addr = ARn ARn = ARn + IR0	С последующим добавлением индекса IR0 и модификацией
01101	*ARn – – (IR0)	addr = ARn ARn = ARn − IR0	С последующим вычитанием индекса IR0 и модификацией
01110	*ARn + + (IR0)%	addr = ARn ARn = circ(ARn + IR0)	С последующим добавлением индекса IR0 и циркулярной модификацией
01111	*ARn – – (IR0)%	addr = ARn ARn = circ(ARn − IR0)	С последующим вычитанием индекса IR0 и циркулярной модификацией
(c) Косвенная адресация с индексным регистром IR1			
10000	* + ARn(IR1)	addr = ARn + IR1	С предварительным добавлением индекса IR1
10001	* – ARn(IR1)	addr = ARn − IR1	С предварительным вычитанием индекса IR1
10010	* + + ARn(IR1)	addr = ARn + IR1 ARn = ARn + IR1	С предварительным добавлением индекса IR1 и модификацией
10011	*– – ARn(IR1)	addr = ARn − IR1 ARn = ARn − IR1	С предварительным вычитанием индекса IR1 и модификацией
10100	*ARn + + (IR1)	addr = ARn ARn = ARn + IR1	С последующим добавлением индекса IR1 и модификацией
10101	*ARn – – (IR1)	addr = ARn ARn = ARn − IR1	С последующим вычитанием индекса IR1 и модификацией
10110	*ARn + + (IR1)%	addr = ARn ARn = circ(ARn + IR1)	С последующим добавлением индекса IR1 и циркулярной модификацией
10111	*ARn – – (IR1)%	addr = ARn ARn = circ(ARn − IR1)	С последующим вычитанием индекса IR1 и циркулярной 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 = вспомогательный регистр AR0 − AR7;
IRn = индексный регистр IR0 или IR1;
disp = смещение;
+ + = сложение и модификация;
– – = вычитание и модификация;
circ() = адрес в циклической адресации;
% = где выполняется циклическая адресация;
B = где выполняется бит-реверсная адресация.

Примеры с 5.18 по 5.35 показывают операции для каждого типа косвенной адресации.

Пример 5.18 – Косвенная адресация с использованием вспомогательного регистра
Вспомогательный регистр ARn содержит адрес операнда.
Операция: адрес операнда = ARn
Ассемблерный синтаксис: *ARn
Модификационное поле: 11000

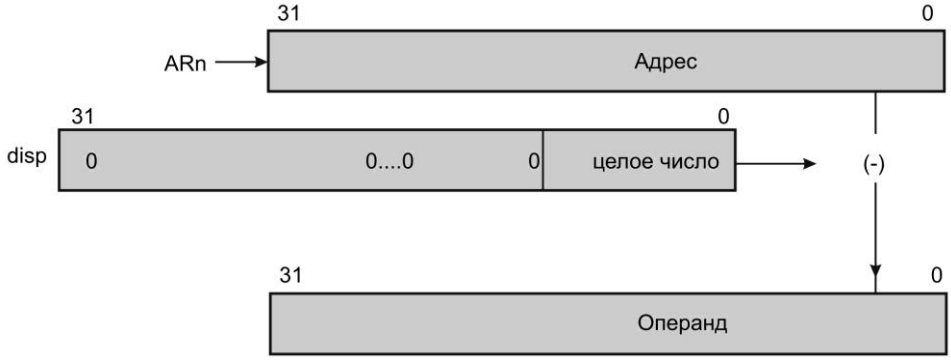
Пример 5.19 – Косвенная адресация с предварительным добавлением смещения
Адрес выбранного операнда суммируется со вспомогательным регистром ARn и смещением (disp). Смещение – это каждый пятый или восьмой бит беззнакового целого, содержащегося в слове инструкции, или подразумевается значение 1.
Операция: адрес операнда = ARn + disp
Ассемблерный синтаксис: *ARn (disp)
Модификационное поле: 00000



Пример 5.20 – Косвенная адресация с предварительным вычитанием смещения

Адрес выбранного операнда – это содержимое вспомогательного регистра ARn минус смещение (disp). Смещение – это каждый восьмой бит беззнакового целого, содержащегося в слове инструкции, или подразумевается значение 1.

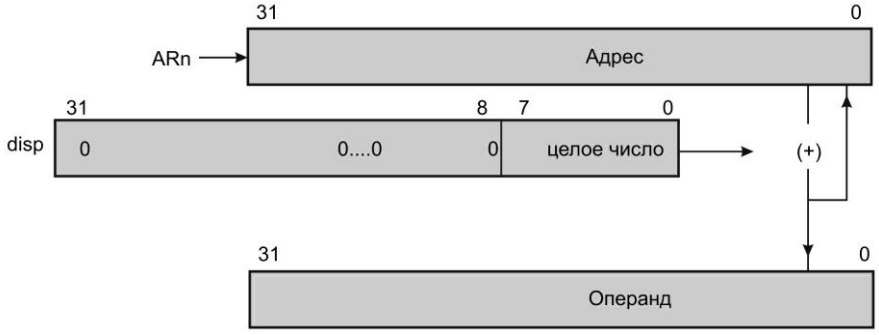
Операция: адрес операнда = ARn – disp
Ассемблерный синтаксис: *ARn (disp)
Модификационное поле: 00001



Пример 5.21 – Косвенная адресация с предварительным добавлением смещения и модификацией

Адрес выбранного операнда – это сумма вспомогательного регистра ARn и смещения (disp). Смещение – это каждый восьмой бит беззнакового целого, содержащегося в слове инструкции, или подразумевается значение 1. После этого данные выбираются, вспомогательный регистр обновляется со сгенерированным адресом.

Операция: адрес операнда = ARn + disp
ARn = ARn + disp
Ассемблерный синтаксис: * + + ARn (disp)
Модификационное поле: 00010



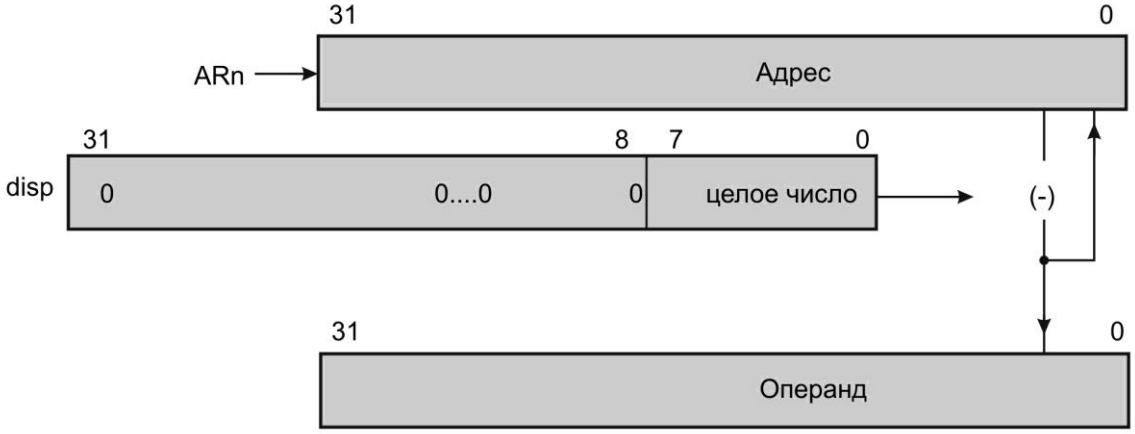
Пример 5.22 – Косвенная адресация с предварительным вычитанием смещения и модификацией

Адрес выбранного операнда – это содержимое вспомогательного регистра ARn минус смещение (disp). Смещение – это каждый восьмой бит беззнакового целого, содержащегося в слове инструкции, или подразумевается значение 1. После этого данные выбираются, вспомогательный регистр обновляется со сгенерированным адресом.

Операция: адрес операнда = ARn – disp
ARn = ARn – disp

Ассемблерный синтаксис: * – – ARn (disp)

Модификационное поле: 00011



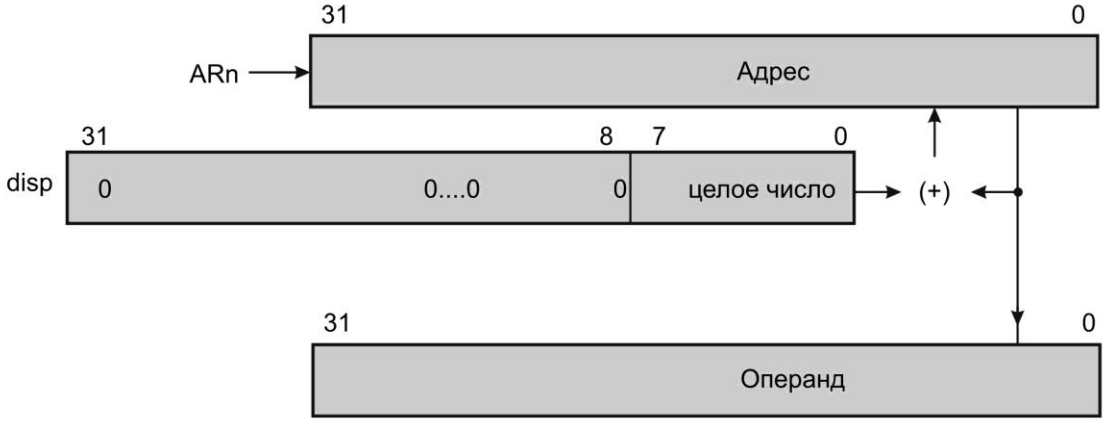
Пример 5.23 – Косвенная адресация с последующим добавлением смещения и модификацией

Адрес выбранного операнда – это содержимое вспомогательного регистра ARn. После выбора операнда добавляется смещение (disp) к вспомогательному регистру. Смещение – это каждый восьмой бит целого без знака, содержащегося в слове инструкции, или подразумевается значение 1.

Операция: адрес операнда = ARn
ARn = ARn + disp

Ассемблерный синтаксис: *ARn + + disp

Модификационное поле: 00100



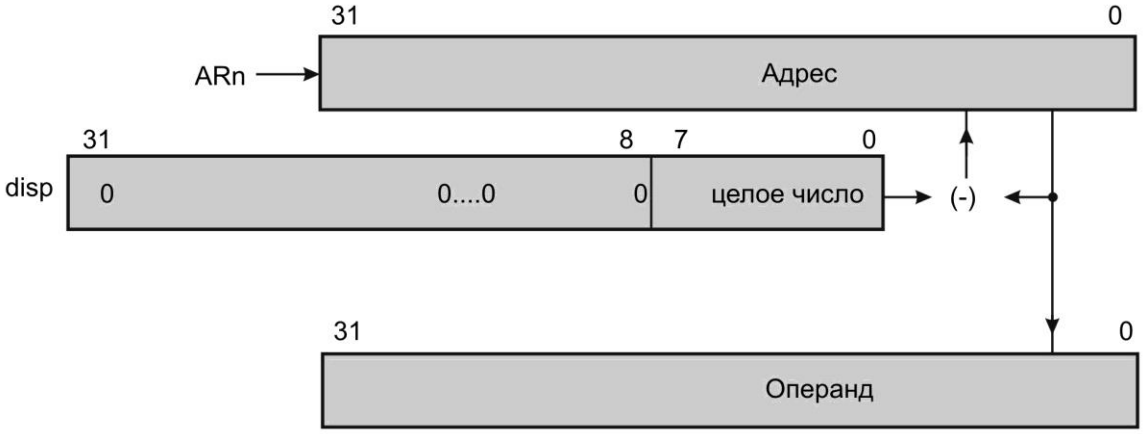
Пример 5.24 – Косвенная адресация с последующим смещением и модификацией

Адрес выбранного операнда – это содержимое вспомогательного регистра ARn. После выбора операнда вычитается смещение (disp) от вспомогательного регистра. Смещение – это каждый восьмой бит целого без знака, содержащегося в слове инструкции, или подразумевается значение 1.

Операция: адрес операнда = ARn
ARn = ARn – disp

Ассемблерный синтаксис: *ARn – –disp

Модификационное поле: 00101



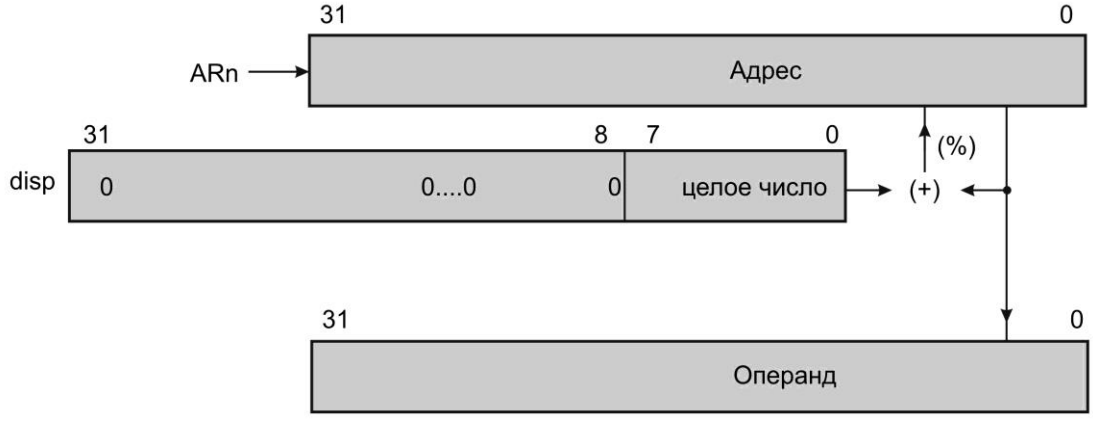
Пример 5.25 – Косвенная адресация с последующим смещением сложения и циркулярная модификация

Адрес выбранного операнда – это содержимое вспомогательного регистра ARn. После выбора операнда добавляется смещение (disp) к содержимому вспомогательного регистра через циркулярную адресацию. Смещение – это каждый восьмой бит целого без знака, содержащегося в слове инструкции, или подразумевается значение 1.

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

Ассемблерный синтаксис: *ARn + + (disp) %

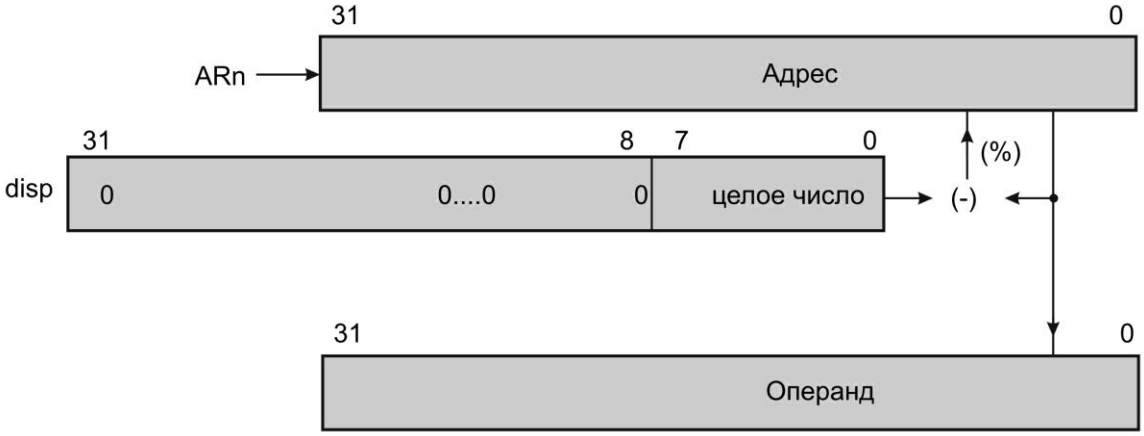
Модификационное поле: 00110



Пример 5.26 – Непрямая адресация с последующим смещением и циркулярной модификацией

Адрес выбранного операнда – это содержимое вспомогательного регистра ARn. После выбора операнда вычитается смещение (disp) от содержимого вспомогательного регистра через циркулярную адресацию. Этот результат используется для обновления вспомогательного регистра. Смещение – это каждый восьмой бит целого без знака, содержащегося в слове инструкции, или подразумевается значение 1.

Операция: адрес операнда = ARn
ARn = circ (ARn − disp)
Ассемблерный синтаксис: *ARn – – (disp) %
Модификационное поле: 00111



Пример 5.27 – Косвенная адресация с предварительным добавлением индекса

Адрес выбранного операнда – это сумма вспомогательного регистра ARn и индексного регистра IR0 или IR1.

Операция: адрес операнда = ARn + IRm
Ассемблерный синтаксис: *+ARn (IRm)
Модификационное поле: 01000, если m = 0
10000, если m = 1

Пример 5.28 – Косвенная адресация с предварительным вычитанием индекса

Адрес выбранного операнда – это разница вспомогательного регистра ARn и индексного регистра IR0 или IR1.

Операция: адрес операнда = ARn – IRm

Ассемблерный синтаксис: * − ARn (IRm)

Модификационное поле: 01001, если m = 0
10001, если m = 1

Пример 5.29 – Косвенная адресация с предварительным добавлением индекса и модификацией

Адрес выбранного операнда – это сумма вспомогательного регистра ARn и индексного регистра IR0 или IR1. После того как данные выбраны, вспомогательный регистр обновлён с генерируемым адресом.

Операция: адрес операнда = ARn + IRm
ARn = ARn + IRm

Ассемблерный синтаксис: * + + ARn (IRm)

Модификационное поле: 01010, если m = 0
10010, если m = 1

Пример 5.30 – Косвенная адресация с предварительным вычитанием индекса и модификацией

Адрес выбранного операнда – это разница вспомогательного регистра ARn и индексного регистра IR0 или IR1. Результирующим адресом становится новое содержимое вспомогательного регистра.

Операция: адрес операнда = ARn – IRm
ARn = ARn – IRm

Ассемблерный синтаксис: * – – ARn (IRm)

Модификационное поле: 01011, если m = 0
10011, если m = 1

Пример 5.31 – Косвенная адресация с последующим добавлением индекса и модификацией

Адрес выбранного операнда – это разница вспомогательного регистра ARn и индексного регистра IR0 или IR1. Результирующим адресом становится новое содержимое вспомогательного регистра.

Операция: адрес операнда = ARn
ARn = ARn + IRm

Ассемблерный синтаксис: *ARn + + (IRm)

Модификационное поле: 01100, если m = 0
10100, если m = 1

Пример 5.32 – Косвенная адресация с последующим вычитанием индекса и модификацией

Адрес выбранного операнда – это содержимое вспомогательного регистра ARn. После выбора операнда индексный регистр IR0 или IR1 вычитается из вспомогательного регистра.

Операция: адрес операнда = ARn
ARn = ARn − IRm

Ассемблерный синтаксис: *ARn – – (IRm)

Модификационное поле: 01101, если m = 0
10101, если m = 1

Пример 5.33 – Косвенная адресация с последующим добавлением и циркулярной модификацией

Адрес выбранного операнда – это содержимое вспомогательного регистра ARn. После выбора операнда индексный регистр IR0 или IR1 добавляется к вспомогательному регистру. Это значение преобразуется через циркулярную адресацию и перезаписывается содержимым вспомогательного регистра.

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

Ассемблерный синтаксис: *ARn + + (IRm) %

Модификационное поле: 01110, если m = 0
10110, если m = 1

Пример 5.34 – Косвенная адресация с последующим вычитанием и циркулярной модификацией

Адрес выбранного операнда – это содержимое вспомогательного регистра ARn. После выбора операнда индексный регистр IR0 или IR1 вычитается из вспомогательного регистра. Результат преобразуется через циркулярную адресацию и перезаписывает содержимое вспомогательного регистра.

Операция: адрес операнда = ARn
ARn = circ (ARn − IRm)
Ассемблерный синтаксис: *ARn – – (IRm) %
Модификационное поле: 01111, если m = 0
10111, если m = 1

Пример 5.35 – Косвенная адресация с последующим добавлением и инвертированием разрядов

Адрес выбранного операнда – это содержимое вспомогательного регистра ARn. После выбора операнда индексный регистр IR0 или IR1 добавляется к вспомогательному регистру. Это сложение преобразуется с воспроизведением обратного переноса и может быть использовано для выполнения бит-реверсной (B) адресации. Это значение перезаписывает содержимое вспомогательного регистра.

Операция: адрес операнда = ARn
ARn = circ (ARn + IRm)
Ассемблерный синтаксис: *ARn + + (IRm) %
Модификационное поле: 01110, если m = 0
10110, если m = 1

5.6.5 Непосредственная адресация

В режиме непосредственной адресации операнд представляет собой 8- или 16-битное непосредственное значение, содержащееся в младших разрядах слова инструкции. В зависимости от типов данных, допустимых для инструкции, непосредственный операнд может целым числом в дополнительном (до двух) коде, целым числом со знаком или без знака или числом с плавающей запятой. Этот режим имеет следующий синтаксис:

Синтаксис: expr

Пример 5.36 показывает принцип работы с данными из предыдущей и последующей инструкции. Заметим, что AND и AND3 дают различные результаты.

Пример 5.36 – Непосредственная адресация

Инструкция		Перед	После
SUBI	1, R0	R0 = 0h	R0 = 00 FFFF FFFFh
LDI	0 FFFFh, R0	R0 = 0h	R0 = 00 FFFF FFFFh
LDF	5.0, R0	R0 = 0h	R0 = 02 2000 0000h
OR	0 FFFFh, 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

5.6.6 Относительная адресация − относительно PC

PC относительная адресация используется при ветвлении. При этом 16 или 24 наименее значащих битов слова инструкции добавляются к содержимому PC регистра. Ассемблер использует src (метку или адрес), определённый пользователем, и генерирует смещение. Если переход стандартный, то смещение равно [метка – (адрес инструкции + 1)]. Если переход отложенный, то смещение эквивалентно [метка – (адрес инструкции + 3)].

Смещение размещается как 16- или 24-битное целое в младших битах командного слова. Смещение добавляется к содержимому PC на фазе декодирования конвейера. Поскольку PC инкрементируется на единицу в фазе выборки, смещение добавляется к этому, уже инкрементированному, значению PC.

Синтаксис: expr (метка или адрес)

Пример 5.37 показывает инструкции с данными до и после выполнения команды.

Пример 5.37 – PC относительная адресация

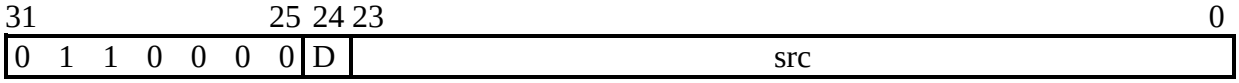
```
            BU    NEWPC     ; адрес BU инструкции = 1,
            …               ; NEWPC метка = 5, смещение = 3
NEWPC       …               ; смещение = 5 – (1 + 1)
```

Перед инструкцией	После инструкции
Фаза декодирования	Фаза выполнения
PC = 2h	PC = 5h

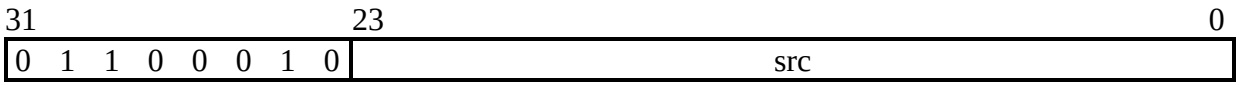
24-битная адресация используется при кодировании инструкций управления программой (например, BR, BRD, CALL, RPTB, RPTBD, LAJ). В зависимости от инструкции новое значение PC получается добавлением 24-битного знакового значения из командного слова к текущему значению PC. Бит 24 определяет тип ветвления (D = 0 – для стандартного ветвления или D = 1 – для отложенного ветвления).

Примеры кодов некоторых их этих команд представлены на рисунке 5.44.

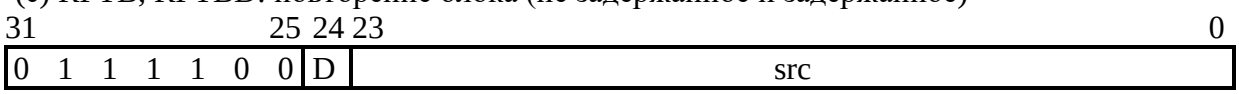
(a) BR, BRD: безусловные ветвления (не задержанное и задержанное)



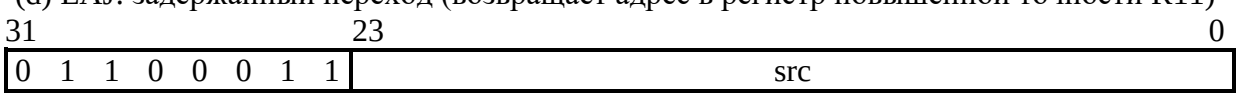
(b) CALL: вызов подпрограммы



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



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



Примечание – Принятое условное обозначение: D (Delay) – задержка выполнения инструкции:

- D = 0 – нет задержки;
- D = 1 – есть задержка.

Рисунок 5.44 − Коды команд при относительной адресации

5.6.7 Группы режимов адресации

Шесть типов адресации используются для формирования следующих пяти групп режимов адресации:

- основные режимы адресации G;
- трёхоперандный режим адресации T;
- параллельные режимы адресации P;
- режимы адресации условных переходов B;
- режим длинной непосредственной адресации.

5.6.7.1 Основные режимы адресации

Инструкции, которые используют режимы основной адресации, являются командами общего назначения, такими как ADDI, MPYF и LSH. Такие команды 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, индицирующие основной режим адресации. Разряды 22 и 21 определяют поле основного режима адресации G, указывая каким образом разряды 15 – 0 интерпретируются для адресации операнда src.

Возможные состояния разрядов 22 и 21 (поле G):

00 – регистровый (все регистры ЦПУ, если не определены иначе);

01 – прямой;

10 – косвенный;

11 – непосредственный.

Если поля src и dst соответствуют спецификациям регистра, то значение в этих полях соответствует адресам регистра ЦПУ, как определено таблицей 5.14. Для режимов основной адресации допустимы следующие значения ARn:

ARn, $0 \le n \le 7$

На рисунке 5.45 приведено кодирование для основных режимов адресации.

Примечание – «modn» обозначает поле модификации, определённое вместе с полем ARn.

		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	0 0 0 0 0	0 0 0	0 0 0	0 0 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 5.45 − Кодирование для основных режимов адресации

5.6.7.2 Трёхоперандные режимы адресации

Команды, использующие трёхоперандные режимы адресации, такие как ADDI3, LSH3, CMPF3 или XOR3 обычно представляются в форме:

src1 операция src2 → dst,

где операнд назначения обозначен «dst», операнды источника «src1» и «src2», «операция» определяет выполняемую операцию. Символ «3» может быть опущен в трёхоперандных инструкциях.

В разрядах 31 – 29 установлено значение 001_2, обозначающее команды трёхоперандного режима адресации. Разряды 22 и 21 определяют поле T трёхоперандного режима адресации, указывая каким образом разряды 15 – 0 будут интерпретироваться для адресации операнда src. Разряды 15 – 8 используются для определения адреса src1, и разряды 7 – 0 используются для определения адреса src2.

Варианты для разрядов 22 и 21:

T src1 src2
00 – регистровый – регистровый
01 – косвенный – регистровый
10 – регистровый – косвенный
11 – косвенный – косвенный

На рисунке 5.46 приведено кодирование для трёхоперандной адресации. Если поля src1 и src2 используют один и тот же вспомогательный регистр, то оба адреса генерируются корректно. Однако, только значение, определяемое полем src1, сохраняется в указанном вспомогательном регистре. Ассемблер выдаёт предупреждение, если src1 и src2 указывают на один и тот же вспомогательный регистр.

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.

При косвенной адресации в трёхоперандном режиме допустимо смещение 0 или 1 (если используется смещение) и могут быть использованы индексные регистры IR0 и IR1. Смещение 1 – неявное и не закодировано явно в командном слове.

		T	Назначение	src1						src2	
31 29	28 23	22 21	20 16	15 13	12 11 10 8			7 5		4 3 2 0	
001	операция	00	dst	000	src1			000		src2	
001	операция	01	dst	modn		ARn		000		src2	
001	операция	10	dst	000	src1			modn			ARn
001	операция	11	dst	modn		ARn		modm			ARm

Рисунок 5.46 − Кодирование для трёхоперандных режимов адресации

5.6.7.3 Параллельные режимы адресации

Команды, использующие параллельную адресацию (обозначены || (две вертикальные линии)), обеспечивают максимально возможный параллелизм. Операнды назначения обозначены d1 и d2, обозначающие dst1 и dst2, соответственно, см. рисунок 5.47. Операнды источника, обозначенные src1 и src2, используют регистры повышенной точности. Выполняемые параллельные операции обозначены как «операция».

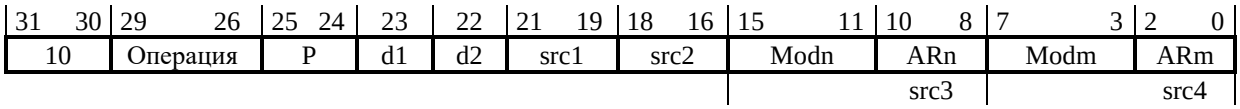


Рисунок 5.47 − Кодирование для параллельных режимов адресации

Поле параллельного режима адресации P определено по использованию операнда, т. е. являются ли они операндами источника или назначения. Связь между полем P и операндами подробно представлена в описании отдельных параллельных команд. Однако операнды всегда кодируются одинаково. В разрядах 31 и 30 установлено значение 10_2, обозначающее команды параллельного режима адресации. Разряды 21 − 0 определяют поле параллельного режима адресации P, которое определяет, как разряды 21 − 0 будут интерпретироваться для адресации операндов 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, какое поле модификации соответствует какому ARn или ARm (вспомогательный регистр) полю.

Операнды параллельной адресации приведены ниже:

src1 $0 \le src1 \le 7$ (регистры повышенной точности R0 − R7)

src2 $0 \le src2 \le 7$ (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)

Необходимо отметить, что в параллельных инструкциях используются только регистры R0 − R7; R8 – R11 не используются при параллельной адресации.

Как и в режиме адресации с тремя операндами, косвенная адресация в режиме параллельной адресации допускает смещение 0 или 1 и использование индексных регистров (IR0 и IR1). Смещение 1 подразумевается и явно не кодируется в командном слове.

В кодировке для этого режима, приведённой на рисунке 5.47, если поля src3 и src4 используют один и тот же вспомогательный регистр, оба адреса генерируются корректно, но только значение, определяемое полем src3, сохраняется в указанном вспомогательном регистре. Ассемблер выдаёт предупреждение, если src3 и src4 указывают на один и тот же вспомогательный регистр.

5.6.7.4 Режим длинной непосредственной адресации

Режим длинной непосредственной адресации используется для кодирования инструкций управления программой BR, BRd и CALL, для которых полезно иметь 24-разрядный адрес, содержащийся в командном слове. Безусловный переход BR (стандартный) и BRD (задержанный) использует режим длинной непосредственной адресации. В разрядах 31 − 25 установлено значение 0110000, обозначающее команды режима длинной непосредственной адресации. Выбор 24 разряда определяет способ ветвления: D = 0 для стандартного перехода или D = 1 для задержанного перехода. Длинный непосредственный операнд используется, это 24-разрядный src. Эти команды записываются, как показано на рисунке 5.48.

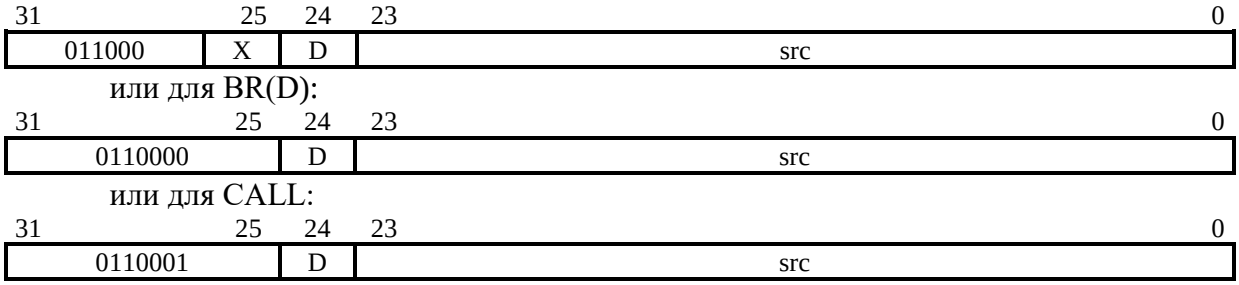


Рисунок 5.48 − Кодирование для режима длинной непосредственной адресации

5.6.7.5 Режимы адресации условных переходов

Команды, использующие режимы адресации условных переходов Bcond, BcondD, CALLcond, DBcond и DBcondD, могут выполнять различные условные операции. В разрядах 31 − 27 установлено значение 01101, обозначающее команды режима адресации условных переходов; разряд 26 установлен в 0 или 1, первый выбирает DBcond, другой − Bcond. Выбор 25 разряда определяет режим адресации условных переходов B. Если B = 0, то используется регистровая адресация, если B = 1, то используется PC-относительная адресация. Выбор 21 разряда устанавливает способ ветвления: D = 0 для стандартного перехода или D = 1 для задержанного перехода. Поле (cond) определяет условие, проверяемое для определения выполняемого действия, т. е. будет ли выполняться ветвление, или продолжится обычное исполнение программы. На рисунке 5.49 приведено кодирование для режима адресации условных переходов.

DBcond(D):

31 26	25	24 22	21	20 16	15 5	4 0
01101 1	B	ARn	D	cond	00000000000	src reg
01101 1	B	ARn	D	cond	непосредственная (относительно PC)	

Bcond(D):

31 26	25	24 22	21	20 16	15 5	4 0
01101 0	B	000	D	cond	00000000000	src reg
01101 0	B	000	D	cond	непосредственная (относительно PC)	

CALLcond(D):

31 26	25	24 22	21	20 16	15 5	4 0
011100	B	000	0	cond	00000000000	src reg
011100	B	000	0	cond	непосредственная (относительно PC)	

Рисунок 5.49 − Кодирование для режимов адресации условных переходов

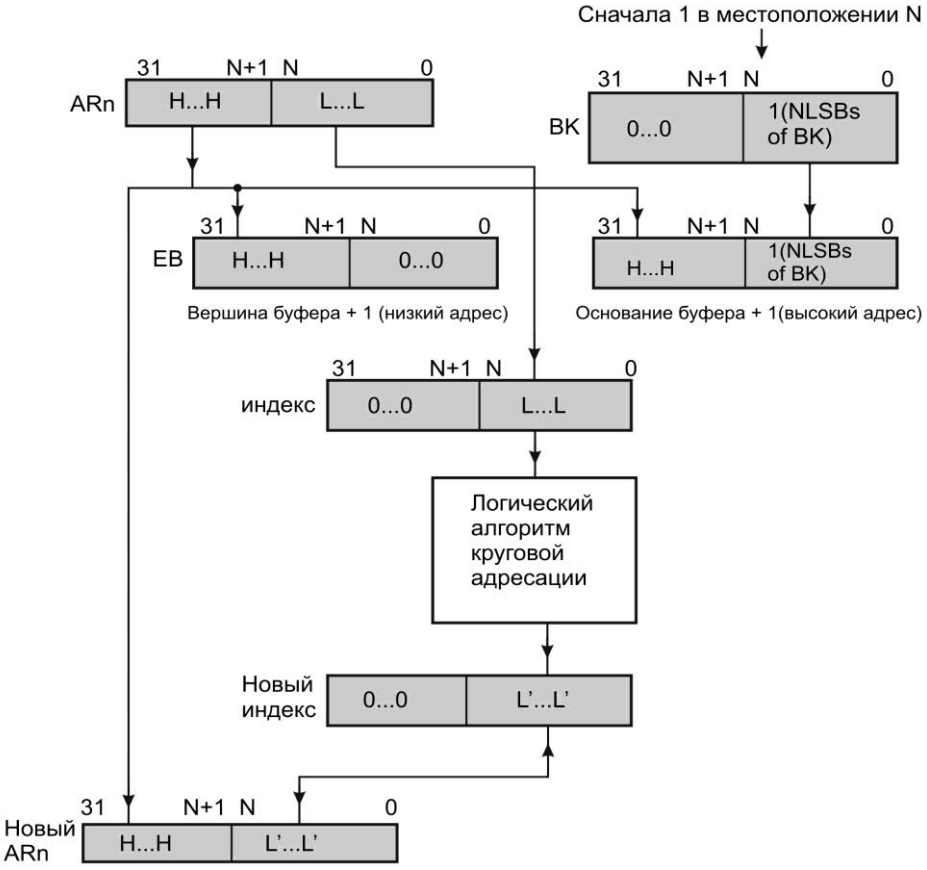
5.6.8 Циклическая адресация

Многие алгоритмы, такие как свёртка и корреляция, требуют организации циклического буфера в памяти. В процессе свёртки и корреляции циклический буфер используется для реализации скользящего окна, содержащего самые последние обрабатываемые значения. По мере того, как вводятся новые данные, они стирают предыдущую информацию. Ключом для реализации циклического буфера является реализация циклического режима адресации. Далее приведено описание циклического режима адресации ПЦОС.

Регистр размера блока BK определяет размер циклического буфера. Дно циклического буфера определяется как первый разряд (считая от старшего значащего разряда к младшему) в младших 16 разрядах регистра BK плюс выбираемый пользователем вспомогательный регистр ARn. Если размещение первого единичного разряда определено как разряд N, то адрес вершины буфера именуется действительным основанием EB и соответствует разрядам с 31 до (N + 1) регистра ARn с нулевыми разрядами с N до 0 EB.

На рисунке 5.50 показана связь между регистром размера блока BK, вспомогательным регистром ARn, дном циклического буфера, вершиной циклического буфера и индексом в циклическом буфере.

Циклический буфер размером R должен начинаться на границе K разряда (младшие значащие разряды циклического буфера должны быть 0), где K − наименьшее целое, удовлетворяющее соотношению 2 > 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. XXXXXXXXXXXXXXXXXXXXXXXXX00000), и значение 31 разряда должно быть загружено в регистр BK.



Принятые условные обозначения:
ARn − вспомогательный регистр n;
BK − регистр размера блока;
EB − действительное основание;
H − старшие разряды;
L − младшие разряды;
L′ − новые младшие разряды;
LSB − младший значащий разряд;
N − значение разряда.

Рисунок 5.50 – Блок-схема циклической адресации

В циклической адресации «index» относится к N младшим значащим разрядам выбранного вспомогательного регистра, «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 ≤ index + step < BK:

index = index + step.

Иначе, если index + step ≥ BK:

index = index + step – BK.

Иначе, если index + step < 0:

index = index + step + BK.

На рисунке 5.51 показана реализация циклического буфера. Это иллюстрирует взаимосвязь сгенерированных величин с элементами в циклическом буфере.

Рисунок 5.51 − Реализация циклического буфера

На рисунке 5.52 приведён пример операции при циклической адресации. Предполагается, что все AR по четыре разряда, AR0 = 0000 и BK = 0110 (размер блока 6). Этот пример показывает последовательность модификаций и результирующее значение AR0. Он также иллюстрирует, как указатель продвигается по 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 (четвёртое значение)
* AR0                ; AR0 = 3 (пятое значение)
```

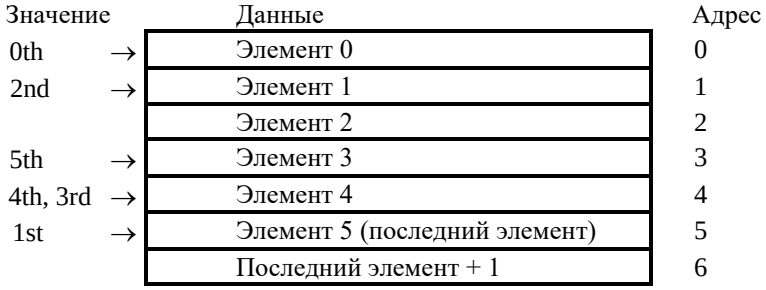


Рисунок 5.52 − Пример циклической адресации

Циклическая адресация особенно полезна для реализации КИХ-фильтров. На рисунке 5.53 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).

Рисунок 5.53 − 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 дно буфера входных отсчётов
*
TOP  LDF  IN, R3 ; Считать входной отсчёт
STF  R3,*AR1++% ; Сохранить с другим отсчётом и указать на вершину буфера
LDF  R0 ; Инициализировать R0
LDF  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
```

5.6.9 Бит-реверсная адресация

Бит-реверсная адресация в ПЦОС улучшает характеристики скорости выполнения и использования памяти программ для алгоритмов БПФ, которые используют различные основания. Базовый адрес бит-реверсной адресации должен быть размещён на границе размера таблицы. Например, если IR0 = 2^{n-1}, 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. IR0 определяет половину размера БПФ (быстрое преобразование Фурье); 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 бит-реверсной форме.

Чтобы проиллюстрировать этот вид адресации, рассмотрим восьмиразрядные вспомогательные регистры. Пусть AR2 содержит значение 0110 0000 (96). Это базовый адрес данных в памяти. Пусть IR0 содержит значение 0000 1000 (8). На рисунке 5.54 приведена последовательность модификаций и результирующие значения AR2.

```
* AR2 + + (IR0)B ; AR2 = 0110 0000 (нулевое значение)
* AR2 + + (IR0)B ; AR2 = 0110 1000 (первое значение)
* AR2 + + (IR0)B ; AR2 = 0110 0100 (второе значение)
* AR2 + + (IR0)B ; AR2 = 0110 1100 (третье значение)
* AR2 + + (IR0)B ; AR2 = 0110 0010 (четвёртое значение)
* AR2 + + (IR0)B ; AR2 = 0110 1010 (пятое значение)
* AR2 + + (IR0)B ; AR2 = 0110 0110 (шестое значение)
* AR2 ;            AR2 = 0110 1110 (седьмое значение)
```

Рисунок 5.54 − Пример бит-реверсной адресации

В таблице 5.16 приведены отношения шагов индекса и четырёх младших разрядов AR2. Как видно, можно найти четыре младших значащих разряда реверсированием набора разрядов шагов.

Таблица 5.16 − Шаг индекса и бит-реверсная адресация

Шаг	Набор разрядов	Бит-реверсный набор	Бит-реверсный 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

5.6.10 Управление системным стеком и стеком пользователя

ПЦОС обеспечивает специальный указатель стека (SP) для построения стеков в памяти. Кроме того, вспомогательные регистры также могут использоваться для создания пользовательских стеков и множества более общих линейных списков. Далее обсуждается реализация следующих способов линейных списков:

- стек − это линейный список, для которого все вставки и удаления выполняются на одном конце списка;
- очередь − это линейный список, для которого все вставки выполняются на одном конце списка, а все удаления выполняются в другом конце списка;
- выборка − это двусторонний линейный список очереди, для которого вставки и удаления могут выполняться в любом конце списка.

Указатель системного стека SP − это 32-разрядный регистр, содержащий адрес вершины системного стека. Системный стек заполняется от младших адресов памяти к старшим, см. рисунок 5.55. SP всегда указывает на следующий элемент, вталкиваемый в стек. При вталкивании данных в стек выполняется прединкремент, а при выталкивании – постдекремент указателя системного стека.

Счётчик команд вталкивает данные в системный стек во время вызовов подпрограмм, прерываний и системных прерываний. Стек может выполнять вталкивание и выталкивание данных, используя команды PUSH, POP, PUSHF и POPF.

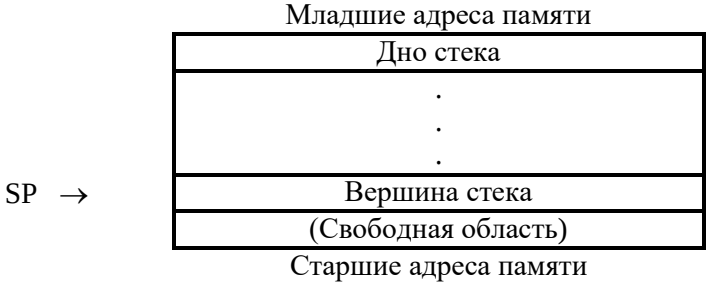


Рисунок 5.55 − Конфигурация системного стека

5.6.10.1 Стеки

Стеки могут быть построены от младших адресов памяти к старшим или от старших к младшим. Стеки могут быть построены, используя режимы прединкремента/декремента и постинкремента/декремента, изменением вспомогательных регистров AR. Изменение стека от старших адресов памяти к младшим может быть выполнено двумя путями:

- вариант 1: записывает в память, используя * -- ARn для вталкивания данных в стек, и считывает из памяти, используя * ARn ++ для выталкивания данных из стека.

- вариант 2: записывает в память, используя * ARn -- для вталкивания данных в стек, и считывает из памяти, используя * ++ ARn для выталкивания данных из стека.

Рисунок 5.56 иллюстрирует два этих варианта. Различие только в том, что при использовании варианта 1, AR всегда указывает на вершину стека, а в варианте 2 AR всегда указывает на следующую свободную область в стеке.

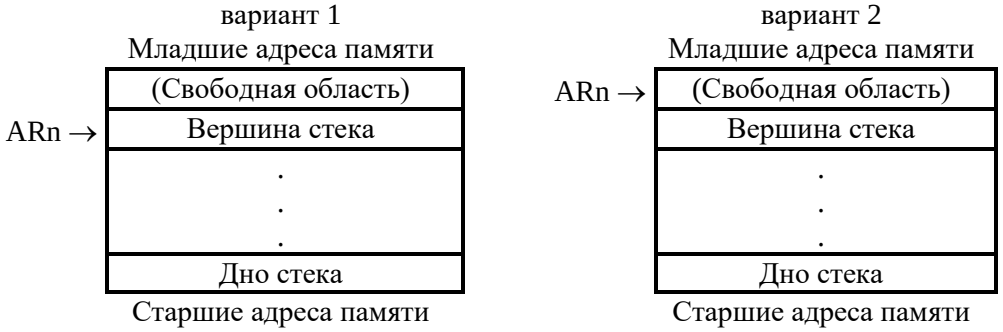


Рисунок 5.56 − Два варианта реализации стека

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

- вариант 3: записывает в память, используя * + + ARn для вталкивания данных в стек, и считывает из памяти, используя * ARn – – для выталкивания данных из стека.

- вариант 4: записывает в память, используя * ARn + + для вталкивания данных в стек, и считывает из памяти, используя * -- ARn для выталкивания данных из стека.

На рисунке 5.57 показаны эти два варианта. В варианте 3 AR всегда указывает на вершину стека. В варианте 4 − AR всегда указывает на следующую свободную область в стеке.

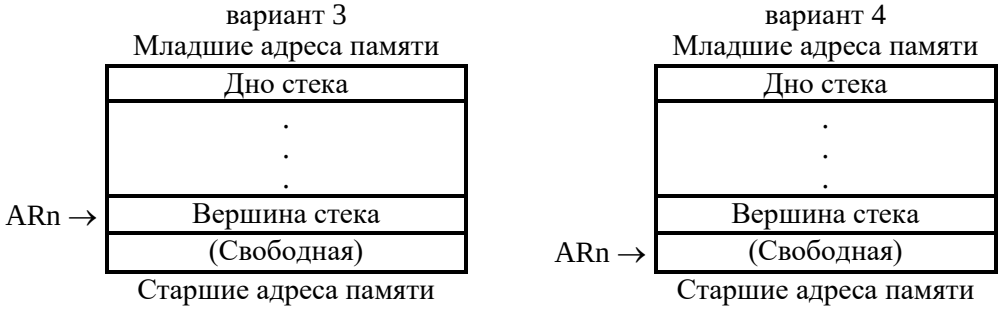


Рисунок 5.57 − Реализация стеков от младших адресов памяти к старшим

5.7 Управление процессом выполнения программы

Ядро процессора 1867ВЦ8Ф1 обеспечивает полное множество гибких и мощных конструкций, которые обеспечивают управление программным и аппаратным обеспечением выполнения программы. Программное управление обеспечивает повторения, ветвления, вызовы, программные прерывания и возвраты. Аппаратное управление включает в себя системные прерывания. Так как программирование включает множество различных конструкций, можно выбрать одну, наиболее подходящую для конкретного случая.

5.7.1 Режим повторений

Режим повторений ядра процессора 1867ВЦ8Ф1 может применяться для реализации циклов без дополнительных потерь. Для многих алгоритмов наибольшее время тратится на реализацию внутреннего ядра программного кода. Использование режима повторений позволяет максимально сократить время выполнения критичных по времени секций программы.

Ядро процессора 1867ВЦ8Ф1 обеспечивает три команды для реализации циклов без дополнительных потерь: RPTB (повторение блока кода), RPTBD (задержанное повторение блока кода) и RPTS (повторение одной команды).

- RPTB и RPTBD позволяют выполнять блок кода определённое количество раз;
- RPTS обеспечивает повторение одной команды определённое количество раз и уменьшает нагрузку шины за счёт однократной выборки инструкции.

RPTB и RPTS – четырёхцикловые инструкции; эти четыре цикла теряются только при первом выполнении цикла. Все последующие выполнения цикла производятся с нулевыми потерями. RPTBD – это одноцикловая инструкция.

Три регистра RS, RE и RC управляют счётчиком команд при его обновлении в режиме повторения. В таблице 5.17 описаны эти регистры.

Таблица 5.17 − Регистры режима повторения

Регистр	Функция
RS	Регистр начального адреса повторений. Содержит адрес первой инструкции повторяемого блока команд.
RE	Регистр конечного адреса повторений. Содержит адрес последней инструкции повторяемого блока кода. RE должен быть больше или равен RS.
RC	Регистр счета повторений. Содержит значение, на единицу меньшее количества повторов блока программного кода.

Для корректного выполнения программы в режимах повтора необходимо, чтобы все вышеперечисленные регистры, а также регистр состояния были правильно инициализированы. RPTB, RPTBD и RPTS выполняют эту инициализацию немного по-разному.

5.7.1.1 Разряды управления

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

RM (флаг режима повторений) разряд в регистре состояния, который определяет, работает ли процессор в режиме повторений. Если RM = 0, выборка в режиме повторений не производится, если RM = 1 − выборка в режиме повторений производится.

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

5.7.1.2 Операции в режиме повторений

Информация в регистрах режима повторения и соответствующих разрядах управления используется для контроля изменений PC, когда производится выборка в режиме повторения. Режимы повторения сравнивают содержимое регистра RE со счётчиком команд PC после выполнения каждой инструкции. Если они совпадают и счётчик повторений неотрицательный, то счётчик повторений уменьшается, в PC загружается начальный адрес повторений, и обработка продолжается. Выборка и соответствующие разряды состояния при необходимости изменяются. Заметим, что счётчик повторений RC никогда не изменяется, если RM равно нулю. Подробно алгоритм изменения PC показан в примере 5.38.

Примечание − Максимальное число повторений возможно, когда RC = 80000000h. Это соответствует 80000001h 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 остаётся установленным в 1. Если записать 0 в счётчик повторений или разряд RM регистра состояния, можно остановить цикл повторений до его завершения.

Пример 5.38 – Алгоритм управления режимом повторения

```
if RM == 1                          ; Если в режиме повторения (RPTB или RPTS)
   if S == 1                        ; если RPTS
     if первый раз                  ; Если это первая выборка
        выборка команды из памяти   ; Выборка команды из памяти
     else                           ; Если не первая выборка
        выборка команды из IR       ; Выборка команды из IR
    RC − 1 → RC                     ; Уменьшение RC
     if RC < 0                      ; Если RC отрицательный
                                    ; Завершение режима повторений одной команды
       0 → ST(RM)                   ; Выключить разряд режима повторений
       0 → S                        ; Очистить S
       PC+1 → PC                    ; Увеличение PC
     else if S == 0                 ; Если RPTB
        выборка команды из памяти   ; Выборка команды из памяти
    if PC == RE                     ; Если это конец блока
        RC − 1 → RC                 ; Уменьшение RC
    if RC → 0                       ; Если RC не отрицательный
        RS → PC                     ; Устанавливает PC в начало блока
    else if RC < 0                  ; Если RC отрицательно
        0 → ST(RM)                  ; Выключить разряды режима повторений
        0 → S                       ; Очистить S
        PC+1 → PC                   ; Увеличение PC
; Очистить S
; Увеличение PC
```

5.7.1.3 RPTB и RPTBD инструкции

Инструкции RPTB и RPTBD повторяют блок программного кода определённое число раз. RPTBD – задержанная форма RPTB инструкции, которая позволяет выполнить ещё три следующие за ней инструкции. Эти три инструкции не являются частью блока повторений, но они выполняются перед началом повтора. Поэтому конвейер остаётся заполненным, и RPTBD инструкция исполняется за один цикл.

Число повторений блока команд равно значению регистра счётчика повторений плюс один 1. Так как RPTB и RPTBD не загружают RC, вы должны сами записывать в него нужное значение. Загрузка значения в RC должна происходить перед выполнением инструкций RPTB/RPTBD. Загрузка регистра RC не может быть одной из трёх инструкций, следующих за RPTBD. В примере 5.39 приведён типовой вариант программирования повтора блока.

Пример 5.39 – Выполнение RPTB

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

Все блоки повторений, инициализированные с помощью RPTB и RPTBD, могут быть прерваны. Однако прерывания невозможны в течение выполнения трёх инструкций, следующих за RPTBD. Ни одна из этих трёх инструкций не может изменить регистр PC или ход выполнения программы. Это ограничение также применимо к задержанным переходам, которые описаны в 5.7.2.

Во время выполнения RPTB src или RPTBD src производится пять последовательных операций:

1) загрузка начального адреса повторений в RS:
   - для RPTB – это адрес, следующий за инструкцией: PC или RPTB + 1 → RS;
   - для RPTBD – это четвёртый адрес, следующий за инструкцией: PC или RPTBD + 4 → RS;
2) загрузка конечного адреса повторений в RE:
   - для RPTB в относительном режиме 24-разрядный операнд плюс RS: src + PC для RPTB + 1 → RE;
   - для RPTBD в относительном режиме 24-разрядный операнд плюс RS: src + PC для RPTBD + 3 → RE;
3) в регистровом режиме, содержимое регистра src – конечный адрес:
   содержимое регистра src → RE;
4) установка регистра состояния для индикации режима повторений:
   1 → RM разряд регистра состояния (флаг режима повторений);
5) индикация – это операция режима повторений:
   0 → S разряд (внутренний для процессора, не программируется).

5.7.1.4 RPTS инструкция

RPTS src инструкция повторяет инструкцию, следующую за RPTS, (src + 1) раз. Повторение одной инструкции, инициализированной с помощью RPTS, не может быть прервано, так как RPTS выбирает код команды только один раз и далее хранит его в регистре инструкций для повторного использования. Прерывание в этом случае может привести к потере командного слова. Повторная выборка командного слова из регистра команд уменьшает количество обращений к памяти и, в результате, действует как программный кэш на одно слово. Если необходимо, чтобы инструкция повторялась и при этом могла бы быть прервана, необходимо использовать RPTB/RPTBD инструкции.

Во время выполнения RPTS src производится следующая последовательность операций:

1) PC + 1 → RS;
2) PC + 1 → RE;
3) 1 → разряд RM регистра состояния;
4) 1 → бит S;
5) src → RC (регистр счётчика повторов).

Команда RPTS загружает все регистры и разряды режима, необходимые для операции повторения одной команды. Операция 1 загружает начальный адрес блока в RS. Операция 2 загружает конечный адрес в RE (конечный адрес блока). Так как это повторение одной команды, то начальный и конечный адреса совпадают. Операция 3 устанавливает регистр состояния для указания режима повторения операции. Операция 4 указывает, что это режим повторения одной команды. Далее операнд src загружается в RC (Операция 5).

5.7.1.5 Ограничивающие правила в режиме повторений

Так как режимы повторений изменяют счётчик команд, другие инструкции не могут изменять счётчик команд в то же самое время. Поэтому применяется два правила.

Правило 1. Блок повторений не может завершаться инструкциями Bcond, DBcond, CALL, CALLcond, TRAPcond, RETIcond, RETScond, IDLE, RPTB или RPTS. Пример 5.40 иллюстрирует некорректное использование стандартных переходов.

Правило 2. Ни одной из 4 последних инструкций в блоке повторений не может быть BcondD, BRD, DBcondD, RPTBD, LAJ, LAJcond, BcondAF, BcondAT, RETIcondD. Пример 5.41 иллюстрирует некорректное использование задержанных переходов.

Пример 5.40 – Некорректное использование стандартных переходов

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

Пример 5.41 – Некорректное использование задержанных переходов

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

            BRD     OOPS            ; нарушение правила 2
            ADDF
            MPYF
ENDLOP      SUBF
```

5.7.1.6 Значение RC регистра 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/RPTBD с размером блока, равным 1, возникает конфликт конвейера, RC регистр уменьшается до 0000 0000h. Пример 5.42 иллюстрирует конфликт конвейера.

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

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

Пример 5.42 – Конфликт конвейера при выполнении инструкции RPTB

```
EDC     .word 40000000h         ; программа расположена в 4000000Fh
        LDP     EDC
        LDI     @EDC, AR0
        LDI     15, RC          ; загрузка 15 в счётчик повторений
        RPTB    ENDLOP          ; выполнение блока кода
ENDLOP  LDI     *AR0, R0        ; чтение *AR0 конфликтует с выбором
                                ; инструкции. RC уменьшается до 0
                                ; если кэш разрешён, RC уменьшается
                                ; до FFFF FFFFh
```

5.7.1.7 Вложенность блоков повторений

Блоки повторений RPTB и RPTBD могут либо иметь вложения или сами являться таковыми. Так как управление блоками повторений зависит от RS, RE, RC, ST регистров, они должны хранить информацию о вложенных блоках повторений. Например, если вы используете программу, работающую с прерываниями с использованием RPTB или RPTBD, 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, 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, вам не требуется сохранять и восстанавливать значения этих регистров.

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

Так как счётчик команд изменяется в конце блока в соответствии с содержимым регистров RS, RE, RC, не должно производиться никаких операций в конце блока, способных изменить счётчик повторений или счётчик команд.

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

Примечание – Для обеспечения корректной работы очень важен порядок использования регистров, в которых осуществляется запись/чтение. Значение ST регистра должно восстанавливаться в конце после RC, RE, RS регистров. Значение ST регистра должно восстанавливаться после восстановления RC, так как разряд RM может быть не установленным в единицу, если RC регистра содержится 0 или –1. По этой причине, если вы используете инструкцию POP ST (где ST(RM) = 1) пока RC = 0, инструкция POP перезаписывает все разряды ST регистра за исключением разряда RM, который остаётся равным 0 (режим повторений отключен). Также RS и RE должны быть корректно установлены перед включением режима повторений.

5.7.2 Задержанные переходы

В ядре процессора 1867ВЦ8Ф1 имеются два основных типа ветвлений: стандартные переходы и задержанные переходы. Стандартные переходы освобождают конвейер до выполнения перехода, чтобы гарантировать корректное управление программным счётчиком. В результате ветвление программы в процессорном ядре 1867ВЦ8Ф1 выполняется за четыре цикла. В этот класс включены стандартные переходы Bcond, повторения, вызовы, возвраты и программные прерывания.

Задержанные переходы не освобождают конвейер, но также гарантируют, что следующие три команды будут выполнены до того, как программный счётчик будет изменён ветвлением. Результатом является переход, требующий только одного цикла, таким образом, скорость работы задержанного перехода очень близка к оптимальному режиму повторения блоков на ядре процессора 1867ВЦ8Ф1. 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, которые возникают в конце команды, непосредственно предшествующей задержанному переходу. Флаги условий не зависят от команд, следующих за задержанным переходом. Время выполнения задержанного перехода остаётся прежним, несмотря на то, произошло ветвление или нет.

Когда задержанные переходы выбраны, они остаются задержанными до тех пор, пока три следующие команды не будут выполнены. Ни одна из трёх команд, следующих за задержанным переходом, не может быть Bcond, BcondD, BcondAF, BcondAT, BR, BRD, DBcond, DBcondD, CALL, CALLcond, IDLE, LAJ, LAJcond, LATcond, RETIcond, RETIcondD, RETScond, RPTB, RPTBD, RPTS или TRAPcond. Это ограничение также применяется для инструкции RPTBD, описанной в 5.7.1.3.

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

При некорректном использовании задержанных переходов PC будет не определен!

Пример 5.43 показывает некорректное размещение задержанного перехода.

Пример 5.43 – Некорректное размещение задержанного перехода.

```
B1:     BD      L1
        NOP
        NOP
B2:     B       L2      ;  Этот переход размещён некорректно
        NOP
                NOP
        NOP
        .
        .
```

Иногда в программе необходим переход, после которого должно выполняться меньше трёх команд. В этом случае задержанный переход также обеспечивает наилучшее быстродействие. Это отражено в примере 5.44, где вместо третьей неиспользуемой инструкции применяется NOP.

Пример 5.44 – 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.

5.7.2.1 Задержанные переходы без аннулирования

Задержанные переходы без аннулирования не очищают конвейер, но гарантируют выполнение следующих за переходом трёх инструкций перед тем, как ветвление изменит значение счётчика команд. Задержанные переходы без аннулирования: BcondD, BRD, DbcondD.

5.7.2.2 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, исполняются три следующие за BcondAF инструкции, и, затем, осуществляется переход. Если условие ложно, переход не осуществляется, отменяются фаза исполнения первой инструкции, следующей за переходом, а также фазы чтения и исполнения следующих второй и третьей инструкций.

BcondAT − если условие истинно, осуществляется переход, отменяются фаза исполнения первой инструкции, следующей за переходом, а также фазы чтения и исполнения следующих второй и третьей инструкций. Если условие ложно, исполняются три следующие за BcondAT инструкции, но переход не осуществляется.

5.7.3 Вызовы, программные прерывания, ветвления, скачки и возвраты

Вызовы и программные прерывания обеспечивают выполнение подпрограммы или функции до возврата в вызывавшую программу.

Команды 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 команд RETScond или RETIcond.

CALL – четырёхцикловая инструкция, в то время как CALLcond и Trapcond – пятицикловые. Трём вышеуказанным инструкциям по выполняемым функциям эквивалентны в соответствующем порядке LAJ, LAJcond, LATcond, которые являются одноцикловыми:

- Команда CALL помещает значение следующего PC в стек и помещает операнд src в PC. Src − 24-разрядное непосредственное значение. На рисунке 5.58 приведена временная диаграмма обработки команды CALL.

- Команда CALLcond подобна команде CALL, за исключением того, что:

  - она выполняется, только если определённое условие есть «истина» (20 условий, включая безусловные, приведены в приложении А);

  - src является либо 24-разрядным относительным (относительно PC) смещением, либо определённым регистровым режимом адресации;

  - команда TRAPcond также выполняется, только если определённое условие есть «истина» (условия те же, что и для CALLcond). При выполнении:

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

  - прерывания запрещаются (GIE = 0) и кэш «замораживается» (CF = 0);

  - следующее значение PC сохраняется в стек;

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

- Использование RETIcond или RETIcondD для возврата повторно разрешает прерывание, если разряд GIE регистра состояния был раньше установлен, а также перезаписывает разряд CF.

- RETScond возвращает выполнение от одной из вышеперечисленных команд, загружая вершину стека в PC. Команда исполняется, если заданное в ней условие – «истина». Условия те же, что и для CALLcond.

- RETIcond возвращает из программного прерывания или вызова аналогично команде RETScond и, кроме того, копирует разряды PGIE и PCF в GIE и CF в регистре состояния. Условия те же, что и для CALLcond.

- RETIcondD возвращает из программного прерывания или вызова аналогично команде RETIcond, но, кроме этого, в первую очередь выполняет три инструкции, следующие непосредственно за RETIcondD. Условия те же, что и для CALLcond.

- LAJ, LAJcond, LATcond обеспечивают возвращение адреса в регистр повышенной точности R11:

  - после выполнения трёх инструкций, следующих за командой скачка, LAJ обеспечивает скачок на адрес, определённый в режиме 24-разрядного относительного смещения (см. 5.6.6);

  - конечный адрес скачка LAJcond также определяется режимом относительного смещения или регистровой адресации. Если условие истинно, LAJcond в первую очередь выполняет три инструкции, следующие непосредственно за LAJcond, перед тем как совершить скачок. Если условие ложно, выполнение программы продолжается сразу после инструкции LAJcond;

  - после выполнения трёх инструкций, следующих за командой LATcond, вызывается один из 512 векторов таблицы программных прерываний в соответствии с TVTP (см. 5.3.2).

Функционально вызовы и программные прерывания предназначены для одной и той же цели – вызывать и выполнять подпрограмму, а затем 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 прерывания. Таким образом, программные прерывания обычно устраняются с помощью 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 рекомпиляции каждого приложения.

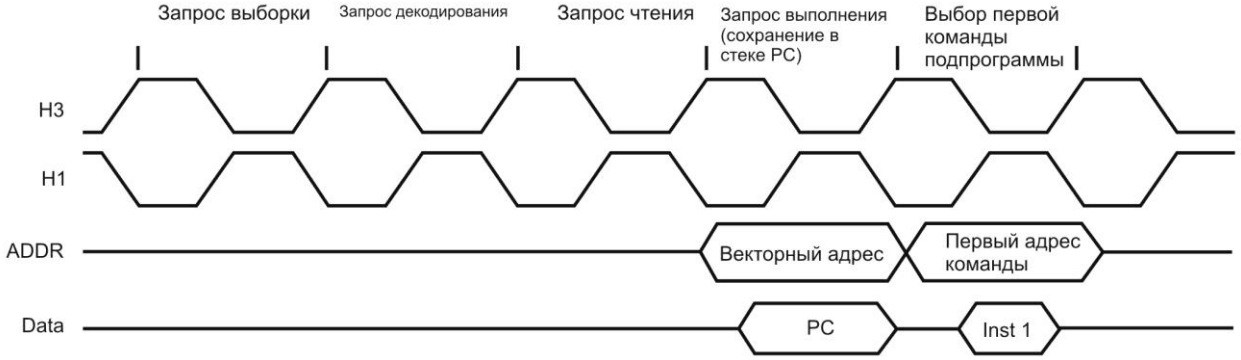


Рисунок 5.58 – Временная диаграмма команды CALL

5.7.4 Прерывания

Ядро процессора 1867ВЦ8Ф1 поддерживает несколько внутренних и внешних прерываний, которые могут использоваться в различных приложениях. Внутренние прерывания генерируются контроллером ПДП, таймерами, коммуникационными портами. Пять внешних сигналов прерываний включают четыре внешних маскируемых прерывания IIOF0# – IIOF3# и одно немаскируемое прерывание NMI#. Сигналы прерываний могут посылаться к ЦПУ и контроллеру ПДП.

В процессорных ядрах ИС 1867ВЦ8Ф1 прерывания автоматически приоритезируются. Это позволяет обрабатывать прерывания, которые происходят одновременно, в заранее определённом порядке. В нижеследующем разделе обсуждается действие данных прерываний.

5.7.4.1 Векторная таблица системных прерываний и приоритеты

Векторная таблица системных прерываний IVT изображена на рисунке 5.59. Вектор прерывания – адрес программы, обслуживающей прерывание, 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. 5.3.2).

Приоритеты означают, что прерывания в старшей позиции векторной таблицы прерываний обслуживаются раньше, чем в низшей 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_x# должен ожидать, пока будут обработаны IIOF2_x#, IIOF1_x#, IIOF0_x# (когда ST(GIE) = 1).

Приоритеты прерываний устанавливаются ЦПУ в соответствии с их позициями в векторной таблице прерываний, начиная со старшего (т. е. приоритет NMI# старше TINT0, который, в свою очередь, старше IIOF0_x# и т. д.). Заметьте, что прерывание TINT0 располагается в IVTP+2, в то время как TINT1 – в IVTP+2Bh после прерываний коммуникационных портов и сопроцессора ПДП.

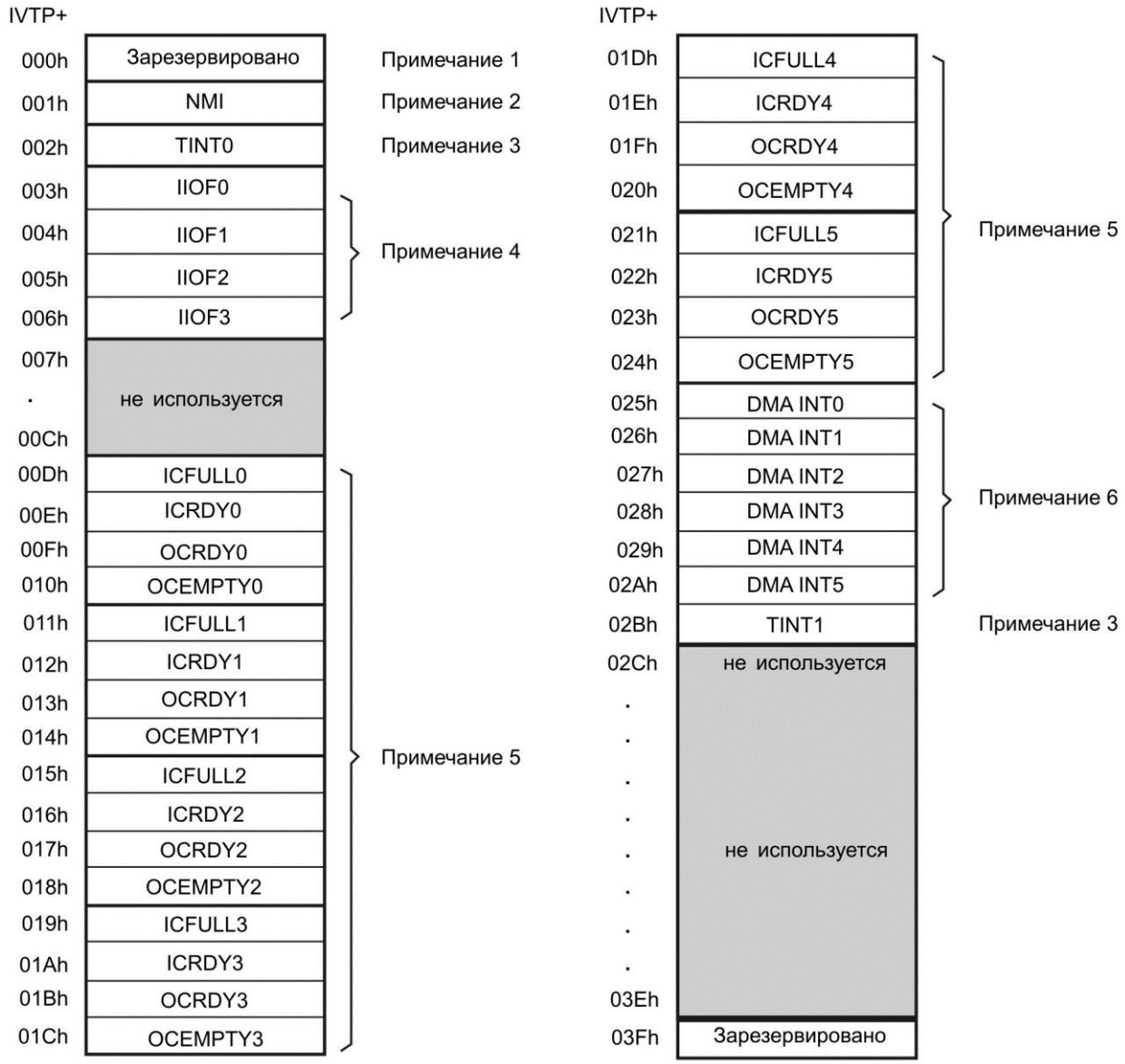


Рисунок 5.59 − Векторная таблица системных прерываний

Примечания

1 Зарезервировано для вектора сброса. См. таблицу 5.20.

2 NMI# (немаскируемое прерывание) описано в 5.7.4.5.

3 Прерывания таймеров TINT0 и TINT1 разрешаются с помощью регистра IIE (см. 5.3.1.9) и отражаются в регистре IIF (см. 5.3.1.10).

4 Внешние сигналы IIOF(0 – 3)_x# программируются в IIF регистре (см. 5.3.1.10).

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

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

5.7.4.2 Разряды управления прерыванием ЦПУ

Регистры ЦПУ содержат разряды для управления прерыванием ЦПУ:

- Регистр состояния ЦПУ(ST). Разряд разрешения глобальных прерываний ЦПУ GIE регистра состояния ST управляет всеми маскируемыми прерываниями ЦПУ. Если разряд установлен в 1, глобальные прерывания ЦПУ разрешены. Если разряд сброшен в 0, все прерывания ЦПУ запрещены (исключая NMI_x# – немаскируемое прерывание), см. 5.3.1.7.

- Регистр разрешения внутренних прерываний IIE – используется для разрешения внутренних системных прерываний (от таймеров, коммуникационных портов и каналов ПДП), см. 5.3.1.9.

- Регистр флагов IIOF (IIF). Регистр IIF содержит разряды флагов прерываний и разряды для определения функции внешних сигналов прерываний IIOF(0 – 3)_x#.

Регистр IIF

Когда происходят внешние прерывания или большинство внутренних прерываний, соответствующие разряды в регистре IIF устанавливаются в 1. Только внутренние прерывания от коммуникационного порта не имеют флагов в регистре IIF.

Когда ЦПУ обрабатывает прерывания, имеющие свой флаг в регистре IIF, или когда контроллер ПДП фиксирует такие прерывания во внутреннем сигнале ПДП, данный разряд флага сбрасывается в 0 с помощью внутреннего сигнала подтверждения прерывания. Однако для прерываний по уровню, если IIOF(0 – 3)_x# остаётся в низком уровне, когда приходит сигнал подтверждения прерывания, разряд 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, разряд флага прерывания может быть равен 0, даже если IIOF(0 – 3)_x# в низком уровне. После сброса в регистр флагов прерываний записывается 0, таким образом, сбрасывая все оставшиеся прерывания.

Чтение/запись разрядов регистра IIF может осуществляться программно. Это обеспечивает доступ к сигналам IIOF(0 – 3)_x#, 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 разряд FLAGx, можно писать во внешние IIOF(0 – 3)_x#. Если FUNCx = 1 (прерывание), то запись 1 в разряд FLAGx регистра IIF будет равнозначна приходу соответствующего прерывания. Поэтому все прерывания могут быть установлены и/или сброшены программно. Так как биты прерываний могут быть прочитаны, они могут быть опрошены программно, когда интерфейс управления прерываниями не требуется.

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

Пример 5.45. Изменение регистра IIF.

```
correct                 incorrect
LDI   @MASK, R0         LDI   IIF, R1
AND   R0, IIF           AND   @MASK, R1
                        LDI   R1, IIF
```

5.7.4.3 Выполнение прерываний

Чтобы произошло прерывание, необходимо два условия:

- все глобальные прерывания должны быть разрешены установкой разряда GIE регистра состояния ST в 0;

- прерывание должно быть разрешено установкой соответствующего разряда в регистре IIE.

Цикл выполнения прерывания (см. рисунок 5.60) включает в себя семь событий. Соответствующий флаг прерывания в регистре IIF сброшен, предыдущие значения разрядов GIE и CF регистра состояния сохранены, кэш «заморожен» (CF = 1), прерывания глобально запрещены (GIE = 0), ЦПУ выполняет все выбранные инструкции. Затем выбирается вектор прерывания и помещается в PC, ЦПУ продолжает выполнение с первой инструкции программы обработки прерывания ISR. Когда вы используете RETIcond или RETIcondD для возврата из подпрограммы обработки прерывания, предыдущие значения GIE и CF восстанавливаются.

Если вы хотите, чтобы программа обработки прерываний не прерывалась, необходимо установить GIE в 1 после входа в ISR. Также можно включить и кэш. Имейте в виду, что разряды 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 выполнения последней инструкции, которая была выбрана перед произошедшим прерыванием. Это гарантирует дальнейшее корректное восстановление значения флага.

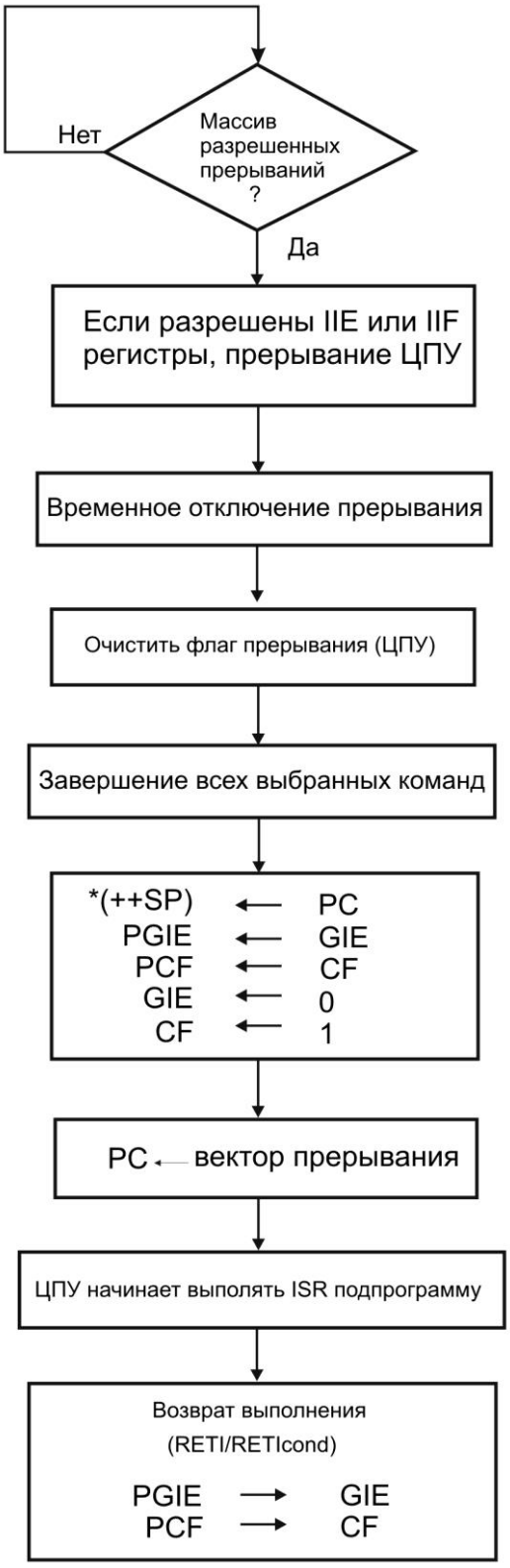


Рисунок 5.60 – Выполнение прерывания ЦПУ

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 внешней индикации обслуживания прерывания. Если в операнде указана внешняя память, IACK управляет сигналом IACK# и выполняет фиктивное чтение. Чтение выполняется из адреса, указанного операндом инструкции IACK. Обычно IACK располагается в начале программы обслуживания прерывания. Однако в зависимости от приложения, она может располагаться в конце программы обслуживания прерывания или в другом месте. Кроме того, не обязательно использовать инструкцию IACK в программе обработки прерывания.

Необходимо обратить внимание на следующие ситуации:

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

- Когда происходит прерывание, фазы декодирования и чтения инструкций продолжаются. Это не влияет на выборку инструкции:

- если прерывание произошло в первом цикле выборки инструкции, выбранная инструкция пропускается (не выполняется), а её адрес вталкивается в стек;

- если прерывание произошло после первого цикла выборки (в случае многоцикловой выборки до состояния ожидания), инструкция выполняется, а адрес инструкции, которая должна выбираться следующей, вталкивается в стек;

- если выборка команд в текущий момент времени не происходит, то и новая выборка не осуществляется.

5.7.4.4 Задержка прерывания

Задержка прерывания, определяемая как время от подтверждения прерывания до выполнения первой инструкции подпрограммы обработки прерывания, составляет не менее восьми машинных циклов. Это проиллюстрировано в таблице 5.18, где прерывание определено как инструкция, предполагая, что все инструкции одноцикловые.

Таблица 5.18 – Задержка 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	Временный запрет прерывания до очистки GIE. Очистка соответствующего флага IIF (если применимо)	–	прерывание	прог а	прог а – 1
3	Чтение таблицы векторов прерываний	–	–	прерывание	прог а
4	Сохранение адреса возврата в стек, разрядов GIE и CF в PGIE и PCF, соответственно. Очистка GIE и установка CF в 1	–	–	–	прерывание
5	Конвейер начинает заполняться инструкциями	isr1	–	–	–
6		isr2	isr1	–	–
7		isr3	isr2	isr1	–
8	Выполнение первой инструкции подпрограммы обработки прерывания	isr4	isr3	isr2	isr1

5.7.4.5 Внешние прерывания

Пять внешних сигналов прерывания включают в себя четыре внешних маскируемых прерывания IIOF0_x# – IIOF3_x# и одно немаскируемое NMI#.

Четыре внешних маскируемых прерывания разрешаются регистром IIF (см. 5.3.1.10) и синхронизируются внутренне. Они распознаются по заднему фронту H1 и проходят через серию внутренних задержек H1/H3. Будучи синхронизованным, вход прерывания устанавливает соответствующий разряд флага в регистре прерываний IIF, если появляется прерывание. Ниже перечислены внешние прерывания и соответствующие им вектора прерываний:

IIOF(0-3)_x# прерывание	Расположение вектора прерывания
IIOF0_x#	IVTP + 003h
IIOF1_x#	IVTP + 004h
IIOF2_x#	IVTP + 005h
IIOF3_x#	IVTP + 006h

Приоритеты прерываний определяются по старшинству в случае прихода двух прерываний в одном машинном цикле (IIOF0_x# – старший, IIOF1_x# − следующий и т. д.). Если получено прерывание, разряд регистра состояния ST(GIE) сбрасывается в 0, запрещая любые другие входящие прерывания, исключая NMI#. Это предотвращает предполагаемое программное управление любыми другими прерываниями IIOF0# – IIOF3# до того момента, пока ST(GIE) будет опять установлен в 1. В дополнение к этому разряд 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 фронтам, либо по уровню в зависимости от полей TYPE, установленных в регистре IIF (см. 5.3.1.10).

Для прерываний по фронтам внешние сигналы должны изменяться из 1 в 0. А затем должны удерживаться в низком уровне 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, оно может быть определено как несколько прерываний. В этом случае нет необходимости обеспечивать фронты.

Примечание − Прерывания по уровню не фиксируемые. Ядро процессора 1867ВЦ8Ф1 определяет их только, если низкий уровень присутствует в течение выборки-декодирования инструкций в конвейере. Это означает, что если конвейер находится в режиме ожидания, 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 разрядом ST(GIE). Хотя NMI# – немаскируемое прерывание, 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 осторожным при использовании NMI# в качестве прерывания второго уровня. Когда ядро процессора 1867ВЦ8Ф1 обслуживает прерывание, другие прерывания запрещаются, за исключением NMI#. Это представляет проблему, так как в регистр состояния ST может быть записано неверное значение, если NMI# выполнится перед первой инструкцией ISR, при которой предыдущее значение ST регистра сохраняется.

Ядро процессора 1867ВЦ8Ф1 поддерживает программно конфигурируемую функцию, которая позволяет принудительно подготовить внутреннюю периферийную шину, при установке сигнала NMI#. NMI# разрешение передачи по шине включается, если разряды 18 и 19 регистра состояния ST установлены в 10_2. Если оно включено, сигнал разрешения передачи по периферийной шине генерируется по заднему фронту NMI#. Если NMI# установлен, но не разрешён, у ЦПУ остаётся доступ к периферийной шине, если шина не готова. Данное условие возникает, если происходит запись в заполненный выходной буфер FIFO или чтение из пустого входного буфера FIFO. Эта функция применяется при коррекции ошибок коммуникационного порта, когда он используется в сочетании с его программным сбросом.

5.7.5 Программные прерывания

В ядре процессора 1867ВЦ8Ф1 программные и системные прерывания обрабатываются одинаково, но по-разному инициализируются.

5.7.5.1 Инициализация программных и системных прерываний

Программные и системные прерывания инициализируются по-разному.

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

Системные прерывания всегда устанавливаются аппаратно (например, внешними прерываниями, прерываниями ПДП или коммуникационного канала).

По этой причине разряд GIE в регистре состояния ST и маскируемые разряды в IIE не применимы к программным прерываниям.

5.7.5.2 Операции программных прерываний

На рисунке 5.61 изображён ход программных прерываний (а также системных).

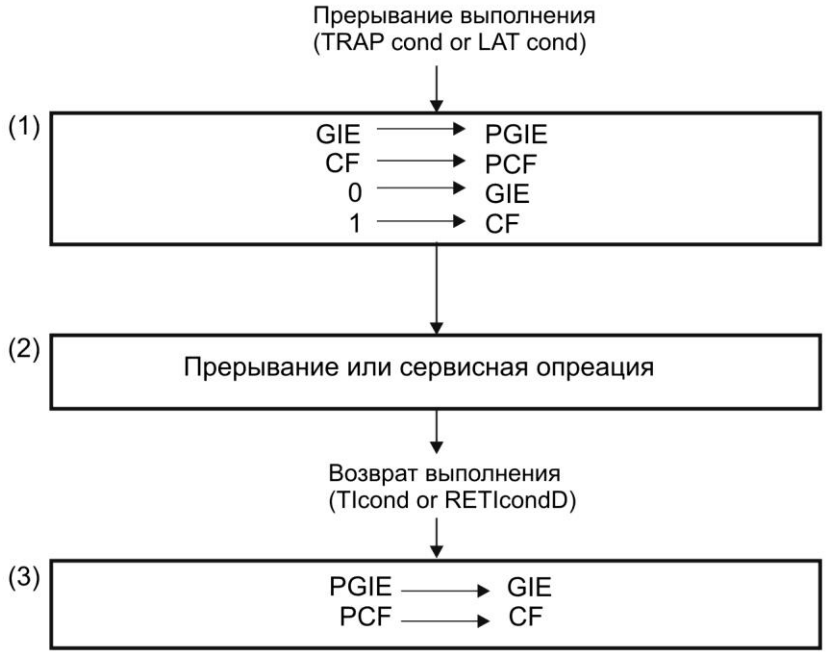


Рисунок 5.61 − Ход программных прерываний

Инструкции RETIcond и RETIcondD оперируют флагами состояния как показано в блоке (3) на рисунке 5.61 RETIcond/RETIcondD обеспечивают возврат/задержанный возврат из программного прерывания.

Как правило, не возникает необходимости напрямую изменять разряды PGIE или PCF регистра состояния за исключением ситуации, когда значение регистра состояния помещается в стек для циклических системных или программных прерываний.

Ядро процессора 1867ВЦ8Ф1 поддерживает 512 различных программных прерываний. Когда выполняются инструкции TRAPcond n или LATcond n, ядро процессора 1867ВЦ8Ф1 переходит по адресу, хранящемуся в области памяти, указанной TVTP+n, где TVTP – регистр-указатель таблицы программных прерываний. 32-разрядный регистр TVTP содержит начальный адрес векторной таблицы. Эта таблица представлена на рисунке 5.62.

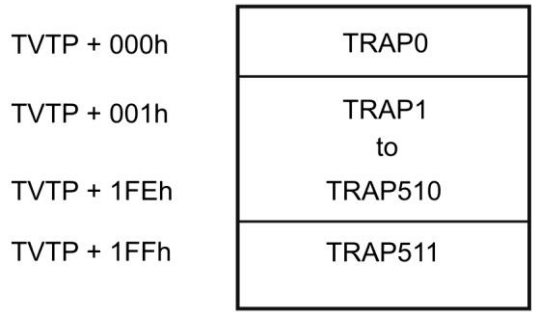


Рисунок 5.62 − Таблица программных прерываний

Как и таблица системных прерываний IVT, таблица программных прерываний TVT занимает область памяти в 512 слов. Регистр-указатель TVT указывает на начало TVT (см. 5.3.2).

Инструкции TRAP или LATcond могут использоваться для генерации программных прерываний и манипулирования флагами состояния, как показано на рисунке 5.61. Инструкция LATcond обеспечивает одноцикловое программное прерывание, которое полезно для нахождения и исправления ошибок.

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

5.7.5.3 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 вектора программных прерываний должны располагаться там, где нет векторов системных прерываний. Например, если вектор 02Ch не используется, можно разместить вектор программного прерывания в IVTP+02Ch (который, если таблицы перекрываются, в свою очередь, является также TVTP+02Ch), а затем вызвать программное прерывание, указав 02Ch в инструкции TRAP.

5.7.6 Прерывания ПДП

Прерывания могут устанавливать операции чтения и записи ПДП. Это называется синхронизация ПДП. Цикл выполнения прерывания ПДП аналогичен циклу прерывания ЦПУ. После сброса соответствующего флага прерывания, сопроцессор ПДП работает в соответствии с разрядами SYNC глобального регистра управления сопроцессора ПДП.

Если прерывание разрешено регистром разрешения прерываний ПДП (DIE), контроллер прерывания автоматически фиксирует его и сохраняет для дальнейшего использования ПДП. Что касается флагов прерываний (таймера, внешнего прерывания), 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 разряды регистра контроля ПДП сброшены в 00_2 в разрядах START (AUX START). Системный сброс ПДП сбрасывает внутренние фиксированные прерывания.

5.7.6.1 Разряды управления прерываниями ПДП

Два регистра содержат разряды управления операциями прерывания ПДП:

- регистр разрешения прерываний ПДП (DIE). Все прерывания ПДП управляются разрядами DIE регистра и разрядами SYNC регистра управления каналом ПДП (см. рисунок 5.109). Прерывания ПДП не зависят от ST(GIE) и локализованы для ПДП;
- регистр управления каналом ПДП. Каждый канал сопроцессора ПДП использует регистр управления каналом для определения режима работы. Этот регистр изображён на рисунке 5.109.

DIE разбит на шесть полей, которые определяют, какое прерывание будет использовано для управления синхронизацией каждого из 6 каналов ПДП. Например, разряды в каждом поле позволяют выбрать, будет ли ПДП синхронизован с коммуникационным портом или с таймером, или с внешним сигналом прерывания.

5.7.6.2 Процесс прерывания ПДП

Рисунок 5.63 показывает выполнение процесса прерывания ПДП. Для более подробной информации см. 5.11.10.

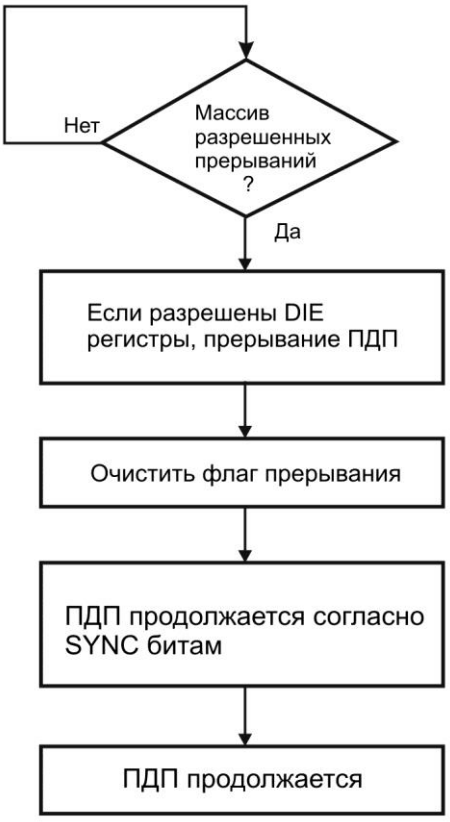


Рисунок 5.63 − Процесс прерывания ПДП

5.7.6.3 Взаимодействие ЦПУ/ПДП прерываний

Сопроцессор ПДП ядра процессора 1867ВЦ8Ф1 не затрагивается обработкой прерываний ЦПУ, даже когда ПДП использует прерывания для синхронизации передачи. Кроме этого, ПДП не затрагивается, даже когда конвейер остановлен.

Ядро процессора 1867ВЦ8Ф1 позволяет ЦПУ и контроллеру ПДП выполнять прерывания и отвечать на них параллельно. Рисунок 5.64 показывает последовательность событий при обработке прерываний для ЦПУ и контроллера ПДП. Точная последовательность событий изложена в таблице 5.18.

Таким образом, возможно, прервать ЦПУ и контроллер ПДП одновременно одинаковыми или разными прерываниями и, в сущности, их синхронизировать. Однако так как сопроцессор ПДП и ЦПУ делят между собой установку флагов прерываний, 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, ПДП может фиксировать прерывание и сбрасывать соответствующий флаг. Если прерывание разрешено в регистре DIE, ЦПУ никогда не «украдёт» прерывание ПДП, потому что ПДП отвечает на прерывание либо с такой же скоростью, как и ЦПУ, либо быстрее.

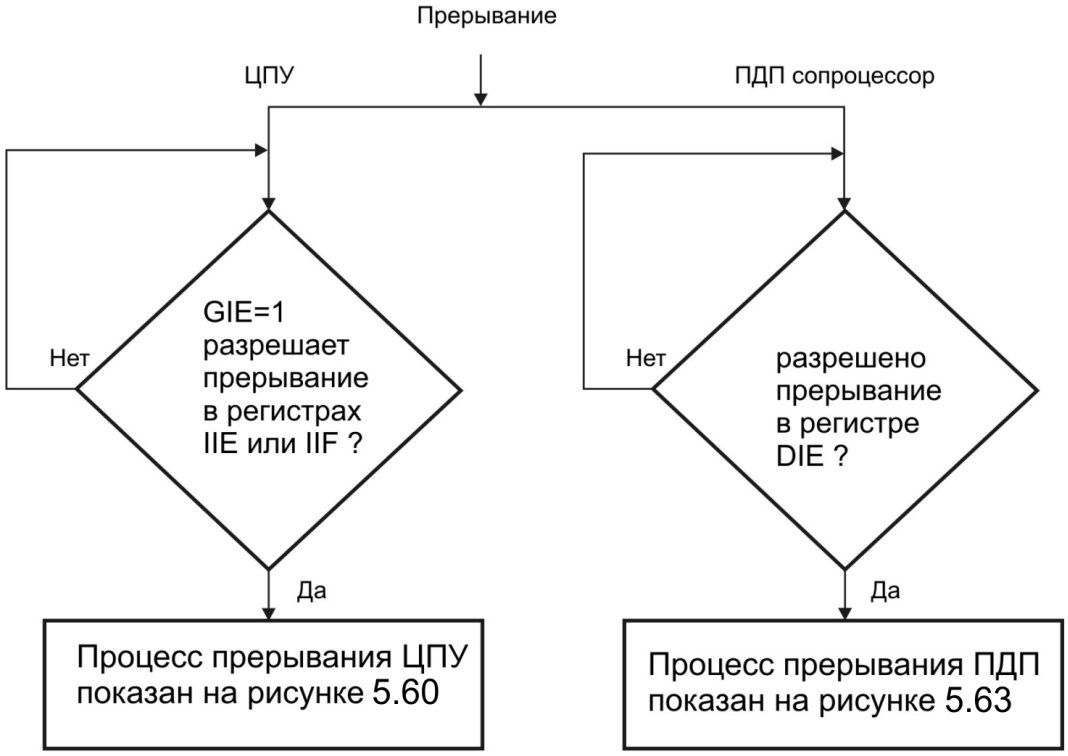


Рисунок 5.64 − Параллельная обработка прерываний ЦПУ и ПДП

5.7.7 Системный сброс

Ядро процессора 1867ВЦ8Ф1 поддерживает немаскируемый внешний сигнал системного сброса RESET#, который используется для выполнения системного сброса системы. Далее описана операция системного сброса.

После включения питания состояние ядра процессора 1867ВЦ8Ф1 не определено. Можно использовать сигнал RESET# для установки процессора в известное состояние. Сигнал должен быть установлен в низком уровне в течение 10 или более циклов H1 для гарантированного системного сброса. H1 – выходной тактовый сигнал, генерируемый ядром процессора 1867ВЦ8Ф1.

Сброс оказывает влияние на:

- некоторые выводы устройства;
- некоторые регистры устройства;
- выполнение программы.

5.7.7.1 Влияние системного сброса на состояние выводов

Системный сброс влияет на состояние выводов устройства синхронно или асинхронно. Синхронный сброс происходит посредством внутренних тактов от генератора ядра процессора 1867ВЦ8Ф1. Асинхронный сброс напрямую воздействует на состояние выводов, и он быстрее, чем синхронный сброс.

Таблица 5.19 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 5.19 − Состояние сигнальных выводов после системного сброса

(а) Тактовые сигналы (5 выводов)

Сигнал	Выводы	I/O**	Тип*	Описание
H1_x	2	O	S	Начинает тактировать, когда RESET# изменяется из 1 в 0
H3_x	2	O	S	Начинает тактировать, когда RESET# изменяется из 1 в 0
X2_CLKIN_COMM	1	I	–	Нет эффекта

Примечания

1 x = 1, 2.

2 Рекомендуемые конденсаторы, подключаемые к устройству, кратны 0,1 и 4,7 мкФ. Их количество зависит от уровня шумов на плате.

_____

* S – синхронный.

** I – вход, O – выход, I/O – вход/выход.

(б) Интерфейс коммуникационного порта 0 (12 выводов)

Сигнал	Выводы	I/O**	Тип*	Описание
C0D(7−0)_1	8	I/O	S	Устанавливается в неопределённое значение
CACK0_1#	1	I/O	A	Устанавливается в Z, когда сброс переходит в низкий уровень, а затем устанавливается в 1, когда сброс переходит в высокий уровень
CRDY0_1#	1	I/O	A	Устанавливается в Z
CREQ0_1#	1	I/O	A	Устанавливается в Z
CSTRB0_1#	1	I/O	A	Устанавливается в Z, когда сброс переходит в низкий уровень, а затем устанавливается в 1, когда сброс переходит в высокий уровень

Примечание – Рекомендуемые конденсаторы, подключаемые к устройству, кратны 0,1 и 4,7 мкФ. Их количество зависит от уровня шумов на плате.

_____

* A – асинхронный, S – синхронный.

** I/O – вход/выход.

(в) Интерфейс коммуникационного порта 1 (24 вывода)

Сигнал	Выводы	I/O**	Тип*	Описание
C1D(7−0)_x	16	I/O	S	Устанавливается в неопределённое значение
CACK1_x#	2	I/O	A	Устанавливается в Z, когда сброс переходит в низкий уровень, а затем устанавливается в 1, когда сброс переходит в высокий уровень
CRDY1_x#	2	I/O	A	Устанавливается в Z
CREQ1_x#	2	I/O	A	Устанавливается в Z
CSTRB1_x#	2	I/O	A	Устанавливается в Z, когда сброс переходит в низкий уровень, а затем устанавливается в 1, когда сброс переходит в высокий уровень

Примечания

1 x = 1, 2.

2 Рекомендуемые конденсаторы, подключаемые к устройству, кратны 0,1 и 4,7 мкФ. Их количество зависит от уровня шумов на плате.

_____

* A – асинхронный, S – синхронный.

** I/O – вход/выход.

Продолжение таблицы 5.19

(г) Интерфейс коммуникационного порта 2 (24 вывода)

Сигнал	Выводы	I/O**	Тип*	Описание
C2D(7−0)_x	16	I/O	S	Устанавливается в неопределённое значение
CACK2_x#	2	I/O	A	Устанавливается в Z, когда сброс переходит в низкий уровень, а затем устанавливается в 1, когда сброс переходит в высокий уровень
CRDY2_x#	2	I/O	A	Устанавливается в Z
CREQ2_x#	2	I/O	A	Устанавливается в Z
CSTRB2_x#	2	I/O	A	Устанавливается в Z, когда сброс переходит в низкий уровень, а затем устанавливается в 1, когда сброс переходит в высокий уровень

Примечания

1 x = 1, 2.

2 Рекомендуемые конденсаторы, подключаемые к устройству, кратны 0,1 и 4,7 мкФ. Их количество зависит от уровня шумов на плате.

_____

* A – асинхронный, S – синхронный.

** I/O – вход/выход.

(д) Интерфейс коммуникационного порта 3 (12 выводов)

Сигнал	Выводы	I/O**	Тип*	Описание
C3D(7−0)_2	8	I/O	S	Устанавливается в неопределённое значение
CACK3_2#	1	I/O	A	Устанавливается в Z, когда сброс переходит в низкий уровень, а затем устанавливается в 1, когда сброс переходит в высокий уровень
CRDY3_2#	1	I/O	A	Устанавливается в Z
CREQ3_2#	1	I/O	A	Устанавливается в Z
CSTRB3_2#	1	I/O	A	Устанавливается в Z, когда сброс переходит в низкий уровень, а затем устанавливается в 1, когда сброс переходит в высокий уровень

Примечание – Рекомендуемые конденсаторы, подключаемые к устройству, кратны 0,1 и 4,7 мкФ. Их количество зависит от уровня шумов на плате.

_____

* A – асинхронный, S – синхронный.

** I/O – вход/выход.

(е) Интерфейс коммуникационного порта 4 (24 вывода)

Сигнал	Выводы	I/O**	Тип*	Описание
C4D(7−0)_x	16	I/O	S	Устанавливается в неопределённое значение
CACK4_x#	2	I/O	A	Устанавливается в Z, когда сброс переходит в низкий уровень, а затем устанавливается в 1, когда сброс переходит в высокий уровень
CRDY4_x#	2	I/O	A	Устанавливается в Z
CREQ4_x#	2	I/O	A	Устанавливается в Z
CSTRB4_x#	2	I/O	A	Устанавливается в Z, когда сброс переходит в низкий уровень, а затем устанавливается в 1, когда сброс переходит в высокий уровень

Примечания

1 x = 1, 2.

2 Рекомендуемые конденсаторы, подключаемые к устройству, кратны 0,1 и 4,7 мкФ. Их количество зависит от уровня шумов на плате.

_____

* A – асинхронный, S – синхронный.

** I/O – вход/выход.

Продолжение таблицы 5.19

(ж) Интерфейс коммуникационного порта 5 (24 вывода)

Сигнал	Выводы	I/O**	Тип*	Описание
C5D(7−0)_x	16	I/O	S	Устанавливается в неопределённое значение
CACK5_x#	2	I/O	A	Устанавливается в Z, когда сброс переходит в низкий уровень, а затем устанавливается в 1, когда сброс переходит в высокий уровень
CRDY5_x#	2	I/O	A	Устанавливается в Z
CREQ5_x#	2	I/O	A	Устанавливается в Z
CSTRB5_x#	2	I/O	A	Устанавливается в Z, когда сброс переходит в низкий уровень, а затем устанавливается в 1, когда сброс переходит в высокий уровень

Примечания

1 x = 1, 2.

2 Рекомендуемые конденсаторы, подключаемые к устройству, кратны 0,1 и 4,7 мкФ. Их количество зависит от уровня шумов на плате.

* A – асинхронный, S – синхронный.

** I/O – вход/выход.

(з) Эмуляция (7 выводов)

Сигнал	Выводы	I/O*	Тип	Описание
EMU0_COMM	1	I/O	–	Не определено
EMU1_COMM	1	I/O	–	Не определено
TCK_COMM	1	I	–	Нет эффекта
TDI_COMM	1	I	–	Нет эффекта
TDO_COMM	1	O	–	Нет эффекта
TMS_COMM	1	I	–	Нет эффекта
TRST_COMM#	1	I	–	Нет эффекта

Примечание – Рекомендуемые конденсаторы, подключаемые к устройству, кратны 0,1 и 4,7 мкФ. Их количество зависит от уровня шумов на плате.

* I – вход, O – выход, I/O – вход/выход.

(и) Внешний интерфейс глобальной шины (160 выводов)

Сигнал	Выводы	I/O**	Тип*	Описание
A(30−0)_x	62	O/Z	S	Устанавливается в Z
AE_x#	2	I	–	Нет эффекта
CE0_x#	2	I	–	Нет эффекта
CE1_x#	2	I	–	Нет эффекта
D(31–0)_x	64	I/O/Z	S	Устанавливается в Z
DE_x#	2	I	–	Нет эффекта
LOCK_x#	2	O	S	Устанавливается в 1
PAGE0_x#	2	O/Z	S	Устанавливается в 0
PAGE1_x#	2	O/Z	S	Устанавливается в 0
RDY0_x#	2	I	–	Нет эффекта
RDY1_x#	2	I	–	Нет эффекта
R/W0_x#	2	O/Z	S	Устанавливается в 1
R/W1_x#	2	O/Z	S	Устанавливается в 1
STAT(3−0)_x	8	O	S	Все устанавливаются в 1
STRB0_x#	2	O/Z	S	Устанавливается в 1
STRB1_x#	2	O/Z	S	Устанавливается в 1

Примечания

1 x = 1, 2.

2 Рекомендуемые конденсаторы, подключаемые к устройству, кратны 0,1 и 4,7 мкФ. Их количество зависит от уровня шумов на плате.

* S – синхронный.

** I – вход, O – выход, I/O – вход/выход, Z – состояние высокого импеданса 1111111.

Окончание таблицы 5.19

(к) Прерывания, флаги ввода-вывода, сброс, таймер, управляющие сигналы (24 вывода)

Сигнал	Выводы	I/O**	Тип*	Описание
IACK_x#	2	I	S	Устанавливается в 1
IIOF(0 – 3)_x#	8	I/O	A	Устанавливается в Z
NMI_x#	2	I	–	Нет эффекта
RESET_COMM#	1	I	–	Вход RESET
RESETLOC(1, 0)_x	4	I	–	Нет эффекта
ROMEN_x	2	I	–	Нет эффекта
TCLK0_x	2	I/O	A	Устанавливается в Z
TCLK1_x	2	I/O	A	Устанавливается в Z
CntrlPLL	1	I	–	Устанавливает режим PLL «X2/CLKIN × 10»

Примечания

1 x = 1, 2.

2 Рекомендуемые конденсаторы, подключаемые к устройству, кратны 0,1 и 4,7 мкФ. Их количество зависит от уровня шумов на плате.

3 Во время сброса CntrlPLL не влияет на тактирование.

* A – асинхронный, S – синхронный.

** I – вход, O – выход, I/O – вход/выход.

5.7.7.2 Размещение вектора системного сброса

После осуществления сброса ядра 1867ВЦ8Ф1 начинают выполнение программы. Начальный адрес программы сохраняется в векторе сброса. Ядро процессора 1867ВЦ8Ф1 позволяет выбрать один из четырёх адресов вектора сброса. Выборка нужного адреса осуществляется посредством выставления уровней RESETLOC1_x и RESETLOC0_x при сбросе. В таблице 5.20 приведены их возможные конфигурации.

Таблица 5.20 − Размещение вектора RESET

RESETLOC1_x	RESETLOC0_x	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

5.7.7.3 Дополнительные операции системного сброса

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

- устанавливаются регистры таймеров:
- глобальный регистр управления таймером устанавливается в 0,
исключая разряд DATIN, который принимает значение сигнала TCLK;
- счётчик 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
(семь состояний ожидания);
- регистр управления каналом ПДП, счётчик передачи ПДП и вспомогательный счётчик передачи ПДП устанавливаются в 0;
0 загружается в следующие регистры ЦПУ: IIE, IIF, DIE, IVTP, TVTP.
- регистр состояния ЦПУ (ST) устанавливается в 0400h, при этом кэш «замораживается».
Вектор сброса считывается из памяти по соответствующему адресу и загружается в PC.

Если RESETLOC(1, 0)_x (x = 1, 2) переводится в низкий уровень, а на ROMEN_x установлен высокий уровень (внутреннее ПЗУ разрешено), процессорное ядро 1867ВЦ8Ф1 начинает выполнять код загрузчика. Иначе, ядро процессора 1867ВЦ8Ф1 начинает выполнение программы с адреса, указанного вектором сброса в соответствии с RESETLOC(1, 0)_x.

Мультипроцессорные системы на базе процессора 1867ВЦ8Ф1 могут сбрасываться и синхронизироваться по одному и тому же тактовому сигналу.

5.8 Работа конвейера

Две характеристики, которые вносят вклад в высокую производительность ПЦОС: конвейеризация и параллельное выполнение операций ЦПУ и ввода-вывода.

Пять функциональных стадий конвейера определяют работу ПЦОС: выборка, декодирование, чтение, выполнение и ПДП. Конвейеризация − это перекрытие или параллельное выполнение уровней основной команды: выборки, декодирования, чтения и выполнения.

Выполняя операции ввода-вывода, сопроцессор ПДП освобождает ЦПУ от выполнения этих функций, уменьшая загрузку конвейера и увеличивая вычислительную мощность.

Основные понятия, обсуждаемые в данном подразделе:

- структура конвейера;
- конфликты конвейера;
- конфликты переходов;
- конфликты регистров;
- конфликты памяти;
- разрешение конфликтов конвейера.

Синхронизация обращений к памяти:

- выборка инструкций;
- загрузка и сохранение данных;
- обращения ПДП.

5.8.1 Структура конвейера

Пять основных стадий структуры конвейера ПЦОС реализуют следующие функции:

- стадия выборки **F** (Fetch) – выборка командного слова из памяти и модификация счётчика команд PC;

- стадия декодирования **D** (Decode) – декодирование командного слова и генерация адреса; управление любыми модификациями вспомогательных регистров и указателя стека;

- стадия чтения **R** (Read) – при необходимости, чтение операнда из памяти;

- стадия исполнения **E** (Execute) – при необходимости, чтение операнда из регистрового файла, исполнение необходимой операции и запись результата в регистровый файл. При необходимости, запись в память результата предыдущей операции;

- каналы ПДП – чтение и запись памяти.

Базовая команда имеет четыре стадии выполнения: выборка, декодирование, чтение и исполнение. На рисунке 5.65 иллюстрируются эти стадии в структуре конвейера. Стадии индексированы в соответствии с циклом команды и исполнения. Полное перекрытие в конвейере, где все четыре элемента работают параллельно, возникает на цикле m. Те уровни, которые должны быть почти исполнены на m+1, уже исполнены на уровне m–1. Конвейер обеспечивает высокоскоростное исполнение в один элемент за цикл. Управление конфликтами конвейера осуществляется таким образом, что они являются прозрачными для пользователя. Не возникает необходимости предпринимать какие-либо специальные меры для гарантии корректного исполнения операций.

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	

Рисунок 5.65 − Структура конвейера ПЦОС

Примечания

1 W, X, Y, Z – обозначают инструкции.

2 F, D, R, E – выборка, декодирование, чтение и исполнение, соответственно.

Приоритеты от высшего к низшему, присвоенные для каждой из функциональных стадий, следующие:

- ПДП – если ПДП сконфигурировано на высший приоритет;
- исполнение;
- чтение;
- декодирование;
- выборка;
- ПДП – если ПДП сконфигурировано на низший приоритет.

Когда в процессе обработки инструкции наступает готовность к переходу на следующий более высокий уровень конвейера, но этот уровень не готов к новому входу, возникает конфликт конвейера. В этом случае блок с более низким приоритетом ожидает, пока блок с более высоким приоритетом завершит выполнение своей текущей функции.

Конфликты ПДП с ЦПУ могут быть минимизированы путём соответствующего структурирования данных, поскольку сопроцессор ПДП имеет собственные шины данных и адреса.

5.8.2 Конфликты конвейера

Конфликты конвейера ПЦОС могут быть сгруппированы в следующие три категории:

- конфликты переходов – включают большинство команд или операций, которые читают и/или модифицируют PC;
- конфликты регистров – включают задержки, которые могут наступить при обращении (на чтение или на запись) в регистры, используемые для генерации адреса, такие как AR0–AR7, IR0, IR1 BK, DP и SP;
- конфликты памяти – возникают, когда внутренние устройства ПЦОС находятся в состязании за ресурсы памяти.

Далее на примерах рассматривается каждый из этих трёх типов конфликтов. При возникновении ситуаций, когда данные перевыбираются или, операция повторяется, символ, представляющий стадию конвейера, дополняется номером. Например, если выборка выполняется снова, мнемоника команды повторяется. Когда обращение задерживается на несколько циклов из-за отсутствия готовности, используются символы RDY# и RDY для указания отсутствия или наличия готовности, соответственно.

5.8.2.1 Конфликты переходов

Первый тип конфликтов конвейера возникает при стандартных (незадержанных) ветвлениях (BR, Bcond, DBcond, CALL, IDLE, RPTB, RPTS, RETIcond, RETScond), а также при прерываниях и сбросе. Конфликты возникают при этих командах и операциях, т. к. в течение их исполнения конвейер используется только для завершения операции; другая информация, выбранная в конвейер, отбрасывается или перевыбирается, или конвейер неактивен. Это называется очисткой конвейера. Очистка конвейера необходима в этих случаях для гарантии, что команды не будут выполнены по частям. TRAPcond и CALLcond классифицируются отдельно от других типов переходов и рассматриваются позже.

В примере 5.46 приведена программа и работа конвейера для стандартного перехода. Обратите внимание, что выполняется одна пустая выборка MPYF и тогда после того, как доступен адрес перехода, выполняется новая выборка (команда OR). Эта пустая выборка влияет на кэш.

Пример 5.46 – Стандартное ветвление.

```
        BR THREE  ; Безусловный переход
        MPYF      ; Не выполняется
        ADD       ; Не выполняется
        SUBF      ; Не выполняется
        AND       ; Не выполняется
        .
        .
        .
THREE   OR; Выбрано 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

Обе команды RPTS и RPTB очищают конвейер, обеспечивая возможность загрузки регистров RS, RE и RC в соответствующее время относительно хода конвейера. Если эти регистры загружены без использования RPTS и RPTB, не происходит очистки конвейера. Если не используется ни один из режимов повторения, RS, RE и RC могут быть использованы как 32-разрядные регистры общего назначения без возникновения конфликтов конвейера. В таких случаях, как вложение RPTB, обусловленное вложением прерываний, может возникнуть необходимость загрузить и сохранить эти регистры непосредственно во время использования режима повторения. Так как до четырёх команд может быть выбрано до введения режима повторения, загрузки должны предшествовать очистке конвейера. Если RC модифицируется инструкцией его загрузки, эта операция имеет приоритет над изменением, выполняемым блоком логики режима повторений.

Задержанные переходы реализованы, чтобы гарантировать выборку и исполнение трёх следующих за переходом инструкций. Задержанные переходы активируются инструкциями BRD, BcondD и DBcondD. В примере 5.47 приведена программа и работа конвейера для задержанного перехода.

Пример 5.47 – 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	–	–	Не выполняется задержка
n+2	ADDF	MPYF	BRD	–	
n+3	SUBF	ADDF	MPYF	BRD	
THREE	MPYF	SUBF	ADDF	MPYF	

THREE → PC

5.8.2.2 Конфликты регистров

Конфликты регистров представляют собой чтение или запись регистров, использованных для адресации. Эти конфликты возникают, когда соответствующий регистр не готов для использования. Некоторые условия, при которых можно избежать конфликтов регистров, обсуждаются в 5.8.3.

Регистры образуют три функциональные группы:

Группа 1: вспомогательные регистры AR0 – AR7, индексные регистры IR0, IR1 и регистр размера блока BK;

Группа 2: указатель страницы данных DP;

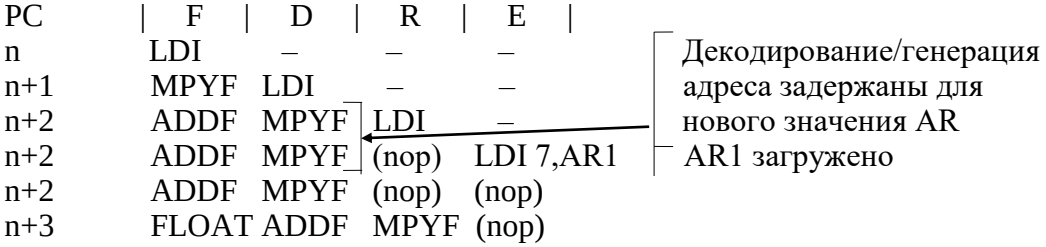
Группа 3: указатель системного стека SP.

Если команда производит запись в одну из этих групп, стадия декодирования не может использовать любой регистр внутри группы до завершения записи, т. е. до завершения исполнения команды. В примере 5.48 вспомогательный регистр загружается, а другой вспомогательный регистр используется для следующей команды. Так как стадия декодирования нуждается в результате записи во вспомогательный регистр, декодирование этой второй команды задерживается на два цикла. Каждый раз при задержке декодирования выполняется перевыборка слова программы, т. е. ADDF выбирается три раза. Так как это действительные перевыборки, они могут привести не только к конфликтам с сопроцессором ПДП, но и к кэш-попаданию и кэш-отсутствию.

Пример 5.48 – Запись в AR с последующей генерацией адреса AR.

```
        LDI   7, AR1        ; 7 → AR1
NEXT    MPYF  *AR2, R0      ; Декодирование задержано на два цикла
        ADDF
        FLOAT
```

Работа конвейера:



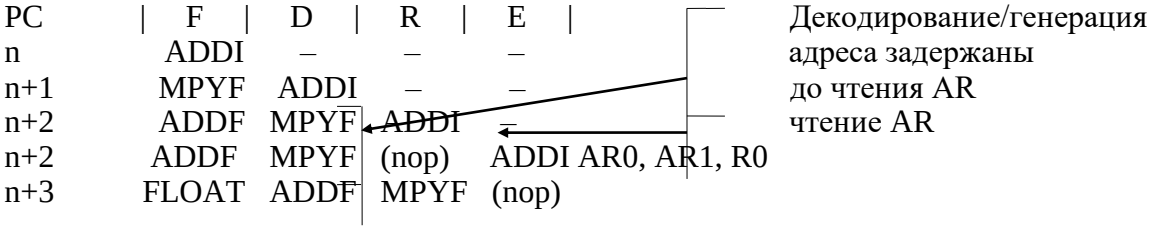
Случай чтения для этих групп аналогичен записи. Если команда должна считать значение в одном из элементов группы, использование той же группы при декодировании следующей команды задерживается до завершения чтения. Регистры считываются в начале цикла исполнения, поэтому требуется только один цикл задержки последующего декодирования. Для регистров IR0, IR1, BK или DP задержка не возникает. Для всех остальных, включая SP, задержка возникает.

В примере 5.49 производится сложение содержимого двух вспомогательных регистров с записью в регистр расширенной точности. Следующая команда использует другой вспомогательный регистр как адресный.

Пример 5.49 – Чтение AR с последующим использованием AR для генерации адреса.

```
        ADDI    AR0, AR1, R1        ; AR0 + AR1 → R1
NEXT    MPYF    *++AR2, R0          ; Decode delayed 1 cycle
        ADDF
        FLOAT
```

Работа конвейера:



Команды DBR (декремент и ветвление) используют вспомогательные регистры в качестве счётчиков цикла, что аналогично использованию их для адресации. Следовательно, операции, приведённые в вышеописанных примерах, также присутствуют и при выполнении этих команд.

5.8.2.3 Конфликты памяти: ожидание программы, выборка программы не завершена, только выполнение, задержание всего

Конфликты памяти могут возникать при превышении пропускной способности физической памяти. Например, блоки 0 и 1 ОЗУ, а также внутренней ПЗУ могут поддерживать только два доступа за один цикл. Внешний интерфейс может поддерживать только один доступ за цикл. Некоторые условия, при которых можно избежать конфликтов памяти, обсуждаются в 5.8.4.

Конфликты конвейера при работе с памятью бывают следующих четырёх типов:

- ожидание программы – запрещается запуск программы;
- выборка программы не завершена – выборка программы начата, но ещё не завершена;
- только выполнение – последовательность команд требует трёх обращений к данным за один цикл;
- задержание всего – операция основной или расширенной шины должна завершиться, прежде чем может продолжиться другая.

Эти четыре типа конфликтов памяти рассматриваются в нижеследующих примерах.

Ожидание программы

Два условия могут задержать выборку инструкции:

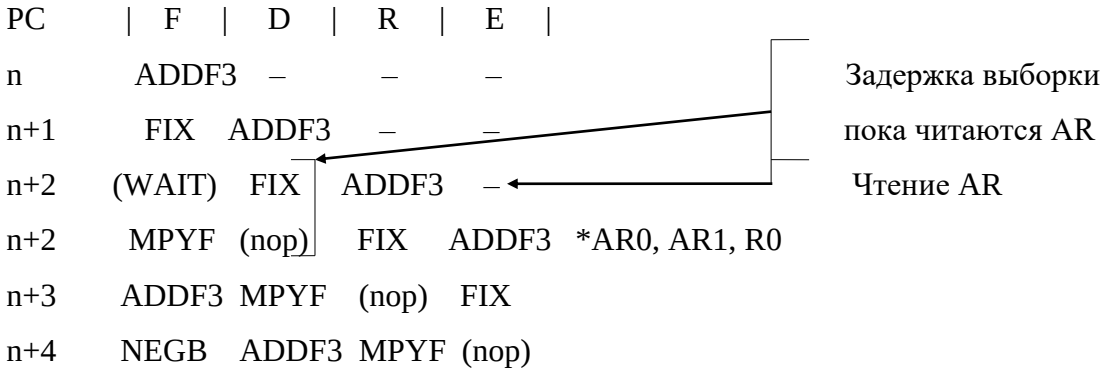
- начало доступа ЦПУ к данным, когда:
  - происходят два обращения ЦПУ к внутренним блокам ОЗУ или ПЗУ, и необходима выборка программы из того же блока;
  - один из внешних портов инициирует доступ к данным ЦПУ, и требуется загрузка программы из того же порта;
- требуется многоцикловое обращение ЦПУ или ПДП к данным через внешнюю шину.

В примере 5.50 иллюстрируется ожидание завершения обращения ЦПУ к данным. В этом случае *AR0 и *AR1 оба указывают на блок 0 ОЗУ, и команда MPYF также выбирает из блока 0 ОЗУ. Это приводит к конфликту, так как может быть произведено не более двух обращений к блоку 0 ОЗУ за один цикл, выборка программы не может быть начата и должна ждать завершения доступа ЦПУ к данным.

Пример 5.50 – Программа ожидает завершения обращения ЦПУ к данным.

```
ADDF3   *AR0, *AR1, R0
FIX
MPYF
ADDF3
NEGB
```

Работа конвейера:



В примере 5.51 иллюстрируется программное ожидание, связанное с многоцикловыми обращениями к данным или к ПДП. ADDF, MPYF и SUBF выбираются из некоторой области памяти, отличной от внешнего порта, необходимого для ПДП. ПДП инициирует многоцикловой доступ. Выборка программы, относящаяся к CALL, выполняется через тот же внешний порт, который использует ПДП. Даже если ПДП имеет самый низкий приоритет, многоцикловой доступ не может быть прерван. Выборка программы должна ожидать завершения доступа к ПДП.

Пример 5.51 – Ожидание программы в связи с многоцикловым обращением.
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

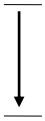
двухцикловое обращение ПДП

Выборка программы не завершена

Незавершённая выборка программы наступает, когда выборка инструкции занимает более одного цикла из-за состояний ожидания. В примере 5.52 MPYF и ADDF выбраны из памяти, которая поддерживает одноцикловое обращение. SUBF выбирается из памяти, требующей одного состояния ожидания. Одним из примеров, демонстрирующих этот конфликт, является выборка через границу банка внешнего порта.

Пример 5.52 – Многоцикловые выборки программной памяти.
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 одно состояние ожидания

Только выполнение

Тип конфликта «Только выполнение» при работе с памятью в конвейере возникает, когда последовательность инструкций требует трёх обращений ЦПУ к данным за один цикл. Это происходит в двух случаях:

- после инструкции, выполняющей сохранение, следует инструкция, которая осуществляет два чтения из памяти;
- инструкция выполняет два сохранения, а в следующей инструкции требуется, как минимум, одно чтение памяти.

Первый случай приведён в примере 5.53. Так как эта последовательность требует трёх обращений к памяти данных, а поддерживается только два, выполняется только фаза исполнения конвейера. Двойные чтения, требуемые LDF||LDF, задерживаются на один цикл. Следует обратить внимание, что может возникнуть перевыборка следующей команды.

Пример 5.53 – 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 R0, *AR1	до того, как смогут
n+4	X	W	LDF || LDF	(nop)	завершиться два чтения.
n+4	Y	X	W	LDF || LDF *AR2, R1 and *AR3, R2	

В примере 5.54 приводится параллельное сохранение, за которым следует одна загрузка или чтение. Так как требуются два параллельных сохранения, следующее чтение данных ЦПУ должно ожидать один цикл до начала. Может иметь место одна перевыборка памяти.

Пример 5.54 – 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 R0, *AR0 and R2, *AR1	
n+4	ASH	IACK	ADDF	(nop)	
n+4	–	ASH	IACK	ADDF	

Задержание всего

Существует три типа конфликтов конвейера при работе с памятью типа «Задержание всего»:

- операция загрузки или сохранения ЦПУ не может быть выполнена из-за того, что занята внешняя шина;
- внешняя загрузка занимает более одного цикла;
- условные вызовы и системные прерывания.

Первый тип конфликта «Задержание всего» возникает, когда один из внешних портов занят из-за обращения, которое началось, но не завершилось. В примере 5.55 первое сохранение − двухцикловое. ЦПУ записывает данные во внешний порт. Управление портом требует два цикла для завершения записи данные-данные. LDF − чтение через тот же внешний порт. Так как сохранение не завершено, ЦПУ продолжает попытки выполнить LDF, пока порт не станет доступен.

Пример 5.55 – Занят внешний порт

```
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	двухцикловое 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 «Задержание всего» представляет собой многоцикловое чтение данных. Чтение началось и продолжается до завершения. В примере 5.56 выполняется LDF из внешней памяти, что требует нескольких циклов для завершения.

Пример 5.56 – Многоцикловое чтение данных

```
LDF     @DMA, R0
```

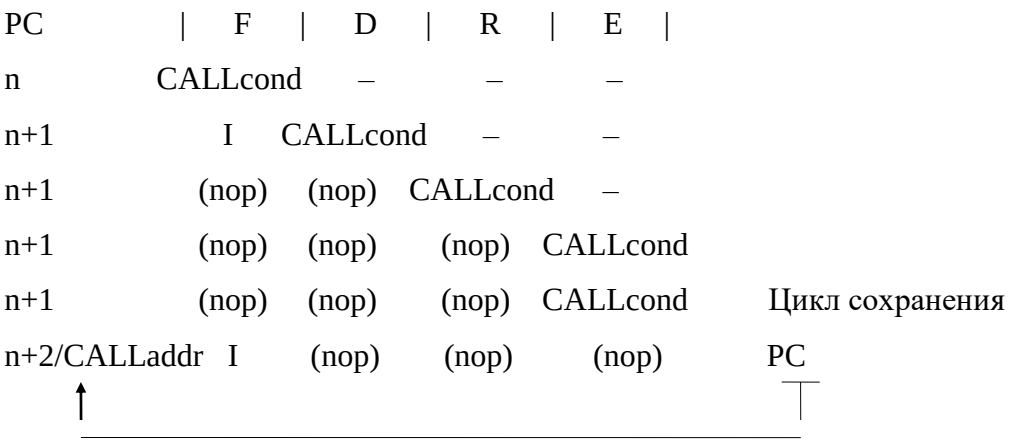
Работа конвейера:

PC	F	D	R	E	
n	LDF	–	–	–	
n+1	I	LDF	–	–	
n+2	J	I	LDF	–	двухцикловое обращение записи
n+3	K(пустой)	I	LDF	–	к внешней памяти
n+3	K2	J	I	LDF	

И последний тип конфликтов «Задержание всего» связан с условными вызовами и системными прерываниями, которые отличаются от других команд ветвления. Поскольку другие команды ветвления выполняют условное сохранение, условные вызовы и системные прерывания выполняют условное сохранение, которое занимает на один цикл больше условного перехода, см. пример 5.57. Дополнительный цикл используется для вталкивания адреса возврата после тестирования условия вызова.

Пример 5.57 – Условные вызовы и системные прерывания

Работа конвейера:



5.8.3 Разрешение конфликтов регистров

Если осуществляется обращение к вспомогательным AR11 – AR0, индексным IR0, IR1 регистрам, указателю страницы данных DP, указателю стека SP с какой-либо иной целью, кроме как для генерации адреса, может возникнуть конфликт конвейера, связанный со следующим обращением к памяти. Конфликты конвейера и задержки представлены в 5.8.2.2.

Примеры с 5.58 по 5.60 демонстрируют общее использование этих регистров, не ведущее к возникновению конфликта, или пути, с помощью которых можно избежать этих конфликтов.

Пример 5.58 – 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 5.59 – 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 команд
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 5.60 – 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	

5.8.4 Разрешение конфликтов памяти

Если производится выборка программы и обращение к данным таким образом, что ресурсы, которые будут использованы, не обеспечивают необходимый диапазон, выборка программы задерживается до завершения выборки данных. Определенные конфигурации выборки программ и обращений к данным обеспечивают условия, при которых ПЦОС имеет максимальную производительность.

В таблице 5.21 приведены сведения о том, сколько выборок данных может быть произведено для разных областей памяти, когда необходимо произвести выборку программы и одиночное обращение к данным и при этом получить максимальную производительность (один цикл). В четырёх случаях получается одноцикловая максимизация.

Таблица 5.21 − Одна выборка программы и одно обращение к данным для максимальной производительности

Случай	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 5.22 показано как много выборок может производиться из различных областей памяти, когда необходимо производить выборку программы и две выборки данных, с целью обеспечения максимальной производительности (один цикл). Шесть случаев приводят к такой производительности.

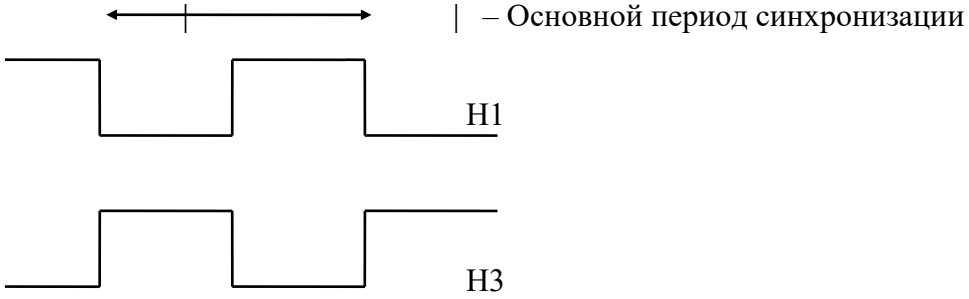
Таблица 5.22 − Одна выборка программы и два обращения к данным для максимальной производительности

Случай	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 1 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

5.8.5 Синхронизация доступа к памяти

В данном разделе обсуждается роль фаз внутренней фазы синхронизации (H1 и H3) и их соотношения для организации множественного доступа к памяти со стороны ПЦОС, В то время, как в предыдущем разделе обсуждалось взаимодействие между последовательностями команд, здесь обсуждается поток данных на основе отдельной команды.

Каждый основной период синхронизации в 10 нс состоит из двух меньших периодов синхросигнала по 20 нс, называемых H3 и H1. Активный период синхросигнала для H3 и H1 − время, когда этот сигнал в высоком уровне.



Точные операции чтения и записи памяти могут быть определены в соответствии с этими меньшими периодами синхронизации. Типы операций, которые могут возникать: выборка программы, загрузка и сохранение данных и обращения ПДП.

5.8.5.1 Выборки программы

Внутренние выборки программы всегда выполняются в течение H3, если другая команда в конвейере не должна произвести одно сохранение данных в то же самое время. В этом случае выборка программы производится в течение H1 и сохранение данных в течение H3.

Внешние выборки программы всегда начинаются в начале H3, с адресом, представляемым на внешней шине. В конце H1 они завершаются с фиксированием слова команды.

5.8.5.2 Загрузка и сохранение данных

Загрузку, чтение и сохранение данных выполняют четыре типа инструкций: двухоперандные, трёхоперандные, операции умножителя/АЛУ с сохранением результата, а также инструкции параллельного умножения и сложения. Более детальная информация о способах адресации содержится в подразделе 5.6.

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 доступа порта, один цикл исполнения ЦПУ. В последующих примерах приведены различия в количестве циклов шины и ЦПУ.

Обращения к памяти двухоперандных команд

Признаком всех двухоперандных команд является содержание значений 000_2 или 010_2 в разрядах 31 − 29 (см. рисунок 5.66). 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 с защёлкиванием слова данных в конце H1.

В случае сохранения данных разряды 15 − 0 являются операндом dst. Внутренние данные сохраняются в течение H3. Внешняя запись данных всегда начинается в начале H3, с адресом и данными, выставляемыми на внешней шине.

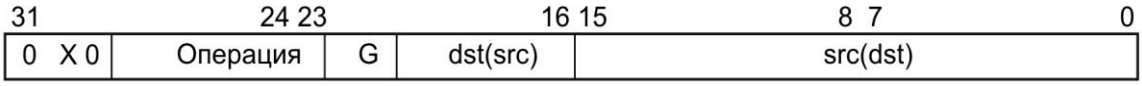


Рисунок 5.66 − Слово двухоперандной команды

Чтение памяти трёхоперандной командой

Признаком всех трёхоперандных команд является содержание в разрядах 31 − 29 значения 001_2 (см. рисунок 5.67). Операнды источника, src1 и src2 приходят либо из регистра, либо из памяти. Когда один или более операнд источника из памяти, эти команды всегда производят чтение памяти.

Если только один из операндов источника содержится во внутренней памяти (либо src1, либо src2), данные считываются в течение H1. Если один операнд памяти источника находится во внешней памяти, чтение данных начинается по H3, с адресом, выставляемым на внешней шине, и завершается с защёлкиванием слова данных в конце H1.

Если оба операнда должны выбираться из памяти, могут возникать различные варианты. Если оба операнда расположены во внутренней памяти, 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, выборка src2 начинается по началу H3 и защёлкивается 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, для завершения чтения требуются два цикла. В первом цикле производится внутренняя выборка src2. Выборка src1 также производится, но не фиксируется до следующего H3.

Если оба операнда src1 и src2 находятся во внешней памяти, 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.

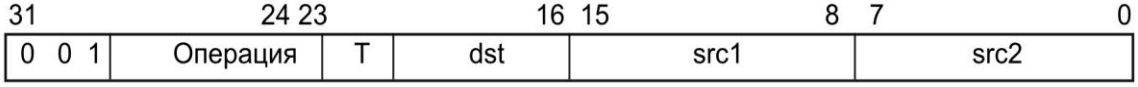


Рисунок 5.67 − Слово трёхоперандной команды

Операции с параллельным сохранением

Следующий класс команд включает все команды, которые производят параллельное сохранение в сочетании с другой командой. Разряды 31 и 30 для этих команд содержат11_2.

Для тех операций, которые производят умножение или операцию АЛУ параллельно с сохранением, формат слова команды приведён на рисунке 5.68. Если операция сохранения для внешнего или внутреннего dst2, она выполняется в течение H3. Два цикла шины требуются для внешних сохранений, но только один цикл ЦПУ необходим для завершения записи.

Если операция чтения памяти – внешняя, она начинается в начале H3 и защёлкивается в конце H1. Если операция чтения памяти является внутренней, она выполняется в течение H1. Помните, что чтения памяти выполняются ЦПУ в течение фазы чтения R конвейера и сохранения выполняются в течения фазы исполнения E.

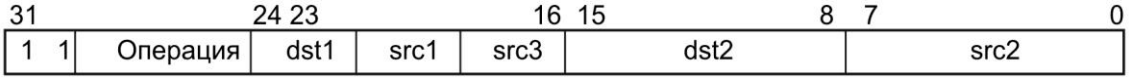


Рисунок 5.68 − Операция умножения или ЦПУ с параллельным сохранением

Формат слова команды для тех команд, в которых присутствуют параллельные сохранения в памяти, приведены на рисунке 5.69. Если оба операнда назначения, dst1 и dst2, расположены во внутренней памяти, dst1 сохраняется в течение H3, а dst2 в течение H1, таким образом, завершая два сохранения в памяти в один цикл.

Если dst1 – во внешней памяти, а dst2 – во внутренней, сохранение dst1 начинается в начале H3. Сохранение dst2 во внутренней памяти выполняется в течение H1. Для внешнего сохранения требуется для внешнего сохранения, но только один цикл ЦПУ требуется для завершения записи. Снова два сохранения в памяти завершается в один цикл.

Если dst1 – во внутренней памяти, а dst2 – во внешней, требуется дополнительный цикл шины для завершения сохранения dst2. Для завершения записи требуется только один цикл ЦПУ, но доступ порта требуется в течение трёх циклов шины. В первом цикле выполняется внутреннее сохранение dst1 в течение H3 и dst2 записывается в порт в течение H1. В течение следующего цикла выполняется сохранение dst2 на внешней шине, начиная с H3, и выполняется как нормальное в течение следующего цикла.

Если оба операнда (dst1 и dst2) пишутся во внешнюю память, 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 четвёртом цикле внешней шины.

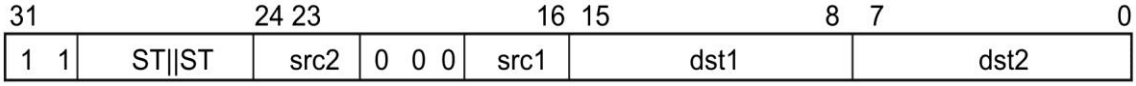


Рисунок 5.69 − Два параллельных сохранения

Параллельные умножения и сложения

Адресация памяти для параллельных умножений и сложений подобна адресации для трёхоперандных команд. Признаком всех команд параллельного умножения и сложения является содержание в разрядах 31 − 30 значения 10_2 (см. рисунок 5.70).

Для этих операций операнды src3 и src4 расположены в памяти. Если оба операнда расположены во внутренней памяти, 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, выборка src4 начинается в начале H3 и защёлкивается 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, для завершения двух операций чтения требуется один цикл. 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 src3.

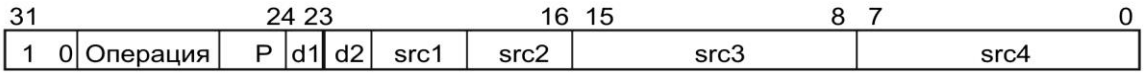


Рисунок 5.70 − Параллельные умножения и суммирования

5.9 Операции внешней шины

Оба процессорных ядра ИС 1867ВЦ8Ф1 имеют в своём составе по два идентичных набора шин (данных, адресов и стробов) внешнего интерфейса. Один из наборов называется интерфейсом глобальной памяти, другой – интерфейсом локальной памяти. Эти наборы шин реализованы для получения высокой пропускной способности за счёт одновременной загрузки и сохранения в различные области внешней памяти.

Представленная в настоящем разделе информация относится как к интерфейсу глобальной памяти, так и к интерфейсу локальной памяти; однако, в некоторых подразделах обсуждается только интерфейс глобальной памяти.

5.9.1 Обзор

Каждый из двух параллельных внешних интерфейсов (глобальной и локальной памяти) процессорных ядер ИС 1867ВЦ8Ф1 реализует следующие возможности:

- разделение конфигураций, где каждая из них имеет собственную 32-разрядную шину данных и 31-разрядную адресную шину;
- одноцикловое чтение и конвейерная запись;
- независимые сигналы разрешения для данных, адресов и управляющих линий;
- сигнализация запроса шины и блокировки шины для разделения параллельных процессов в памяти;
- управляемое пользователем распределение адресов для каждого из двух наборов независимых стробов для различных скоростей памяти;
- предварительный просмотр сигналов состояния шины для определения текущей и запрашиваемой операций при организации параллельных процессов;
- настраиваемые состояния ожидания (управляемые программно или аппаратно);
- сигналы, индицирующие, переходы на границах страниц памяти (от одной страницы памяти к другой).

Примечание – Интерфейс глобальной памяти в целом 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» (см. примечание к рисунку 5.71). Во всем этом подразделе нет различий между сигналами локального и глобального интерфейсов, а также STRB0# и STRB1#, за исключением случаев, когда это делается для большей ясности.

Сигналы, индицирующие, переходы на границах страниц памяти, поддерживают три типа памяти:

- страничный режим и статическое декодирование DRAM по столбцам;
- высокоскоростные SRAM банки;
- низкоскоростные банки памяти и устройства ввода-вывода.

5.9.2 Сигналы интерфейса памяти

Как показано на рисунке 5.71, интерфейс глобальной памяти имеет два набора управляющих сигналов, STRB0# и STRB1#. Регистры управления портами глобальной памяти, см. раздел 5.9.3, определяют, какой из регистровых наборов активен в настоящий момент.

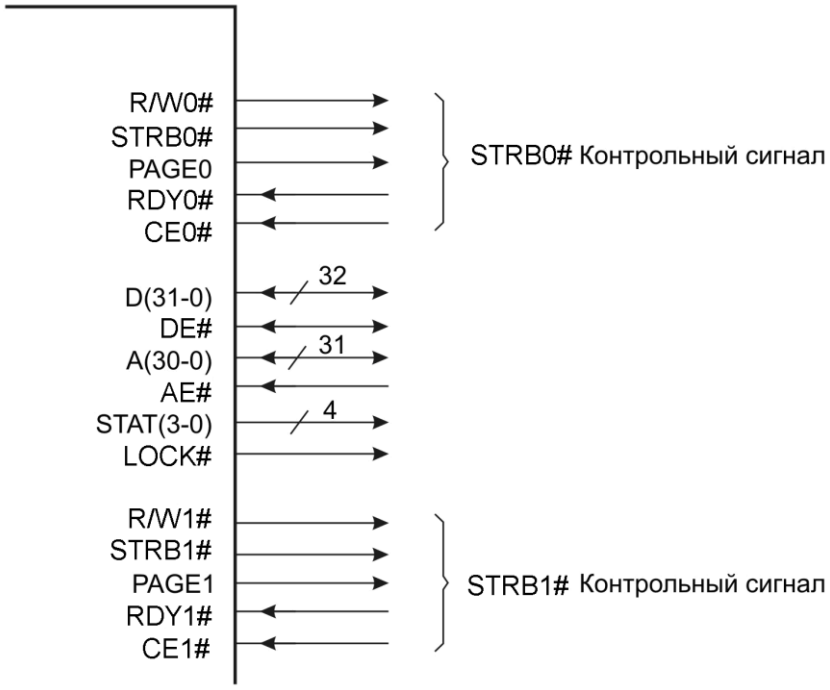


Рисунок 5.71 – Сигналы управления интерфейсами глобальной и локальной памяти

Примечание – Сигналы, использованные на рисунке 5.71, – для интерфейса глобальной памяти. Сигналы интерфейса локальной памяти имеют сходную конфигурацию и дополнительный «L» префикс, добавленный к каждому сигналу (например, STRB0# становится LSTRB0# и т. д.)

Таблица 5.23 − Сигналы интерфейса глобальной памяти

Сигнал	Тип	Описание	Значение после сброса	Режим ожидания
1	2	3	4	5
AE#*	I	Сигнал разрешения адресной шины для интерфейса глобальной памяти. При высоком уровне (установлена 1) устанавливает A(30–0) в высокоимпедансное состояние	Нет #	Игнорирован
CE(0, 1)#*	I	Сигнал разрешения для R/Wx#, STRBx# и PAGEx сигналов. При высоком уровне (установлена 1), следующие сигналы R/Wx#, STRBx# и PAGEx в высокоимпедансном состоянии (x = 0 для CE0# и x = 1 для CE1#)	Нет	Игнорирован
DE#*	I	Сигнал разрешения шины данных для интерфейса глобальной памяти. При высоком уровне (установлена 1), D(31-0) в высокоимпедансном состоянии. Чтение может быть по-прежнему допустимо, но записи не может быть	Нет	Игнорирован
LOCK#**	O	Сигнал блокировки для интерфейса глобальной шины. Показывает, есть ли блокировка доступа (0 – доступ возможен, 1 – доступ невозможен). LOCK# изменяется только посредством инструкции блокировки	1	1
PAGE(0, 1)	O/Z	Сигнал разрешения доступа к страницам памяти для STRB(0, 1)#	0	0
RDY(0,1)	I	Индикаторы готовности внешней памяти к доступу	Нет	Игнорирован
R/W(0, 1)#	O/Z	Определяет чтение из памяти (включается высоким уровнем) или запись в память (включается низким уровнем)	1	1

Окончание таблицы 5.23

1	2	3	4	5
STAT(3 – 0)**	O	Четыре сигнала, определяющие состояние или функцию порта памяти, как показано в таблице 5.24	Все 1	Все 1
STRB(0, 1)#	O/Z	Строб доступа к интерфейсу	1	1
A(30 – 0)	O/Z	Адресная шина. Адресные сигналы всегда управляемы. Они сохраняют адрес последнего доступа	Hi-Z	Адрес последнего доступа
D(31 – 0)	I/O/Z	Шина данных. Между циклами записи переходят в высокоимпедансное состояние	Hi-Z	Hi-Z

Примечания

1 Ноль указывает на сигналы управления STRB0#, а единица указывает на сигналы управления STRB1#.

2 Принятые условные обозначения: O – выход; I – вход; Z – высокоимпедансное состояние.

_____

* Сигнал, обозначенный *, может быть использован в конфигурации разделения шин и удержания их в выключенном состоянии при многопроцессорном доступе к памяти и периферии.

** STAT(3–0) и LOCK# не могут управляться внешним сигналом управления.

Нет # – означает отсутствие эффекта.

|| Режим ожидания – нет доступа к внешней памяти

В таблицы 5.24 приведены доступные состояния порта глобальной памяти в зависимости от комбинации значений сигналов STAT3–STAT0. Для доступа к шине эти сигналы предоставляют информацию о доступе, который вот-вот начнётся. Код для инструкции чтения SIGI может использоваться для того, чтобы отличить SIGI чтение от LDII или LDFI чтения.

Код режима ожидания шины 1111_2 (последняя строка таблицы 5.24) упрощает модульные многопроцессорные интерфейсы с общей шиной, поскольку подтягивающие резисторы могут использоваться для сигнализации состояния ожидания, когда процессорные карты не подключены к общей шине.

Таблица 5.24 − Состояние порта глобальной памяти для доступа 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

_____

* Таблица применима к обоим интерфейсам – глобальной памяти и локальной памяти (для сигналов локального интерфейса добавляется префикс «L»: LSTAT3, LSTAT2 и т. д.).

5.9.3 Регистры управления интерфейсом памяти

Рисунок 5.72 иллюстрирует размещение в памяти (адреса) регистров управления интерфейсами глобальной и локальной памяти. На рисунке 5.73 приведён формат полей этих регистров. Каждый регистр может быть запрограммирован для управления соответствующим интерфейсом памяти, чтобы задать:

- размер страницы, используемый для двух стробов каждого порта;
- диапазоны адресов, в которых стробы активны;
- состояния ожидания;
- другие операции управления интерфейсом памяти.

На рисунке 5.73 приведены названия, варианты доступа (только чтение, чтение/запись), состояние после сброса (для каждого бита) и формат полей этих регистров. Разряды 3 – 0 определяют состояние сигналов AE#, DE#, CE1# и CE0#.

После сброса в поля регистров управления локальным и глобальным интерфейсами загружаются следующие значения:

- поля PAGE SIZE 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 (разряд 29) устанавливается в 1 для добавления цикла между обратным повторным 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 формирования сигнала внутренней готовности, представляющего из себя логическое И внешнего сигнала готовности READY (RDY#) и сигнала готовности, генерируемого внутренним счётчиком состояний ожидания (RDY_{wtcnt}).

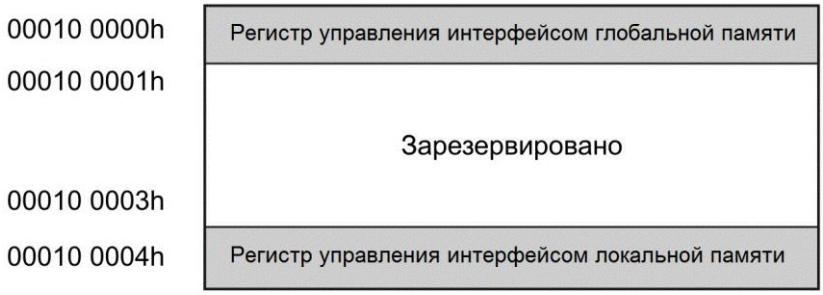


Рисунок 5.72 − Размещение регистров управления интерфейсом памяти

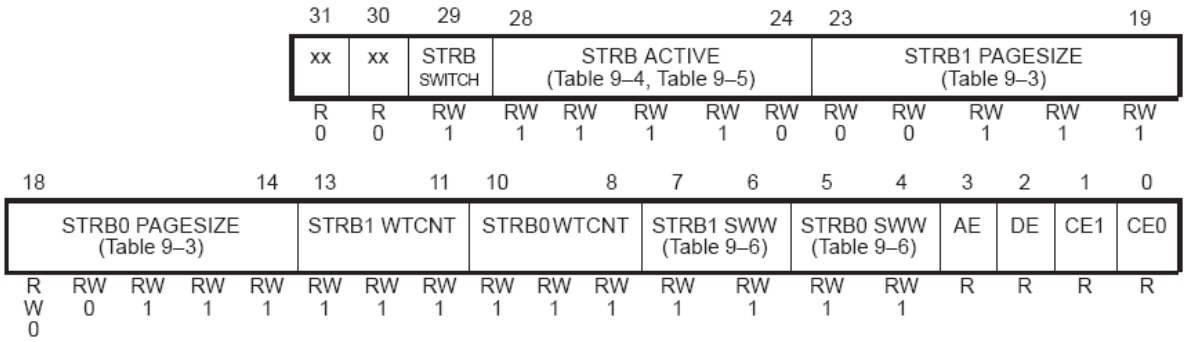


Рисунок 5.73 − Поля регистров управления интерфейсом памяти

Примечания к рисунку 5.73:

1 Рисунок ячеек регистра содержит мнемоники регистра управления интерфейсом глобальной памяти. К мнемоникам регистра управления интерфейсом локальной памяти добавляется префикс L (т. е. LSTRB SWW, LCE# и т. д.).

2 «1» и «0» под каждым разрядом – это двоичное значение, записываемое в указанный бит регистра при сбросе. Значения разрядов 3 − 0 определяются состоянием в момент сброса соответствующих им внешних сигналов (AE#, DE#, CE1# и CE0#).

3 RW = чтение/запись; R = чтение.

Описание полей регистров управления интерфейсами глобальной и локальной памяти приведено в таблице 5.25.

Таблица 5.25 − Описание полей регистра управления

CE0	Состояние внешнего сигнала CE0# (после его прохождения через внутренний синхронизатор). Значение нефиксируемое.
CE1	Состояние внешнего сигнала CE1# (после его прохождения через внутренний синхронизатор). Значение нефиксируемое.
DE	Состояние внешнего сигнала DE# (после его прохождения через внутренний синхронизатор). Значение нефиксируемое.
AE	Состояние внешнего сигнала AE# (после его прохождения через внутренний синхронизатор). Значение нефиксируемое.
STRB0 SWW	Состояние ожидания программы для STRB0# доступа. Совместно с STRB0 WTCNT, это поле определяет режим генерации состояния ожидания. Реальные состояния ожидания рассматриваются в 5.9.4 и в таблице 5.29.
STRB1 SWW	Состояние ожидания программы для STRB1# доступа. Совместно с STRB0 WTCNT, это поле определяет режим генерации состояния ожидания. Реальные состояния ожидания рассматриваются в 5.9.4 и в таблице 5.29.
STRB0 WTCNT	Программный счёт состояний ожидания для доступов STRB0#. Определяет количество циклов, используемых при включении программных состояний ожидания. Диапазон трёх разрядов – от 000_2 (ноль) до 111_2 (семь).
STRB1 WTCNT	Программный счёт состояний ожидания для доступов STRB1#. Определяет количество циклов, используемых при включении программных состояний ожидания. Диапазон трёх разрядов – от 000_2 (ноль) до 111_2 (семь).
STRB0 PAGE SIZE	Размер страницы для STRB0# доступов. Определяет количество старших разрядов адреса, используемых для определения размера банка для STRB0# доступов. См. диапазоны в таблице 5.26 и 5.9.3.2.
STRB1 PAGE SIZE	Размер страницы для STRB1# доступов. Определяет количество старших разрядов адреса, используемых для определения размера банка для STRB0# доступов. См. диапазоны в таблице 5.26 и 5.9.3.2.
STRB ACTIVE	Определяет диапазоны адресов, при которых STRB0#* и STRB1#* активны. См. диапазоны в таблице 5.27 для STRB ACTIVE и в таблице 5.28 для LSTRB ACTIVE.
STRB SWITCH	Управляет добавлением дополнительного цикла между повторными обратными чтениями, который переключает STRB0# в STRB1# (или наоборот). 1 – добавляется цикл. 0 – цикл не добавляется.
xx	Зарезервировано; читаются нули.

Таблица 5.26 − Размеры страниц, определяемые разрядами STRB(0,1)# PAGESIZE*

STRBx PAGE SIZE (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} = 1K
01010	30 – 11	10 – 0	2^{11} = 2K
01011	30 – 12	11 – 0	2^{12} = 4K
01100	30 – 13	12 – 0	2^{13} = 8K
01101	30 – 14	13 – 0	2^{14} = 16K
01110	30 – 15	14 – 0	2^{15} = 32K
01111	30 – 16	15 – 0	2^{16} = 64K
10000	30 – 17	16 – 0	2^{17} = 128K
10001	30 – 18	17 – 0	2^{18} = 256K
10010	30 – 19	18 – 0	2^{19} = 512K
10011	30 – 20	19 – 0	2^{20} = 1M
10100	30 – 21	20 – 0	2^{21} = 2M
10101	30 – 22	21 – 0	2^{22} = 3M
10110****	30 – 23	22 – 0	2^{23} = 8M
10111	30 – 24	23 – 0	2^{24} = 16M
11000	30 – 25	24 – 0	2^{25} = 32M
11001	30 – 26	25 – 0	2^{26} = 64M
11010	30 – 27	26 – 0	2^{27} = 128M
11011	30 – 28	27 – 0	2^{28} = 256M
11100	30 – 29	28 – 0	2^{29} = 512M
11101	30	29 – 0	2^{30} = 1Г
11110	Нет	30 – 0	2^{31} = 2Г
11111	Зарезервирован	Зарезервирован	Зарезервирован

_____

* Использованы мнемоники для регистра управления интерфейсом глобальной памяти. Для регистров управления интерфейсом локальной памяти вначале каждой мнемоники добавляется префикс L (например, LSTRB0 PAGESIZE, LSTRB1 PAGESIZE и т. д.). Размеры страниц, определяемые разрядами LSTRB(0,1)# PAGESIZE регистра управления интерфейсом локальной памяти, такие же.

** x в STRBx означает, что данные в колонках актуальны для STRB0# или STRB1#.

*** Значение при сбросе.

**** Поле STRBx PAGESIZE 10110_2 изображено на рисунке 5.75.

Таблица 5.27 − Диапазоны адресов, определяемые разрядами STRB ACTIVE*****

STRBx PAGESIZE (разр.24 − 28)**	STRB0 ACTIVE диапазон адресов	Размер диапазона адресов STRB0 ACTIVE	STRB1 ACTIVE диапазон адресов
1	2	3	4
00000 – 01110	Зарезервированы	Зарезервированы	Зарезервированы
01111	8000 0000 − 8000 FFFF	2^{16} = 64K	8001 0000 − FFFF FFFF
10000	8000 0000 − 8001 FFFF	2^{17} = 128K	8002 0000 − FFFF FFFF
10001	8000 0000 − 8003 FFFF	2^{18} = 256K	8004 0000 − FFFF FFFF
10010	8000 0000 − 8007 FFFF	2^{19} = 512K	8008 0000 − FFFF FFFF
10011	8000 0000 − 800F FFFF	2^{20} = 1M	8010 0000 − FFFF FFFF
10100	8000 0000 − 801F FFFF	2^{21} = 2M	8020 0000 − FFFF FFFF

Окончание таблицы 5.27

1	2	3	4
10101	8000 0000 − 803F FFFF	2^{22} = 4M	8040 0000 − FFFF FFFF
10110	8000 0000 − 807F FFFF	2^{23} = 8M	8080 0000 − FFFF FFFF
10111	8000 0000 − 80FF FFFF	2^{24} = 16M	8100 0000 − FFFF FFFF
11000	8000 0000 − 81FF FFFF	225 = 32M	8200 0000 − FFFF FFFF
11001	8000 0000 − 83FF FFFF	226 = 64M	8400 0000 − FFFF FFFF
11010	8000 0000 − 87FF FFFF	227 = 128M	8800 0000 − FFFF FFFF
11011	8000 0000 − 8FFF FFFF	228 = 256M	9000 0000 − FFFF FFFF
11100	8000 0000 − 9FFF FFFF	229 = 512M	A000 0000 − FFFF FFFF
11101	8000 0000 − BFFF FFFF	230 = 1Г	C000 0000 − FFFF FFFF
11110	8000 0000 − FFFF FFFF	231 = 2Г	нет
11111	Зарезервирован	Зарезервирован	Зарезервирован

_____

* Диапазон адресов определён разрядами LSTRB ACTIVE, список которых приведён в таблице 5.28.

** Значение при сбросе.

Таблица 5.28 − Диапазоны адресов, определяемые разрядами LSTRB ACTIVE *

LSTRBx PAGESIZE (разряды 24 − 28)**	LSTRB0 ACTIVE диапазон адресов	Размер диапазона адресов LSTRB0 ACTIVE	LSTRB1 ACTIVE диапазон адресов
00000-01110	Зарезервированы	Зарезервированы	Зарезервированы
01111	0000 0000 − 0000 FFFF	2^{16} = 64K	0001 0000 − 7FFF FFFF
10000	0000 0000 − 0001 FFFF	2^{17} = 128K	0002 0000 − 7FFF FFFF
10001	0000 0000 − 0003 FFFF	2^{18} = 256K	0004 0000 − 7FFF FFFF
10010	0000 0000 − 0007 FFFF	2^{19} = 512K	0008 0000 − 7FFF FFFF
10011	0000 0000 − 000F FFFF	2^{20} = 1M	0010 0000 − 7FFF FFFF
10100	0000 0000 − 001F FFFF	2^{21} = 2M	0020 0000 − 7FFF FFFF
10101	0000 0000 − 003F FFFF	2^{22} = 4M	0040 0000 − 7FFF FFFF
10110	0000 0000 − 007F FFFF	2^{23} = 8M	0080 0000 − 7FFF FFFF
10111	0000 0000 − 00FF FFFF	2^{24} = 16M	0100 0000 − 7FFF FFFF
11000	0000 0000 − 01FF FFFF	2^{25} = 32M	0200 0000 − 7FFF FFFF
11001	0000 0000 − 03FF FFFF	2^{26} = 64M	0400 0000 − 7FFF FFFF
11010	0000 0000 − 07FF FFFF	2^{27} = 128M	0800 0000 − 7FFF FFFF
11011	0000 0000 − 0FFF FFFF	2^{28} = 256M	1000 0000 − 7FFF FFFF
11100	0000 0000 − 1FFF FFFF	2^{29} = 512M	2000 0000 − 7FFF FFFF
11101	0000 0000 − 3FFF FFFF	2^{30} = 1Г	4000 0000 − 7FFF FFFF
11110	0000 0000 − 7FFF FFFF	2^{31} = 2Г	нет
11111	Зарезервирован	Зарезервирован	Зарезервирован

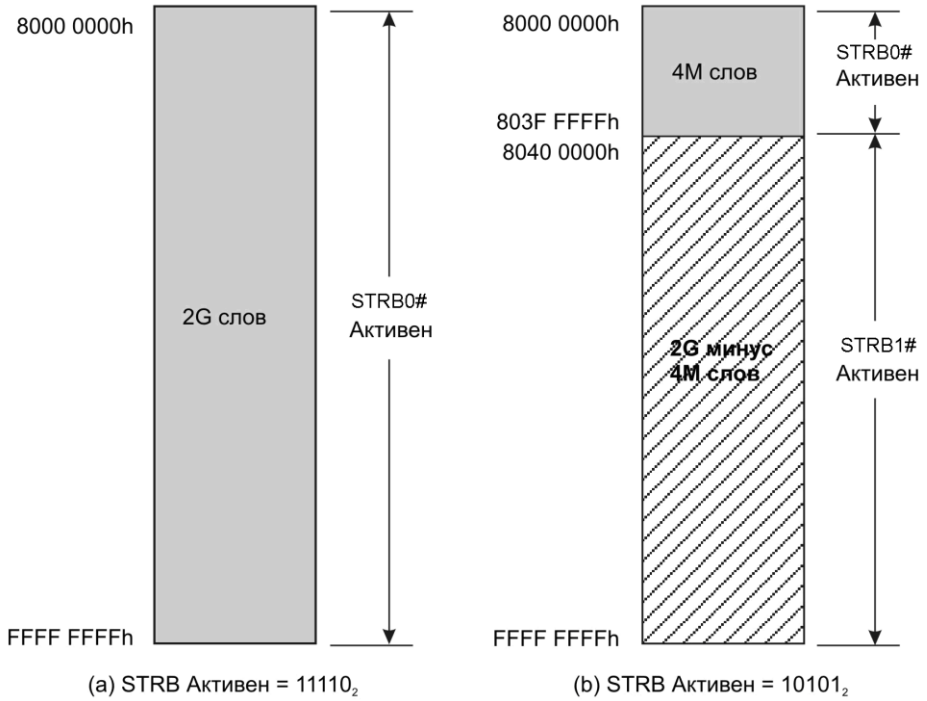
_____

* Диапазоны адресов ниже 0030 0000h действительны только в микропроцессорном режиме (ROMEN_x = 0). Доступ к зарезервированной, периферийной и внутренней областям памяти не активирует LSTRB# сигналы.

** Значение при сбросе.

5.9.3.1 Карта адресации к стробам

Рисунок 5.74 демонстрирует взаимосвязь между STRB ACTIVE разрядами и диапазонами адресов, при которых сигналы STRB0# и STRB1# активны. Заметим, что области адресов STRBx# и LSTRBx# также управляют областями ассоциированных сигналов – RDYx#, LRDYx#, R/Wx#, LR/Wx#, PAGEx, LPAGEx и т. д. (где x = 1 или 0).



Примечание – Здесь показаны два примера для карты глобальной памяти. Полная карта памяти (локальная и глобальная) показана на рисунке 5.4. Заметьте, что старший адрес для LSTRB1# (локальная шина) 7FFF FFFFh.

Рисунок 5.74 − Влияние STRB ACTIVE на карту памяти шины глобальной памяти

Пример (а) на рисунке 5.74 показывает условие сброса (STRB ACTIVE = 11110_2). В этом случае, сигнал STRB0# активен на всей адресной области шины глобальной памяти (см. таблицу 5.23).

Пример (б) на том же рисунке показывает карту глобальной шины памяти, когда STRB ACTIVE = 10101_2. В этом случае, STRB0# запускается из адресов 8000 0000h-803F FFFFh, и STRB1# запускается из адресов 8040 0000h-FFF FFFFh (как показано в таблице 5.27 для STRB ACTIVE 10101_2).

5.9.3.2 Операция размера страницы

С учётом области памяти, выбранной любыми четырьмя стробами, внешний интерфейс процессорного ядра 1867ВЦ8Ф1 позволяет вам в дальнейшем разделить область на страницы выбранного размера. Это обеспечивает гибкость при разработке высокоскоростных систем памяти с высокой плотностью размещения, в сочетании с более медленными периферийными устройствами; каждый раз, как только пересекается граница страницы, добавляется цикл, позволяющий произвести реконфигурацию внешней логики.

Каждое поле PAGESIZE в регистре управления интерфейсом памяти (показано на рисунке 5.73) работает сходным образом с определением размера страницы для его соответствующего строба. Размер страницы начинается от 256 слов (с разрядами 7 − 0 внешней адресной шины, определяющими адрес на странице) и область достигает до 2Г слов с разрядами 30 − 0 внешней адресной шины, определяющими положение на странице. Пример на рисунке 5.75 показывает, как значение поля размера страницы 10110_2 переводится в разряды 30 − 23, определяющие текущую страницу и разряды 22 − 0, определяющие адрес на странице.

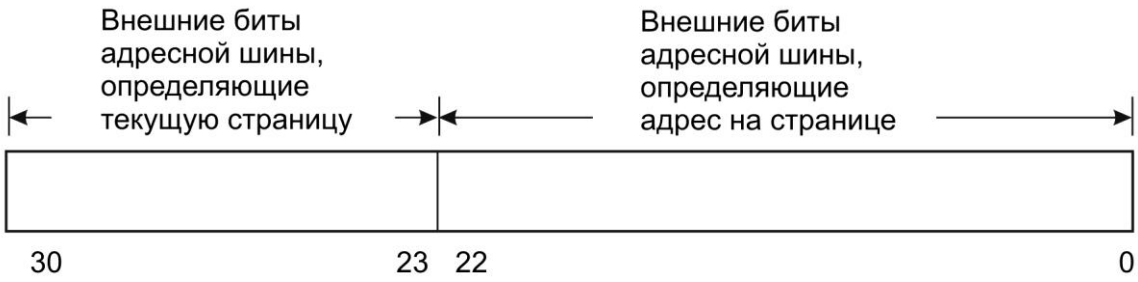


Рисунок 5.75 − Пример полей STRBx# PAGESIZE

Примечание − Рисунок 5.75 представляет значение поля STRBx PAGESIZE 10110_2 (как показано в таблице 5.26).

Переход от одной страницы к другой вызывает добавление цикла в последовательность внешнего доступа, позволяя переконфигурировать внешнюю логику. Например, добавочный цикл даёт время для замедления устройства, освобождения шины, таким образом, устраняется конфликт шины. Логика управления интерфейсом памяти сохраняет последовательность использованных адресов для последнего доступа для каждого STRB#. Когда начинается обращение, сигнал PAGE, соответствующий активному STRB#, становится неактивным (высокий уровень), если обращение происходит к новой странице. Сигналы PAGE0 и PAGE1 независимы один от другого, каждый имеет собственную логику размера страницы.

При сбросе логика управления страницей инициализируется, так что дополнительный цикл добавляется для первого доступа к двум стробам интерфейса.

Регистр управления интерфейсом локальной памяти функционируют аналогично регистру управления интерфейсом глобальной памяти.

5.9.4 Программируемые состояния ожидания

Ядро процессора 1867ВЦ8Ф1 имеет способность внутренне программно конфигурироваться чтобы сгенерировать сигналы для каждого строба. Этот генератор состояний ожидания управляется программированием двух полей в регистрах управления глобальным и локальным интерфейсами. Используйте поле STRBx WTCNT (разряды 8 − 10 и 11 − 13) для определения количества сгенерированных программных состояний ожидания, и поле STRBx SWW (разряды 6, 7 и 4, 5) для выбора одного из следующих четырёх режимов генерации состояний ожидания:

- внешний RDY# (SWW = 0). Состояния ожидания генерируются исключительно внешним RDY# (программа состояний ожидания игнорируется);
- полученный из WTCNT RDY_{wtcnt}# (SWW = 01_2). Состояния ожидания генерируются исключительно программой-генератором состояний ожидания (внешний RDY# игнорирован);
- логическое OR (ИЛИ) над RDY# и RDY_{wtcnt}# (SWW = 10_2). Состояние ожидания генерируется логическим ИЛИ внутреннего и внешнего сигналов готовности. Каждый сигнал может генерировать готовность;
- логическое AND (И) над RDY# и RDY_{wtcnt}# (SWW = 11_2). Состояния ожидания генерируются логическим AND внутреннего и внешнего сигналов готовности. Оба сигнала должны присутствовать.

Четыре режима используются для выработки внутреннего сигнала готовности, RDYint, который управляет доступом (обращениями). Пока RDYint = 1, текущий внешний запрос расширен. Когда RDYint = 0, выполняется текущий запрос. Так как использование программируемых состояний ожидания для обоих внешних интерфейсов идентично, в следующих разделах обсуждается только интерфейс глобальной шины.

RDYwtcnt – внутренний сигнал готовности. Когда начинается внешнее обращение, значение WTCNT загружается в счётчик. WTCNT может иметь значение от 0 до 7. Счётчик уменьшается каждый цикл H1/H3, пока не станет равным 0. После установки счётчика в 0 его значение не меняется до следующего обращения. Если счётчик не равен 0, RDY_{wtcnt} = 1. Если счётчик равен 0, RDY_{wtcnt} = 0.

Замечание: чтобы гарантировать функционирование системы с медленной внешней памятью, при сбросе ядро процессора 1867ВЦ8Ф1 добавляет семь состояний ожидания при каждом доступе вовне. Для повышения производительности системы при использовании быстрой памяти необходимо уменьшить число состояний ожидания.

Таблица 5.29 − 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	

5.9.5 Временная диаграмма интерфейса памяти

Ядро процессора 1867ВЦ8Ф1 выполняет одноцикловое внешнее чтение и конвейерную внешнюю запись, за исключением некоторых случаев, которые рассмотрены ниже в этом разделе. Запись производится в два 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, ПДП или ЦПУ должны ждать, и запись, таким образом, занимает два цикла.

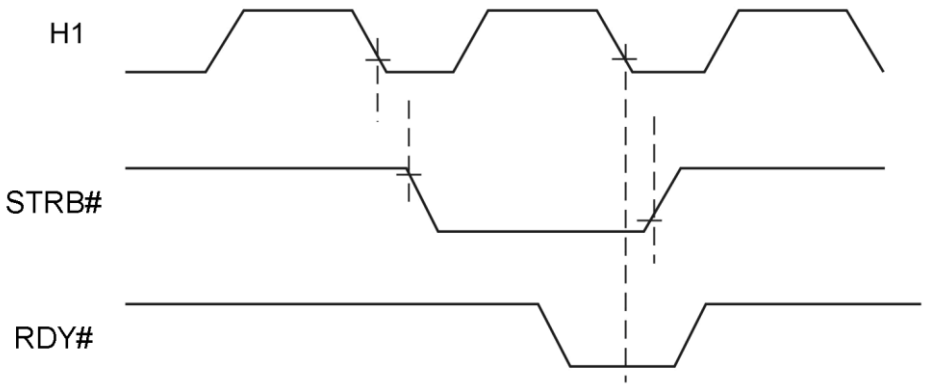


Рисунок 5.76 − Временная диаграмма STRB# и RDY#

Замечание: пунктир показывает взаимосвязь между сигналами.

Как показано на рисунке 5.76, изменение STRB# происходит по заднему фронту H1; по заднему же фронту H1 стробируется RDY#. Для других временных диаграмм, указанных в этом разделе, применяются следующие правила:

- изменение R/W# происходит посредством STRB#;
- пересечение границы страниц STRB# отражается в установке PAGE в высокий уровень на один цикл;
- изменение R/W#всегда происходит по переднему фронту H1;
- изменение STRB всегда происходит по заднему фронту H1;
- RDY# стробируется по заднему фронту H1;
- данные всегда стробируются по заднему фронту H1;
- данные всегда выставляются по заднему фронту H1;
- данные всегда остановлены и не управляются в течение записи по переднему фронту H1;
- сигналы состояния и PAGE, следующие за чтением, изменяются по заднему фронту H1. Адрес также изменяется по заднему фронту H1;
- выборка вектора прерывания для внешнего интерфейса определяется сигналами состояния для этого интерфейса STAT и LSTAT как чтение данных;
- сигналы состояния операций блокировки (LOCK# и LLOCK#) имеют одну и ту же временную диаграмму, что и сигналы состояния STAT и LSTAT соответственно;
- если PAGE переходит в высокий уровень, STRB# также переходит в высокий уровень.

Замечание: если нет внешнего порта, обращающегося к памяти (режим ожидания), сигналы управления находятся в неактивном состоянии (RDY# игнорирован, STRB в высоком уровне, STATx переходят в высокий уровень), шина адреса использует их последние значения и переходит в высокоимпедансное состояние (см. рисунок 5.86).

Рисунок 5.77 иллюстрирует последовательность чтения, чтения, записи. На рисунке предполагается, что по STRB1# происходит обращение при трёх обращениях к одной и той же странице. Временная диаграмма показывает следующее:

- повторное чтение одной и той же страницы как одноцикловое обращение;
- STRB# остаётся в низком уровне в течение повторного чтения;
- после перехода от чтения к записи STRB# переходит в высокий уровень на один цикл для изменения сигнала R/W#.

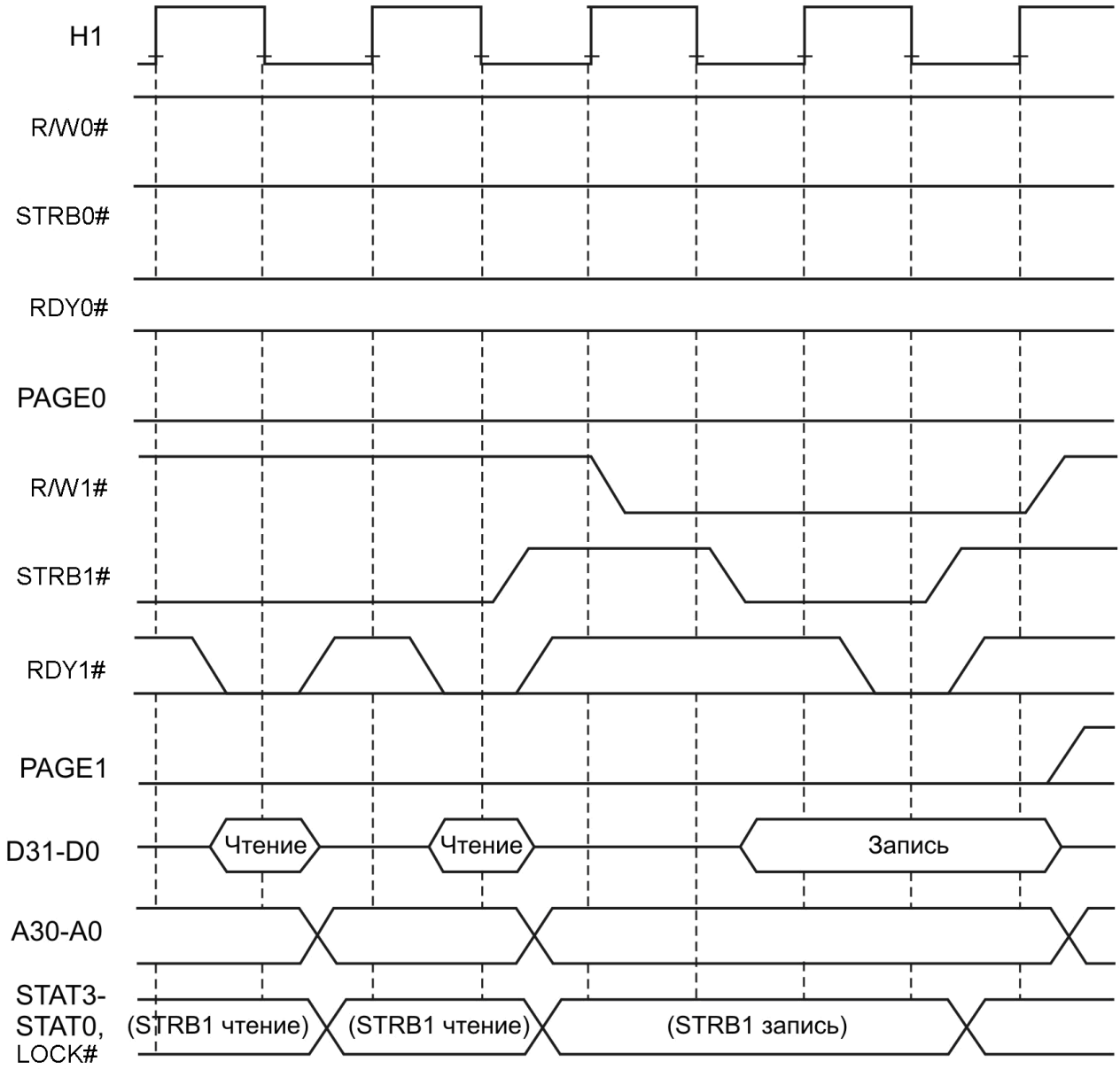


Рисунок 5.77 − Последовательность чтения, чтения, записи одной и той же страницы

Рисунок 5.78 показывает следующее:

- для предотвращения нежелательной записи, STRB# переходит в высокий уровень между повторными записями для отключения памяти, на время установки нового адреса;
- как показано на рисунке 5.77, STRB# переходит в высокий уровень между записью и чтением для изменения R/W#;
- чтение, следующее после записи, с той же шины, занимает 2 цикла. Это происходит независимо от того, считывается или нет соответствующий строб и/или страница;
- последовательная запись занимает 2 цикла.

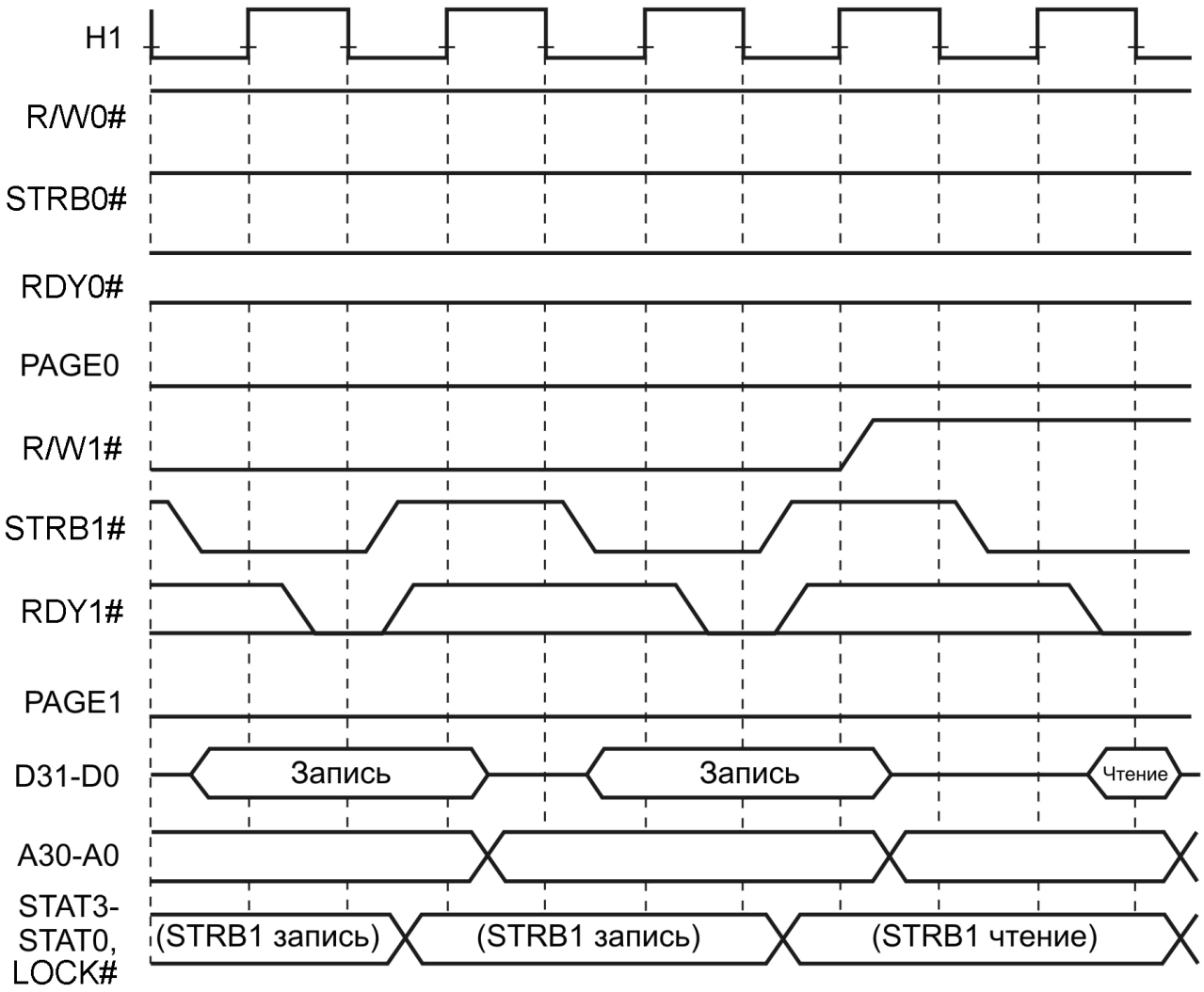


Рисунок 5.78 − Последовательность записи, записи, чтения одной и той же страницы

Рисунок 5.79 показывает изменения сигналов при чтении с разных страниц:
- добавляется цикл, чтобы выбрать следующую область памяти;
- при выполнении перехода на новую страницу PAGE на один цикл становится высоким;
- STRB1# переходит в высокий уровень на один цикл.

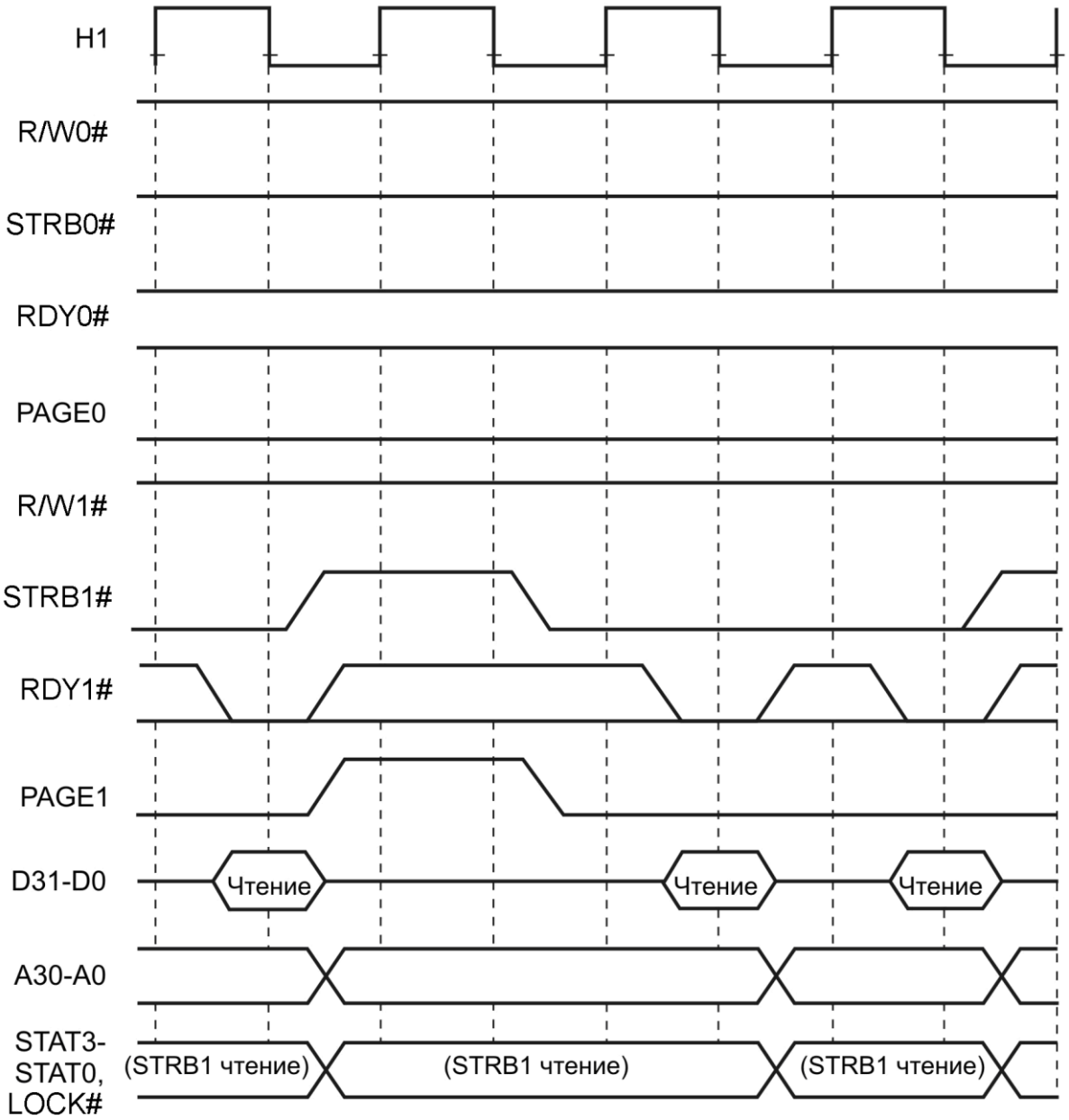


Рисунок 5.79 − Последовательность чтения одной страницы, чтения другой страницы, чтения той же страницы

Рисунок 5.80 показывает изменения сигналов при записи в разные страницы:

- PAGE1 это отображает, переходя в высокий уровень на один цикл;
- дополнительный цикл не добавляется, так как циклы записи показывают присущую полуциклу H1 установку информации об адресе перед тем, как STRB# перейдёт в низкий уровень.

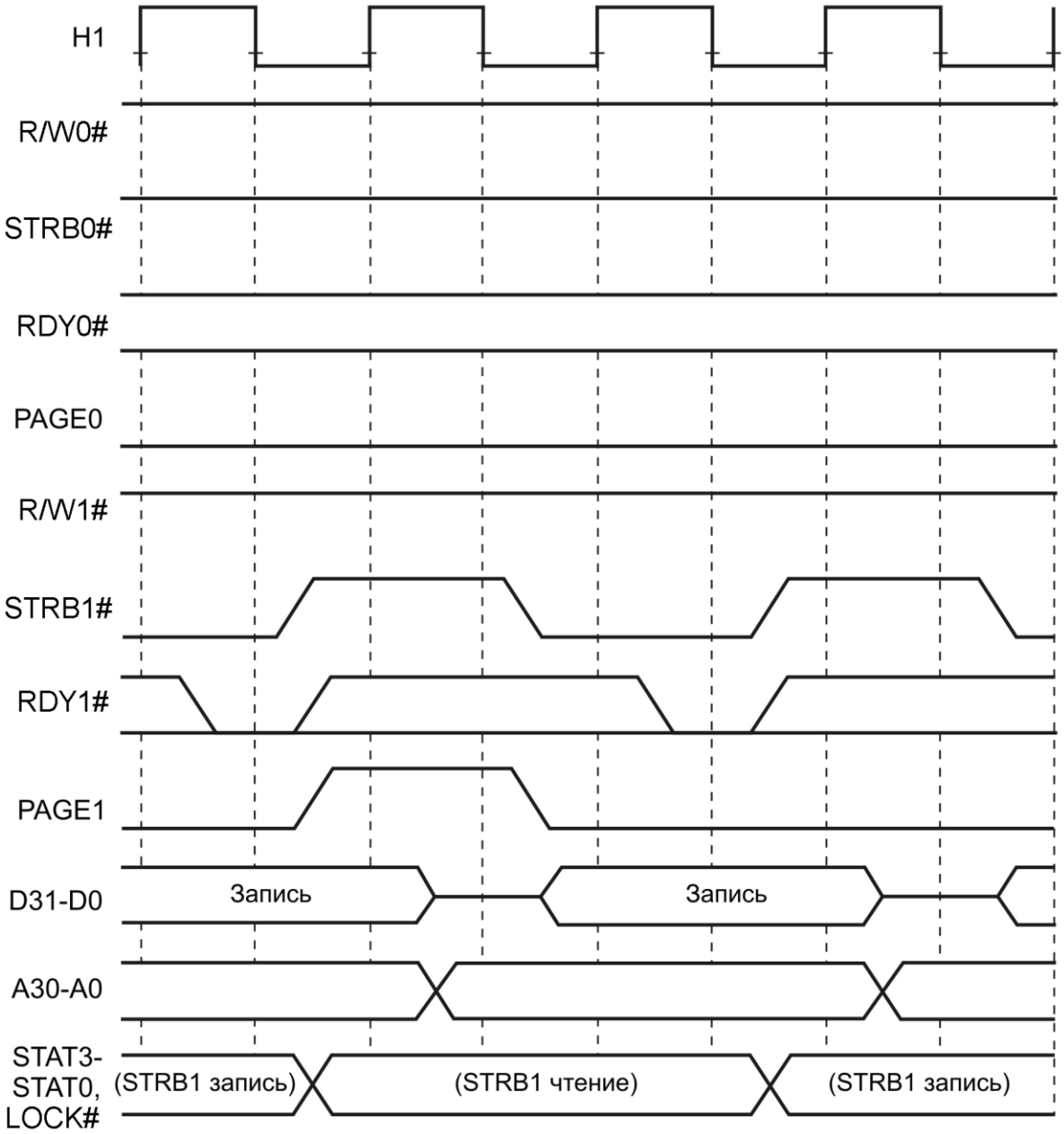


Рисунок 5.80 − Последовательность записи одной страницы, записи другой страницы, записи той же страницы

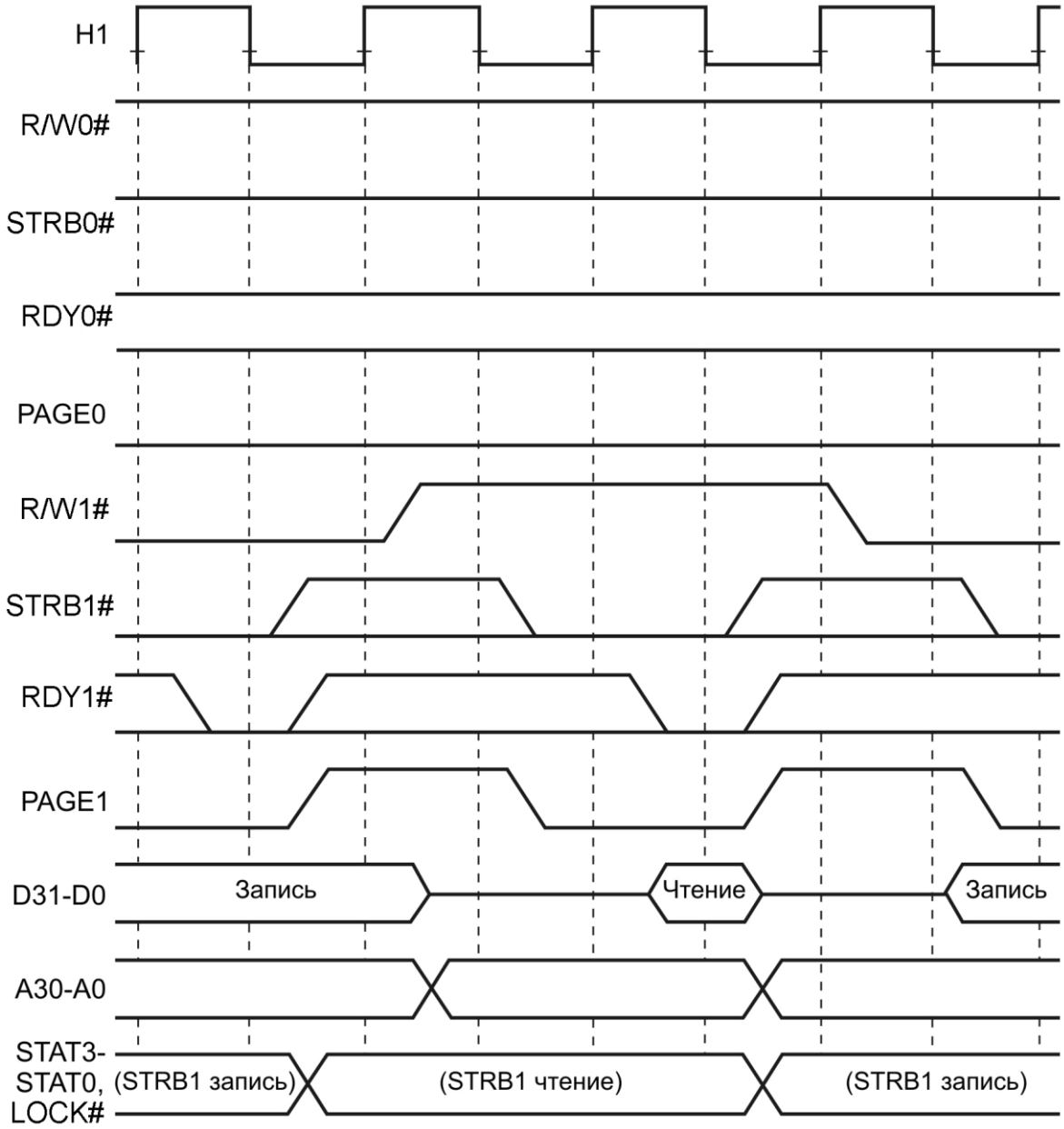


Рисунок 5.81 − Последовательность записи одной страницы, чтения другой страницы, записи другой страницы

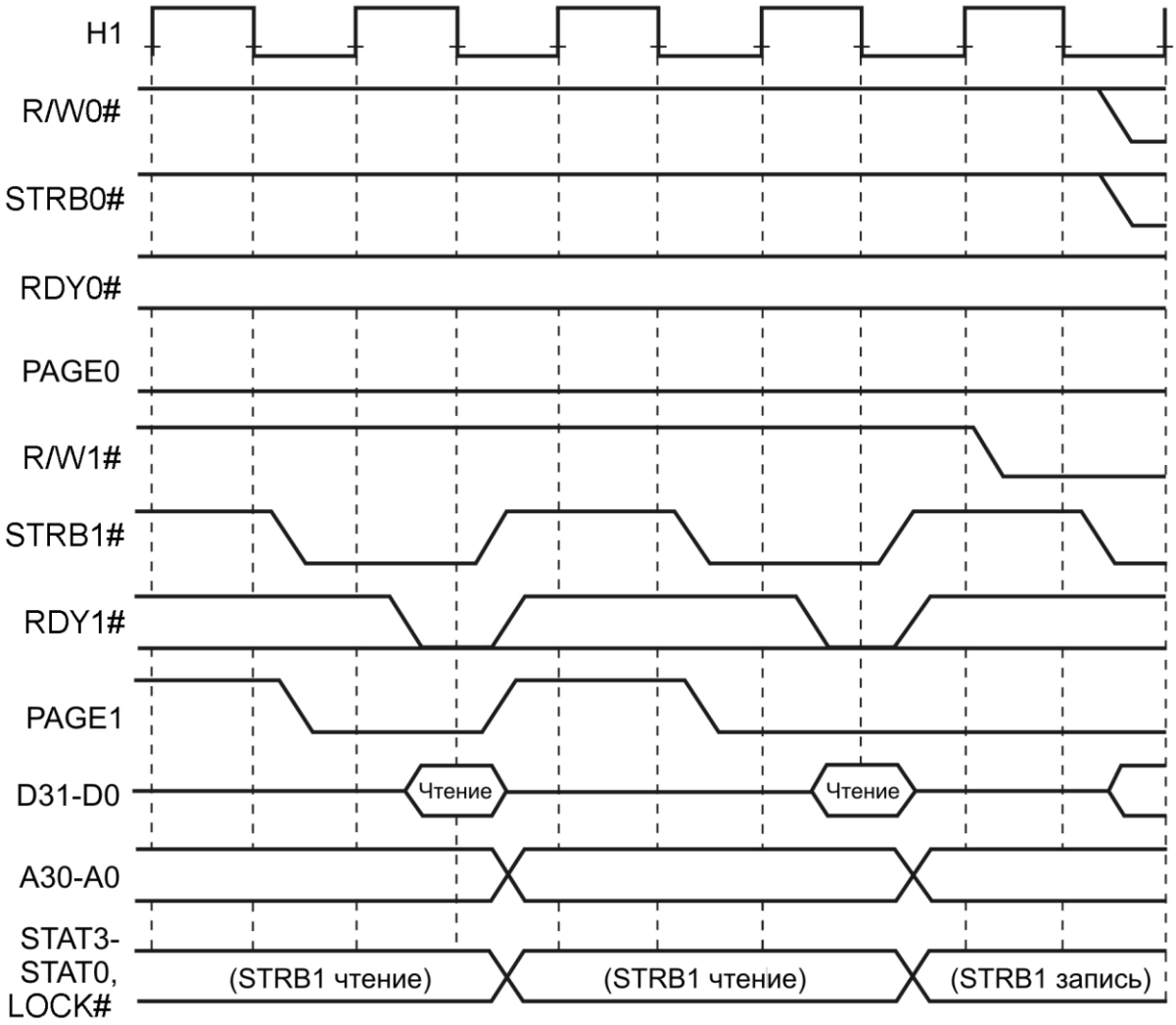


Рисунок 5.82 − Последовательность чтения другой страницы, чтения другой страницы, записи той же страницы

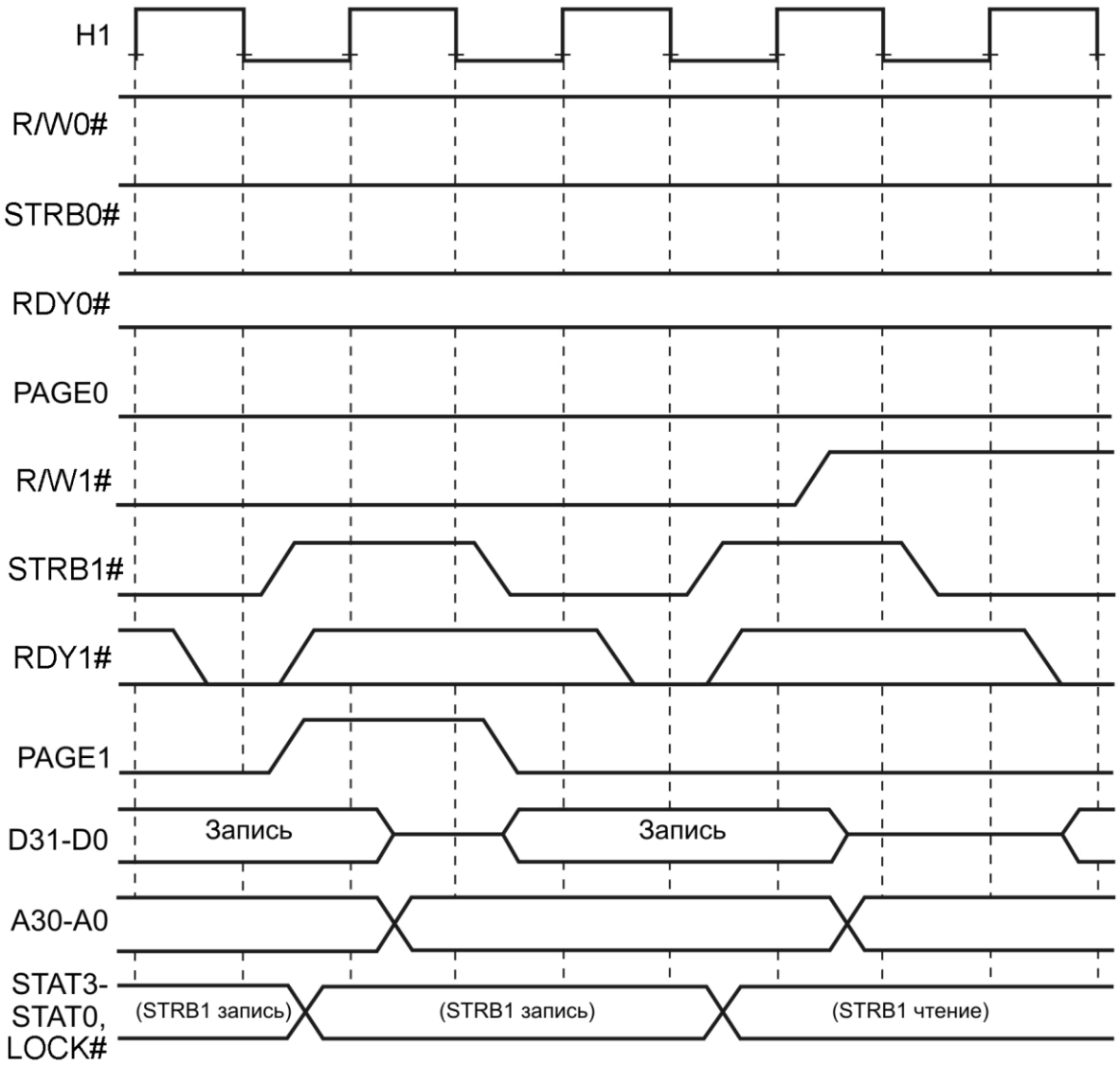


Рисунок 5.83 − Последовательность записи другой страницы, записи другой страницы, чтения той же страницы

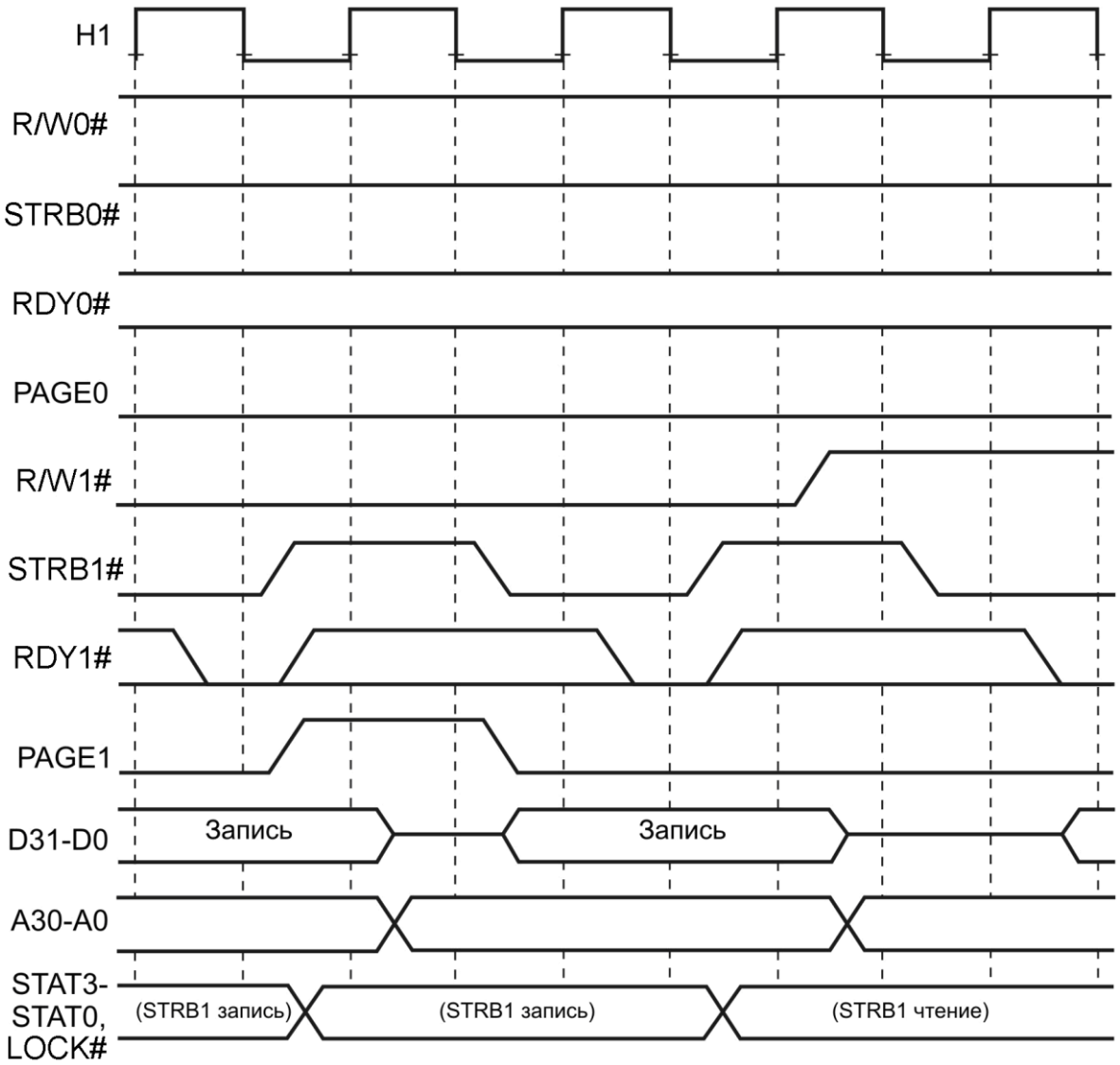


Рисунок 5.84 − Последовательность записи одной страницы, записи другой страницы, чтения той же страницы

На рисунках 5.85 – 5.89 проиллюстрированы циклы шины в режиме ожидания. Временная диаграмма цикла шины в режиме ожидания аналогична временной диаграмме цикла чтения. Основное различие – никакие данные не считываются, STRB# удерживается в высоком уровне, RDY# игнорируется.

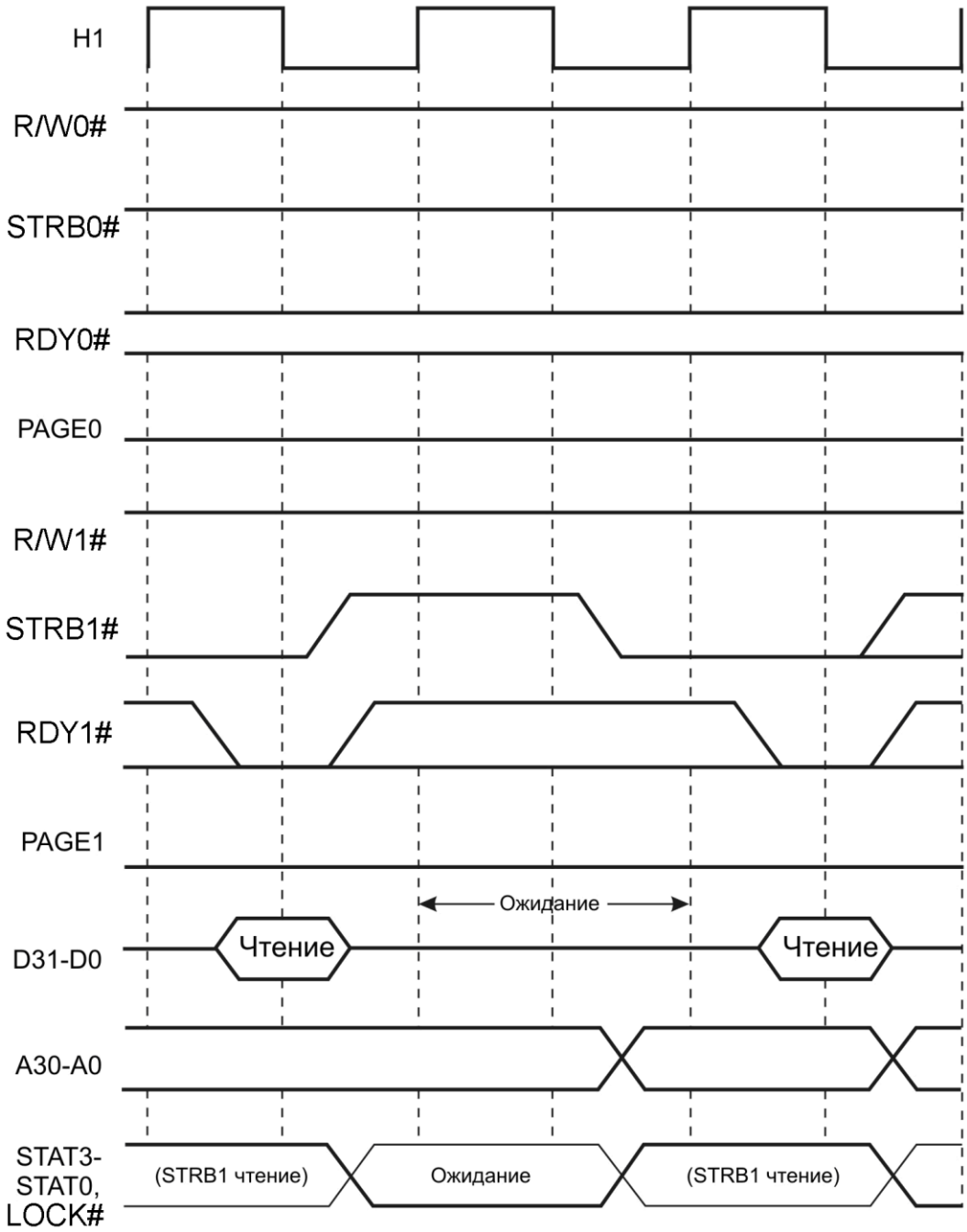


Рисунок 5.85 − Последовательность чтения одной страницы, один цикл ожидания, чтение той же страницы

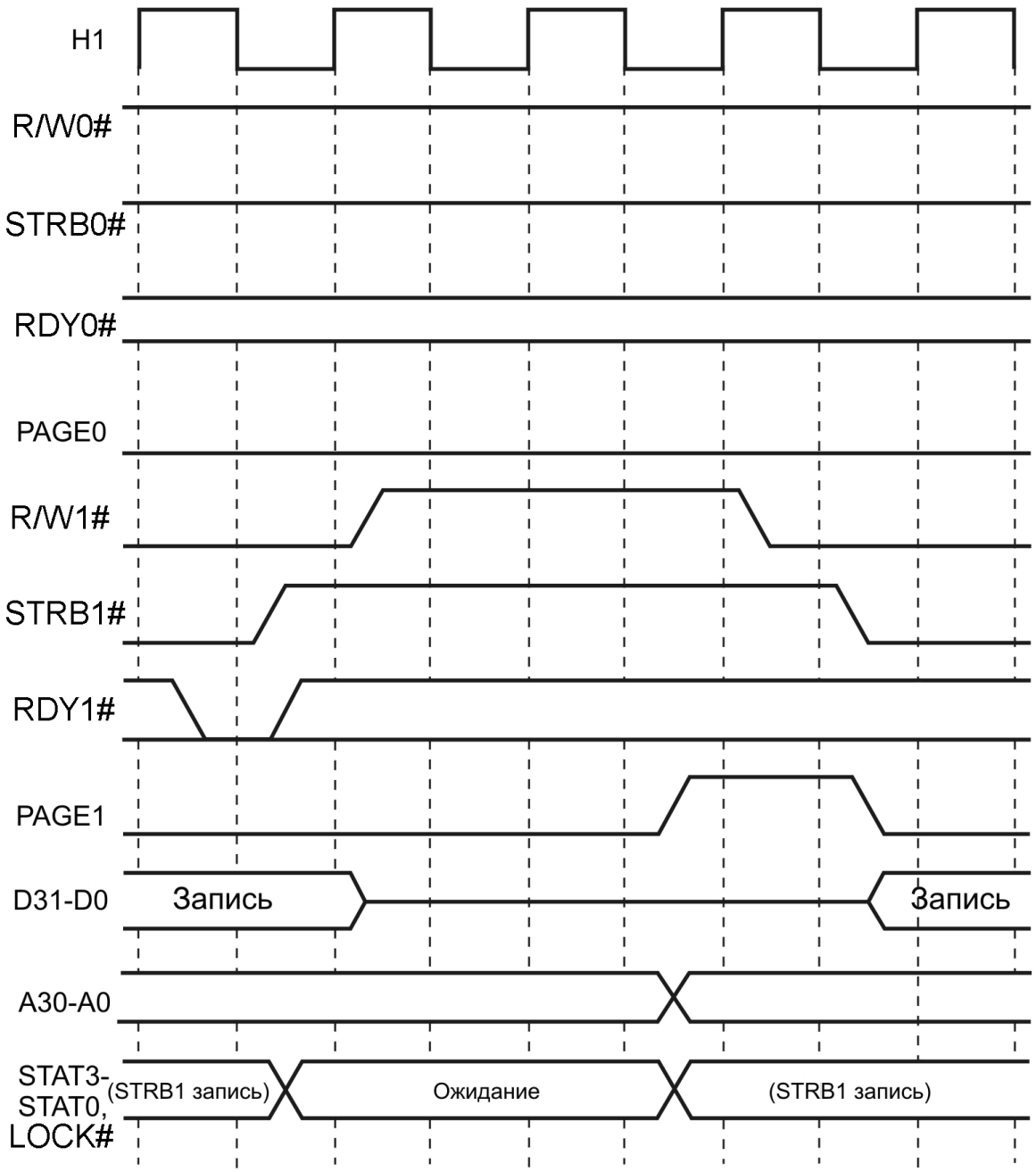


Рисунок 5.86 – Последовательность записи одной страницы, один цикл ожидания, записи другой страницы

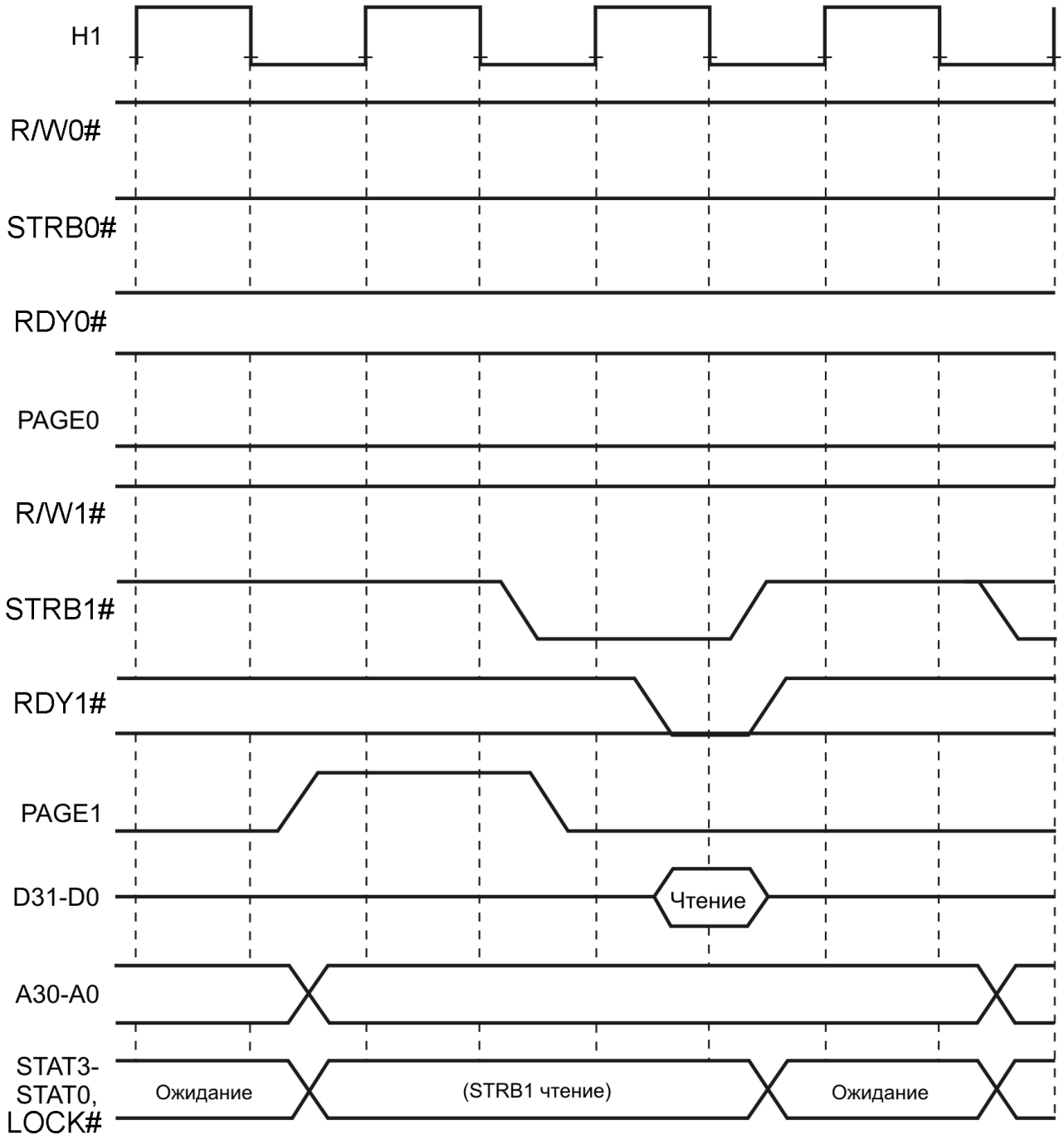


Рисунок 5.87 − Последовательность ожидания, чтения другой страницы, ожидания

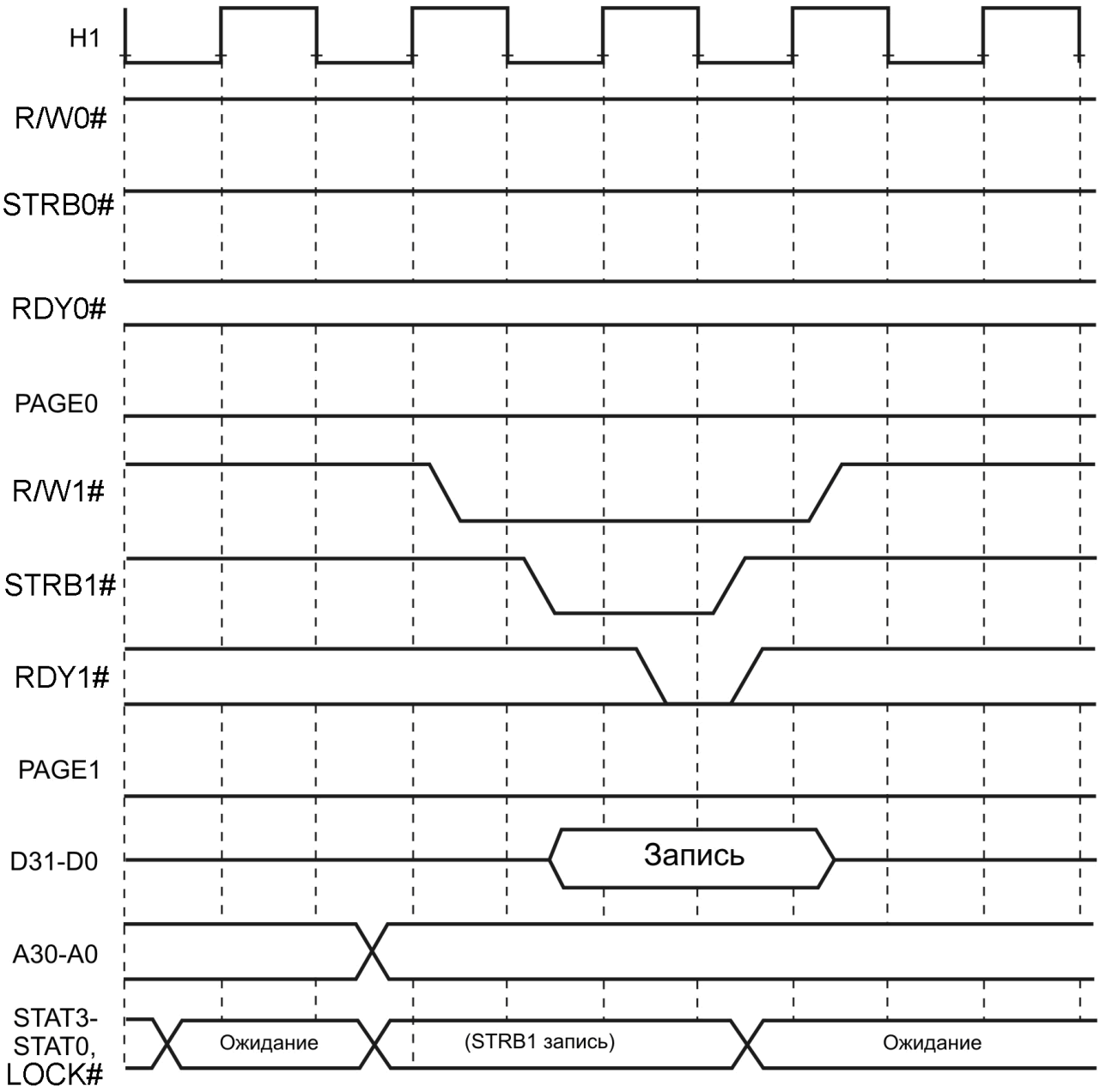


Рисунок 5.88 − Последовательность ожидания, записи одной страницы, ожидания

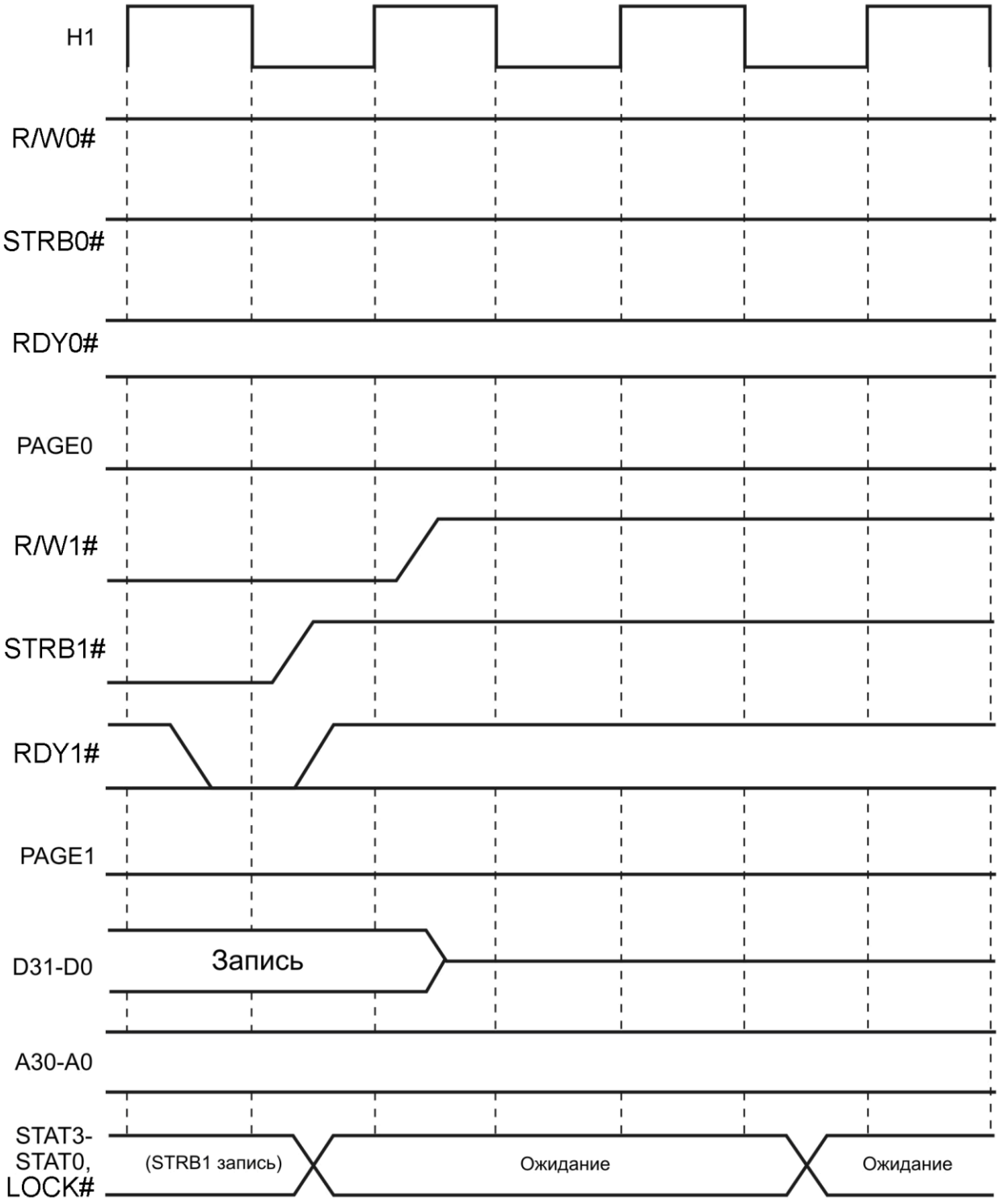


Рисунок 5.89 − Последовательность записи другой или предыдущей страницы, ожидания, ожидания

Рисунок 5.90 показывает чтение по STRB1#, следующее за чтением по STRB0# при STRB SWITCH = 0. Это позволяет осуществлять повторное чтение без добавления цикла между ними, когда они активированы разными стробами.

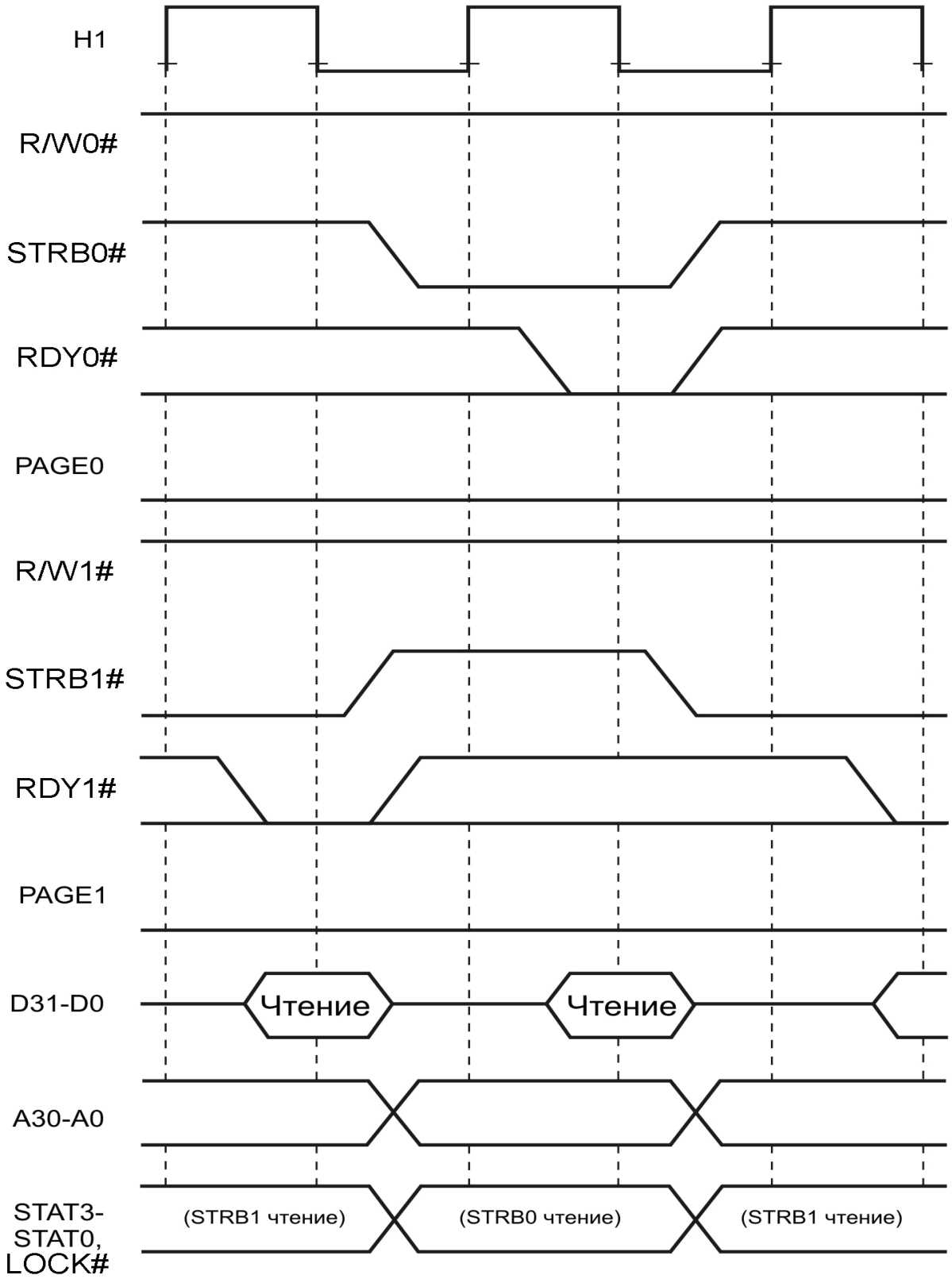


Рисунок 5.90 − Последовательность чтения одной страницы по STRB1#, STRB0#, STRB1# при STRB SWITCH = 0

Рисунок 5.91 аналогичен рисунку 5.90 за исключением того, что второе чтение по STRB1# осуществляется из другой страницы.

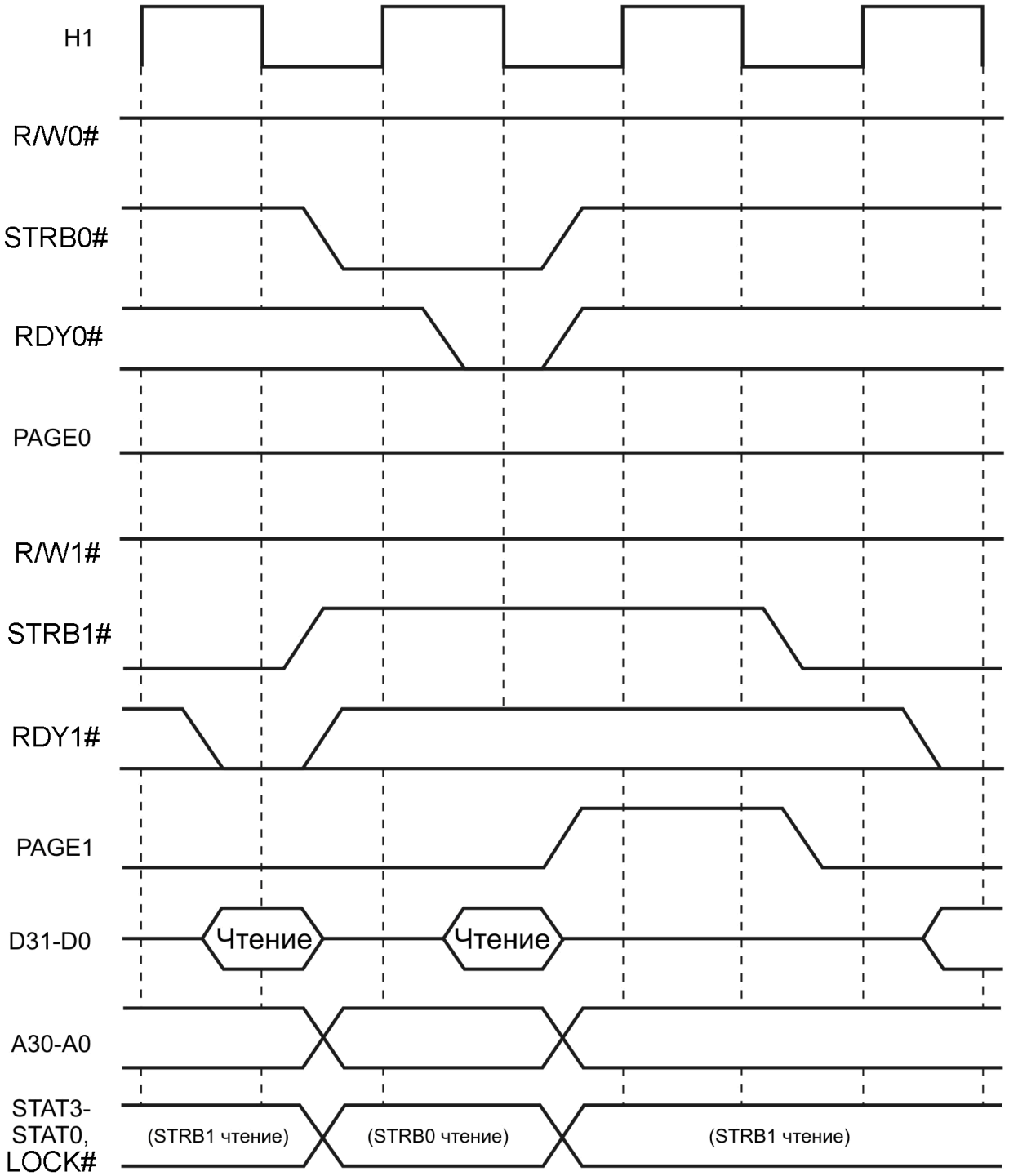


Рисунок 5.91 − Последовательность чтения одной страницы по STRB1#, STRB0#, чтения другой страницы по STRB1# при STRB SWITCH = 0

Рисунок 5.92 показывает чтение по STRB1#, следующее за чтением по STRB0# при STRB SWITCH = 1. В этом режиме добавляется цикл между чтениями, который устанавливает разные стробы. Некоторые конфигурации памяти требуют этот цикл между стробами для предотвращения конфликтов в течение повторного чтения по разным стробам.

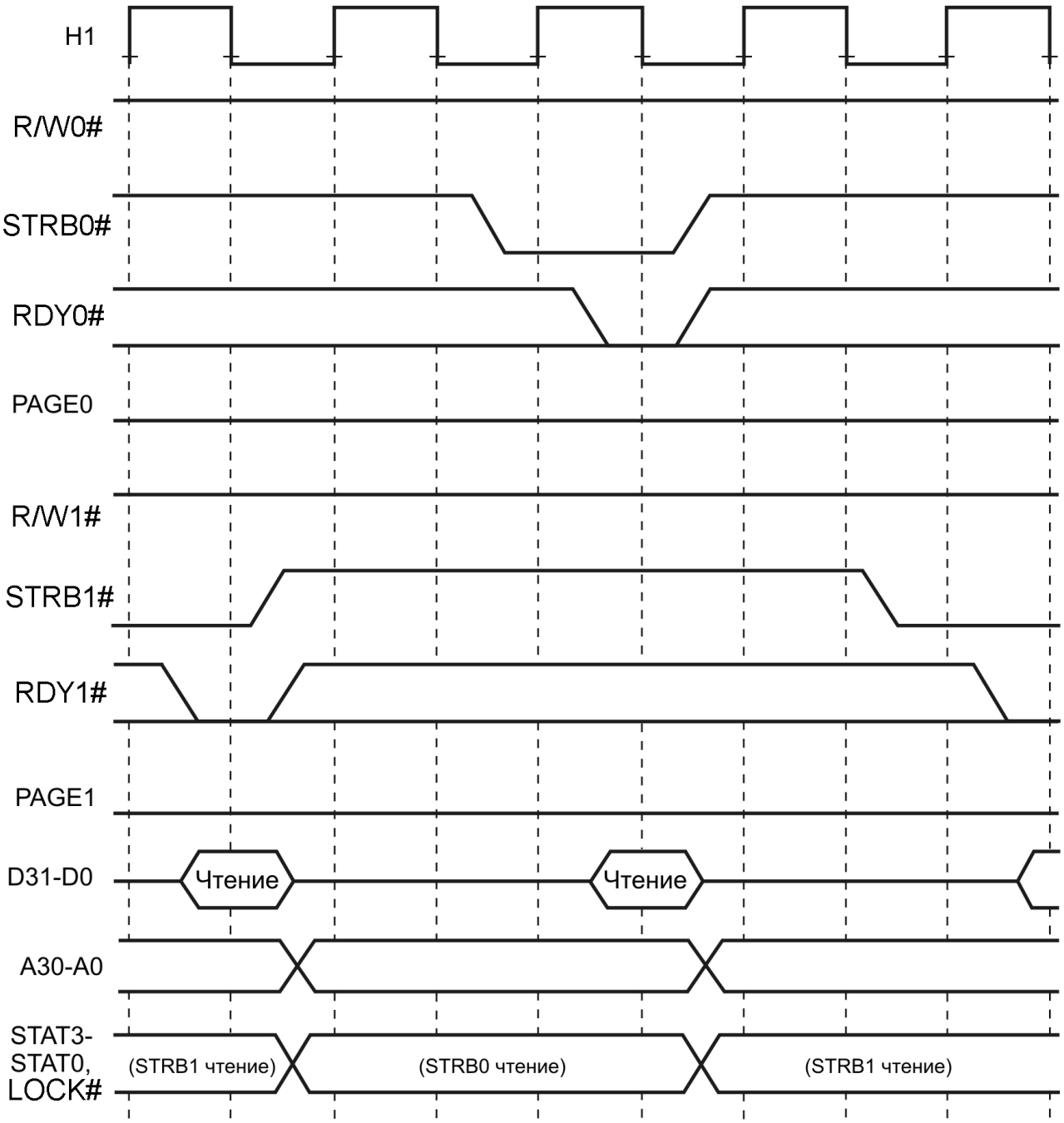


Рисунок 5.92 − Последовательность чтения одной страницы по STRB1#, STRB0#, STRB1# при STRB SWITCH = 1

Рисунок 5.93 аналогичен рисунку 5.92 за исключением того, что второе чтение по STRB1# осуществляется из другой страницы.

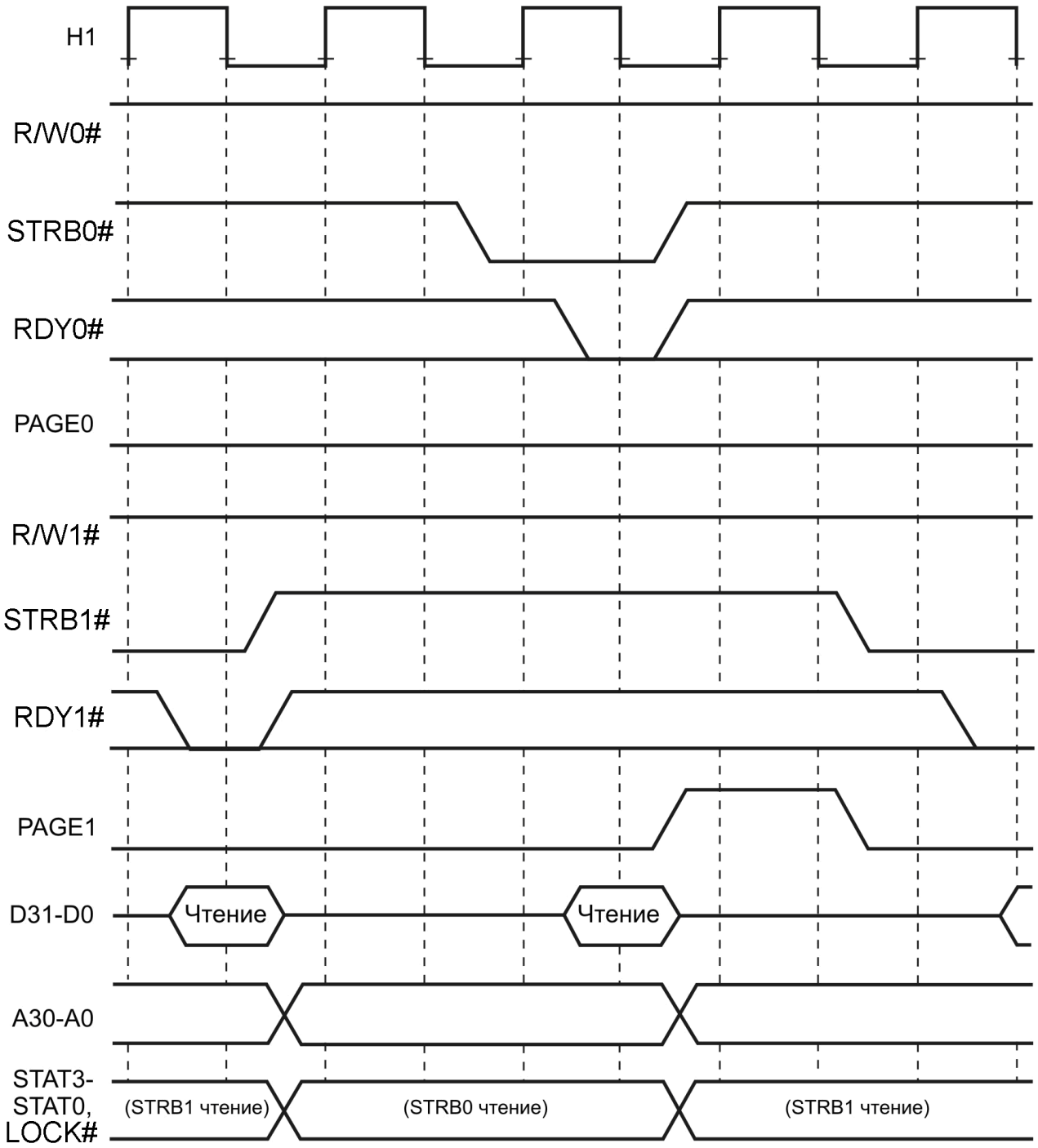


Рисунок 5.93 − Последовательность чтения одной страницы по STRB1#, STRB0#, чтения другой страницы по STRB1# при STRB SWITCH = 1

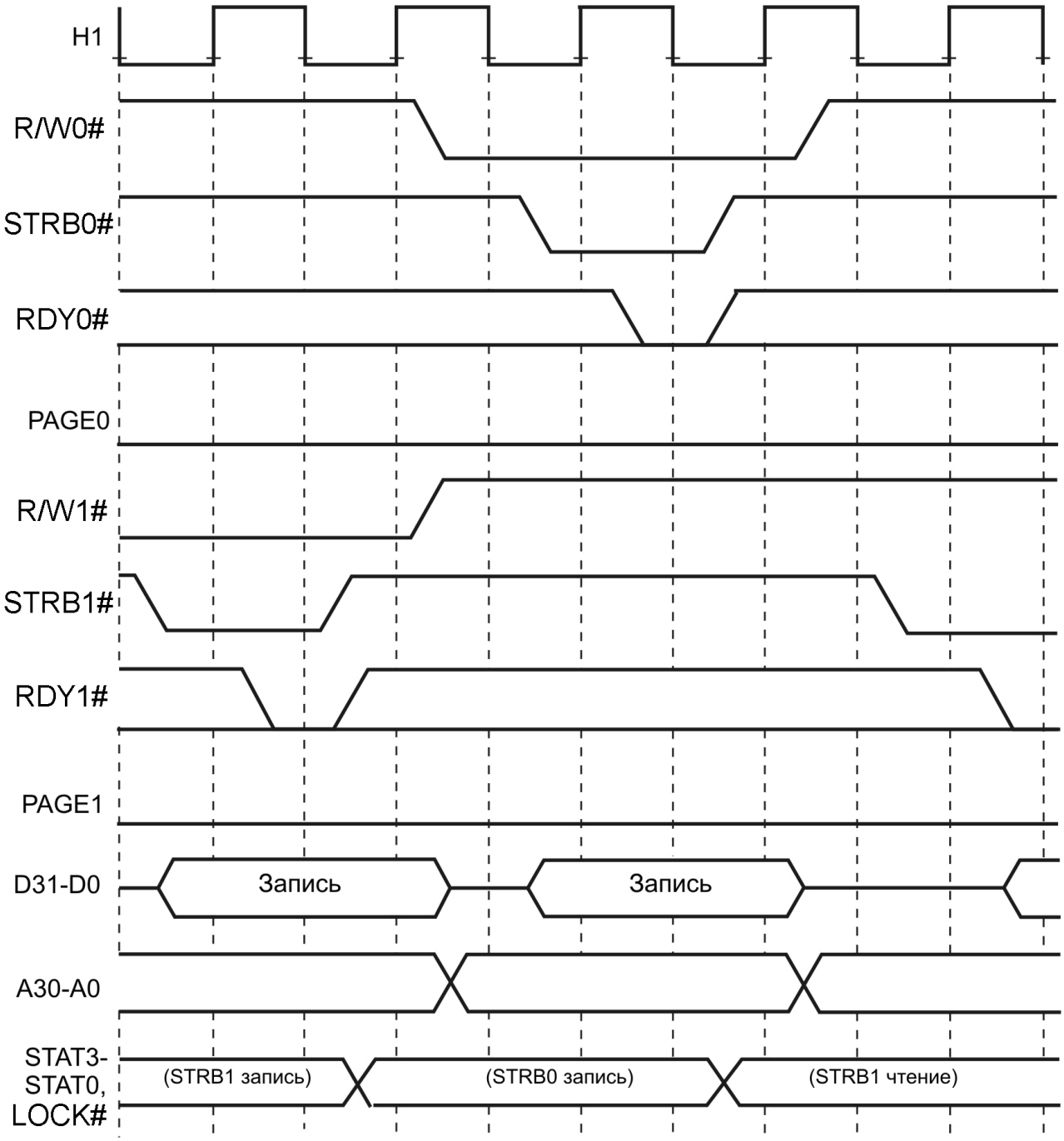


Рисунок 5.94 − Последовательность записи одной страницы по STRB1#, STRB0#, чтения той же страницы по STRB1#

Рисунок 5.95 и рисунок 5.96 показывают соответственно операции чтения и записи с одним циклом ожидания.

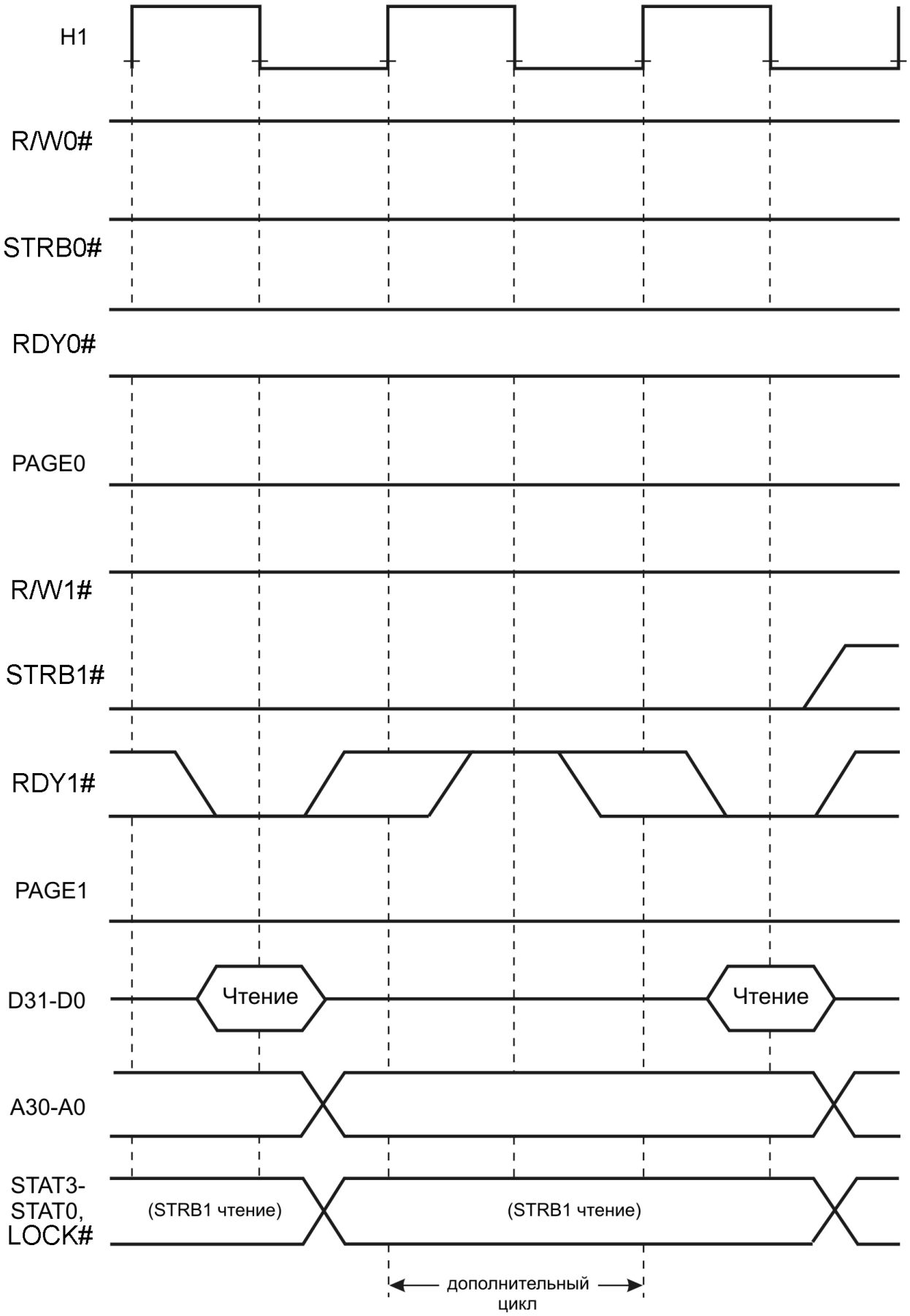


Рисунок 5.95 − Чтение с одним циклом ожидания

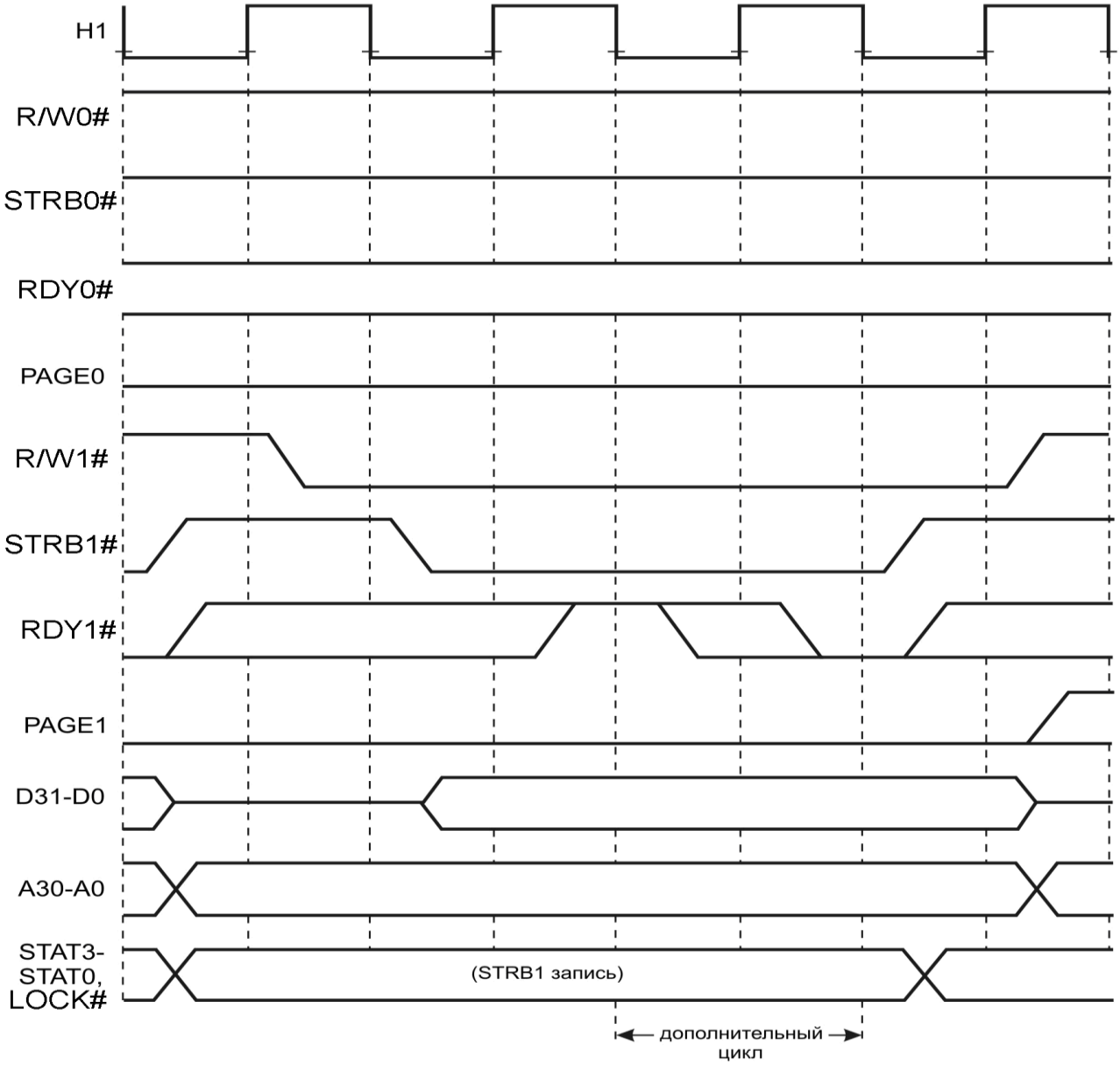


Рисунок 5.96 − Запись с одним циклом ожидания

5.9.6 Использование сигналов разрешения для управления сигнальными группами

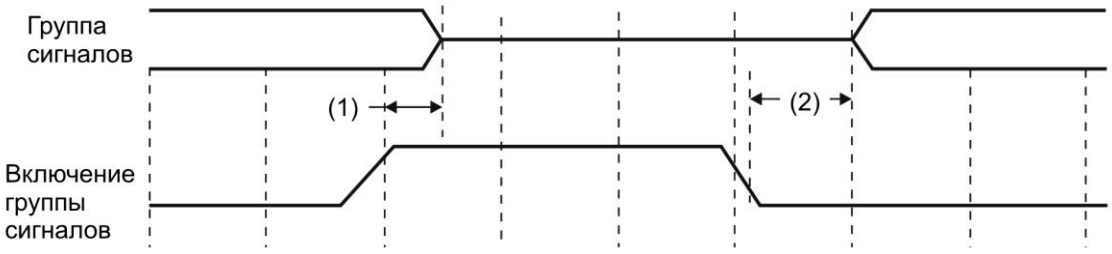


Рисунок 5.97 − Использование сигналов разрешения для установки сигнальных групп в состояние высокого импеданса

Рисунок 5.97 показывает, как разрешающий сигнал управляет соответствующей сигнальной группой. Например, сигнал DE# управляет сигналами данных внешнего глобального интерфейса. Сигналы разрешения – не синхронизованные входа, которые выключают соответствующие выходные буферы. После перехода сигнала разрешения в высокий уровень плюс время (1) на рисунке 5.97, соответствующая группа сигналов переходит высокоимпедансное состояние. Затем после перехода сигнала разрешения в низкий уровень плюс время (2) на рисунке 5.97, группа сигналов выходит из высокоимпедансного состояния. Если сигнальная группа уже была в высокоимпедансном состоянии перед тем, как сигнал разрешения перешёл в высокий уровень, 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 вы планируете использовать внутренне генерируемые состояния ожидания, проверяйте, чтобы данные не читались с шины и не записывались в неё, когда она отключена. При внутренне генерируемых состояниях ожидания шина может быть в режиме высокоимпедансного состояния. В этом случае записанные данные не будут доступны внешне, а читаемые данные в любом случае будут иметь состояние высокого импеданса.

5.9.7 Операции блокировки

Одна из большинства конфигураций параллельно выполняемых команд разделяет глобальную память между процессорами. Для многопроцессорной системы доступ к глобальной памяти разделён между процессорами одинаково, при этом необходима выборка арбитража и подтверждение связи. Подробнее в 5.9.7.5.

Пять инструкций ядра процессора 1867ВЦ8Ф1 определены как операции блокировки. Используя внешние сигналы, эти инструкции обеспечивают мощные механизмы синхронизации. Они также гарантируют целостность коммуникации и приводят к быстродействующей операции. Группа инструкций блокировки представлена в таблице 5.30.

Таблица 5.30 – Операции блокировки

Инструкция	Описание	Операция
LDFI	Загрузка значения с ПЗ из памяти в регистр; блокировка при доступе к внешней памяти	Сигнал блокировки src → dst
LDII	Загрузка целого из памяти в регистр; блокировка при доступе к внешней памяти	Сигнал блокировки src → dst
SIGI	Загрузка значения с ПЗ из памяти в регистр; блокировка при доступе к внешней памяти	Сигнал блокировки Сброс блокировки
STFI	Сохранение значения с ПЗ из регистра в память; блокировка при доступе к внешней памяти	src → dst Сброс блокировки
STII	Сохранение целого из регистра в память; блокировка при доступе к внешней памяти	src → dst Сброс блокировки

Операции блокировки используют сигналы глобальной и локальной шины, LOCK# и LLOCK# для отражения текущей выполняемой операции блокировки. Этот сигнал активен (низкий уровень), когда выполняется какая-либо инструкция блокировки из таблицы 5.30.

Внешняя временная диаграмма блокированных загрузок и сохранений такая же, как и для стандартных загрузок и сохранений. Вы можете расширить загрузки и сохранения стандартными обращениями, используя соответствующие сигналы (RDYx# или LRDYx#).

5.9.7.1 LDFI и LDII

Инструкции LDFI и LDII выполняют следующее:

- переводят (L)LOCK# в низкий уровень;
- выполняют LDF или LDI инструкции;
- увеличивают цикл чтения, пока не придёт соответствующий сигнал готовности. Выполняют операцию;
- оставляют (L)LOCK# активным в низком уровне, пока он не будет изменён с помощью STFI, STII, SIGI.

Операции чтения/записи аналогичны другим циклам чтения/записи, исключая специальное использование (L)LOCK#. Операнд src для LDFI и LDII всегда прямой или косвенный адрес. (L)LOCK# устанавливается в 0, только если src расположен вовне ядра процессора 1867ВЦ8Ф1 (т. е. STRB# или LSTRB# активны). Если имеет место доступ к внутренней памяти, (L)LOCK# не устанавливается, и операция выполняется как LDF или LDI из внутренней памяти.

5.9.7.2 STFI и STII

Инструкции STFI и STII выполняют следующее:

- начинается цикл записи. Состояние (L)LOCK# не изменяется. Если он в низком уровне, происходит операция блокировки. Если он в высоком уровне, операция выполняется как STF или STI (без блокировки);
- выполняется инструкция STF или STI и добавляется цикл записи, пока не придёт соответствующий сигнал готовности;
- после цикла записи (L)LOCK# становится неактивным (высокий уровень).

Как и в случае с LDFI и LDII, dst инструкций STFI и STII влияет на (L)LOCK#. Если dst расположен вовне ядра процессора 1867ВЦ8Ф1 (STRB(0,1)# или LSTRB(0,1)# активны), (L)LOCK# устанавливается в 1. Если доступ происходит к внутренней памяти, (L)LOCK# не устанавливается, и операция выполняется как STF или STI во внутреннюю память.

5.9.7.3 SIGI

Инструкция SIGI может использоваться по-разному. В некоторых приложениях, вы можете захотеть изменить сигнализацию внешне, может быть с логикой специального назначения. Если это так, SIGI может использоваться для выполнения одноцикловой блокировки сигнализации. Инструкция SIGI может также использоваться просто для выполнения внешнего чтения и для сигнализации, что соответствующее место в программе достигнуто.

Инструкция SIGI выполняет следующее:

- переводит (L)LOCK# в низкий уровень;
- выполняет LDI инструкцию;
- увеличивает цикл чтения, пока не придёт соответствующий сигнал готовности. Выполняет операцию;
- переводит (L)LOCK# обратно в неактивное состояние (высокий уровень).

Операции блокировки могут использоваться для выполнения циклов ожидания-занятости, для управления многопроцессорным счётчиком, для выполнения простого механизма сигнализации или для синхронизации между двумя ядрами процессора 1867ВЦ8Ф1. Следующие ниже примеры показывают полезность операций блокировки.

5.9.7.4 Примеры блокировки

Примеры в этом разделе показывают, как использовать операции блокировки:

- цикл ожидания-занятости для синхронизации процессоров программно (пример 5.61);
- счётчик, распределённый между совокупностью процессоров, определяющий число раз выполнения заданий процессорами (пример 5.62);
- сигнализация для программирования критических секций (пример 5.63 и 5.64).

Пример 5.61 показывает применение цикла ожидания-занятости. Ядро процессора 1867ВЦ8Ф1 остаётся в цикле, пока другой процессор не запишет 0 в @LOCK. Если размещение LOCK блокировано для критической секции программы, и не ноль означает, что блокировка занята, алгоритм для цикла ожидания-занятости может использоваться, как показано.

Пример 5.61 – Цикл ожидания-занятости

```
        LDI     1, R0           ; помещает 1 в R0
L1:     LDII    @LOCK, R1       ; загружает значение блокировки в R1
        STII    R0, @LOCK       ;  устанавливает значение блокировки в 1
        BNZ     L1              ; если R1 (предыдущее значение блокировки) ≠ 0,
                                ; считывает его снова
```

Пример 5.62 иллюстрирует, как размещение COUNT может содержать счётчик количества выполнений операции. Операция может выполняться любым процессором системы. Если счёт равен нулю, процессор ожидает, пока станет не нулём перед началом выполнения. Пример также показывает, как корректно модифицировать COUNT.

Пример 5.62 – Изменение счётчика заданий

```
        LDI     0, R0
WAIT    LDII    @COUNT, R1      ; чтение текущего значения счётчика
        BZD     WAIT            ; если COUNT = 0, новая попытка
        LDNZ    1, R0           ; если COUNT ≠ 0, декремент COUNT
        SUBI    R0, R1
        STII    R1, @COUNT      ; обновление COUNT
```

Рисунок 5.98 иллюстрирует распределение глобальной памяти в системе процессоров 1867ВЦ8Ф1 с применением инструкций блокировки, приведённым в примерах 5.63, 5.64.

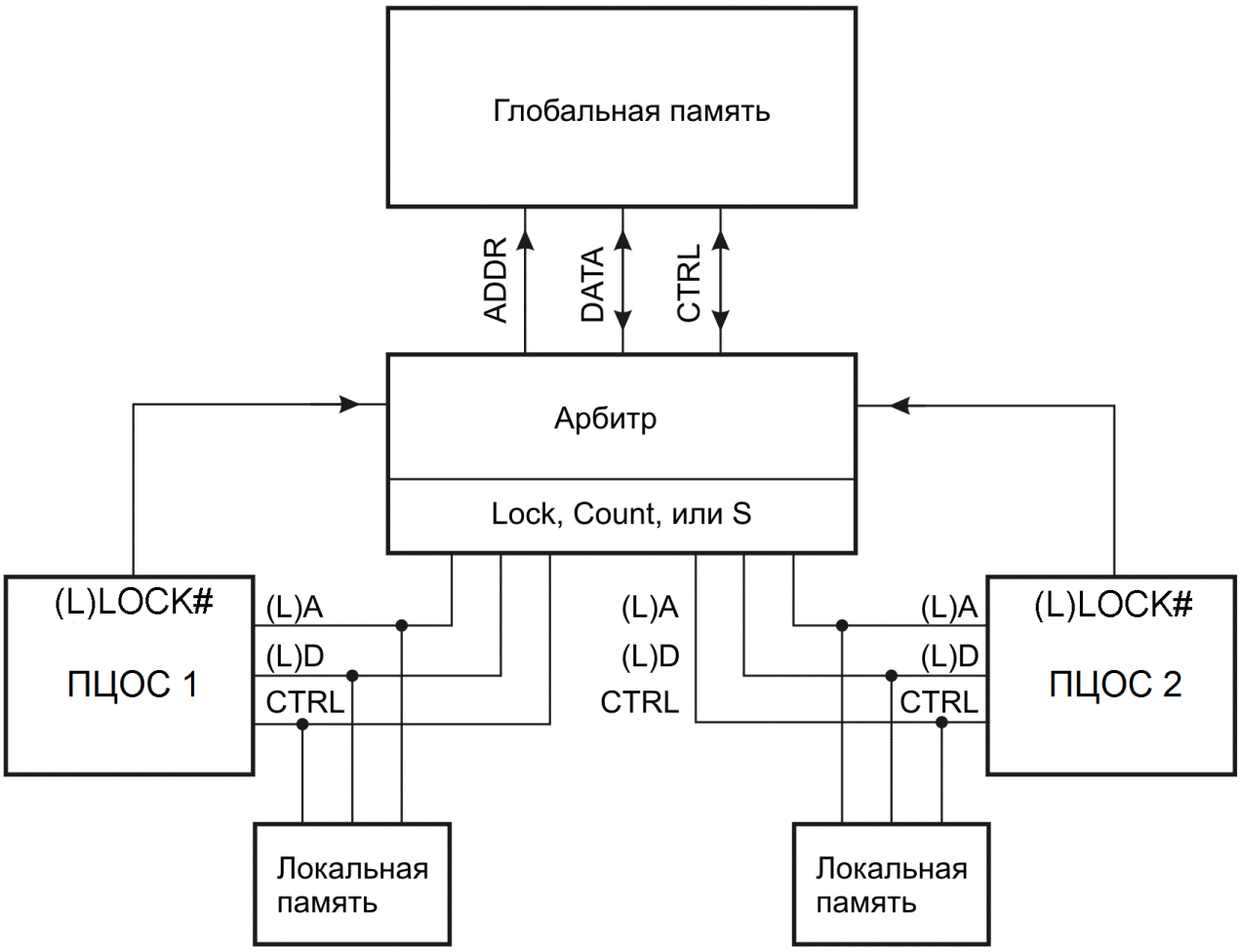


Рисунок 5.98 − Система микропроцессоров 1867ВЦ8Ф1
с распределённой между ними глобальной памятью

Пример 5.63 – Применение V(S)

```
V:      LDII    @S, R0
        ADDI    1, R0
        STII    R0, @S          ; S + 1 → S
```

Пример 5.64 – Применение P(S)

```
        LDI     0, R0
P:      LDII    @S, R1          ; чтение текущего значения семафора
        BZD     P               ; если S = 0, переход к P и новая попытка
        LDNZ    1, R0           ; если S ≠ 0, декремент S
        SUBI    R0, R1
        STII    R1, @S          ; обновление S
```

Иногда сразу несколько процессоров (процессорных ядер) одновременно пытаются получить доступ к распределённым между ними данным или другим общим ресурсам. Часть программного кода, обслуживающая такие одновременные обращения, называется критической секцией.

Для упрощения программирования критических секций можно использовать семафоры. Семафоры – это переменные, которые могут принимать только неотрицательные целые значения. Для семафоров определены две простые неделимые операции (где S – семафор):

V(S): S + 1 → S
P(S): P: если (S = = 0), перейти к P
иначе S – 1 → S

Неделимость V(S) и P(S) означает, что, когда происходит обращение к ним и изменение семафора, происходит только это.

Для того, чтобы войти в критическую секцию, операция P выполняется над общим семафором, например, S (S инициализируется в 1). Первый процессор, выполняя 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 P(S) успешно.

Код ядра процессора 1867ВЦ8Ф1 для V(S) приведён в примере 5.63, код для P(S) приведён в примере 5.64. Сравните код в примере 5.64 с кодом в примере 5.62, в котором не используются семафоры.

5.9.7.5 Временные диаграммы сигналов шины и блокировки шины

Временные диаграммы для сигналов LOCK#и LLOCK# такие же, как и для STAT(3 − 0) и LSTAT(3 − 0). Инструкции LDII, LDFI, STII, STFI, SIGI изменяют сигналы блокировки шины, только когда происходит обращение к внешней памяти.

LDII, LDFI, SIGI сбрасывают LOCK# или LLOCK# в 0 в начале цикла чтения по заднему фронту H1. LDII, LDFI, SIGI устанавливают LOCK# или LLOCK# в 1 в конце цикла доступа по заднему фронту H1.

Рисунки 5.99 – 5.102 показывают временные диаграммы характеристик шины для внешнего доступа, использующего STII, LDII, STFI, LDFI, SIGI.

На рисунке 5.99 представлен пример внешнего доступа LDII или LDFI.

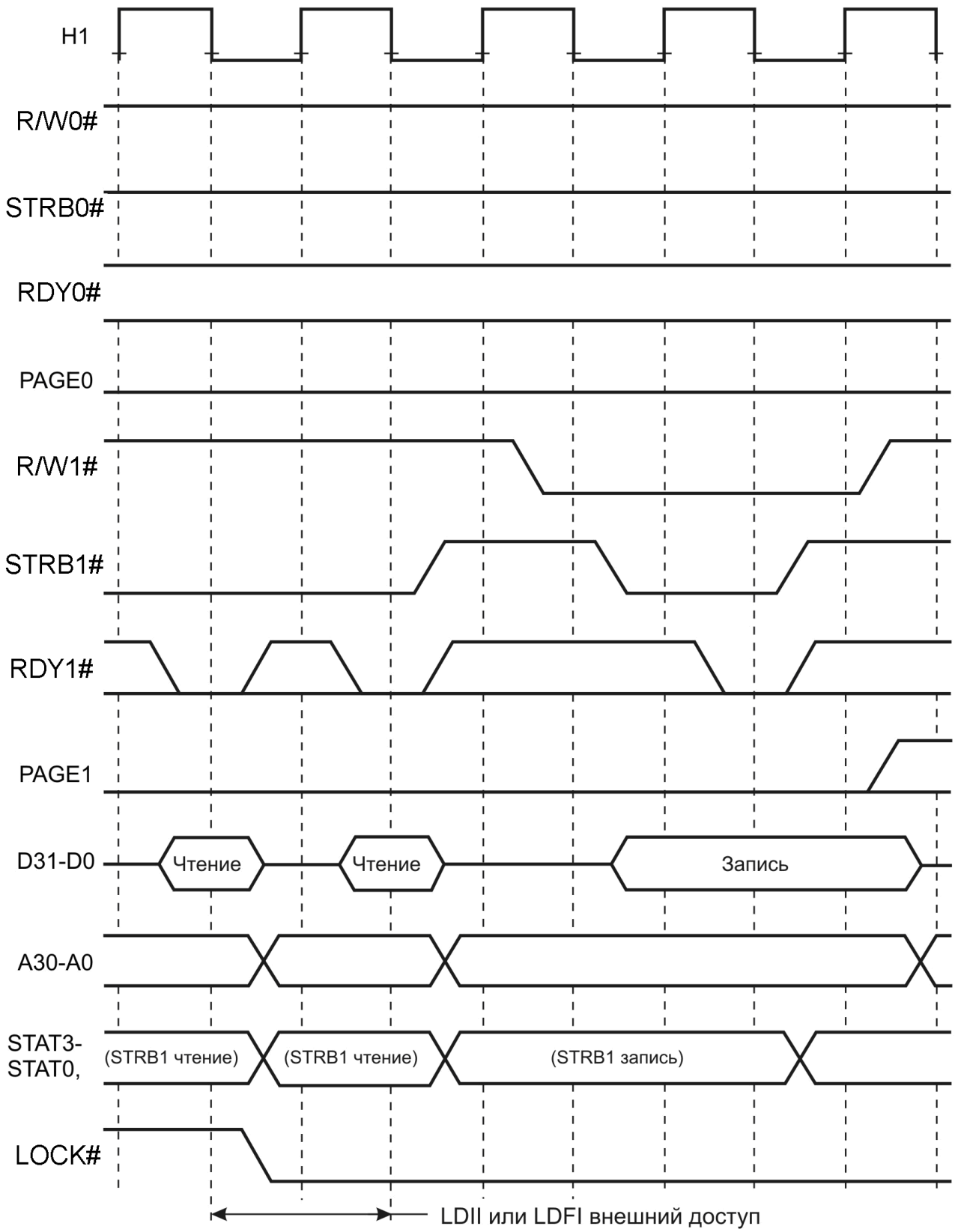


Рисунок 5.99 − Внешний доступ LDII или LDFI

Рисунок 5.100 – пример внешнего доступа STII или STFI, следующего за предшествующей загрузкой (рисунок 5.99), и цикл ожидания. Это временная диаграмма для последовательности блокированных загрузок/сохранений.

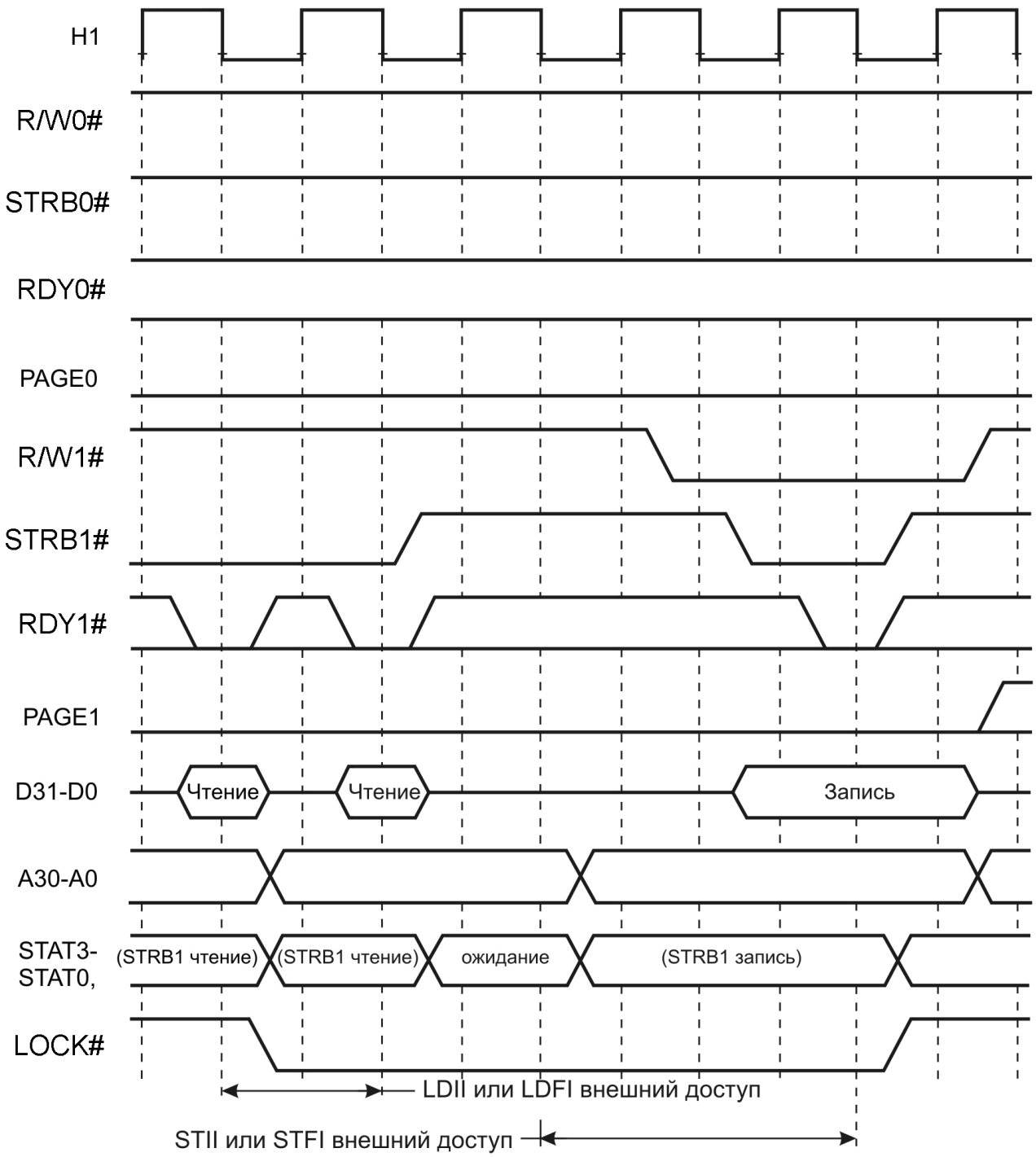


Рисунок 5.100 − Внешний доступ LDII или LDFI и STII или STFI

Рисунок 5.101 – пример внешнего доступа SIGI.

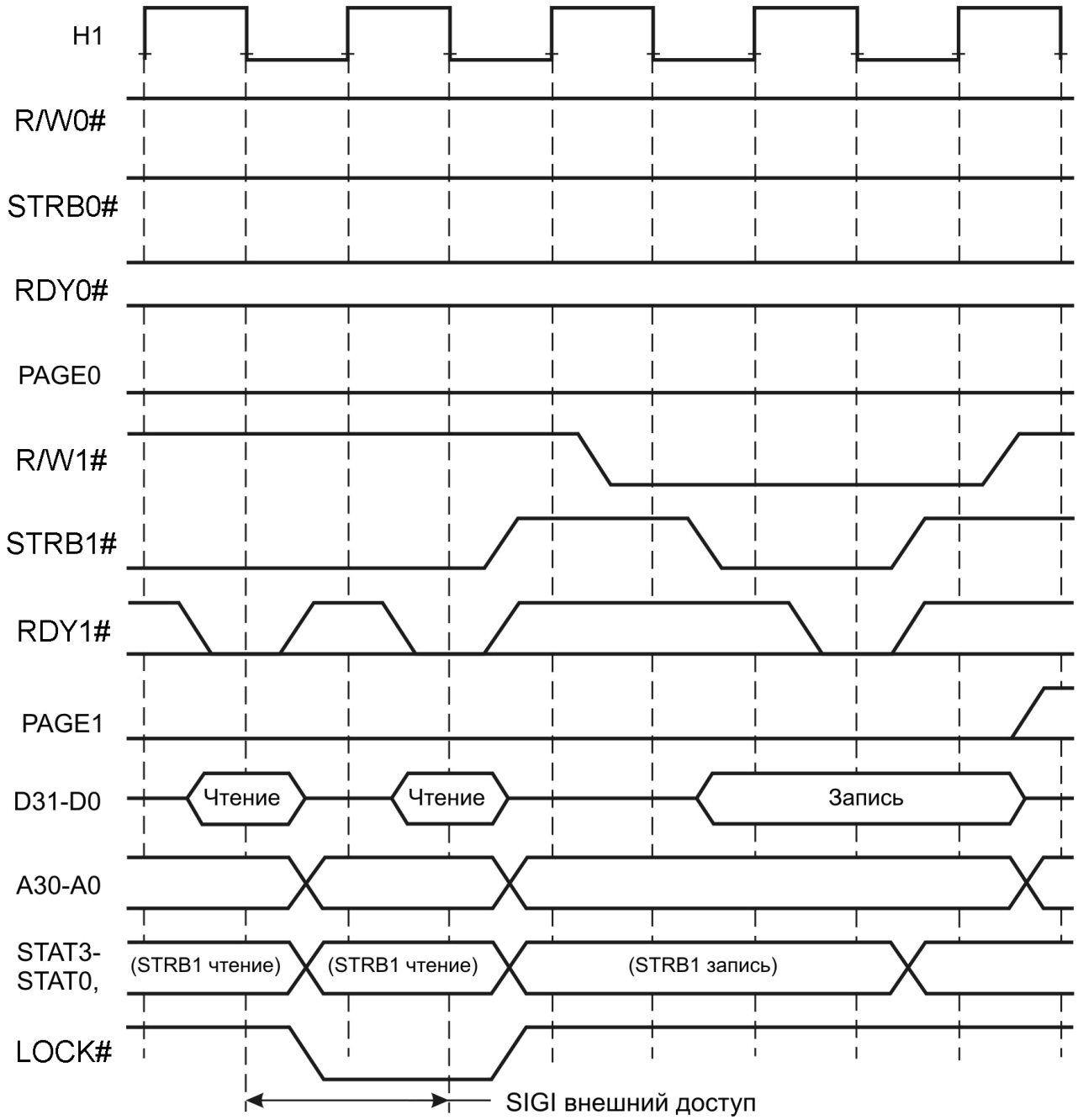


Рисунок 5.101 − Временная диаграмма внешнего доступа SIGI

Рисунок 5.102 показывает временную диаграмму для SIGI, если сигнал LOCK# уже пребывает в низком уровне. Это может произойти, если SIGI следует за LDII инструкцией. Так как LOCK# уже в низком уровне, единственное, что может SIGI – это перевести его в высокий уровень.

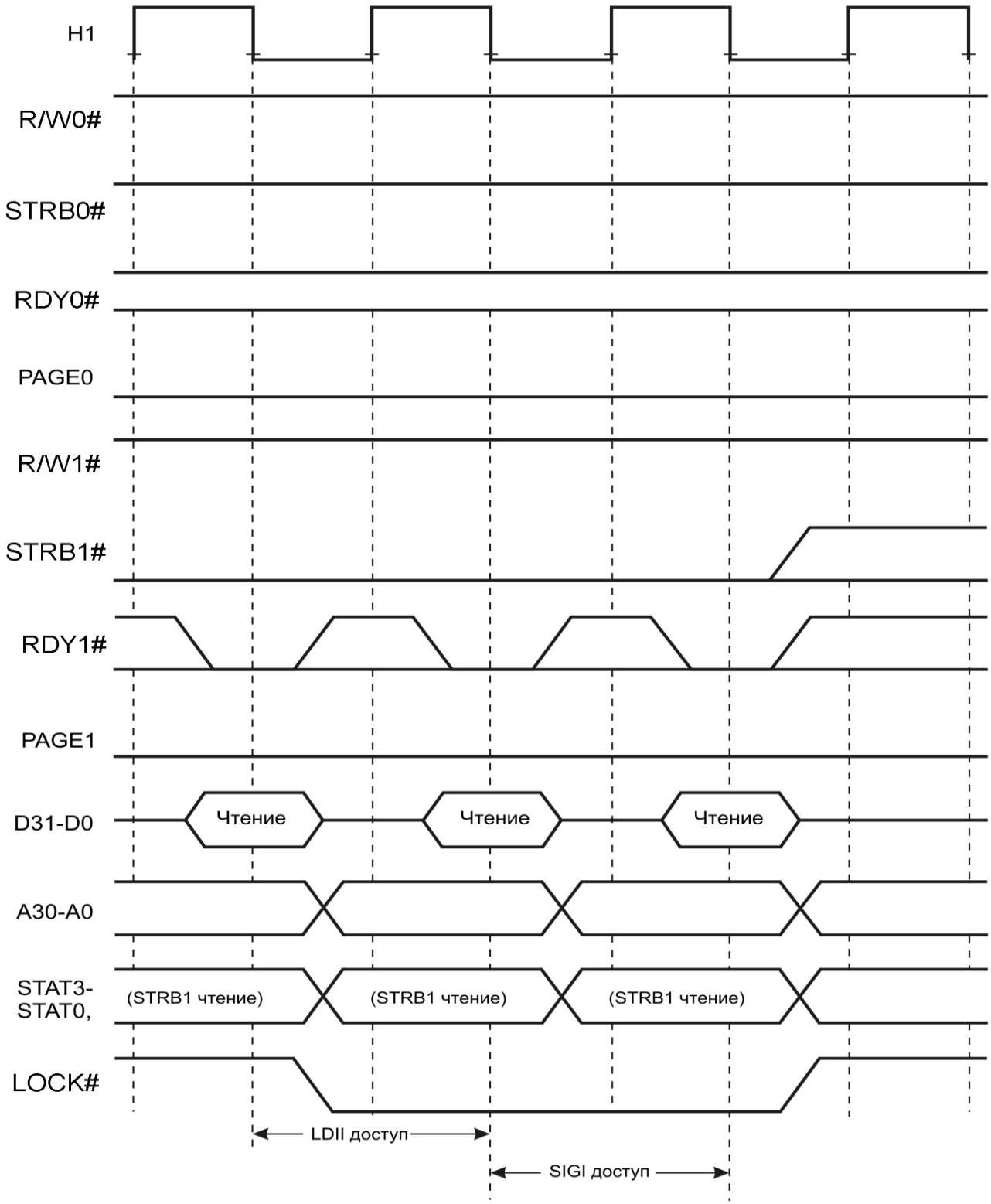


Рисунок 5.102 − SIGI, если LOCK# уже в низком уровне

5.9.8 Временная диаграмма IACK

Сигнал IACK# управляется инструкцией IACK. Его временная диаграмма аналогична сигналу LOCK#, когда используется инструкция SIGI. Поведение IACK# похоже на поведение LOCK# или STAT. Единственное различие – существует только один IACK# сигнал.

Временная диаграмма для IACK# показана на рисунке 5.103. Как и операции блокировки, IACK инструкция воздействует на IACK# только при внешнем доступе.

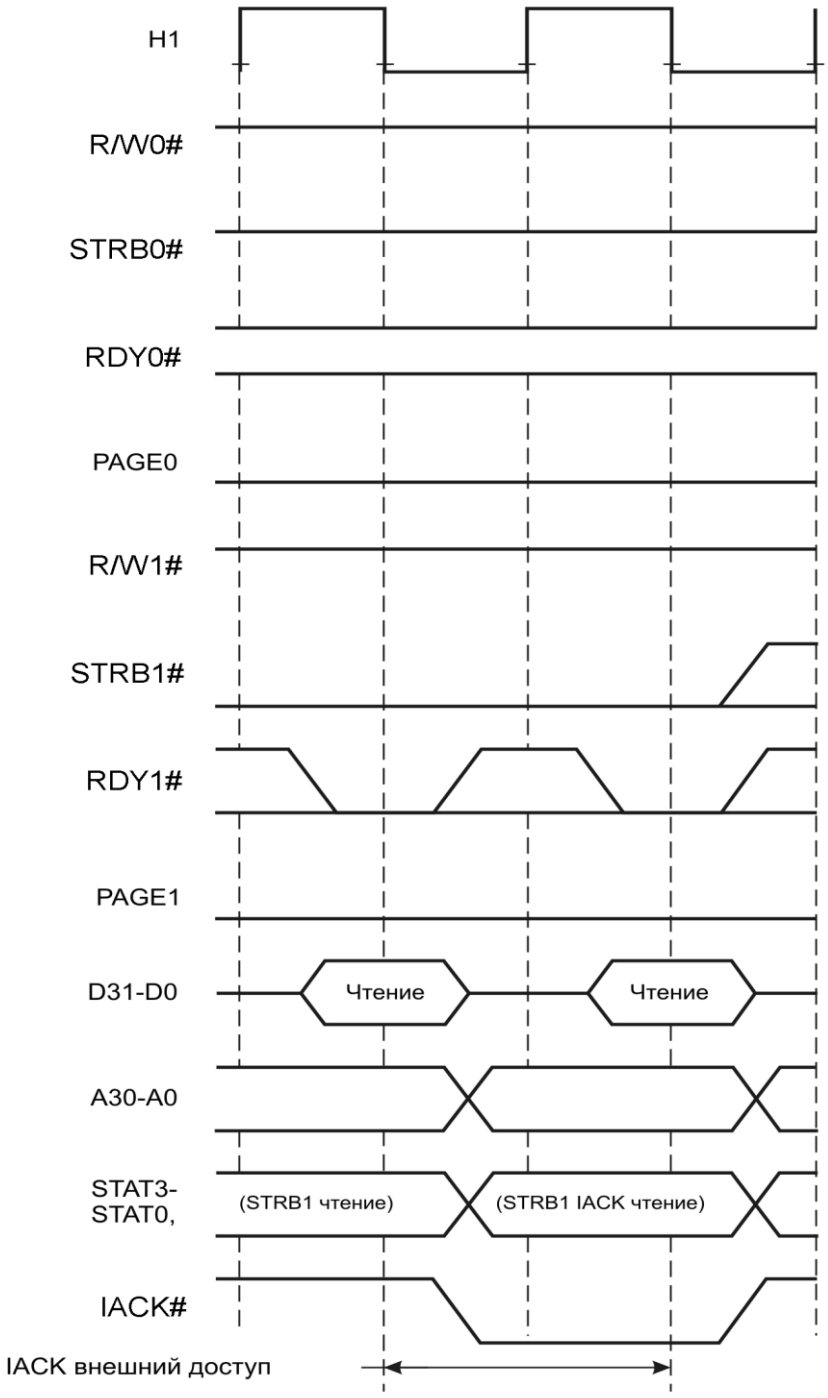


Рисунок 5.103 − Временная диаграмма IACK#

5.10 Загрузчик пользовательской программы

Начальный загрузчик (bootloader), реализованный во внутрикристальных масочных ПЗУ процессорных ядер ИС 1867ВЦ8Ф1, используется для передачи кода пользовательской программы с того или иного внешнего источника во внутреннюю или внешнюю оперативную память с последующим запуском этого кода. Для обеспечения необходимой гибкости при реализации системы, информация от внешнего источника может передаваться как параллельно (из внешней памяти или через коммуникационные порты), так и последовательно (через периферийные устройства UART, Ethernet и USB). Загрузчик активируется при условии, что RESETLOC(1, 0)_x (x = 1, 2) переводится в низкий уровень, а на ROMEN_x установлен высокий уровень (внутреннее ПЗУ разрешено). Выбор варианта (источника) загрузки кода зависит от состояния выводов IIOF(3 – 0)_x#.

5.10.1 Описание загрузчика

Программный загрузчик размещается во внутреннем ПЗУ ядра ПЦОС начиная со стартового адреса 0x0. Исполняемый код (листинг программы) загрузчика приведён в 5.10.7.

5.10.2 Выбор режима

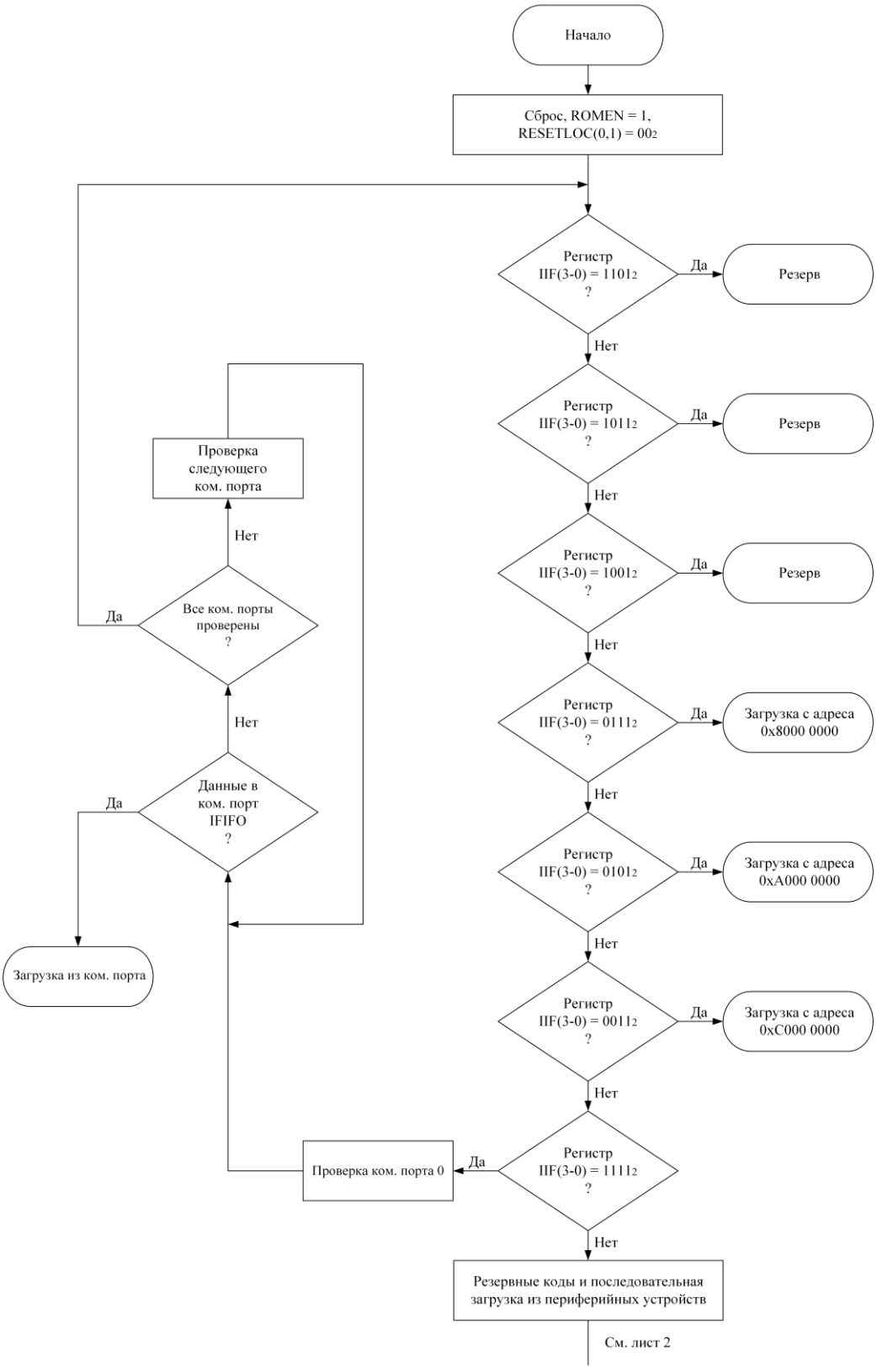
Основной функцией загрузчика является закачивание программы из памяти, от одного из коммуникационных портов или периферийных устройств UART, Ethernet и USB. Выбор источника загрузки определяется выводами IIOF3_1# − IIOF0_1#, IIOF3_2# − IIOF0_2#. Режимы работы загрузчика, в зависимости от состояния этих выводов, представлены в таблице 5.31 и приведены на рисунке 5.104. Кодирование вариантов выбора источника для параллельной загрузки одинаковое для обоих ядер ИС 1867ВЦ8Ф1 (см. таблицу 5.31):

- загрузка из внешней памяти. В этом режиме загрузчик выполняет закачку кода исходной программы из блока памяти следующих разрядностей: 8, 16 или 32 бита. Исходные программы для загрузки должны постоянно находиться в одной из шести предопределённых адресных областей внешней памяти, которые перечислены в таблице 5.31. Стробирование загружаемых программ должно производиться сигналами STRB0# (LSTRB0#), потому что именно эти стробы активны после системного сброса. Рисунок 5.105 показывает порядок установки этого режима работы ИС в ходе работы загрузки из внешней памяти;
- загрузка через коммуникационные порты. Загрузчик ждёт первого ввода данных от одного коммуникационного порта из шести. Формат поступающего программного потока является подобным потоку при загрузке из внешней памяти, за исключением того, что нет шага выбора разрядности входных данных. Рисунок 5.106 показывает порядок установки режима работы ИС в ходе загрузки исходной программы через коммуникационные порты.

Кодирование вариантов выбора источника для последовательной загрузки программы через периферийные устройства UART, Ethernet или USB отличается для процессорных ядер ПЦОС 1 и ПЦОС 2 (см. таблицу 5.31). Последовательная загрузка осуществляется с помощью прикладного программного обеспечения для ПК, работающего под управлением ОС Windows. Особенности работы этих программ описаны в соответствующей документации. Формат потока входных данных соответствует формату данных для загрузки из внешней памяти разрядностью 32 бита.

Таблица 5.31 − Выбор источника загрузки в зависимости от состояния выводов IIOF(3 – 0)_x#

Состояние внешних выводов				Источник загрузки
общие варианты выбора источника загрузки для ПЦОС x (где x 1 или 2)				
IIOF3_x#	IIOF2_x#	IIOF1_x#	IIOF0_x#	
1	1	0	1	Резерв
1	0	1	1	Резерв
1	0	0	1	Резерв
0	1	1	1	Загрузка с адреса 0x8000 0000
0	1	0	1	Загрузка с адреса 0xA000 0000
0	0	1	1	Загрузка с адреса 0xC000 0000
1	1	1	1	Через один из коммуникационных портов
0	0	0	1	Резерв (загрузчик останавливается)
0	0	0	0	Резерв
1	1	1	0	Резерв
варианты выбора источника загрузки ПЦОС 1 через периферийные устройства				
1	0	1	0	Загрузка из USB 2.0
0	1	1	0	Загрузка из Ethernet
0	0	1	0	Загрузка из UART
варианты выбора источника загрузки ПЦОС 2 через периферийные устройства				
1	1	0	0	Загрузка из USB 2.0
1	0	0	0	Загрузка из Ethernet
0	1	0	0	Загрузка из UART



Принятое условное обозначение: ком. порт – коммуникационный порт.

Рисунок 5.104, лист 1 − Выбор режима загрузчика
в зависимости от состояния выводов IIOF(3 − 0)_x#

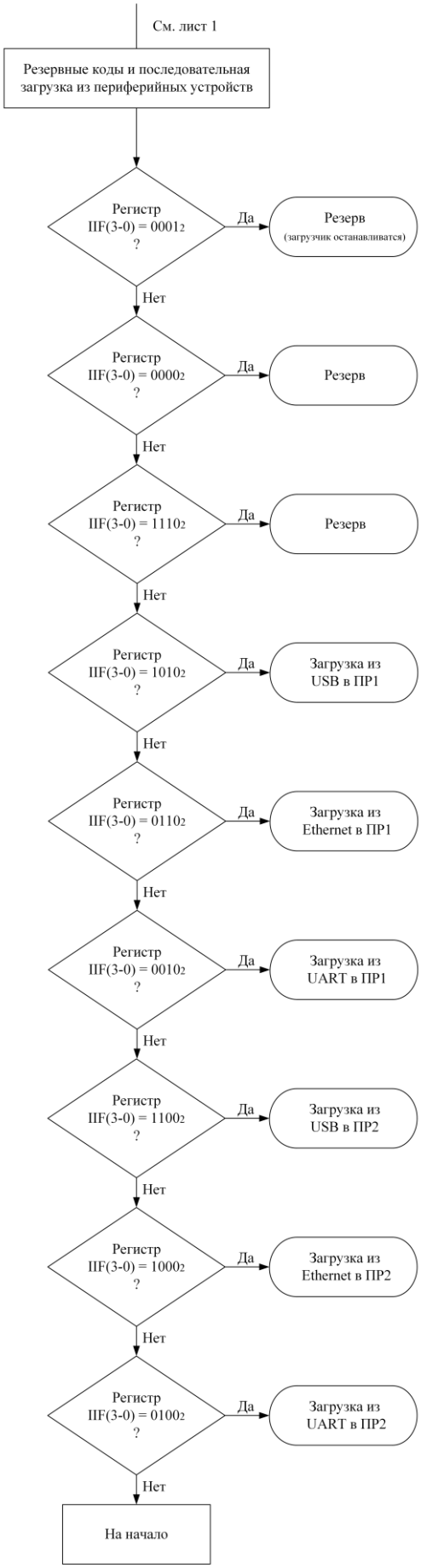


Рисунок 5.104, лист 2

5.10.3 Порядок работы загрузчика

Стандартная последовательность действий по инициализации загрузчика ядра ПЦОС, который закачивает исходную программу в оперативную память, выглядит следующим образом:

- установить выводы RESETLOC(1, 0)_x (x = 1, 2) в низкий уровень;
- установить вывод ROMEN_x в высокий уровень.

Примечание – На выводе ROMEN_x должен быть установлен высокий уровень в течение всего сеанса работы загрузчика. Состояние ROMEN_x может быть изменено в любое время после завершения сеанса загрузки;

- установить выводы IIOF3_1# – IIOF0_1#, IIOF3_2# – IIOF0_2# в соответствующее выбранному варианту загрузки состояние.

Установленный в высокий уровень вывод ROMEN_x разрешает доступ к внутреннему ПЗУ выбранного ПЦОС. Состояние внешних выводов IIOF3_1# − IIOF0_1#, IIOF3_2# − IIOF0_2#, указывающее на источник загрузки, программно опрашивается путём чтения соответствующих им флагов IIOF в регистре IIF. Варианты выбора источника в зависимости от комбинации значений этих флагов подробно рассматриваются в таблице 5.31 и на рисунке 5.104.

Когда расположение загрузочной таблицы определено, программа загружается по адресу, указанному в пятом слове таблицы (формат приведён в таблице 5.32), используя разрядность, задаваемую в первом слове (8, 16 или 32 бита). Загрузчик не может загрузить программу в первые 1000h адресов. Первые пять слов загрузочной таблицы определяют критерии загрузки и исполнения программы. Остальные слова – это, собственно, сама программа и указатели на таблицу векторов, как показано в таблице 5.32.

В качестве индикации завершения процесса загрузки используется инструкция IACK. Такая индикация может потребоваться для смены режима микрокомпьютера (ROMEN_x = 1) на режим микропроцессора (ROMEN_x = 0), и для разрешения использования многофункциональных выводов IIOF3_1# − IIOF0_1#, IIOF3_2# − IIOF0_2# в других целях.

Далее начинается исполнение загруженной программы (точкой входа является первое слово загруженной программы).

Поток данных с исходной программой должен иметь формат, представленный в таблице 5.32. Содержимое слов, начиная с четвёртого и до n варьируется для разных исходных программ, загружаемых по всему потоку данных. Первые три слова и последние три слова не влияют на блоки исходной программы. Восемь наименьших значащих битов (младшие биты) первого слова – разрядность внешней памяти. Если выбирается побайтовая загрузка или загрузка полусловами, то загрузка выполняется последовательно от младших битов к старшим.

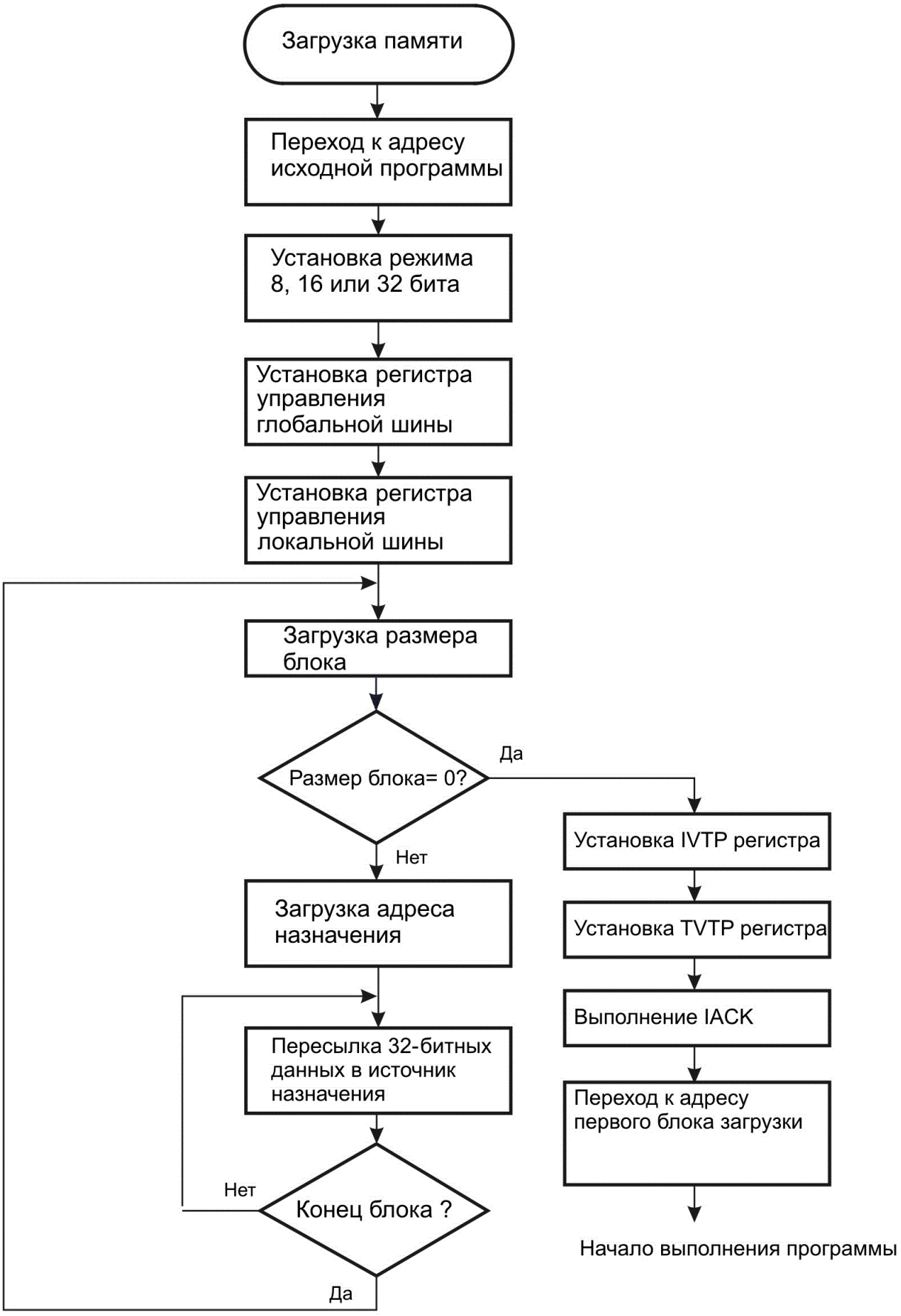
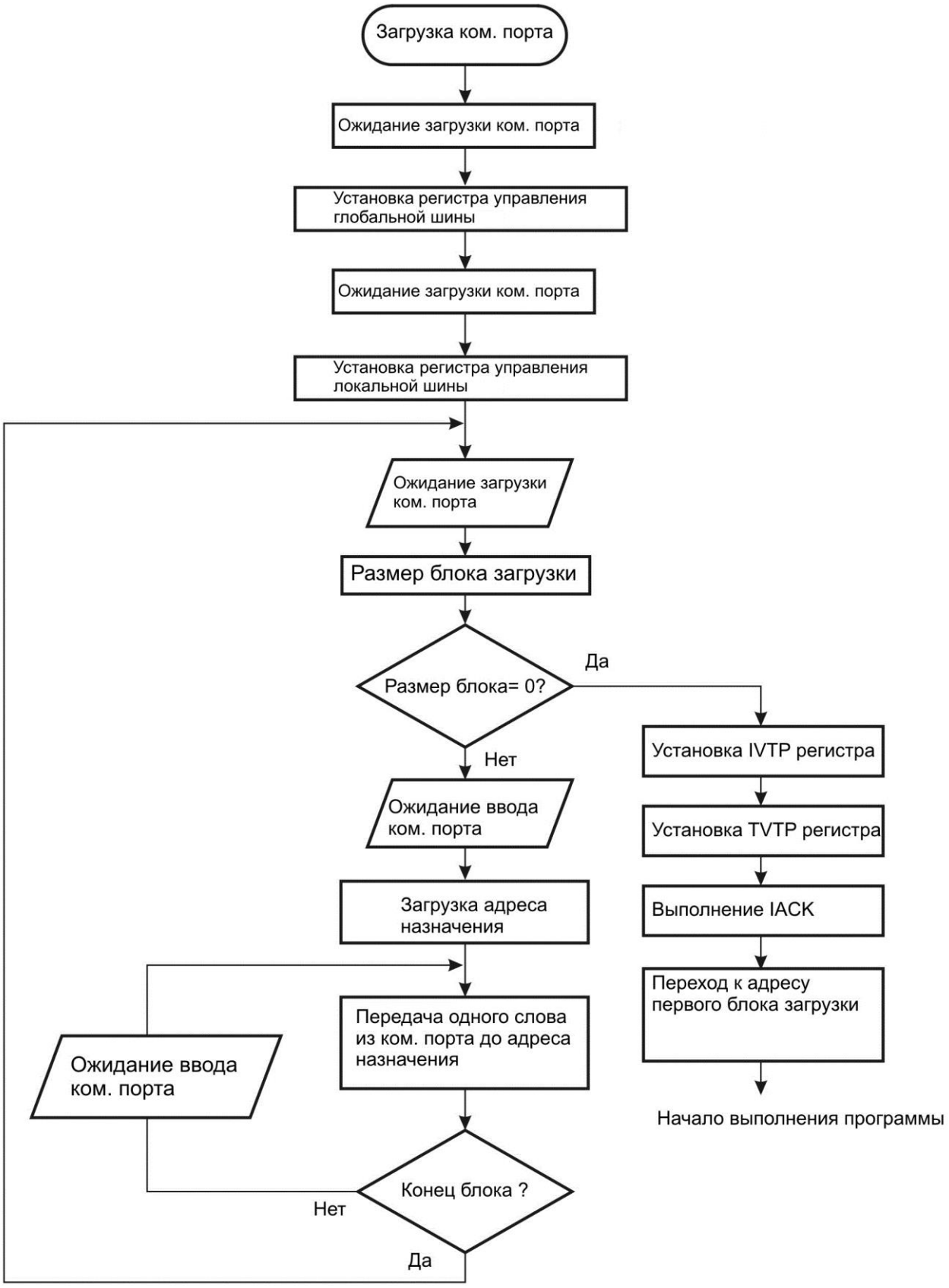


Рисунок 5.105 – Порядок установки режима работы ПЦОС
в процессе выполнения загрузки из внешней памяти



Принятое условное обозначение: ком. порт – коммуникационный порт.

Рисунок 5.106 – Порядок установки режима работы ПЦОС
в процессе выполнения загрузки через коммуникационные порты

Таблица 5.32 − Структура потока данных исходной программы

Номер слова	Описание обработки слова данных
1	Размерность внешней памяти, из которой будет производиться загрузка, 8, 16 или 32 разряда
2	Значение установки регистра управления глобальной шиной
3	Значение установки регистра управления локальной шиной
4	Размер блока 32-разрядных слов. Нулевой размер блока означает окончание загрузки данных
5	Адрес, с которого будет производиться загрузка программы
6	Первое слово загружаемой программы
n	Последнее слово загружаемой программы
n+1	Слово, состоящее из всех нулей. В случае, если было отправлено несколько блоков исходной программы, то слово, следующее за словом n, будет последним словом последнего блока исходной программы. Каждый блок исходной программы будет иметь формат, показанный в словах с четвёртого по n. Это слово, состоящее из всех нулей, будет следующим за последним блоком исходной программы
n+2	Значение регистра IVTP
n+3	Значение регистра TVTP
n+4	Ячейка памяти для инструкции IACK
Примечание – Заштрихованная область обозначает исходный программный блок.	

Каждая исходная программа при передаче нескольких программных блоков может быть загружена в различные места адресного пространства. В начале каждого программного блока задаётся его размер и адрес назначения. Завершается функция программной загрузки словом из всех нулей (00000000h).

В последних словах указываются адреса для размещения таблиц системных и программных прерываний. В последнем слове указывается адрес размещения инструкции IACK. Если этот адрес находится во внешней памяти, доступной системе, исполнение инструкции IACK приводит к появлению низкого уровня на одноимённом выводе (IACK#) индицируя завершение работы загрузчика и начало исполнения процессором первого блока кода.

5.10.4 Пример загрузки ПЦОС из внешней памяти

Если во время системного сброса ROMEN_x = 1, RESETLOC(1, 0)_x = 00_2, загрузчик может закачать программный код, расположенный во внешней 8-, 16- или 32-разрядной памяти. Начальный адрес загружаемого кода определяется состоянием входов IIOF(0–3)_x#. При разработке пользовательской прошивки для внутрикристального масочного ПЗУ нельзя размещать в нулевом адресе вектор системного сброса ПЦОС, поскольку он зарезервирован для загрузчика.

Восемь младших бит первого слова данных определяют разрядность внешней памяти (8, 16 или 32 бита) в соответствии с таблицами 5.33, 5.34 и 5.35:

- 8-разрядная память: 08h;
- 16-разрядная память: 0010h;
- 32-разрядная память: 00000020h.

Если используется 8- или 16-разрядная память, загрузка выполняется в последовательности от младших битов к старшим. При загрузке 16-разрядных слов сначала происходит загрузка младшего слова, затем старшего и далее эти 16-разрядные слова объединяются, образуя полное 32-разрядное слово. Аналогично осуществляется и побайтовое чтение. Чтение 32-разрядных слов выполняется параллельно.

При использовании 16-разрядной внешней памяти она должна быть подключена к выводам (L)D15 − 0, для 8-разрядной внешней памяти – к выводам (L)D7 − 0. Все, не задействованные выводы шин данных, следует подключить к питанию через отдельные резисторы номиналом 22 кОм.

Таблицы 5.33, 5.34 и 5.35 показывают примеры использования 8-, 16- и 32-разрядной внешней памяти.

В данных примерах используются следующие установки:

- После системного сброса выводы IIOF(3−0)_x# имеют состояние 0111_2.
- Внешняя память находится по адресу 80000000h и имеет следующие параметры:
  - память имеет разрядность: 8, 16 и 32 бита.
- Память глобальной шины имеет одно программное состояние ожидания внешнего сигнала RDY (SWW = 11), размер страницы равен 64K слов для обоих стробов (STRB0# и STRB1#) и размер банка памяти в 1 Гбайт на каждый строб.
- Память локальной шины имеет два программных состояния ожидания (SWW = 01), размер страницы равен 32K слов для обоих стробов и размер банка памяти в 1 Гбайт на каждый строб.
- Первый блок программы содержит 294 слова и размещается по адресу 002FF840h.
- Второй блок программы содержит 64 слова и размещается по адресу 002FF800h.
- Указатели страниц регистров IVTP и TVTP находятся в начале ОЗУ ПЦОС.
- Команда IACK находится по адресу 80000000h.

Таблица 5.33 – Пример загрузки ПЦОС из 8-разрядной памяти

Номер слова	Адрес	Значение данных	Комментарии
1	80000000h	08h	Разрядность внешней памяти равна 8 битам
	80000001h	00h	
	80000002h	00h	
	80000003h	00h	
2	80000004h	F0h	Значение регистра управления глобальной памятью равно 1D7BC9F0h
	80000005h	C9h	
	80000006h	7Bh	
	80000007h	1Dh	
3	80000008h	50h	Значение регистра управления локальной памятью равно 1D73 9250h
	80000009h	92h	
	8000000Ah	73h	
	8000000Bh	1Dh	
4	8000000Ch	26h	Размерность первого блока программы равна 126h
	8000000Dh	01h	
	8000000Eh	00h	
	8000000Fh	00h	
5	80000010h	40h	Стартовый адрес первого блока программы расположен по адресу 002FF840h
	80000011h	F8h	
	80000012h	2Fh	
	80000013h	00h	
6 по 299	80000014h • • 800004ABh		Начало первого блока программного кода • • Окончание первого блока программного кода
300	800004ACh	40h	Размерность второго блока программы равна 40h
	800004ADh	00h	
	800004AEh	00h	
	800004AFh	00h	
301	800004B0h	00h	Стартовый адрес второго блока программы расположен по адресу 002FF800h
	800004B1h	F8h	
	800004B2h	2Fh	
	800004B3h	00h	
302 по 365	800004B4h • • 800005B3h		Начало второго блока программного кода • • Окончание второго блока программного кода
366	800005B4h	00h	Нулевое слово (00000000h), указывающее на конец загрузки блоков программы
	800005B5h	00h	
	800005B6h	00h	
	800005B7h	00h	
367	800005B8h	00h	Значение регистра IVTP равно 002FF800h
	800005B9h	F8h	
	800005Bah	2Fh	
	800005BBh	00h	
368	800005BCh	00h	Значение регистра TVTP равно 002FF800h
	800005BDh	F8h	
	800005BEh	2Fh	
	800005BFh	00h	
369	800005C0h	00h	Команда IACK находится по адресу 80000000h (и это будет последним словом в загрузке, далее ПЦОС переходит к выполнению загруженной программы)
	800005C1h	00h	
	800005C2h	30h	
	800005C3h	00h	

Таблица 5.34 – Пример загрузки ПЦОС из 16-разрядной памяти

Номер слова	Адрес	Значение данных	Комментарии
1	80000000h	0010h	Разрядность внешней памяти равна 16 битам
	80000001h	0000h	
2	80000002h	C9F0h	Значение регистра управления глобальной памятью равно 1D7BC9F0h
	80000003h	1D7Bh	
3	80000004h	9250h	Значение регистра управления локальной памятью равно 1D739250h
	80000005h	1D73h	
4	80000006h	0126h	Размерность первого блока программы равна 126h
	80000007h	0000h	
5	80000008h	F840h	Стартовый адрес первого блока программы расположен по адресу 002FF840h
	80000009h	002Fh	
6 по 299	8000000Ah • • 80000255h		Начало первого блока программного кода • • Окончание первого блока программного кода
300	80000256h	0040h	Размерность второго блока программы равна 40h
	80000257h	0000h	
301	80000258h	F800h	Стартовый адрес второго блока программы расположен по адресу 002FF800h
	80000259h	002Fh	
302 по 365	8000025Ah • • 800002D9h		Начало второго блока программного кода • • Окончание второго блока программного кода
366	800002Dah	0000h	Нулевое слово (00000000h), указывающее на конец загрузки блоков программы
	800002DBh	0000h	
367	800002DCh	F800h	Значение регистра IVTP равно 002FF800h
	800002DDh	002Fh	
368	800002DEh	F800h	Значение регистра TVTP равно 002FF800h
	800002DFh	002Fh	
369	800002E0h	0000h	Команда IACK находится по адресу 80000000h (и это будет последним словом в загрузке, далее ПЦОС переходит к выполнению загруженной программы)
	800002E1h	8000h	

Таблица 5.35 − Пример загрузки ПЦОС из 32-разрядной памяти

Номер слова	Адрес	Значение данных	Комментарии
1	80000000h	00000020h	Разрядность внешней памяти равна 32 битам
2	80000001h	1D7BC9F0h	Значение регистра управления глобальной памятью равно 1D7BC9F0h
3	80000002h	1D739250h	Значение регистра управления локальной памятью равно 1D739250h
4	80000003h	00000126h	Размерность первого блока программы равна 126h
5	80000004h	002FF840h	Стартовый адрес первого блока программы расположен по адресу 002FF840h
6 по 299	80000005h • 8000012Ah		Начало первого блока программного кода • Окончание первого блока программного кода
300	8000012Bh	00000040h	Размерность второго блока программы равна 40h
301	8000012Ch	002FF800h	Стартовый адрес второго блока программы расположен по адресу 002FF800h
302 по 365	8000012Dh • 8000016Ch		Начало второго блока программного кода • Окончание второго блока программного кода
366	8000016Dh	00000000h	Нулевое слово (00000000h), указывающее на конец загрузки блоков программы
367	8000016Eh	002FF800h	Значение регистра IVTP равно 002FF800h
368	8000016Fh	002FF800h	Значение регистра TVTP равно 002FF800h
369	80000170h	80000000h	Команда IACK находится по адресу 80000000h, (и это будет последним словом в загрузке, далее ПЦОС переходит к выполнению загруженной программы)

5.10.5 Пример загрузки ПЦОС через коммуникационные порты

Если во время системного сброса на выводах ROMEN_x и IIOF(3 − 0)_x# установлен высокий уровень, загрузчик переходит к варианту закачки кода через коммуникационные порты. ПЦОС осуществляет опрос коммуникационных портов, начиная с нулевого и заканчивая пятым; как только на одном из портов обнаружены входные данные, стартует загрузка кода. Она выполняется аналогично загрузке из внешней памяти, за исключением этапа выбора разрядности внешней памяти, так как к коммуникационному порту возможно подключение только восьмиразрядной памяти. В примере 5.65 представлен листинг программы для загрузки мультипроцессорной системы через коммуникационные порты.

Пример 5.65 – Листинг программы для загрузки мультипроцессорной системы через коммуникационные порты.

```
*_____________________________________________________
* MASTER PROCESSOR BOOT TABLE
*_____________________________________________________
.text
.word       32                  ; memory width
.word       3003c000h           ; MASTER global control register
                                ; (system specific !!)
.word       3d79c210h           ; master local control register
                                ; (system specific !!)
*_____________________________________________________
```

```
*_____________________________________________________
* MASTER PROCESSOR PROGRAM BLOCK
*_____________________________________________________
.word       10                      ; block size
.word       2ff800h                 ; block dest addr
* Code for master processor: this code sends boot table to slave processor
ldi         8, rc                   ; loop 9 times: size of slave processor
; boot table
rptbd       endb1
ldp         src                     ; src in external memory
ldi         @src, ar0
ldi         @dst, ar1
ldi         *ar0 + + (1), r0        ; block start
endb1:  sti r0, *ar1
bu          $                       ; master processor loops forever
src         .word   BOOT_TABLE2     ; address of boot table of slave
; processor
dst         .word   100042h         ; address of OFIFO connected to slave
; processor
*_____________________________________________________
* END OF ALL BLOCKS
*_____________________________________________________
.word       0                       ; master end of bootload sequence
.word       2ffd00h                 ; master IVTP value
.word       2ffd00h                 ; master TVTP value
.word       40000000h               ; master address for iack
*_____________________________________________________
* END OF MASTER PROCESSOR BOOT TABLE : size = 9 words
*_____________________________________________________
*_____________________________________________________
* SLAVE PROCESSOR BOOT TABLE
*_____________________________________________________
BOOT_TABLE2:                        ; slave BOOT TABLE
.word       3003c000h               ; slave global control register
                                    ; (system specific !!!)
.word       3d79c210h               ; slave local control register
                                    ; (system specific !!!)
.word       1                       ; block size
.word       2ff800h                 ; dst load address
bu          $                       ; slave processor loops forever
.word       0                       ; slave end of bootload sequence
.word       2ffd00h                 ; slave IVTP value
.word       2ffd00h                 ; slave TVTP value
.word       40000000h               ; slave address for iack
*_____________________________________________________
* END OF EPROM CODE
*_____________________________________________________
```

5.10.6 Изменение состояния выводов IIOF(3–0)_x# после завершения работы загрузчика

Так как во время работы загрузчика необходимо, чтобы значение выводов IIOF(3–0)_x# (где x = 1, 2) было постоянно задано в нужной комбинации для требуемого режима загрузки, то для дальнейшего использования этих выводов (после завершения работы загрузчика), необходимо использовать внешние схемотехнические решения. На рисунке 5.107 представлена схема, которая выставляет вывод IIOF(3–0)_x# в низкий уровень и удерживает его до тех пор, пока не придёт сигнал подтверждения прерывания IACK#, далее вывод IIOF(3–0)_x# доступен для использования внешними источниками прерываний.

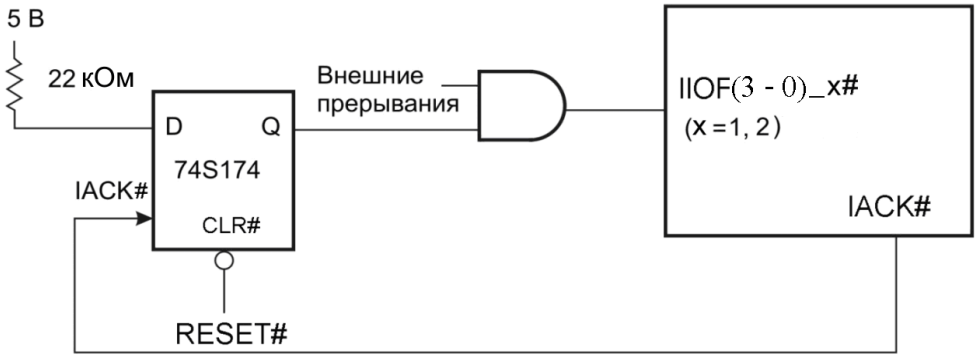


Рисунок 5.107 − Схема управления выводами IIOF(3–0)_x#

5.10.7 Исполняемый код (листинг программы) загрузчика

```
*******************************************************************************
* Program Name
* boot_v5.asm
*******************************************************************************
* Program Intent
* Осуществляет загрузку программы из внешней памяти, коммуникационного порта,
* устройства UART, устройства Ethernet или устройства USB 2.0
*******************************************************************************
* Program Description
* Подробное описание программы приведено в подразделе Загрузчик
*******************************************************************************
* Program Notes
* Кодировка функционального назначения выводов IIOF(0–3)_x#:
*
*     IIOF3     IIOF2     IIOF1     IIOF0     Функция
*   ---------------------------------------------------------
*     1         1         0         1         Резерв
*     1         0         1         1         Резерв
*     1         0         0         1         Резерв
*     0         1         1         1         Загрузка из памяти 80000000h
*     0         1         0         1         Загрузка из памяти A0000000h
*     0         0         1         1         Загрузка из памяти C0000000h
*     0         0         0         1         Останов загрузчика
*     1         1         1         1         Загрузка из коммуникационного порта
*     0         0         1         0         Загрузка ПЦОС 1 из UART
*     0         1         0         0         Загрузка ПЦОС 2 из UART
*     0         1         1         0         Загрузка ПЦОС 1 по Ethernet
*     1         0         0         0         Загрузка ПЦОС 2 по Ethernet
*     1         0         1         0         Загрузка ПЦОС 1 по USB 2.0
*     1         1         0         0         Загрузка ПЦОС 2 по USB 2.0
```

```
*******************************************************************************
* Program Results
* Управление передаётся на начальный адрес первого принятого блока
*******************************************************************************
* version 1.51
* 25.03.2013
* review 07.06.2013
* - Изменена таблица векторов прерываний
* - Добавлена инициализация регистра управления локальной памятью
*******************************************************************************
        .asg AR0, w_address
        .asg AR1, temp2
        .asg AR2, data
        .asg AR3, status
        .asg AR4, address2
        .asg AR5, address
        .asg AR6, counter
        .asg AR7, temp
        .asg R2, data2
        .asg R11, pointer
        .asg R9, pgm_start
        .asg R8, flag
        .asg R5, crc
        .asg R6, last_byte
*******************************************************************************
* Модуль USB 2.0, смещение относительно 60 0000h
*******************************************************************************
CSR                     .set    00h ; Регистр управления и состояния
FA                      .set    04h ; Адрес функции
INT_MSK                 .set    08h ; Маска прерывания для endpoints независимо от источника
INT_SRC                 .set    0Ch ; Регистр источника прерывания
FRM_NAT                 .set    010h ; Число, фрейм и время
REV                     .set    014h ; RTL Revision
GPOUT                   .set    018h ; General Purpose Outputs
GPIN                    .set    01Ch ; General Purpose Inputs
RSUSPSSM                .set    020h ; SuspendM/Resume
EP0_CSR                 .set    040h ; endpoint CSR
EP0_INT                 .set    044h ; регистр прерываний
EP0_OBA                 .set    048h ; регистр указателя буфера 0
EP0_IBA                 .set    04Ch ; регистр указателя буфера 1
EP1_CSR                 .set    050h ; endpoint CSR
EP1_INT                 .set    054h ; регистр прерываний
EP1_OBA                 .set    058h ; регистр указателя буфера 0
EP1_IBA                 .set    05Ch ; регистр указателя буфера 1
EP2_CSR                 .set    060h ; endpoint CSR
EP2_INT                 .set    064h ; регистр прерываний
EP2_OBA                 .set    068h ; регистр указателя буфера 0
EP2_IBA                 .set    06Ch ; регистр указателя буфера 1
```

```
; Маски
SetBit0                 .set 1h
SetBit1                 .set 2h
SetBit2                 .set 4h
SetBit3                 .set 8h
SetBit4                 .set 10h
SetBit5                 .set 20h
SetBit6                 .set 40h
SetBit7                 .set 80h
SetBit8                 .set 100h
SetBit9                 .set 200h
SetBit10                .set 400h
SetBit11                .set 800h
SetBit1                 .set 1000h
SetBit13                .set 2000h
SetBit14                .set 4000h
SetBit15                .set 8000h

        .sect "boot"
BOOT:
        LDI             COM_LOAD, R10    ; адрес подпрограммы загрузки из ком. порта → R10
        LDHI  0010h, AR0
; адрес 0010 0000 → AR0 (AR0 хранит начальный адрес периферийной карты памяти)
;-------------------------------------------------------------
; подпрограмма проверки входов IIOF3-1
;-------------------------------------------------------------
CHECK:
        LDHI  0030h, AR1
        CMPI  04404h, IIF
        BEQ             MEMORY

        LDHI  04000h, AR1
        CMPI  04044h, IIF
        BEQ             MEMORY

        LDHI  06000h, AR1
        CMPI  04004h, IIF
        BEQ             MEMORY

        LDHI  08000h, AR1               ; адрес памяти 8000 0000 → AR1
        CMPI  00444h, IIF               ; если IIOF3-1 == 011, то
        BEQ             MEMORY     ; выполнить загрузку из памяти

        LDHI  0A000h, AR1               ; адрес памяти A000 0000 → AR1
        CMPI  00404h, IIF               ; если IIOF3-1 == 010, то
        BEQ             MEMORY     ; выполнить загрузку из памяти

        LDHI  0C000h, AR1               ; адрес памяти C000 0000 → AR1
        CMPI  00044h, IIF               ; если IIOF3-1 == 001, то
        BEQ             MEMORY          ; выполнить загрузку из памяти
```

```
        CMPI    4444h, IIF          ; если IIOF3-1 == 111, то
        BEQ     BOOT_COM            ; загрузка из COM порта
        B       BOOT_OTHER          ; иначе загрузка с помощью других устройств
        BNE     BOOT_OTHER
;-------------------------------------------------------------
; загрузка из коммуникационных портов
;-------------------------------------------------------------
; AR0 – адрес 0010 0000 (начальный адрес периферийной карты памяти)
BOOT_COM:
        ADDI  040h, AR0, AR3
; AR0+40h → AR3 (адрес 0010 0040 → R3, R3 хранит адрес регистра управления ком. порта 0)
        LDI   5, AR1                ; установка счётчика цикла CHECK_CH
CHECK_CH:
        LSH3  –9, *AR3, R1          ; R1 ← *AR3>>9
        BNZ   LOAD0
; если входные данные есть, запускается загрузчик коммуникационного порта
        ADDI  010h, AR3
; R3 указывает на регистры управления следующего коммуникационного порта
        DBU   AR1,  CHECK_CH  ; AR1 ← AR1 – 1 затем CHECK_CH → PC
        B               CHECK       ; перепроверка входных флагов
;-------------------------------------------------------------
; Загрузчик данных из памяти
;-------------------------------------------------------------
; AR1 – адрес откуда будет производиться загрузка
; проверка размерности памяти
MEMORY:
        LDI             *AR1 + + (1), R1
; загружаем размерность внешней памяти, из которой будет производиться загрузка в R1
        LDI             W_WIDE, R10     ; адрес подпрограммы W_WIDE → R10
        LSH             26, R1          ; проверяем бит 5
        BN              LOAD0
; если '1', то начинаем загрузку программы (по 32 разряда)

        NOP             *AR1 + + (1)    ; указываем на следующую половину слова
        LDI             H_WIDE, R10     ; адрес подпрограммы H_WIDE → R10
        LSH             1, R1
        BN              LOAD0

        NOP             *AR1 + + (1)    ; указываем на следующий байт
        LDI             B_WIDE, R10     ; адрес подпрограммы B_WIDE → R10
        NOP             *AR1 + + (1)    ; указываем на следующий байт
; начало загрузки программы
LOAD0:
        LAJU            R10
; загружаем новое слово в соответствии с размером памяти
        LDHI            0010h, AR0
        LDI             1, R0           ; выключаем флаг стартового адреса
        NOP
        LAJU            R10
; загружаем новое слово в соответствии с разрядностью памяти
        STI             AR2, *AR0
```

```
; устанавливаем регистр управления глобальной шиной
        NOP
        NOP
        STI             AR2, * + AR0(4)
; устанавливаем регистр управления локальной шиной
LOAD2:
        LAJU  R10                   ; загружаем новое слово в соответствии с разрядностью памяти
        ADDI  1, R0                 ; выключаем флаг стартового адреса
        NOP
        NOP
        CMPI  0, R2                 ; если размер блока 0, то переходим
        BEQ             IVTP_LOAD   ; к настройке векторов прерываний
        LAJU  R10                   ; загружаем новое слово в соответствии с разрядностью памяти
        SUBI3 1, AR2, RC            ; устанавливаем размер блока для повтора в цикле
        NOP
        SUBI  1, R10
        LDI             R0, R0      ; проверяем флаг стартового адреса
        LDIP  AR2, R9               ; загружаем стартовый адрес если флаг выключен
        LAJU  R10                   ; загружаем новое слово в соответствии с разрядностью памяти
        LDI             AR2, AR0    ; установка адреса загрузки
        LDI             –1, R0      ; включаем флаг стартового адреса и флаг адреса загрузки
        ADDI  1, R10
        B LOAD2                     ; загружаем новый блок по окончании цикла
*
* инициализация регистров IVTP и TVTP
*
IVTP_LOAD:
        LAJU  R10                   ; загружаем новое слово в соответствии с разрядностью памяти
        NOP
        NOP
        NOP
TVTP_LOAD:
        LAJU  R10                   ; загружаем новое слово в соответствии с разрядностью памяти
        LDPE  AR2, IVTP             ; устанавливаем указатель на IVTP
        NOP
        NOP
        LAJU  R10                   ; загружаем новое слово в соответствии с разрядностью памяти
        LDPE  AR2, TVTP             ; устанавливаем указатель на TVTP
        NOP
        NOP
        IACK  *AR2                  ; вывод сигнала IACK
        BU    R9                    ; переход к выполнению загруженной программы
;————————————————————————————————————————————————————————————;
; Подпрограмма загрузки 8 бит
;————————————————————————————————————————————————————————————;
LOOP_B:
        RPTB LOAD_B                 ; Цикл загрузки программы
B_WIDE:
        LWL0 *AR1 + + (1), AR2      ; загрузка нулевого байта
        NOP                         ; задержка для установки 1 на STRB
        LWL1 *AR1 + + (1), AR2      ; объединяем байт 1 с байтом 0
        NOP                         ; задержка для установки 1 на STRB
        LWL2 *AR1 + + (1), AR2      ; объединяем байт 2 с байтами 0 и 1
```

```
        NOP                             ; задержка для установки 1 на STRB
        LWL3 *AR1 + + (1), AR2          ; объединяем байт 3 с байтами 0, 1 и 2
        LDI     R0, R0                  ; проверяем флаг загрузки адреса
        BNN     B_END
LOAD_B:
        STI     AR2, *AR0 + + (1)       ; сохраняем новое слово по адресу назначения
B_END:
        BU R11                          ; возвращаемся из подпрограммы
;-------------------------------------------------------------------;
; Подпрограмма загрузки 16 бит;
;-------------------------------------------------------------------;
LOOP_H:         RPTB LOAD_H             ; Цикл загрузки программы
H_WIDE:
        LWL0    *AR1 + + (1), AR2       ; загрузка младших 16 бит слова
        NOP                             ; задержка для установки 1 на STRB
        LWL2    *AR1 + + (1), AR2       ; объединяем старшие и младшие 16 бит слова
        LDI     R0, R0                  ; проверяем флаг загрузки адреса
        BNN     H_END
LOAD_H:
        STI     AR2, *AR0 + + (1)       ; сохраняем новое слово по адресу назначения
H_END:  BU R11                          ; возвращаемся из подпрограммы
;-------------------------------------------------------------------;
; Подпрограмма загрузки 32 бит;
;-------------------------------------------------------------------;
LOOP_W:         RPTB LOAD_W             ; Цикл загрузки программы
W_WIDE:
        LDI     *AR1 + + (1), AR2       ; читаем новое 32-разрядное слово
        LDI     R0, R0                  ; проверяем флаг загрузки адреса
        BNN     W_END
LOAD_W:
        STI     AR2, *AR0 + + (1)       ; сохраняем новое слово по адресу назначения
W_END:  BU R11                          ; возвращаемся из подпрограммы
;-------------------------------------------------------------------;
; Подпрограмма загрузки из коммуникационного порта;
;-------------------------------------------------------------------;
LOOP_C:         RPTB LOAD_C             ; Цикл загрузки программы
COM_LOAD:
        LSH3    –9, *AR3, R1            ; проверяем вход ком. порта
        BZ      COM_LOAD                ; ожидаем вход ком. порта
        LDI     * + AR3(1), AR2         ; читаем новое 32-разрядное слово
        LDI     R0, R0                  ; проверяем флаг загрузки адреса
        BNN     C_END
LOAD_C:
        STI AR2, *AR0++(1)              ; сохраняем новое слово по адресу назначения
C_END:  BU R11                          ; возвращаемся из подпрограммы
;-------------------------------------------------------------------;
; Дополнительные опции загрузки
;-------------------------------------------------------------------;
BOOT_OTHER:
; инициализируем управление локальной шиной
        ldp localcontol
        ldi @localcontol, R6
```

```
        ldhi 10h, AR4
        sti R6, * + AR4(4)

        CMPI  0040h, IIF        ; если IIOF3-0 = 0010, то
        BEQ         INITUART1   ; загрузка с UART в ПЦОС 1
        CMPI  0400h, IIF        ; если IIOF3-0 = 0100, то
        BEQ         INITUART2   ; загрузка с UART в ПЦОС 2

   LDI 1, R4                    ; Номер ПЦОС = 1
   CMPI 00440H, IIF             ; Если IIOF3-0 = 0110
        BEQ Ethernet            ; Выполнение Ethernet bootloader
   LDI 2, R4                    ; Номер ПЦОС = 2
   CMPI 04000H, IIF             ; Если IIOF3-0 = 1000
        BEQ Ethernet            ; Выполнение Ethernet bootloader

   LDI 1, R4                    ; Номер ПЦОС = 1
   CMPI 04040H, IIF             ; Если IIOF3-0 = 1010
        BEQ USB20               ; Выполнение USB bootloader
   LDI 2, R4                    ; Номер ПЦОС = 2
   CMPI 04400H, IIF             ; Если IIOF3-0 = 1100
        BEQ USB20               ; Выполнение USB bootloader

        CMPI 00004H, IIF        ; если IIOF3-0 = 000
        BEQ Terminate           ; остановить Bootloader
        B           CHECK
*-------------------------------------------------------------------------------;
* TERMINATE BOOTLOADER
*-------------------------------------------------------------------------------;
Terminate: BU Terminate
; указатели на подпрограммы загрузчика
A_CMP20         .word CMP20
A_SETGM         .word SETGM
A_SETLM         .word SETLM
A_CMP0          .word CMP0
A_SETADDRESS    .word SETADDRESS
A_WRITE         .word WRITE
A_SETIVTP       .word SETIVTP
A_SETTVTP       .word SETTVTP
A_SETIACK       .word SETIACK
;-------------------------------------------------------------------------
; Настройка параметров UART
;-------------------------------------------------------------------------
INITUART1:                      ; загрузка с UART в ПЦОС 1
        ldi         01h, data   ; установка в 01 битов 05 – 04 регистра RC
        B           INITUART3   ; для коммутации UART с ПЦОС 1
INITUART2:                      ; загрузка с UART в ПЦОС 2
        ldi         02h, data
; установка в 10₂ бит 05 – 04 для коммутации UART с ПЦОС 2
INITUART3:
        BU          SetFlags
```

```
; установка начальных значений флага загрузки первого блока
SetDP:                                  ; и контрольной суммы
        ldp     A_CMP20                 ; устанавливаем регистр DP
        ldhi    030h, address2          ; загружаем старшие 16 бит адреса
        addi    1, address2             ; address2 теперь содержит адрес RC
        ldhi    030h, address           ; загрузка адреса RS в address
WaitRS:
        ldii    *address, R0
        cmpi    0, R0                   ; проверяем RS
        bnz     WaitRS                  ; переход на метку WaitRS, если RS занят
        sti     data, *address2
; модифицируем RC новым значением для коммутации UART с ПЦОС
        ldhi    0040h, address          ; загружаем старшие 16 бит адреса
        ldi     * + address2 (3), data  ; считываем регистр LCR
        or      0080h, data             ; устанавливаем 7 бит в 1
        sti     data, * + address (3)   ; сохраняем LCR

        ldi     0, data
        sti     data, *+address (1)
        ldi     0078h, data
; устанавливаем параметры делителя для частоты 9600 бод/с
        sti     data, *address2
; записываем 120 в младший байт регистра делителя LSB
        ldi     0, data
        sti     data, *+address (1)
        ldi     001Bh, data             ; отключаем доступ к делителю и настраиваем UART
        sti     data, *+ address (3)    ; чётное число 1, 8 бит данных, 1 стоповый бит
        ldi     0006h, data
        sti     data, *+ address (2)    ; очистим FIFO передатчика и приёмника
;-------------------------------------------------------------------------
; Ожидание признака начала загрузки программы
;-------------------------------------------------------------------------
Wait20:
        ldi     @A_CMP20, pointer ; сохраняем адрес возврата
        B       LOADWORD          ; считываем новое слово из UART
CMP20:
        CMPI    020h, data
        BEQ     SETRGM
; если получено 020h, то переходим на метку SETRGM
        B       Wait20                  ; ждём, пока не получено слово 020h
;-------------------------------------------------------------------------
; Установка регистра управления глобальной памятью
;-------------------------------------------------------------------------
SETRGM:
        ldhi    0010h, address2         ; начальный адрес периферийной карты памяти
        ldi     @A_SETGM, pointer; сохраняем адрес возврата
        B       LOADWORD          ; считываем новое слово из UART
SETGM:                                  ; устанавливаем регистр управления
        sti     data, *address2         ; глобальной памятью
```

```
;-------------------------------------------------------------------------
; Установка регистра управления локальной памятью
;-------------------------------------------------------------------------
        ldi             @A_SETLM, pointer ; сохраняем адрес возврата
        B               LOADWORD          ; считываем новое слово из UART
SETLM:                                    ; устанавливаем регистр управления
        sti             data, *+address2(4) ; локальной памятью
;-------------------------------------------------------------------------
; Считываем количество слов в переменную счётчик
;-------------------------------------------------------------------------
STARTREAD:
        ldi             @A_CMP0, pointer  ; сохраняем адрес возврата
        B               LOADWORD          ; считываем новое слово из UART
CMP0:
        cmpi            0, data           ; если 0, то переходим к установке регистра
        beq             LOADIVTP          ; IVTP

        subi3   1, data, counter          ; считываем размер программы

;-------------------------------------------------------------------------
; Считываем адрес загрузки
;-------------------------------------------------------------------------
        ldi             @A_SETADDRESS, pointer ; сохраняем адрес возврата
        B               LOADWORD               ; считываем новое слово из UART
SETADDRESS:
        ldi             data, w_address        ; считываем адрес загрузки
;-------------------------------------------------------------------------
; Запоминаем адрес первой подпрограммы
;-------------------------------------------------------------------------
        cmpi            0, flag           ; если загружается первая программа
        bne             READ
        ldi             w_address, pgm_start ; запоминаем адрес загрузки первой программы
        ldi             1, flag           ; изменяем флаг загрузки первого блока программы
;-------------------------------------------------------------------------
; Считываем программу в цикле
;-------------------------------------------------------------------------
READ:
        ldi             @A_WRITE, pointer ; сохраняем адрес возврата
        B               LOADWORD          ; считываем новое слово из UART
WRITE:
        sti             data, *w_address++(1)  ; запись по адресу и инкремент адреса
        dbu             counter, READ
        B               STARTREAD         ; переходим к загрузке нового блока
;-------------------------------------------------------------------------
; Считываем значение регистра IVTP
;-------------------------------------------------------------------------
LOADIVTP:
        ldi             @A_SETIVTP, pointer    ; сохраняем адрес возврата
        B               LOADWORD               ; считываем новое слово из UART
SETIVTP:
        ldpe            data, IVTP             ; установка IVTP
```

```
;-------------------------------------------------------------------------
; Считываем значение регистра TVTP
;-------------------------------------------------------------------------
LOADTVTP:
        ldi     @A_SETTVTP, pointer     ; сохраняем адрес возврата
        B       LOADWORD                ; считываем новое слово из UART
SETTVTP:
        ldpe    data, TVTP              ; установка TVTP
;-------------------------------------------------------------------------
; Вывод сигнала IACK
;-------------------------------------------------------------------------
        ldi     @A_SETIACK, pointer     ; сохраняем адрес возврата
        B       LOADWORD                ; считываем новое слово из UART
SETIACK:
        BU      SendSumm    ; вывод сигнала IACK и отправки контрольной суммы
UnlockRC:
;-------------------------------------------------------------------------
; Разблокировка регистра коммутации
;-------------------------------------------------------------------------
        ldhi    030h, address2          ; загружаем старшие 16 бит адреса
        addi    1, address2             ; address2 теперь содержит адрес RC
        stii    data, *address2         ; разблокировка регистра коммутации
;-------------------------------------------------------------------------
; Выполнение программы
;-------------------------------------------------------------------------
        B       pgm_start               ; Выполнение программы
;-------------------------------------------------------------------------
; Подпрограмма загрузки слова с UART
;-------------------------------------------------------------------------
LOADWORD:
        ldhi    0040h, address          ; загружаем старшие 16 бит адреса
        ldi     0, data                 ; обнуляем регистр AR2
        ldi     3, temp                 ; переменная цикла
WaitByte:
        lsh     –8, data                ; сдвиг на два байта вправо
TestFlag:
        ldi     *+address(5), status
        tstb    1, status               ; проверяем бит 0 (Data Ready) регистра LSR
        bz      TestFlag                ; если 0, то продолжаем ждать приём
        lwl3    *address, data          ; сохраняем бит данных
        lbu3    data, last_byte ; сохраняем принятый бит для подсчёта контрольной суммы
        BU      test_parity
RNextB:
        dbu     temp, WaitByte          ; повторяем в цикле
        B       pointer                 ; переход по адресу в pointer
```

```
;-------------------------------------------------------------------------;
; Подпрограмма Ethernet BOOTLOADER
;-------------------------------------------------------------------------;
Ethernet:
        ldhi 2Fh, SP
        or 0FFF0h, SP                   ; Stack Pointer
; захват регистра коммутации RC
* AR7 – адрес регистра коммутации RC
* AR6 – адрес регистра семафора RS
GetRC:
; устанавливаем LLOCK в низкий уровень и читаем RCC в R0
   ; загрузка адреса RC в AR7
        ldhi 30h, AR7 ;         загружаем старшие 16 бит
        or 1, AR7 ;             AR7 = 30 0001h
        ; загрузка адреса RS в AR6
        ldhi 30h, AR6 ;         загружаем старшие 16 бит
L1:     ldii *AR6, R0
        cmpi 0, R0
        bnz L1 ;                переход на метку L1, если RC занят
        ldi *AR7, R1 ;          загружаем значение RC в R1
        ; модификация R1
        ; установка в 01₂ бит 03-02
        ldi 4, R1 ; подключение к процессору 1
        cmpi 2, R4
        ldiz 8, R1 ;            подключение к процессору 2
        ; модифицируем RCC новым значением и устанавливаем LLOCK – высокий
        sti R1, *AR7
        stii R1, *AR6 ;         разблокировка регистра коммутации
;
; Инициализация регистров Ethernet
;
        ldpk 70h; Set Data Pointer
        ldi 1h, AR0 ; // MAC1 (Rx Enable)
        sti AR0, @0h
    ldi 15h, AR0 ; // MAC2 (FULL-DUPLEX, HUGE FRAME, CRC ENABLE)
    sti AR0, @1h
    ldi 15h, AR0 ; // IPGT
    sti AR0, @2h
    ldi 0C12h, AR0 ; // IPGR (Recommended)
    sti AR0, @3h
    ldi 370Fh, AR0 ; // COLL
    sti AR0, @4h
    ldi 0600h, AR0 ; // MAXFRM
    sti AR0, @5h
    ldi 0, AR0
    or 8000h, AR0 ; // SUPP RESET INTERFACE MODULE
    sti AR0, @6h
    nop
    stik 0, @6h
    stik 0, @7h ; // TEST
    ldi 0, AR0
```

```
    or 0A5A4h, AR0 ; // SA0
    sti AR0, @10h
    ldi 0, AR0
    or 0A3A2h, AR0 ; // SA1
    sti AR0, @11h
    ldi 0, AR0
    or 0A1A0h, AR0 ; // SA2
    sti AR0, @12h
    stik 8, @8h ; // MIICNFG Host Clock, делённый на 6
    ldhi 0, R0
    or 0FF00h, R0
    sti R0, @0Fh ; // MMIIFIF cfg 0
    ldhi 0AAAh, R0
    or 0555h, R0
    sti R0, @014h ; // High and low pause control watermark
    ldhi 0555h, R0
    or 0FFFh, R0
    sti R0, @015h ; // Fabric tx watermark and tx cut thru
    ldhi 0, R0
    or 0FFFFh, R0
    sti R0, @017h ; // MMIIFIF cfg 5 assign no cenmode
    lhu1 @65h, R0 ; считываем счётчик принятых пакетов
    rpts R0
    stik 1, @65h ; DMARxStatus – обнуляем счётчик принятых пакетов
;
; Загрузка дескрипторов
;
        ldpk 71h ; Set Data Pointer
        ldhi 8000h, R1 ; load empty frame flag
        ldhi 71h, AR0
        or 51h, AR0
        rpts 9
        sti R1, *AR0++(3)

        ldi 01400h, AR0 ;     // DMA rx descriptor 1 frame start address (500h)
        sti AR0, @50h
        ldi 014Ch, AR0 ;      // DMA rx descriptor 1 Next descriptor start address
        sti AR0, @52h

        ldi 01C00h, AR0 ;     // DMA rx descriptor 2 frame start address (700h)
        sti AR0, @53h
        ldi 0158h, AR0 ;      // DMA rx descriptor 2 Next descriptor start address
        sti AR0, @55h

        ldi 02400h, AR0 ;     // DMA rx descriptor 3 frame start address (900h)
        sti AR0, @56h
        ldi 0164h, AR0 ;      // DMA rx descriptor 3 Next descriptor start address
        sti AR0, @58h

        ldi 02C00h, AR0 ;     // DMA rx descriptor 4 frame start address (B00h)
        sti AR0, @59h
```

```
        ldi 0170h, AR0 ;        // DMA rx descriptor 4 Next descriptor start address
        sti AR0, @5Bh

        ldi 03400h, AR0 ;       // DMA rx descriptor 5 frame start address (D00h)
        sti AR0, @5Ch
        ldi 017Ch, AR0 ;        // DMA rx descriptor 5 Next descriptor start address
        sti AR0, @5Eh

        ldi 03C00h, AR0 ;       // DMA rx descriptor 6 frame start address (F00h)
        sti AR0, @5Fh
        ldi 0188h, AR0 ;        // DMA rx descriptor 6 Next descriptor start address
        sti AR0, @61h

        ldi 04400h, AR0 ;       // DMA rx descriptor 7 frame start address (1100h)
        sti AR0, @62h
        ldi 0194h, AR0 ;        // DMA rx descriptor 7 Next descriptor start address
        sti AR0, @64h

        ldi 04C00h, AR0 ;       // DMA rx descriptor 8 frame start address (1300h)
        sti AR0, @65h
        ldi 01A0h, AR0 ;        // DMA rx descriptor 8 Next descriptor start address
        sti AR0, @67h

        ldi 05400h, AR0 ;       // DMA rx descriptor 9 frame start address (1500h)
        sti AR0, @68h
        ldi 01ACh, AR0 ;        // DMA rx descriptor 9 Next descriptor start address
        sti AR0, @6Ah

        ldi 05C00h, AR0 ;       // DMA rx descriptor 10 frame start address (1700h)
        sti AR0, @6Bh
        ldi 0140h, AR0 ;        // DMA rx descriptor 10 Next descriptor start address
        sti AR0, @6Dh

        ldpk 70h
        ; Receive
        ldi 140h, AR0
    sti AR0, @64h ;             DMA rx descriptor byte address
    stik 1, @63h ;              Enable DMA rx

wait ; ожидание приёма 10 пакетов
        lhu1 @65h, R0 ;         считываем счётчик принятых пакетов
        cmpi 0Ah, R0 ;          сравниваем его с 10
        blt wait ;              если меньше, то ждём
;
; Обработка принятых данных (10 пакетов)
;
   ldpk 71h
   ldhi 71h, AR0
   or 2000h, AR0 ;              AR0 = 712000 – адрес для сохранения данных
   ldi 0, R0 ;                  счётчик принятых слов
   ldhi 0FFFFh, R8 ;            константы для проверки MAC адресов
```

```
   or 0FFFFh, R8
   ldhi 05555h, R9
   or 0FFFFh, R9
; Обработка 1-го пакета
   ldhi 71h, AR1
   or 500h, AR1 ; адрес первого дескриптора в памяти Ethernet
   ldi @51h, R1 ; загрузка размера пакета в байтах
         call TestLen
         call ProcPacket
; Обработка 2-го пакета
   ldhi 71h, AR1
   or 700h, AR1 ; адрес первого дескриптора в памяти Ethernet
   ldi @54h, R1 ; загрузка размера пакета в байтах
         call TestLen
         call ProcPacket
; Обработка 3-го пакета
   ldhi 71h, AR1
   or 900h, AR1 ; адрес первого дескриптора в памяти Ethernet
   ldi @57h, R1 ; загрузка размера пакета в байтах
         call TestLen
         call ProcPacket
; Обработка 4-го пакета
   ldhi 71h, AR1
   or 0B00h, AR1 ; адрес первого дескриптора в памяти Ethernet
   ldi @5Ah, R1 ; загрузка размера пакета в байтах
         call TestLen
         call ProcPacket
; Обработка 5-го пакета
   ldhi 71h, AR1
   or 0D00h, AR1 ; адрес первого дескриптора в памяти Ethernet
   ldi @5Dh, R1 ; загрузка размера пакета в байтах
         call TestLen
         call ProcPacket
; Обработка 6-го пакета
   ldhi 71h, AR1
   or 0F00h, AR1 ; адрес первого дескриптора в памяти Ethernet
   ldi @60h, R1 ; загрузка размера пакета в байтах
         call TestLen
         call ProcPacket
; Обработка 7-го пакета
   ldhi 71h, AR1
   or 1100h, AR1 ; адрес первого дескриптора в памяти Ethernet
   ldi @63h, R1 ; загрузка размера пакета в байтах
         call TestLen
         call ProcPacket
; Обработка 8-го пакета
   ldhi 71h, AR1
   or 1300h, AR1 ; адрес первого дескриптора в памяти Ethernet
   ldi @66h, R1 ; загрузка размера пакета в байтах
         call TestLen
         call ProcPacket
```

```
; Обработка 9-го пакета
   ldhi 71h, AR1
   or 1500h, AR1 ; адрес первого дескриптора в памяти Ethernet
   ldi @69h, R1 ; загрузка размера пакета в байтах
         call TestLen
         call ProcPacket
; Обработка 10-го пакета
   ldhi 71h, AR1
   or 1700h, AR1 ; адрес первого дескриптора в памяти Ethernet
   ldi @6Ch, R1 ; загрузка размера пакета в байтах
         call TestLen
         call ProcPacket
PacketsEnd

   ldpk 71h
   ldi 1000h, R7
   sti R7, @0h ; // DMA tx descriptor 1 start address
   ldi 50h, R7
   sti R7, @1h ; // DMA tx descriptor 1 frame size bytes
   ldi 400h, R7
   sti R7, @2h ; // DMA tx descriptor 1 next descriptor start address
   ; load tx data
   ldhi 71h, AR0
   or 400h, AR0
   ldhi 0FFFFh, R3
   or 0FFFFh, R3
   rpts 2h
   sti R3, *AR0++
   sti R0, *AR0++ ; слово данных с количеством принятых слов
   rpts 14h
   stik 0, *AR0++

   ldpk 70h
   ldi 0h, AR0
   sti AR0, @61h ; DMA tx descriptor byte address
   stik 1, @60h ; Enable DMA tx
   ; загрузка адреса для MEMORY Bootloader
   ldhi 71h, AR1
   or 2000h, AR1
   BR MEMORY

STOP: BR STOP ; останов
;
; Подпрограмма обработки принятого пакета
; AR1 – адрес пакета в памяти
; R1 – длина пакета
;
ProcPacket
   cmpi *AR1, R8 ; проверяем адрес назначения
   bnz EndProcPacket ; в конец, если неверный
   cmpi *+AR1(1), R9 ; проверяем адрес источника
```

```
   bnz EndProcPacket ; в конец, если неверный
   lsh -2, R1 ; получаем размер пакета в словах
   cmpi 6, R1 ; проверяем длину
   blt PacketsEnd ; если меньше, то
; копирование данных пакета
        addi R1, R0 ; увеличиваем счётчик принятых слов
        subi 4, R0 ; убираем адреса и CRC
        addi 3, AR1 ; теперь AR1 указывает на первое слово данных
        subi 6, R1 ; 3 - адреса, 1 - CRC
   ldi *AR1++, R2 ; загрузка первого слова
   rpts R1
   ldi *AR1++, R2
||sti R2, *AR0++
   sti R2, *AR0++ ; сохранение последнего слова
EndProcPacket
        rets
;
; Подпрограмма корректирует длину пакета,
; если длина = 40 байт
;
TestLen
   cmpi 40h, R1 ; сравниваем с минимальной длиной пакета
   bnz sk
   lbu0 *+AR1(2), R1 ; при минимальной корректируем длину пакета
   addi 4, R1 ; из-за CRC
sk rets
;-------------------------------------------------------------------------;
; Подпрограмма USB 2.0 BOOTLOADER
;-------------------------------------------------------------------------;

; Данные
;-------------------------------------------------------------------------;
*
* Стандартный дескриптор устройства
*
                    .data
_myvect:
                    .word BOOT          ; Reset
                    .word STOP          ; NMI
                    .word STOP          ; TINI0
                    .word EXT0          ; IIOF0 – прерывания от USB
                    .word STOP
                    .word STOP
                    .word STOP
                    .word STOP
                    .word STOP
                    .word STOP

                    .word STOP
                    .word STOP
                    .word STOP
```

```
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP
                .word STOP

DevDescrWord0   .word 02000112h
DevDescrWord1   .word 40000000h
DevDescrWord2   .word 00077007h
DevDescrWord3   .word 00000001h
DevDescrWord4   .word 00000100h
*
*Дескриптор конфигурации
*
ConfDescrWord0  .word 00200209h
ConfDescrWord1  .word 40000101h
ConfDescrWord2  .word 00000050h
*
* Дескрипторы конечной точки и др. (конфигурации интерфейса)
*
ConfIntEndpDescrWord0 .word 00200209h
ConfIntEndpDescrWord1 .word 40000101h
ConfIntEndpDescrWord2 .word 00040950h
ConfIntEndpDescrWord3 .word 0FF000200h
ConfIntEndpDescrWord4 .word 050700FFh
```

```
ConfIntEndpDescrWord5 .word 02000281h
ConfIntEndpDescrWord6 .word 02050700h
ConfIntEndpDescrWord7 .word 00020002h

ivta            .word _myvect ; адрес секции векторов прерываний
localcontol     .word 1EF78000h ; Local bus – no extra cycles, 2Gb page size

        .sect "uart_subr"
SetFlags:
        ldi     0, flag ; флаг загрузки первого блока программы
        ldi     0FFh, crc ; начальное значение контрольной суммы
   BU           SetDP

test_parity:
   XOR          last_byte, crc ; подсчитываем контрольную сумму
   ldi          7, RC
   rptb         calc_crc
   lsh          1, crc
   tstb         0100h, crc
   bz           calc_crc
   XOR          031h, crc
calc_crc:
        nop

        tstb    080h, status ; проверяем флаг ошибки паритета
        bz      RNextB ; если ошибки нет – получаем следующий байт
        stik    0Fh, *address ; иначе – отсылаем код ошибки и загрузчик
        BU      Terminate ; завершает работу

SendSumm:
        iack    *data ; вывод сигнала IACK
        stik    0Ah, *address
        LBU0    crc, R7
        sti     R7, *address
        BU      UnlockRC
;-------------------------------------------------------------;
        .sect "boot"
USB20:
   ldhi 2Fh, SP
   or 0FF00h, SP ; Указатель стека
        LDP ivta
        LDI  @ivta, AR0
        LDPE AR0, IVTP ; Указатель на таблицу прерываний
        ldi 9h, IIF ; Разрешаем прерывания по IIOF0
;
; Коммутация устройства USB 2.0
;
* AR7 – адрес регистра коммутации RC
* AR6 – адрес регистра семафора RS
; захват регистра коммутации RC
; устанавливаем LLOCK в низкий уровень и читаем RCC в R0
```

```
; загрузка адреса RC в AR7
        ldhi 30h, AR7 ; загружаем старшие 16 бит
        or 1, AR7 ; AR7 = 30 0001h
        ; загрузка адреса RS в AR6
        ldhi 30h, AR6 ; загружаем старшие 16 бит
LL1: ldii *AR6, R0
        cmpi  0, R0
    bnz  LL1 ; переход на метку LL1, если RC занят
        ldi  *AR7, R1 ; загружаем значение RC в R1
        ; модификация R1
        ; установка в 01₂ бит 05-04
        ldi 10h, R1 ; код подключения USB к процессору 1
        cmpi 2, R4
        ldiz 20h, R1 ; код подключения USB к процессору 2
        ; модифицируем RCC новым значением и устанавливаем LLOCK – высокий
        sti R1, *AR7
        stii R1, *AR6 ; разблокировка регистра коммутации
; записываем DeviceDescriptor в USB RAM (610080h)
        ldhi 61h, AR0
        or 80h, AR0
        ldi @DevDescrWord0, R0
        sti R0, *AR0++
        ldi @DevDescrWord1, R0
        sti R0, *AR0++
        ldi @DevDescrWord2, R0
        sti R0, *AR0++
        ldi @DevDescrWord3, R0
        sti R0, *AR0++
        ldi @DevDescrWord4, R0
        sti R0, *AR0
; записываем Configuration Descriptor в USB RAM (610090h)
        ldhi 61h, AR0
        or 90h, AR0
        ldi @ConfDescrWord0, R0
        sti R0, *AR0++
        ldi @ConfDescrWord1, R0
        sti R0, *AR0++
        ldi @ConfDescrWord2, R0
        sti R0, *AR0
; записываем Config, Interface, EndPoints Descr в USB RAM (610100h)
        ldhi 61h, AR0
        or 100h, AR0
        ldi @ConfIntEndpDescrWord0, R0
        sti R0, *AR0++
        ldi @ConfIntEndpDescrWord1, R0
        sti R0, *AR0++
        ldi @ConfIntEndpDescrWord2, R0
        sti R0, *AR0++
        ldi @ConfIntEndpDescrWord3, R0
        sti R0, *AR0++
        ldi @ConfIntEndpDescrWord4, R0
```

```
        sti R0, *AR0++
        ldi @ConfIntEndpDescrWord5, R0
        sti R0, *AR0++
        ldi @ConfIntEndpDescrWord6, R0
        sti R0, *AR0++
        ldi @ConfIntEndpDescrWord7, R0
        sti R0, *AR0
        ldpk 60h ; установка data pointer
;
; инициализация регистров USB
;
        ; 2000840 - Control, Bulk, MAX_PL_SZ = 64
        ldhi 200h, AR0
        or 0840h, AR0
        sti AR0, @EP0_CSR
        ; 6040840 - IN, Bulk, MAX_PL_SZ = 64
        ldhi 604h, AR0
        or 0840h, AR0
        sti AR0, @EP1_CSR
        ; A080A00 - OUT, Bulk, MAX_PL_SZ = 512
        ldhi 0A08h, AR0
        or 0A00h, AR0
        sti AR0, @EP2_CSR
        ; инициализация буферов
        ; 400 0000 - Buffer size = 512 byte Pointer = 0
        ldhi 400h, AR0
        sti AR0, @EP0_OBA

        ; 0400 1400 - Buffer size = 512 byte Pointer = 1400h (610500h)
        ; 02 1400 - Buffer size = 1 byte Pointer = 1400h (610500h)
        ; 0 - данные не готовы
        stik 0, @EP1_IBA
        ldhi 2h, AR0
        or 1400h, AR0
        sti AR0, @EP1_IBA

        ; 4000 1800 - Buffer size = 8192 byte Pointer = 1800h (610600h)
        ldhi 4000h, AR0
        or 1800h, AR0
        sti AR0, @EP2_OBA

        ; разрешение прерываний
        ldhi 040h, AR0 ; Receive Control Packet
        sti AR0, @EP0_INT
        ldhi 4000h, AR0
        sti AR0, @EP1_INT ; Transmitted Data Packed

        ldhi 2000h, AR 0 ; Receive Data Packed
        sti AR0, @EP2_INT
        or 2000h, ST ; Разрешаем прерывания глобально
```

```
        ldi 0, R10 ; счётчик прерываний
        ldi 1h, R11
        ldi 0, R9
        ldhi 61h, AR7 ; начальный адрес для считывания дескриптора
        ldhi 61h, AR5
        or 85h, AR5 ; начальный адрес для записи данных Control EP0
;
; start USB
;
        ldi 0, AR0
        or 080h, AR0
        sti AR0, @CSR ; Enable Core, Full Speed

en      br en ; останов

*****************************************************************************
* Обработчик прерываний USB
*****************************************************************************
* AR7 – адрес для считывания дескриптора
* R9 – предыдущее значение буфера
* R8 – количество слов для считывания
*****************************************************************************
EXT0:           ; обработчик прерываний USB
                ldi @INT_SRC, R0
                tstb SetBit0, R0
                bz n1 ; если прерывание не от Control Endpoint
                ldi @EP0_INT, R0 ; сброс прерывания
                lhu0 @EP0_OBA, R0 ; half-word0 в R0
                lsh –2, R0 ; делим на 4
    subi3 R9, R0, R8 ; R8 теперь содержит количество слов для считывания
    ldi R0, R9
    cmpi 1, R8
    callgt ReadControl ; вызываем, если R1 > 1
    br endisr
n1:  tstb SetBit1, R0
                callnz EP1INT ; прерывание от IN Endpoint
                tstb SetBit2, R0
                callnz EP2INT ; прерывание от OUT Endpoint
                ; br endisr
endisr: RETI

*****************************************************************************
* Обработка управляющего пакета
*****************************************************************************
* R0 – первое слово дескриптора
* R1 – второе слово дескриптора
* R2 – младшее слово дескриптора
*****************************************************************************
ReadControl:
RC1     ldi *AR7++, R0 ; R0 – первое слово дескриптора
        ldi *AR7++, R1 ; R1 – второе слово дескриптора
```

```
        lhu0 R0, R2 ; R2 – младшее слово дескриптора
        cmpi 0680h, R2 ; Запрос Device Descriptor
        callz DeviceDescr
        cmpi 0500h, R2 ; Запрос Set Address
        callz SetAdrDescr
        subi 2, R8
        bp RC1 ; если больше одного дескриптора
RCEndrets

DeviceDescr
        lhu1 R0, R2 ; R2 – старшее слово дескриптора
        cmpi 0100h, R2
        bnz next1 ; если не Device Descriptor
        ; Стандартный дескриптор устройства
        ldhi 24h, AR0 ; BUF_SZ=12h
        or 200h, AR0 ; BUF_PTR=80h
        sti AR0, @EP0_IBA
        br endDD ; в конец подпрограммы

next1: cmpi 0200h, R2
                bnz endDD ; в конец, если это не дескриптор конфигурации
                lhu1 R1, R2 ; старшее полуслово второго слова дескриптора
                cmpi 0009h, R2
                bnz next2
                addi 1, R7 ; tmp
                ; дескриптор конфигурации
        ldhi 12h, AR0 ; BUF_SZ=9h
        or 240h, AR0 ; BUF_PTR=90h
        sti AR0, @EP0_IBA
        br endDD ; в конец подпрограммы
                ; полный дескриптор конфигурации
next2: ldhi 40h, AR0 ; BUF_SZ=20h
        or 400h, AR0 ; BUF_PTR=400h
        sti AR0, @EP0_IBA
        br endDD ; в конец подпрограммы
endDDrets; возврат из подпрограммы

*****************************************************************************
* Установка Function Address
*****************************************************************************
SetAdrDescr
        lbu2 R0, AR0
        sti AR0, @FA
                rets
*****************************************************************************
* Обработчик прерываний IN Enpoint1
*****************************************************************************
EP1INT
                rets
*****************************************************************************
```

```
*************************************************************************
* Обработчик прерываний OUT Endpoint2
*************************************************************************
* R10 – счётчик прерываний
*************************************************************************
EP2INT
                push AR0
                ldi @EP2_INT, AR0
                tstb SetBit9, AR0 ; пакет получен?
                bz skip
                addi 1, R10 ; счётчик вызовов
                cmpi 20h, R10 ; ожидаем 32 пакета
                bnz skip
        ldhi 61h, AR1
        or 0600h, AR1
        ldi 0, IIF ; запрещаем внешние прерывания
                br MEMORY
skip
                pop AR0
                rets
                .end
```

5.11 Сопроцессор ПДП

Сопроцессор прямого доступа к памяти (ПДП) – встроенное программируемое устройство, позволяющее одновременно осуществлять передачу данных памяти и выполнять операции ЦПУ с наименьшими потерями. В этом разделе описывается сопроцессор ПДП и приводятся советы по его программированию.

5.11.1 Ключевые свойства ПДП

Сопроцессор ПДП – самопрограммируемое периферийное устройство, которое максимизирует вычислительную производительность ЦПУ, избавляя его от затратного обмена пакетами данных и от обременительных операций ввода-вывода. Далее приводятся ключевые особенности сопроцессора ПДП.

- Передача в память и из памяти по всей карте памяти процессора. Например, передача может осуществляться во внутреннюю и внешнюю память и обратно, а также через любой из шести коммуникационных портов.
- Шесть каналов ПДП для передачи из памяти в память в основном режиме; специальный режим разделения поддерживает 12 каналов ПДП для передачи данных между коммуникационными портами и памятью.
- Автоматическая инициализация регистров через связанные списки, хранящиеся в памяти, что позволяет ПДП работать непрерывно без вмешательства ЦПУ.
- Одновременная работа ЦПУ и сопроцессора ПДП. При этом передачи ПДП осуществляются с такой скоростью, как и передачи ЦПУ (обеспечивается выделенными внутренними шинами адреса и данных ПДП).
- Регистры исходного и конечного адреса с переменными индексами, что позволяет перемещаться в матрицах по строкам или столбцам.
- Бит-реверсная адресация для БПФ (FFT).
- Синхронизация передачи данных посредством внешних и внутренних прерываний.

5.11.2 Функциональное описание ПДП

Сопроцессор ПДП поддерживает шесть каналов ПДП, которые выполняют передачу данных из любого места карты памяти ядра процессора 1867ВЦ8Ф1.

Каждый канал ПДП управляется девятью регистрами, которые расположены в периферийном адресном пространстве ядра процессора 1867ВЦ8Ф1 (см. рисунок 5.108). Основные регистры ПДП описаны в 5.11.3.

Сопроцессор ПДП имеет выделенные внутренние шины адреса и данных. Все обращения, выполняемые шестью каналами ПДП, осуществляются по этим выделенным шинам и арбитрируются сопроцессором ПДП. Шесть каналов ПДП передают данные в режиме с последовательным разделением времени (не одновременно), поскольку при этом используются одни и те же общие шины.

Каналы ПДП могут работать постоянно или запускаться от внешних (IIOF(3-0)_x#) или внутренних (встроенные таймеры и коммуникационные порты) прерываний.

Сопроцессор ПДП может передавать данные в бит-реверсной (для приложений БПФ) или в линейной форме; он также может передавать матричные данные в виде строк и столбцов.

Сопроцессор ПДП поддерживает два базовых операционных режима:

- основной режим: используется для передачи из памяти в память. Режим описан в 5.11.4. Последовательность передачи в основном режиме приведена в 5.11.2.1;
- режим разделения: используется для двустороннего обмена данными между памятью и коммуникационными портами. Режим описан в 5.11.5.

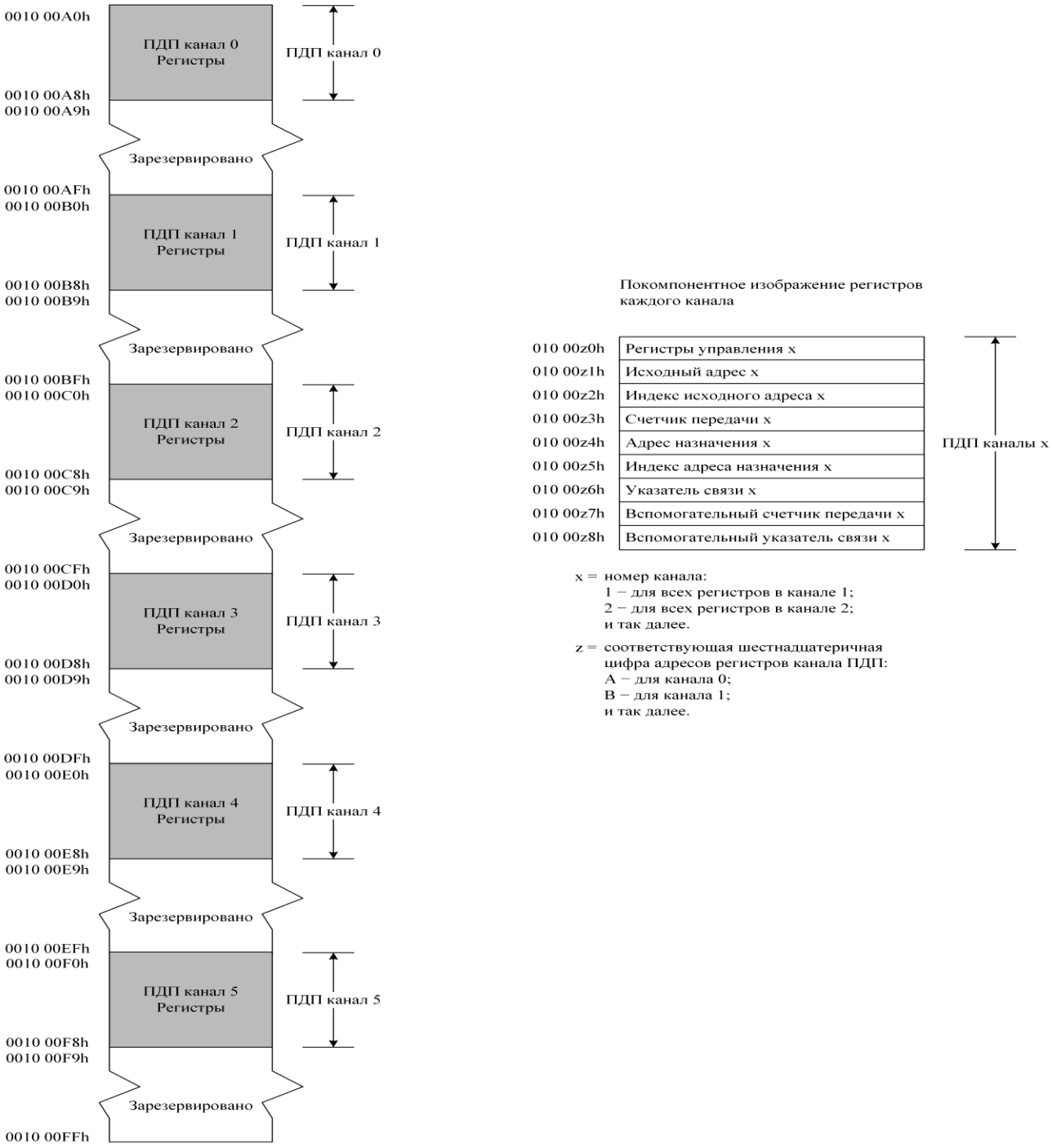


Рисунок 5.108 − Карта памяти сопроцессора ПДП

5.11.2.1 Базовые операции ПДП

Если блок данных передаётся из одной области памяти в другую (основной режим), выполняются следующие операции:

Инициализация 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 счётчик передачи загружается число передаваемых слов.
- В индексный регистр начального/конечного адреса загружается шаг обновления 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.

Запуск ПДП

- Сопроцессор ПДП запускается посредством поля 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 счётчик передачи декрементируется, и значение индексного регистра конечного адреса добавляется к значению регистра конечного адреса.

Замечание: оба индексных регистра (начального и конечного адреса) содержат знаковые значения. Этим допускается переменная величина шага или продолжение чтения из памяти и/или запись в память. Если индексный регистр равен 0, сопроцессор ПДП передаёт данные между определёнными фиксированными адресами.

- В течение каждого цикла записи, счётчик передачи декрементируется. Блок передаваемых данных заканчивается, когда счётчик передачи достигает 0 и завершается последняя операция записи передаваемых данных. Канал ПДП устанавливает флаг прерывания счётчика передачи (TCINT) в регистре управления каналом ПДП.

После завершения передачи блока данных сопроцессор ПДП может быть запрограммирован для следующего:

- остановлен до перепрограммирования (разряды TRANSFER MODE = 01_2);
- продолжение передачи данных (разряды TRANSFER MODE = 00_2);
- сгенерировать прерывание для сигнализации ЦПУ, что передача данных завершена (разряд TCC = 1_2);
- автоинициализироваться для запуска передачи следующего блока данных (разряды TRANSFER MODE = 10_2 или 11_2).

Каждый канал ПДП считывает новое значение регистра ПДП из памяти, загружает его в соответствующий регистровый файл, в соответствии с загруженным значением начинает новую передачу. ЦПУ в любом случае должен инициализировать передачи, определяемые значениями разрядов режима передачи:

- автоинициализация разрядами TRANSFER MODE = 10_2 выполняется без вмешательства ЦПУ;
- автоинициализация разрядами TRANSFER MODE = 11_2 требует вмешательства ЦПУ для запуска ПДП.

5.11.3 Регистры ПДП

Каждый канал ПДП имеет девять регистров:

- Регистр управления: содержит информацию о состоянии и режиме канала.
- Регистр начального адреса: содержит адрес памяти, по которому будут считываться данные.
- Индексный регистр начального адреса: содержит значение шага (знаковое 32-разрядное число), используемое для инкрементирования или декрементирования регистра начального адреса.
- Регистр конечного адреса: содержит адрес памяти, куда будут записываться данные.
- Индексный регистр конечного адреса: содержит значение шага (знаковое 32-разрядное число), используемое для инкрементирования или декрементирования регистра конечного адреса.
- Регистр-счётчик передачи: содержит размер блока данных, передаваемых в основном или раздельном режиме (основной канал).
- Вспомогательный регистр-счётчик 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, счётчик передачи, вспомогательный счётчик передачи устанавливаются в 0, а остальные регистры в неопределённое состояние.

5.11.3.1 Регистр управления

Формат регистра управления каналом ПДП показан на рисунке 5.109. 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 начальные и конечные адреса для просчёта посредством прямой адресации, конфигурирует канал ПДП для работы в основном режиме.

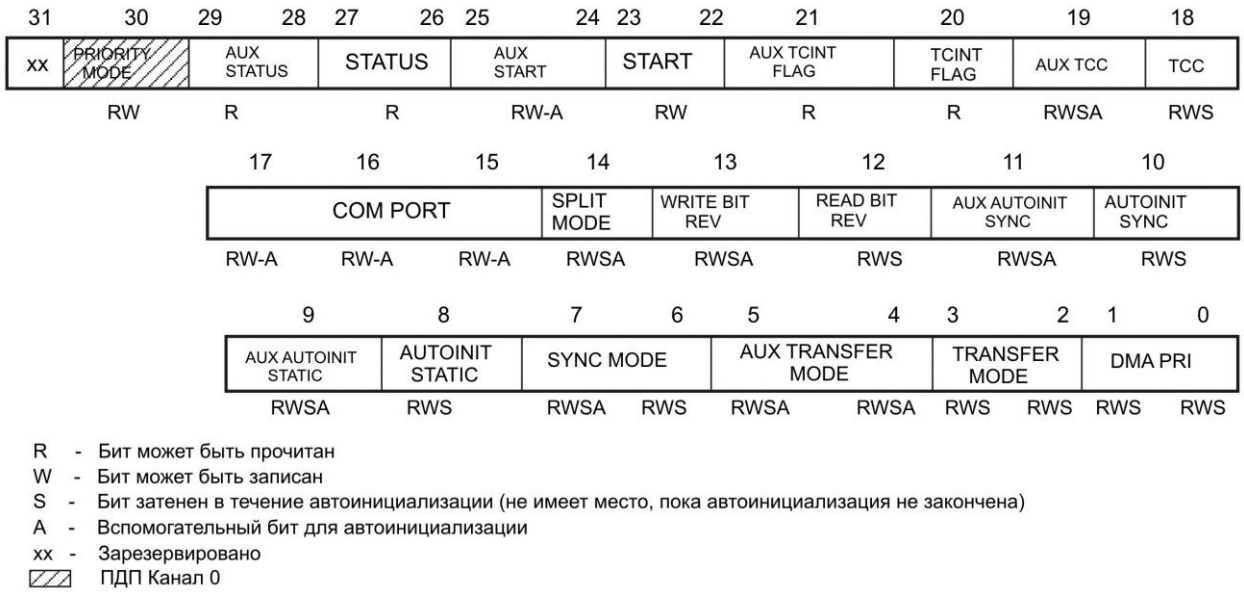


Рисунок 5.109 − Регистр управления каналом ПДП

DMA PRI − устанавливает приоритет сопроцессора ПДП. Определяет правила арбитража, которые будут использоваться, когда канал ПДП и ЦПУ обращаются к одному источнику. Правила изложены в таблице 5.36.

TRANSFER MODE − определяет режим передачи канала ПДП. Влияет на основной режим и на основной канал в режиме разделения. Разряды определены в таблице 5.37.

AUX TRANSFER MODE − определяет режим передачи канала ПДП. Влияет на вспомогательный канал в режиме разделения. Разряды определены в таблице 5.37.

SYNC MODE − определяет режим синхронизации для выполнения передачи данных.

Эти разряды работают по-разному в основном и раздельном режимах. В таблицах 5.38 и 5.39 приведены описания 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, прерывание для записи фиксируется каналом ПДП. После завершения чтения может быть выполнена запись.

AUTOINIT STATIC − этот разряд 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 разряд = 0, указатель инкрементируется в течение автоинициализации. Если разряд = 1, указатель не инкрементируется (он статичен) в течение автоинициализации.

AUX AUTOINIT STATIC − работает как и AUTOINIT STATIC, но только применительно к вспомогательному каналу в режиме разделения.

AUTOINIT SYNC − этот разряд работает только в соответствии с режимом синхронизации ПДП, определяемой разрядами 6 − 7. Он влияет на прерывание, которое включается регистром разрешения прерываний ПДП, используемый для чтений ПДП: если разряд = 0, прерывание игнорируется, автоинициализированные чтения не синхронизируются с сигналами прерываний; если разряд = 1, прерывание определяется и используется для синхронизации автоинициализированных чтений. Этот разряд влияет на основной режим и на основной канал в режиме разделения (смотри разряд SPLIT MODE). Влияние разрядов AUTOINIT SYNC и SYNC MODE на автоинициализацию приведено в таблице 5.44.

AUX AUTOINIT SYNC − работает как и AUTOINIT SYNC, но только применительно к вспомогательному каналу в режиме разделения. Влияние разрядов AUX AUTOINIT SYNC и SYNC MODE на автоинициализацию приведено в таблице 5.44.

READ BIT REV − выбирает тип адресации для изменения начального адреса. Если разряд = 0, начальный адрес изменяется посредством 32-разрядной прямой адресации. Если разряд = 1, начальный адрес изменяется посредством 24-разрядной бит-реверсной адресации. Этот разряд влияет на основной режим и на чтения (начального адреса) основного канала в режиме разделения.

WRITE BIT REV − выбирает тип адресации для изменения начального адреса. Если разряд = 0, конечный адрес изменяется посредством 32-разрядной прямой адресации. Если разряд = 1, конечный адрес изменяется посредством 24-разрядной бит-реверсной адресации. Этот разряд влияет на основной режим и на записи (конечного адреса) вспомогательного канала в режиме разделения.

SPLIT MODE − этот разряд управляет режимами сопроцессора ПДП. Если разряд = 0, ПДП осуществляет передачу из памяти в память. Этот режим называется основным. Если разряд = 1, ПДП переходит в режим разделения, при котором каждый канал ПДП разделяется на два канала, позволяя 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 ниже в 5.11.5.

COM PORT − эти разряды определяют, какой коммуникационный порт (от 0002 до 1012) будет использоваться для передачи данных ПДП. Если SPLIT MODE = 0, COM PORT не влияет на операции канала ПДП. Если SPLIT MODE = 1, COM PORT определяет, какой из шести коммуникационных портов будет использоваться с каналом ПДП. COM PORT может быть изменён посредством автоинициализации в основном режиме или посредством автоинициализации вспомогательного канала в режиме разделения.

TCC − управляет прерыванием счётчика передачи. Если TCC = 1, импульс прерывания канала ПДП посылается к ЦПУ после перехода счётчика передачи через 0 и записи последних переданных данных. Если включен, соответствующее прерывание ПДП (ПДП INT0-INT1) происходит в соответствии с вектором, показанным на рисунке 5.59. Если TCC = 0, импульс прерывания канала ПДП не посылается к ЦПУ, когда счётчик передачи переходит через 0. Этот разряд влияет на основной режим и на основной канал в режиме разделения.

AUX TCC − управляет прерыванием вспомогательного счётчика передачи. Если разряд = 1, импульс прерывания канала ПДП посылается к ЦПУ после того, как вспомогательный счётчик передачи перейдёт через 0 и будет завершена запись последних переданных данных. Если включено соответствующее прерывание ПДП (ПДП INT0 − INT1), происходит как показано на рисунке 5.59. Если разряд = 0, импульс прерывания канала ПДП не посылается к ЦПУ, когда вспомогательный счётчик передачи переходит через 0. Этот разряд влияет только на вспомогательный канал в режиме разделения.

TCINT FLAG − флаг прерывания счётчика передачи. Этот флаг устанавливается в 1, если счётчик передачи переходит через 0 и завершается запись последних переданных данных. Когда считывается регистр управления каналом ПДП, этот флаг сбрасывается, пока не будет установлен посредством ПДП в том же цикле, что и чтение. TCINT FLAG управляется посредством основного режима и основным каналом в режиме разделения.

AUX TCINT FLAG − флаг прерывания вспомогательного счётчика передачи. Этот флаг устанавливается в 1, если вспомогательный счётчик передачи переходит через 0 и завершается запись последних переданных данных. Когда считывается регистр управления каналом ПДП, этот флаг сбрасывается, пока не будет установлен посредством ПДП в том же цикле, что и чтение. AUX TCINT FLAG управляется посредством вспомогательного канала в режиме разделения. Так как для канала ПДП доступно только одно прерывание, вы можете определить, какое событие устанавливает прерывание посредством TCINT FLAG и AUX TCINT FLAG.

START − запускает и останавливает канал ПДП разными способами (описаны в таблице 5.40). START влияет на основной режим и на основной канал в режиме разделения. Если эти разряды используются для удержания канала между последовательными автоинициализациями, разряды START и AUX START должны удерживать последовательность автоинициализации. Если разряды START и AUX START изменяются посредством канала ПДП (например, подавая код останова 10_2 на счётчик передачи для остановки передачи) и запись выполняется из внешнего источника в регистр управления каналом ПДП, внутреннее изменение разрядов START и AUX START каналом ПДП имеет приоритет. Смотри значения разрядов TRANSFER MODE 01_2 в таблице 5.37 для более подробной информации.

AUX START − запускает и останавливает канал ПДП разными способами (описаны в таблице 5.40). AUX START влияет только на вспомогательный канал в режиме разделения.

STATUS − отображает состояние канала ПДП как показано в таблице 5.41. STATUS обновляется в основном режиме и посредством основного канала в режиме разделения. Обновления происходят в каждом цикле. Разряды STATUS и AUX STATUS также определяются, если канал ПДП был остановлен или был сброшен после записи в разряды START и AUX START.

AUX STATUS − отображает состояние канала ПДП как показано в таблице 5.41. AUX STATUS обновляется только посредством вспомогательного канала в режиме разделения. Обновления происходят в каждом цикле.

PRIORITY MODE − режим приоритета доступа канала ПДП: если разряд = 0, приоритет определяется в соответствии с циклом, описанным в 5.11.6. Если разряд = 1, приоритет фиксирован, как описано в 5.11.6. Этот разряд доступен, только канал ПДП равен 0.

Таблица 5.36 − Разряды DMA PRI и правила арбитража ЦПУ/ПДП

Разряды DMA PRI 1−0	Описание
0 0	Доступ сопроцессора ПДП имеет приоритет ниже приоритета доступа ЦПУ. Если канал ПДП и ЦПУ обращаются к одному ресурсу памяти, работает ЦПУ. Эти разряды выставляются так при сбросе
0 1	Эти установки выбирают 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. Сначала канал ПДП и ЦПУ запрашивают один и тот же ресурс памяти, ЦПУ имеет приоритет. Если в следующем цикле инструкций сопроцессор ПДП и ЦПУ снова обращаются к одному и тому же ресурсу памяти, 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, ЦПУ имеет приоритет
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, работает ПДП

Таблица 5.37 − Описание поля TRANSFER MODE (AUX TRANSFER MODE)

Разряды TRANSFER MODE 3 − 2/(5 − 4)	Описание
0 0	Передача данных не прерывается счётчиком передачи, и не выполняется автоинициализация. TCINT (прерывание счётчика передачи) и AUX TCINT при этом могут использоваться для обозначения прерывания, когда счётчик передачи переходит через 0. Канал ПДП продолжает работу. Заметьте, что адрес продолжает инкрементироваться, пока счётчик передачи не достигнет максимального значения 0FFF FFFFh
0 1	Передачи данных отменяются счётчиком передачи. Автоинициализация не выполняется. Код останова 10_2 помещается в поле START (или AUX START), когда передача данных завершена
1 0	Автоинициализация выполняется, когда счётчик 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 счётчик передачи обращается в 0, операция останавливается до тех пор, пока ЦПУ запустит сопроцессор ПДП, используя поле START (AUX START) в регистре управления каналом ПДП (разряды 22-23 и 24-25, таблица 5.40). Код останова 10_2 помещается в поле START (или AUX START) сопроцессором ПДП

Таблица 5.38 − Описание поля SYNC MODE в основном режиме

Разряды SYNC MODE 7 – 6	Описание
0 0	Синхронизация отсутствует. Прерывания игнорированы, см. рисунок 5.134
0 1	Синхронизация с источником данных. Чтение не выполняется до тех пор, пока произойдёт разрешение прерывания (смотри рисунок 5.135а). Прерывание определяется полем DMAx READ регистра разрешения прерываний ПДП (DIE). (см. 5.11.10.1)
1 0	Синхронизация с приёмником данных. Запись не выполняется до тех пор, пока произойдёт разрешение прерывания (смотри рисунок 5.136а). Прерывание определяется полем DMAx WRITE регистра разрешения прерываний ПДП (DIE). (см. 5.11.10.1)
1 1	Синхронизация с источником и приёмником 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 5.137

Таблица 5.39 − Описание поля SYNC MODE в режиме разделения

Разряды SYNC MODE 7 – 6	Описание
0 0	Синхронизация отсутствует. Прерывания игнорированы, см. рисунок 5.134
0 1	Синхронизация с приёмником данных. Запись из основного канала в выходной буфер FIFO коммуникационного порта не выполняется до тех пор, пока произойдёт разрешение прерывания (смотри рисунок 5.136б). Прерывание определяется полем DMAx PRIMARY WRITE регистра разрешения прерываний ПДП (DIE) (см. 5.11.10.1)
1 0	Синхронизация с источником данных. Чтение во вспомогательный канал из входного буфера FIFO коммуникационного порта не выполняется до тех пор, пока произойдёт разрешение прерывания (смотри рисунок 5.135б). Прерывание определяется полем DMAx AUXILIARY READ регистра разрешения прерываний ПДП (DIE) (см. 5.11.10.1)
1 1	Синхронизация с источником и приёмником 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) (см. 5.11.10.1)

Таблица 5.40 − Описание поля START (AUX START)

Разряды START (AUX START) 23 − 22 (25 − 24)	Описание
0 0	Сброс канала ПДП. Циклы чтения или записи канала ПДП завершены (не прерваны); любое чтение данных игнорируется. Любые незаконченные (не запущенные) чтения и записи отменяются. Вспомогательный (AUX START = 00_2) и основной (START = 00_2) счётчики передачи сбрасываются в 0. Когда канал ПДП стартует после сброса, начинается новая передача, поэтому выполняется чтение. В этом режиме производится непосредственная остановка без загрузки каких-либо других регистров
0 1	Остановка ПДП на границе чтения и записи. Канал ПДП останавливается на первой доступной границе между чтением и записью. Если чтение или запись уже начаты, чтение или запись завершается перед остановкой. Если чтение или запись ещё не начаты, чтение или запись не начинается. В этом режиме производится непосредственная остановка без загрузки каких-либо других регистров
1 0	Остановка ПДП на границе передачи данных. Останавливает канал ПДП на первой доступной границе передачи данных. Если передача данных ПДП уже начата, полная передача данных продолжается до завершения, включая циклы чтения и записи, до остановки. Если передача данных не была начата, ничего не запускается. В этом режиме производится непосредственная остановка без загрузки каких-либо других регистров. Это также относится к значениям после завершения передачи данных ПДП
1 1	Запуск ПДП. Запись 11_2 в это поле запускает выполнение ПДП, используя значения регистров канала ПДП разных каналов (рисунок 5.108). Если ПДП в режиме автоинициализации, все регистры ПДП загружаются значениями перед началом операции. Сопроцессор ПДП запускается после сброса, если до этого был сброс (разряды START или AUX START равны 00_2) или повторный запуск из предыдущего состояния, если был остановлен (разряды START или AUX START равны 01_2 или 10_2)

Таблица 5.41 − Описание поля STATUS (AUX STATUS)

Разряды START (AUX START) 27 − 26 (29 − 28)	Описание
0 0	Канал ПДП удерживается на границе передачи данных (запись завершена, чтение не начато). Это значение при сбросе после останова на границе передачи 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

5.11.3.2 Адресные и индексные регистры

Как показано на рисунке 5.110, регистры начального и конечного адреса канала ПДП имеют соответствующие им индексные регистры. После каждого цикла чтения канала ПДП (из начального адреса) или записи (по конечному адресу) соответствующий генератор адреса (начального или конечного) добавляет значение индексного регистра к значению адресного регистра и помещает результат в адресный регистр. Таким образом, адресный регистр работает в качестве аккумулятора, так как возвращает значение суммы своего предыдущего значения и значения индексного регистра, как показано ниже:

Адресный регистр + Индексный регистр → Адресный регистр

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:

- прямое (нормальное сложение): READ BIT REV = 0 или WRITE BIT REV = 0;
- бит-реверсное (обратное с переносом): READ BIT REV = 1 или WRITE BIT REV = 1.

Значения индексных регистров (начального и конечного адреса) являются знаковыми.

(a) Операции регистра начального адреса

(b) Операции регистра конечного адреса

Рисунок 5.110 − Генерация адресов сопроцессора ПДП

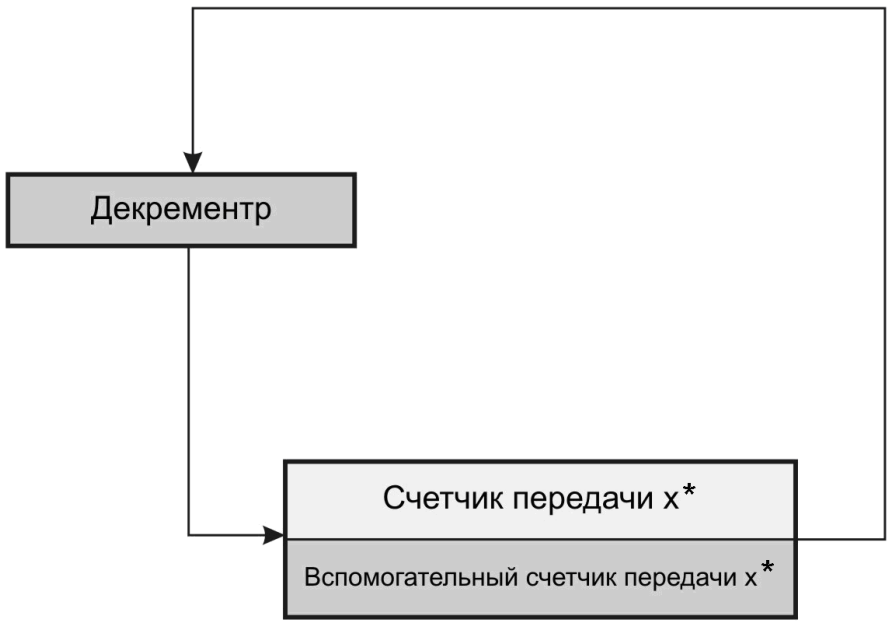
5.11.3.3 Счётчик передачи и вспомогательный счётчик передачи

Эти регистры содержат значение числа передаваемых слов.

Рисунок 5.111 показывает шесть счётчиков передачи и шесть вспомогательных счётчиков передачи. Канал ПДП в режиме разделения использует вспомогательный счётчик передачи для вспомогательного канала и основной счётчик передачи для основного канала. При сбросе значения этих регистров сбрасываются в 0.

Счётчики декрементируются после завершения выборки адреса для записи блока передаваемых данных. TCINT FLAG и AUX TCINT FLAG (разряды 20 и 21 регистра управления каналом ПДП, рисунок 5.109) не устанавливаются до тех пор, пока счётчик не декрементируется и не будет завершена передача последнего блока данных. Соответственно, контроллер прерываний ЦПУ не увидит прерывание до тех пор, пока счётчик не декрементируется и не будет завершена передача последнего блока данных.

Декрементр проверяет, обратился или нет счётчик передачи в 0 после того, как был произведён декремент. В результате, если счётчик имеет значение, равное 1, канал ПДП будет остановлен после выполнения одной передачи. Таким образом, устанавливая счётчик передачи в 1, канал ПДП передаёт минимально возможное число слов (1 раз). Счёт предполагается беззнаковым целым. Передача может быть остановлена, если после декремента счётчик обратился в 0. Если канал ПДП не останавливается после того, как счётчик обратился в 0, счётчик продолжает декрементировать значение ниже 0. Таким образом, устанавливая счётчик передачи в 0, канал ПДП передаёт максимально возможное число слов (10000 0000h раз).



* x – Номер канала ПДП (0 – 5).

Рисунок 5.111 − Счётчик передачи

5.11.3.4 Регистр-указатель и вспомогательный регистр-указатель

Указатели определяют адрес, из которого будут загружаться новые значения регистра канала ПДП, когда выполняется автоинициализация. Когда канал опустошён (счётчик передачи = 0), он может (если соответственно сконфигурирован) использовать указатель для повторной самозагрузки. На рисунке 5.112 приведены регистры-указатели адреса сопроцессора ПДП. Значения этих регистров при сбросе не определены.

Например, при автоинициализации для загрузки регистров ПДП канала 0 необходимо следующее:

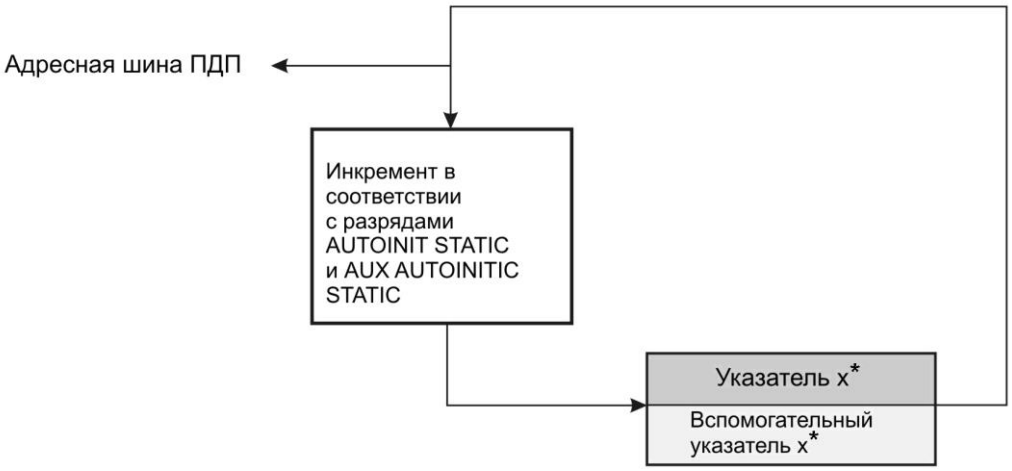
Получить значение указателя для следующей операции ПДП. Оно является адресом памяти, по которому хранится значение первого регистра ПДП канала 0 (регистр управления каналом показан на рисунке 5.109).

Взять данные из этого адреса и записать по адресу 0010 00A0h (первое слово регистра ПДП канала 0, рисунок 5.109).

Инкрементировать регистр-указатель. Если разряд AUTOINIT STATIC = 1, этот этап пропускается.

Взять следующее слово и записать по адресу 0010 00A1h.

Повторять до тех пор, пока весь блок регистров ПДП канала 0 не будет загружен (семь регистров в основном режиме, пять регистров в режиме разделения).



* x – Номер канала ПДП (0 – 5).

Рисунок 5.112 − Регистры-указатели

5.11.4 Основной режим ПДП

Основной режим – режим ПДП по умолчанию. Он используется для передачи данных из памяти в память. Для выбора основного режима необходимо сбросить разряд SPLIT MODE (разряд 14 регистра управления каналом ПДП на рисунке 5.109). Таким образом, просто запишите в него 0 (0 – значение, устанавливаемое при сбросе).

Последовательность передачи блока данных в основном режиме описана в 5.11.2.1. Арбитраж канала ПДП в основном режиме описан в 5.11.6. Синхронизация ПДП с прерываниями описана в 5.11.10. Автоинициализация в основном режиме описана в 5.11.9.1.

Основной режим ПДП передачи данных включает два этапа (см. рисунок 5.113):

Канал ПДП считывает данные из адреса, указанного регистром начального адреса, а затем сохраняет их во временный регистр.

Канал ПДП считывает значение временного регистра и записывает его по адресу, указанному регистром конечного адреса.

Вы можете использовать основной режим для обмена данными с коммуникационным портом, в основном, при однонаправленных передачах. Для двунаправленной передачи данных следует использовать режим разделения.

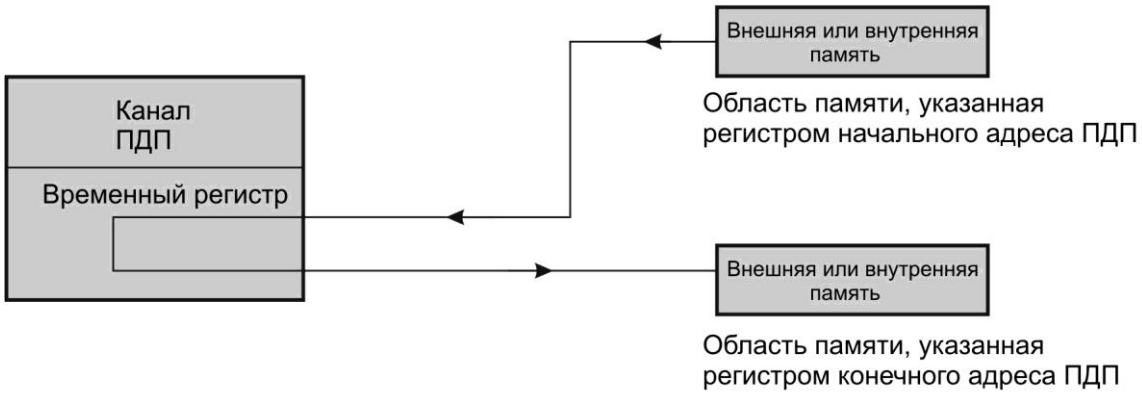


Рисунок 5.113 − Типичная конфигурация канала ПДП в основном режиме

5.11.5 Режим разделения ПДП

Режим разделения ПДП (см. рисунок 5.114) позволяет использовать один канал ПДП и для чтения, и для записи коммуникационного порта. Режим разделения преобразует один канал ПДП в два канала ПДП:

- основной канал: определяет данные чтения из области памяти (внешней/внутренней) и записывает их в выходной буфер FIFO коммуникационного порта;

- вспомогательный канал: определяет получение данных с входного буфера FIFO коммуникационного порта и записывает их в область памяти.

Для выбора режима разделения установите разряд SPLIT MODE (разряд 14 регистра управления каналом ПДП, рисунок 5.109) в 1.

Все шесть каналов ПДП поддерживают режим разделения для всех коммуникационных портов. Поле COM PORT (разряды 15 − 17, как показано на рисунке 5.109) регистра управления каналом ПДП определяет, какой коммуникационный порт будет использован (порт 0 − 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. 5.11.10). На рисунке 5.114 приведена типичная операция в режиме разделения с одним коммуникационным портом.

Передача данных в режиме разделения аналогична передаче в основном режиме, за исключением следующих отличий:

- основной канал считывает данные из адреса, указанного регистром начального адреса, и записывает их во временный регистр в пределах сопроцессора ПДП. Затем записывает значение временного регистра в выходной буфер FIFO коммуникационного порта, определённого полем COM PORT. Регистры, которые управляют основным каналом ПДП: регистр управления каналом ПДП, регистр начального адреса, индексный регистр начального адреса (его значение добавляется к значению регистра начального адреса), регистр-счётчик передачи, регистр-указатель.

- вспомогательный канал считывает слово с входного буфера FIFO коммуникационного порта, определённого полем COM PORT. И записывает его во временный регистр в пределах сопроцессора ПДП. Затем записывает значение временного регистра по адресу, указанному регистром конечного адреса. Регистры, управляющие вспомогательным каналом, – регистр управления каналом ПДП, регистр конечного адреса, индексный регистр конечного адреса (его значение добавляется к значению регистра конечного адреса), вспомогательный регистр-счётчик передачи, вспомогательный регистр-указатель.

Арбитраж канала ПДП в режиме разделения описан в 5.11.6.3. Синхронизация ПДП с прерываниями описана в 5.11.10. Автоинициализация в режиме разделения описана в 5.11.9.2.

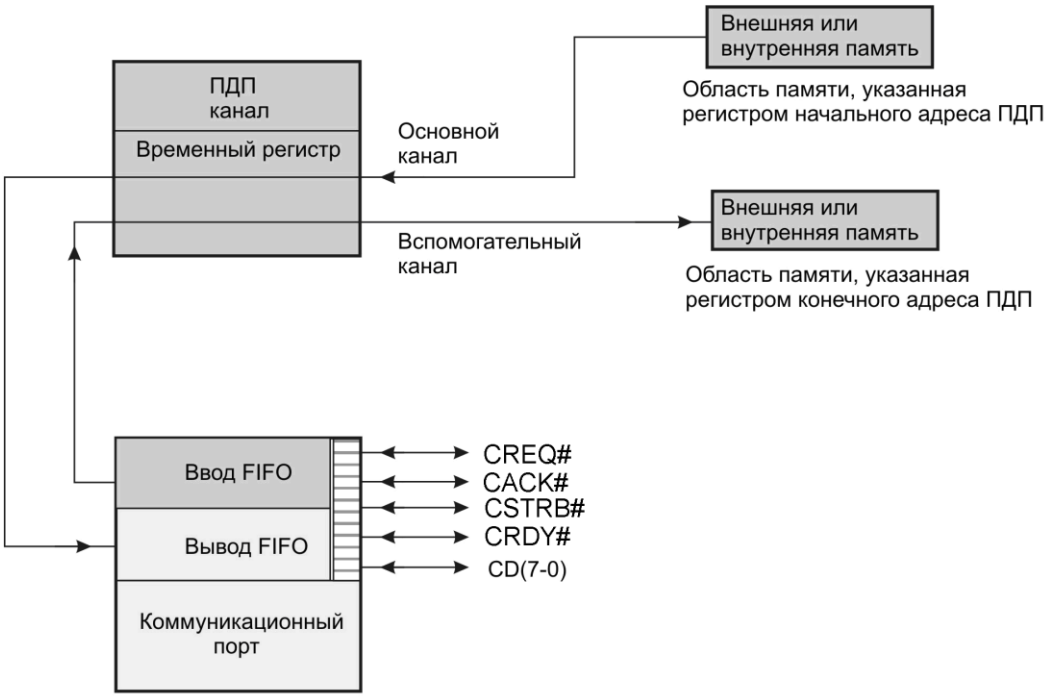


Рисунок 5.114 − Типичная конфигурация режима разделения ПДП

Примечание – Существует только один временный регистр в каждом канале ПДП. Таким образом, сначала должна завершиться операция основного канала, прежде чем начнёт работу вспомогательный канал и наоборот.

Основной и вспомогательный каналы 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 каналами.
- AUX STATUS, AUX START, AUX TCINT флаг, AUX TCC, WRITE BIT REV, SYNC MODE (разряд 7), AUX TRANSFER MODE используются только вспомогательным каналом.
- STATUS, START, TCINT флаг, READ BIT REV, SYNC MODE (разряд 6), TRANSFER MODE используются только основным каналом.

5.11.6 Схемы внутренних приоритетов ПДП

Так как все обращения шесть каналов ПДП относятся к общим внутренним шинам данных и адресов ПДП, для арбитража шин требуется схема приоритетов. В пределах сопроцессора ПДП используется две схемы приоритетов для определения, какой канал обслуживается следующим:

- схема с фиксированными приоритетами, где канал 0 имеет высший приоритет, а канал 5 – низший;
- схема циклических приоритетов, где только что обслуженный канал помещается в конец списка приоритетов (устанавливается при сбросе по умолчанию).

5.11.6.1 Схема с фиксированными приоритетами

Эта схема обеспечивает фиксированные (неизменяемые) приоритеты для каждого канала:

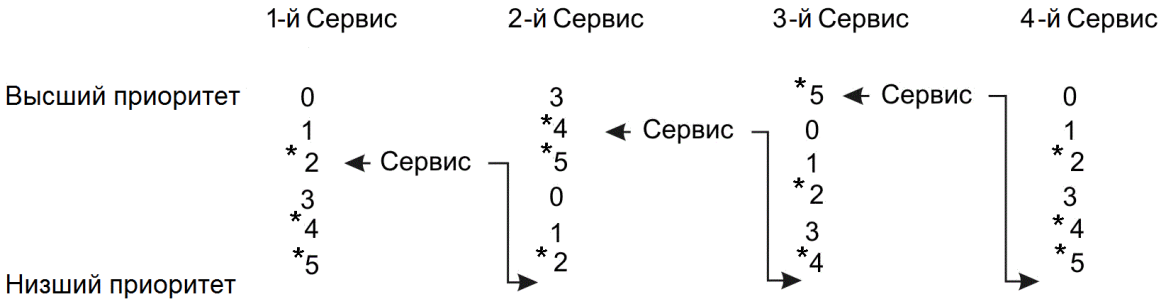
Высший приоритет 0
1
2
3
4
Низший приоритет 5

Для выбора фиксированных приоритетов установите разряд PRIORITY MODE (разряд 30) регистра управления каналом ПДП канала 0 в 1.

5.11.6.2 Схема циклических приоритетов

В схеме с циклическими приоритетами каналу, использованному последним, присваивается низший приоритет. Остальная последовательность каналов циклически сдвигается, и каналу, следующему за каналом, использованным последним, присваивается высший приоритет при следующем запросе. Приоритеты циклически изменяются по завершении каждой передачи. На рисунках 5.115 и 5.117 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 установите разряд PRIORITY MODE (разряд 30) регистра управления каналом ПДП канала 0 в 0.

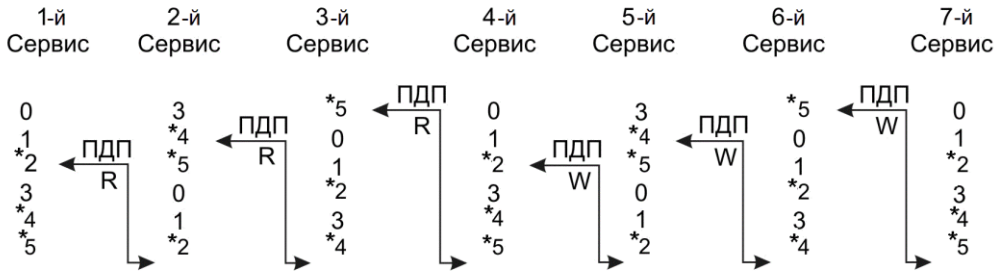


* Запрос доступа каналом ПДП.

Рисунок 5.115 − Пример схемы циклических приоритетов сопроцессора ПДП

При запуске на рисунке 5.115 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 сервисе приоритеты каналов 4 и 5 изменяются аналогично. На рисунке 5.116 приведена полная последовательность чтения и записи.

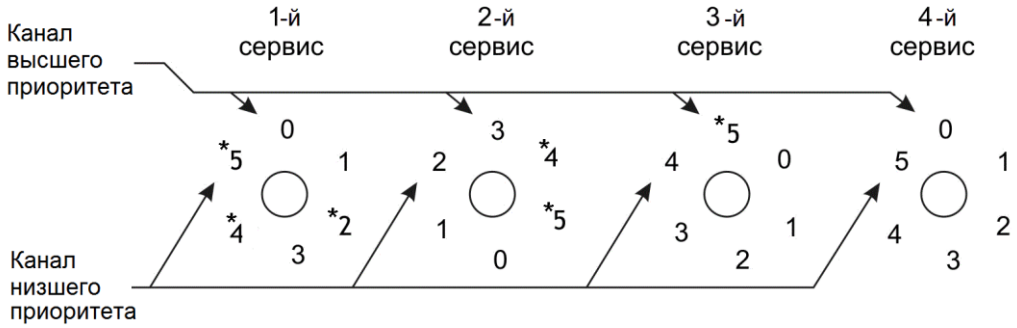
Замечание: каждый сервис означает один доступ чтения или один доступ записи. Сопроцессор ПДП устанавливает арбитраж каналов на основе «доступ за доступом»; поэтому канал ПДП должен конкурировать между доступами чтения и записи и для общего режима, и для режима разделения.



* Запрос доступа каналом ПДП

Рисунок 5.116 − Пример последовательности циклических приоритетов для чтения и записи (основной режим)

На рисунке 5.117 приведена такая же схема циклических приоритетов, как и на рисунке 5.115, но в другом представлении. Приоритеты уменьшаются от высшего к низшему по часовой стрелке. Приоритет канала, обслуженного последним, переходит по часовой стрелке и становится низшим.



* Запрос доступа каналом ПДП.

Рисунок 5.117 − Пример циклических приоритетов

В схеме циклических приоритетов запрос любого канала ПДП будет обслужен после обслуживания числа запросов с более высоким приоритетом. Максимальное число запросов:

- 5 в основном режиме;
- 11 в режиме разделения.

Это позволяет избежать монополизации системы одним каналом.

Каналы ПДП, которые запущены и не синхронизированы посредством прерываний, всегда запрашивают обслуживание (сервис).

5.11.6.3 Арбитраж канала ПДП в режиме разделения

Если канал ПДП запущен в режиме разделения, арбитраж между каналами осуществляется посредством циклических приоритетов. Канал ПДП в режиме разделения имеет такой же приоритет, как и канал ПДП в основном режиме. Единственным конфликтом может стать арбитраж между основным и вспомогательным разделёнными каналами. Изменение приоритетов разделённых каналов происходит посредством схемы циклических приоритетов.

Если канал ПДП в режиме разделения одновременно запускается посредством разрядов START и AUX START, выходной (основной) канал имеет более высокий приоритет, чем входной (вспомогательный) канал. Оба разряда START и AUX START должны записываться в одно время, чтобы достичь условия сброса.

Схема приоритетов каналов для режима разделения 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 разделения 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 5.118 показаны два канала, конкурирующие за шину ПДП: канал 2 (разделённый канал) и канал 4.

Высший приоритет 0
1
*[2pri
2aux]
3
**4
Низший приоритет 5

2pri – основной канал ПДП канала 2.
2aux – вспомогательный канал ПДП канала 2.

_____
* Запрос доступа каналами ПДП в режиме разделения.
** Запрос доступа каналом ПДП.

Рисунок 5.118 − Пример схемы приоритетов каналов в режиме разделения

Схема приоритетов канала, показанная на рисунке 5.118, последовательно представлена на рисунке 5.119. На схеме показано восемь этапов:

Первый сервис является запросом основного канала разделённого ПДП канала 2 (2pri). 2pri осуществляет чтение, а затем приоритет канала 2 становится низшим, но 2pri остаётся каналом более высокого приоритета в канале 2.

Во втором сервисе канал 4, который получает более высокий приоритет, считывает свой начальный адрес и получает низший приоритет.

В третьем сервисе значение, считанное посредством 2pri, записывается по конечному адресу, и канал 2 получает низший приоритет. Также 2pri получает более низкий приоритет, чем 2aux. Заметьте, что разделённый канал, который только завершил чтение, остаётся с более высоким приоритетом, чем другой разделённый канал до тех пор, пока данные не будут записаны по конечному адресу.

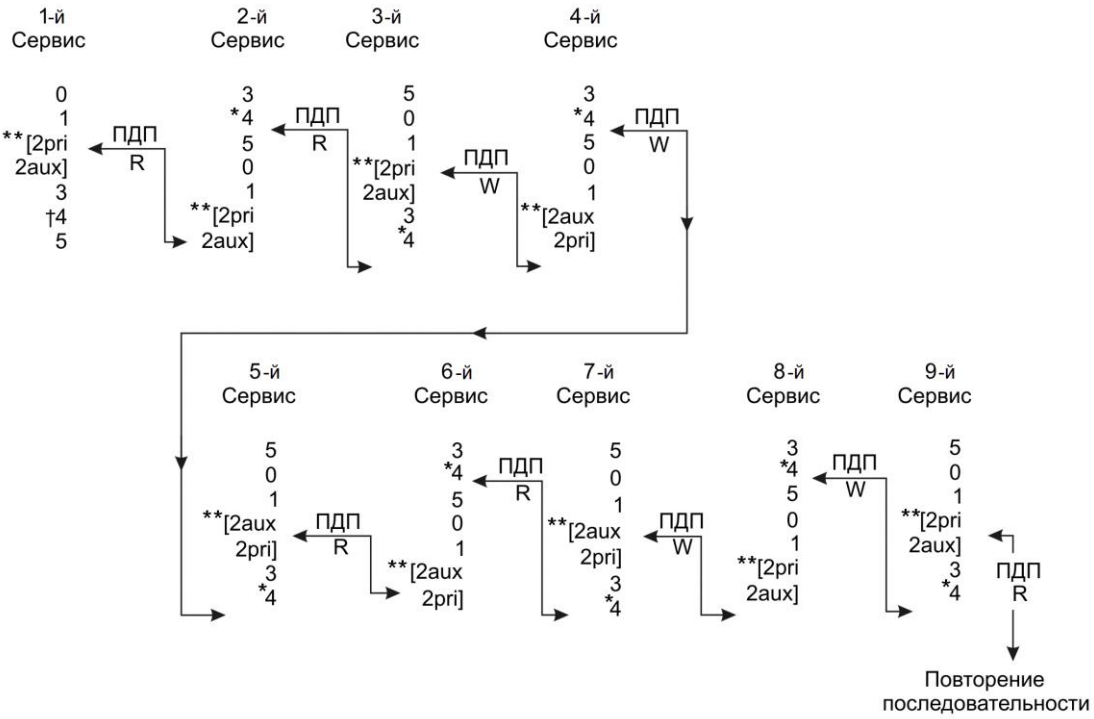
В четвёртом сервисе значение, считанное каналом 4 во втором сервисе, теперь записывается по его конечному адресу, и канал получает низший приоритет.

В пятом сервисе 2aux выполняет чтение, и канал 2 получает низший приоритет.

В шестом сервисе канал 4 снова выполняет чтение и получает низший приоритет.

В седьмом и восьмом сервисе значения 2aux и канала 4, которые были прочитаны в сервисах 5 и 6, теперь записываются по их конечному адресу. После записи канала, он получает низший приоритет.

В девятом сервисе 2pri снова выполняет чтение, как и в пятом сервисе, и циклы чтения/записи продолжаются, начиная с первого сервиса.



* Запрос доступа каналом ПДП.

** Запрос доступа каналами ПДП в режиме разделения.

2pri – Основной канал ПДП канала 2.

2aux – Вспомогательный канал ПДП канала 2.

Рисунок 5.119 − Пример сервисной последовательности смены приоритетов в режиме разделения

5.11.7 Арбитраж ЦПУ и сопроцессора ПДП

Сопроцессор ПДП передаёт данные по своим внутренним шинам. Арбитраж необходим только тогда, когда возникает конфликт при обращении к источнику данных ЦПУ и сопроцессором ПДП. Арбитраж не должен иметь задержки. Если конфликт отсутствует, ЦПУ и сопроцессор ПДП осуществляют доступы параллельно.

Все варианты арбитража между ЦПУ и сопроцессором ПДП построены на базе доступа, таким образом, сопроцессор ПДП должен конкурировать за доступы чтения/записи в основном режиме и режиме разделения. Доступ сопроцессора ПДП к внутренней памяти начинается в течение H3 (смотри 5.8.5 для более подробной информации).

Если ЦПУ и сопроцессор ПДП обращаются к одному и тому же источнику данных, разряды DMA PRI (разряды 0 и 1 регистра управления каналом) определяют правила арбитража (как показано в таблице 5.42). 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.

5.11.8 Режимы передачи данных

Каждый канал ПДП может работать в четырёх режимах передачи данных. Они отличаются друг от друга следующим:

- используется ли автоинициализация или нет;
- как работают, в зависимости от того, включена автоинициализация или нет.

Таблица 5.43 и следующие за ней разделы описывают режимы передачи данных.

Таблица 5.42 − Разряды DMA PRI и правила арбитража ЦПУ/ПДП

Разряды DMA PRI 1−0	Описание
0 0	Доступ сопроцессора ПДП имеет приоритет ниже приоритета доступа ЦПУ. Если канал ПДП и ЦПУ обращаются к одному ресурсу памяти, работает ЦПУ. Эти разряды выставляются так при сбросе
0 1	Эти установки выбирают 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. Сначала канал ПДП и ЦПУ запрашивают один и тот же ресурс памяти, ЦПУ имеет приоритет. Если в следующем цикле инструкций сопроцессор ПДП и ЦПУ снова обращаются к одному и тому ресурсу памяти, 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, ЦПУ имеет приоритет
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, работает ПДП

Таблица 5.43 − Описание поля TRANSFER MODE (AUX TRANSFER MODE)

Разряды TRANSFER MODE 3−2/(5−4)	Описание
0 0	Передача данных не прерывается счётчиком передачи, и не выполняется автоинициализация. TCINT (прерывание счётчика передачи) и AUX TCINT при этом могут использоваться для обозначения прерывания, когда счётчик передачи переходит через 0. Канал ПДП продолжает работу
0 1	Передачи данных отменяются счётчиком передачи. Автоинициализация не выполняется. Код останова 10_2 помещается в поле START (или AUX START), когда передача данных завершена
1 0	Автоинициализация 1. Автоинициализация выполняется, когда счётчик 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 счётчик передачи обращается в 0, операция останавливается до тех пор, пока ЦПУ запустит сопроцессор ПДП, используя поле START (AUX START) в регистре управления каналом ПДП (разряды 22−23 и 24−25 регистра управления каналом ПДП). Код останова 10_2 помещается в поле START (или AUX START) сопроцессором ПДП

5.11.8.1 Работа в режиме TRANSFER MODE = 00_2

Когда TRANSFER MODE = 00_2, передача данных не прекращается при обнулении счётчика передачи, и автоинициализация не выполняется. Хотя счётчик передачи не останавливает передачу данных, прерывание может быть сгенерировано при переходе счётчика передач через 0, путём установки разряда TCINT FLAG в 1. Если канал сопроцессора ПДП не был остановлен после достижения счётчиком передачи 0, счётчик продолжит декрементироваться ниже 0.

5.11.8.2 Работа в режиме TRANSFER MODE = 01_2

Когда TRANSFER MODE = 01_2, передача данных прекращается при обнулении счётчика передачи, и автоинициализация не выполняется. Когда счётчик передачи достигает 0, канал ПДП останавливается посредством записи 10_2 в поле START или AUX START.

5.11.8.3 Работа в режиме TRANSFER MODE = 10_2 (Автоинициализация 1)

Этот режим передачи позволяет каналу ПДП работать непрерывно, изменять указатели и синхронизацию процедурой автоинициализации, самостоятельно отключать её. Поддерживаются два различных метода автоинициализации:

Метод автоинициализации 1а всегда запускается после системного сброса, после сброса канала ПДП (00_2 записывается в разряды START или AUX START), после останова канала ПДП (01_2 или 10_2 записывается в разряды START или AUX START). Для выбора режима передачи 10_2 (метод автоинициализации 1а) необходимо (см. рисунок 5.120):

- установить регистр управления каналом ПДП в режим передачи 10_2 и сбросить или остановить канал ПДП для автоинициализации;
- установить счётчик 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 11_2 в разряды START или AUX START;
- канал ПДП выполняет последовательность автоинициализации и передачи блока данных.

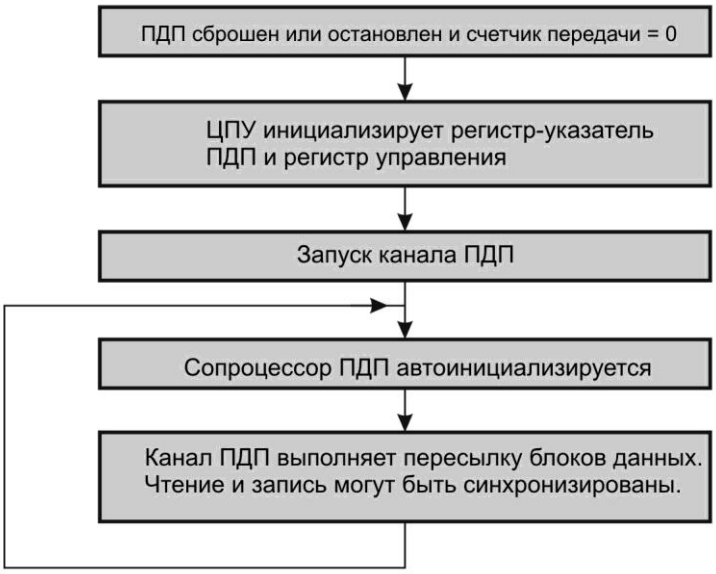


Рисунок 5.120 − Работа канала ПДП в режиме TRANSFER MODE = 10_2 (метод автоинициализации 1а)

Метод автоинициализации 1б запускается, когда счётчик передачи не равен нулю. ПДП запускает передачу данных и автоинициализируется по завершении операции передачи (когда счётчик передачи переходит в 0). Для выбора режима передачи 10_2 (метод автоинициализации 1б) необходимо следующее (смотри рисунок 5.121):

- 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, счётчик передачи и т. д.) в соответствии с желаемой операцией передачи. Следует отметить, что счётчик 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 11_2 в разряды START или AUX START;

- канал ПДП выполняет последовательность передачи блока данных и автоинициализации (обратный порядок в отличие от метода 1а).

Заметьте, что если канал ПДП запрограммирован для выполнения передачи n блоков данных, метод автоинициализации 1а требует n значений автоинициализации ПДП. Метод автоинициализации 1б требует n–1 значение автоинициализации ПДП, так как первая передача данных выполняется в течение инициализации передачи ПДП. Это означает сохранение некоторой памяти, но последующие аналогичные операции ПДП требуют дополнительных циклов для установки значений регистров ПДП снова.

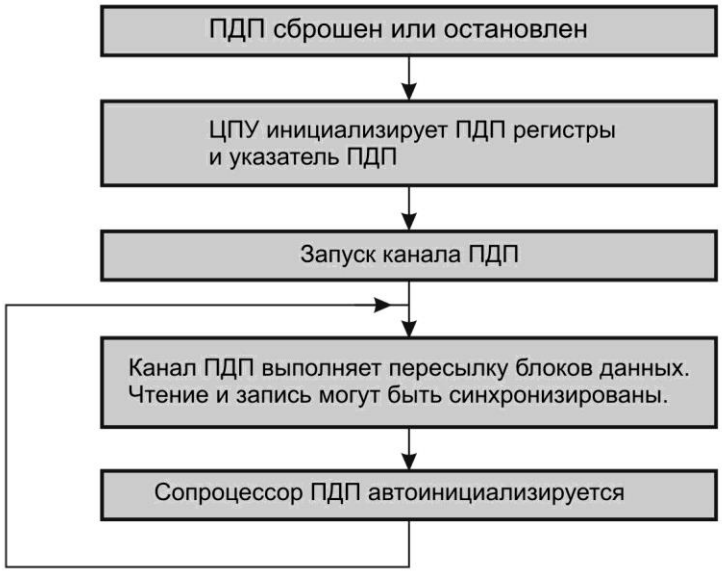


Рисунок 5.121 − Работа канала ПДП в режиме TRANSFER MODE = 10_2 (метод автоинициализации 1б)

5.11.8.4 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 разряды START или AUX START), после останова канала ПДП (01_2 или 10_2 записывается в разряды START или AUX START). Для выбора режима передачи 11_2 (метод автоинициализации 2а) необходимо (см. рисунок 5.122):

- установить регистр управления каналом ПДП в режим передачи 11_2 и сбросить или остановить канал ПДП для автоинициализации;

- установить счётчик 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 11_2 в разряды START или AUX START;

- канал ПДП автоинициализируется и выполняет передачу блока данных;

- когда счётчик передачи обнуляется, ПДП будет ожидать пока ЦПУ запишет 11_2 в поле START (или AUX START) регистра управления каналом ПДП и автоинициализируется;

- повторение последовательности автоинициализации, передачи, ожидания;

- когда счётчик передачи достигнет 0, можно остановить канал ПДП путём записи 10_2 в поле START (или AUX START).

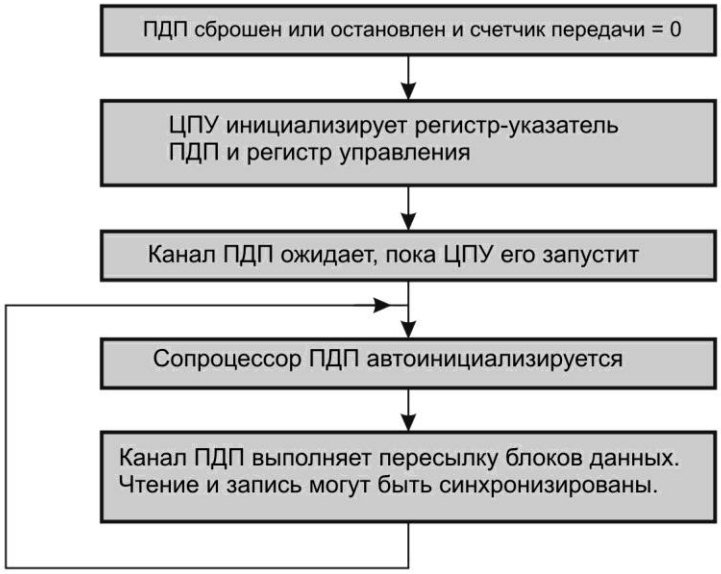


Рисунок 5.122 − Работа канала ПДП в режиме TRANSFER MODE = 11_2 (метод автоинициализации 2а)

Автоинициализация – это метод перезагрузки регистрового файла канала ПДП при обнулении счётчика передачи. Когда канал ПДП функционирует в режиме автоинициализации, 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 5.108. Регистры-указатели описаны в 5.11.3.4.

Автоинициализация – это постоянная операция передачи блока данных, при которой конечным адресом является регистровый файл сопроцессора ПДП. ПДП считывает значение из адреса, указанного регистром-указателем, и записывает его в регистр ПДП через периферийную шину в следующем доступном цикле. Соответственно, автоинициализация доступов чтения/записи также является предметом конфликта доступов ЦПУ/ПДП.

Автоинициализация может произойти:

- без вмешательства ЦПУ, если биты TRANSFER MODE = 10_2 (автоинициализация 1) (см. 5.11.8.3);
- с вмешательством ЦПУ, если биты TRANSFER MODE = 11_2 (автоинициализация 2). В этом случае ЦПУ должен перезапустить канал ПДП до выполнения автоинициализации (см. 5.11.8.4);
- до передачи блока данных (метод автоинициализации 1а). ПДП запускается, когда счётчик передачи равен 0, затем инициализируется и выполняет передачу блока данных;
- после передачи блока данных (метод автоинициализации 1б). ПДП запускается, начиная с постоянного блока передачи, а затем, после обнуления счётчика передачи, автоинициализируется.

Автоинициализации 1 или 2 могут использовать методы а или б.

Автоинициализация зависит от текущего режима канала ПДП: основного режима или режима разделения. Режим операции управляется разрядом SPLIT MODE (разряд 14 на рисунке 5.109). Во время автоинициализации сопроцессора ПДП, не изменяйте разряд SPLIT MODE. Этот разряд следует изменять только тогда, когда сопроцессор ПДП сброшен или остановлен (смотри описание разряда ПДП START в таблице 5.40 для более подробной информации).

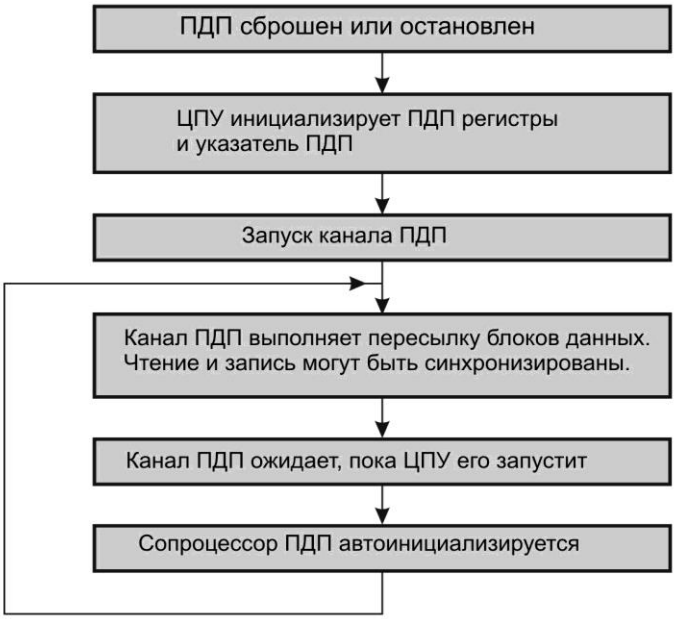


Рисунок 5.123 − Работа канала ПДП в режиме TRANSFER MODE = 11_2
(метод автоинициализации 2б)

5.11.9 Автоинициализация

Автоинициализация – это метод перезагрузки регистрового файла канала ПДП, когда счётчик передачи обращается в 0. Когда канал ПДП оперирует в режиме автоинициализации, 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 5.108. Регистры-указатели описаны в 5.11.3.4.

Автоинициализация – это постоянная операция передачи блока данных, при которой конечным адресом является регистровый файл сопроцессора ПДП. ПДП считывает значение из адреса, указанного регистром-указателем, и записывает его в регистр ПДП через периферийную шину в следующем доступном цикле. Соответственно, автоинициализация доступов чтения/записи также является предметом конфликтов доступов ЦПУ/ПДП.

Автоинициализация может произойти:

- без вмешательства ЦПУ, если разряды TRANSFER MODE = 10_2 (автоинициализация 1) (см. 5.11.8.3);
- с вмешательством ЦПУ, если разряды TRANSFER MODE = 11_2 (автоинициализация 2). В этом случае ЦПУ должен перезапустить канал ПДП до выполнения автоинициализации (см. 5.11.8.4);
- до передачи блока данных (метод автоинициализации 1а). ПДП запускается, когда счётчик передачи равен 0, затем инициализируется и выполняет передачу блока данных;
- после передачи блока данных (метод автоинициализации 1б). ПДП запускается, начиная с постоянного блока передачи, а затем, после обнуления счётчика передачи, автоинициализируется.

Автоинициализации 1 или 2 могут использовать методы а или б.

Автоинициализация зависит от текущего режима канала ПДП: основного режима или режима разделения. Режим операции управляется разрядом SPLIT MODE (разряд 14 на рисунке 5.109). Во время автоинициализации сопроцессора ПДП, не изменяйте разряд SPLIT MODE. Этот разряд следует изменять только тогда, когда сопроцессор ПДП сброшен или остановлен (смотри описание разряда ПДП START в таблице 5.40 для более подробной информации).

5.11.9.1 Основной режим

Если канал ПДП запущен в основном режиме (SPLIT MODE = 0), используется регистр-указатель, и регистры канала ПДП загружаются в следующем порядке:

- регистр управления каналом ПДП;
- регистр начального адреса;
- индексный регистр начального адреса;
- регистр-счётчик повторения;
- регистр конечного адреса;
- индексный регистр конечного адреса;
- регистр-указатель.

Размещение новых значений этих регистров в памяти показано на рисунке 5.124.

Рисунок 5.124 − Размещение новых значений регистров канала ПДП в памяти (SPLIT MODE = 0)

5.11.9.2 Режим разделения

Если канал ПДП запущен в режиме разделения (SPLIT MODE = 1), последовательность автоинициализации зависит от того, какой счётчик передачи будет остановлен.

Если счётчик передачи переходит в 0 при SPLIT MODE = 1, 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-счётчик передачи;
- регистр-указатель.

Размещение новых значений этих регистров в памяти показано на рисунке 5.125.

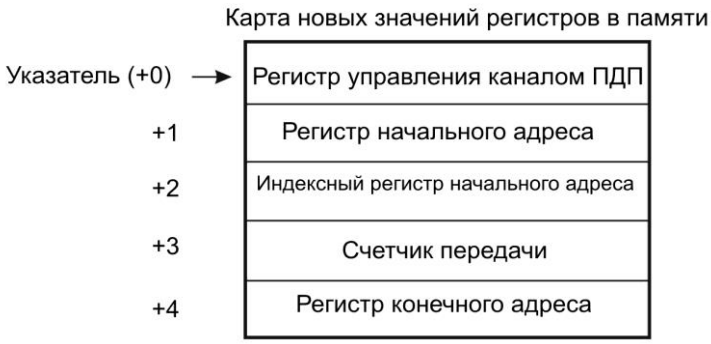


Рисунок 5.125 − Размещение новых значений регистров канала ПДП в памяти (SPLIT MODE = 1 и счётчик передачи = 0)

Если вспомогательный счётчик передачи переходит в 0 при SPLIT MODE = 1, для автоинициализации используется вспомогательный регистр-указатель. В этом случае регистры ПДП загружаются в следующем порядке:

- регистр управления каналом ПДП;
- регистр конечного адреса;
- индексный регистр конечного адреса;
- вспомогательный счётчик передачи;
- вспомогательный регистр-указатель.

Размещение новых значений этих регистров в памяти показано на рисунке 5.126.

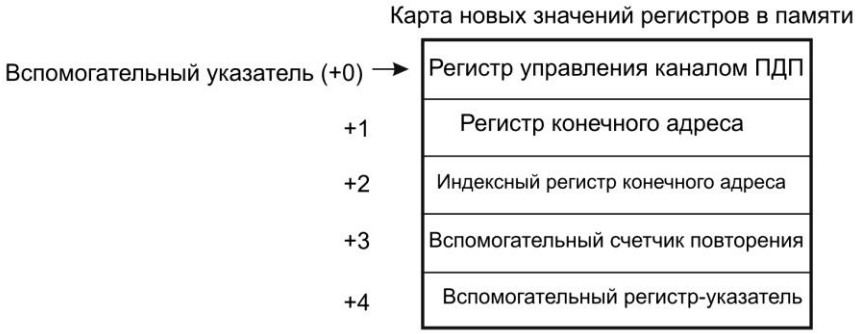


Рисунок 5.126 − Размещение новых значений регистров канала ПДП в памяти (SPLIT MODE = 1 и вспомогательный счётчик передачи = 0)

5.11.9.3 Инкрементирование регистра-указателя

Во время автоинициализации регистр-указатель может инкрементироваться или оставаться неизменным:

Если указатель инкрементируется, значения автоинициализации сохраняются последовательно в области памяти, и регистр-указатель и вспомогательный регистр-указатель инкрементируются в порядке доступа к областям памяти.

Если канал ПДП автоинициализируется посредством устройства с пакетной передачей данных, например, внутреннего коммуникационного порта или внешних FIFO, значение регистра-указателя следует удерживать постоянным.

Инкрементирование управляется битами регистра управления каналом ПДП AUTOINIT STATIC и AUX AUTOINIT STATIC:

В основном режиме разряд AUTOINIT STATIC управляет указателем.

В режиме разделения AUTOINIT STATIC управляет указателем (основного канала) и AUX AUTOINIT STATIC управляет вспомогательным указателем.

Если разряд AUTOINIT STATIC (AUX AUTOINIT STATIC) равен 0, регистр-указатель инкрементируется, если равен 1, значение регистра-указателя не изменяется.

5.11.9.4 Синхронизация

Обычно данные автоинициализации хранятся в памяти, и автоинициализация не является необходимостью. В некоторых случаях вам может потребоваться передать данные автоинициализации таким же образом, как синхронизованные данные чтения/записи.

Синхронизация автоинициализации – это функция:

- разрядов SYNC MODE (разряды 6 и 7 регистра управления каналом ПДП), которые управляют синхронизацией передачи данных;
- разрядов AUTOINIT SYNC (разряды 10 и 11 регистра управления каналом ПДП), которые влияют только на синхронизацию автоинициализации.

Если разряды SYNC MODE не установлены для синхронизации передачи данных (т. е. если предшествующая передача данных не синхронизирована прерываниями), последовательность автоинициализации канала ПДП также не синхронизируется. Если разряды SYNC MODE устанавливаются синхронно для 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 5.44. Заметим, что когда оба режима устанавливают «нет синхронизации», последовательность автоинициализации канала ПДП не синхронизируется прерываниями.

В основном режиме синхронизация записи для операции автоинициализации отсутствует, так как конечным адресом является регистр ПДП, который уже готов.

В режиме разделения разряд 6 регистра управления каналом ПДП управляет синхронизацией автоинициализации основного канала ПДП, а разряд 7 управляет синхронизацией автоинициализации вспомогательного канала ПДП.

Если используется синхронизация автоинициализации основного канала, чтение ПДП значений автоинициализации, хранящихся в памяти, не производится до тех пор, пока не придёт прерывание, определённое полем записи основного канала в регистре DIE.

Если используется синхронизация автоинициализации вспомогательного канала, чтение ПДП значений автоинициализации, хранящихся в памяти, не производится до тех пор, пока не придёт прерывание, определённое полем чтения вспомогательного канала в регистре DIE.

Таблица 5.44 − Влияние разрядов SYNC MODE и AUTOINIT MODE при автоинициализации

SYNC MODE Разряды 7−6	AUTOINIT SYNC Разряды 11−10	Основной режим	Режим разделения
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

5.11.9.5 Влияние разрядов регистра управления ПДП

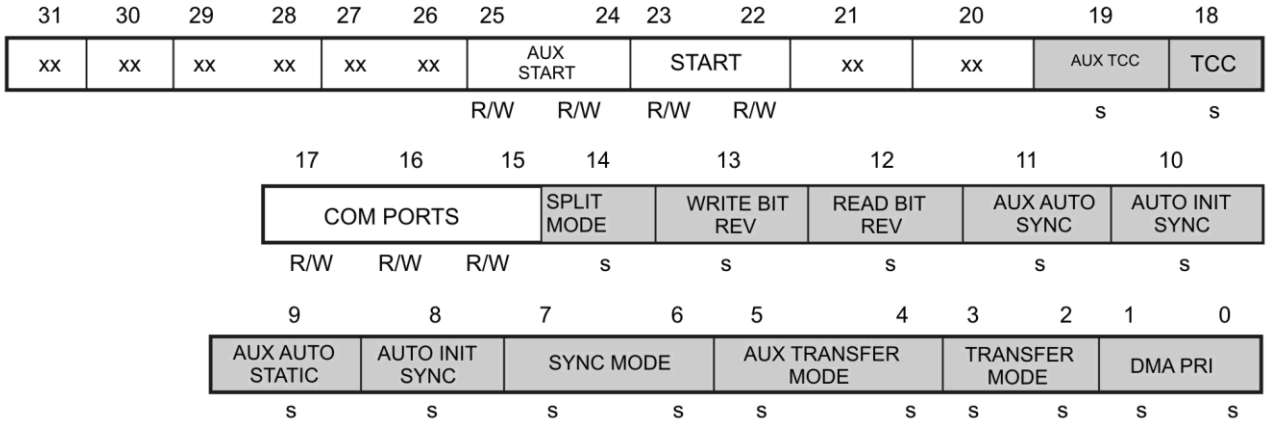
В основном режиме все записываемые разряды регистра управления управляются автоинициализацией. Эти разряды приведены на рисунке 5.127.

В режиме разделения во время автоинициализации основного канала ПДП, записываемые невспомогательные биты могут быть изменены, а вспомогательные разряды защищены от записи (см. рисунок 5.128). Другими словами, посредством ЦПУ или сопроцессора ПДП изменяются только невспомогательные разряды. Также, если автоинициализирован вспомогательный канал ПДП, могут быть изменены вспомогательные записываемые разряды, а невспомогательные разряды защищены от записи. Эти разряды приведены на рисунке 5.129.

Хотя затенённые разряды (определённые s на рисунке 5.127) изменяются в течение автоинициализации, изменения вступают в силу только после завершения автоинициализации. Незатенённые разряды вступают в силу немедленно, воздействуя на последовательность автоинициализации. Другими словами, при автоинициализации новые значения затенённых разрядов вводятся последними после загрузки всех регистров (определённых регистром-указателем).

Несмотря на то, запущен ли канал ПДП в основном режиме или режиме ПДП, 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 затенённые.

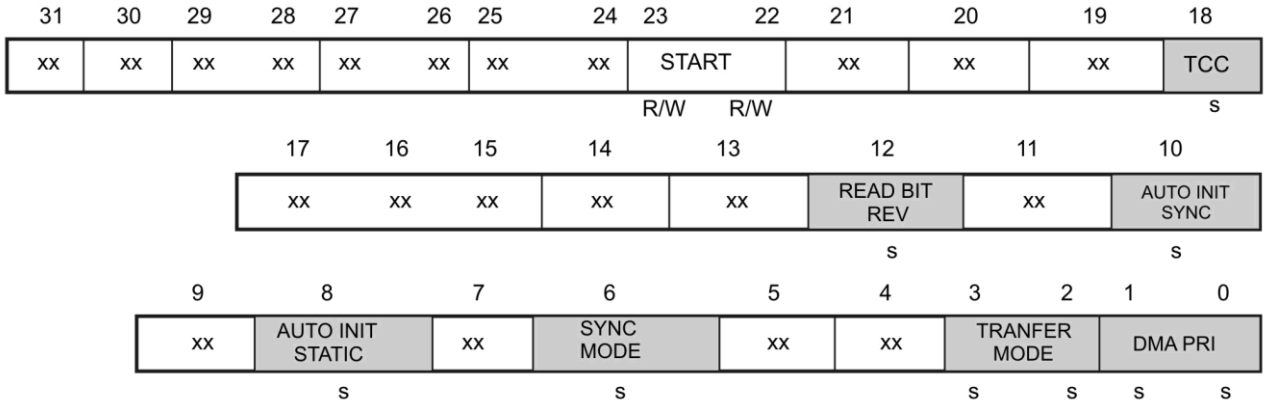
Примечание − Если ЦПУ пишет в регистр управления ПДП в течение автоинициализации, запись ЦПУ вступает в силу только после завершения последовательности автоинициализации. Хотя операция автоинициализации регистров ПДП не подвергается воздействию, последовательность данных передачи может быть изменена.



Примечание – Принятые условные обозначения:

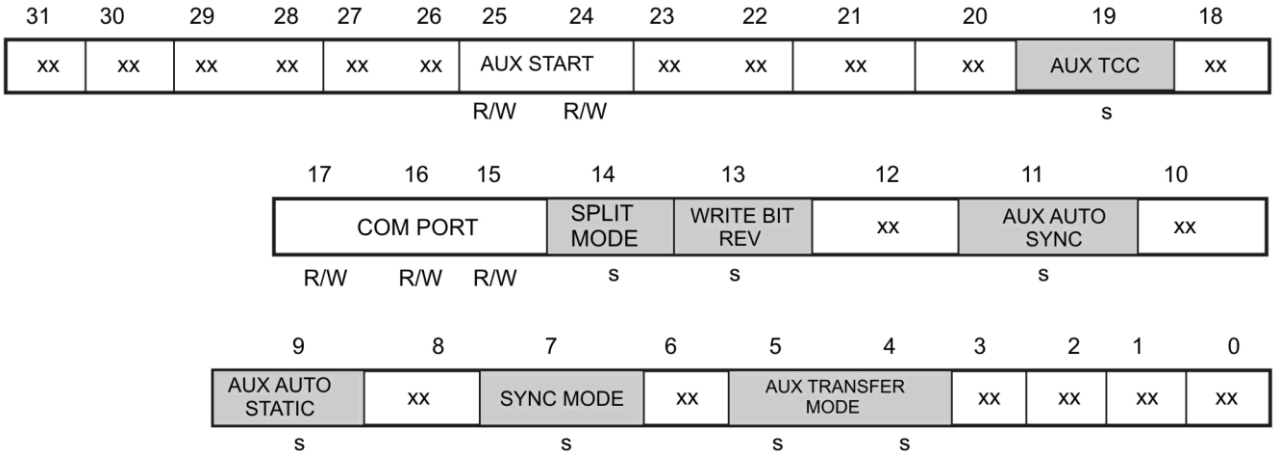
- s – затенённые биты не вступают в силу, пока автоинициализация не завершена.
- xx – защищённый от записи в течение автоинициализации.

Рисунок 5.127 − Разряды регистра управления каналом ПДП, изменяемые посредством автоинициализации в основном режиме



Примечание – Принятые условные обозначения:
- s – затенённые биты не вступают в силу, пока автоинициализация не завершена.
- xx – защищённый от записи в течение автоинициализации.

Рисунок 5.128 − Разряды регистра управления каналом ПДП, изменяемые посредством автоинициализации основного канала в режиме разделения



Примечание – Принятые условные обозначения:
- s – затенённые биты не вступают в силу, пока автоинициализация не завершена.
- xx – защищённый от записи в течение автоинициализации.

Рисунок 5.129 − Разряды регистра управления каналом ПДП, изменяемые посредством автоинициализации вспомогательного канала в режиме разделения

5.11.9.6 Последовательные автоинициализации

Для многих приложений достаточно каждый раз 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 5.130. В этом отдельном примере предполагается, что канал не запущен в режиме разделения.

При необходимости можно сделать новый указатель ссылки на новый набор значений регистров, как показано на рисунке 5.131. Это может быть продолжено на любом уровне.

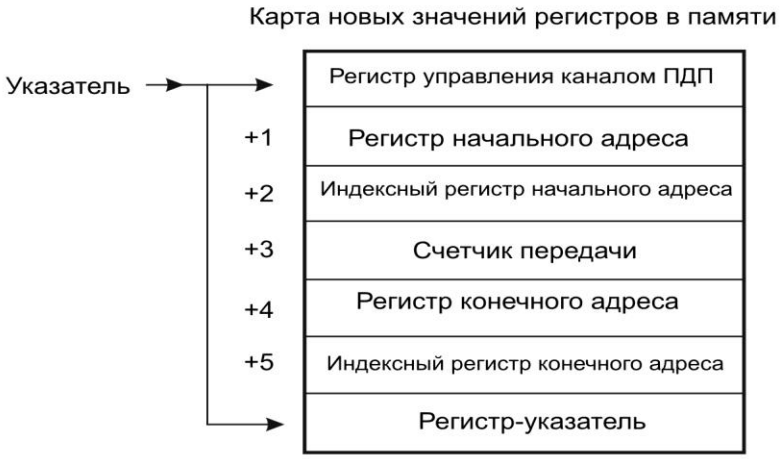


Рисунок 5.130 − Самоотносимый регистр-указатель

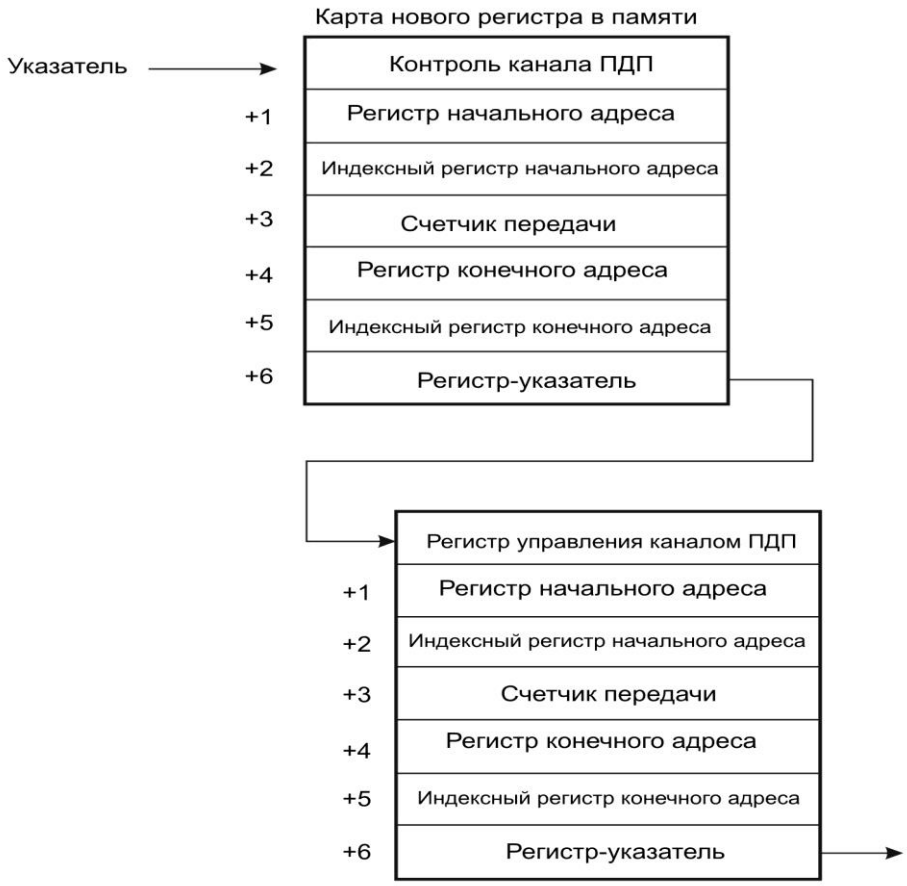


Рисунок 5.131 − Ссылка на новый регистр-указатель

5.11.10 ПДП и прерывания

Сопроцессор ПДП может использовать прерывания следующим образом:

- посылать сигналы прерываний к ЦПУ, когда завершается передача блока данных;
- принимать прерываний от внешних сигналов (IIOF(3 – 0)#), таймеров, коммуникационных портов (ICRDY, OCRDY).

В этом разделе описано, как ПДП принимает прерывания. Этот процесс называется синхронизацией. Все прерывания сопроцессора ПДП, в первую очередь, приходят на контроллер прерываний ЦПУ. Прерывания по фронту фиксируются ЦПУ установкой соответствующего флага; прерывания по уровню не фиксируются.

Когда внешние прерывания (IIOF(3 – 0)_x#) используются для синхронизации передачи данных сопроцессором ПДП, ЦПУ отвечает за конфигурирование внешних прерываний, как по фронту, так и по уровню (в зависимости от разрядов FUNCx и TYPEx регистра флагов прерываний, описанного в 5.3.1.10).

Прерывания по фронту, следующие: прерывания таймеров, прерывания ПДП, внешние прерывания, которые сконфигурированы как прерывания по фронту. Подробнее описано в 5.7.4, 5.7.6. Когда контроллер прерываний определяет, что прерывание по фронту, которое ожидает канал ПДП (разряды регистра DIE установлены в 1), зафиксировано регистром флагов, 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, фиксированное прерывание сбрасывается сопроцессором ПДП за два 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 (разряды регистра DIE установлены в 1), получено, ЦПУ посылает импульс прерывания к каналу ПДП. Канал ПДП локально фиксирует прерывание до тех пор, пока оно не будет обслужено. В то же время, фиксированное прерывание сбрасывается сопроцессором ПДП за два 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, канал ПДП с более низким приоритетом будет обслуживаться в промежутках времени между обслуживаниями высокоприоритетного канала.

В отличие от 1867ВЦ3Ф, на работу ядра процессора 1867ВЦ8Ф1 процесс обработки прерываний не влияет, даже когда выборка конвейера остановлена. Когда прерывания разрешены в регистре DIE, прерывания фиксируются автоматически контроллером прерываний ПДП и сохраняются для дальнейшего использования посредством ПДП. Когда флаг прерывания (таймера, внешнего прерывания) зафиксирован, IIF флаг сбрасывается. Заметьте, что IIF флаги сбрасываются только тогда, когда контроллер прерываний ЦПУ фиксирует прерывание, но не когда ПДП отвечает на него. Даже если ПДП не был запущен, фиксирование прерываний происходит, за исключением случая, когда разряды запуска в регистре управления ПДП имеют значения сброса (START или AUX START установлены в 00_2). Сброс ПДП сбрасывает фиксированные прерывания. Во избежание потери ранее полученных прерываний рекомендуется инициализировать регистр DIE после запуска ПДП, когда разряды запуска ПДП имеют значение 11_2. Заметьте, что когда ПДП завершает передачу данных, разряды запуска (AUX START) устанавливаются в 10_2. Поэтому ПДП не пропустит какое-либо прерывание между передачами данных.

ПДП и ЦПУ могут отвечать на одно и то же прерывание, если ЦПУ не связано с конфликтом конвейера или с какой-либо инструкцией, которая останавливает выборку инструкций. Смотри 5.7.4.1 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 начального и конечного адреса, циклы чтения и записи разрешаются одним сигналом прерывания.

Внутренний конвейер ядра процессора 1867ВЦ8Ф1 гарантирует корректное взаимодействие между коммуникационным портом, который генерирует прерывания по уровню, и каналом ПДП, который синхронизируется теми же прерываниями по уровню.

Замечание: когда вы синхронизируете каналы ПДП по внешним прерываниям, лучше конфигурировать сигналы прерываний в качестве сигналов прерываний по фронту, чтобы быть уверенным, что прерывание будет распознано только как одно.

5.11.10.1 Прерывания и синхронизация каналов ПДП

При использовании прерывания для синхронизации передачи канала ПДП для установки синхронного режима передачи требуется следующее:

Установить разряды SYNC MODE (разряды 6, 7) регистра управления каналом ПДП в соответствии с желаемой синхронизацией источника данных и приёмника данных. Смотри 5.11.10.2 для получения более подробной информации.

Установить регистр DIE для разрешения соответствующего прерывания для желаемой синхронной передачи. На рисунках 5.132 и 5.133 показан регистр DIE для основного режима и режима разделения соответственно. В таблицах 5.45 и 5.46 представлены различные прерывания синхронизации для основного режима, а в таблицах 5.47 и 5.48 – для режима разделения.

Рекомендуется инициализировать регистр DIE после запуска ПДП, когда разряды запуска принимают значение 11_2. Это предотвратит потерю ранее полученных прерываний, которая может произойти, если вы включите регистр DIE, когда разряды запуска сброшены в 00_2 (значение при сбросе).

Примечание – R – чтение, W – запись.

Рисунок 5.132 − Функции разрядов регистра DIE для основного режима ПДП

Таблица 5.45 − Синхронизация прерываний ПДП каналов 0 и 1 (ПДП0 и ПДП1) для основного режима

Значение разряда (в ПДП0 или ПДП1)	Прерывание разрешено для ПДП0 или ПДП1				Источник прерываний для синхронизации ПДП
	ПДП0 чтение	ПДП0 запись	ПДП1 чтение	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 IIOF0# – IIOF3#
1 1	TIM0	TIM0	TIM0	TIM0	От таймера TIM0

* Канал ПДП остановлен (операции чтения или записи не выполняются), если используется синхронная передача данных.

Таблица 5.46 − Синхронизация прерываний ПДП каналов от 2 до 5 (от ПДП2 до ПДП5) для основного режима

Значение разряда (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 IIOF0# – IIOF3#
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	

* x в ПДПх обозначает номер канала ПДП, а также номер соответствующего прерывания ICRDYx и OCRDYx. Например, если и ПДП2 чтение, и ПДП5 запись содержат 001_2, то разрешаются прерывания ICRDY2 и OCRDY2 соответственно. Остальные значения разрядов (010_2 и 111_2) такие же (как показано в таблице) для каналов от ПДП2 до ПДП5.

** Канал ПДП остановлен (операции чтения или записи не выполняются), если используется синхронная передача данных.

Примечание – R – чтение, W – запись.

Рисунок 5.133 − Функции разрядов регистра DIE для режима разделения ПДП

Таблица 5.47 − Синхронизация прерываний ПДП каналов 0 и 1 (ПДП0 и ПДП1) для режима разделения

Значение разряда (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 IIOF0# – IIOF3#
1 1	TIM0	TIM0	TIM0	TIM0	От таймера TIM0

* Канал ПДП остановлен (операции чтения или записи не выполняются), если используется синхронная передача данных.

Таблица 5.48 − Синхронизация прерываний ПДП каналов от 2 до 5 (от ПДП2 до ПДП5) для режима разделения

Значение разряда (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 IIOF0# – IIOF3#
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	

* x 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) для каналов от ПДП2 до ПДП5.

** Канал ПДП остановлен (операции чтения или записи не выполняются), если используется синхронная передача данных.

5.11.10.2 Разряды режима синхронизации

Таблицы 5.38 и 5.39 показывают, как значение поля SYNC MODE регистра управления каналом ПДП определяет синхронизацию в основном режиме и режиме разделения, 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 приёмника данных:
  - для основного режима (SYNC MODE = 10_2);
  - для режима разделения (SYNC MODE = 01_2);
- синхронизация источника и приёмника данных (SYNC MODE=11_2).

Когда ядро процессора 1867ВЦ8Ф1 в режиме разделения, основной канал поддерживает только синхронизацию записи (приёмника) передаваемых данных, вспомогательный канал поддерживает только синхронизацию чтения (источника) передаваемых данных. В режиме разделения разряды 6 и 7 регистра управления каналом ПДП (см. таблицу 5.33) используются для управления синхронизацией канала:

- разряд 6 управляет синхронизацией записи основного канала (синхронизация приёмника);
- разряд 7 управляет синхронизацией чтения вспомогательного канала (синхронизация источника).

Скорость передачи данных ПДП в режиме синхронизации описана в 5.11.11.2.

Нет синхронизации

Когда SYNC MODE = 00_2, синхронизация не выполняется. ПДП выполняет циклы чтения и записи, пока имеет приоритет для использования шины ПДП. Все прерывания игнорируются. Заметьте, что есть разница между этим режимом и режимом, когда в поля чтения или записи регистра DIE записаны нули. Когда в полях чтения/записи регистра DIE записаны нули, происходит полная остановка ПДП, если используется синхронизация, однако, при оставлении SYNC MODE = 00_2, канал ПДП свободно работает. Рисунок 5.134 иллюстрирует механизм, используемый при SYNC MODE = 00_2.

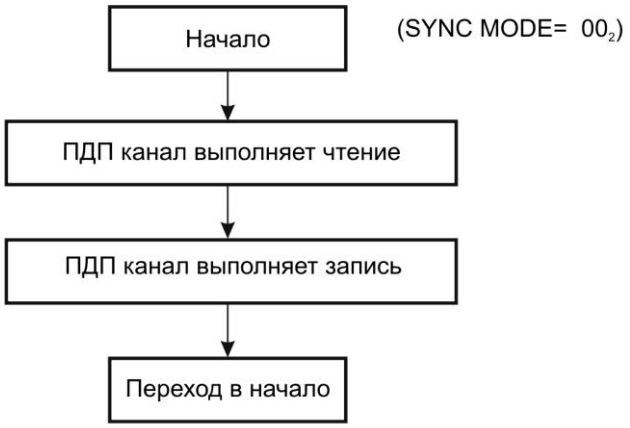


Рисунок 5.134 − Нет синхронизации ПДП

Синхронизация источника данных

Когда SYNC MODE = 01_2 (для основного режима), или когда SYNC MODE = 10_2 (для вспомогательного канала в режиме разделения), сопроцессор ПДП синхронизируется с источником данных (см. рисунок 5.135). 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.

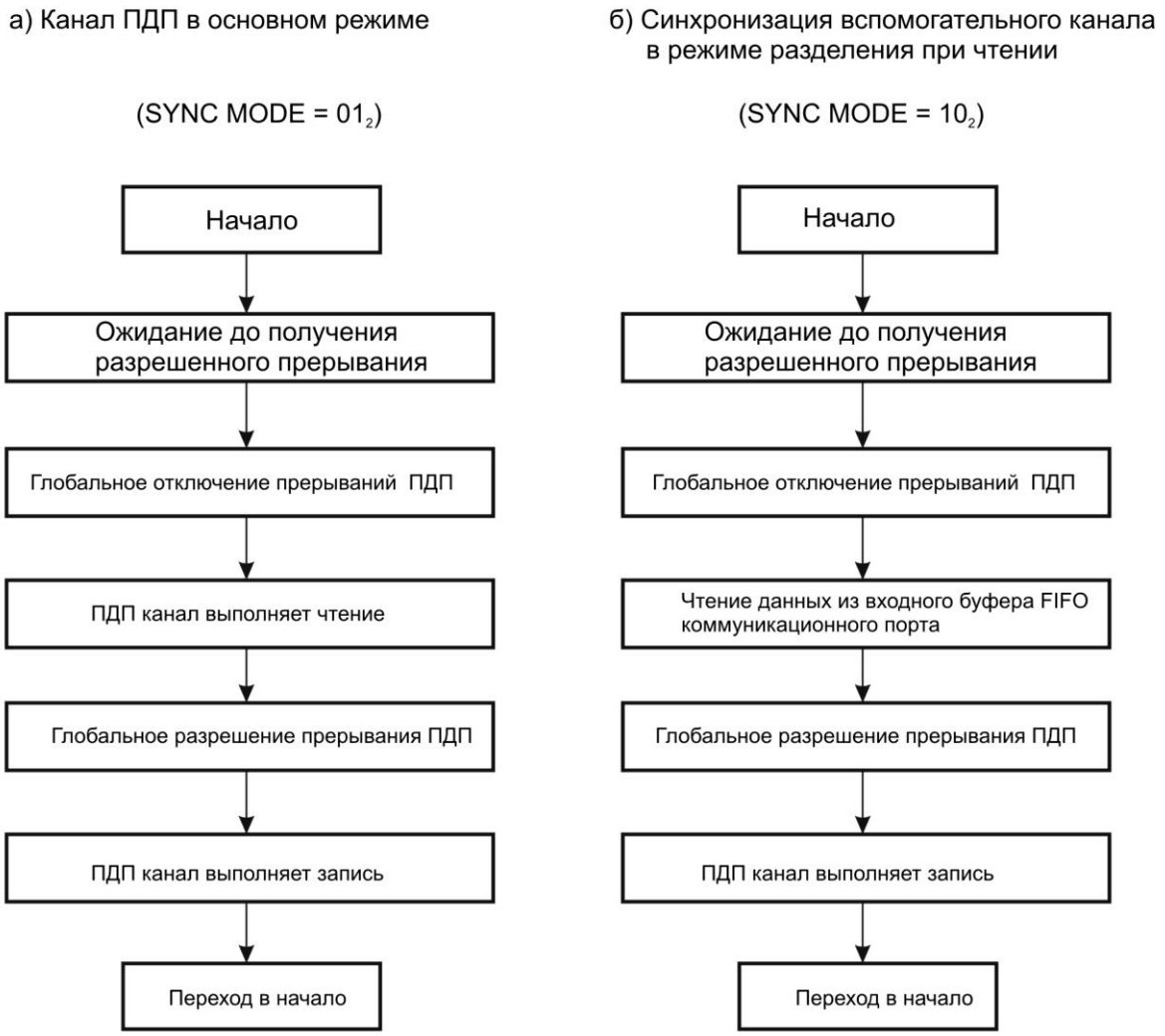


Рисунок 5.135 − Синхронизация источника ПДП

Когда SYNC MODE = 10_2 (для основного режима), или когда SYNC MODE = 01_2 (для основного канала в режиме разделения), сопроцессор ПДП синхронизируется с приёмником данных (см. рисунок 5.136). Запись не выполняется, пока канал ПДП не получит сигнал прерывания.

В основном режиме чтение выполняется без ожидания прерывания. Однако в режиме разделения чтение выполняется только тогда, когда получено прерывание сигнала, разрешающего 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.

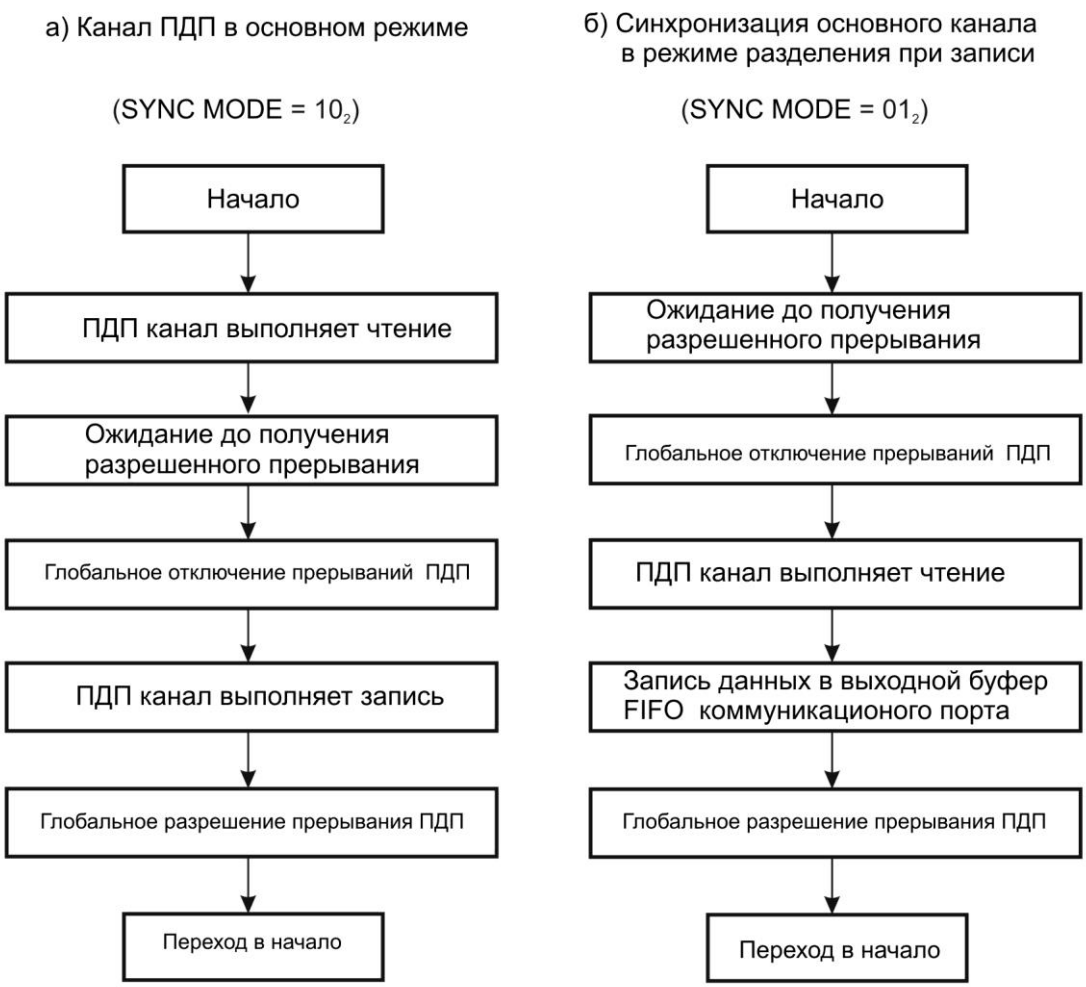


Рисунок 5.136 − Синхронизация приёмника ПДП

Синхронизация источника и приёмника данных.

Когда SYNC MODE = 11_2, чтение выполняется, когда приходит прерывание чтения, запись выполняется, когда приходит прерывание записи. Если прерывание записи получено до прерывания чтения, прерывание записи фиксируется, и запись данных ПДП не выполняется, пока не закончится чтение. Синхронизация источника и приёмника данных (SYNC MODE = 11_2) показана на рисунке 5.137.

Если выбран режим разделения, он представляется как два независимых типа синхронизации для основного (синхронизация записи) и вспомогательного (синхронизация чтения) каналов. Если выбрано одно и то же прерывание и для чтения, и для записи (как в основном режиме, так и в режиме разделения), только одно единственное прерывание разрешает операции чтения и записи.

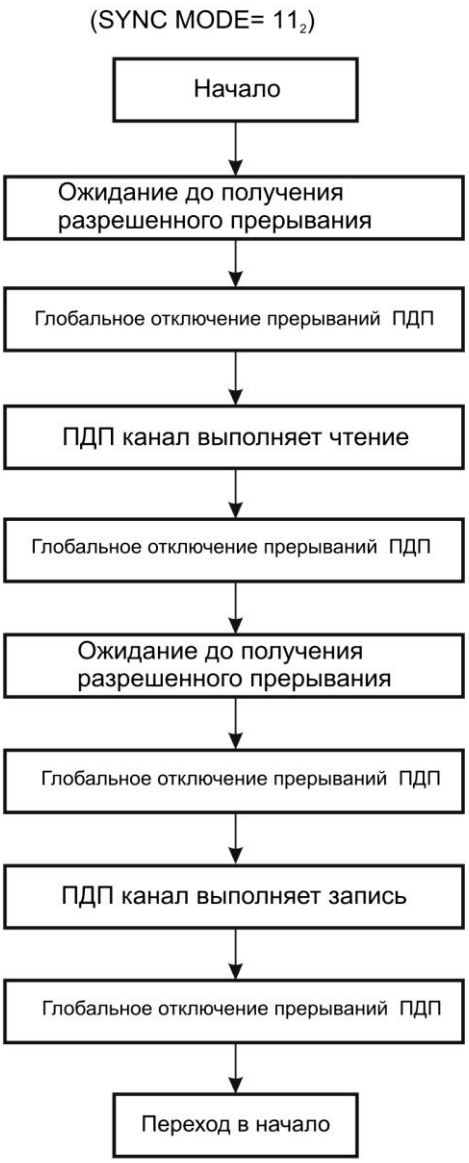


Рисунок 5.137 − Синхронизация источника и приёмника ПДП в основном режиме

5.11.11 Временные диаграммы передачи данных памяти ПДП

Ядро процессора 1867ВЦ8Ф1 поддерживает шесть каналов ПДП (12 каналов ПДП, если они все в режиме разделения) со схемами арбитража с фиксированными/циклическими приоритетами и конфигурируемые схемы приоритетов ЦПУ/ПДП (см. 5.11.6 и 5.11.7).

Максимальная скорость передачи данных возможна, если ядро процессора 1867ВЦ8Ф1 передаёт одно слово каждые два цикла. Шесть каналов ПДП передают данные в последовательности, разделённой по времени, а не одновременно, так как они используют общие шины.

Временные диаграммы передачи данных памяти ПДП могут быть очень сложными, особенно если возникает конфликт источника. Однако некоторые правила помогут вам посчитать временные диаграммы передачи данных для определённых установок ПДП. Для упрощения следующие подразделы сфокусированы на одноканальной передаче данных без конфликтов с ЦПУ или другими каналами ПДП. Вы можете получить истинную временную диаграмму передачи данных ПДП посредством совокупного подсчёта для одноканальных передач с учётом конфликтных ситуаций.

5.11.11.1 Временная диаграмма одноканальной передачи данных памяти ПДП

Если передача данных памяти ПДП не имеет конфликтов с ЦПУ или другими каналами ПДП, число циклов передачи зависит от того, находятся ли начальные и конечные адреса во внутренней памяти, периферии или во внешних портах. Если используется внешний порт, скорость передачи данных зависит от двух факторов: состояния ожидания внешней шины и конфликта чтения/записи (например, если запись следует за чтением, чтение занимает два цикла). В таблицах 5.49 – 5.51 показано требуемое число циклов передачи данных ПДП от различных источников к разным приёмникам. Содержимое таблицы показывает число циклов, необходимое для Т передач, исходя из предположения, что отсутствуют конфликты конвейера.

Таблица 5.49 − Временная 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
Внутренний начальный адрес	9	R		R		R		R		R		R		R		R		R	
Внутренний конечный адрес			W		W		W		W		W		W		W		W		W
Начальный адрес – локальная шина	4	R	R	R		R	R	R		R	R	R		R	R	R			
			Cr				Cr				Cr				Cr				
Внутренний конечный адрес					W				W				W				W		
Начальный адрес – глобальная шина	4	R	R	R		R	R	R		R	R	R		R	R	R			
			Cr				Cr				Cr				Cr				
Внутренний конечный адрес					W				W				W				W		

Таблица 5.50

Начальный адрес	Конечный адрес: внутренний
Внутренний	(1+1)T
Локальная шина	[(1+Cr)+1]T
Глобальная шина	[(1+Cr)+1]T

Принятые условные обозначения:

- T − число передач;
- Cr − состояние ожидания чтения источника;
- R − одноцикловые чтения;
- W − одноцикловые записи;
- R R − многоцикловые чтения.

Таблица 5.51 − Временная диаграмма и число циклов передачи данных ПДП на локальную шину

Циклы	T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Внутренний начальный адрес	4	R		R				R				R							
			W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	
Конечный адрес – локальная шина					Cw				Cw				Cw				Cw		
Начальный адрес – локальная шина	2	R	R	R					R	R	R	R							
			Cr								Cr								
Конечный адрес – локальная шина					W	W	W	W					W	W	W	W			
							Cw								Cw				
Начальный адрес – глобальная шина	3	R	R	R		R	R	R		R	R	R							
			Cr				Cr				Cr								
Конечный адрес – локальная шина					W	W	W	W	W	W	W	W	W	W	W	W			
							Cw				Cw				Cw				

Таблица 5.52

Начальный адрес	Конечный адрес: 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)

Принятые условные обозначения:
- T − число передач;
- Cr − состояние ожидания чтения источника;
- Cw − состояние ожидания записи приёмника;
- R − одноцикловые чтения;
- R R − многоцикловые чтения;
- W W − многоцикловые записи.

Таблица 5.53 − Временная диаграмма и число циклов передачи данных ПДП на глобальную шину

Циклы	T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Внутренний начальный адрес	4	R		R				R				R							
			W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	
Конечный адрес – глобальная шина					Cw				Cw				Cw				Cw		
Начальный адрес – локальная шина	3	R	R	R		R	R	R		R	R	R							
			Cr				Cr				Cr								
Конечный адрес – глобальная шина					W	W	W	W	W	W	W	W	W	W	W	W			
							Cw				Cw				Cw				
Начальный адрес – глобальная шина	2	R	R	R					R	R	R	R							
			C r								Cr								
Конечный адрес – глобальная шина					W	W	W	W					W	W	W	W			
							Cw								Cw				

Таблица 5.54

Начальный адрес	Конечный адрес: 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

Принятые условные обозначения:
- T − число передач;
- Cr − состояние ожидания чтения источника;
- Cw − состояние ожидания записи приёмника;
- R − одноцикловые чтения;
- R R − многоцикловые чтения;
- W W − многоцикловые записи.

Внешне, на глобальной и локальной шинах, запись занимает не менее двух циклов. Однако внутри ЦПУ/ПДП требуется один цикл для выполнения записи во внешнюю память. Следовательно, ПДП/ЦПУ могут передавать данные в следующем цикле, если они не на ту же внешнюю шину. Например, ПДП передаёт 1024 слова из внутренней памяти блока ОЗУ 1 на глобальную шину, в память с одним состоянием ожидания, в то время как ЦПУ осуществляет доступ к памяти на локальной шине и выбирает операнды из блока ОЗУ 0. Время передачи ПДП вычисляется по таблице 5.54 как 1 + (2 + 1) 1024 = 1 + 3072 = 3073 цикла.

5.11.11.2 Скорость передачи данных ПДП в режиме синхронизации

Режим синхронизации, используемый для передачи данных, также влияет на скорость передачи ПДП. Скорость передачи данных ПДП ниже, если используется синхронизация, потому что для сброса запроса от прерывания требуется два цикла. Однако эти два дополнительных цикла могут быть поглощены, если одновременно работают несколько каналов ПДП.

В основном режиме с использованием синхронизации максимальная скорость передачи составляет одно слово каждые три цикла. В таблице 5.56 приведено количество циклов, необходимых для передач ПДП в основном режиме с различными типами синхронизации. Для упрощения рассматривается одноканальная передача данных ПДП без конфликтов с ЦПУ или с другими каналами, без состояний ожидания и всегда активными прерываниями.

Таблица 5.55 − Временная диаграмма основного режима ПДП при различных типах синхронизации

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				

Таблица 5.56

Синхронизация	Длительность
Нет синхронизации	2T
Синхронизация чтения	3T
Синхронизация записи	1+3T
Синхронизация чтения и записи	1+3T

Принятые условные обозначения:
- T − число передач;
- R − одноцикловые чтения;
- W − одноцикловые записи;
- Rr − сброс флага чтения (два цикла);
- Wr − сброс флага записи (два цикла).

В режиме разделения с использованием синхронизации максимальная скорость передачи и для основного, и для вспомогательного канала составляет одно слово каждые четыре цикла. Когда вспомогательный и основной каналы запущены одновременно, дополнительные два цикла для сброса прерываний поглощаются, и максимальная скорость может быть равна одному слову каждые два цикла. В таблице 5.57 показано число циклов передач ПДП, необходимых в режиме разделения при различных типах синхронизации. Для упрощения рассматривается одноканальная передача данных ПДП без конфликтов с ЦПУ или с другими каналами, без состояний ожидания и всегда активными прерываниями.

Таблица 5.57 − Временная диаграмма основного режима ПДП при различных типах синхронизации

Циклы	T	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

Таблица 5.58

Синхронизация	Длительность
Нет синхронизации (оба канала запущены)	2T
Синхронизация основного канала (вспомогательный канал не запущен)	4T
Синхронизация вспомогательного канала (основной канал не запущен)	4T
Синхронизация основного и вспомогательного каналов (оба канала запущены)	2T+2

Принятые условные обозначения:
- T − число передач;
- R − одноцикловое чтение основного канала;
- R’ − одноцикловое чтение вспомогательного канала;
- W − одноцикловая запись основного канала;
- W’ − одноцикловая запись вспомогательного канала;
- Pr − сброс флага основного канала (два цикла);
- Ar − сброс флага дополнительного канала (два цикла).

5.12 Коммуникационные порты

ПЦОС имеет шесть коммуникационных портов для связи с другими аналогичными процессорами и/или внешними устройствами. Кроме того, порты могут работать с сопроцессором прямого доступа к памяти, чтобы передать либо принять данные без вмешательства в работу центрального процессора, что позволяет освободить его для выполнения других задач. Эта глава описывает основные особенности, карту памяти, регистры и операции коммуникационных портов ПЦОС.

5.12.1 Основные функции коммуникационных портов

Каждый коммуникационный порт ПЦОС имеет несколько основных особенностей:

- Максимальная скорость двунаправленной передачи данных 350 Мбайт в секунду (при 10 нс входных тактах).
- Простая связь между микропроцессорами через восьмиразрядные шины данных и четыре сигнала контроля передачи.
- Буферизация FIFO памяти всех передаваемых/принимаемых данных.
- Автоматический арбитр 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, двухмерные Евклидовые сетки, гексагональные сетки и трёхмерные сетки.
- Программный сброс коммуникационного порта.

5.12.2 Краткий обзор коммуникационных портов

ПЦОС имеет шесть идентичных быстродействующих портов, каждый из которых обеспечивает двунаправленную связь с подобными ПЦОС или другими внешними устройствами. На рисунке 5.138 приведена структурная схема коммуникационного порта. Каждый порт содержит следующие компоненты:

- входной буфер FIFO – представляет собой восьмиуровневый 32-разрядный стек. Входной буфер FIFO изолирует ПЦОС от случайных внешних данных на шине данных порта и буферизирует синхронизированные полученные данные;
- выходной буфер FIFO − представляет собой восьмиуровневый 32-разрядный стек. Выходной буфер FIFO изолирует ПЦОС 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 5.12.4;
- регистр команд коммуникационного порта CPCR − позволяет управлять функциями коммуникационного порта и операциями передачи данных между ПЦОС и внешними устройствами;
- регистр программного сброса коммуникационного порта позволяет очистить входной буфер FIFO и выходной буфер FIFO порта. Это объясняется в 5.12.3.4.

Коммуникационный порт передаёт сохранённые в его выходном буфере FIFO 32-разрядные слова, побайтно. Поскольку сигналы управления передачей данных и сами шины передачи данных двунаправленные, каждый ПЦОС должен иметь монопольное право на использование информационной шины коммуникационного порта прежде, чем начинать передачу данных. Имитированный указатель передачи используется, чтобы определять монопольное использование шины: коммуникационный порт, который имеет указатель передачи, может передать данные.

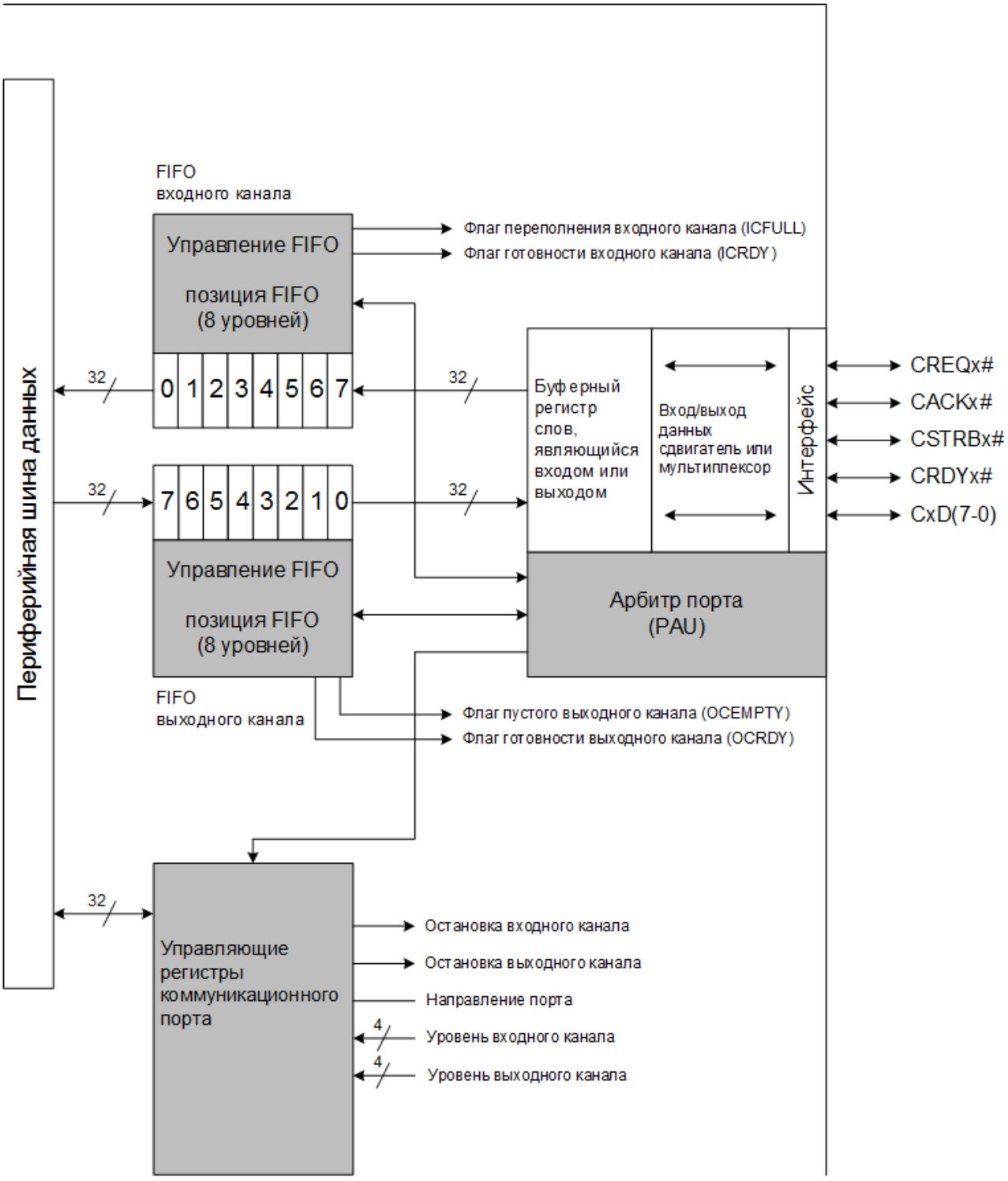


Рисунок 5.138 – Структурная схема коммуникационного порта

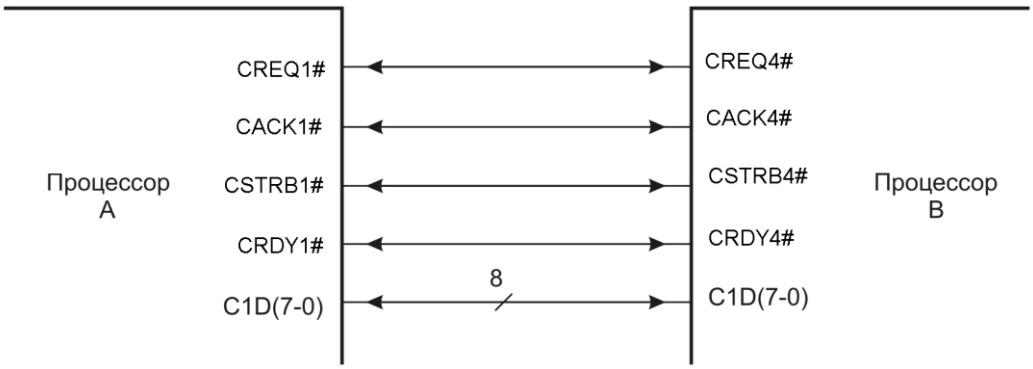


Рисунок 5.139 − Пример соединения процессорных ядер ИС 1867ВЦ8Ф1 через их коммуникационные порты

На рисунке 5.139 приведён пример соединения двух процессорных ядер ИС 1867ВЦ8Ф1 через их коммуникационные порты. Это соединение обеспечивают следующие двунаправленные сигналы управления и шины передачи данных:

CREQx# − запрос указателя передачи коммуникационного порта. ПЦОС активирует этот сигнал для запроса использования шины данных коммуникационного порта;

CACKx# − подтверждение передачи указателя передачи коммуникационного порта. ПЦОС активирует это сигнал, когда передаёт права на монопольное использование информационной шины, это сигнал является откликом на сигнал запроса указателя передачи CREQx# от другого ПЦОС;

CSTRBx# − строб шины данных коммуникационного порта. ПЦОС активирует этот сигнал указывая на то, что он разместил байт достоверных данных на информационной шине порта;

CRDYx# − сигнал готовности коммуникационного порта. ПЦОС активирует этот сигнал указывая на то, что байт данных получен через информационную шину коммуникационного порта;

CxD(7 − 0) − информационная шина коммуникационного порта. Эта шина служит для двунаправленной передачи данных между ПЦОС или другими внешними устройствами.

5.12.2.1 Операция передачи указателя передачи

Чтобы передавать указатель передачи, арбитры двух ПЦОС взаимодействуют для генерации сигналов и управляющих последовательностей, необходимых для гарантированной передачи данных. Чтобы избегать конфликтов на шине данных, арбитры выносят решение о том какому ПЦОС передать монопольное использование шины данных.

Через сигналы CREQx# и CACKx# арбитры обрабатывают процедуру установки связи между двумя ПЦОС, осуществляя это следующим образом:

- арбитр, который не имеет монопольного доступа к информационной шине (CxD(7 − 0)) активирует CREQx# для запроса на монопольное использование шины;

- арбитр, который имеет монопольный доступ к информационной шине, в ответ активирует CACKx#, чтобы подтвердить запрос и передать монопольное использование шины запрашивающему арбитру.

Таким способом, эти сигналы передают указатель передачи (или приоритет) от одного арбитра к другому. См. 5.12.7 для детального рассмотрения данного вопроса.

5.12.2.2 Операция передачи данных

Операция передачи данных осуществляется в четыре основных этапа:

- ЦПУ или сопроцессор ПДП записывает 32-разрядные данные в выходной буфер FIFO (коммуникационного порта), через адрес картированный в карте памяти коммуникационных портов, которая представлена на рисунке 5.140;
- коммуникационный порт побайтно размещает 32-разрядное информационное слово на шине CxD(7 − 0), стробируя каждый байт сигналом CSTRBx#, сообщая этим, что каждый байт выставлен на шине достоверно;
- после получения каждого байта данных, принимающий коммуникационный порт генерирует сигнал CRDYx#, указывая на то, что байт данных получен, и порт готов принять следующий байт;
- после получения четырёх байт 32-разрядного слова, ЦПУ или сопроцессор ПДП может начать считывание данных с входного буфера FIFO через адрес, картированный в памяти коммуникационных портов, которая представлена на рисунке 5.140.

Как входной, так и выходной буферы FIFO, могут буферизировать максимум по восемь 32-разрядных слов.

Карта памяти и регистры коммуникационных портов

На рисунке 5.140 представлена карта памяти коммуникационных портов ПЦОС. На карте памяти представлены адреса: регистров CPCRs, входных и выходных буферов FIFO коммуникационных портов. Назначение битов регистра CPCR представлено на рисунке 5.141.

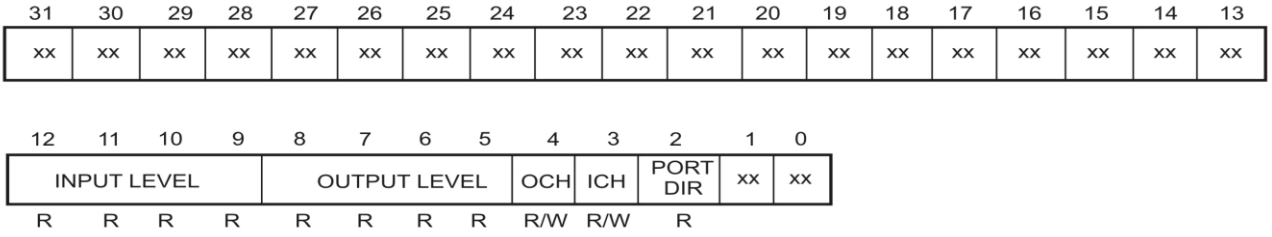
0010 0040h	CPCR0
0010 0041h	Входной буфер FIFO порта 0
0010 0042h	Выходной буфер FIFO порта 0
0010 0043h	Программный сброс порта 0
0010 0050h	CPCR1
0010 0051h	Входной буфер FIFO порта 1
0010 0052h	Выходной буфер FIFO порта 1
0010 0053h	Программный сброс порта 1
0010 0060h	CPCR2
0010 0061h	Входной буфер FIFO порта 2
0010 0062h	Выходной буфер FIFO порта 2
0010 0063h	Программный сброс порта 2
0010 0070h	CPCR3
0010 0071h	Входной буфер FIFO порта 3
0010 0072h	Выходной буфер FIFO порта 3
0010 0073h	Программный сброс порта 3
0010 0080h	CPCR4
0010 0081h	Входной буфер FIFO порта 4
0010 0082h	Выходной буфер FIFO порта 4
0010 0083h	Программный сброс порта 4
0010 0090h	CPCR5
0010 0091h	Входной буфер FIFO порта 5
0010 0092h	Выходной буфер FIFO порта 5
0010 0093h	Программный сброс порта 5
0010 009Fh	

Рисунок 5.140 − Карта памяти коммуникационных портов

5.12.3 Регистры коммуникационных портов

5.12.3.1 Регистр управления коммуникационного порта CPCR

На рисунке 5.141 приведено назначение и описание битов регистра управления порта CPCR.



Примечания

1 xx – зарезервированный бит.

2 R – чтение, W – запись.

Рисунок 5.141 − Регистр команд коммуникационного порта (CPCR)

PORT DIR – направление порта. Этот разряд определяет направление операций передачи данных для коммуникационного порта. PORT DIR = 0: порт находится в выходном режиме. PORT DIR = 1: порт находится во входном режиме. Этот разряд доступен только на чтение. Невозможно изменить направление порта программным способом.

ICH – останов входного канала. Запись 1 в разряд ICH останавливает входной канал порта. При записи 0 в разряд ICH входной канал возобновляет свою работу. Входной канал не может внешним образом индицировать готовность к приёму.

OCH – останов выходного канала. Запись 1 в разряд OCH немедленно останавливает выходной канал порта. Однако порт по-прежнему может принимать запрос указателя от входного канала. При записи 0 в разряд OCH выходной канал возобновляет свою работу.

OUTPUT LEVEL – уровень выходного буфера FIFO. Содержимое этого четырёхбитного поля: 0000_2(0): указывает на то, что выходной буфер FIFO пуст. 0001_2(1) – 0111_2(7): индицирует количество заполненных уровней в выходном буфере FIFO. 1111_2(15): 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, прерывание OCEMPTY очищается (OCEMPTY = 0), и находится в этом состоянии, пока буфер снова не очистится. Выходной буфер FIFO с одним или более свободными уровнями также вырабатывает прерывание (OCRDY = 1) на ЦПУ и сопроцессор ПДП. По этому условию ЦПУ или сопроцессор ПДП могут записать данные в выходной буфер FIFO. Смотри раздел 5.12.6 для более детальной информации.

INPUT LEVEL – уровень входного буфера FIFO. Содержимое этого четырёхбитного поля: 0000_2(0): указывает на то, что входной буфер FIFO пуст. 0001_2(1) – 0111_2(7): индицирует количество заполненных уровней во входном буфере FIFO. 1111_2(15): указывает на то, что входной буфер FIFO заполнен. Если входной буфер FIFO заполнен (INPUT LEVEL = 1111_2) срабатывает прерывание (ICFULL = 1). Когда ЦПУ или сопроцессор ПДП считают все данные из входного буфера, прерывание ICFULL очищается (ICFULL = 0) и находится в этом состоянии, пока буфер снова не заполнится. Входной буфер FIFO с одним или более свободными уровнями также вырабатывает прерывание (ICRDY = 1) на ЦПУ и сопроцессор ПДП. По этому условию ЦПУ или сопроцессор ПДП могут считывать данные с входного буфера FIFO.

Зарезервированный – эти разряды не определены.

5.12.3.2 Регистр входного буфера FIFO

В этом доступном только на чтение регистре находится содержимое позиции 0, самого старого значения входного FIFO. Если в этот регистр производится запись, его содержимое остаётся неизменным. Чтение из пустого входного FIFO вызывает остановку работы ЦПУ или ПДП и остановку периферийной шины.

5.12.3.3 Регистр выходного буфера FIFO

Этот доступный только на запись регистр взаимодействует с позицией 7 (самым новым значением) выходного FIFO. Если этот регистр читается, его содержимое остаётся неизменным, а прочитанное значение не определено. Если в выходной FIFO выполняется запись, интерфейс периферийной шины фиксирует слово и возвращает сигнал неготовности. Это условие исчезает, когда в выходном FIFO появляется пустая позиция и данные по шине передаются в FIFO.

Этот регистр используется для записи данных в выходной буфер FIFO и дальнейшей выдачи их на шину данных порта.

5.12.3.4 Регистр программного сброса коммуникационного порта

Заполненные уровни входного и выходного буферов FIFO коммуникационного порта могут быть очищены при записи данных по адресу регистра программного сброса порта. В таблице 5.59 указаны адреса регистров программного сброса коммуникационных портов. Программный сброс портов не оказывает влияния на состояние внешних выводов.

Таблица 5.59 – Адреса регистров программного сброса коммуникационных портов

Порт	Адрес регистра программного сброса порта
0	0100043h
1	0100053h
2	0100063h
3	0100073h
4	0100083h
5	0100093h

В примере 5.66 представлен листинг программы, иллюстрирующий методику программного сброса коммуникационного порта.

```
; ------------------------------------------------;
; RESET1:Flushes FIFOs data for communication port 1;
; ------------------------------------------------;
RESET1 push  AR0          ;  Save registers
       push  R0           ;
       push  RC           ;
       ldhi  010h,AR0     ;  Set AR0 to base address of COM 1
       or    050h,AR0     ;
FLUSH: rpts  1            ;Flush FIFO data with back-to-back writ
       sti   R0,*+AR0(3)  ;
       rpts  10           ;  Wait
       nop                ;
       ldi   *+AR0(0),R0  ;  Check for new data from other port
       and   01FE0h,R0    ;
       bnz   FLUSH        ;
       pop   RC           ;  Restore registers
       pop   R0           ;
       pop   AR0          ;
       rets               ;  Return
```

5.12.4 Арбитры коммуникационного порта

Арбитры выносят решение между двумя ПЦОС, чтобы определить, какой из процессоров имеет право на монопольное владение информационной шиной данных коммуникационного порта. Для работы арбитр использует сигналы CREQ# и CACK#, которые сообщают ему какому ПЦОС передать указатель передачи монопольного использования шины. Операция передачи указателя передачи подробно описывается в 5.12.7.

После системного сброса, половина портов ПЦОС имеет указатель передачи (коммуникационные порты 0, 1, 2), а другая половина (коммуникационные порты 3, 4, 5) не имеет.

Арбитр является синхронным конечным автоматом с четырьмя состояниями, как это показано в таблице 5.60. Изменение этих состояний невозможно программным способом.

Таблица 5.60 − Состояния автомата арбитра

Номер состояния	Краткое описание	Состояние арбитра
Состояние 0: Простой с указателем передачи	1. Арбитр имеет указатель передачи (PORT DIR = 0). 2. Канал не используется.	Арбитр в настоящее время имеет указатель передачи и право на монопольное использование шины данных, но шина в данный момент не используется. При этом условии, разряд PORT DIR регистра CPCR равен нулю (выход). После системного сброса это состояние имеют 0, 1 и 2 порт.
Состояние 1: Простой без указателя передачи	1. Арбитр не имеет указателя передачи (PORT DIR = 1). 2. Арбитр не производит запрос указателя передачи (OUTPUT LEVEL = 0).	Арбитр в настоящее время не имеет указателя передачи и права на монопольное использование шины данных и не производит запрос указателя передачи. При этом условии, разряд PORT DIR регистра CPCR равен единице (вход). После системного сброса это состояние имеют 3, 4 и 5 порт.
Состояние 2: Активизация	1. Арбитр имеет указатель передачи (PORT DIR = 0). 2. Канал занят (OUTPUT LEVEL ≠ 0).	Арбитр в настоящее время имеет указатель передачи, и производит использование шины данных. При этом условии, разряд PORT DIR равен нулю (выход), и поле OUTPUT LEVEL не равно нулю.
Состояние 3: Ожидание указателя передачи	1. Арбитр не имеет указателя передачи (PORT DIR = 1). 2. Арбитр производит запрос указателя передачи (OUTPUT LEVEL ≠ 0).	Арбитр в настоящее время не имеет указателя передачи и права на монопольное использование шины данных, но производит запрос указателя передачи. При этом условии, разряд PORT DIR регистра CPCR равен единице (вход), и поле OUTPUT LEVEL не равно нулю.

На рисунке 5.142 приведена диаграмма состояния и управления переходами состояний.

Чтобы размещать данные на информационной шине коммуникационного порта, арбитр должен вынести решение между двумя типами запросов:

- внутренние запросы к выходным данным в выходном буфере FIFO (показанный как BUSRQ = 1 на рисунке 5.142);

- внешние запросы, полученные через сигнал CREQ# (показанный как TOKRQ = 1 на рисунке 5.142).

Далее проведём исследование схемы работы арбитра порта, представленной на рисунке 5.142, рассмотрим операцию передачи данных от ПЦОС А к ПЦОС В. Передача начинается с нулевого состояния арбитра (простой с указателем передачи) и арбитр ПЦОС В находится в состоянии 1 (простой без указателя передачи).

Если арбитр А получает запрос на передачу данных с выходного буфера (BUSRQ = 1), то арбитр переходит в состояние 2 (активный). После того, как буфер вывода передаёт одно слово, арбитр удаляет запрос шины (BUSRQ = 0), и арбитр возвращается к состоянию 0 (простой с указателем передачи).

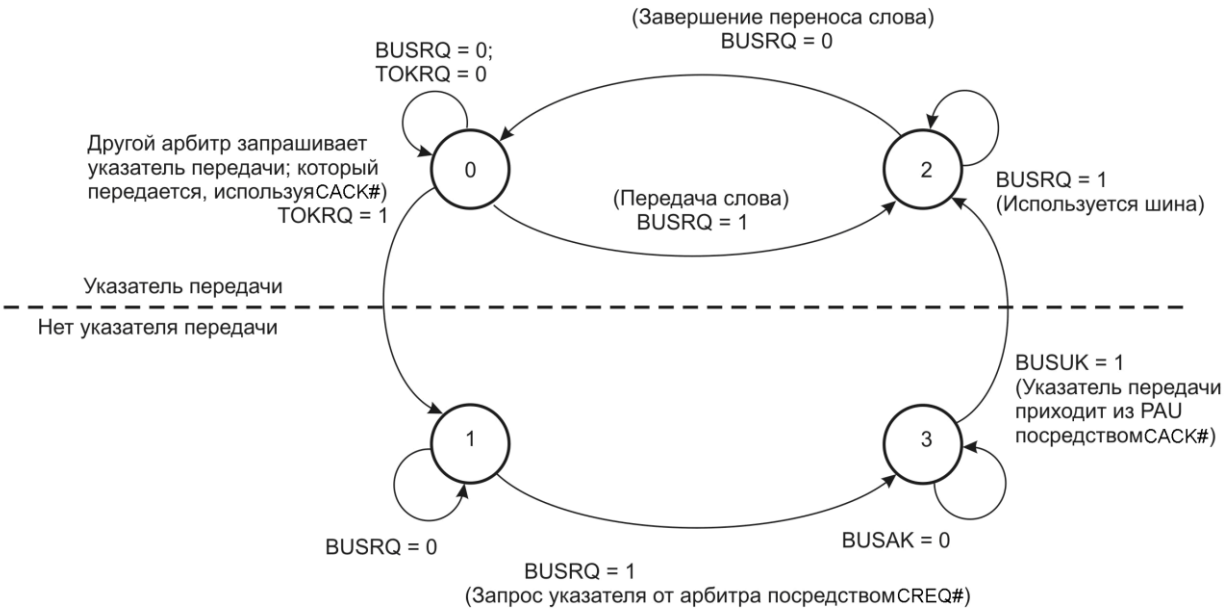


Рисунок 5.142 − Диаграмма состояния и управления переходами состояний

Если арбитр B получает запрос на передачу данных с выходного буфера, чтобы использовать шину, то арбитр активирует сигнал CREQ#, чтобы запросить указатель передачи от арбитра A. Арбитр A производит поиск запроса через переменную TOKRQ=1 и затем активирует сигнал CACK#, чтобы передать указатель передачи монопольного использования шины данных к арбитру B. После этого арбитр B генерирует внутренний сигнал, подтверждения (BUSAK = 1), чтобы указать, что он получил монопольное использование шины. В результате операции передачи указателя передачи, арбитр A входит в состояние 1 (простой без указателя передачи), а арбитр B переходит в состояние 2 (активный).

Чтобы препятствовать любому коммуникационному порту монополизировать шину данных, арбитр всегда возвращается к состоянию 0 (простой с указателем передачи) и производит запрос указателя передачи (активный CREQ#) от внешнего устройства после каждой передачи слова.

Если запрос указателя передачи активен, указатель передачи переходит к запрашивающему ПЦОС для передачи слова. Если ПЦОС A и ПЦОС B имеют информацию к отправке в своих выходных буферах FIFO, они чередуют использование информационной шины, чтобы обеспечить двунаправленный информационный канал.

Если запрос указателя передачи получается в конце передачи слова и передающий ПЦОС имеет другое слово для отправки в своём выходном буфере FIFO, 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 ПЦОС прекращает передачу данных и начинает передачу указателя передачи;

- если сигнал CREQ# переходит в низкий уровень после или одновременно с переходом сигнала CRDY# в низкий уровень, для последнего байта слова, то ПЦОС продолжает передавать следующее слово и передаёт указатель передачи только после окончания передачи всего слова.

В итоге, ПЦОС не будет передавать указатель передачи, пока не закончится передача четырёх байтов.

5.12.5 Остановка работы буферов ввода и вывода порта

ПЦОС 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 третий разряд (ICH = 1) регистра управления коммуникационного порта CPCR. Этот разряд также может читаться, чтобы решить, что порт останавливается или является в состоянии получить данные. Необходимо записать 0 в разряд ICH, чтобы не останавливать работу входного буфера FIFO.

Чтобы остановить выходной буфер FIFO, необходимо записать 1 в четвёртый разряд (OCH = 1) регистра управления коммуникационного порта CPCR. Этот разряд также может читаться, чтобы решить, что порт останавливается или является в состоянии передать данные. Необходимо записать 0 в разряд OCH, чтобы не останавливать работу выходного буфера FIFO. Операции останова/пуска буферов FIFO рассматриваются далее. Итоговые сведения представлены в таблице 5.61.

Таблица 5.61 − Операции останова/пуска буферов FIFO

Комбинации состояний	Если порт имеет указатель передачи	Если порт не имеет указателя передачи
Входной остановлен; выходной не остановлен	А. Не реализует передачу указателя передачи	А. Если остановка происходит после принятия слова, то порт не выдаёт сигнал готовности на принятие нового слова. Если остановка происходит во время принятия слова, то порт принимает одно слово и останавливается
	Б. Передаёт данные	Б. Если остановка происходит после принятия первого байта слова, то порт принимает конец слова и останавливается
Входной не остановлен; выходной остановлен	А. Не передаёт данные	А. Принимает данные
	Б. Если остановка происходит после передачи первого байта слова, то порт передаёт конец слова и останавливается	Б. Не отвечает на запросы передачи указателя передачи
	В. Передаёт указатель передачи данных	
Входной остановлен; выходной остановлен	А. Не реализует передачу указателя передачи	А. Если остановка происходит после принятия слова, то порт не выдаёт сигнал готовности на принятие нового слова. Если остановка происходит во время принятия слова, то порт принимает одно слово и останавливается
	Б. Не передаёт данные	Б. Если остановка происходит после принятия первого байта слова, то порт принимает конец слова и останавливается
	В. Если остановка происходит после передачи первого байта слова, то порт передаёт конец слова и останавливается	В. Не отвечает на запросы передачи указателя передачи

5.12.5.1 Описание остановки входного буфера FIFO

Цель останова входного буфера FIFO состоит в том, чтобы остановить работу входного буфера FIFO как можно скорее, не теряя входных данных.

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 указателя передачи низким уровнем сигнала CACK# на низкий уровень сигнала CREQ#, то это указывает на то, что система 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:

- если коммуникационный порт получает бит останова входного буфера, когда не происходит приём слова, входной буфер FIFO не останавливается немедленно, а ждёт, 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 приём слова, входной буфер FIFO не останавливается немедленно, а ждёт, чтобы получить слово до конца и затем остановиться. Это состояние сохраняется до тех пор, пока не произойдёт сброс бита останова либо сброс порта. При сбросе бита останова приём данных продолжается, без каких-либо потерь.

Если входной буфер FIFO коммуникационного порта останавливается в течение запроса указателя передачи от другого коммуникационного порта, с которым он соединён, тогда запрос указателя передачи подтверждается перед остановкой входного буфера FIFO.

5.12.5.2 Описание остановки выходного буфера FIFO

Остановка выходного буфера FIFO аналогична остановке входного буфера FIFO. Если выходной буфер остановлен, возможны две ситуации, в зависимости от того, имеет или не имеет порт указатель передачи.

Если порт не имеет указатель передачи:

- выходной буфер FIFO останавливается немедленно и не реагирует на запрос указателя передачи;
- если коммуникационный порт, запрашивающий указатель передачи, останавливается после отправки низкого уровня сигнала CREQ#, то коммуникационный порт принимает указатель передачи и останавливается немедленно после этого.

Если порт имеет указатель передачи:

- если в настоящее время порт передаёт слово, тогда только после передачи слова произойдёт остановка выходного буфера, и никакие новые передачи не произойдут;
- если в настоящее время порт не передаёт слово, тогда происходит немедленная остановка буфера и никакие новые передачи не произойдут;
- если входной буфер FIFO не останавливается, а выходной буфер FIFO останавливается, тогда порт отвечает на запрос указателя 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 указатель передачи, в выходном буфере имеются данные для передачи, и происходит очищение бита останова выходного буфера, то порт возобновляет передачу данных.

5.12.6 Организация 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 ПДП или ЦПУ пытаются читать пустой входной буфер FIFO или записывать данные в заполненный 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. Сигналом готовности для выходного канала является OCRDY (выходной канал готов), этот сигнал также является сигналом прерывания. Сигналом готовности для входного канала является ICRDY (входной канал готов), который также является сигналом прерывания.

При синхронизации по сигналам прерываний, каждый коммуникационный порт генерирует четыре других сигнала прерывания, описание которых представлено ниже:

- ICFULL (входной канал заполнен): указывает, что входной буфер FIFO имеет восемь слов;
- ICRDY (входной канал готов): указывает, что, по крайней мере, одно слово находится во входном буфере FIFO;
- OCRDY (выходной канал готов): указывает, что, по крайней мере, одно слово находится в выходном буфере FIFO;
- OCEMPTY (выходной канал пуст): указывает, что выходной буфер FIFO пуст.

Центральный процессор может ответить на все четыре из этих сигналов 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 – зависит от настроенной конфигурации.

Следует обратить внимание, что ни один из четырёх сигналов прерывания коммуникационного порта не имеет флагов в регистре IIF. Эти четыре сигнала состояния коммуникационного порта (ICFULL, ICRDY, OCRDY и OCEMPTY) могут быть получены при проверке входных и выходных уровней в регистре управления коммуникационного порта (CPCR). Например, чтобы определить есть ли прерывание ICFULL, достаточно проверить 12-разрядный регистр CPCR.

Максимальная устойчивая скорость передачи данных любого одиночного коммуникационного порта при входной тактовой частоте процессора, равной 100 МГц, равна 40 Мбайт/с.

5.12.7 Операция передачи указателя передачи

Операция передачи указателя передачи запрашивает процедуру установления связи сигналов через выводы CREQ# и CACK#. Это поясняется на рисунке 5.143. Для ясности, суффикс идентифицирует сигналы, выходящие из соответствующего микропроцессора. Например, CREQb# обозначает сигнал CREQ#, выходящий из микропроцессора B. Таблица 5.62 представляет список событий, а рисунок 5.143 графически показывает процедуру установления связи.

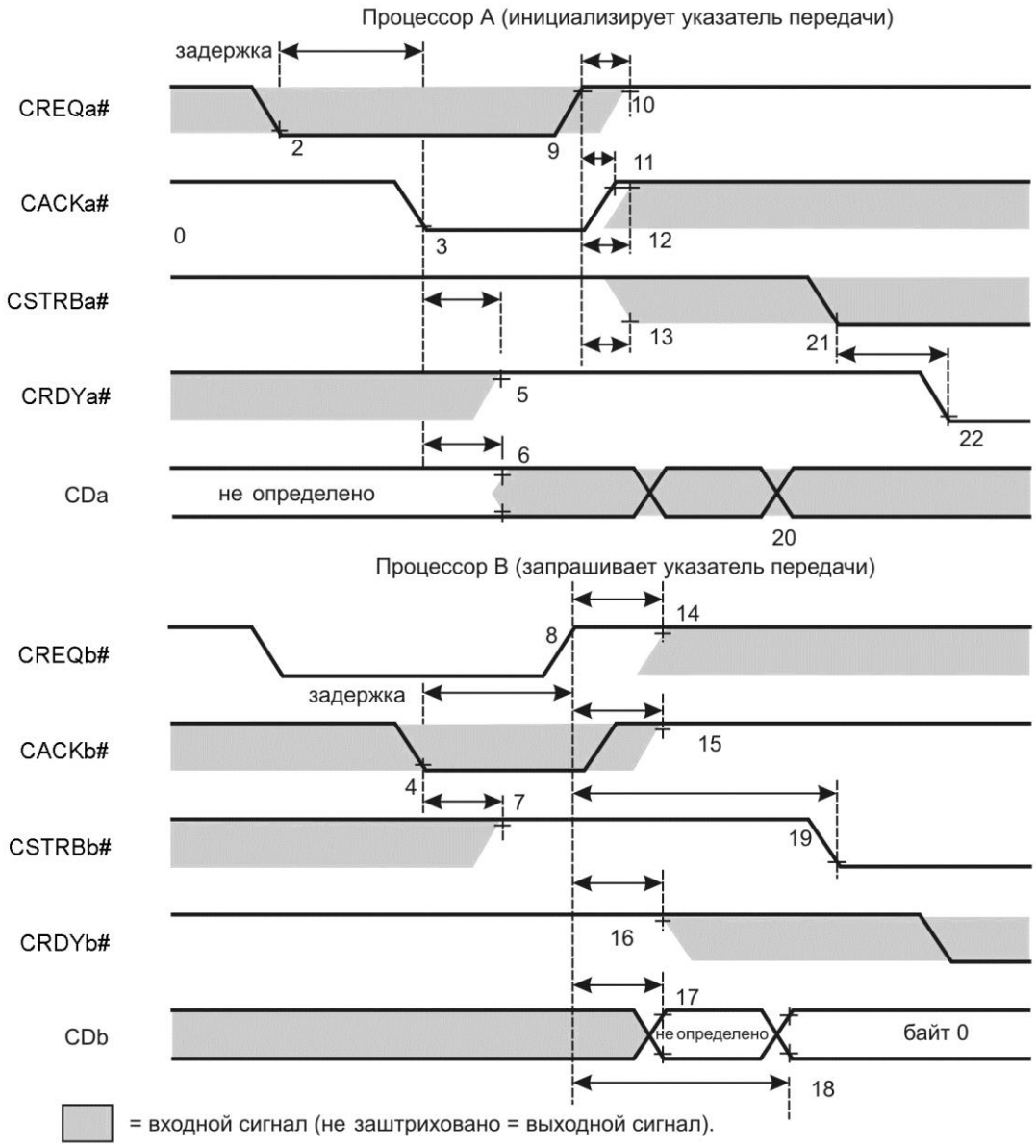


Рисунок 5.143 − Операция передачи указателя передачи

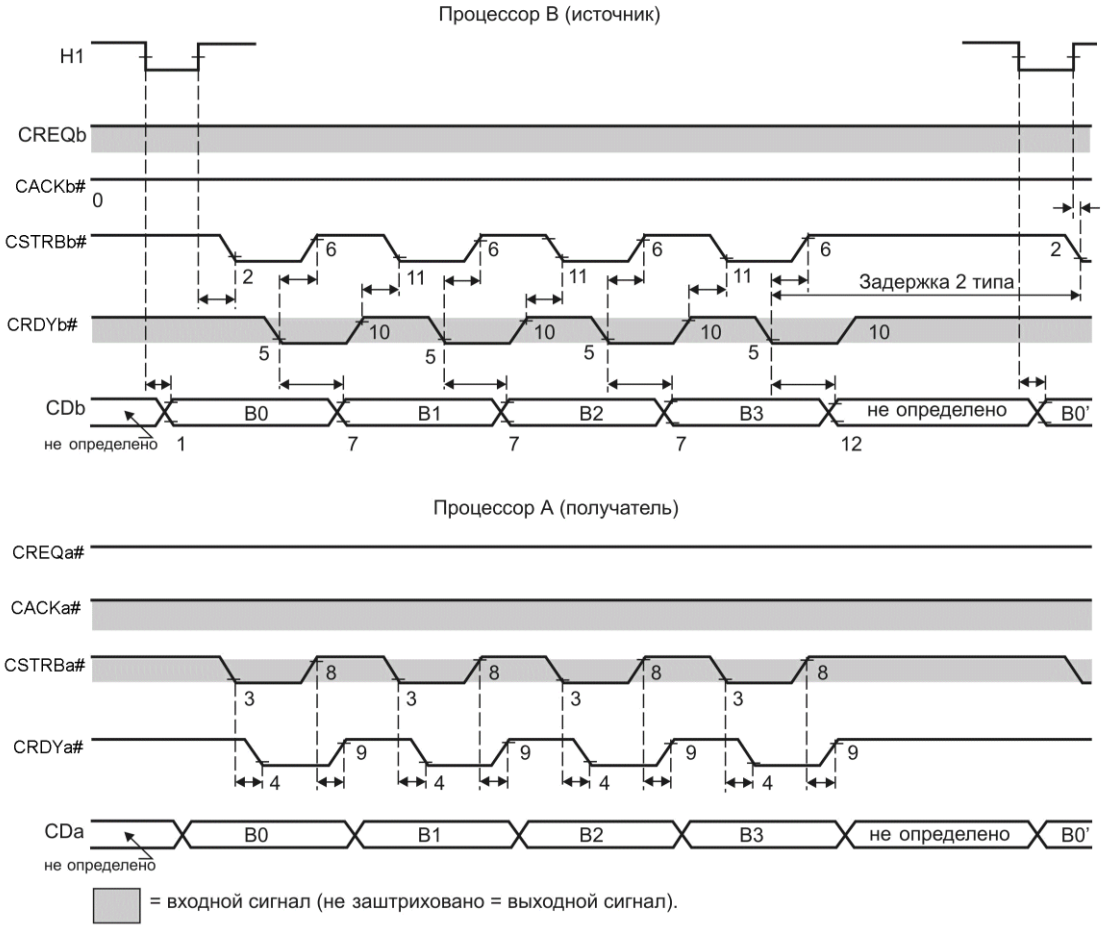
Таблица 5.62 − Операция передачи указателя передачи

Номер события	Описание
0	Первоначально A имеет указатель передачи и простаивает
1	B хочет отправить данные и запрашивает указатель передачи, устанавливая сигнал CREQb# в низкий уровень
2	После задержки A видит запрос указателя передачи, когда сигнал CREQa# переходит в низкий уровень
3	После 1 типа задержки от падения CREQa# A подтверждает запрос при помощи сброса сигнала CACKa# в низкий уровень
4	После задержки B видит 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	A переводит шину CDa(7−0) в высокоимпедансное состояние после падения CACKa# в низкий уровень
7	B переводит CSTRBb# из высокоимпедансного состояния в высокий уровень после падения CACKb# в низкий уровень
8	B переводит CREQb# в высокий уровень после 1 типа задержки от падения CACKb#
9	После задержки A видит, что CREQa# находится в высоком уровне
10	Сигнал CREQa# переходит от высокоимпедансного состояния до высокого после получения высокого состояния на CREQa#
11	A переводит CACKa# в высокий уровень после того, как CREQa# приходит в высоком уровне
12	A переводит CACKa# в высокоимпедансное состояние после того, как CREQa# станет высоким и после того, как CACKa# станет высоким
13	A переводит CSTRBa# в высокоимпедансное состояние после того, как CREQa# станет высоким
14	B переводит CREQb# в высокоимпедансное состояния после того, как CREQb# станет высоким
15	B переводит CACKb# от высокоимпедансного состояния до высокого после того, как CREQb# станет высоким
16	B переводит CRDYb# в высокоимпедансное состояния после того, как CREQb# станет высоким
17	B переводит шину CDb из входного в выходное состояние после того, как CREQb# станет высоким и начинает управлять неопределённым значением
18	B управляет первым байтом на шине CDb(7−0) на фронте H1 после того, как CREQb# станет высоким
19	B переводит CSTRBb# в низкий уровень на втором фронте H1 после фронта CREQb#
20	После задержки A видит первый байт на CDa(7−0)
21	После задержки A видит, что CSTRBa# переходит в низкий уровень, сообщая о достоверности данных
22	Чтение данных и передача CRDYa# с низким уровнем

5.12.8 Операция передачи слова

Коммуникационные порты ПЦОС передают слова побайтно, начиная с младшего байта. Операция передачи байта происходит с использованием сигналов CSTRB# и CRDY#. Пример передачи одного слова представлен на рисунке 5.144. Для ясности, суффикс идентифицирует сигналы, выходящие из соответствующего микропроцессора. Передача слова происходит следующим образом:

- процессор B выставляет младший байт на шине данных порта и через задержку выставляет CSTRBb# в низкий уровень, сообщая этим, что на шине выставлены достоверные данные;
- процессор A принимает CSTRBa# с низким уровнем, через задержку выставляет CRDYa# в низкий уровень и принимает данные;
- после принятия процессором B сигнала CRDYb# с низким уровнем, он через задержку восстанавливает CSTRBb# в высокий уровень;
- после принятия данных процессор A возвращает CRDYa# в высокий уровень;
- после принятия процессором B сигнала CRDYb# с высоким уровнем, он начинает передачу следующего байта.



Примечание – B0 = 0 байт нового слова.

Рисунок 5.144 − Операция передачи слова

5.12.9 Синхронизация

В течение времени на границе передачи слов и в течение передачи указателя передачи требуется синхронизация H1/H3. Арбитр порта включает три типа синхронизации:

- Синхронизация первого типа вызывает задержки, которые изменяются от одного до двух машинных циклов от получения сигнала на входе до ответа на выходе (игнорируя аналоговые задержки). Входной сигнал распознается, когда H1 в высоком уровне; затем он проходит через серию задержек H3/H1 высокого уровня. Ответ происходит при запуске следующего H3 в высоком уровне.

Минимальная задержка синхронизации первого типа (один машинный цикл) может происходить только тогда, когда входной сигнал изменяется перед тем, как H1 перейдёт в низкий уровень. Задержка показана на рисунке 5.145.

Максимальная задержка синхронизации первого типа (два машинных цикла) может происходить только тогда, когда входной сигнал изменяется после того, как H1 перейдёт в низкий уровень. Задержка показана на рисунке 5.146.

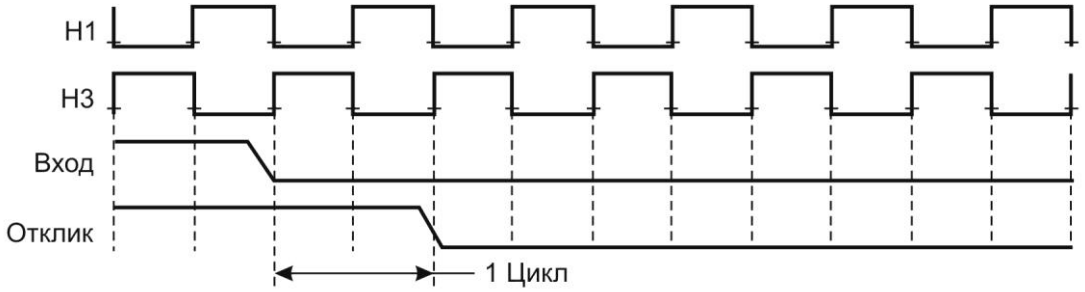


Рисунок 5.145 − Минимальная задержка синхронизации первого типа

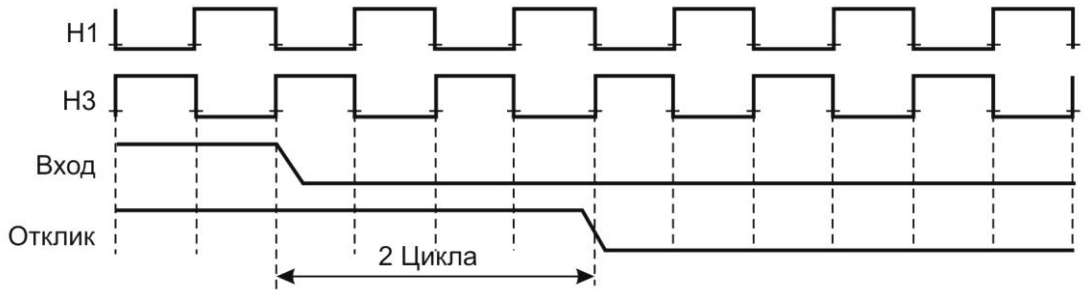


Рисунок 5.146 − Максимальная задержка синхронизации первого типа

- Синхронизация второго типа вызывает задержки, которые изменяются от 1,5 до 2,5 машинного цикла от получения сигнала на входе до ответа на выходе (игнорируя аналоговые задержки). Входной сигнал распознается, когда H1 в высоком уровне; затем он проходит через серию задержек H3/H1/H3 высокого уровня. Ответ происходит при запуске следующего H1 в высоком уровне.

Минимальная задержка синхронизации второго типа (1,5 машинного цикла) может происходить только тогда, когда входной сигнал изменяется перед тем, как H1 перейдёт в низкий уровень. Задержка показана на рисунке 5.147.

Максимальная задержка синхронизации второго типа (2,5 машинных цикла) может происходить только тогда, когда входной сигнал изменяется после того, как H1 перейдёт в низкий уровень. Задержка показана на рисунке 5.148.

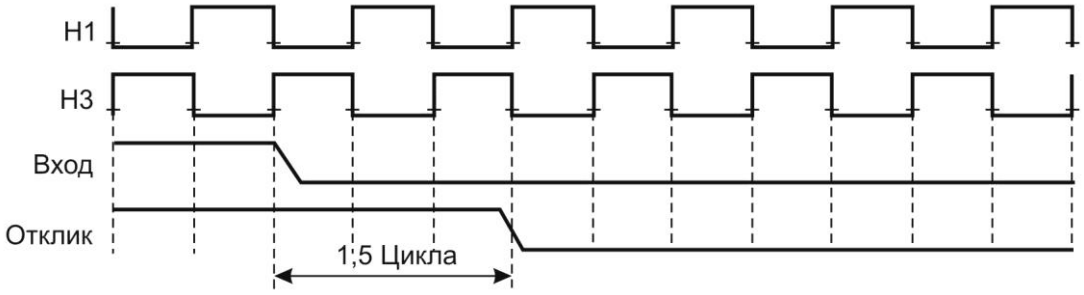


Рисунок 5.147 − Минимальная задержка синхронизации второго типа

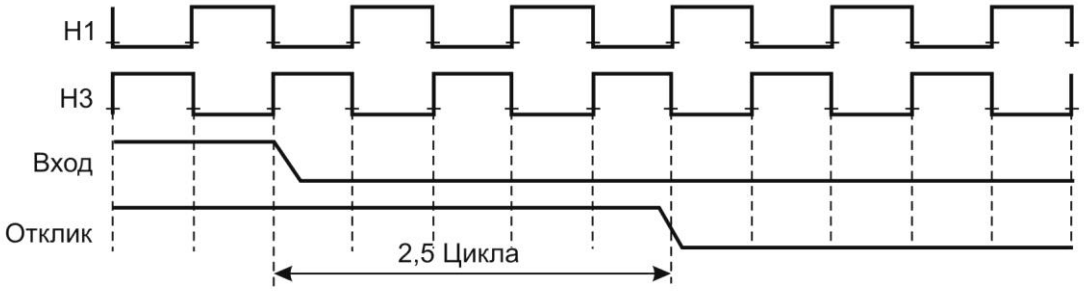


Рисунок 5.148 − Максимальная задержка синхронизации второго типа

- Синхронизация третьего типа вызывает 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 распознается, когда H1 в высоком уровне; затем он проходит через серию задержек H3 высокого уровня. Ответ происходит при запуске следующего H1 в высоком уровне.

Минимальная задержка синхронизации третьего типа (0,5 машинного цикла) может происходить только тогда, когда входной сигнал изменяется перед тем, как H1 перейдёт в низкий уровень. Задержка показана на рисунке 5.149.

Максимальная задержка синхронизации третьего типа (1,5 машинного цикла) может происходить только тогда, когда входной сигнал изменяется после того, как H1 перейдёт в низкий уровень. Задержка показана на рисунке 5.150.

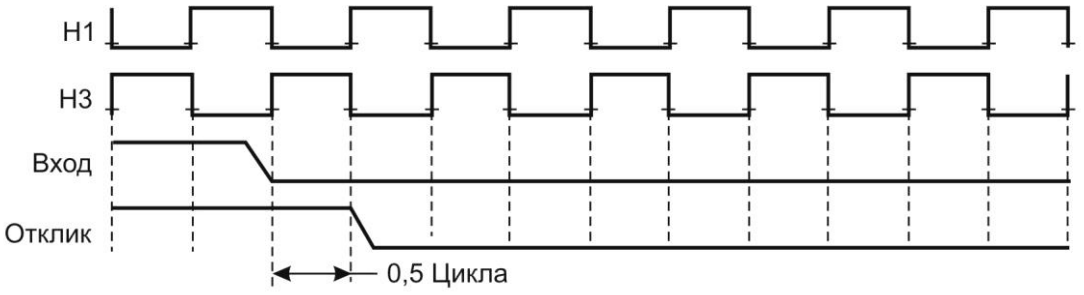


Рисунок 5.149 − Минимальная задержка синхронизации третьего типа

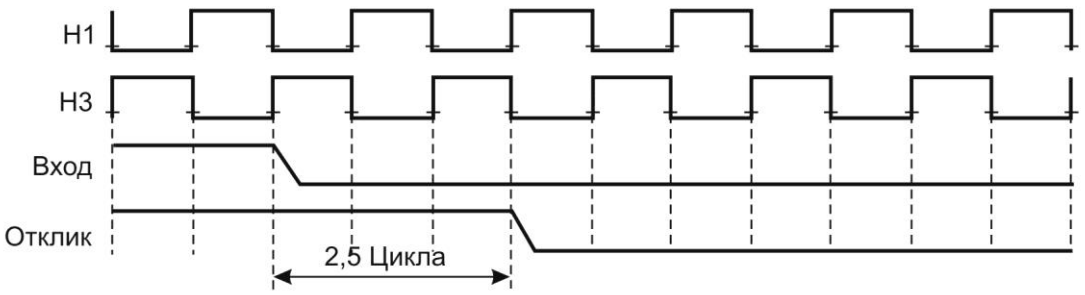


Рисунок 5.150 − Максимальная задержка синхронизации третьего типа

В таблице 5.63 приведены типы задержек синхронизации для сигналов коммуникационного порта.

Таблица 5.63 − 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, верная 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

5.12.10 Сброс

Далее описывается состояние коммуникационных портов процессорного ядра ИС 1867ВЦ8Ф1 после включения питания, а также во время и после системного сброса.

После включения питания состояние зависит от сигнала RESET_COMM#:

Если RESET_COMM# в низком уровне, ядро процессора 1867ВЦ8Ф1 сбрасывается немедленно, при этом применимо описание «при сбросе» (см. ниже).

Если RESET_COMM# не в низком уровне, ядро процессора 1867ВЦ8Ф1 будет находиться в неопределённом состоянии. Сигналы компорта могут иметь комбинацию состояний.

При сбросе (пока RESET_COMM# = 0), все выводы коммуникационного порта переводятся в высокоимпедансное состояние. Входные и выходные каналы очищаются, что приводит к потере значений во входных и выходных буферах. На всех линиях управления следует использовать подтягивающие резисторы, чтобы обеспечить на них высокий уровень, если сброс не применяется одновременно для всех процессоров многопроцессорной системы.

После сброса (по фронту RESET_COMM#) коммуникационные порты 0, 1, 2 конфигурируются как выходные и имеют следующие состояния:

- арбитр сбрасывается в 0: арбитр имеет маркер владения шиной и находится в состоянии ожидания;
- выводы устанавливаются в следующие состояния (смотри рисунок 5.151):
  - сигналы CxD(7−0) в неопределённое значение;
  - сигналы CACK# и CSTRB# в 1 (пассивны);
  - сигналы CREQ# и CRDY# остаются в высокоимпедансном состоянии.

Примечание – Функция программного сброса отдельного коммуникационного порта только очищает FIFO, но не оказывает никакого влияния на его внешние выводы.

Регистр управления коммуникационным портом сбрасывается в 0:
- PORT DIR = 0: коммуникационный порт конфигурируется для операций выхода;
- INPUT LEVEL = 0: входной буфер FIFO пуст;
- OUTPUT LEVEL = 0: выходной буфер FIFO пуст;
- ICH = 0: входной буфер FIFO не остановлен;
- OCH = 0: выходной буфер FIFO не остановлен;
- ICRDY = 0: входной буфер FIFO пуст и не готов для чтения из него;
- OCRDY = 0: выходной буфер FIFO не заполнен и готов для записи в него.

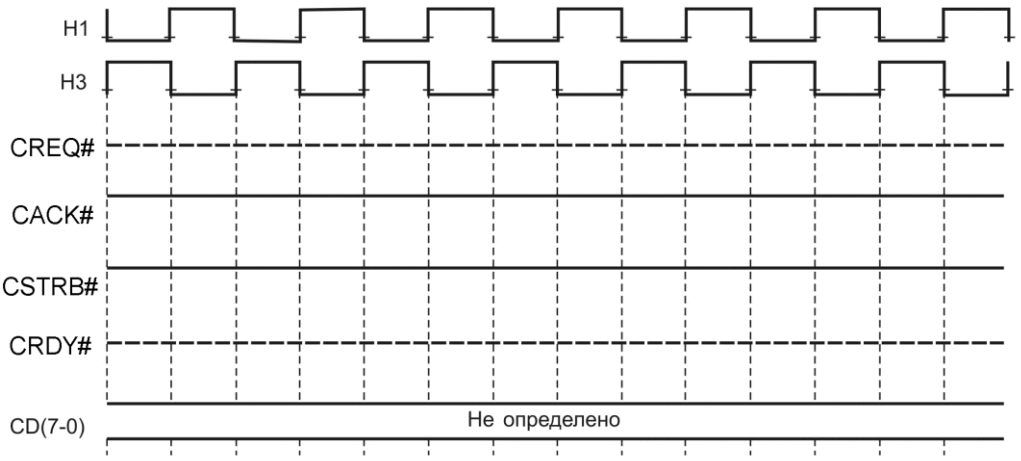


Рисунок 5.151 − Состояние после сброса выходного порта

После сброса (по фронту RESET_COMM#) коммуникационные порты 3, 4, 5 конфигурируются как входные и имеют следующие состояния:

- арбитр устанавливается в 1: арбитр не имеет маркера владения шиной и этот маркер не запрашивается;
- выводы устанавливаются в следующие состояния (смотри рисунок 5.152):
  - сигналы CxD(7−0) остаются в высокоимпедансном состоянии;
  - сигналы CREQ# и CRDY# в 1 (пассивны);
  - сигналы CACK# и CSTRB# остаются в высокоимпедансном состоянии.

Регистр управления коммуникационным портом устанавливается в 04h:
- PORT DIR = 1: коммуникационный порт конфигурируется для операций входа;
- INPUT LEVEL = 0: входной буфер FIFO пуст;
- OUTPUT LEVEL = 0: выходной буфер FIFO пуст;
- ICH = 0: входной буфер FIFO не остановлен;
- OCH = 0: выходной буфер FIFO не остановлен;
- ICRDY = 0: входной буфер FIFO пуст и не готов для чтения из него;
- OCRDY = 0: выходной буфер FIFO не заполнен и готов для записи в него.

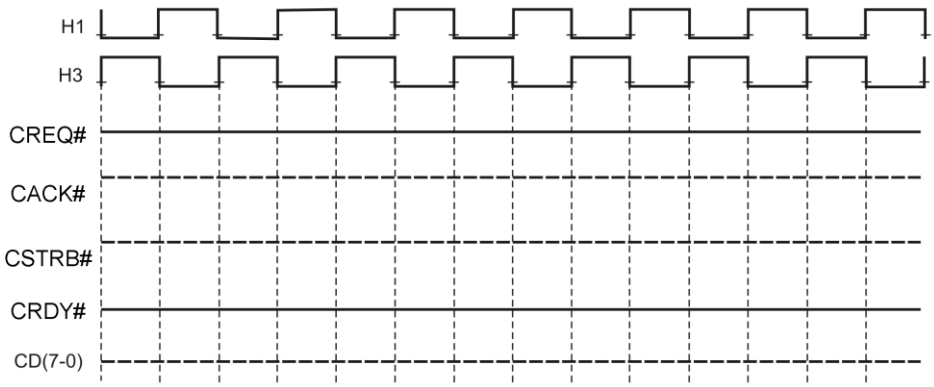


Рисунок 5.152 − Состояние после сброса входного порта

Примечание – При сбросе порты 0, 1 и 2 будут сконфигурированы как выходные (PORT DIR = 0), а порты 3, 4 и 5 – как входные (PORT DIR = 1). При соединении двух ПЦОС 1867ВЦ8Ф1 их порты необходимо подключить в противоположных друг другу направлениях передачи данных, то есть любой порт 0, 1 или 2 подключить к любому из портов 3, 4 или 5.

5.13 Таймеры

5.13.1 Обзор таймеров

Модули таймера процессорного ядра ИС 1867ВЦ8Ф1 – 32-разрядные счётчики общего назначения, работающие как таймер или счётчик событий с двумя режимами сигнализации и внутренним или внешним тактированием. Модули таймера могут быть использованы для выдачи через определённые временные интервалы сигналов процессорному ядру 1867ВЦ8Ф1 (или внешним устройствам) или для подсчёта внешних событий. Реализованный в каждом таймере двунаправленный внешний вход/выход может быть использован как его тактовый вход или как выходной сигнал синхронизации или как вход/выход общего назначения.

Таймер с внутренним тактированием может, например, инициировать старт преобразования внешним ЦАП или прервать контроллер ПДП ядра процессора 1867ВЦ8Ф1 для начала пересылки данных.

Каждый таймер состоит из 32-разрядного счётчика, компаратора, селектора входных тактовых сигналов, импульсного генератора и вспомогательного оборудования.

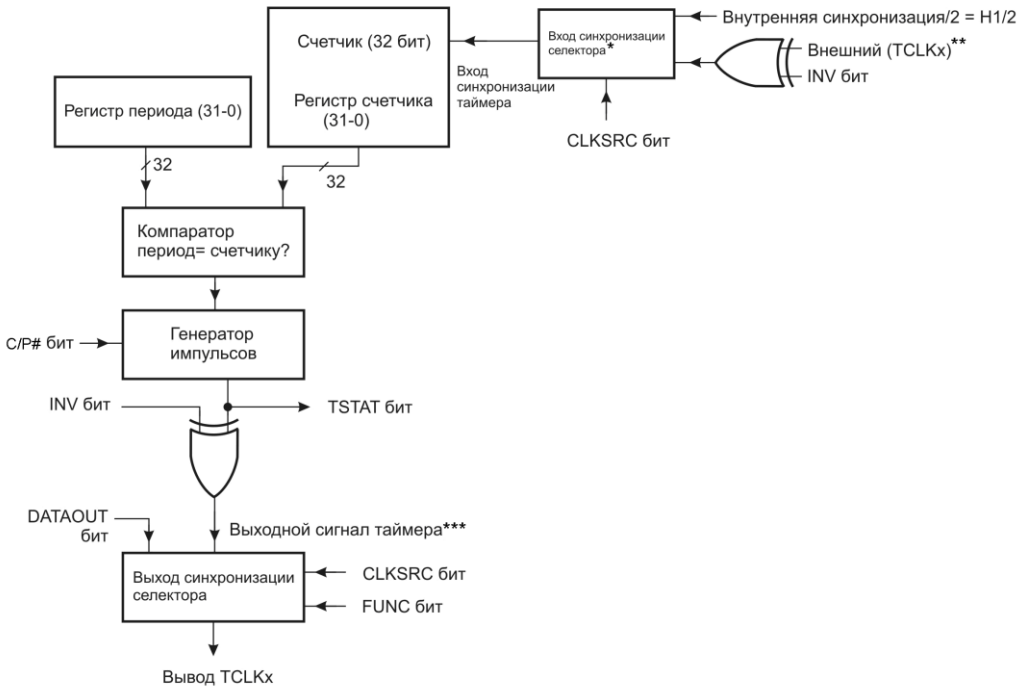
Таймер процессорного ядра 1867ВЦ8Ф1 подсчитывает циклы входного тактового сигнала. Когда этот счёт (регистр счётчика) доходит до значения, сохранённого в регистре периода, счётчик сбрасывается в 0 и генерирует выходной сигнал.

Входным тактовым сигналом таймера может быть либо делённый на два внутренний тактовый сигнал процессорного ядра 1867ВЦ8Ф1 (H1/2) либо внешние тактовые сигналы TCLK0_x, TCLK1_x (x = 1, 2). Это определяется разрядом CLKSRC в регистре управления таймером. Если используется внешняя синхронизация, таймер может считать как в прямом (от 0 к 1), так и в обратном (от 1 к 0) направлении, в зависимости от состояния разряда INV.

Выходной сигнал таймера зависит от режима сигнализации, выбираемого разрядом C/P (режим тактов или импульсов), см. 5.13.4.

Выход таймера может быть направлен через выводы TCLK0_x, TCLK1_x (x = 1, 2), которые могут использоваться также в качестве входов/выходов общего назначения.

На рисунке 5.153 представлена структурная схема модуля таймера процессорного ядра ИС 1867ВЦ8Ф1.



* Селектор управляется разрядом CLKSRC.

** Максимальная частота равна f(H1)/2,6.

*** Если CLKSRC = 1 и FUNC = 1, этот сигнал передаётся на TCLK

Рисунок 5.153 – Структурная схема таймера

5.13.2 Выводы таймеров

Каждый таймер имеет один вывод, связанный с выводом тактового сигнала таймера TCLK. Этот вывод используется как вход/выход общего назначения, как выход таймера или как вход внешней синхронизации для таймера. Каждый таймер имеет вывод TCLK: TCLK0_x относится к таймеру 0 и TCLK1_x относится к таймеру 1, где x = 1, 2.

5.13.3 Регистры управления таймером

Таймеры управляются тремя регистрами, показанными в таблице 5.64, где приведена их карта памяти:

- Регистр управления. Этот регистр определяет режим работы таймера, отражает его состояние и управляет функциональностью выводов входа/выхода TCLK0_x, TCLK1_x, где x = 1, 2.
- Регистр периода. Этот регистр определяет частоту выдачи сигналов таймером.
- Регистр-счётчик. Этот регистр содержит текущее значение инкрементируемого счётчика.

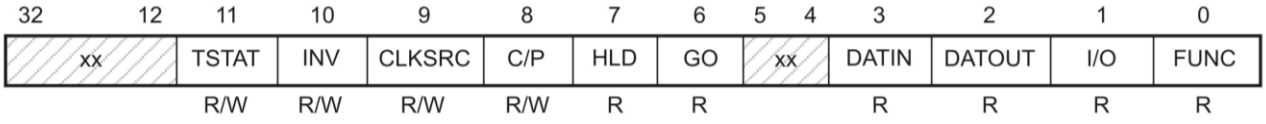
Таблица 5.64 − Адреса картированных в памяти регистров таймера

Регистр	Адрес периферии	
	Таймер 0	Таймер 1
Регистр управления таймером	100020h	100030h
Зарезервировано	100021h	100031h
Зарезервировано	100022h	100032h
Зарезервировано	100023h	100033h
Регистр счётчика таймера	100024h	100034h
Зарезервировано	100025h	100035h
Зарезервировано	100026h	100036h
Зарезервировано	100027h	100037h
Регистр периода таймера	100028h	100038h
Зарезервировано	100029h	100039h
Зарезервировано	10002Ah	10003Ah
Зарезервировано	10002Bh	10003Bh
Зарезервировано	10002Ch	10003Ch
Зарезервировано	10002Dh	10003Dh
Зарезервировано	10002Eh	10003Eh
Зарезервировано	10002Fh	10003Fh

5.13.3.1 Регистр управления таймером

Регистр управления таймером расположен по адресу 100020h для таймера 0 и по адресу 100030h для таймера 1.

Регистр глобального управления таймером − 32-разрядный регистр, содержащий разряды глобального управления и управления портом для модуля таймера. На рисунке 5.154 показаны разряды регистра, их имена и функции. Разряды (3−0) − разряды управления портом; разряды (11−6) − разряды глобального управления таймером. Необходимо обратить внимание, что при сбросе все разряды устанавливаются в 0, за исключением DATIN (устанавливается по значению на TCLK).



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

Рисунок 5.154 − Регистр управления таймером

FUNC – разряд функциональности. Разряд FUNC управляет функциональностью TCLK. Если FUNC = 0, TCLK конфигурируется как цифровой вход/выход общего назначения. Если FUNC = 1, TCLK конфигурируется как сигнал таймера.

I/O – разряд вход/выход. Если I/O = 1 и FUNC = 0, тогда TCLK конфигурируется как вход. Если I/O = 0 и FUNC = 0, тогда TCLK конфигурируется как выход.

DATAOUT – разряд выхода данных. DATAOUT управляет TCLK, когда таймер ядра процессора 1867ВЦ8Ф1 пребывает в режиме порта ввода-вывода. DATAOUT также может использоваться как вход таймера.

DATAIN – разряд входа данных. Вход данных на TCLK или DATOUT. Запись в этот разряд ни на что не влияет.

В таблице 5.65 описывается влияние разрядов CO и HLD регистра управление таймером на его операции.

GO – разряд GO. Разряд GO сбрасывает и запускает счётчик таймера. Когда GO = 1 и таймер не удерживается, счётчик обнуляется и начинает инкрементироваться по следующему возрастающему фронту тактового входа таймера. GO очищается по тому же возрастающему фронту. GO = 0 не влияет на таймер.

HLD – разряд удержания счётчика. Когда этот разряд равен нулю, счётчик запрещён и удерживается в его текущем состоянии. Когда таймер управляется TCLK, состояние TCLK тоже удерживается. Внутренний счётчик (с делением на два) тоже удерживается таким образом, что счёт может продолжиться с состояния останова при установке HLD в 1. Регистры таймера могут считываться и модифицироваться, пока счётчик удерживается. RESET_COMM# имеет более высокий приоритет, чем HLD.

Таблица 5.65 – Операция таймера в зависимости от состояния разрядов 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 счётчика. Разряд GO не очищается до того, как таймер выйдет из состояния удержания
1	1	Таймер сбрасывается и запускается

C/P# – управление режимом такт/импульс. Когда C/P = 1, выбирается режим такта, что указывает, что флаг состояния и внешний выход будут иметь цикл со скважностью 50 %. Когда C/P = 0, флаг состояния и внешний выход будут активными для одного цикла H1 в течение каждого периода таймера.

CLKSRC – определяет источник тактирования таймера. Когда CLKSRC = 1, для инкрементирования счётчика будет использоваться внутренний такт с частотой в половину частоты H1. Разряд INV не влияет на внутренний источник тактирования. Когда CLKSRC = 0, для инкрементирования счётчика может использоваться внешний сигнал с вывода TCLK. Внешний такт синхронизирован изнутри, обеспечивая, таким образом, возможность работы с внешними асинхронными источниками тактирования, которые не превышают максимально поддерживаемой частоты внешнего такта f(H1)/2,6.

INV – разряд управления инверсией. Если использован внешний источник тактирования и INV = 1, то внешний такт инвертируется при входе в счётчик. Если выход генератора импульсов подключен к TCLK и INV=1, выход инвертируется до вывода TCLK. Если INV = 0 инверсии не происходит ни на входе, ни на выходе таймера. При использовании TCLK, как входа/выхода порта, состояние INV, вне зависимости от его значения, не влияет на функционирование.

TSTAT – указывает на состояние таймера. Он отслеживает выход неинвертированного вывода TCLK. Этот флаг выставляет прерывание ЦПУ при переходе из 0 в 1. Запись в этот разряд ни на что не влияет.

5.13.3.2 Регистр периода таймера

Регистр периода таймера размещается по адресу 100028h для таймера 0 и по адресу 100038h для таймера 1.

32-разрядный регистр периода таймера используется для программирования частоты выдачи сигнала таймера.

Частота сигнализации таймера определяется частотой входной синхронизации таймера и регистром периода. Следующие уравнения верны для внутренней и внешней синхронизации:

f(режим импульса) = f(синхронизация таймера)/регистр периода;

f(режим такта) = f(синхронизация таймера)/регистр периода.

При сбросе регистр периода таймера сбрасывается в 0.

5.13.3.3 Регистр счётчика таймера

Регистр счётчика таймера размещается по адресу 100024h для таймера 0 и по адресу 100034h для таймера 1.

32-разрядный регистр счётчика таймера инкрементируется с каждым циклом входной синхронизации таймера. Счётчик 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 счётчик таймера инкрементируется только по переднему фронту. Счётчик обнуляется всякий раз, когда достигает значения регистра периода. При сбросе этот регистр сбрасывается в 0.

5.13.3.4 Граничные условия в регистрах управления

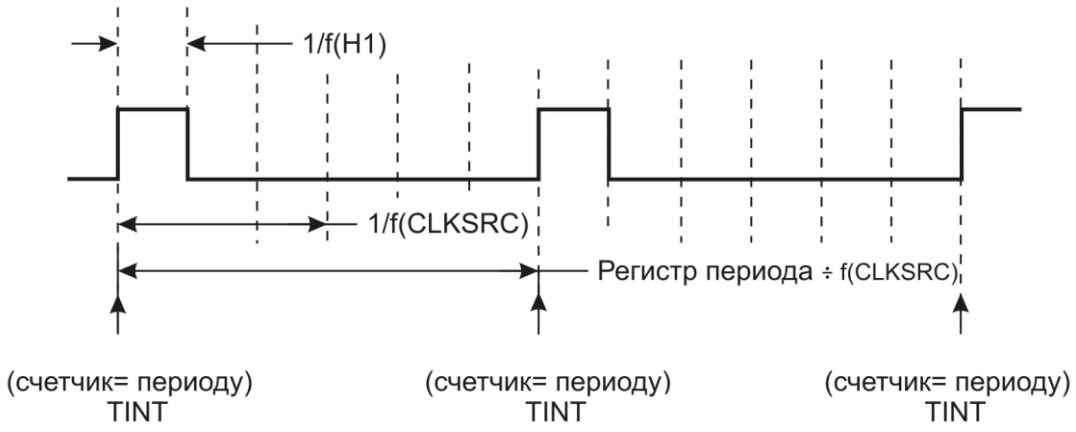
Определённые условия, такие как ноль в регистре периода и переполнение счётчика, влияют на работу таймера. Эти условия перечислены ниже:

- когда регистры периода и счётчика установлены в 0, работа таймера зависит от режима, выбранного через C/P. При выборе импульсного режима C/P = 0 TSTAT устанавливается и остаётся в установленном состоянии. При тактовом режиме C/P = 1 цикл составляет 2/f(H1), и внешний такт игнорируется;
- когда регистр счётчика не 0, а регистр периода равен 0, счётчик будет считать, перейдёт через 0 и будет вести себя как описано выше;
- когда регистр счётчика установлен на значение, большее значения регистра периода, счётчик может переполниться при инкрементировании. Когда счётчик достигнет максимального 32-разрядного значения 0FFFFFFFFh, он просто перейдёт через 0 и продолжит счёт.

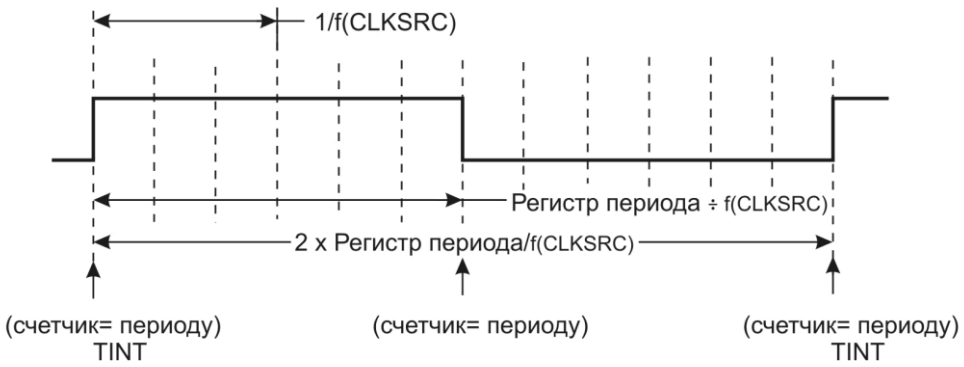
Примечание – Записи с периферийной шины имеют приоритет над изменениями регистра из счётчика, и новое состояние изменяет значение регистра управления.

5.13.4 Генерация импульсов таймера

Генератор импульсов таймера, см. рисунок 5.153, может генерировать несколько различных внешних сигналов. Эти сигналы могут быть инвертированы разрядом INV. На рисунке 5.155 приведены два основных режима: импульсный и тактовый. Источник внутреннего такта f(такт таймера) в обоих случаях имеет частоты f(H1)/2, а генерируемый извне источник такта f(такт таймера) может иметь максимальную 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, делённому на частоту входной синхронизации.



(a) TSTAT и выход таймера (INV = 0), когда C/P# = 0 (импульсный режим)



(b) TSTAT и выход таймера (INV = 0), когда C/P# = 1 (тактовый режим)

Рисунок 5.155 − Синхронизация таймера

Период выдачи сигнала таймера определяется частотой его входного тактирования 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, смотри 5.13.3.2.

На рисунке 5.156 приведены несколько примеров выхода TCLK при установке различных значений регистра периода и выборе импульсного или тактового режима. Синхронизация таймера – внутренняя (f(H1), f(H2)).

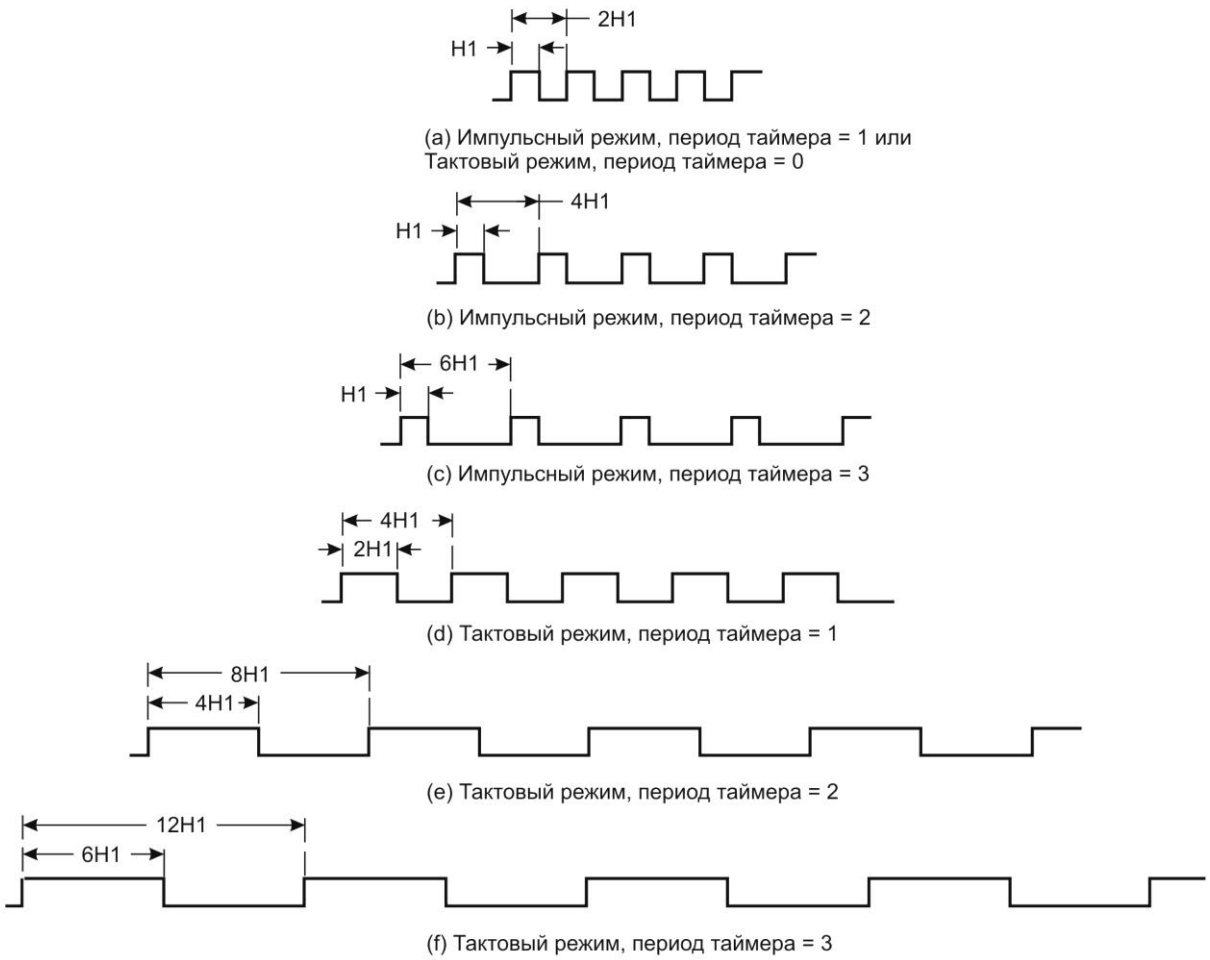


Рисунок 5.156 − Примеры генерации выхода таймера

5.13.5 Прерывания таймера

Каждый таймер может послать сигнал прерывания ЦПУ, когда сигнал TSTAT изменяется из 0 в 1. Таймер 0 посылает TINT0, а таймер 1 посылает TINT1.

5.13.5.1 Прерывания таймера и их векторы

TINT0 относится к таймеру 0. Это прерывание имеет вектор прерывания с адресом IVTP+002h и приоритет второго уровня, т. е. после NMI# и RESET_COMM#.

TINT1 относится к таймеру 1. Это прерывание имеет вектор прерывания с адресом IVTP+02Bh и низший приоритет из всех прерываний.

5.13.5.2 Операция прерывания таймера

Прерывание 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:

f(прерывания) = f(такта таймера)/регистр периода

В режиме такта, частота прерывания:

f(прерывания) = f(такта таймера)/(2 × регистр периода)

Если регистр периода равен 0, см. 5.13.3.4.

Прерывания таймера могут использоваться для прерываний ЦПУ или сопроцессора ПДП.

Разряды разрешения прерываний таймера для ЦПУ находятся в IIE регистре. Разряд 0 в IIE относится к TINT0, а разряд 1 относится к TINT1. Для более подробной информации о IIE регистре см. 5.3.1.9.

Разряды разрешения прерываний таймера для регистра управления ПДП находятся в DIE регистре. Некоторые разряды в этом регистре управляют ответами канала ПДП на сигналы от таймеров. Для более подробной информации о DIE регистре см. 5.13.3.4.

5.13.5.3 Использование прерываний таймера

Основное внимание при использовании таймеров для прерываний процессорного ядра необходимо уделить приоритету этой операции. Если требуется низкий приоритет прерывания по сравнению с другими устройствами, используйте таймер 1, так как прерывание этого таймера имеет самый низкий приоритет среди всех прерываний. Если необходим более высокий приоритет прерывания по сравнению с другими устройствами, используйте таймер 0, так как прерывание этого таймера является вторым по приоритету после NMI#.

5.13.6 Выборка значений CLKSRC и FUNC

Таймер может быть синхронизирован в различных режимах в зависимости от CLKSRC, FUNC, I/O. Четыре режима таймера определяются значениями CLKSRC и FUNC в глобальном регистре управления.

5.13.6.1 CLKSRC = 1 и FUNC = 0

Если CLKSRC = 1 и FUNC = 0 (см. рисунок 5.157), на вход таймера подаётся внутренняя синхронизация. Прерывания также генерируются при переходе TSTAT из 0 в 1. Разряд INV глобального 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, TCLK конфигурируется как вход общего назначения, при этом его состояние может быть считано из DATIN. DATOUT не влияет на TCLK или DATIN. Если I/O = 1, TCLK конфигурируется как выход общего назначения. DATOUT устанавливается на TCLK и может быть считан из DATIN.

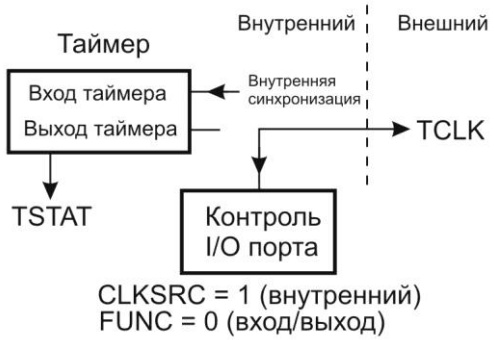


Рисунок 5.157 − Конфигурация таймера при CLKSRC = 1 и FUNC = 0

5.13.6.2 CLKSRC = 1 и FUNC = 1

Если CLKSRC = 1 и FUNC = 1 (см. рисунок 5.158), на вход таймера приходит внутренняя синхронизация, а выход подключается к TCLK. Значение TCLK можно инвертировать, устанавливая INV в 1. Также значение TCLK может быть считано из DATIN.

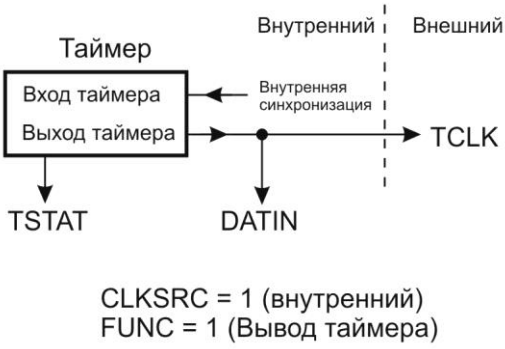


Рисунок 5.158 − Конфигурация таймера при CLKSRC = 1 и FUNC = 1

5.13.6.3 CLKSRC = 0 и FUNC = 0

Если CLKSRC = 0 и FUNC = 0 (см. рисунок 5.159), таймер продолжает генерировать сигналы прерывания и управляется в соответствии с разрядом I/O:

- если I/O = 0, таймер синхронизируется от TCLK. Значение TCLK можно инвертировать, устанавливая INV в 1. Также значение TCLK может быть считано из DATIN;

- если I/O = 1, TCLK становится выходом, и TCLK, и таймер управляются DATOUT. Все переходы DATOUT из 0 в 1 инкрементируют счётчик. INV не влияет на DATOUT. Значение DATOUT может быть считано из DATIN.

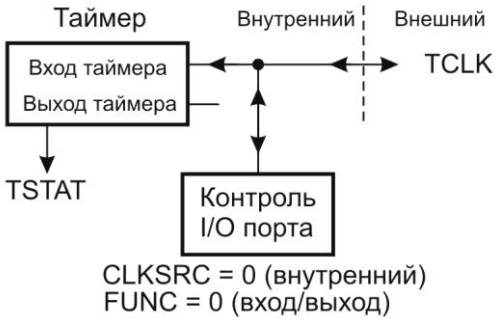


Рисунок 5.159 − Конфигурация таймера при CLKSRC = 0 и FUNC = 0

5.13.6.4 CLKSRC = 0 и FUNC = 1

Если CLKSRC = 0 и FUNC = 1, TCLK управляет таймером. Если INV = 0, все переходы TCLK из 0 в 1 инкрементируют счётчик. Если INV = 1, все переходы TCLK из 1 в 0 инкрементируют счётчик. Значение TCLK может быть считано из DATIN.

5.13.7 Использование TCLK в качестве входов/выходов общего назначения

Если FUNC = 0, TCLK0_x, TCLK1_x (x = 1, 2) могут использоваться в качестве входов/выходов общего назначения.

На рисунках 5.160 и 5.161 изображено подключение TCLK в случае использования их в качестве входов/выходов общего назначения. На рисунке 5.160 I/O = 0 и TCLK конфигурируется как вход, значение которого может быть считано из разряда DATIN. На рисунке 5.161 I/O = 1 и TCLK конфигурируется как выход, значение которого записывается в разряд DATOUT.

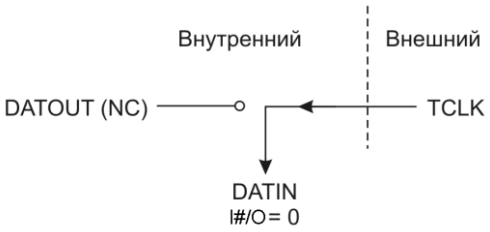


Рисунок 5.160 − TCLK как вход (I/O = 0)

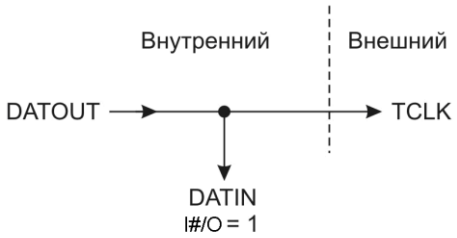


Рисунок 5.161 − TCLK как выход (I/O = 1)

5.13.8 Конфигурация таймера

Таймер конфигурируется следующим образом:

- остановить таймер, очистив разряды GO/HLD в регистре глобального управления таймера. Чтобы сделать это, необходимо записать 0 в регистр глобального управления таймера. Необходимо обратить внимание, что таймер останавливается по RESET_COMM#;
- сконфигурировать таймер через регистр глобального управления таймера (с GO = HLD = 0), регистр счётчика таймера и регистр периода таймера, если необходимо;
- запустить таймер установкой разрядов GO/HLD в 1 в регистре глобального управления таймера.

Пример 5.67 показывает, как установить таймер ядра процессора 1867ВЦ8Ф1 для генерации максимальной частоты тактового сигнала на выводах TCLK0_x, TCLK1_x (x = 1, 2).

Пример 5.67 – Установка максимальной тактовой частоты таймера

```
* Установка максимальной тактовой частоты таймера.
* Пример показывает, как установить таймер для генерации максимальной
* тактовой частоты, используя внутреннюю синхронизацию.
* Секция «TIMR_REGISTER» располагается, начиная с 100020h.

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
```

6 Описание коммутатора

Коммутатор предназначен для коммутации (подключения) периферийных устройств (UART, Ethernet 10/100, MIL-STD-1553и USB 2.0) к одному из ПЦОС. Коммутация периферийных устройств осуществляется программно и может быть выполнена любым из ПЦОС в любое время.

Для обеспечения доступа одного из ПЦОС к периферийному устройству необходимо установить соответствующий код коммутации в регистре коммутации RC. Доступ к регистру RC осуществляется через механизм доступа к разделяемым ресурсам, предусмотренным в ядре ПЦОС. Запись/модификация кода в регистре RC разрешена в случае, если содержимое регистра семафора RS равно 0 и запрещена, если содержимое регистра семафора RS равно 1. Чтение из регистра коммутации RC одним из процессоров разрешено в любой момент времени при любом значении регистра семафора RS (занят/свободен).

Процедура модификации регистра RC состоит из четырёх фаз:

- Фаза 1. Проверка состояния регистра семафора RS.
- Фаза 2. Захват регистра коммутации RC.
- Фаза 3. Модификация регистра коммутации RC.
- Фаза 4. Освобождение регистра коммутации RC.

Предоставление регистра коммутации RC запрашивающему ПЦОС осуществляется по принципу «первый пришёл – первый обслужен». Если оба ядра одновременно затребовали разделяемый ресурс (выполняют инструкцию LDII**)**, то разделяемый ресурс предоставляется ядру ПЦОС 1.

После сброса все устройства отключены от ПЦОС, и сигналы с выводов IIOF3_1#, IIOF2_1#, IIOF1_1# и IIOF0_1# поступают на ПЦОС 1, а сигналы с выводов IIOF3_2#, IIOF2_2#, IIOF1_2# и IIOF0_2# поступают на ПЦОС 2 и функционируют как входы/выходы общего назначения. Это даёт возможность осуществить старт ядер ПЦОС стандартным для них способом, за исключением старта с адресов локальной шины.

Для включения одного из выводов IIOF(3-0)_x# в режим обработки прерывания необходимо установить соответствующий бит FUNCx в «1» в регистре Interrupt Flag Register IIF, см. раздел 5.

Все периферийные устройства самостоятельно сбрасывают сигнал запроса на прерывание, поэтому в ПЦОС 1 и ПЦОС 2 захват прерывания от периферийного устройства должен осуществляться по фронту (бит TYPEx = 0 в регистре Interrupt Flag Register IIF), см. раздел 5.

После коммутации периферийного устройства к одному из ПЦОС прерывания от скоммутированного устройства поступают на соответствующие входы прерываний ПЦОС, см. таблицу 6.1, и отключают выводы IIOF(3-0)_x# от внешних выводов ИС 1867ВЦ8Ф1.

Таблица 6.1 – Приоритеты и номера прерываний, поступающие от периферийных устройств

Устройство	Прерывание	Приоритет
USB 2.0	IIOF0_x#	самый высокий
Ethernet	IIOF1_x#	высокий
MilStd 1553	IIOF2_x#	низкий
UART16550	IIOF3_x#	самый низкий
Примечание – x = 1 или 2.		

6.1 Архитектура коммутатора

Коммутатор состоит из двух программно доступных регистров:

- Регистр семафора RS.

Регистр семафора RS предназначен для доступа к регистру коммутации RC. Процессор, который подключает к своей локальной шине (LB_1/LB_2) одно из внешних устройств (UART, USB2.0, Ethernet 10/100, MilStd 1553) должен выполнить инструкции LDII, которые формируют на локальной шине сигнал LLOCK_1/LLOCK_2. Если арбитр коммутатора предоставляет доступ ПЦОС 1 или ПЦОС 2 к регистру коммутации RC, то для этого процессора содержимое RS будет равно 0x0000_0000. Для другого процессора в это время содержимое RS будет равно 0x0000_0001. Содержимое RS равно 0x0000_0000 будет означать, что доступ к регистру RC разрешён и данный процессор может модифицировать его содержимое.

- Регистр коммутации RC.

Регистр коммутации RC предназначен для коммутации (подключения) любого из внешних устройств к своей локальной шине, путём занесения соответствующего кода в этот регистр. После сброса, внешние устройства не подключены ни к одному из процессоров, а выводы IIOF(3-0)_x# подключены к соответствующим выводам. Если внешнее устройство подключено к процессору, то сигнал прерывания от этого внешнего устройства подключается к соответствующему выводу сигнала прерывания процессора. В таблице 6.1 приведено распределение прерываний между внешними устройствами.

6.1.1 Описание регистров коммутатора

В таблице 6.2 приведён список регистров коммутатора, в таблице 6.3 приведена структура регистра семафора RS и в таблице 6.4 приведена структура регистра коммутации RC.

Таблица 6.2 – Список регистров коммутатора

Наименование	Адрес
RS	0x00030 0000
RC	0x00030 0001

Таблица 6.3 – Структура регистра семафора RS

31-1	0
0	1
R	RC

Примечание – R – возможно чтение этого бита, C – бит сбрасывается, 0/1 – значение после сброса.

Таблица 6.4 – Структура регистра коммутации RC

31-8	7	6	5	4	3	2	1	0
rsrv	CMILSTD1	CMILSTD0	CUSB1	CUSB0	CETH1	CETH0	CUART1	CUART0
0	0	0	0	0	0	0	0	0
R	RWC	RWC	RWC	RWC	RWC	RWC	RWC	RWC

Примечание – R – возможно чтение этого бита, W – возможна запись в этот бит, C – бит сбрасывается, 0 – значение после сброса

6.1.2 Описание битов регистров RS и RC

Таблица 6.5 – Описание битов регистра семафора RS

Бит	Имя	Доступ	Описание
31–1	rsrv	R	Резервные биты. Читаются как 0
0	S	RC	Семафор регистра коммутации S = 0. Запись в регистр коммутации RC разблокирована S = 1. Запись в регистр коммутации RC заблокирована Чтение из этого регистра любой инструкцией кроме инструкции LDII всегда даёт значение в этом бите «1» После сброса S = 1. Для доступа к регистру коммутации RC необходимо разблокировать запись в этот регистр. Для этого необходимо выполнить команду LDII @RS, RX. Если содержимое регистра RS = 0x0000_0000, то запись в регистр коммутации RC разблокирована. После записи в регистр коммутации RC нужного значения необходимо выполнить команду STII RX, @RS, которая разблокирует запись в регистр коммутации RC и даёт возможность модифицировать регистр коммутации от другого процессора

Таблица 6.6 – Описание битов регистра коммутации RC

Бит	Имя	Доступ	Описание
1	2	3	4
31–8	rsrv	R	Резервные биты. Читаются как 0
7–6	CMILSTD1– CMILSTD0	RWC	Поле коммутации MIL-STD-1553 CMILSTD = 01 – MIL-STD-1553 подключен к ПЦОС 1. CMILSTD = 10 – MIL-STD-1553 подключен к ПЦОС 2. CMILSTD = 00/11– MIL-STD-1553 не подключен ни к ПЦОС 1, ни к ПЦОС 2. После сброса поле CMILSTD = 00 Системный клок (H1_1 или H1_2) не поступают на устройство. Примечание – При обращении к невыбранному устройству (на чтение или запись) сигналы LRDY_1 или LRDY_2 не формируются. Запись в это поле заблокирована, если бит S = 1
5–4	CUSB1– CUSB0	RWC	Поле коммутации USB 2.0 CUSB = 01 – USB 2.0 подключен к ПЦОС 1. CUSB = 10 – USB 2.0 подключен к ПЦОС 2. CUSB = 00/11 – USB 2.0 не подключен ни к ПЦОС 1, ни ПЦОС 2. После сброса поле CUSB = 00 Системный клок (H1_1 или H1_2) не поступают на устройство. Примечание – При обращении к невыбранному устройству (на чтение или запись) сигналы LRDY_1 или LRDY_2 не формируются. Запись в это поле заблокирована, если бит S = 1
3–2	CETH1– CETH0	RWC	Поле коммутации Ethernet CETH = 01 – Ethernet подключен к ПЦОС 1. CETH = 10 – Ethernet подключен к ПЦОС 2. CETH = 00/11 – Ethernet не подключен ни к ПЦОС 1, ни к ПЦОС 2. После сброса поле CETH = 00 Системный клок (H1_1 или H1_2) не поступают на устройство. Примечание – При обращении к невыбранному устройству (на чтение или запись) сигналы LRDY_1 или LRDY_2 не формируются. Запись в это поле заблокирована, если бит S = 1

Окончание таблицы 6.6

1	2	3	4
1–0	CUART1–CUART0	RWC	Поле коммутации UART CUART = 01 – UART и UART PLL подключены к ПЦОС 1. CUART = 10 – UART и UART PLL подключены к ПЦОС 2. CUART = 00/11 – UART и UART PLL не подключены ни к ПЦОС 1, ни к ПЦОС 2. После сброса поле CUART = 00. Системный клок (H1_1 или H1_2) не поступают на устройство. Примечание – При обращении к невыбранному устройству (на чтение или запись) сигналы LRDY_1 или LRDY_2 не формируются. Запись в это поле заблокирована, если бит S = 1.
Примечание – R – возможно чтение этого бита, W- возможна запись в этот бит, C – бит сбрасывается, 0 – значение после сброса.			

6.2 Пример программирования коммутатора

Для коммутации одного или нескольких устройств (UART, USB, MIL-STD-1553 или Ethernet) необходимо выполнить следующую последовательность действий:

- прочитать регистр семафора RS, используя инструкцию LDII;
- модифицировать соответствующие биты регистра RC для коммутации необходимых устройств;
- разблокировать регистр коммутации RC, для этого в RS необходимо записать любые данные с помощью инструкции STII.

Ниже приведён пример кода коммутации USB 2.0 к процессору CPU0.

Подпрограмма GetRS читает регистр RS до тех пор, пока значение регистра RS не станет равным нулю, после этого работа подпрограммы прекращается.

```
;////////////////////////////////////////////////Начало фрагмента программы////////////////////////////////
GetRS:
  ; загрузка адреса RC в AR7
  ldhi 30h, AR7; загружаем старшие 16 бит
  addi 1, AR7; AR7 = 30 0001h
  ; загрузка адреса RS в AR6
  ldhi 30h, AR6; загружаем старшие 16 бит

L1: ldii *AR6, R0
     cmpi 0, R0;
     bnz L1; переход на метку L1, если RS занят
  rets
;////////////////////////////////////////////////Конец фрагмента программы////////////////////////////////
```

Коммутация устройства USB осуществляется с помощью следующего макроса.

```
;////////////////////////////////////////////////Начало фрагмента программы////////////////////////////////
ConnectUSB0   .macro
  call GetRS; захват семафора
  ldi *AR7, R1; загружаем значение RC в R1
; модификация R1
; установка в 01 бит 05-04 для коммутации USB
            and 0FFDFh, R1
  or 010h, R1
; модифицируем RCC новым значением
  sti R1, *AR7
  .endm
;////////////////////////////////////////////////Конец фрагмента программы////////////////////////////////
```

После выполнения данного макроса можно обращаться к регистрам и памяти USB.

Для разблокировки возможности записи в RC можно воспользоваться следующим макросом.

```
;////////////////////////////////////////////////Начало фрагмента программы////////////////////////////////
DisConnectUSB          .macro
  ldhi 30h, AR0
  stii R1, *AR0; разблокировка регистра коммутации
  .endm
;////////////////////////////////////////////////Конец фрагмента программы////////////////////////////////
```

7 Периферийное устройство Ethernet 10/100

7.1 Описание периферийного устройства Ethernet 10/100

Внешнее периферийное устройство Ethernet 10/100 (далее – устройство Ethernet 10/100) осуществляет сопряжение с интерфейсом MII (интерфейс для связи с устройством формирования сигналов линии связи 10/100 Мб/с в соответствии со стандартом IEEE Std 802.3u – 1995), а также выполняет функции управления потоками данных в соответствии со стандартом IEEE Std 802.3x – 1997, включая управление интерфейсами MACMII, RMII, SMII, PMD, ENDEC. Устройство Ethernet 10/100 использует 32-битный интерфейс для связи с ПЦОС 1 или ПЦОС 2 и осуществляет обмен транслируемыми данными с ПЦОС 1 или ПЦОС 2 через оперативную память, имеющей структуру 16K × 32 (16K 32-разрядных слов). Для накопления и формирования принимаемых и передаваемых пакетов имеются два FIFO: для приёма – 4K × 36, для передачи – 2K × 40. Обмен данными с ПЦОС 1 или ПЦОС 2 осуществляется с частотой до 50 МГц. Обмен данными с устройством, работающим на физическую линию. PHY с частотой – 25 МГц для 100 Мбитного режима, 2,5 МГц для 10 Мбитного режима.

Интерфейсный блок устройства Ethernet 10/100 содержит контроллер прямого доступа к памяти ПДП, имеющий два канала, которые используются для операций Transmit и Receive. В устройстве ПДП оба канала конкурируют за использование контроллера прямого доступа в память, реализуя циклический алгоритм обслуживания конкурирующих запросов (round-robin priority algorithm).

При типовой передаче данных в любом направлении используется кольцевой буфер в пределах предназначенной для устройства Ethernet 10/100 памяти объёмом 16K × 32. Кольцевой буфер для операций Transmit определён закрытым связанным списком Tx-дескрипторов. Кольцевой буфер для операций Receive определён закрытым связанным списком Rx-дескрипторов. Два кольцевых буфера формируются из равноразмерных 32-разрядных сегментов памяти, способных сохранять пакет максимальной длины. Эти кольцевые буфера должны быть кратными 1 Кбайтной области памяти, и располагаться последовательно, не затрагивая области других компонент (область дескрипторов и область кольцевых буферов) устройства Ethernet 10/100.

Предварительно проинициализированный, контроллер ПДП может автономно и непрерывно заполнять/освобождать указанные кольцевые буфера. Программное обеспечение может использовать систему прерываний или опрос флагов дескрипторов для поддержания синхронизации потоков данных между устройством Ethernet 10/100 и ПЦОС 1 или ПЦОС 2.

7.1.1 Операции передачи

Перед передачей пакета, должна быть записана группа Tx дескрипторов, определяющих кольцевой буфер для операций передачи. Стартовые адреса начала всех сегментов должны быть 32-битные, сегменты равные по размерам должны быть достаточны для обработки пакета максимальной длины. Кроме того, в поле PacketSize дескриптора передачи должна быть записана длина пакета, а 31 бит Empty Flag должен быть установлен в 1, чтобы указать, что кольцевой буфер пока не содержит достоверных данных.

Четыре младших бита Регистра Прерываний DMAInterrupt так же должны быть установлены, чтобы специфицировать типы генерируемых прерываний.

После этого процессоры должны записать в кольцевые буфера и в дескрипторы, которые связаны с этими сегментами памяти, данные для передачи одного или более пакетов. Затем записывается в поле PacketSize длину пакета и бит Empty Flag очищается, сигнализируя контроллеру ПДП о наличии достоверных данных для передачи. Далее данные в кольцевые буфера разрешено записывать, если бит Empty Flag соответствующего дескриптора установлен в «1». Местоположение точки входа в кольцевой буфер указывается адресом соответствующего дескриптора в регистре DMATxDescriptor.

Для начала передачи необходимо установить TxEnable бит в «1». В этом случае контроллер ПДП прочтёт DMATxDescriptor, определит адрес стартового дескриптора, прочтёт дескриптор, если Empty Flag равен «0», затем определит размер пакета и адрес начала буферного сегмента. Если Empty Flag равен «1», то дескриптор не связан с достоверными данными. В этом случае ПДП контроллер прекращает последовательную передачу пакетов, устанавливает TxUnderrun бит в DMATxStatus регистре и очищает TxEnable бит в DMATxCtr регистре. Если разрешено прерывание, то оно будет сгенерировано по флагу TxUnderrun. Для возобновления передачи потребуется обновить регистр DMATxCtr и установить бит TxEnable в «1».

Передача пакета будет стартовать, если FIFO подтвердит контроллеру ПДП о наличии в FIFO достаточного места для приёма передаваемого пакета максимальной длины. Если передача завершена успешно, то контроллер ПДП запишет «1» в бит 31 компоненты PacketSize дескриптора, установит TxPktSent в DMATxStatus регистре, а также сгенерирует TxPktSent прерывание, если оно разрешено, и увеличит на 1 число, записанное в поле TxPktCount регистра DMATxStatus. После этого контроллер ПДП перейдёт к обработке пакета, сохранённого в следующем сегменте кольцевого буфера. Адрес нового дескриптора будет выбираться из компоненты NextDescriptor текущего дескриптора.

Если при передаче в канале связи произойдёт ошибка, контроллер ПДП прекратит последовательную передачу пакетов, установит бит BusError в 1 в DMATxStatus регистре и очистит TxEnable бит в DMATxCtrl регистре, сгенерирует BusError прерывание, если это прерывание разрешено. Для последующей передачи потребуется обновление DMATxStatus регистра, для того чтобы установить новую стартовую позицию в кольцевом буфере, и установить бит TxEnable в «1».

7.1.2 Операции приёма

Перед приёмом пакета, должна быть записана группа Rx дескрипторов, определяющих кольцевой буфер для операций приёма. Стартовые адреса начала всех сегментов должны быть 32-битные, сегменты должны быть равные по размеру и достаточные для обработки пакета максимальной длины. Кроме того, поле PacketSize дескриптора приёма должно быть заполнено, а бит 31 (Empty Flag) должен быть установлен в «1», для указания что кольцевой буфер приёма не содержит принятых пакетов.

Биты [7:4] Регистра Прерываний (DMAInterrupt) должны быть установлены, чтобы специфицировать типы генерируемых прерываний.

Вместе с этими полями в DMARxDescriptor должен быть записан адрес стартового Rx дескриптора и установлен в «1» RxEnable бит DMARxCtrl регистра для разрешения контроллеру ПДП обрабатывать принимаемый пакет.

Встроенный контроллер ПДП читает DMARxDescriptor для определения адреса первого дескриптора, затем читает этот дескриптор для того, что бы, во-первых, проверить доступность области памяти для сохранения пакета (Empty Flag в бите 31 компоненты PacketSize дескриптора должна быть «1»), во-вторых, определить стартовый адрес кольцевого буфера для хранения информации.

Если Empty Flag очищен, это означает, что предыдущий пакет, записанный на это место, ещё не был считан программой. В этом случае, контроллер ПДП прекращает последовательный приём пакетов, устанавливает бит RxOverflow в «1» в DMARxStatus регистре и очищает RxEnable бит в DMARxCtrl регистре. Если разрешено прерывание, то оно будет сгенерировано с источником RxOverflow. Любой последующий приём будет возможен только после обновления регистра DMARxDescriptor и новой стартовой позиции кольцевого буфера, а также установки в «1» RxEnable бита.

Контроллер ПДП начнёт приём пакета, если FIFO сообщит контроллеру ПДП о наличии принятого пакета в FIFO.

Если приём пакета закончен успешно, то контроллер прямого доступа в память сделает запись числа принятых байт, в битах [11:0] компонента PacketSize дескриптора приёма и очистит бит 31, помечая принятый пакет сохранённым в кольцевом буфере приёма. Флаг RxPktReceived в регистре DMARxStatus будет так же установлен. Прерывание RxPktReceived будет выставлено, если оно разрешено и число, записанное в поле RxPktCount (биты [23:16]) будет увеличено на «1». Если FIFO сообщит, что имеется принятый пакет, то контроллер ПДП начнёт передачу этого пакета в следующий сегмент кольцевого буфера. Адрес следующего дескриптора определён в компоненте NextDescriptor текущего дескриптора.

Программное обеспечение должно ответить на прерывание RxPktReceived считыванием пакета из этого места в кольцевом буфере приёма с последующей установкой Empty Flag соответствующего дескриптора в «1», помечая этот сегмент кольцевого буфера как доступный для сохранения следующего пакета.

Если произошла ошибка при приёме, то контроллер ПДП прекращает последовательную обработку принимаемых пакетов, устанавливает бит BusError регистра DMARxStatus и очищает бит RxEnable регистра DMARxCtrl. Выставляется прерывание по флагу RxBusError, если оно разрешено.

Любой следующий приём требует обновления регистра DMARxDescriptor, записи правильной стартовой позиции кольцевого буфера и установки бита DMARxDescriptor опять в «1».

7.2 Описание регистров периферийного устройства Ethernet 10/100

В таблице 7.1 приведена карта памяти периферийного устройства Ethernet 10/100.

Таблица 7.1 – Карта памяти периферийного устройства Ethernet 10/100

Адрес шестнадцатеричный	Обозначение	Функциональное назначение	Операции чтение/запись R/W
1	2	3	4
0x00700000	MAC1	Регистр 1 конфигурации MAC	R/W
0x00700001	MAC2	Регистр 2 конфигурации MAC	R/W
0x00700002	IPGT	Back-to-Back Inter-Packet-Gap регистр	R/W
0x00700003	IPGR	Non-Back-to-Back Inter-Packet-Gap регистр	R/W
0x00700004	CLRT	Регистр окна коллизий/повторов	R/W
0x00700005	MAXF	Регистр верхнего ограничения размера Frame	R/W
0x00700006	SUPP	Регистр поддержки PHY-интерфейса	R/W
0x00700007	TEST	Тестовый регистр	*R/W
0x00700008	MCFG	Регистр управления конфигурацией MII	R/W
0x00700009	MCMD	Регистр команд MII	R/W
0x0070000A	MADR	Регистр адреса MII	R/W
0x0070000B	MWTD	Регистр записываемых данных в MII	W
0x0070000C	MRDD	Регистр считываемых данных из MII	R
0x0070000D	MIND	Регистр индикации состояния MII	R
0x0070000E	SMII	SMII-статусный регистр	R
0x0070000F	MIIFIFC0	Регистр 0 конфигурации MIIFIF	*R/W
0x00700010	SA0	Регистр адреса станции SA0	R/W
0x00700011	SA1	Регистр адреса станции SA1	R/W
0x00700012	SA2	Регистр адреса станции SA2	R/W

Окончание таблицы 7.1

1	2	3	4
0x00700013	MIIFIFC1	Регистр 1 конфигурации MIIFIF	*R/W
0x00700014	MIIFIFC2	Регистр 2 конфигурации MIIFIF	*R/W
0x00700015	MIIFIFC3	Регистр 3 конфигурации MIIFIF	*R/W
0x00700016	MIIFIFC4	Регистр 4 конфигурации MIIFIF	*R/W
0x00700017	MIIFIFC5	Регистр 5 конфигурации MIIFIF	*R/W
0x00700018	MIIFIFRAM0	Регистр 0 обращения к FIFO	*R/W
0x00700019	MIIFIFRAM1	Регистр 1 обращения к FIFO	*R/W
0x0070001A	MIIFIFRAM2	Регистр 2 обращения к FIFO	*R/W
0x0070001B	MIIFIFRAM3	Регистр 3 обращения к FIFO	*R
0x0070001C	MIIFIFRAM4	Регистр 4 обращения к FIFO	*R/W
0x0070001D	MIIFIFRAM5	Регистр 5 обращения к FIFO	*R/W
0x0070001E	MIIFIFRAM6	Регистр 6 обращения к FIFO	*R/W
0x0070001F	MIIFIFRAM7	Регистр 7 обращения к FIFO	*R
0x00700020÷ 0x0070005F		Не используется	
0x00700060	DMATxCtrl	Регистр управления передачей	R/W
0x00700061	DMATxDescriptor	Регистр указателя дескриптора передачи	R/W
0x00700062	DMATxStatus	Регистр статуса передачи	R/WC
0x00700063	DMARxCtrl	Регистр управления приёмом	R/W
0x00700064	DMARxDescriptor	Регистр указателя дескриптора приёма	R/W
0x00700065	DMARxStatus	Регистр статуса приёма	R/WC
0x00700066	DMAIntrMask	Регистр маски прерывания	R/W
0x00700067	DMAInterrupt	Регистр прерываний	R
0x00710000÷ 0x00713FFF	RAM	Буферное ОЗУ Ethernet 16K × 16	R/W

_____

* Используется только для тестирования.

Регистр 1 конфигурации MAC

В таблице 7.2 приведена структура регистра 1 конфигурации MAC.

Таблица 7.2 – Структура регистра 1 конфигурации MAC

31–16	15	14	13–12	11	10	9	8
Резервные	Soft Reset	Sim Reset	Резервные	Reset RMCS	Reset RFUN	Reset TMCS	Reset TFUN
X	1	0	0	0	0	0	0
R	RW	RW		RW	RW	RW	RW
7	6	5	4	3	2	1	0
Резервные	Резервные	Резервные	LOOP BACK	Tx PAUSE	Rx PAUSE	Pass ALL	Rx Enable
0	0	0	0	0	0	0	0
			RW	RW	RW	RW	RW

В таблице 7.3 приведено описание и функциональное назначение битов регистра 1 конфигурации MAC.

Таблица 7.3 – Описание и функциональное назначение битов регистра 1 конфигурации MAC

Бит	Имя	Доступ	Описание
31–16	rsrv	R	Резерв
15	SOFT RESET	RW	Установка в «1» выполняет сброс блока MAC устройства Ethernet 10/100 кроме блока связи с процессором.
14	SIMULATION RESET	RW	Установка бита в «1» сбрасывает генератор случайных чисел устройства передачи.
11	RESET PEMCS/Rx	RW	Установка бита в «1» сбрасывает устройства MAC уровня, отвечающие за управление фильтрацией адресов при приёме пакетов (контроль домена).
10	RESET PERFUN	RW	Установка бита в «1» сбрасывает логику устройств приёма пакетов.
9	RESET PEMCS/Tx	RW	Установка бита сбрасывает устройства MAC уровня, отвечающие за управление адресами (контроль домена) при передаче информации.
8	RESET PETFUN	RW	Установка бита в «1» сбрасывает логику устройств передачи пакетов.
4	LOOPBACK	RW	Установка бита в «1» вызывает приём передаваемых пакетов обратно через MAC Receive интерфейс. Очистка бита не вызывает приём передаваемых пакетов обратно.
3	TX FLOW CONTROL	RW	Установка бита «1» позволяет передавать паузы, очистка блокирует передачу пауз в Frame.
2	RX FLOW CONTROL	RW	При установленном бите пауза принимается как часть Frame. При очищенном бите – пауза игнорируется.
1	PASS ALL RECEIVE FRAMES	RW	При установленном бите MAC выдаёт PASS для текущего принимаемого Frame независимо от их типа (для всех Frame). При очищенном бите PASS подтверждается для текущего принимаемого Frame при успешной передаче Frame.
0	RECEIVE ENABLE	RW	Установка бита позволяет принимать Frame. Внутренняя MAC синхронизация использует этот бит для приёма входящего потока и выхода SYNCHRONIZED RECEIVE ENABLE, используемого MAC для уточнения принимаемого фрейма.

Регистр 2 конфигурации MAC

Таблица 7.4 – Структура регистра 2 конфигурации MAC

31–16	15	14	13	12	11–10	9	8
Резервные	Резерв-ные	Excess Defer	B.P/ No Backoff	No Backoff	Резерв-ные	Long Pre	Pure Pre
X	0	0	0	0	0	0	0
R	R	RW	RW	RW	RW	RW	RW
7	6	5	4	3	2	1	0
Auto Pad	VLAN Pad	Pad Enable	CRC Enable	Delay CRC	Huge Frame	Length Check	Full Duplex
0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW

Таблица 7.5 – Описание и функциональное назначение битов регистра 2 конфигурации MAC

Бит	Имя	Доступ	Описание
31–16	rsrv	R	Резерв
14	EXCESS DEFER	RW	При установленном бите MAC будет неопределённо долго осуществлять обработку пакета (в соответствии со стандартом). При очищенном бите обработка пакета будет прервана в случае избыточной задержки, которая превышает установленный лимит
13	BACKPRESSURE / NO BACKOFF	RW	При установленном бите в случае коллизии повторная передача будет повторена немедленно, без периода ожидания, с целью повышения вероятности успешной передачи пакета
12	NO BACKOFF	RW	При установленном бите в случае коллизии период ожидания до повторной передачи определяется бинарно-экспоненциальным алгоритмом в соответствии со стандартом IEEE 802-3
9	LONG PREAMBLE ENFORCEMENT	RW	При установленном бите MAC позволяет принимать только пакеты, преамбула в которых менее 12 байт. При очищенном бите, MAC допускает приём пакетов с преамбулами любой длины, как в стандарте
8	PURE PREAMBLE ENFORCEMENT	RW	При установленном бите MAC верифицирует содержимое преамбулы для гарантирования, что она не содержит ошибок. При этом обработка пакета с ошибочной преамбулой прекращается. При очищенном бите проверка преамбулы не проводится
7	AUTO DETECT PAD ENABLE	RW	При установленном бите MAC автоматически определяет тип Frame, теговый или не теговый, сравнивая два октета адреса последующего источника с Ox8100 (VLAN протокол ID) или PAD, соответственно. Этот бит игнорируется, если бит PAD/CRC ENABLE очищен
6	VLAN PAD ENABLE	RW	При установленном бите MAC добавляет PAD ко всем коротким Frame, доводя их длину до 64 байт, и присоединяет значение CRC. Этот бит игнорируется, если бит PAD/CRC ENABLE очищен
5	PAD / CRC ENABLE	RW	При установленном бите MAC добавляет PAD во все короткие Frame. Если все Frame имеют правильную длину, то очистите этот бит. Этот бит используется совместно с битами AUTO PAD ENABLE или VLAN PAD ENABLE
4	CRC ENABLE	RW	Установленный бит обеспечивает автоматическое добавление CRC к Frame, независимо от того, требовалось это или нет. Если CRC уже включен во Frame, то бит необходимо очистить. В противном случае, если CRC необходимо добавлять в PAD, то бит нужно установить
3	DELAYED CRC	RW	Установка этого бита добавляет при передаче и вырезает при приёме четырёх байт CRC поля служебной информации, которые расположены в начале Frame. Если добавление отсутствует, то этот бит очистить. Если CRC добавляется к первым четырём байтам Frame служебной информации, то этот бит установить в «1». В противном случае установить в «0»
2	HUGE FRAME ENABLE	RW	Если бит установлен, то Frame произвольной длины принимаются и передаются
1	FRAME LENGTH CHECKING	RW	Если бит установлен, то длина и передаваемого, и принимаемого Frame компарируется с полем Length/Type Frame. При успешной компарации – проверка выполнена. О несовпадениях выдаётся сообщение в Transmit/Receive Statistics Vector
0	FULL-DUPLEX	RW	Если бит установлен, MAC операции выполняются в режиме Full-Duplex, в противном случае – в режиме Half-Duplex

PAD операции

Таблица 7.6 – PAD операции

Тип	ADPEN [7]	VLPEN [6]	PADEN [5]	Вид Frame
Все	x	x	0	Без PAD, проверка CRC
Все	0	0	1	PAD 60 бит, присоединено CRC
Все	x	1	1	PAD 64 бит, присоединено CRC
Все	1	0	1	Если безтеговый: PAD 60 бит, присоединено CRC Если VLAN – теги: PAD 64 бит, присоединено CRC

Регистр Back-to-Back Inter-Packet-Gap

Таблица 7.7 – Структура регистра Back-to-Back Inter-Packet-Gap

31–16	15–7	6÷0
Резервные		Inter-Packet-Gap Back-to-Back Transmits
X	0	0
R	RW	RW

Таблица 7.8 – Описание и функциональное назначение битов регистра Back-to-Back Inter-Packet-Gap

Бит	Имя	Доступ	Описание
31–16	rsrv	R	Резерв
15–7	rsrv	RW	Резерв
6–0	BACK-TO-BACK INTER-PACKET-GAP	RW	Это программируемое поле устанавливает минимальное время после передачи последнего полубайта пакета и началом передачи следующего пакета. В Full-Duplex режиме значение этого регистра рекомендуется равным периоду передачи полубайта минус 3. В Half-Duplex режиме значение этого регистра рекомендуется равным периоду передачи полубайта минус 6. В Full-Duplex режиме рекомендуется значение 15h, или 21 десятичное, который обеспечит минимальный IPG = 0,96 мкс (для 100 Мб/с) или 9,6 мкс (для 10 Мб/с). В Half-Duplex режиме рекомендуется значение 12h, или 18 десятичное, который, так же, обеспечит минимальный IPG = 0,96 мкс (для 100 Мб/с) или 9,6 мкс (для 10 Мб/с).

Non-Back-to-Back Inter-Packet-Gap регистр

Таблица 7.9 – Структура регистра Non-Back-to-Back Inter-Packet-Gap

31–16	15	14–8	7	6–0
Резервные		Inter-Packet-Gap - Part 1 Non-Back-to-Back Transmits	Резервные	Inter-Packet-Gap - Part 2 Non-Back-to-Back Transmits
X	0	0	0	0
R	RW	RW	RW	RW

Таблица 7.10 – Описание и функциональное назначение битов регистра Non-Back-to-Back Inter-Packet-Gap

Бит	Имя	Доступ	Описание
31–16	rsrv	R	Резерв
15	rsrv	RW	Резерв
14–8	NON-BACK-TO-BACK INTER-PACKET-GAP − PART 1	RW	Это программируемое поле представляет собой опцию carrierSense (определение несущей частоты), описанную в IEEE 802.3/4.2.3.2.1 «Carrier Deference». Если несущая частота идентифицирована во время синхронизации IPGR1, то MAC подстраивается под несущую частоту. Но, однако, если несущая частота определена после IPGR1, то MAC в течение IPGR2 продолжает передавать, принудительно вызывая возможность коллизии, таким образом, обеспечивая явный доступ к передающей среде. Его значение от 0h до IPGR2. Рекомендованное значение Ch (12 десятичное)
7	rsrv	RW	Резерв
6–0	NON-BACK-TO-BACK INTER-PACKET-GAP − PART 2	RW	Это программируемое поле представляет собой non-back-to-back Inter-Packet-Gap. Рекомендуемое значение 12h (18 десятичное), что обеспечивает минимальное IPG = 0,96 мкс (для 100 Мб/с) или 9,6 мкс (для 10 Мб/с)

Регистр окна коллизий/повторов (CLRT)

Таблица 7.11 – Структура регистра окна коллизий/повторов (CLRT)

31–16	15–14	13–8	7–4	3–0
Резервные		Collision Window	Резервные	Retransmission Maximum
X	0	37	0	F
R	RW	RW	RW	RW

Таблица 7.12 – Описание и функциональное назначение битов регистра окна коллизий/повторов (CLRT)

Бит	Имя	Доступ	Описание
31–16	rsrv	R	Резерв
15–14	rsrv	RW	Резерв
13–8	COLLISION WINDOW	RW	Это программируемое поле, представляющее время слота или окна коллизий, во время которого возможны коллизии в сконфигурированных сетях. Окно коллизий начинается с начала преамбулы, включая SFD. Значение окна коллизий по умолчанию соответствует числу байтов Frame в конце окна 37h (55 десятичное)
7–4	rsrv	RW	Резерв
3–0	Attempt Limit	RW	Это программируемое поле определяет число попыток повторной передачи после коллизии, прежде чем прервать передачу пакета из-за превышения числа коллизий. Стандарт специфицирует Attempt Limit числом Fh (15 десятичное)

Регистр максимальной длины Frame (MAXF)

Таблица 7.13 – Структура регистра максимальной длины Frame (MAXF)

31–16	15–0
Резервные	MaximumFrame Length
X	0600
R	RWC

Таблица 7.14 – Описание и функциональное назначение битов регистра максимальной длины Frame (MAXF)

Бит	Имя	Доступ	Описание
31–16	rsrv	R	Резерв
15–0	MAXIMUM FRAME LENGTH	RWC	В этот регистр после сброса записывается значение 0600h, что является максимально возможной длиной Frame, равной 1536 октет. Безтеговый максимальный размер Frame равен 1518 октет. К размеру тегового Frame добавляется четыре октета, и он составляет не более 1522 октет. Если необходимы более короткие Frame, то следует программировать этот регистр нужным значением. Примечание – Для VLAN тегового Frame, у которого в заголовке прибавляется 4 байт, следует к размеру Frame прибавить 4 байт

Регистр управления PHY-интерфейса (SUPP)

Таблица 7.15 – Структура регистра управления PHY-интерфейса (SUPP)

31–16	15	14–13	12	11	10–9	8
Резервные	Reset INT	Резервные	PHY mode	Reset RMII	Резервные	Speed
X	0	0	1	0	0	0
R	RW	R	RW	RW	R	RW
{ PE-SMII }{ PE-RMII }						

Продолжение таблицы 7.15

7	6	5	4	3	2	1	0
Reset 100X	Force Quiet	No Cipher	Link Fail	Reset 10T	Резервные	Enable Jabber	Bit Mode
0	0	0	0	0	0	0	0
RW	RW	RW	RW	R	R	RW	RW
{ PE-PMD }{ PE-ENDEC }							

Таблица 7.16 – Описание и функциональное назначение битов регистра управления PHY-интерфейса (SUPP)

Бит	Имя	Доступ	Описание
1	2	3	4
31–16	rsrv	R	Резерв
15	RESET INTERFACE MODULE	RW	Установка этого бита в «1» сбрасывает выбранный модуль физического интерфейса (устройство связи с физической линией). Очистка этого бита приводит к нормальным операциям с модулем физического интерфейса. Этот бит можно использовать, если подключен один модуль физического интерфейса, вместо битов 11, 7 и 3
12	PHY MODE	RW	Этот бит конфигурирует последовательный MII с подключенными SMII устройствами. Этот бит используют при подключении SMII PHY. Очисткой этого бита – устанавливают функции SMII MAC. Если выбран SMII MAC, то операции приёма/передачи пакетов выполняются на частоте 100 Мб/с в режиме Full Duplex
11	RESET PERMII	RW	Этот бит сбрасывает логику упрощённого (Reduced) MII

Окончание таблицы 7.16

1	2	3	4
10–9	rsrv	R	Резерв
8	SPEED	RW	Этот бит конфигурирует логику упрощённого (Reduced) MII для текущей рабочей скорости. При установке бита, выбран режим 100 Мб/с. Когда очищено, режим 10 Мб/с
7	RESET PE100X	RW	Этот бит сбрасывает модуль, который содержит логику шифратора/дешифратора символов 4 бит/5 бит (4B/5B)
6	FORCE QUIET	RW	Если бит установлен, то передаются на выход шифрованные (4B/5B) данные, при очищенном бите выполняется нормальная операция (без шифрации)
5	NO CIPHER	RW	Если бит установлен, то осуществляется передача 5B символов без шифрования, при очищенном – происходит передача с нормальным шифрованием
4	DISABLE LINK FAIL	RW	Если бит установлен, то 330 мс Link Fail таймер отключается для возможности короткого моделирования. Если очищен – выполняется нормальная операция
3	RESET PE10T	RW	Этот бит сбрасывает модуль, который преобразует потоки полубайтов MII в последовательный поток двоичных сигналов приёмопередатчика режима 10T
2	rsrv	R	Резерв
1	ENABLE JABBER PROTECTION	RW	Этот бит делает доступной защиту от неправильных данных при передаче 10T в ENDEC режиме. Условием защиты является передача одного значения в линию более 50 мс и служит для устранения этого условия для передач другими станциями
0	BIT MODE	RW	Если бит установлен, MAC работает в 10BASE-T ENDEC режиме, при котором синхронизация происходит по фактическим данным, а не на основе тактового генератора полубайтов

Тестовый регистр (TEST)

Таблица 7.17 – Структура тестового регистра (TEST)

31–16	15–3	2	1	0
Резервные	Резервные	Test Backpressure	Test Pause	Short Quanta
X	0	0	0	0
R	R	RW	RW	RW

Таблица 7.18 – Описание и функциональное назначение битов тестового регистра (TEST)

Бит	Имя	Доступ	Описание
2	TEST BACK-PRESSURE	RW	Установка этого бита заставит MAC подтвердить Backpressure на линии. Backpressure позволяет передать преамбулу при появлении несущей частоты. Переданный от системы пакет будет послан во время Backpressure.
1	TEST PAUSE	RW	Установка этого бита заставит MAC Control подуровень запретить передачи, как если бы был принят PAUSE Receive Control Frame с ненулевым параметром времени паузы. Примечание – Используется только в тестовых целях.
0	SHORTCUT PAUSE QUANTA	RW	Этот бит уменьшает эффективную PAUSE Quanta от 64-байтного периода до однобайтного периода. Примечание – Используется только в тестовых целях.

Регистр управления конфигурацией MII (MCFG)

Таблица 7.19 – Структура регистра управления конфигурацией MII (MCFG)

31–16	15	14–5	4–2	1
Резервные	Reset Mgmt	Резервные	Clock Select	No Pre
X	0	0	0	0
R	RW	R	RW	RW

Таблица 7.20 – Описание и функциональное назначение битов регистра управления конфигурацией MII (MCFG)

Бит	Имя	Доступ	Описание
15	RESET MII MGMT	RW	Этот бит сбрасывает MII-менеджмент модуль.
4–2	CLOCK SELECT	RW	Это поле используется логикой деления частоты при формировании MII Management Clock (MDC), который определён в стандарте IEEE 802.3u как более медленный, чем 2,5 МГц Примечание – Некоторые PHY поддерживают частоту до 12,5 МГц. Значение коэффициентов деления приведены в таблице 7.21 «Кодирование тактового генератора».
1	SUPPRESS PREAMBLE	RW	Установка этого бита вызывает модуль управления MII выполнять циклы чтения/записи без поля 32-битной преамбулы. При очистке этого бита поддерживаются нормальные циклы чтения/записи. Некоторые PHY поддерживают работу без преамбулы
0	SCAN INCREMENT	RW	Установка этого бита позволяет модулю MII выполнить чтение всех адресов PHY. Если бит установлен, то MII осуществляет чтение по всем PHY адресам, начиная с адреса 1 (поле возможных адресов [4:0]), Очистка бита приводит к непрерывному чтению одного PHY

Кодирование тактового генератора

Таблица 7.21 – Кодирование тактового генератора

CLOCK SELECT	4	3	2
Host Clock делённый на 4	0	0	X
Host Clock делённый на 6	0	1	0
Host Clock делённый на 8	0	1	1
Host Clock делённый на 10	1	0	0
Host Clock делённый на 14	1	0	1
Host Clock делённый на 20	1	1	0
Host Clock делённый на 28	1	1	1

Регистр команд MII (MCMD)

Таблица 7.22 – Структура регистра команд MII MCMD

31–16	15–2	1	0
Резервные	Резервные	Scan	Read
X	0	0	0
R	R	RW	RW

Таблица 7.23 – Описание и функциональное назначение битов регистра команд (MCMD)

Бит	Имя	Доступ	Описание
1	SCAN	RW	Установка этого бита заставляет модуль управления MII выполнять циклы чтения непрерывно. Это полезно, к примеру, для мониторинга сбоя Link Fail
0	READ	RW	Установка этого бита заставляет модуль управления MII выполнять одиночные циклы чтения. Прочитанные данные возвращаются в регистр MRDD по адресу Ch

Регистр адреса MII (MADR)

Таблица 7.24 – Структура регистра адреса (MII MADR)

31–16	15–13	12–8	7–5	4–0
Резервные	Резервные	PHY Address	Резервные	Register Address
X	0	0	0	0
R	RW	RW	RW	RW

Таблица 7.25 – Описание и функциональное назначение битов регистра адреса (MII MADR)

Бит	Имя	Доступ	Описание
12–8	PHY ADDRESS	RW	Это поле представляет собой 5-битный адрес PHY устройства для циклов управления от MII. Можно адресовать до 31 PHY. Адрес «0» – резервный
4–0	REGISTER ADDRESS	RW	Это поле представляет собой 5-битный адрес регистра PHY устройства для циклов управления от MII. Можно адресовать до 32 регистров

Регистр записи данных от MII (MWTD)

Таблица 7.26 – Структура регистра записи данных от MII (MWTD)

31–16	15–0
Резервные	Write Data
X	0000
R	W

Таблица 7.27 – Описание и функциональное назначение битов регистра записи данных от MII (MWTD)

Бит	Имя	Доступ	Описание
15–0	WRITE DATA	W	При цикле записи модуль управления MII использует данные этого регистра и предварительно сконфигурированные адрес PHY и регистра адреса MII для записи в PHY устройство.

Регистр чтения данных в MII (MRDD)

Таблица 7.28 – Структура регистра чтения данных в MII (MRDD)

31–16	15–0
Резервные	Read Data
X	0000
R	R

Таблица 7.29 – Описание и функциональное назначение битов регистра чтения данных в MII (MRDD)

Бит	Имя	Доступ	Описание
15–0	READ DATA	R	После цикла чтения модулем управления MII из PHY устройства прочитанные данные можно считать из этого регистра (по адресу 0070000Ch)

Регистр индикации состояния MII (MIND)

Таблица 7.30 – Структура регистра индикации состояния MII (MIND)

31–16	15–4	3	2	1	0
Резервные	Резервные	Link Fail	Not Valid	Scan	Busy
X	0	0	0	0	0
R	R	R	R	R	R

Таблица 7.31 – Описание и функциональное назначение битов регистра индикации состояния MII (MRDD)

Бит	Имя	Доступ	Описание
3	MII Link Fail	R	Возвращённая «1» указывает на сбой при управлении MII управления (управление PHY устройством)
2	NOT VALID	R	Возвращённая «1» указывает, что цикл чтения MII управления не завершён и данные для чтения ещё недоступны
1	SCANNING	R	Возвращённая «1» индицирует продолжающуюся скан операцию (продолжаются циклы чтения MII управления)
0	BUSY	R	Возвращённая «1» индицирует продолжающийся цикл чтения или записи MII управления

SMII-статусный регистр

Таблица 7.32 – Структура SMII-статусного регистра

31–16	15–5	4	3	2	1	0
Резервные	Резервные	CLASH	JABBER	Link	Duplex	Speed
X	0	0	0	0	0	0
R	R	R	R	R	R	R

Таблица 7.33 – Описание и функциональное назначение битов SMII-статусного регистра

Бит	Имя	Доступ	Описание
4	CLASH	R	Если возвращена «1», то она указывает на выбранный режим MAC-to-MAC, за исключением, если обнаружен PHY
3	JABBER	R	Возвращённая «1» указывает на возникновение Jabber условия
2	LINK	R	Возвращённая «1» индицирует LINK OK
1	DUPLEX	R	Возвращённая «1» индицирует операции в режиме Full-Duplex, если «0», то в режиме Half-Duplex
0	BUSY	R	Возвращённая «1» индицирует частоту 100Мбит/с, «0» – 10Мбит/с

Регистр адреса станции (SA0)

Таблица 7.34 – Структура регистра адреса станции (SA0)

31–16	15–8	7–0
Резервные	1st octet of Station Address	2nd octet of Station Address
X	0	0
R	RW	RW

Таблица 7.35 – Описание и функциональное назначение битов регистра адреса станции (SA0)

Бит	Имя	Доступ	Описание
15–8	STATION ADDRESS, 1st octet	R	Это поле содержит 1-й октет адреса станции, первый октет содержится в разрядах 15–8
7–0	STATION ADDRESS, 2nd octet	R	Это поле содержит 2-й октет адреса станции, второй октет содержится в разрядах 7–0

Регистр адрес станции (SA1)

Таблица 7.36 – Структура регистра адреса станции (SA1)

31–16	15–8	7–0
Резервные	3rd octet of Station Address	4th octet of Station Address
X	0	0
R	RW	RW

Таблица 7.37 – Описание и функциональное назначение битов регистра адреса станции (SA1)

Бит	Имя	Доступ	Описание
15–8	STATION ADDRESS, 3rd octet	R	Это поле содержит 3-й октет адреса станции, третий октет содержится в разрядах 15–8.
7–0	STATION ADDRESS, 4th octet	R	Это поле содержит 4-й октет адреса станции, четвёртый октет содержится в разрядах 7–0.

Регистр адрес станции (SA2)

Таблица 7.38 – Структура регистра адреса станции (SA2)

31–16	15–8	7–0
Резервные	5th octet of Station Address	6th octet of Station Address
X	0	0
R	RW	RW

Таблица 7.39 – Описание и функциональное назначение битов регистра адреса станции (SA2)

Бит	Имя	Доступ	Описание
15–8	STATION ADDRESS, 5th octet	R	Это поле содержит 5-й октет адреса станции, третий октет содержится в разрядах 15–8.
7–0	STATION ADDRESS, 6th octet	R	Это поле содержит 6-й октет адреса станции, четвёртый октет содержится в разрядах 7–0.

Регистр управления передачей (DMATxCtrl)

Таблица 7.40 – Структура регистра управления передачей (DMATxCtrl)

31–1	0
Резервные	Tx Enable
0	0
R	RW

Таблица 7.41 – Описание и функциональное назначение битов регистра управления передачей (DMATxCtrl)

Бит	Имя	Доступ	Описание
0	Tx Enable	R	Установка этого бита делает доступным устройству ПДП пакеты, предназначенные для передачи. Этот бит всякий раз очищается встроенным ПДП-контроллером, если происходит неполная передача или возникает состояние ошибки в передающей линии

Указатель дескриптора передачи (DMATxDescriptor)

Таблица 7.42 – Структура указателя дескриптора передачи (DMATxDescriptor)

31–2	1–0
Bits [31–2] of Descriptor Address	Игнорируется
0	0
RW	

Таблица 7.43 – Описание и функциональное назначение битов указателя дескриптора передачи (DMATxDescriptor)

Бит	Имя	Доступ	Описание
31–2	Bits [31–2] of Descriptor Address	RW	Если TxEnable установлен процессором, то встроенный ПДП контроллер читает этот регистр с целью получения адреса регистра памяти, в котором расположены данные о первом передаваемом пакете
1–0	Bits [1–0] of Descriptor Address	RW	Игнорируется с целью приведения в соответствие со стандартом IEEE 802-3

Регистр статуса передачи (DMATxStatus)

Таблица 7.44 – Структура регистра статуса передачи (DMATxStatus)

31–24	23–16	15–4	3	2	1	0
Резервные	TxPktCount	Резервные	Bus Error	0	TxUnderrun	TxPktSent
0	0	0	0	0	0	0
	RWC		RWC		RWC	RW

Таблица 7.45 – Описание и функциональное назначение битов регистра статуса передачи (DMATxStatus)

Бит	Имя	Доступ	Описание
23–16	TxPktCount	RWC	8-битный счётчик пакетов передачи, который увеличивается на единицу каждый раз, когда встроенный ПДП контроллер успешно завершает передачу пакета, и декремент, если головной процессор запишет «1» в бит 0 этого регистра
3	Bus Error	RWC	Установленный бит указывает на то, что произошла ошибка на шине либо в режиме обмена данными процессором с регистрами устройства, либо в режиме прямого доступа к памяти
1	TxUnderrun	RWC	Устанавливается всякий раз, когда ПДП контроллер читает и устанавливает в «1» Empty Flag в дескрипторе. Это означает, что идёт обработка данных
0	TxPktSent	RW	Установленная «1» индицирует, что один или более пакетов были успешно переданы. Запись «1» в этот бит приводит к уменьшению значения TxPktCount на единицу. Бит очищается всякий раз, когда TxPktCount становится нулём

Регистр управления приёмом (DMARxCtrl)

Таблица 7.46 – Структура регистра управления приёмом (DMARxCtrl)

31–1	0
Резервные	Rx Enable
0	0
R	RW

Таблица 7.47 – Описание и функциональное назначение битов регистра управления приёмом (DMARxCtrl)

Бит	Имя	Доступ	Описание
0	Rx Enable	R	Установка этого бита делает доступным устройству ПДП пакеты, предназначенные для приёма. При установленном бите встроенный ПДП контроллер готов начать принимать новый пакет, как только FIFO покажет, что новый пакет доступен (FRSOF подтверждён). Этот бит всякий раз очищается, когда встречается Rx Overflow или Bus Error состояние

Регистр указателя дескриптора приёма (DMARxDescriptor)

Таблица 7.48 – Структура регистра указателя дескриптора приёма (DMARxDescriptor)

31–2	1–0
Bits [31–2] of Descriptor Address	Игнорируется
0	0
RW	

Таблица 7.49 – Описание и функциональное назначение битов регистра указателя дескриптора приёма (DMARxDescriptor)

Бит	Имя	Доступ	Описание
31–2	Bits [31–2] of Descriptor Address	RW	Если RxEnable установлен процессором, то встроенный ПДП контроллер читает этот регистр с целью получения адреса регистра памяти, в которую будут записываться данные о первом принятом пакете
1–0	Bits [1–0] of Descriptor Address	R	Игнорируется с целью приведения в соответствие со стандартом IEEE 802-3

Регистр статуса приёма (DMARxStatus)

Таблица 7.50 – Структура регистра статуса приёма (DMARxStatus)

31–24	23–16	15–4	3	2	1	0
Резервные	RxPktCount	Резервные	Bus Error	RxOverflow	0	RxPktReceived
0	0	0	0	0	0	0
	RWC		RWC	RWC		RW

Таблица 7.51 – Описание и функциональное назначение битов регистра статуса приёма (DMARxStatus)

Бит	Имя	Доступ	Описание
23–16	TxPktCount	RWC	8-битный счётчик принятых пакетов, который увеличивается на 1 каждый раз, когда встроенный ПДП контроллер успешно завершает транзакцию пакета, и уменьшается на 1 каждый раз, когда процессор записывает «1» в бит 0 этого регистра
15–4	rsrv	R	Зарезервировано
3	Bus Error	RWC	Установленный бит указывает на то, что произошла ошибка на шине либо в режиме обмена данными процессором с регистрами устройства, либо в режиме прямого доступа к памяти
1	RxOverflow	RWC	Устанавливается всякий раз, когда ПДП контроллер читает «0» из бита Empty Flag в дескрипторе приёма во время обработки данных
0	RxPktSent	RW	Установленная «1» индицирует, что один или более пакетов были успешно приняты. Запись «1» в этот бит уменьшает значение RxPktCount на «1». Бит очищается всякий раз, когда RxPktCount становится нулём

Регистр маски прерывания (DMAIntrMask)

Таблица 7.52 – Структура регистра маски прерывания (DMAIntrMask)

31–8	7	6	5	4	3	2	1	0
Резервные	Bus Error (Rx)	Rx Overflow	0	RxPkt Received	Bus Error (Tx)	0	Tx Underrun	TxPkt Sent
0	0	0	0	0	0	0	0	0
R	RW	RW	R	RW	RW	R	RW	RW

Таблица 7.53 – Описание и функциональное назначение битов регистра маски прерывания (DMAIntrMask)

Бит	Имя	Доступ	Описание
7	Bus Error Mask	RW	Установка этого бита разрешает прерывание от источника прерывания BusError бит регистра «Статус приёма»
6	Rx Overflow Mask	RW	Установка этого бита разрешает прерывание от источника прерывания RxOverflow бит регистра «Статус приёма»
4	RxPktReceived Mask	RW	Установка этого бита разрешает прерывание от источника прерывания RxPktReceived бит регистра «Статус приёма»
3	Bus Error Mask	RW	Установка этого бита разрешает прерывание от источника прерывания BusError бит регистра «Статус передачи»
1	Tx Underrun Mask	RW	Установка этого бита разрешает прерывание от источника прерывания TxUnderrun бит регистра «Статус передачи»
0	TxPktSent Mask	RW	Установка этого бита разрешает прерывание от источника прерывания TxPktSent бит регистра «Статус передачи»

Регистр прерываний (DMAInterrupt)

Таблица 7.54 – Структура регистра прерываний (DMAInterrupt)

31–8	7	6	5	4	3	2	1	0
Резервные	Bus Error (Rx)	Rx Overflow	0	RxPkt Received	Bus Error(Tx)	0	Tx Underrun	TxPktSent
0	0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R

Таблица 7.55 – Описание и функциональное назначение битов регистра прерываний (DMAInterrupt)

Бит	Имя	Доступ	Описание
7	Receive Bus Error	R	При установленном бите записывается прерывание Receive Bus Error, в том случае если Bus Error бит регистра «Статус приёма» и бит 7 регистра «Маска прерывания» оба установлены в «1»
6	Rx Overflow	R	При установленном бите записывается прерывание RxOverflow, в том случае если RxOverflow бит регистра «Статус приёма» и бит 6 регистра «Маска прерывания» оба установлены в «1»
4	RxPktReceived	R	При установленном бите записывается прерывание RxPktReceived, в том случае если RxPktReceived бит регистра «Статус приёма» и бит 4 регистра «Маска прерывания» оба установлены в «1»
3	Transmit Bus Error	R	При установленном бите записывается прерывание Transmit BusError, в том случае если BusError бит регистра «Статус передачи» и бит 3 регистра «Маска прерывания» оба установлены в «1»
1	Tx Underrun	R	При установленном бите записывается прерывание TxUnderrun, в том случае если TxUnderrun бит регистра «Статус передачи» и бит 1 регистра «Маска прерывания» оба установлены в «1»
0	TxPktSent	R	При установленном бите записывается прерывание TxPktSent, в том случае если TxPktSent бит регистра «Статус передачи» и бит 0 регистра «Маска прерывания» оба установлены в «1»

Компоненты дескриптора

Таблица 7.56 – Компоненты дескриптора

Адрес шестнадцатеричный	Обозначение	Функциональное назначение	Размер бит
+0	PacketStartAddr	Стартовый адрес для пакета данных	32
+4	PacketSize	Размер пакета, Overrides и Empty Flag	32
+8	NextDescriptor	Адрес следующего дескриптора	32

Стартовый адрес для пакета данных

Таблица 7.57 – Структура стартового адреса для пакета данных

31–2	1	0
Bits [31:2] of Packet Start Address	0	0
–	0	0
RW	R	R

Таблица 7.58 – Описание и функциональное назначение битов стартового адреса для пакета данных

Бит	Имя	Доступ	Описание
31–2	Top 30 bits of Packet Start Address	RW	Встроенный ПДП контроллер читает этот регистр для определения положения первого байта данных в памяти.
1–0		RW	Игнорируется, поскольку память 32-разрядная, и по одному адресу находятся сразу четыре байта данных.
Примечание – Область отведённой памяти должна быть достаточной для пакета максимальной длины.			

Размер пакета, Overrides и Empty Flag

Таблица 7.59 – Структура Overrides и Empty Flag

31	30–21	20–16	15–12	11–0
Empty Flag	Резервные	FTPP Overrides	Резервные	Packet Size
0	0	0	0	0
RW	RW	RW	RW	RW

Таблица 7.60 – Описание и функциональное назначение битов Overrides и Empty Flag

Бит	Имя	Доступ	Описание
1	2	3	4
31	Empty Flag	RW	Для операций передачи этот бит указывает на доступность данных передачи, связанных с пакетом. Для операций приёма, этот бит показывает наличие места для сохранения принимаемого пакета. Установка этого флага используется для проверки правильности дескриптора. Замечание. При завершении операции передачи, ПДП контроллер пишет «1» в этот бит, что говорит, что эти данные использованы для передачи. При успешном завершении операции приёма, ПДП контроллер пишет «0» в этот бит, что указывает на то, что отведённое место использовано для сохранения пакета. Первое действие гарантирует, что данные не будут переданы дважды, второе – что сохранённые данные не будут испорчены (стёрты) записью поверх них следующего пакета

Окончание таблицы 7.60

1	2	3	4
20–16	FTPP Overrides	RW	5-битное поле содержит флаги управления FIFO в течение обмена пакетами. Эти биты декодируются следующим образом: Бит 20 (FTCFRM) – FIFO передача Control Frame Биты 19–18 (FTPPPADMODE) – FIFO передача пакета в режиме PAD Mode. Бит 17 (FTPPGENFCS) – FIFO передача пакета с генерацией FCS. Бит 16 (FTPPEN) – Разрешение FIFO на передачу пакета
11–0	Packet Size	RW	12-битное поле для операций передачи даёт размер пакета передачи в байтах. При приёме это поле заполняет контроллер ПДП. Значение этого поля до приёма будет проигнорировано

Адрес следующего дескриптора

Таблица 7.61 – Структура адреса следующего дескриптора

31–2	1	0
Bits [31–2] of Descriptor Address	0	0
–	0	0
RW	R	R

Таблица 7.62 – Описание и функциональное назначение битов в адресе следующего дескриптора

Бит	Имя	Доступ	Описание
31–2	Top 30 bits of Descriptor Address	RW	Встроенный ПДП контроллер читает этот регистр для определения положения в памяти следующего дескриптора для следующего пакета в последовательности. Дескрипторы должны сформировать связанный список до закрытия Link.
1–0		RW	Игнорируется, поскольку память 32-разрядная, и по одному адресу находится сразу четыре байта данных.

Примечание – Адресация оперативной памяти устройства Ethernet 10/100 на LocalBus и контроллере ПДП различается и связана соотношением Адрес(LocalBus) = 00710000h + ((Адрес (ПДП)) / 4).

Таким образом, Адресу (на LocalBus) = 00710000h соответствует Адрес (ПДП) равный 0000h, а Адресу (на LocalBus) = 00710001h соответствует Адрес (ПДП) равный 0004h.

8 Описание периферийного устройства USB 2.0

Периферийное устройство USB 2.0 (далее USB 2.0) поддерживает стандарт Universal Serial Bus 2.0 High Speed и обеспечивает подключение между ядром ПЦОС 1 или ПЦОС 2 и периферийными устройствами на протокольном уровне.

USB 2.0 поддерживает высокоскоростной – 480 Мбит/с High Speed и полноскоростной – 12 Мбит/с (Full Speed) режимы пакетного, по прерыванию или изохорного обмена данными. Для подключения к физическому интерфейсу PHY периферийное устройство USB 2.0 управляет интерфейсом UTMI для восьмибитного режима обмена данными. Тактовая частота UTMI интерфейса должна быть равной 60 МГц для обеспечения 480 Мбит/с (60 Мбайт/с) скорости передачи данных.

Периферийное устройство USB 2.0 поддерживает 16 конечных точек (End Point), которые могут программироваться от процессора через регистры конечных точек EPn_CSR, EPn_INT, EPn_OBA, EPn_IBA. Процессор должен однозначно нумеровать конечные точки для обеспечения ссылки к внешнему устройству (Device), подключенному к конечной точке.

Каждая из точек может быть определена как IN, OUT или CONTROL типа в режимах пакетного, по прерыванию или изохорного обмена данными. Конечная точка типа CONTROL использует два буфера EPn_IBA и EPn_OBA; конечные точки IN и OUT используют один тип буфера соответственно.

Буферизация осуществляется через назначение указателей буферов и их размеров в регистрах EPn_IBA и EPn_OBA. Буфера размещаются в диапазоне адресов 0x00610000 – 0x00613FFF. Таким образом, максимальный размер буфера составляет 16K 32-разрядных слов.

В периферийном устройстве USB 2.0 предусмотрена двухуровневая система прерываний. Вырабатываемые прерывания от USB 2.0 коммутируются на IIOF0_1# для ПЦОС 1 или IIOF0_2# для ПЦОС 2. Основной регистр источника прерывания (INT_SRC) – это первый уровень, обеспечивает регистрацию всех прерываний возможных от USB 2.0. Если требуется уточнение источника прерывания, вырабатываемого конечной точкой, то необходимо выполнить чтение регистра источника прерываний для конкретной точки (EPn_INT) – это второй уровень. Для маскирования прерываний предусмотрены соответствующие биты регистров маскирования (INT_MSK и EPn_INT).

Передача данных (идентификатор IN) обеспечивается следующим алгоритмом:

1. Установить биты регистра управления/состояния в требуемые значения, например, CSR = 00008080h (высокоскоростной режим, USB разрешён, интервал дескриптизации отсутствует).

2. Определить адрес функции через установление значения в регистре адреса функции. Значение адреса функции устанавливает USB-хост в фазе SETUP.

3. Установить биты регистра управления конечной точки в требуемые значения, например, EPn_CSR = 06040100h (IN тип, пакетная передача данных, точка номер 1, 256 байт максимальный размер пакета).

4. Установить буфер конечной точки, указав размер буфера и его начальный адрес (указатель) в регистре буфера конечной точки, например, EPn_IBA = 40000000h (размер буфера выбран максимальным – 16 383 байт, буфер начинается с нулевого адреса).

5. Установить биты регистров маскирования прерываний (INT_MSK, EPn_INT) в требуемые значения, например, INT_MSK = 0000007FFh, EPn_MSK = 7FF00000h (разрешить все прерывания первого уровня типа A, разрешить все прерывания второго уровня, вырабатываемые конечной точкой).

6. Загрузить данные для передачи в буфер, указанный в пункте 4.

7. Ожидать пакета IN от USB-хоста, при необходимости анализировать состояние USB 2.0 чтением статусных регистров (CSR, EPn_CSR) или обрабатывать прерывание IIOF0_n процессора.

Приём данных (идентификатор OUT) обеспечивается следующим алгоритмом:

1. Установить биты регистра управления/состояния в требуемые значения, например, CSR = 00008080h (высокоскоростной режим, USB 2.0 разрешён, интервал дескриптизации отсутствует).

2. Определить адрес функции через установление значения в регистре адреса функции. Значение адреса функции раздаёт USB-хост в фазе SETUP.

3. Установить биты регистра управления конечной точки в требуемые значения, например, EPn_CSR = 02040100h (OUT-тип, пакетная передача данных, точка номер 1, 256 байт максимальный размер пакета).

4. Установить буфер конечной точки, указав размер буфера и его начальный адрес (указатель) в регистре буфера конечной точки, например, EPn_IBA = 40000000h (размер буфера выбран максимальным – 16 383 байта, буфер начинается с нулевого адреса).

5. Установить биты регистров маскирования прерываний (INT_MSK, EPn_INT) в требуемые значения, например, INT_MSK = 0000007FFh, EPn_MSK = 7FF00000h (разрешить все прерывания типа A первого уровня, разрешить все прерывания второго уровня, вырабатываемые конечной точкой).

6. Ожидать пакета OUT/DATA от USB-хоста, при необходимости анализировать состояние USB 2.0 чтением статусных регистров (CSR, EPn_CSR) или обрабатывать прерывание IIOF0_x# процессора.

7. Выгрузить данные из буфера приёма, указанного в перечислении 5.

8.1 Описание регистров периферийного устройства USB 2.0

В данном разделе представлены описания регистров и буфера конечной точки для управления и обработки данных периферийного устройства USB 2.0.

В таблице 8.1 описываются регистры USB 2.0.

Таблица 8.1 – Регистры USB 2.0

Адрес	Имя	Доступ	Описание
1	2	3	4
0x00600000	CSR	RW	Регистр управления/состояния
0x00600004	FA	RW	Регистр адреса функции
0x00600008	INT_MSK	RW	Регистр маскирования прерываний
0x0060000C	INT_SRC	ROC	Регистр источника прерываний
0x00600010	FRM_NAT	RO	Регистр подсчёта кадров и времени
0x00600040	EP0_CSR	RW	Конечная точка 0: Регистр управления/состояния
0x00600044	EP0_INT	RW	Конечная точка 0: Регистр прерываний
0x00600048	EP0_OBA	RW	Конечная точка 0: Регистр указателя буфера вывода
0x0060004C	EP0_IBA	RW	Конечная точка 0: Регистр указателя буфера ввода
0x00600050	EP1_CSR	RW	Конечная точка 1: Регистр управления/состояния
0x00600054	EP1_INT	RW	Конечная точка 1: Регистр прерываний
0x00600058	EP1_OBA	RW	Конечная точка 1: Регистр указателя буфера вывода
0x0060005C	EP1_IBA	RW	Конечная точка 1: Регистр указателя буфера ввода
0x00600050	EP2_CSR	RW	Конечная точка 2: Регистр управления/состояния
0x00600054	EP2_INT	RW	Конечная точка 2: Регистр прерываний
0x00600068	EP2_OBA	RW	Конечная точка 2: Регистр указателя буфера вывода
0x0060006C	EP2_IBA	RW	Конечная точка 2: Регистр указателя буфера ввода
0x00600070	EP3_CSR	RW	Конечная точка 3: Регистр управления/состояния
0x00600074	EP3_INT	RW	Конечная точка 3: Регистр прерываний
0x00600078	EP3_OBA	RW	Конечная точка 3: Регистр указателя буфера вывода
0x0060007C	EP3_IBA	RW	Конечная точка 3: Регистр указателя буфера ввода
0x00600080	EP4_CSR	RW	Конечная точка 4: Регистр управления/состояния
0x00600084	EP4_INT	RW	Конечная точка 4: Регистр прерываний

Окончание таблицы 8.1

1	2	3	4
0x00600088	EP4_OBA	RW	Конечная точка 4: Регистр указателя буфера вывода
0x0060008C	EP4_IBA	RW	Конечная точка 4: Регистр указателя буфера ввода
0x00600090	EP5_CSR	RW	Конечная точка 5: Регистр управления/состояния
0x00600094	EP5_INT	RW	Конечная точка 5: Регистр прерываний
0x00600098	EP5_OBA	RW	Конечная точка 5: Регистр указателя буфера вывода
0x0060009C	EP5_IBA	RW	Конечная точка 5: Регистр указателя буфера ввода
0x006000A0	EP6_CSR	RW	Конечная точка 6: Регистр управления/состояния
0x006000A4	EP6_INT	RW	Конечная точка 6: Регистр прерываний
0x006000A8	EP6_OBA	RW	Конечная точка 6: Регистр указателя буфера вывода
0x006000AC	EP6_IBA	RW	Конечная точка 6: Регистр указателя буфера ввода
0x006000B0	EP7_CSR	RW	Конечная точка 7: Регистр управления/состояния
0x006000B4	EP7_INT	RW	Конечная точка 7: Регистр прерываний
0x006000B8	EP7_OBA	RW	Конечная точка 7: Регистр указателя буфера вывода
0x006000BC	EP7_IBA	RW	Конечная точка 7: Регистр указателя буфера ввода
0x006000C0	EP8_CSR	RW	Конечная точка 8: Регистр управления/состояния
0x006000C4	EP8_INT	RW	Конечная точка 8: Регистр прерываний
0x006000C8	EP8_OBA	RW	Конечная точка 8: Регистр указателя буфера вывода
0x006000CC	EP8_IBA	RW	Конечная точка 8: Регистр указателя буфера ввода
0x006000D0	EP9_CSR	RW	Конечная точка 9: Регистр управления/состояния
0x006000D4	EP9_INT	RW	Конечная точка 9: Регистр прерываний
0x006000D8	EP9_OBA	RW	Конечная точка 9: Регистр указателя буфера вывода
0x006000DC	EP9_IBA	RW	Конечная точка 9: Регистр указателя буфера ввода
0x006000D0	EP10_CSR	RW	Конечная точка 10: Регистр управления/состояния
0x006000D4	EP10_INT	RW	Конечная точка 10: Регистр прерываний
0x006000D8	EP10_OBA	RW	Конечная точка 10: Регистр указателя буфера вывода
0x006000EC	EP10_IBA	RW	Конечная точка 10: Регистр указателя буфера ввода
0x006000F0	EP11_CSR	RW	Конечная точка 11: Регистр управления/состояния
0x006000F4	EP11_INT	RW	Конечная точка 11: Регистр прерываний
0x006000F8	EP11_OBA	RW	Конечная точка 11: Регистр указателя буфера вывода
0x006000FC	EP11_IBA	RW	Конечная точка 11: Регистр указателя буфера ввода
0x00600100	EP12_CSR	RW	Конечная точка 12: Регистр управления/состояния
0x00600104	EP12_INT	RW	Конечная точка 12: Регистр прерываний
0x00600108	EP12_OBA	RW	Конечная точка 12: Регистр указателя буфера вывода
0x0060010C	EP12_IBA	RW	Конечная точка 12: Регистр указателя буфера ввода
0x00600110	EP13_CSR	RW	Конечная точка 13: Регистр управления/состояния
0x00600114	EP13_INT	RW	Конечная точка 13: Регистр прерываний
0x00600118	EP13_OBA	RW	Конечная точка 13: Регистр указателя буфера вывода
0x0060011C	EP13_IBA	RW	Конечная точка 13: Регистр указателя буфера ввода
0x00600120	EP14_CSR	RW	Конечная точка 14: Регистр управления/состояния
0x00600124	EP14_INT	RW	Конечная точка 14: Регистр прерываний
0x00600128	EP14_OBA	RW	Конечная точка 14: Регистр указателя буфера вывода
0x0060012C	EP14_IBA	RW	Конечная точка 14: Регистр указателя буфера ввода
0x00600130	EP15_CSR	RW	Конечная точка 15: Регистр управления/состояния
0x00600134	EP15_INT	RW	Конечная точка 15: Регистр прерываний
0x00600138	EP15_OBA	RW	Конечная точка 15: Регистр указателя буфера вывода
0x0060013C	EP15_IBA	RW	Конечная точка 15: Регистр указателя буфера ввода
Примечание – R – регистр может читаться, W – в регистр можно записывать			

В таблицах 8.2 и 8.3 приведены структура и описание битов регистра управления/состояния CSR. Значения битов после сброса 0000h.

Таблица 8.2 – Структура регистра управления/состояния CSR

31–16	15	14	13	12	11-10	9-8
RSRV	SM	PHYSUSP	RSRV	RESUME	TDD	RSRV
R	RW	W	R	W	RW	R
0	0	0	0	0	0	0
7	6	5	4-3	2	1	0
CE	IDLE	SE0	RSRV	ATTACH	FSHS	ESUSP
RW	R	R	R	R	R	R
0	0	0	0	0	0	0

Таблица 8.3 – Описание битов регистра управления/состояния

Бит	Имя	Доступ	Описание
31–16	RSRV	R	Зарезервированы
15	SM	RW	Установка этого бита в «1» запрещает высокоскоростной режим
14	PHYSUSP	W	Запись в этот бит «1» означает, что физический интерфейс будет переведён в SUSPEND. Физический интерфейс автоматически выходит из SUSPEND если физическая линия не находится в IDLE, или бит 0 регистра ожидания устанавливается в «1»
13	RSRV	R	Зарезервирован
12	RESUME	W	Запись в этот бит «1» выводит физический интерфейс из режима SUSPEND
11–10	TDD	RW	Интервал выполнения дескриптора передаваемых данных (интервал дескриптизации) 00 – отсутствует 01 – каждые 0,25 мс 10 – каждые 0,5 мс 11 – каждую 1 мс
9–8	RSRV	R	Зарезервированы
7	CE	RW	0 – ядро USB запрещено 1 – ядро USB разрешено
6	IDLE	R	Бит устанавливается в «1» когда на линии более 3 мс удерживается полноскоростной IDLE режим
5	SE0	R	Бит устанавливается в «1» когда на линии более 2 мс удерживается SE0 режим
4–3	RSRV	R	Зарезервированы
2	ATTACH	R	0 – физическая линия не подключена 1 – физическая линия подключена
1	FSHS	R	0 – полноскоростной режим UTMI интерфейса 1 – высокоскоростной режим UTMI интерфейса
0	ESUSP	R	0 – нормальный режим USB 1 – USB устанавливается в SUSPEND со стороны физического интерфейса
Примечание – R – регистр может читаться, W – в регистр можно записывать.			

Биты регистра адреса функции (FA) устанавливается процессором, когда конфигурируются функции контроллера. Значение адреса функции получают от USB-хоста в фазе SETUP. В таблицах 8.4 и 8.5 приведены структура и описание битов регистра адреса функции.

Значения битов после сброса 0000h.

Таблица 8.4 – Структура адреса функции

31–16	7–0
RSRV	FA
R	R
0	0

Таблица 8.5 – Описание битов регистра адреса функции

Бит	Доступ	Описание
31–8	R	Резерв
7–0	R	Адрес функции
Примечание – R – регистр может читаться, W – в регистр можно записывать		

В таблицах 8.6 и 8.7 приведены структура и описание битов регистра маскирования прерываний INT_MSK.

«0» – прерывание маскируется, «1» – прерывание разрешено.

Значения битов после сброса 0000h.

Таблица 8.6 – Структура регистра маскирования прерываний

31–27	26	25	24	23	22	21
RSRV	B_PHYSE 0	B_IDLE	B_RESET	B_RXERR	B_DEATTACH	B_ATTACH
R	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0
22	23	24	25	26	15	14
B_RESUME	B_SUSP	B_EPERR	B_PIDERR	B_CRC5ERR	RSRV	RSRV
RW	RW	RW	RW	RW	R	R
0	0	0	0	0	0	0
13	12	11	10	9	8	7
RSRV	RSRV	RSRV	A_PHYSE0	A_IDLE	A_RESET	A_RXERR
R	R	R	RW	RW	RW	RW
0	0	0	0	0	0	0
6	5	4	3	2	1	0
A_DEATTACH	A_ATTACH	A_RESUME	A_SUSP	A_EPERR	A_PIDERR	A_CRC5ERR
RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0

Таблица 8.7 – Описание битов регистра маскирования прерываний

Бит	Имя	Доступ	Описание
31–27	RSRV	R	Зарезервировано
26	B_PHYSE0	RW	Маска прерывания B: физическая линия в SE0
25	B_IDLE	RW	Маска прерывания B: физическая линия в IDLE
24	B_RESET	RW	Маска прерывания B: физическая линия формирует RESET
23	B_RXERR	RW	Маска прерывания B: ошибка приёмника (Rx)
22	B_DEATTACH	RW	Маска прерывания B: физическая линия отключена
21	B_ATTACH	RW	Маска прерывания B: физическая линия подключена
20	B_RESUME	RW	Маска прерывания B: физическая линия установлена в SUSPEND
19	B_SUSP	RW	Маска прерывания B: физическая линия входит в SUSPEND
18	B_EPERR	RW	Маска прерывания B: не существует конечной точки
17	B_PIDERR	RW	Маска прерывания B: ошибка контрольной суммы идентификатора пакета
16	B_CRC5ERR	RW	Маска прерывания B: ошибка контрольной суммы типа пакета
15–11	RSRV	R	Зарезервировано
10	A_PHYSE0	RW	Маска прерывания A: физическая линия в SE0
9	A_IDLE	RW	Маска прерывания A: физическая линия в IDLE
8	A_RESET	RW	Маска прерывания A: физическая линия формирует RESET
7	A_RXERR	RW	Маска прерывания A: ошибка приёмника (Rx)
6	A_DEATTACH	RW	Маска прерывания A: физическая линия отключена
5	A_ATTACH	RW	Маска прерывания A: физическая линия подключена
4	A_RESUME	RW	Маска прерывания A: физическая линия установлена в SUSPEND
3	A_SUSP	RW	Маска прерывания A: физическая линия входит в SUSPEND
2	A_EPERR	RW	Маска прерывания A: не существует конечной точки
1	A_PIDERR	RW	Маска прерывания A: ошибка контрольной суммы идентификатора пакета
0	A_CRC5ERR	RW	Маска прерывания A: ошибка контрольной суммы типа пакета
Примечание – R – регистр может читаться, W – в регистр можно записывать.			

В таблицах 8.8 и 8.9 приведены структура и описание битов регистра источника прерываний INT_SRC.

«1» – Прерывание возникло. Биты регистра очищаются при чтении из него значения.

Значения битов после сброса 0000h.

Таблица 8.8 – Структура регистра источника прерываний

31	30	29	28	27	26	25	24
RSRV	SE0	IDLE	RESET	RXERR	ATTACH	DEATTACH	RESUME
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16
SUSP	EPERR	PIDERR	CRC5ERR	RSRV	RSRV	RSRV	RSRV
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8
EP15_INT	EP14_INT	EP13_INT	EP12_INT	EP11_INT	EP10_INT	EP9_INT	EP8_INT
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
EP7_INT	EP6_INT	EP5_INT	EP4_INT	EP3_INT	EP2_INT	EP1_INT	EP0_INT
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0

Таблица 8.9 – Описание битов регистра источника прерываний

Бит	Имя	Доступ	Описание
31	RSRV	R	Зарезервировано
30	SE0	R	Физическая линия в SE0 (физическая линия находится в состоянии SE0 более 2 мс)
29	IDLE	R	Физическая линия в IDLE (физическая линия находится в состоянии полноскоростном IDLE более 3 мс)
28	RESET	R	Физическая линия формирует RESET
27	RXERR	R	Ошибка приёмника (Rx)
26	ATTACH	R	Физическая линия отключена
25	DEATTACH	R	Физическая линия подключена
24	RESUME	R	Физическая линия запрашивает вывод USB из SUSPEND режима
23	SUSP	R	Физическая линия установлена в SUSPEND
22	EPERR	R	Не существует конечной точки
21	PIDERR	R	Ошибка контрольной суммы идентификатора пакета данных
20	CRC5ERR	R	Ошибка контрольной суммы типа пакета
19–16	RSRV	R	Зарезервировано
15	EP15_INT	R	Прерывания от конечной точки номер 15
14	EP14_INT	R	Прерывания от конечной точки номер 14
13	EP13_INT	R	Прерывания от конечной точки номер 13
12	EP12_INT	R	Прерывания от конечной точки номер 12
11	EP11_INT	R	Прерывания от конечной точки номер 11
10	EP10_INT	R	Прерывания от конечной точки номер 10
9	EP9_INT	R	Прерывания от конечной точки номер 9
8	EP8_INT	R	Прерывания от конечной точки номер 8
7	EP7_INT	R	Прерывания от конечной точки номер 7
6	EP6_INT	R	Прерывания от конечной точки номер 6
5	EP5_INT	R	Прерывания от конечной точки номер 5
4	EP4_INT	R	Прерывания от конечной точки номер 4
3	EP3_INT	R	Прерывания от конечной точки номер 3
2	EP2_INT	R	Прерывания от конечной точки номер 2
1	EP1_INT	R	Прерывания от конечной точки номер 1
0	EP0_INT	R	Прерывания от конечной точки номер 0
Примечание – R – регистр может читаться, W – в регистр можно записывать.			

В таблицах 8.10 и 8.11 приведены структура и описание битов регистра подсчёта кадров и времени.

Значения битов после сброса 0000h.

Таблица 8.10 – Структура регистра подсчёта кадров и времени

31 – 28	27	26 – 16	15 – 12	11 – 0
FRCOUNT	RSRV	FRNO	RSRV	TIMESOF
R	R	R	R	R
0	0	0	0	0

Таблица 8.11 – Описание битов регистра подсчёта кадров и времени

Бит	Имя	Доступ	Описание
31–28	FRCOUNT	R	Количество кадров с одинаковым номером кадров (это поле может использоваться для определения микрокадров)
27	RSRV	R	Зарезервирован
26–16	FRNO	R	Номер кадра, полученный от маркера SOF
15–12	RSRV	R	Зарезервирован
11–0	TIMESOF	R	Время от последнего маркера SOF с точностью 0,5 мкс
Примечание – R – регистр может читаться, W – в регистр можно записывать.			

В таблицах 8.12 и 8.13 приведены структура и описание битов регистра управления/состояния конечной точки (EPn_CSR).

Значения битов после сброса 0000h.

Таблица 8.12 – Структура регистра управления/состояния конечной точки

31–28	27–26	25–24	23–22	21–18	
RSRV	EPTYPE	TRTYPE	HALT	EPNO	
R	RW	RW	RW	RW	
0	0	0	0	0	
17	16	15–14	13	12–11	10–0
NSETUP	ADV17	RSRV	NIN	TRCOUNT	MAXPL
RW	RW	R	RW	RW	RW
0	0	0	0	0	0

Таблица 8.13 – Описание битов регистра управления/состояния конечной точки

Бит	Имя	Доступ	Описание
31–28	RSRV	R	Зарезервировано
27–26	EPTYPE	RW	Тип конечной точки 00 – CONTROL (управление) 01 – IN (входная) 10 – OUT (выходная)
25–24	TRTYPE	RW	Тип передачи 00 – по прерыванию 01 – изохорная 10 – пакетная
23–22	HALT	RW	Временная блокировка конечной точки 00 – нормальная работа 01 – USB игнорирует все обращения к этой конечной точке 10 – конечная точка устанавливается в состояние HALT 11 – конечная точка устанавливается в состояние BUSY (формируется NACK пакет для всех обменов)
21–18	EPNO	RW	Номер конечной точки
17	NSETUP	RW	При установке бита в «1» конечная точка формирует NACK пакет при получении SETUP пакетов. Эта особенность обеспечивает медленным обменам подготовку данных в фазах обмена DATA и STATUS
16	ADV17	RW	При установке бита в «1» бит 17 автоматически устанавливается в «1» после следующего успешного обмена
15–14	RSRV	R	Зарезервировано
13	NIN	RW	При установке бита в «1» конечная точка формирует NACK пакет вместо нулевого пакета данных, когда IN буфер пуст
12–11	TRCOUNT	RW	Количество обменов в течение микрокадра
10–0	MAXPL	RW	Максимальный размер пакетных данных в байтах
Примечание – R – регистр может читаться, W – в регистр можно записывать.			

В таблицах 8.14 и 8.15 приведены структура и описание битов регистра маскирования/источника прерывания конечной точки EPn_INT.

Значения битов после сброса 0000h.

Таблица 8.14 – Структура регистра маскирования/источника прерывания конечной точки

31	30	29	28	27	26	25	24
RSRV	A_TD	A_RD	A_TFIFO	A_RFIFO	A_MAXPL	A_PIDSEQ	A_BUEMP
R	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16
A_BUFULL	A_RCNTRL	A_CRC16	A_TIMEOUT	B_TFIFO	B_RFIFO	B_MAXPL	B_PIDSEQ
RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8
B_BUEMP	B_BUFULL	B_RCNTRL	B_CRC16	B_TIMEOUT	TD	RD	TFIFO
RW	RW	RW	RW	RW	R	R	R
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
RFIFO	MAXPL	PIDSEQ	BUEMP	BUFULL	RCNTRL	CRC16	TIMEOUT
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0

Таблица 8.15 – Описание битов регистра маскирования/источника прерывания конечной точки

Бит	Имя	Доступ	Описание
1	2	3	4
31	RSRV	R	Зарезервировано
30	A_TD	RW	Маска прерывания A – пакет данных передан
29	A_RD	RW	Маска прерывания A – пакет данных принят
28	A_TFIFO	RW	Маска прерывания A – FIFO передачи заполнен
27	A_RFIFO	RW	Маска прерывания A – FIFO приёмника пуст
26	A_MAXPL	RW	Маска прерывания A – принятый пакет больше максимального размера
25	A_PIDSEQ	RW	Маска прерывания A – ошибка последовательности идентификаторов
24	A_BUEMP	RO	Маска прерывания A – буфер конечной точки пуст
23	A_BUFULL	RW	Маска прерывания A – буфер конечной точки заполнен
22	A_RCNTRL	RW	Маска прерывания A – принят пакет управления
21	A_CRC16	RW	Маска прерывания A – ошибка контрольной суммы пакета
20	A_TIMEOUT	RW	Маска прерывания A – превышено время ожидания (ожидаются пакеты ACK или DATA)
19	B_TFIFO	RW	Маска прерывания B – FIFO передачи заполнен
18	B_RFIFO	RW	Маска прерывания B – FIFO приёмника пуст
17	B_MAXPL	RW	Маска прерывания B – принятый пакет больше максимального размера
16	B_PIDSEQ	RW	Маска прерывания B – ошибка последовательности идентификаторов
15	B_BUEMP	RW	Маска прерывания B – буфер конечной точки пуст
14	B_BUFULL	RW	Маска прерывания B – буфер конечной точки заполнен
13	B_RCNTRL	RW	Маска прерывания B – принят пакет управления
12	B_CRC16	RW	Маска прерывания B – ошибка контрольной суммы пакета
11	B_TIMEOUT	RW	Маска прерывания B – превышено время ожидания
10	TD	R	Прерывание – пакет данных передан
9	RD	R	Прерывание – пакет данных принят
8	TFIFO	R	Прерывание – FIFO данных заполнен
7	RFIFO	R	Прерывание – FIFO приёмника пуст
6	MAXPL	R	Прерывание – принятый пакет больше максимального размера (определяется битами 10:0 EPn_CSR)

Окончание таблицы 8.15

1	2	3	4
5	PIDSEQ	R	Прерывание – ошибка последовательности идентификаторов
4	BUEMP	R	Прерывание – буфер конечной точки пуст
3	BUFULL	R	Прерывание – буфер конечной точки заполнен
2	RCNTRL	R	Прерывание – принят пакет управления
1	CRC16	R	Прерывание – ошибка контрольной суммы пакета
0	TIMEOUT	R	Прерывание – превышено время ожидания (ожидаются пакеты ACK или DATA)
Примечание – R – регистр может читаться, W – в регистр можно записывать.			

В таблицах 8.16 и 8.17 приведены структура и описание битов регистра буфера конечной точки (EPn_OBA, EPn_IBA).

Значения битов после сброса 0000h.

Таблица 8.16 – Структура регистра буфера конечной точки

31–17	16–0
BUFSZ	BUFPT
RW	RW
0	0

Таблица 8.17 – Описание битов регистра буфера конечной точки

Бит	Имя	Доступ	Описание
31–17	BUFSZ	RW	Размер буфера (максимум 16 383 байт)
16–0	BUFPT	RW	Указатель буфера
Примечание – R – бит может читаться, W – в бит можно записывать.			

9 Описание периферийного устройства MIL-STD-1553

Контроллер МКО (магистрального канала обмена) MIL-STD-1553 предназначен для подключения к магистральному последовательному интерфейсу (МПИ) системы электронных модулей в соответствии с ГОСТ Р 52070-2003 и выполнения функций контроллера (К), оконечного устройства (ОУ) и монитора (М).

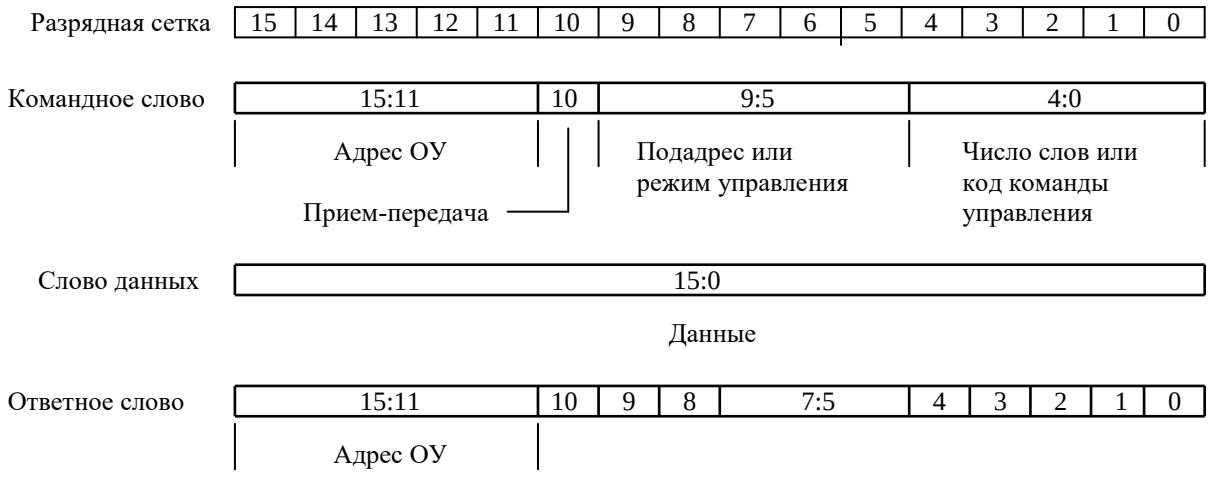
MIL-STD-1553 имеет возможность одновременно функционировать в режимах контроллера, оконечного устройства и монитора. Одновременное функционирование в различных режимах может использоваться при тестировании MIL-STD-1553.

Устройство MIL-STD-1553 подключается к двум шинам магистрального интерфейса (основной и резервной) по схеме подключения ответвителя с согласующим трансформатором. Устройство MIL-STD-1553 обеспечивает реализацию всех команд, предусмотренных стандартом как в режиме контроллера, так и в режиме оконечного устройства.

Сообщения, которыми обменивается устройство MIL-STD-1553 с другими устройствами интерфейса, состоят из командных (CW), ответных слов (AW) и слов данных (DW). Их структура приведена на рисунке 9.1.

Если содержимое CW[9–5] ≠ 00000 или CW[9–5] ≠ 11111, то данная команда является либо командой приёма информации (RC), выдаваемой из К в ОУ (при CW[10] = 0), либо командой выдачи информации (TR) из ОУ в К (при CW[10]=1). Количество передаваемых слов при этом задаётся в CW[4–0]. Команды RC и TR, выдаваемые контроллером одна за другой без пауз, инициируют команду обмена (EX) между оконечными устройствами, которым они адресованы.

Если содержимое CW[9–5] = 00000 или CW[9–5] = 11111, то данная команда является одной из команд управления, приведённых в таблице 9.1.



10 - Ошибка в сообщении
9 - Признак передачи ответного слова
8 - Запрос на обслуживание
7 - 5 – Резерв
4 - Принята групповая команда
3 - Абонент занят
2 - Неисправность абонента
1 - Принято управление интерфейсом
0 - Неисправность ОУ

Рисунок 9.1 – Структура передаваемых слов

Таблица 9.1 – Команды управления устройства MIL-STD-1553

Обозначение команды	Наименование команды	CW[10]	CW[4–0]	Применение в групповом сообщении	Применение со словом данных
RCCC	Принять управление каналом	1	00000	нет	нет
S	Синхронизация	1	00001	да	нет
TRAW	Передать ответное слово	1	00010	нет	нет
BDS	Начать самоконтроль ОУ	1	00011	да	нет
BLTR	Блокировать передатчик	1	00100	да	нет
RBLTR	Разблокировать передатчик	1	00101	да	нет
BLER	Блокировать признак неисправности ОУ	1	00110	да	нет
RBLER	Разблокировать признак неисправности ОУ	1	00111	да	нет
RES	Установить ОУ в исходное состояние	1	01000	да	нет
	Резерв		01001–01111		
TRVW	Передать векторное слово	1	10000	нет	да
SD	Синхронизация со словом данных	0	10001	да	да
TRLCW	Передать последнее команд-ное слово	1	10010	нет	да
TRBCW	Передать слово встроенного контроля ОУ	1	10011	нет	да
BLTRI	Блокировать i-й передатчик	0	10100		да
RBLTRI	Разблокировать i-й передатчик	0	10101		да
	Резерв		10110–11111		

9.1 Описание регистров периферийного устройства MIL-STD-1553

Память MIL-STD-1553 включает управляющие регистры и оперативное запоминающее устройство (RAM) объёмом 3K × 18 (два разряда – контрольные). Выборка MIL-STD-1553 осуществляется по сигналу _CS0/4_S. Для обращения к регистрам и памяти MIL-STD-1553 используется 14-разрядный адрес AS[13–0]. Распределение общего адресного пространства MIL-STD-1553 приведено в таблице 9.2, а адреса конкретных регистров – в таблице 9.3.

Таблица 9.2 – Распределение адресного пространства MIL-STD-1553

Диапазон адресов	Устройство
0050 0000h – 0050 000Dh	Регистры MIL-STD-15530
0050 1000h – 0050 1BFFh	RAM MIL-STD-15530

Таблица 9.3 – Распределение регистров в адресном пространстве MIL-STD-1553

Адрес	Разряды шины DS 31–16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	Тип доступа
0050 0000h	0 Запись 0h приводит к сбросу MIL-STD-1553	wr
0050 0001h	0 резерв CNR0[8–0]	wr/rd
0050 0002h	0 резерв CNR1[6–0]	wr/rd
0050 0003h	0 резерв ewr[3–0][1] CNR2[3–0]	wr/rd
0050 0004h[2]	0 резерв ACIO[11–0]	wr/rd
0050 0005h[3]	0 резерв ACIO[11–0]	wr/rd
0050 0006h[4]	0 резерв ACIO[11–0]	wr/rd
0050 0007h[5]	0 резерв ACIO[11–0]	wr/rd
0050 0008h	0 WATCH0[6]	wr/rd
0050 0009h	0 резерв WATCH1[6]	wr/rd
0050 000Ah	0 резерв ISR [12–0][7]	wr/rd
0050 000Bh	0 резерв ISIO[11–0]	rd
0050 000Ch	0 резерв IER[5–0]	wr/rd
0050 000Dh	0 резерв[8] BCW[4–0]	rd

[1] ewr[i]=DS[i+4] (i=3–0). Условие ewr[i]=1 при записи в регистр CNR2 обеспечивает запись содержимого разряда DS[i] в CNR[i].

[2] Запись начального адреса команды ввода-вывода в регистр ACIO по адресу 0004h инициирует исполнение любой команды ввода-вывода в режиме контроллера, содержащейся в RAM по записываемому адресу, за исключением команды EX (обмен информацией между ОУ).

[3] Запись начального адреса команды ввода-вывода в регистр ACIO по адресу 0005h инициирует исполнение команды EX в режиме контроллера, находящейся в RAM по записываемому адресу.

[4] Запись адреса в регистр ACIO по адресу 0006h инициирует циклическую выдачу в одну из линий интерфейса кода, находящегося в ячейке RAM по записываемому адресу, с синхросигналом командного слова. Код выдаётся в линию 33 раза без пауз, после 33-й передачи выдерживается пауза длительностью (5 – 7) мкс, и цикл выдачи возобновляется.

[5] Запись адреса в регистр ACIO по адресу 0007h инициирует циклическую выдачу в одну из линий интерфейса кода, находящегося в ячейке RAM по записываемому адресу, с синхросигналом слова данных. Код выдаётся в линию 33 раза без пауз, после 33-й передачи выдерживается пауза длительностью (5 – 7) мкс, и цикл выдачи возобновляется.

[6] При установке значения часов необходимо сначала записать младшие разряды значения часов, а затем старшие. При считывании содержимого часов сначала считываются младшие разряды, а затем старшие.

[7] Запись в регистр ISR используется для обнуления разрядов. Запись «1» приводит к обнулению соответствующего разряда регистра.

[8] При считывании содержимого регистров резервные разряды выдаются в шину DS нулевыми значениями.

Распределение ячеек оперативной памяти RAM в адресном пространстве MIL-STD-1553 приведено в таблице 9.4.

Таблица 9.4 – Распределение ячеек оперативной памяти RAM

Адрес	Назначение ячейки
1	2
0050 1000h	LCW – последнее командное слово, полученное ОУ
0050 1001h– 0050 101Eh	CNIN1 – CNIN30 – используются при вводе информации во время исполнения команды RC с подадресами 00001–11110
0050 101Fh	Не используется

Окончание таблицы 9.4

1	2
0050 1020h	DW_RT – используется в режиме ОУ для приёма и хранения: слова данных DW_SD, полученного при исполнении команды SD слова данных DW_ BLTRI, полученного при исполнении команды BLTRI слова данных DW_ RBLTRI, полученного при исполнении команды RBLTRI
0050 1021h– 0050 103Eh	CNOUT1 – CNOUT30 – используются при выводе информации во время исполнения команды TR с подадресами 00001–11110
0050 103Fh	Не используется
0050 1040h	Используется ОУ для хранения VW, выдаваемого контроллеру при исполнении команды TRVW
0050 1041h– 0050 105Eh	CNING1 – CNING30 – используются при вводе информации во время исполнения групповой команды RC с подадресами 00001–11110
0050 105Fh	Не используется
0050 1060h	AW1 – используется в режиме контроллера для приёма и хранения ответных слов при исполнении всех команд за исключением команды EX (обмен между ОУ), а при исполнении команды EX для приёма и хранения ответного слова, полученного от принимающего ОУ
0050 1061h	DW_C – используется в режиме контроллера При исполнении команды EX используется для приёма и хранения ответного слова (AW2), полученного от передающего ОУ При исполнении команды TRVW используется для приёма и хранения векторного слова (VW), полученного от ОУ При исполнении команды TRLCW используется для приёма и хранения последнего командного слова (LCW), полученного от ОУ При исполнении команды TRBCW используется для приёма и хранения слова встроенного контроля (BCW), полученного от ОУ
0050 1062h– 0050 1BFFh	В режиме контроллера могут использоваться для хранения команд ввода-вывода и принимаемых и выдаваемых данных В режиме ОУ могут использоваться для хранения принимаемых и выдаваемых данных
0050 1800h– 0050 1BFFh	Используются в режиме монитора для хранения принимаемых сообщений

9.2 Регистр программного сброса RESET

Запись 0h по адресу 0000h приводит к сбросу MIL-STD-1553.

Таблица 9.5 – Регистр управления CNR0. Адрес: 0001

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								BLGRC	ECC	CART	ART					
0																1
R								RWC								

Таблица 9.6 – Назначение битов CNR0

Биты	Назначение	Тип доступа	Значение после сброса
1	2	3	4
8	BLGRC – блокировка приёма групповой команды. Условие BLGRC=1 блокирует приём групповых команд, т. е. команд, у которых адрес ОУ равен 11111	Запись/чтение	0
7	ECC – разрешение управления каналом. Условие ECC = 1 разрешает MIL-STD-1553, функционирующему в режиме ОУ, принять управление каналом при получении им от контроллера команды ПРИНЯТЬ УПРАВЛЕНИЕ КАНАЛОМ (RCCC)	Запись/чтение	0

Окончание таблицы 9.6

1	2	3	4
6	CART – коммутация адреса оконечного устройства (ОУ). Условие CART=1 задаёт адрес ОУ не по содержимому ART[5–0], а фиксированное значение 3Eh	Запись/чтение	0
5–0	ART[5–0] – адрес оконечного устройства. ART[5] –старший разряд адреса ART[0] – контрольный разряд адреса, дополняющий число 1 в адресе до нечётного значения	Запись/чтение	000001
Примечание – При CART = 0 считывается содержимое регистра ART[5–0]. При CART=1 считывается значение адреса ОУ, скоммутированное на внешнем разъёме ИС. Подключение 0 В на контакт разъёма, соответствующего некоторому разряду адреса, задаёт значение логической единицы данного разряда.			

Таблица 9.7 – Регистр управления CNR1. Адрес: 0002

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										EXP	NL	CC	AB	RG_M	RG_RT	RG_C
0																
R										RWC						

Таблица 9.8 – Назначение битов CNR1

Биты	Назначение	Тип доступа	Значение после сброса
6	EXP – изменение значения контрольного разряда, выдаваемого в шину слова таким образом, чтобы число «1» в слове, включая контрольный разряд, было чётным (т. е. неправильным). Используется при тестировании MIL-STD-1553	Запись/чтение	0
5	NL – номер линии. NL = 0 задаёт работу MIL-STD-1553 в режимах контроллера и монитора по основной шине LA магистрального интерфейса. NL = 1 задаёт работу MIL-STD-1553 в режимах контроллера и монитора по резервной шине LB магистрального интерфейса	Запись/чтение	0
4	CC – коммутация каналов. Условие CC=1 обеспечивает блокировку выдачи информации из MIL-STD-1553 в шины интерфейса, блокировку приёма информации в MIL-STD-1553 из шин интерфейса и внутреннюю коммутацию выхода кодирующей схемы на вход декодирующей схемы, задаваемой значением NL	Запись/чтение	0
3	AB – аппаратный бит. Условие AB = 1 задаёт режим работы MIL-STD-1553 с аппаратным битом. В этом случае ОУ принимает только те CW, у которых CW[9] = 1	Запись/чтение	0
2	RG_M – режим монитора. Условие RG_M = 1 задаёт работу MIL-STD-1553 в режиме монитора	Запись/чтение	0
1	RG_RT – режим оконечного устройства. Условие RG_RT = 1 задаёт работу MIL-STD-1553 в режиме оконечного устройства	Запись/чтение	0
0	RG_C – режим контроллера. Условие RG_C=1 задаёт работу MIL-STD-1553 в режиме контроллера	Запись/чтение	0

Таблица 9.9 – Регистр управления CNR2. Адрес: 0003

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													BLTR2	BLTR1	ERA	RQS
0																
R													RWC			

Таблица 9.10 – Назначение битов CNR2

Биты	Назначение	Тип доступа	Значение после сброса
3	BLTR2 – блокировка выдачи информации в линию LB. Может устанавливаться в 1 при получении от контроллера команды БЛОКИРОВАТЬ ПЕРЕДАТЧИК (BLTR) и сбрасываться в 0 при получении команды РАЗБЛОКИРОВАТЬ ПЕРЕДАТЧИК (RBLTR) по линии LA	Запись[1)]/чтение	0
2	BLTR1 – блокировка выдачи информации в линию LA. Может устанавливаться в 1 при получении от контроллера команды БЛОКИРОВАТЬ ПЕРЕДАТЧИК (BLTR) и сбрасываться в 0 при получении команды РАЗБЛОКИРОВАТЬ ПЕРЕДАТЧИК (RBLTR) по линии LB	Запись[1)]/чтение	0
1	ERA – неисправность абонента используется при формировании признака НЕИСПРАВНОСТЬ АБОНЕНТА в AW[2]. При формировании ответного слова AW[2] = ERA	Запись[1)]/чтение	0
0	RQS – запрос обслуживания. Используется при формировании признака ЗАПРОС ОБСЛУЖИВАНИЯ в AW[8]. При формировании ответного слова AW[8] = RQS	Запись[1)]/чтение	0

[1)] Условие ewr[i]=1 (ewr[i]=DS[i+4]) при записи в регистр CNR2 обеспечивает запись содержимого разряда DS[i] в CNR[i].

Таблица 9.11 – Регистр адреса команды ввода-вывода ACIO. Адрес: 0004, 0005, 0006, 0007

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					ACIO											
0																
R					RWC											

Таблица 9.12 – Назначение битов ACIO

Биты	Назначение	Тип доступа	Значение после сброса
11–00	ACIO[11–0] – начальный адрес команды ввода-вывода	Запись/чтение	0

Регистр ACIO доступен по записи и чтению по адресам: 0004h, 0005h, 0006h, 0007h. По сбросу регистр ACIO устанавливается в нулевое состояние.

Запись начального адреса команды ввода-вывода в регистр ACIO по адресу 0004h инициирует исполнение любой команды ввода-вывода в режиме контроллера, содержащейся в RAM по записываемому адресу, за исключением команды EX (обмен информацией между ОУ).

Запись начального адреса команды ввода-вывода в регистр ACIO по адресу 0005h инициирует исполнение команды EX в режиме контроллера, находящейся в RAM по записываемому адресу.

Запись адреса в регистр ACIO по адресу 0006h инициирует циклическую выдачу в одну из линий интерфейса кода, находящегося в ячейке RAM по записываемому адресу, с синхросигналом командного слова. Код выдаётся в линию 33 раза без пауз, после 33-й передачи выдерживается пауза длительностью (5 – 7) мкс, и цикл выдачи возобновляется.

Запись адреса в регистр ACIO по адресу 0007h инициирует циклическую выдачу в одну из линий интерфейса кода, находящегося в ячейке RAM по записываемому адресу, с синхросигналом слова данных. Код выдаётся в линию 33 раза без пауз, после 33-й передачи выдерживается пауза длительностью (5 – 7) мкс., и цикл выдачи возобновляется.

После считывания очередного слова команды ввода-вывода из RAM содержимое регистра ACIO увеличивается на единицу.

Таблица 9.13 – Регистр младших разрядов значения часов WATCH0. Адрес: 0008

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	W_S						W_MS									
0																
RWC																

Таблица 9.14 – Назначение битов WATCH0

Биты	Назначение	Тип доступа	Значение после сброса
15–10	W_S[5–0] – значение в секундах	Запись/чтение	0
09–00	W_MS[9–0] – значение в миллисекундах	Запись/чтение	0
Примечание – При установке значения часов необходимо сначала записать младшие разряды значения часов, а затем старшие. При считывании содержимого часов сначала считываются младшие разряды, а затем старшие.			

Таблица 9.15 – Регистр старших разрядов значения часов WATCH1. Адрес: 0009

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						W_H					W_M					
0																
R						RWC										

Таблица 9.16 – Назначение битов WATCH1

Биты	Назначение	Тип доступа	Значение после сброса
10–06	W_H[4–0] – значение в часах	Запись/чтение	0
05–00	W_M[5–0] – значение в минутах	Запись/чтение	0

Таблица 9.17 – Регистр причин прерываний ISR. Адрес: 000A

31–16	15	14	13	12	11	10	9	8
0				I_ERRAM	I_M	I_RBLTRI	I_BLTRI	I_RES
0								
R				RWC				
7	6	5	4	3	2	1	0	
I_RCCC	I_BDS	I_TRVW	I_SD	I_S	I_IO	I_ERART	I_C	
0								
RWC								

Таблица 9.18 – Назначение битов ISR

Биты	Назначение	Тип доступа	Значение после сброса
12	INT_ER_RAM – прерывание, формируемое при обнаружении ошибки памяти RAM во время её чтения процессором	Запись[1]/чтение	0
11	INT_M – прерывание, формируемое в режиме монитора, при получении сообщения из линии интерфейса, определяемой значением NL	Запись[2]/чтение	0
10	INT_RBLTRI – прерывание при завершении исполнения команды РАЗБЛОКИРОВАТЬ i-й ПЕРЕДАТЧИК (RBLTRI) в режиме ОУ	Запись[1]/чтение	0
9	INT_BLTRI – прерывание при завершении исполнения команды БЛОКИРОВАТЬ i-й ПЕРЕДАТЧИК (BLTRI) в режиме ОУ	Запись[1]/чтение	0
8	INT_RES – прерывание при завершении исполнения команды СБРОС ОУ (RES) в режиме ОУ	Запись[1]/чтение	0
7	INT_RCCC – прерывание при завершении исполнения команды ПРИНЯТЬ УПРАВЛЕНИЕ КАНАЛОМ (RCCC) в режиме ОУ. Устанавливается только при выполнении условия ECC=1	Запись[1]/чтение	0
6	INT_BDS – прерывание при завершении исполнения команды НАЧАТЬ САМОКОНТРОЛЬ (BDS) в режиме ОУ	Запись[1]/чтение	0
5	INT_TRVW – прерывание при завершении исполнения команды ПЕРЕДАТЬ ВЕКТОРНОЕ СЛОВО (TRVW) в режиме ОУ	Запись[1]/чтение	0
4	INT_SD – прерывание при завершении исполнения команды СИНХРОНИЗАЦИЯ СО СЛОВОМ ДАННЫХ (SD) в режиме ОУ	Запись[1]/чтение	0
3	INT_S – прерывание при завершении исполнения команды СИНХРОНИЗАЦИЯ (S) в режиме ОУ	Запись[1]/чтение	0
2	INT_IO – прерывание при завершении команд приёма (RC) или выдачи (TR) информации в режиме ОУ	Запись[3]/чтение	0
1	INT_ER_ART – прерывание при чётном числе 1 (включая контрольный разряд) в адресе ОУ. Адрес ОУ может задаваться либо CNR0[5–0], либо коммутацией на разъёме ИС	Запись[4]/чтение	0
0	INT_C – прерывание при завершении исполнения команды в режиме контроллера[5]	Запись[1]/чтение	0

[1] Запись 1 в данный разряд регистра приводит к установке его в 0.

[2] INT_M устанавливается в 0 путём такого количества записей 1 в ISR[11], которое соответствует количеству принятых монитором сообщений.

[3] INT_IO устанавливается в нулевое состояние только после считывания всех хранящихся в буфере FIFO слов состояния прерываний по вводу-выводу ISIO.

[4] Установка INT_ER_ART в 0 может быть выполнена только путём записи в CNR0[5–0] кода, содержащего нечётное число 1 (при CART = 0) либо путём перекоммутации кода на разъёме ИС таким образом, чтобы число 1 в коде содержало нечётное количество 1 (при CART=1).

[5] Т. к. исполнение команды в режиме контроллера может завершиться как без ошибок, так и с ошибками, то после обнаружения прерывания INT_C необходимо проверить содержимое регистра встроенного контроля BCW, которое индицирует причину завершения исполнения команды.

Таблица 9.19 – Слово состояния прерывания ввода-вывода ISIO. Адрес: 000B

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					CW	GRC	CW									
0																
R					RC											

После каждого успешного (без ошибок) завершения приёма информации по команде RC или выдачи информации по команде TR ОУ формирует и записывает в буфер слово состояния прерывания ввода-вывода (ISIO). Буфер состояний представляет собой FIFO на восемь 12-разрядных слов. Первым при чтении ISIO считывается первое записанное слово состояния.

Таблица 9.20 – Назначение битов ISIO

Биты	Назначение	Тип доступа	Значение после сброса
11	Признак приёма или передачи исполненной команды (CW[10])	чтение	0
10	Признак приёма групповой команды	чтение	0
9–5	Значение поля подадреса исполненной команды (CW[9–5])	чтение	00000
4–0	Значение поля числа слов исполненной команды (CW[4–0])	чтение	00000

Таблица 9.21 – Регистр разрешения прерываний IER. Адрес: 000C

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											EI_ERR	EI_M	EI_CC	EI_IO	EI_ERA	EI_C
0																
R											RWC					

Таблица 9.22 – Назначение битов IER

Биты	Назначение	Тип доступа	Значение после сброса
5	Разрешение формирования прерываний при обнаружении ошибки в считываемой модулем вычислительным из памяти RAM информации	Запись/чтение	0
4	Разрешение формирования прерываний в режиме монитора	Запись/чтение	0
3	Разрешение формирования прерываний в режиме ОУ по завершении исполнения команд управления	Запись/чтение	0
2	Разрешение формирования прерываний в режиме ОУ по завершении исполнения команд RC или TR	Запись/чтение	0
1	Разрешение формирования прерываний при нарушении нечётности в адресе ОУ	Запись/чтение	0
0	Разрешение формирования прерываний в режиме контроллера	Запись/чтение	0
Примечание – Разрешение прерывания задаётся единичным значением в соответствующем разряде регистра.			

Таблица 9.23 – Регистр встроенного контроля BCW. Адрес: 000D

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												BCW				
0																
R												RC				

Таблица 9.24 – Назначение битов BCW

Биты	Назначение	Тип доступа	Значение после сброса
04–00	На регистре BCW фиксируется код, определяющий причину завершения исполнения команды в режиме контроллера. Ниже приведены коды и соответствующие им причины, вызвавшие завершение исполнения команды. 00001 Обнаружена ошибка при чтении RAM 00010 Принято к исполнению недопустимое командное слово 00011 Зафиксировано отсутствие ответного слова AW1 00100 Зафиксировано отсутствие ответного слова AW2 00101 Зафиксировано отсутствие паузы после AW1 00110 Число слов в сообщении меньше заданного 00111 Число слов в сообщении больше заданного 01000 Недопустимый синхросигнал при приёме AW1 01001 Недопустимый синхросигнал при приёме AW2 01010 Недопустимый синхросигнал при приёме слова данных 01011 Обнаружена ошибка паритета при приёме слова из интерфейса 01100 Обнаружена ошибка кода при приёме слова из интерфейса 01101 Ошибка сравнения выданного и возвращённого из интерфейса слов 01110 Отсутствие начала приёма возвращаемого слова из интерфейса при его выдаче 01111 Обнаружена ошибка кода в возвращаемом из интерфейса слове при его выдаче 10000 Превышение допустимого времени исполнения команды	чтение	00000
Примечание – Регистр BCW сбрасывается также при инициировании исполнения команды в режиме контроллера.			

9.3 Назначение ячеек памяти RAM

Таблица 9.25 – Ячейка последнего командного слова LCW. Адрес: 1000

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCW																
–																
R																

Ячейка LCW используется для хранения CW, получаемого ОУ из интерфейса. В ячейку LCW записываются все получаемые CW за исключением CW, задающего команду TRLCW.

Таблица 9.26 – Ячейки управления приёмом слов данных с непосредственной адресацией CNINi. Адрес: 1001–101E

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	F	RGINT	RGF	0	AM											
–																
R																

Ячейки CNINi используются ОУ для управления приёмом информации при исполнении команды RC. Назначение битов ячейки CNINi описано в таблице 9.27.

Таблица 9.27 – Назначение битов CNINi

Биты	Назначение
15	F – значение флага. При условии F&RGF=01 запись принимаемых слов разрешается после завершения приёма и записи всех слов данных F=1. При условии F&RGF =11 запись принимаемых слов запрещается и формируется признак ЗАНЯТОСТЬ АБОНЕНТА в ответном слове. При RGF=0 анализ флага не производится.
14	RGINT – задаёт режим работы ОУ с прерыванием по завершении исполнения команды RC. После записи всех слов данных в RAM при условии RGINT=1 в буфер FIFO записывается значение ISIO и формируется прерывание INT_IO.
13	RGF – задаёт режим работы ОУ с флагом, содержащимся в 15-м разряде.
12	0
11–0	AM – начальный адрес памяти для записи принимаемых данных

Таблица 9.28 – Ячейка приёма слова данных в режиме ОУ DW_RT. Адрес: 1020

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	DW_RT															
–																
R																

Ячейка DW_RT используется для приёма и хранения слов данных: DW_SD, DW_BLTRI, DW_RBLTRI, полученных при исполнении команд SD, BLTRI, RBLTRI соответственно.

Таблица 9.29 – Ячейки управления выдачей слов данных CNOUTi. Адрес: 1021–103E

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	F	RGINT	RGF	0	AM											
–																
R																

Ячейки CNOUTi используются ОУ для управления выдачей информации при исполнении команды TR. Назначение битов ячейки CNOUTi описано в таблице 9.30.

Таблица 9.30 – Назначение битов CNOUTi

Биты	Назначение
15	F – значение флага. При условии F&RGF=11 выдача слов разрешается, после завершения выдачи всех слов данных F=0. При условии F&RGF =01 выдача слов запрещается и формируется признак ЗАНЯТОСТЬ АБОНЕНТА в ответном слове. При RGF=0 анализ флага не производится.
14	RGINT – задаёт режим работы ОУ с прерыванием по завершении исполнения команды TR. После выдачи всех слов данных из RAM при условии RGINT=1 в буфер FIFO записывается значение ISIO и формируется прерывание INT_IO
13	RGF – задаёт режим работы ОУ с флагом, содержащимся в 15-м разряде
12	0
11–0	AM – начальный адрес памяти для записи принимаемых данных

Таблица 9.31 – Ячейка хранения векторного слова VW. Адрес: 1040

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	VW															
–																
R																

Ячейка VW используется для хранения векторного слова в режиме ОУ, выдаваемого контроллеру при получении от него команды TRVW.

Таблица 9.32 – Ячейки управления приёмом слов данных с групповой адресацией CNINi. Адрес: 1041–105E

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	F	RGINT	RGF	0	AM											
–																
R																

Ячейки CNINGi используются ОУ для управления приёмом информации при исполнении команды RC с групповой адресацией. Назначение битов ячейки CNINGi описано в таблице 9.33.

Таблица 9.33 – Назначение битов CNINGi

Биты	Назначение
15	F – значение флага. При условии F&RGF=01 запись принимаемых слов разрешается, после завершения приёма и записи всех слов данных F=1. При условии F&RGF =11 запись принимаемых слов запрещается и формируется признак ЗАНЯТОСТЬ АБОНЕНТА в ответном слове. При RGF=0 анализ флага не производится
14	RGINT – задаёт режим работы ОУ с прерыванием по завершении исполнения команды RC. После записи всех слов данных в RAM при условии RGINT=1 в буфер FIFO записывается значение ISIO и формируется прерывание INT_IO
13	RGF – задаёт режим работы ОУ с флагом, содержащимся в 15-м разряде
12	0
11–0	AM – начальный адрес памяти для записи принимаемых данных

Таблица 9.34 – Ячейка для приёма первого ответного слова AW1. Адрес: 1060

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	AW1															
–																
R																

Ячейка AW1 используется в режиме контроллера для приёма и хранения ответных слов (при исполнении команды EX для приёма и хранения ответного слова, полученного от принимающего ОУ).

Таблица 9.35 – Ячейка для приёма информации, полученной от ОУ контроллером DW_C. Адрес: 1061

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DW_C															
–																
R																

Ячейка DW_C используется в режиме контроллера:

- при исполнении команды EX для приёма и хранения ответного слова (AW2), полученного от передающего ОУ;
- при исполнении команды TRVW для приёма и хранения векторного слова (VW), полученного от ОУ;
- при исполнении команды TRLCW для приёма и хранения последнего командного слова LCW, полученного от ОУ;
- при исполнении команды TRBCW для приёма и хранения слова встроенного контроля (BCW), полученного от ОУ.

9.4 Структура команд ввода-вывода в режиме контроллера

Команды ввода-вывода, выполняемые MIL-STD-1553 в режиме контроллера, должны быть предварительно записаны в память RAM MIL-STD-1553. Команды состоят из одного или из двух слов. Первым словом команд RC, TR, SD, BLTRI, RBLTRI является непосредственно CW, выдаваемое ОУ. Вторым словом этих команд является начальный адрес памяти RAM откуда считываются выдаваемые данные либо куда принимаемые данные записываются. Для команды EX первым словом является CW1, выдаваемое принимающему (принимающим) ОУ, а вторым словом – CW2, выдаваемое передающему ОУ. Все остальные команды состоят только из одного CW.

9.5 Структура слова встроенного контроля в режиме оконечного устройства

Слово встроенного контроля ОУ (BCW_RT), считываемое из ОУ по команде TRBCW, позволяет определить тип ошибки, вызвавшей неправильное исполнение текущей команды. Структура BCW_RT имеет следующий вид.

Таблица 9.36 – Структура BCW_RT

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												BCW_RT				
–																
R																

Типы ошибок и соответствующие им коды BCW_RT приведены в таблице 9.37.

Таблица 9.37 – Типы ошибок в BCW_RT

Код	Наименование ошибки
00001	Принято недопустимое командное слово
00010	Отсутствие паузы после приёма CW1
00011	Число слов в сообщении меньше заданного
00100	Число слов в сообщении больше заданного
00101	Ошибка синхросигнала AW2
00110	Ошибка синхросигнала DW
00111	Отсутствие паузы после приёма CW2
01000	Отсутствие AW2
01001	Ошибка кода в принимаемом слове
01010	Обнаружена ошибка паритета при приёме слова из интерфейса
01011	Ошибка сравнения выданного и возвращённого из интерфейса слов
01100	Отсутствие начала приёма возвращаемого слова из интерфейса при его выдаче
01101	Обнаружена ошибка кода в возвращаемом из интерфейса слове при его выдаче
01110	Обнаружена ошибка при чтении RAM
10000	Превышение допустимого времени выдачи информации в интерфейс
10001	Превышение допустимого времени исполнения команды

9.6 Структура сообщений в режиме монитора

Структура сообщений, фиксируемых MIL-STD-1553 в RAM при функционировании в режиме монитора, состоит из слова состояния монитора (SWM), командного слова (или командных слов при реализации команды EX), ответных слов и слов данных, передаваемых между контроллером и оконечными устройствами в порядке их следования. Структура сообщения дополняется двумя словами значения времени окончания приёма информации из интерфейса (сначала следует значение WATCH0, а затем WATCH1).

Сообщения, передаваемые по линии МКО, фиксируются монитором в памяти MIL-STD-1553 в порядке их следования, начиная с адреса 1800. После заполнения сообщениями всей зоны памяти, отведённой монитору (1800–1BFF), они продолжают фиксироваться с начального адреса зоны. Поэтому для того, чтобы не было потерь принимаемых сообщений, вычислительный модуль должен успеть считать принятые сообщения до записи в ячейки памяти, где они расположены, новых сообщений.

Первым словом в структуре сообщения является SWM. Структура разрядов SWM имеет следующий вид.

Таблица 9.38 – Структура разрядов SWM

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	ER	CER				NL	TS				NW					
–																
R																

Назначение разрядов SWM приведено в таблице 9.39.

Таблица 9.39 – Назначение разрядов SWM

Биты	Назначение
1	2
15	ER – Сбойное сообщение
14–11	CER[3–0] – Код ошибки сообщения: 0000 – Отсутствие ошибок 0001 – Отсутствие паузы после CW1 0010 – Отсутствие паузы после CW2 0011 – Число слов в сообщении меньше заданного 0100 – Число слов в сообщении больше заданного 0101 – Ошибка синхросигнала AW1 0110 – Ошибка синхросигнала AW2 0111 – Ошибка синхросигнала DW 1000 – Отсутствие AW1 1001 – Отсутствие AW2 1010 – Отсутствие паузы после приёма AW1 1011 – Ошибка кода в принимаемом слове 1100– Обнаружена ошибка паритета при приёме слова из интерфейса 1101 – Превышение допустимого времени передачи сообщения 1110 – Сообщение содержит недопустимое командное слово

Окончание таблицы 9.39

1	2
10	NL – номер линии, по которой принимается сообщение
9–6	TS[3–0] – тип сообщения: 0001 – Передача CW и DW от контроллера к оконечному устройству при исполнении команды RC с непосредственной адресацией 0010 – Выдача контроллером CW и передача AW и DW от оконечного устройства к контроллеру при исполнении команды TR 0011 – Обмен между оконечными устройствами при исполнении команды EX с непосредственной адресацией 0101 – Передача CW и DW от контроллера к оконечным устройствам при исполнении команды RC с групповой адресацией 0111 – Передача DW от оконечного устройства к оконечным устройствам с групповой адресацией при исполнении команды EX 1001 – Передача CW и одного DW от контроллера к оконечному устройству при исполнении команды управления с непосредственной адресацией 1010 – Передача контроллером оконечному устройству управляющей команды с непосредственной адресацией и выдача оконечным устройством AW и DW 1000 – Передача контроллером оконечному устройству управляющей команды с непосредственной адресацией и выдача оконечным устройством только AW 1100 – Передача контроллером оконечному устройству управляющей команды с групповой адресацией 1101 – Передача CW и одного DW от контроллера к оконечному устройству при исполнении команды управления с групповой адресацией
11–0	NW[5–0] – число слов в сообщении, включая SWM и два слова значения времени приёма сообщения

9.7 Функционирование MIL-STD-1553

Функционирование MIL-STD-1553 в режиме контроллера

Для инициализации работы MIL-STD-1553 в режиме контроллера необходимо выполнить следующие действия:

- записать в регистры CNR1 и IER необходимую информацию;
- записать в память RAM команду ввода-вывода и (если необходимо) данные;
- записать в регистр ACIO начальный адрес команды ввода-вывода в памяти RAM.

Для этого используется адрес регистра 0004 (если команда ввода-вывода не является командой EX), либо адрес регистра 0005 (если команда ввода-вывода является командой EX).

После записи в регистр ACIO начинается исполнение заданной команды. При завершении реализации команды ISR[0]=1 и, если IER[0]=1 MIL-STD-1553 формирует прерывание INTIO2 в MB2. В том случае, если в процессе исполнения команды была обнаружена ошибка, код ошибки фиксируется в регистре встроенного контроля контроллера BCW.

Функционирование MIL-STD-1553 в режиме оконечного устройства

Для инициализации работы MIL-STD-1553 в режиме оконечного устройства необходимо выполнить следующие действия:

- записать в ячейки памяти RAM, используемые при работе MIL-STD-1553 в режиме оконечного устройства, необходимую информацию;
- записать в регистры CNR0, IER, CNR2 и CNR1 необходимую информацию.

После установки признака RG_RT=1 в регистре CNR1 считается, что MIL-STD-1553 готов к работе в режиме оконечного устройства (готов к приёму команд от контроллера МКО).

Окончание исполнения команд ввода-вывода фиксируется в разрядах регистра ISR и, при условии, что соответствующие им разрешения прерываний в регистре IER установлены, MIL-STD-1553 формирует прерывание INTIO2 в MB2.

Функционирование MIL-STD-1553 в режиме монитора

Для инициализации работы MIL-STD-1553 в режиме монитора необходимо записать в регистры WATCH0, WATCH1, IER и CNR1 необходимую информацию.

После установки признака RG_M=1 в регистре CNR1 считается, что MIL-STD-1553 готов к работе в режиме монитора. При окончании записи первого сообщения в память RAM MIL-STD-1553 формирует признак ISR[11]=1 и при наличии разрешения прерывания по данному условию выдаёт в MB2 прерывание _INTIO2.

Тестирование MIL-STD-1553

Поскольку MIL-STD-1553 может одновременно работать в режимах контроллера, оконечного устройства и монитора, для тестирования всех трёх режимов необходимо установить разрешения работы в этих режимах: RG_C=1, RG_RT=1, RG_M=1, предварительно выполнив приведённые выше действия по инициализации работы MIL-STD-1553 в каждом из режимов.

Если необходимо провести тестирование MIL-STD-1553 без выдачи информации в линии МКО и приёма информации из линий МКО, необходимо установить признак CC=1.

Проверка временных параметров сигналов, выдаваемых MIL-STD-1553

Для проверки временных параметров сигналов, выдаваемых MIL-STD-1553 в линии МКО, необходимо выполнить следующие действия:

- записать в ячейку памяти RAM код слова, которое должно выдаваться в линию МКО в регистре CNR1 установить признак RG_C=1 и указать номер линии, по которой необходимо выполнять передачу информации;
- записать в регистр ACIO адрес ячейки памяти RAM, содержащей выдаваемый код. Для этого используется адрес регистра 0006 или адрес регистра 0007.

После записи в регистр ACIO начинается циклическая выдача кода в заданную линию МКО. Запись нового кода в ячейку памяти приведёт к выдаче вновь записанного кода.

9.8 Пример программирования периферийного устройства MIL-STD-1553

В следующей программе осуществляется передача одного командного слова (CW1) и одного слова данных (DW) от контроллера к оконечному устройству. Команда передаётся с групповой адресацией.

```
;////////////////////////////////////////////////Начало фрагмента программы////////////////////////////////
; CW1(f821) = RC(grc) =>L1, DW=>L1: nl=0, rg_c=1, rg_rt=1, rg_m=1, loop=1, blgrc=1, nw=1, ART=00010.
; Командное слово – CW1=RC (ART=111110→grc, r/t=0, SA=2, NW=1)=>ram AS=13'h1400
        Write 0F821h, 50h, 1400h
; AM=408=>ram AS=13'h1401
        Write 0408h, 50h, 1401h
; Слово данных DW=1234=>ram AS=13'h1408
        Write 01234h, 50h, 1408h
; CNIN=6410=>ram AS=13'h1041
        Write 06410h, 50h, 1041h
; R_CN0<=4 (blgrc=1, ART=2)
        Write 0104h, 50h, 01h
; R_CN1<=13 (loop=1, rg_c=1, rg_rt=1)
        Write 017h, 50h, 02h
; WATCH0<=8f78
        Write 08F78h, 50h, 8h
; WATCH1<=506 (ART=2)
        Write 0506h, 50h, 09h
; запись LCW=0
```

```
        Write 0h, 50h, 1000h
; запись DW=0
        Write 00h, 50h, 1410h
; запись AW1=0
        Write 00h, 50h, 1060h
; запись IER<=3f
        Write 03Fh, 50h, 0Ch
; запись R_ACIO<=400
        Write 0400h, 50h, 4h
; задержка
        rpts 4000
        nop
; чтение ISR
        Read 50h, 0Ah
; чтение BCW_C
        Read 50h, 0dh
; чтение AW1
        Read 50h, 01060h
; запись ISR<=0001(ISR[0]<=0)
        Write 1, 50h, 0Ah
; чтение CNIN
        Read 50h, 1041h
; чтение LCW
        Read 50h, 01000h
; чтение DW
        Read 50h, 01410h
; чтение SWM
        Read 50h, 01800h
; чтение CW1
        Read 50h, 01801h
; чтение DW
        Read 50h, 01802h
; чтение WATCH0
        Read 50h, 01803h
; чтение WATCH1
        Read 50h, 01804h
; запись ISR<=0800(ISR[11]<=0)
        Write 800h, 50h, 0Ah
;////////////////////////////////////////////////Конец фрагмента программы//////////////////////////////////
```

Ниже приведены макросы Read и Write.

```
;////////////////////////////////////////////////Начало фрагмента программы//////////////////////////////////
Read .macro hb, lb
        ldhi hb, AR0
        or lb, AR0
        ldi *AR0, AR6
        .endm
Write .macro inf, hb, lb
        ldi inf, AR6
        ldhi hb, AR0
        or lb, AR0
        sti AR6, *AR0
        .endm
;////////////////////////////////////////////////Конец фрагмента программы//////////////////////////////////
```

10 Описание периферийного устройства UART

Устройство UART (Universal Asynchronous Receiver/Transmitter – универсальный асинхронный приёмник/передатчик) обеспечивает последовательный обмен данными, что позволяет взаимодействовать с модемом или другим внешним устройством как, к примеру, компьютер использует протокол RS232. Это устройство максимально совместимо с индустриальным стандартом National Semiconductor's 16550A device.

В таблице 10.1 представлены скорости передачи по последовательному каналу.

Таблица 10.1 – Скорости передачи по последовательному каналу

Скорость передачи	Генерируемые частоты UART PLL при Clk PLL UART=18 432 000 Гц			Ошибка, %	Значение делителя UART, шестнадцатеричное значение
	Значение полей регистра UART PLL				
	N=4, M=1	N=13, M=3	N=9, M=2		
	73 728 000	79 872 000	82 944 000		
	Значение делителя UART, десятичное значение				
50	92160	-	-	0	0xFFFF
75	61440	-	-	0	0xF000
110	41891	-	-	0,00022	0xA3A3
134,5	34260	-	-	0,00065	0x85D4
150	30720	-	-	0	0x7800
300	15360	-	-	0	0x3C00
600	7680	-	-	0	0x1E00
1200	3840	-	-	0	0x0F00
1800	2560	-	-	0	0x0A00
2000	2304	-	-	0	0x0900
2400	1920	-	-	0	0x0780
3600	1280	-	-	0	0x0500
4800	960	-	-	0	0x03C0
7200	640	-	-	0	0x0280
9600	480	-	-	0	0x01E0
19200	240	-	-	0	0x00F0
38400	120	-	-	0	0x0078
56000	82	-	-	0,34843	0x0052
128000	36	-	-	0	0x0024
256000	18	-	-	0	0x0012
512000	9	-	-	0	0x0009
576000	8	-	-	0	0x0008
768000	6	-	-	0	0x0006
921600	5	-	-	0	0x0005
1152000	4	-	-	0	0x0004
1536000	3	-	-	0	0x0003
2304000	2	-	-	0	0x0002
4608000	1	-	-	0	0x0001
4992000		1	-	0	0x0001
5184000		-	1	0	0x0001

Входной кварцевый генератор выбирается с частотой, равной 18 432 000 Гц. Чтобы получить другие скорости приёма/передачи потребуется подбор коэффициентов N и M (в соответствии с требованиями к PLL) для обеспечения тактирования последовательного канала UART и расчёт коэффициента делителя UART. Тактовая частота последовательного канала UART (входная частота PLL) должна быть в диапазоне (65,536 – 100) МГц.

10.1 Описание регистров периферийного устройства UART

Модуль UART имеет архитектуру устройства 16550. В таблицах 10.2 – 10.4 приведено описание и адреса регистров модуля UART.

Таблица 10.2 – Регистры FIFO, модема и линии модуля UART

Адрес	Доступ	Мнемоника	Описание
0x0040 0000	R	RB	Выход FIFO приёмника
0x0040 0000	W	THR	Вход FIFO передатчика
0x0040 0001	RW	IER	Регистр разрешения (маски) прерываний
0x0040 0002	R	II	Регистр идентификации прерываний
0x0040 0002	W	FC	Регистр управления FIFO
0x0040 0003	RW	LCR	Регистр управления линией
0x0040 0004	W	MCR	Регистр управления модемом
0x0040 0005	R	LSR	Регистр состояния линии
0x0040 0006	R	MSR	Регистр состояния модема

Таблица 10.3 – Регистры делителя модуля UART

Адрес	Доступ	Имя регистра	Описание
0x0040 0000	RW	Divisor Latch Byte 1 (LSB)	Младший байт регистра делителя
0x0040 0001	RW	Divisor Latch Byte 2	Старший байт регистра делителя

Таблица 10.4 – Отладочные регистры модуля UART

Адрес	Доступ	Имя регистра	Описание
0x0040 0008	R	Debug 1	Первый отладочный регистр
0x0040 0012	R	Debug 2	Второй отладочный регистр

10.1.1 Регистр разрешения прерываний (IER)

Таблицы 10.5, 10.6 содержат формат и описание разрядов регистра IER. Этот регистр позволяет разрешать и запрещать генерацию прерываний UART.

Таблица 10.5 – Структура регистра разрешения прерываний (IER)

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Резерв													MOD	RSUR	FIFOEMP	DATRSRV
0																
R													RW			

Таблица 10.6 – Описание регистра IER

Бит	Доступ	Мнемоника	Описание
31–4	R	rsrv	Резерв
3	RW	MOD	Прерывание от модема 0 – запрещено 1 – разрешено
2	RW	RSUR	Прерывание от приёмника 0 – запрещено 1 – разрешено
1	RW	FIFOEMP	Прерывание от передатчика при пустом FIFO 0 – запрещено 1 – разрешено
0	RW	DATRSRV	Прерывания при приёме данных 0 – запрещено 1 – разрешено
Примечание – Значение после сброса: 0x0000 0000.			

10.1.2 Регистр идентификации прерываний (IIR)

Регистр IIR позволяет определить, какой текущий приоритет у прерывания. Бит 0, когда равен 0, индицирует, что прерывание обрабатывается. Когда этот бит равен 1, то нет необработанных прерываний.

В таблице 10.7 приведён список возможных прерываний с их приоритетом и источником и способом управления их сбросом.

Таблица 10.7 – Список возможных прерываний

Бит 3	Бит 2	Бит 1	Приоритет	Тип прерывания	Источник прерывания	Управление сбросом прерывания
0	1	1	1st	От приёмника	Ошибки паритета, переполнение, данных или формата или обрыв линии	Чтение регистра Line Status
0	1	0	2nd	Есть принятые данные	FIFO заполнен	Чтение FIFO ниже уровня заполнения
1	1	0	2nd	Индикация таймаута	Имеется, по крайней мере, один символ в FIFO, но он не был прочитан в течение времени приёма четырёх символов	Чтение из FIFO приёмника
0	0	1	3rd	Регистр передатчика пуст	Регистр передатчика пуст	Запись в регистр передатчика или чтение IIR
0	0	0	4th	Статус модема	CTS, DSR, RI или DCD	Чтение регистра статуса модема

Примечания
1 Биты 4 и 5 равны 0.
2 Биты 6 и 7 равны 1.
3 Биты 31 – 8 резервные.
4 Значение после сброса: 0x0000 00C1.

10.1.3 Регистр управления FIFO (FC)

Таблицы 10.8, 10.9 содержат формат и описание разрядов регистра FC. Регистр FC позволяет выбрать количество байт в FIFO, требуемых для разрешения прерывания по приёму данных. В дополнение FIFO может сбрасываться, используя этот регистр.

Таблица 10.8 – Структура регистра управления FIFO (FC)

31–8	7	6	5	4	3	2	1	0
Резерв	TRL		Резерв			FTRESET	FRRESET	Резерв
0			0					0
R	W							

Таблица 10.9 – Описание битов регистра FC

Бит	Мнемоника	Доступ	Описание
31–08	rsrv	R	Резерв
07–06	TRL	W	Определяет уровень прерывания FIFO приёмника 00 – 1 байт, 01 – 4 байт, 10 – 8 байт, 11 – 14 байт
05–03	rsrv	W	Игнорируется
02	FTRESET	W	Запись 1 в этот бит очищает FIFO передатчика, но не сбрасывает сдвиговый регистр. Передача текущего символа продолжается
01	FRRESET	W	Запись в этот бит 1 очищает FIFO приёмника, но не сбрасывает сдвиговый регистр. Приём текущего символа продолжается
00	rsrv	W	Игнорируется

10.1.4 Регистр управления линией (LCR)

Таблицы 10.10, 10.11 содержат формат и описание разрядов регистра LCR. Регистр управления линией позволяет задать формат асинхронных данных. Бит 7 регистра позволяет получить доступ к защёлкам делителя, которые определяют скорость передачи. Чтение этого регистра позволяет проверить текущее состояние соединения.

Таблица 10.10 – Структура регистра управления линией (LCR)

31–8	7	6	5	4	3	2	1	0
Резерв	ENDL	BREAK	PAR	ODD_EVEN	ENPAR	STBIT	TRL	
0								
R	RW							

Таблица 10.11 – Описание битов регистра LCR

Бит	Доступ	Описание
31–08		Резерв
07	RW	Доступ к делителю (ENDL) «1» – защёлки делителя могут адресоваться. «0» – нормальный доступ к регистрам
06	RW	Управляющий бит обрыв BREAK «0» – обрыв запрещён «1» – последовательный выход устанавливается в 0
05	RW	Добавление бита паритета (PAR) «0» – добавление бита паритета запрещено «1» – если бит 3 и 4 равны 1, то бит паритета передаётся и принимается как 0 если бит 3 = 1, бит 4 = 0, тогда бит паритета передаётся и проверяется как 1
04	RW	Чётный паритет (ODD EVEN) «0» – нечётное число 1 «1» – чётное число единиц
03	RW	Разрешение паритета (ENPAR) «0» – нет паритета «1» – бит паритета формируется и проверяется на каждом входящем символе
02	RW	Определяет число стоповых бит (STBIT) «0» – 1 стоповый бит «1» – 1,5 бита «1» – 2 стоповых бита, когда выбирается длина символа 5 бит
01–00	RW	Выбор числа битов в каждом символе (TRL1, TRL0): 00 – 5 бит, 01 – 6 бит, 10 – 7 бит, 11 – 8 бит
Примечание – Значение после сброса: 0x0000 011B.		

10.1.5 Регистр управления модемом (MCR)

Регистр MCR позволяет передавать управляющие сигналы на модем, подключенный к UART. Таблицы 10.12, 10.13 содержат формат и описание разрядов регистра MCR.

Таблица 10.12 – Структура регистра управления модемом (MCR)

31–5	4	3	2	1	0
Резерв	LOOP	OUT1	OUT2	RTS	DTR
0					
R	W				

Таблица 10.13 – Биты регистра MCR

Бит	Доступ	Описание
31–05		Резерв
04	W	Режим «петли» (LOOP) 0 – нормальный режим 1 – режим «петли». Сигнал Serial Output Signal (STX_PAD_O) устанавливается в 1. Сигнал сдвигового регистра передатчика внутри подсоединяется к входу регистра сдвига приёмника. Следующие **соединения** выполняются: DTR → DSR RTS → CTS Out1 → RI Out2 → DCD
3	W	Выход 2 (OUT2). В режиме «петля» подсоединяется к входу Data Carrier Detect (DCD)
2	W	Выход 1 (OUT1). В режиме «петля» подсоединяется к входному сигналу Ring Indicator (RI)
1	W	Сигнал запроса на передачу данных (RTS) 0 – RTS есть 1 1 – RTS есть 0
0	W	Сигнал готовности терминала данных (DTR) 0 – DTR есть1 1 – DTR есть 0
Примечание – Значение после сброса: 0x0000 0000.		

10.1.6 Регистр состояния линии (LSR)

Регистр LSR текущее состояние линии приёма/передачи. В таблице 10.14 приведено описание разрядов регистра LSR.

Таблица 10.14 – Биты регистра LSR

Бит	Доступ	Описание
1	2	3
31–8		Резерв
07	R	Ошибка (ERROR) 0 – ни один символ в FIFO не имеет ошибки 1 – По крайней мере, один принятый символ имеет ошибку либо паритета, либо ошибку кадра, либо обрыв линии Бит сбрасывается чтением из этого регистра
06	R	Передатчик пуст (TE) 1 – FIFO и сдвиговый регистр передатчика пусты Сбрасывается, когда данные записываются в FIFO 0 – либо FIFO, либо сдвиговый регистр имеют символ
05	R	FIFO передатчика пуст (FTE) 0 – FIFO не пуст 1– FIFO передатчика пуст, генерируется прерывание «Регистр передатчика пуст». Бит сбрасывается записью символа в FIFO
04	R	Индикатор обрыва (BI) 0 – нет обрыва в текущем символе 1 – обрыв произошёл на текущем символе. Сигнал «Обрыв» вырабатывается, когда линия держится в 0 во время передачи одного символа (стартовый бит + данные + паритет + стоповый бит (start bit + data + parity + stop bit)). В этом случае один нулевой символ заносится в FIFO, и UART ждёт стартовый бит для приёма следующего символа. Бит сбрасывается чтением из этого регистра. Генерируется прерывание Receiver Line Status

Окончание таблицы 10.14

1	2	3
03	R	Индикатор ошибки кадра (FE) 0 – нет ошибки кадра в текущем символе 1 – символ, который находится в вершине стека, принят с ошибкой кадра (не правильный стоповый бит). Бит сбрасывается чтением из регистра. Генерирует прерывание Receiver Line Status.
02	R	Индикатор ошибки паритета. (PE) 0 – нет ошибки паритета в текущем символе 1 – символ, который находится в вершине стека, принят с ошибкой паритета. Бит сбрасывается чтением из регистра. Генерирует прерывание Receiver Line Status
01	R	Индикатор ошибки переполнения (OE) 0 – нет состояния переполнения 1 – если FIFO полон и следующий символ принят в сдвиговый регистр. Если следующий символ принимается в сдвиговый регистр, то это будет вызывать ошибку переполнения, но FIFO будет оставаться не повреждённым. Этот бит сбрасывается чтением из регистра. Генерирует прерывание Receiver Line Status
00	R	Индикатор готовности данных (DR). 0 – нет символа в FIFO 1 – по крайней мере, один символ принят и находится в FIFO.

10.1.7 Регистр состояния модема (MSR)

Регистр отображает текущее состояние управляющих линий модема, см. таблицу 10.15. Четыре бита индицируют состояние статусных линий модема. Эти биты устанавливаются в 1, когда обнаруживается изменение в соответствующей линии и сбрасываются, когда читается регистр. В таблице 10.16 описаны биты регистра MSR.

Таблица 10.15 – Структура регистра статуса модема (MSR)

31–8	7	6	5	4	3	2	1	0
Резерв	ADVDCD	ADVRI	ADVDSR	ADVCTS	DDCD	TERI	DDSR	DCTS
0								
R	RC							

Таблица 10.16 – Биты регистра MSR

Бит	Доступ	Описание
31–8		Резерв
07	RC	Дополнение DCD или равный Out2 в режиме «петли» (ADVDCD)
06	RC	Дополнение RI или равный Out1 в режиме «петли» (ADVRI)
05	RC	Дополнение входа DSR или равный DTR в режиме «петли» (ADVDSR)
04	RC	Дополнение входа CTS или равный RTS в режиме «петли» (ADVCTS)
03	RC	Индикатор несущей Delta Data Carrier Detect (DDCD) 1 – линия DCD изменила своё состояние
02	RC	Детектор Trailing Edge of Ring Indicator (TERI) detector. Линия RI изменила своё состояние из низкого в высокое
01	RC	Индикатор Delta Data Set Ready (DDSR) 1 – линия DSR изменила своё состояние
00	RC	Индикатор Delta Clear To Send (DCTS) 1 – линия CTS изменяет своё состояние

10.1.8 Регистр делителя DL

Доступ к защёлкам делителя разрешается установкой в «1» бита 7 регистра LCR. Необходимо восстановить значение этого бита в «0» после установки защёлок делителя для разрешения доступа к другим регистрам с теми же адресами. Для нормальной операции должны быть установлены оба байта.

Регистр по умолчанию установлен в 0, который запрещает все операции последовательного ввода/вывода для того, чтобы явно установить регистры программой. Устанавливаемое значение должно быть эквивалентно «скорость системного такта»/16 × «желаемая скорость».

Внутренний счётчик стартует, когда младший байт делителя установлен. (Первым записывается старший байт и последним младший байт).

10.1.9 Регистр отладки 1

Этот регистр используется для отладочных целей и не является частью спецификации UART 16550. В таблице 10.17 описаны биты регистра отладки 1.

Таблица 10.17 – Биты регистра отладки 1

Бит	Доступ	Описание
31–24	R	Значение регистра состояния модема (MSR)
23–16	R	Значение регистра управления линией (LCR)
15–12	R	Значение битов 3–0 регистра идентификации прерывания (II)
11–8	R	Значение битов 3–0 регистра разрешения прерываний (IER)
7–0	R	Значение регистра состояния линии (LSR)

10.1.10 Регистр отладки 2

Этот регистр используется для отладочных целей и не является частью спецификации UART 16550. В таблице 10.18 описаны биты регистра отладки 2.

Таблица 10.18 – Биты регистра отладки 2

Бит	Доступ	Описание
31–24	R	Резерв
23–19	R	Значение регистра управления FIFO (FC)
18–17	R	Значение битов 4–0 регистра управления модемом (MCR)
16–12	R	Число символов в FIFO приёмника (RFCOUNT)
11–8	R	Состояние FSM приёмника (RSTATE)
7–3	R	Количество символов в FIFO передатчика (TFCOUNT)
2–0	R	Состояние FSM передатчика (TSTATE)

10.2 Пример программирования периферийного устройства UART

После RESET ядро UART выполняет следующие действия:

- FIFO передатчика и приёмника очищаются;
- сдвиговые регистры передатчик и приёмника также очищаются;
- регистр делителей устанавливается в ноль;
- прерывания все запрещаются.

Для приёма и/или передачи символов необходимо выполнить следующие действия:

- установить желаемые параметры регистра Line Control, установить бит 7 в единицу, чтобы получить доступ к делителям;
- записать необходимые значения в делитель, сначала необходимо записать значение в старший байт, затем в младший байт;
- установить значение седьмого бита регистра Line Control в ноль для запрета возможности обращения к делителям. После этого данные могут отправляться и приниматься.

Ниже приведён код, устанавливающий в делителях значение, необходимое для частоты 38400 бит/с (устанавливаемое значение есть («тактовая частота»)/(16 × «желаемая скорость»):

```
        ; установка бита 7 LCR в 1, для получения доступа к делителю
                ldi @Line_Control, AR0; считываем Line Control Register
                or 80h, AR0; бит 7 устанавливаем в 1
                sti AR0, @Line_Control; теперь делители доступны
                ldi 0, AR0
                sti AR0, @Divisor_Latch_Byte2
                ldi 78h, AR0
                sti AR0, @Divisor_Latch_Byte1
        ; установка бита 7 LCR в 0
                ldi @Line_Control, AR0
                and 0FF7Fh, AR0
                sti AR0, @Line_Control
```

После установки делителей устанавливают необходимый уровень опустошения FIFO и желаемые прерывания. Ниже приведён код, разрешающий внешние прерывания процессора, прерывание UART. Принятые данные готовы и установлен уровень FIFO равным 8 байт.

```
                or 2000h, ST; разрешаем глобальные прерывания ПЦОС
                or 9999h, IIF; разрешаем внешние прерывания
                ldi 086h, AR0; trigger level – 8 байт и очистка FIFO
                ldi @Interrupt_Enable, AR0
                or 1, AR0
                sti AR0, @Interrupt_Enable; разрешаем прерывание
```

Отправка символа осуществляется записью передатчика:

```
                ldi 0AAh, AR0
                sti AR0, @Transmitter_Holding
```

Приём символа осуществляется чтением приёмника:

```
                ldi @Receiver_Buffer, AR0
```

В следующем фрагменте программы осуществляется отправка и приём одного символа в режиме внутренней петли.

```
                ldi 0FFh, AR0
        ; очистка FIFO передатчика и приёмника
                sti AR0, @FIFO_Conrol
        ; активация режима внутренней петли
                ldi @Modem_Control, AR0
                or 10h, AR0
                sti AR0, @Modem_Control
        ; установка режима (8 – 1 - NoParity)
                ldi 3, AR0;
                sti AR0, @Line_Control
        ; отправка символа AA
                ldi 0AAh, AR0
                sti AR0, @Transmitter_Holding
Wait:           nop
                ldi @Line_Status, AR2
        ; проверка бита 0 (Data Ready)
                        tstb 1, AR2
                bz Wait; если 0, то продолжаем ждать
        ; приём
                ldi @Receiver_Buffer, AR1
```

11 Описание периферийного устройства UART PLL

Периферийное устройство UART PLL формирует тактовый сигнал с программируемой частотой для получения требуемого значения скорости приёма/передачи периферийного устройства UART. Управление частотой UART PLL осуществляется путём программирования регистра RUARTPLL. В таблице 10.1 приведён пример значений полей NS и MS для различных скоростей приёма/передачи данных.

После сброса UART PLL установлен в начальное состояние, при котором выходная частота UART PLL равна частоте, подаваемой на вход Clk PLL UART 1867ВЦ8Ф1.

После программирования UART PLL его выходная частота равняется частоте, подаваемой на вход Clk PLL UART. Спустя 500 мкс (время установления выходной частоты) выходная частота UART PLL становится равной частоте, которая была запрограммирована.

Примечание – Значение времени установления выходной частоты приведено для Clk PLL UART = 18 432 кГц.

Выходная частота UART PLL определяется по формуле

$$F_{UART\ PLL} = NS[5–0]/MS[5–0] \quad (11.1)$$

Примечание – Выходная частота $F_{UART\ PLL}$ должна быть в диапазоне от 5 до 100 МГц, а выходная – в диапазоне от 20 до 100 МГц.

11.1 Описание регистра RUARTPLL

Далее приводится формат регистра RUARTPLL, описание и функциональное назначение его полей и отдельных разрядов. В таблицах 11.1, 11.2 и 11.3 приведены адрес, структура и функциональное назначение битов регистра RUARTPLL, соответственно.

Таблица 11.1 – Адрес регистра RUARTPLL

Наименование	Адрес
RUARTPLL	004001FFh

Таблица 11.2 – Структура регистра RUARTPLL

31–14	13	12	11–6	5–0
rsrv	EnPLL	FRANGE	NS	MS
0	0	0	4	1
R	RWC	RWC	RWC	RWC
Примечание – R – возможно чтение этого бита, W – возможна запись в этот бит, C – бит сбрасывается, 0 – значение после сброса.				

Таблица 11.3 – Описание битов регистра RUARTPLL

Бит	Имя	Доступ	Описание
31–14	rsrv	R	Резервные биты. Читаются как 0
13	EnPLL	RWC	Разрешение UART PLL. Если EnPLL = 0, то $F_{UART\ PLL}$ = Clk PLL UART Если EnPLL = 1, то $F_{UART\ PLL}$ = NS[5–0]/MS[5–0] спустя 500 мкс программирования.
12	FRANGE	RWC	Выбор диапазона выходной частоты UART PLL. Если FRANGE = 0, то $F_{UART\ PLL}$ = (20–100) МГц. Если FRANGE = 1, то $F_{UART\ PLL}$ = (100–300) МГц. После сброса FRANGE = 0
11–6	NS	RWC	Поле множителя. После сброса NS = 4.
5–0	MS	RWC	Поле делителя. Поле делителя не может быть равно 0. При попытке записи в поле MS значения равного 0 в это поле запишется значение 1. После сброса MS = 1.
Примечание – R – возможно чтение этого бита, W – возможна запись в этот бит, C – бит сбрасывается, 0 – значение после сброса.			

11.2 Программирование периферийного устройства UART PLL

После записи в регистр PLL_UART нужного значения выходная частота переключается на опорную частоту и начинает отсчитываться счётчик ожидания установки стабильной частоты PLL. Время ожидания следует выбирать не менее 500 мкс. После завершения работы счётчика выходная частота PLL будет равна выбранной частоте.

Пример программирования регистра PLL_UART для установки частоты 73 728 Гц при начальной частоте 18 432 Гц (поле множителя равно четырём, поле делителя равно единице):

```
;//////////////////////////////////Начало фрагмента программы//////////////////////////////////////////////

        ldhi 40h, AR0
        or 1FFh, AR0; теперь AR0 = 4001FF – адрес UART PLL
        ldi 2101h, AR4; 18432*4/1 = 73728
        sti AR4, *AR0

Организация задержки более 500 микросекунд:
        ldp 100000h;
; timer reset
ldi 0, AR0
sti AR0, @20h
; программирование счётчика таймера
sti AR0, @24h
        ldhi 0FFFh, AR0
        or 0FFFFh, AR0
        sti AR0, @28h; установка периода 0FFF FFFFh
; config timer control
ldi 3c1h, AR0
sti AR0, @20h
; 500mks 500 000 / 20 = 61A8h
ldi 61A8h, AR7;
WTimer:
        Nop
ldi @24h, AR1; считываем счётчик
cmpi AR1, AR7
bgt WTimer; Если AR1 меньше AR7, то ждём

;//////////////////////////////////Конец фрагмента программы//////////////////////////////////////////////
```

12 Рекомендации по подключению питания и входных сигналов ИС 1867ВЦ8Ф1

ИС 1867ВЦ8Ф1 имеет несколько групп выводов питания:

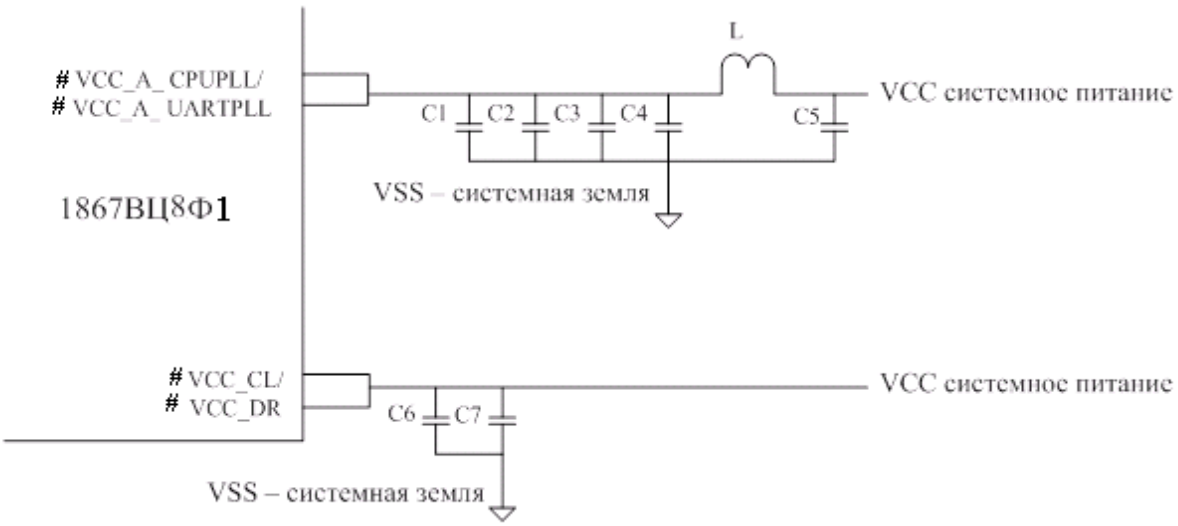
- #VCC_CL – выводы питания ядра ИС U_{VCC_CL} = 1,8 В;
- ∩VSS_CL – выводы земли ядра ИС;
- #VCC_DR – выводы питания буферов ввода-вывода U_{VCC_DR} = 3,3 В;
- ∩VSS_DR – земляные выводы буферов ввода-вывода;
- #VCC_A_ CPUPLL – аналоговый вывод питания PLL процессорных ядер;
- ∩VSS_A_CPUPLL – аналоговый вывод земли PLL процессорных ядер;
- #VCC_A_ UARTPLL – аналоговый вывод питания PLL UART;
- ∩VSS_A_UARTPLL – аналоговый вывод земли PLL UART;

Выводы #VCC_CL, ∩VSS_CL, #VCC_DR и ∩VSS_DR питают цифровую часть схемы ИС. Выводы #VCC_A_CPUPLL, ∩VSS_A_CPUPLL, #VCC_A_ UARTPLL и ∩VSS_A_UARTPLL питают аналоговую часть ИС (блоки PLL).

Порядок подачи и снятия напряжения питания и входных сигналов на микросхему должен быть следующим: сначала подаются напряжения питания, а затем (или одновременно) входные напряжения высокого и низкого уровней. Порядок снятия напряжений с выводов микросхемы при выключении должен быть обратным: первыми снимаются входные сигналы, а затем (или одновременно) напряжения питания. Для выводов ввода/вывода, которые могут использоваться как в режиме ввода, так и в режиме вывода, следует использовать отдельные резисторы подтяжки, если эти выводы не используются.

Поскольку блоки PLL (CPU PLL и UART PLL) чувствительны к шумам по аналоговому питанию, приводящим к фазовому джиттеру, в микросхеме предусмотрены отдельные выводы аналогового питания PLL для минимизации этих шумов.

Чтобы уменьшить высокочастотные и низкочастотные шумы по питанию необходимо использовать развязывающие конденсаторы подходящего номинала. При незначительных шумах по питанию цифровой части платы аналоговое питание PLL можно осуществлять непосредственно от системного питания, как это показано на рисунке 12.1.

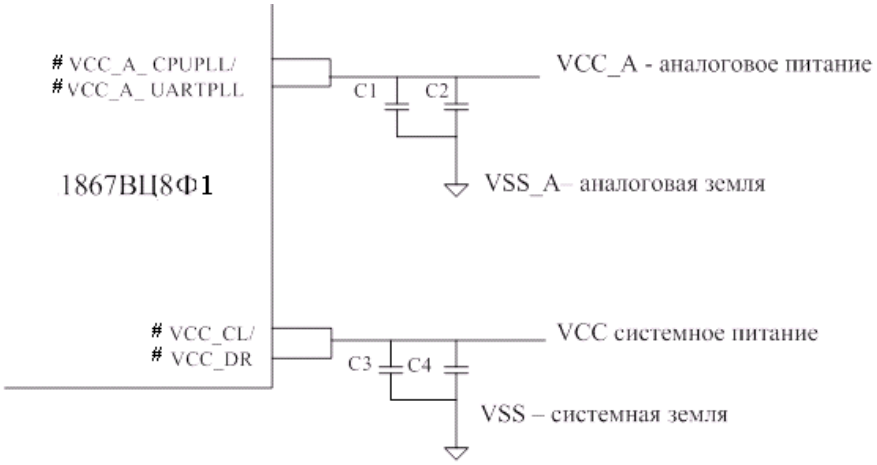


Примечание – C1 = 0,001 мкФ, C2 = 0,01 мкФ, C3 = 0,1 мкФ, C4 = 1 мкФ, C5 = 10 мкФ, C6 = 0,1 мкФ, C7 = 10 мкФ, L = 10 мкГн.

Рисунок 12.1 – Схема подключения питания ИС 1867ВЦ8Ф1 на плате при небольших шумах по питанию логической части схемы

При значительных шумах по питанию цифровой части платы питание PLL нужно осуществлять от отдельного источника питания, и устанавливать развязывающие конденсаторы, как это показано на рисунке 12.2.

Развязывающие конденсаторы с номиналом 0,1 мкФ должны быть расположены как можно ближе к выводам питания ИС. Все шины питания должны иметь минимальную длину.



Примечание – C1 = 0, 1 мкФ, C2 = 10 мкФ, C3 = 10 мкФ.

Рисунок 12.2 – Схема подключения питания ИС 1867ВЦ8Ф1 на плате при значительных шумах по питанию логической части схемы

Приложение А
(обязательное)

Инструкции языка Ассемблер

А.1 Общие положения

Реализованная в процессорных ядрах ИС 1867ВЦ8Ф1 система команд (набор инструкций языка Ассемблер) поддерживает высокопроизводительные вычислительные функции и обеспечивает как решение задач собственно цифровой обработки сигналов, так и задач общего назначения. Набор инструкций, в зависимости от их назначения, можно разделить на несколько основных групп, включающих инструкции загрузки и сохранения, двух- и трёхоперандные арифметико-логические, параллельные инструкции, а также инструкции программного управления и блокировки для реализации межпроцессорного (межъядерного) взаимодействия.

Инструкции языка Ассемблер ИС 1867ВЦ8Ф1 могут также использовать один из 20 кодов условий с одной из 10 условных инструкций, таких как LDFcond. Коды условий и флаги подробно описаны в разделе А.10.

Ассемблер позволяет использовать дополнительные синтаксические формы для упрощённой записи некоторых инструкций. Описание этих дополнительных форм приведено в разделе А.11.2.

Детальное описание всех имеющихся инструкций приведено в их алфавитном порядке. В Примере (см. раздел А.11.4) подробно рассматривается состав и формат индивидуального описания каждой инструкции (синтаксис, операнды, операционный код, количество циклов выполнения и т.д.).

А.2 Функциональные группы инструкций языка Ассемблер

Набор инструкций процессорного ядра ИС 1867ВЦ8Ф1 оптимизирован для решения задач цифровой обработки сигналов и приложений, требующих высокопроизводительных вычислений. Все инструкции имеют длину в одно машинное слово, и большинство из них выполняется за один цикл. Помимо инструкций умножения и накопления, процессорное ядро ИС 1867ВЦ8Ф1 поддерживает полный набор инструкций общего назначения.

Набор из 147 инструкций можно разделить на следующие функциональные группы:

- 25 инструкций загрузки и сохранения;
- 43 двухоперандных арифметико-логических инструкций;
- 19 трёхоперандных арифметико-логических инструкций;
- 25 инструкций управления выполнением программы;
- 5 инструкций управления блокировкой;
- 30 инструкций параллельных операций.

В последующих разделах описывается состав и назначение этих групп инструкций.

А.3 Инструкции загрузки и сохранения

Ядро процессора ИС 1867ВЦ8Ф1 поддерживает 25 инструкций загрузки и сохранения (см. таблицу А.1). Эти инструкции позволяют:

- загружать слово из памяти в регистр;
- сохранять слово из регистра в память;
- управлять данными в системном стеке;
- передавать данные между основным и расширенным регистровыми файлами.

Две из этих инструкций могут загружать данные по условию. Это используется для нахождения минимального или максимального значения из набора данных.

Таблица А.1 – Инструкции загрузки и сохранения

Инструкция	Описание	Инструкция	Описание
1	2	3	4
LBb	Загрузка байта (со знаком)	LDE	Загрузка экспоненты с ПЗ (плавающей запятой)
LBUb	Загрузка байта (без знака)	LDEP	Загрузка целого, расширенный файловый регистр в основной регистр
LDA	Загрузка адресного регистра	LDF	Загрузка значения с ПЗ

Окончание таблицы А.1

1	2	3	4
LDFcond	Загрузка значения с ПЗ по условию	LHw	Загрузка полуслова со знаком
LDHI	Непосредственная загрузка 16 разрядов без знака в 16 старших разрядов	LHUw	Загрузка беззнакового полуслова
		LWLct	Загрузка слова с левым сдвигом
LDI	Загрузка целого	LWRct	Загрузка слова с правым сдвигом
LDIcond	Загрузка целого по условию	POP	Выталкивание целого из стека
LDM	Загрузка мантиссы с ПЗ	POPF	Выталкивание значения с ПЗ из стека
LDP	Загрузка указателя страницы памяти данных (регистр DP)	PUSH	Вталкивание целого в стек
		PUSHF	Вталкивание в стек значения с ПЗ
LDPE	Загрузка целого, основной регистр в расширенный файловый регистр	STF	Сохранение значения с ПЗ
		STI	Сохранение целого
LDPK	Непосредственная загрузка указателя страницы памяти данных (регистр DP)	STIK	Непосредственное сохранение целого

А.4 Инструкции с двумя операндами

ИС 1867ВЦ8Ф1 поддерживает полную систему из 43 двухоперандных арифметических и логических инструкций. Два операнда являются исходным операндом и операндом результата. Исходным операндом может быть слово памяти, регистр или константа. Операнд результата – всегда регистр.

Эти инструкции обеспечивают целочисленную арифметику, арифметику с ПЗ или логические операции и арифметику повышенной точности. В таблице А.2 приведены эти инструкции.

Таблица А.2 – Двухоперандные инструкции

Инструкция	Описание	Инструкция	Описание
1	2	3	4
ABSF	Абсолютное значение числа с ПЗ	MPYF*	Умножить значения с ПЗ
ABSI	Абсолютное значение целого	MPYI*	Умножить целые
ADDC*	Сложить целое с переносом	MPYSHI*	Умножение целого со знаком, результат – 32 старших разряда
ADDF*	Сложить значение с ПЗ	MPYUHI*	Умножение беззнакового целого, результат – 32 старших разряда
ADDI*	Сложить целые	NEGB	Отрицание целого с заёмом
AND*	Поразрядное логическое И	NEGF	Отрицание числа в формате с ПЗ
ANDN*	Поразрядное логическое И с дополнением	NEGI	Отрицание целого
ASH*	Арифметический сдвиг	NORM	Нормализовать значение с ПЗ
CMPF*	Сравнить значения с ПЗ	NOT	Поразрядное логическое дополнение
CMPI*	Сравнить целые	OR*	Поразрядное логическое ИЛИ
FIX	Преобразовать число с ПЗ в целое	RCPF*	Обратная величина числа с ПЗ
FLOAT	Преобразовать целое в число с ПЗ	RND	Округлить значение с ПЗ
FRIEEE	Преобразует IEEE формат с плавающей запятой в формат с плавающей запятой в дополнительном коде	ROL	Циклический сдвиг влево
LSH*	Логический сдвиг	ROLC	Левый циклический сдвиг с переносом
MBct	Слияние байта, левое смещение	ROR	Циклический сдвиг вправо
MHct	Слияние полуслова, левое смещение	RORC	Циклический сдвиг вправо с переносом

Окончание таблицы А.2

1	2	3	4
RSQRF*	Обратная величина квадратного корня	SUBRF	Вычесть обратное число с ПЗ с заёмом
SUBB*	Вычитание целых с заёмом	SUBRI	Вычесть обратное целое
SUBC	Вычитание целых с условием	TOIEEE	Преобразует формат с плавающей запятой в дополнительном коде в IEEE формат
SUBF	Вычитание значений с ПЗ	TSTB*	Тестировать разрядные поля
SUBI	Вычесть целое	XOR*	Поразрядное исключающее ИЛИ
SUBRB	Вычесть обратное целое с заёмом		

_____

* Двухоперандные и трёхоперандные версии.

А.5 Инструкции с тремя операндами

Большинство инструкций имеет только два операнда, однако, некоторые арифметические и логические инструкции имеют трёхоперандные версии. 19 трёхоперандных инструкций позволяют ядру процессора ИС 1867ВЦ8Ф1 считывать два операнда из памяти или регистрового файла ЦПУ в один цикл и сохранять результаты в регистре. Ниже описываются различия между двух- и трёхоперандными инструкциями:

- Двухоперандные инструкции имеют один исходный операнд (или сдвиг счётчика) и один операнд результата.

- Трёхоперандные инструкции могут иметь два исходных операнда (или один исходный операнд и операнд счётчика) и один операнд результата. Исходным операндом является слово памяти, регистр или константа. Операнд результата в трёхоперандных инструкциях всегда регистр.

В таблице А.3 приведён список трёхоперандных инструкций. Необходимо обратить внимание, что символ «3» может быть опущен в мнемонике трёхоперандной инструкции.

Таблица А.3 – Трёхоперандные инструкции

Инструкция	Описание	Инструкция	Описание
ADDC3	Сложение с переносом	MPYI3	Умножить целые
ADDF3	Сложить значения с ПЗ	MPYSHI3	Умножение целых со знаком, результат – 32 старших разряда
ADDI3	Сложить целые	MPYUHI3	Умножение беззнаковых целых, результат – 32 старших разряда
AND3	Поразрядное логическое И	OR3	Поразрядное логическое ИЛИ
ANDN3	Поразрядное логическое И с дополнением	SUBB3	Вычитание целых с заёмом
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.6 Инструкции программного управления

Группа инструкций программного управления состоит из 25 инструкций, определяющих последовательность выполнения программы. Режим повторения обеспечивает повтор 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. 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
CALL cond	Вызов подпрограммы по условию	RPTB	Повтор блока инструкций
DBcond	Декремент и переход по условию (стандартный)	RPTB	Повтор блока инструкций, задержанный
DBcondD	Декремент и переход по условию (задержанный)	RPTS	Повтор отдельной инструкции
IACK	Подтверждение прерывания	SWI	Программное прерывание
IDLE	Холостая работа до прерывания	TRAPcond	Условное программное прерывание
IDLE2	Холостая работа до прерывания с выключением синхронизации внутренних устройств		

А.7 Инструкции операций блокировки

Инструкции операций блокировки поддерживают мультипроцессорные связи и используют внешние сигналы для обеспечения мощного механизма синхронизации. Они также гарантируют целостность связи и результатов в высокоскоростных операциях.

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.8 Инструкции параллельных операций

Группа параллельных инструкций делает возможным высокую степень параллелизма. Некоторые инструкции ядра процессора 1867ВЦ8Ф1 могут объединяться парами, которые выполняются параллельно. Данные инструкции обеспечивают следующие возможности:

- параллельную загрузку регистров;
- параллельное сохранение;
- параллельные арифметические операции;
- арифметико-логические инструкции, выполняемые параллельно с инструкциями сохранения.

Каждая инструкция в паре вводится как отдельный исходный оператор. Вторая инструкция в паре должна быть отделена двумя вертикальными чёрточками (||). В таблице А.6 приведён список пар инструкций.

Таблица А.6 – Параллельные инструкции

Мнемоника	Описание
1	2
(а) Инструкции параллельного выполнения арифметических операций и сохранения	
ABSF||STF	Абсолютное значение числа в формате с ПЗ и сохранить значение с ПЗ
ABSI||STI	Абсолютное значение целого числа и сохранить целое
ADDF3||STF	Сложить значения в формате с ПЗ и сохранить значение с ПЗ
ADDI3||STI	Сложить целые и сохранить целое
AND3||STI	Поразрядное логическое И и сохранить целое
ASH3||STI	Арифметический сдвиг и целое
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
LSH3||STI	Логический сдвиг и сохранить целое
MPYF3||STF	Умножить значения в формате с ПЗ и сохранить значение с ПЗ
MPYI3||STI	Умножить целое и сохранить целое
NEGF||STF	Обратное значение в формате с ПЗ и сохранить значение с ПЗ
NEGI||STI	Обратное целое и сохранить целое
NOT||STI	Логическое дополнение (поразрядная инверсия) значения и сохранить целое
OR3||STI	Поразрядное логическое ИЛИ и сохранить целое

Окончание таблицы А.6

1	2
STF||STF	Сохранить значения в формате с ПЗ
STI||STI	Сохранить целые
SUBF3||STF	Вычесть значение в формате с ПЗ и сохранить значение в формате с ПЗ
SUBI3||STI	Вычесть целое и сохранить целое
TOIEEE||STF	Преобразовать в IEEE формат и сохранить
XOR3||STI	Поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ значений и сохранить целое
(b) Инструкции параллельной загрузки	
LDF||LDF	Загрузить значение в формате с ПЗ
LDI||LDI	Загрузить целое
(c) 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
MPYI3||SUBI3	Умножить и вычесть целое

А.9 Недопустимые инструкции

В процессорном ядре 1867ВЦ8Ф1 не предусмотрен механизм распознавания недопустимых кодов инструкций. Результатом выборки такого кода может стать выполнение неопределённой операции. К генерации недопустимых кодов инструкций может привести некорректное использование программного обеспечения, ошибка прошивки ПЗУ или дефекты ОЗУ.

А.10 Коды условий и флаги

Ядро процессора 1867ВЦ8Ф1 использует 20 кодов условий (с 00000 по 10100, за исключением 01011), которые могут быть использованы с условными инструкциями, такими как RETScond или LDFcond. Условиями являются знаковые и беззнаковые сравнения, сравнение с нулём 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 свойствах результата арифметических или логических инструкций. Флаги условий хранятся в регистре состояния (ST); влияние инструкции на флаг условия зависит от значения поля SET COND (разряд 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, ST флаг состояния установлен, если операционный объект – это один из регистров с повышенной точностью (R0 – R11).

Если SET COND = 1, ST флаг состояния также установлен, если операционный объект – это один из основных регистровых файлов, за исключением регистра состояния.

Флаги условий (LUF, LV, UF, N, Z, V, C) могут изменяться многими инструкциями либо когда предшествующие условия установлены, либо при выполнении следующих условий:

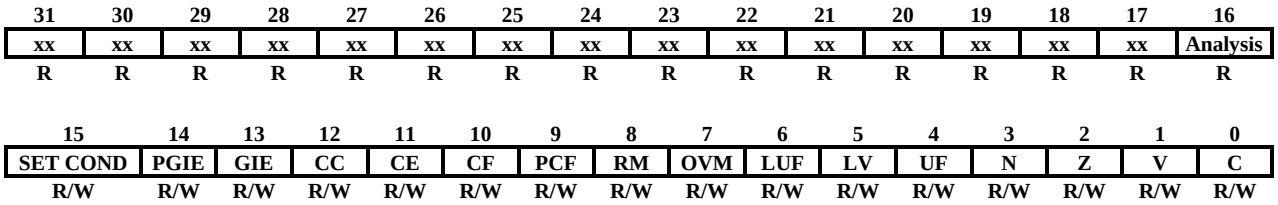
- Результат получен, когда определённая операция выполнена с бесконечной точностью. Это подходит для случая инструкций сравнения и тестирования, которые не сохраняют результат в регистре. Это также может подходить для арифметических инструкций, которые приводят к переполнению или отрицательному переполнению (потере значимости).

- Выход записан в регистр результата, как показано в таблице А.7. Это 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. Ниже рисунка следует список флагов условий регистра состояния и описание того, как данные флаги устанавливаются различными инструкциями.



Примечание – xx – резервные разряды, R – чтение, W – запись.

Рисунок А.1 – Регистр состояния

LUF – фиксируемый флаг условия отрицательного переполнения при работе с ПЗ. Устанавливается всякий раз при установке UF (флаг отрицательного переполнения при работе в формате с ПЗ). LUF может быть очищен только при сбросе или при изменении регистра состояния ST.

LV – фиксируемый флаг условия переполнения. Устанавливается всякий раз при установке V (флаг условия переполнения, LV может быть очищен только при сбросе или при изменении регистра состояния ST).

UF – флаг условия 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 UF устанавливается, и выходное значение устанавливается в 0. Если не происходит отрицательного переполнения, UF очищается.

N – флаг отрицательного условия. Логические операции присваивают N значение старшего значащего разряда выходного значения. Для целочисленных операций и операций с ПЗ N устанавливается, если результат отрицательный, и очищается – в противном случае. Ноль означает положительное значение.

Z – флаг нулевого условия. Для логических, целочисленных и ПЗ операций Z устанавливается при равенстве выходного результата 0 и очищается в противном случае.

V – флаг условия переполнения. Для целочисленных операций V устанавливается, если результат не вписывается в формат, определённый для назначения (т.е. $-2^{32} \le$ результат $\le 2^{32} - 1$). В противном случае V очищается. Для операций в формате с ПЗ V устанавливается, если экспонента результата больше 127; в противном случае V очищается. Логические операции всегда очищают V.

C – при выполнении целочисленного сложения C очищается в случае возникновения переноса относительно старшего значащего разряда выходного значения. При выполнении целочисленного вычитания C устанавливается при возникновении заёма в разряде, относящемся к старшему значащему разряду выходного значения. В противном случае при целочисленных операциях 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 список мнемоник, кодов, описаний и флагов для каждого из 19 кодов условий.

Таблица А.8 – Коды условий и флаги

Условие	Код	Описание	Флаг*
(a) Безусловные сравнения			
U	00000	Безусловный	Безусловное
(b) 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
(c) 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
(d) Сравнение с нулём			
Z	00101	Ноль	Z
NZ	00110	Не ноль	~Z
P	01001	Положительное	~N И ~Z
N	00111	Отрицательное	N
NN	01010	Не отрицательное	~N
(e) 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 – Знак ~ означает инверсию.

_____

* Означает логическое дополнение (состояние «ложь»).

A.11 Описание инструкций языка Ассемблер

Этот раздел содержит описание инструкций языка Ассемблер для процессорного ядра ИС 1867ВЦ8Ф1. Инструкции следуют в алфавитном порядке. Информация о каждой из них включает синтаксис, выполняемую операцию, операнды, код, описание, количество циклов, разряды состояния, режимные разряды и примеры. Определение символов и аббревиатур также, как и дополнительных синтаксических форм, поддерживаемых языком Ассемблер, находится в начале раздела описания инструкций. Также приводится пример инструкции с описанием используемого специального формата и объясняется её контекст.

A.11.1 Символы и аббревиатуры

В таблице А.9 приведены символы и аббревиатуры, используемые при описании инструкций.

Таблица А.9 – Символы инструкций

Символ	Назначение
src	Исходный операнд
src1	Исходный операнд 1
src2	Исходный операнд 2
src3	Исходный операнд 3
src4	Исходный операнд 4
dst	Операнд назначения
dst1	Операнд назначения 1
dst2	Операнд назначения 2
disp	Смещение
cond	Условие
count	Счётчик сдвига
G	Основные режимы адресации
T	Трёхоперандные режимы адресации
P	Параллельные режимы адресации
B	Режимы адресации с условным переходом
ARn	Вспомогательный регистр n
IRn	Индексный регистр n
Rn	Регистр повышенной точности n
RC	Регистр счётчика повторений
RE	Регистр конечного адреса повторений
RS	Регистр начального адреса повторений
ST	Регистр состояния
C	Разряд переноса
GIE	Разряд разрешения глобальных прерываний
N	Вектор программного прерывания (trap)
PC	Счётчик инструкций
RM	Флаг режима повторения
SP	Указатель системного стека
|x|	Абсолютное значение x
x → y	Присваивает значение x значению y
x(man)	Поле мантиссы (знак + дробная часть) от x
x(exp)	Поле экспоненты
op1||op2	Операция 1 выполняется параллельно операции 2
x AND y	Поразрядное логическое И от x и y
x OR y	Поразрядное логическое ИЛИ от x и y
x XOR y	Поразрядное логическое ИСКЛЮЧАЮЩЕЕ ИЛИ от x и y
~x	Поразрядное логическое дополнение (инверсия) от x
x << y	Сдвиг x влево на y разрядов
x >> y	Сдвиг x вправо на y разрядов
*+ +SP	Инкрементировать SP и использовать инкрементированный SP как адрес
*SP– –	Использовать SP как адрес и декрементировать SP

A.11.2 Дополнительный синтаксис языка Ассемблер

Ассемблер допускает укороченные синтаксические формы для некоторых инструкций. Эти дополнительные формы упрощают запись инструкций. Далее приведён список этих опционных форм синтаксиса:

- Регистр назначения может быть опущен в одинарных арифметических и логических операциях, когда тот же регистр используется в качестве источника. Например, ABSI R0, R0 может быть записана как ABSI R0. Используемые инструкции: ABSI, ABSF, FIX, FLOAT, NEGB, NEGF, NEGI, NORM, NOT, RND.

- Мнемоники всех трёхоперандных инструкций могут быть записаны без символа «3», например, ADDI3 R0, R1, R2 может быть записана как ADDI R0, R1, R2. Используемые инструкции: ADDC3, ADDF3, ADDI3, AND3, ANDN3, ASH3, LSH3, MPYF3, MPYI3, OR3, SUBB3, SUBF3, SUBI3, XOR3, MPYSHI3, MPYUHI3, CMPI3, CMPF3, TSTB3.

- Разрешены непрямые операнды с явно заданным нулевым смещением. В трёхоперандных или параллельных инструкциях операнды с нулевым смещением автоматически преобразуются в режим с отсутствием смещения.

Например:
LDI*+AR0(0), R1 – разрешено.
Также
ADDI3* + AR0(0), R1, R2
эквивалентно
ADDI3 *+AR0, R1, R2

- Непрямые операнды могут быть записаны без смещения, в таком случае предполагается единичное смещение.

Например:
LDI *AR0++(1), R0
может быть записано как
LDI *AR0++, R0

- Все условные инструкции включают в себя суффикс U для обозначения безусловной операции. Также, суффикс U может быть исключён из инструкции короткого безусловного перехода. Например, BU метка может быть записана как B метка.

- Метки могут быть записаны как с символом двоеточия (:), так и без него.

Например:
label0: NOP
label1 NOP
label2: (метка относится к следующей строке)

- Запрещены пустые выражения для указания смещения в косвенном режиме: LDI *+AR0(), R0 – запрещено.

- Операнды назначения длинного непосредственного режима для операторов BR или CALL могут быть написаны с символом @: BR метка может быть записано как BR @метка.

- Псевдооперация LDP может быть использована для загрузки регистра (обычно DP) 16 старшими разрядами перемещаемого адреса следующим образом: LDP addr, REG или LDP @addr, REG или LDP addr. Символ @ является дополнительным. LDP генерирует инструкцию LDIU с непосредственным операндом и специальным типом смещения.

- Параллельные инструкции могут быть написаны в обратном порядке.

Например:
ADDI
|| STI
может быть записано как
STI
|| ADDI

- Символ ||, обозначающий вторую часть параллельной инструкции, может быть написан в любом месте строки, начиная с нулевого столбца.

Например:

```
ADDI
|| STI
```

может быть записано как

```
ADDI
    || STI
```

- Если второй операнд параллельной инструкции тот же самый, что и третий (регистр назначения), третий операнд может быть опущен. Это даёт возможность написания трёх-операндной инструкции, которая выглядит как обычная двухоперандная инструкция. Например:

```
ADDI *AR0, R2, R2
|| MPYI *AR1, R0, R0
```

может быть записано как

```
ADD *AR0, R2
|| MPYI *AR1, R0
```

Инструкции (применимо для всех параллельных инструкций, имеющих в качестве второго операнда регистр), к которым относится вышесказанное: ADDI, ADDF, AND, MPYI, MPYF, OR, SUBI, SUBF, XOR.

- Все коммутируемые операнды в параллельных инструкциях могут быть записаны в ином порядке. Например, часть параллельной инструкции ADDI может быть записана одним из двух способов:

```
ADDI *AR0, R1, R2 или
ADDI R1, *AR0, R2
```

- Вышеописанное относится к параллельным инструкциям, содержащим одну из перечисленных инструкций: ADDI, ADDF, MPYI, MPYF, AND, OR, XOR.

A.11.3 Индивидуальное описание инструкций

В данном подразделе описана каждая инструкция языка Ассемблер ИС 1867ВЦ8Ф1 в алфавитном порядке. Информация о каждой инструкции включает синтаксис языка Ассемблер, выполняемую операцию, операнды, код, описание, количество циклов, разряды состояния и режима, примеры.

В таблице А.10 приведён набор регистров, используемых в операндах инструкций.

Таблица А.10 – Состав регистров процессорного ядра

Мнемоника регистра	Машинный 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 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
SP	14	Указатель системного стека
ST	15	Регистр состояния
DIE	16	Регистр разрешения прерываний ПДП
IIE	17	Регистр разрешения внутренних прерываний
IIF	18	IIOF(0-3)_1#, IIOF(0-3)_2# выводы и регистр флагов прерываний
RS	19	Регистр адреса начала повторения
RE	1A	Регистр адреса конца повторения
RC	1B	Счётчик повторений
IVTP	00	Указатель векторной таблицы системных прерываний
TVTP	01	Указатель векторной таблицы программных прерываний

A.11.4 Пример описания инструкций

На примере абстрактной инструкции INST подробно рассмотрим формат описания инструкций 1867ВЦ8Ф1.

INST инструкция

Синтаксис:

INST src, dst

или

INST1 src2, dst1

|| **INST2** src3, dst2

Каждая инструкция начинается с синтаксического выражения языка Ассемблер. Метки могут размещаться либо до инструкции (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, инструкция, операнд и поле комментария).

Примеры синтаксиса иллюстрируют общий однострочковый и двухстрочковый синтаксис, используемый в параллельной адресации. Необходимо обратить внимание, что символ (||), обозначающий параллельную адресную пару, может быть размещён в любом месте перед мнемоникой во второй строке. Первая инструкция в паре может иметь метку, но вторая иметь метки не может.

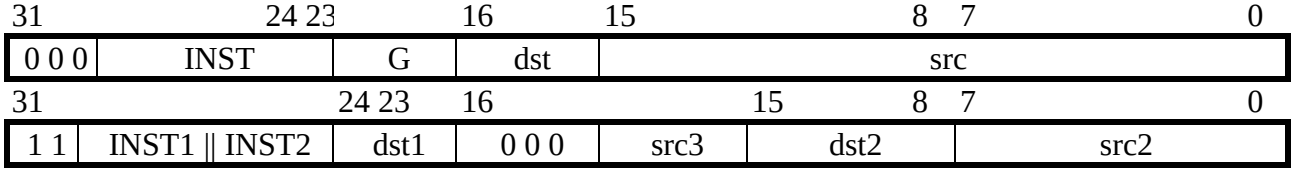
Операнды:

src: основные режимы адресации (G)

dst: регистр (R0 – R11)

Здесь представлен список типов операндов, которые используются в инструкции.

Код:



Примеры кодов приведены с использованием основной и параллельной адресации. Пара инструкций для примера параллельной адресации состоит из INST1 и INST2. Заметим, что оба отдельных операционных кода в этом случае – это 32-разрядные инструкции в процессорном ядре 1867ВЦ8Ф1.

Поля слова:

G	src режимы адресации
00	Регистр (R0 – R11)
01	Прямая
10	Косвенная
11	Непосредственная

Поля слов описываются адресным режимом, который соответствует каждому значению поля слова в операционном коде. Поле слова описано в таблице, соответствующей полю, указанному под операндами.

Операция:
|src| → dst
или
|src2| → dst1
|| src3 → dst2
Последовательность работы инструкции описывает процесс, который имеет место при исполнении инструкции. Для параллельных инструкции рабочая последовательность выполняется параллельно. Условия регистра состояния определяют режимы для инструкций с условиями, например Bcond.
dst: регистр (любой регистр или главный регистровый файл в ЦПУ)
или
src2: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (смещение = 0, 1, IR0, IR1)
Операнды определены в соответствии с режимом адресации и/или типом использованной адресации. Необходимо обратить внимание, что косвенная адресация использует смещения и индексные регистры.
Описание:
Описано исполнение инструкции и её влияние на состояние процессора и содержимое памяти. Описаны также ограничения, накладываемые на операнды процессором или языком Ассемблер. Описание идёт параллельно и дополняет информацию, приведённую в описании работы.
Разряды состояния:

LUF – фиксируемый флаг условия отрицательного переполнения при работе с ПЗ. Устанавливается всякий раз при установке UF (флаг отрицательного переполнения при работе в формате с ПЗ), в противном случае не изменяется.

LV – фиксируемый флаг условия 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 UF устанавливается в 1, в противном случае – 0.

N – флаг отрицательного условия. Для некоторых операций N имеем значение старшего значащего разряда выходного значения 1, если результат отрицательный, и 0 – в противном случае.

Z – флаг нулевого условия. 1 – при равенстве выходного результата 0 и 0 – в противном случае. Для логических инструкций и инструкций сдвига 1 при генерации 0 выхода, 0 – в противном случае.

V – флаг условия переполнения. Устанавливается в 1 при переполнении, иначе – в 0.

C – 1 флаг переноса. Устанавливается при возникновении заёма или переноса, 0 – в противном случае. Для инструкций сдвига этот флаг устанавливается по значению последнего сдвигаемого разряда; при нулевом сдвиге устанавливается в ноль.

Семь флагов условий, сохраняемые в регистре состояния (ST). Они обеспечивают информацию о свойствах результата или выхода арифметических или логических операций.
Разряд режима: Флаг режима переполнения OVM. В общем случае, на выполнение целочисленных операций влияет значение разряда OVM.

В основном, целые операции воздействуют на OVM значение бита.

Циклы:
1
Цифра определяет число циклов, необходимое для осуществления инструкции.
Пример:
INST @98AEh, R5

Перед инструкцией

DP	80h	DP
R5	07 6690 0000h	2,30562500e + 02

Память на 0080 98AEh

	5CDFh	1,00001107e + 00
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

80h		
R5	00 6690 0000h	1,80126593e + 00

Память на 80 98AEh

	5CDFh	1,00001107e + 00
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

Примерный код, представленный в вышеописанном формате, показывает влияние инструкции на системные указатели (DP) или (SP), регистры (R1) или (R5), значение в памяти по определённому адресу и семь разрядов состояния. Значения, заданные для регистров, включают начальный ноль для указания экспоненты в операциях с ПЗ. Для всех регистров и адресов памяти представлен перевод значений в десятичную систему. Разряды состояния перечислены в порядке, в котором они представлены в языке Ассемблер или в симуляторе (см. таблицу А.8 для более полной информации о разрядах состояния).

ABSF инструкция

Синтаксис:

ABSF src, dst

Операнды:

src: основные режимы адресации (G)

dst: регистры (R0 – R11)

Код:

31 29	23	21	16	0
0 0 0	0 0 0 0 0 0 0	G	dst	src

Поля слова:

G	src режимы адресации
00	Регистр (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

Разряды состояния:

LUF	Не изменяется.
LV	1 при возникновении ПЗ переполнения, в противном случае не изменяется.
UF	0.
N	0.
Z	1 при генерации нулевого результата, иначе 0.
V	1 при возникновении ПЗ переполнения, иначе 0.
C	Не изменяется.

Разряд режима:

OVM операция зависит от значения разряда OVM.

Циклы:

1

Пример:

ABSF R4, R7

Перед инструкцией

R4	05C8000F971h	–9,90337307e+27
R7	07D251100AEh	5,48527255e+37
LV	0	
Z	0	
V	0	

После инструкции

R4	05C8000F971h	–9,90337307e+27
R7	05C7FFF068Fh	9,90337307e+27
LV	0	
Z	0	
V	0	

ABSF || STF инструкция

Синтаксис:

ABSF src2, dst1
|| **STF** src3, dst2

Операнды:

src2: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (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

Поля слова:

Нет

Операция:

|src2| → dst1
|| src3 → dst2

Описание:

Абсолютное значение числа в формате с ПЗ и параллельно сохранить значение 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, 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, src2 считывается до записи в dst2.
Если src3 и dst1 указывают на один и тот же адрес, src3 считывается до записи в dst1.
Переполнение возникает, если src(man) = 80000000h и src(exp) = 7Fh. Результат – dst(man) = 7FFFFFFFh и dst(exp) = 7Fh.

Циклы:

1

Разряды состояния:

LUF	Не изменяется.
LV	1 при возникновении ПЗ переполнения, в противном случае не изменяется.
UF	0.
N	0.
Z	1 при генерации нулевого результата, иначе 0.
V	1 при возникновении ПЗ переполнения, иначе 0.
C	Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Пример:

```
ABSF  *++AR3(IR1) ,R4
|| STF   R4,*–AR7(1)
```

Перед инструкцией

AR3	80 9800h	
IR1	0AFh	
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	
R4	574C0 0000h	6,118750e+01
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	

ABSI инструкция

Синтаксис:

ABSI src, dst

Операнды:

src: основные режимы адресации (G)

dst: регистр (любой регистр в основном регистровом файле ЦПУ)

Код:

31 29	24 23	21	16	0
0 0 0	0 0 0 0 0 1	G	dst	src

Поля слова:

G	src режимы адресации
00	Регистр (любой регистр из основного регистрового файла ЦПУ)
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, результатом будет dst = 7FFF FFFFh. Если ST(OVM) = 0, результатом будет dst = 8000 0000h.

Разряды состояния:

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, в противном случае не изменяется.
UF 0.
N 0.
Z 1 при генерации нулевого результата, иначе 0.
V 1 при возникновении целочисленного переполнения, иначе 0.
C Не изменяется.

Разряд режима:
OVM Операция зависит от значения разряда 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

ABSI || STI инструкция
Синтаксис:
ABSI src2, dst1
|| **STI** src3, dst2
Операнды:
src2: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (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

Поля слова:
Нет
Операция:
|src2| → dst1
|| src3 → dst2
Описание:
Абсолютное значение числа целого и сохранить значение 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 (ABSI), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено инструкцией ABSI.
Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Переполнение возникает, если src = 80000000h. Если ST(OVM) = 1, результат – dst = 7FFFFFFFh. Если ST(OVM) = 0, результат – dst = 80000000h.

Разряды состояния:

LUF	Не изменяется.
LV	1 при возникновении целочисленного переполнения, в противном случае не изменяется.
UF	0.
N	0.
Z	1 при генерации нулевого результата, иначе 0.
V	1 при возникновении целочисленного переполнения, иначе 0.
C	Не изменяется.

Разряд режима:

OVM операция зависит от значения разряда OVM.

Циклы:

1

Пример:

```
ABSI *–AR5(1) ,R5
|| STI R1,*AR2 – – (IR1)
```

Перед инструкцией			После инструкции		
AR5	80 99E2h		AR5	80 99E2h	
R5	0h		R5	35h	53
R1	42h	66	R1	42h	66
AR2	80 98FFh		AR1	80 98F0h	
IR1	0Fh		IR1	0Fh	
Данные в 80 99E1h			Данные в 80 99E1h		
	58B 4000h	–53		58B 4000h	–53
Данные в 80 98FFh			Данные в 80 98FFh		
	0h	2		733 C000h	66
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	

ADDC инструкция

Синтаксис:

ADDC src, dst

Операнды:

src: основные режимы адресации (G)

dst: регистр (любой регистр в основном регистровом файле ЦПУ)

Код:

31 24	23 16	15 8	7 0
0 0 0 0 0 0 0 1 0	G	dst	src

Поля слова:

G	src режимы адресации
00	Регистр (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
dst + src + C → dst
Описание:
Сумма операндов dst и src и флаг C (перенос) загружается в регистр dst. Операнды src и dst являются целыми со знаком.
Разряды состояния:
Флаги состояния изменяются только если регистр назначения – один из R7 – R0.

LUF	Не изменяется.
LV	1 при возникновении целочисленного переполнения, в противном случае не изменяется.
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.

Разряд режима:
OVM операция зависит от значения разряда OVM.
Циклы:
1
Пример:
ADDC R1, R5

	Перед инструкцией			После инструкции		
R1	00FFFF 5C25h		–41 947	R1	00FFFF 5C25h	–41 947
R5	00FFFF 019Eh		–65 122	R5	00FFFE 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	

ADDC3 инструкция

Синтаксис:
ADDC3 src2, src1, dst
Операнды:
src1, src2: тип 1 или тип 2 трёхоперандной адресной формы
dst: регистровый режим (любой регистр ЦПУ в основном регистровом файле)
Код:
Тип 1

31 … 24	23	22 … 16	15 … 8	7 … 0
0 0 1 0 0 0 0 0 0	T	dst	src1	src2

Тип 2

31 … 24	23	22 … 16	15 … 8	7 … 0
0 0 1 1 0 0 0 0 0	T	dst	src1	src2

Поля слова:
Тип 1

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	Регистр (любой регистр ЦПУ)
01	Косвенная (смещение = 0, 1, IR0, IR1)	Регистр (любой регистр ЦПУ)
10	Регистр (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	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	8-разрядная знаковая непосредственная
01	Регистр (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 флаг C (перенос) загружается в регистр dst. Операнды src1, src2 и dst являются целыми со знаком.

Разряды состояния:

Если ST(SET COND) = 0, флаги состояния изменяются только в том случае, если регистр назначения – один из R0 – R11. Если ST(SET COND) = 1, они изменяются для всех регистров.

LUF	Не изменяется.
LV	1 при возникновении целочисленного переполнения, в противном случае не изменяется.
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.

Разряд режима:

OVM операция зависит от значения разряда OVM.

Циклы:

1

Пример:

Нет

ADDF инструкция

Синтаксис:

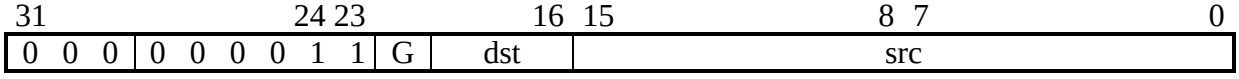
ADDF src, dst

Операнды:

src: основные режимы адресации (G)

dst: регистр (R0 – R11)

Код:



Поля слова:

G	src режимы адресации
00	Регистр (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.

Разряды состояния:

LUF	1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	1 при возникновении ПЗ-переполнения, в противном случае 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	1 при возникновении ПЗ-переполнения, иначе 0.
C	Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

ADDF *AR4++(IR1),R5

Перед инструкцией			После инструкции		
AR4	80 9800h		AR4	80 992Bh	
R1	12Bh	66	R1	12Bh	
R5	057980 0000h	6,23750e+01	R5	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		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	

ADDF3 инструкция

Синтаксис:

ADDF3 src2, src1, dst

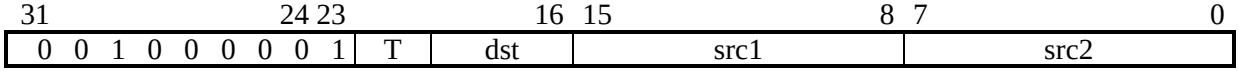
Операнды:

src1, src2: тип 1 или тип 2 трёхоперандного адресного режима

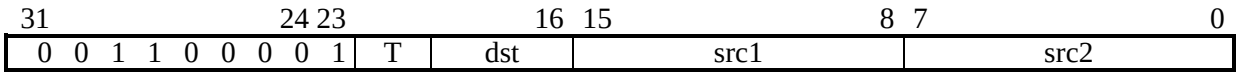
dst: регистровый режим (R0 – R11)

Код:

Тип 1



Тип 2



Поля слова:

Тип 1

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (R0 – R11)	Регистр (R0 – R11)
01	Косвенная (смещение = 0, 1, IR0, IR1)	Регистр (R0 – R11)
10	Регистр (R0 – R11)	Косвенная (смещение = 0, 1, IR0, IR1)
11	Косвенная (смещение = 0, 1, IR0, IR1)	Косвенная (смещение = 0, 1, IR0, IR1)

Тип 2

T	src1 режимы адресации	src2 режимы адресации
01	Регистр (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 → dst

Описание:

Сумма операндов src1 и src2 загружается в регистр dst. Операнды src1, src2 и dst являются числами в формате с ПЗ.

Разряды состояния:

LUF 1 при потере значимости результата с ПЗ, иначе не изменяется.
LV 1 при возникновении ПЗ-переполнения, в противном случае 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 1 при возникновении ПЗ-переполнения, иначе 0.
C Не изменяется.

Циклы:

1

Разряд режима:

OVM 2 операция не зависит от значения разряда OVM.

Пример:

ADDF3 *+AR1(2),*+AR1(8),R4

	Перед инструкцией		После инструкции	
AR1	2FF820h		2FF820h	
R4	0h		070DB2 0000h	1,41695313e+02
Данные в 22F F822h			Данные в 22F F828h	
	700 F000h	1,28940e+02	34C 2000h	1,27590e+01
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		0	
V	0		0	
C	0		0	

ADDF3 || STF инструкция

Синтаксис:

ADDF3 src2, src1, dst1
|| **STF** src3, dst2

Операнды

src1: регистр (R0 – R7)
src2: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (смещение = 0, 1, IR0, IR1)

Код:

31							24	23			16	15			8	7			0
1	1	0	0	1	1	0	dst1		src1		src3			dst2				src2	

Поля слова:

Нет

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, STF имеет на входе содержимое регистра до того, как оно было изменено инструкцией ADDF3.

Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Разряды состояния:

LUF	1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	1 при возникновении ПЗ-переполнения, в противном случае 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	1 при возникновении ПЗ-переполнения, иначе 0.
C	Не изменяется.

Разряд режима:

OVM Операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

```
ADDF3 *+AR3(IR1),R2,R5
|| STF    R4,*AR2
```

Перед инструкцией			После инструкции		
AR3	80 9800h		AR3	80 9800h	
IR1	0A5h		IR1	0A5h	
R2	070C80 0000h	1,4050e+02	R2	070C80 0000h	1,4050e+02
R5	0h		R5	082020 0000h	3,20250e+02
R4	057B40 0000h	6,281250e+01	R4	057B40 0000h	6,281250e+01
AR2	80 98F3h		AR2	80 98F3h	
Данные в 80 98A5h			Данные в 80 98A5h		
	733 C000h	1,79750e+02		733 C000h	1,79750e+02
Данные в 80 98F3h			Данные в 80 98F3h		
	0h			57B 4000h	6.28125e+01
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	

ADDI инструкция

Синтаксис:

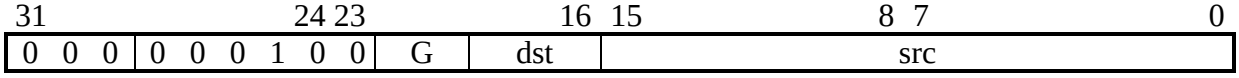
ADDI src, dst

Операнды:

src: основные режимы адресации (G)

dst: регистр (любой регистр из основного регистрового файла ЦПУ)

Код:



Операция:

dst + src → dst

Поля слова:

G	src режимы адресации
00	Регистр (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.

Разряды состояния:

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 изменяется.
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.

Разряд режима:

OVM Операция зависит от значения разряда 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	

ADDI3 инструкция

Синтаксис:

ADDI3 src2, src1, dst

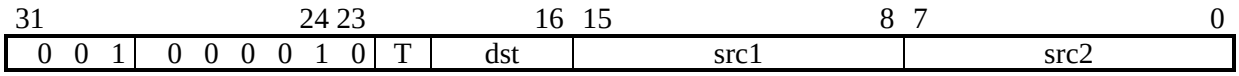
Операнды:

src1, src2: тип 1 или тип 2 трёхоперандного адресного режима

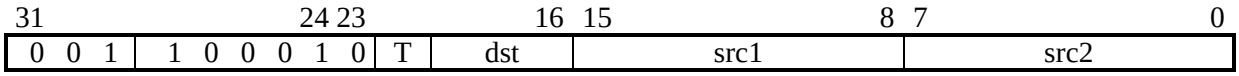
dst: регистровый режим (любой регистр основного регистрового файла ЦПУ)

Код:

Тип 1



Тип 2



Поля слова:

Тип 1

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой ЦПУ регистр)	Регистр (любой ЦПУ регистр)
01	Косвенная (смещение = 0, 1, IR0, IR1)	Регистр (любой ЦПУ регистр)
10	Регистр (любой ЦПУ регистр)	Косвенная (смещение = 0, 1, IR0, IR1)
11	Косвенная (смещение = 0, 1, IR0, IR1)	Косвенная (смещение = 0, 1, IR0, IR1)

Тип 2

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	8-битная знаковая непосредственная
01	Регистр (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-битная знаковая непосредственная
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.

Разряды состояния:

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 изменяется.

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.

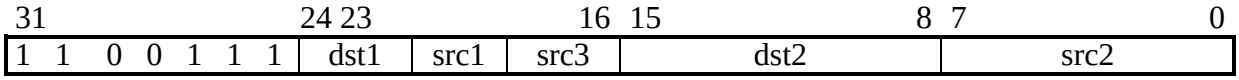
Разряд режима:

OVM операция зависит от значения разряда OVM.

Циклы:
1
Пример:
Нет

ADDI3 || STI инструкция

Синтаксис:
ADDI3 src2, src1, dst1
|| **STI** src3, dst2
Операнды:
src1: регистр (R0 – R7)
src2: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (смещение = 0, 1, IR0, IR1)
Код:



Поля слова:
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 инструкцией ADDI3.
Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.
Разряды состояния:

LUF	Не изменяется.
LV	1 при возникновении целочисленного переполнения, в противном случае не изменяется.
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.

Разряд режима:
OVM операция зависит от значения разряда 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	

AND инструкция

Синтаксис:

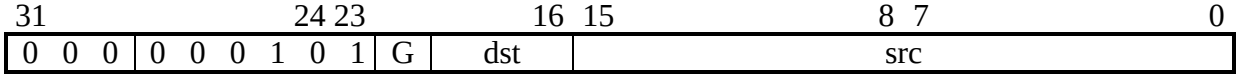
AND src, dst

Операнды:

src: основные режимы адресации (G)

dst: регистр (любой регистр из основного регистрового файла ЦПУ)

Код:



Поля слова:

G	src режимы адресации
00	Регистр (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:

dst AND src → dst

Описание:

Результат поразрядного логического И между операндами dst и src записывается в регистр dst. Предполагается, что src и dst являются целыми без знака.

Разряды состояния:

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.

Разряд режима:
OVM операция не зависит от значения разряда OVM.

Циклы:
1

Пример:
AND R1, R2

	Перед инструкцией
R1	80h
R2	0AFFh
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	1

	После инструкции
R1	80h
R2	80h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	1

AND3 инструкция

Синтаксис:
AND3 src2, src1, dst

Операнды:
src1, src2: тип 1 или тип 2 трёхоперандного адресного режима
dst: режим регистра (любой регистр главного регистрового файла ЦПУ)

Код:
Тип 1

31 … 24	23 … 16	15 … 8	7 … 0
0 0 1 0 0 0 0 1 1 T	dst	src1	src2

Тип 2

31 … 24	23 … 16	15 … 8	7 … 0
0 0 1 1 0 0 0 1 1 T	dst	src1	src2

Поля слова:
Тип 1

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой ЦПУ регистр)	Регистр (любой ЦПУ регистр)
01	Косвенная (смещение = 0, 1, IR0, IR1)	Регистр (любой ЦПУ регистр)
10	Регистр (любой ЦПУ регистр)	Косвенная (смещение = 0, 1, IR0, IR1)
11	Косвенная (смещение = 0, 1, IR0, IR1)	Косвенная (смещение = 0, 1, IR0, IR1)

Тип 2

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	8-битная знаковая непосредственная
01	Регистр (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-битная знаковая непосредственная
11	Косвенная *+ARn1 (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:
src1 & src2 → dst
Описание:
Результат поразрядного логического И между операндами src1 и src2 загружается в регистр dst. Предполагается, что операнды src1, src1 и dst являются целыми числами без знака. В режиме непосредственной адресации src2 расширяется на знак.
Разряды состояния:
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.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
Различие между AND и AND3 проиллюстрировано в этом примере:

AND3 80h, R0, R0	R0 = FFFF FFFFh	R0 = FFFF FF80h
AND 80h, R0	R0 = FFFF FFFFh	R0 = 0000 0080h

AND3 || STI инструкция

Синтаксис:
AND src2, src1, dst1
||**STI** src3, dst2
Операнды:
src1: регистр (R0 – R7)
src2: косвенная (смещение = 0, 1, IR0, IR1)
dst: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (смещение = 0, 1, IR0, IR1)
Код:

31	24 23		16 15	8 7	0
1 1 0 1 0 0 0	dst1	src1	src3	dst2	src2

Поля слова:
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 инструкцией AND3.
Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы:
1
Разряды состояния:

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.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Пример:

```
AND3 *+AR1(IR0),R4,R7
|| STI   R3,*AR2
```

Перед инструкцией

AR1	80 99F1h
IR0	8h
R4	0A323h
R7	0h
R3	35h
AR2	80 983Fh

53

Данные в 80 99F9h

5C53h

Данные в 80 983Fh

	0h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

AR1	80 99F1h
IR0	8h
R4	0A323h
R7	03h
R3	35h
AR2	80 983Fh

53

Данные в 80 99F9h

5C53h

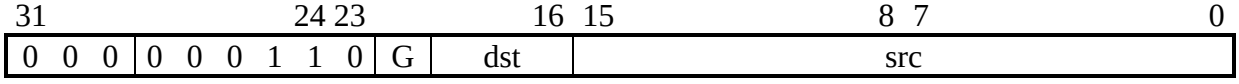
Данные в 80 983Fh

	35h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

53

ANDN инструкция

Синтаксис:
ANDN src, dst
Операнды:
src: основные режимы адресации (G)
dst: регистр (любой регистр из основного регистрового файла ЦПУ)
Код:



Поля слова:

G	src режимы адресации
00	Регистр (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 без-знаковыми целыми.

Разряды состояния:
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.

Разряд режима:
OVM операция не зависит от значения разряда 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

Данные в 80 980Ch

	0A02h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

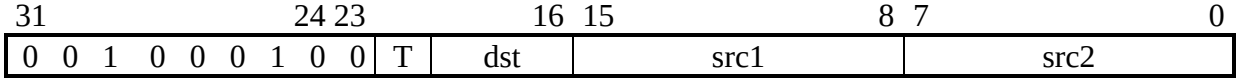
ANDN3 инструкция

Синтаксис:
ANDN3 src2, src1, dst

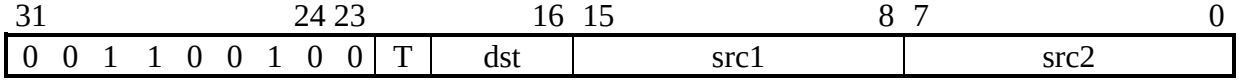
Операнды:

src1, src2:	тип 1 или тип 2 трёхоперандного адресного режима
dst:	регистровый режим (любой регистр главного регистрового файла ЦПУ)

Код:
Тип 1



Тип 2



Поля слова:

Тип 1

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой ЦПУ регистр)	Регистр (любой ЦПУ регистр)
01	Косвенная (смещение = 0, 1, IR0, IR1)	Регистр (любой ЦПУ регистр)
10	Регистр (любой ЦПУ регистр)	Косвенная (смещение = 0, 1, IR0, IR1)
11	Косвенная (смещение = 0, 1, IR0, IR1)	Косвенная (смещение = 0, 1, IR0, IR1)

Тип 2

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	8-битная знаковая непосредственная
01	Регистр (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-битная знаковая непосредственная
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.

Разряды состояния:

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.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

Нет

ASH инструкция

Синтаксис:

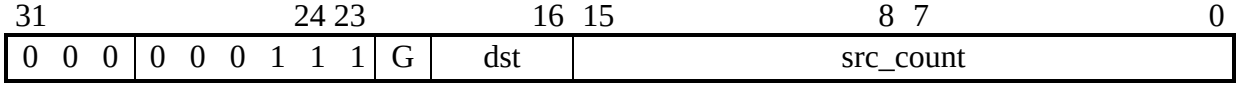
ASH src_count, dst

Операнды:

src_count: основные режимы адресации (G)

dst: регистр (любой регистр из основного регистрового файла ЦПУ)

Код:



Поля слова:

G	src режимы адресации
00	Регистр (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:

Если (count ≥ 0):

dst << count → dst

В противном случае:

dst >> |count| → dst

Описание:

Младшие 8 значащих разрядов оператора count используются для генерации двоичного дополнения счётчика 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.

Разряды состояния:

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 изменяется.
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 счётчик сдвига 0.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример 1:

ASH R1,R3

	Перед инструкцией			После инструкции
R1	10h	16	R1	10h
R3	0A E000h		R3	0E000 0000h
LUF	0		LUF	0
LV	0		LV	1
UF	0		UF	0
N	0		N	1
Z	0		Z	0
V	0		V	1
C	0		C	0

Пример 2:

ASH @98C3h,R5

	Перед инструкцией			После инструкции	
DP	80h	16	DP	80h	
R5	0AEC0 0001h		R5	0FFFF FFAEh	
Данные в 80 98C3h			Данные в 80 98C3h		
	0FFE8h	–24		0FFE8h	–24
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	1	

ASH3 инструкция

Синтаксис:

ASH3 src_count, src, dst

Операнды:

src, src_count: тип 1 или тип 2 трёхоперандного адресного режима

dst: регистровый режим (любой регистр главного регистрового файла ЦПУ)

Код:

Тип 1

31 … 24	23	22 … 16	15 … 8	7 … 0
0 0 1 0 0 0 1 0 1	T	dst	src	src_count

Тип 2

31 … 24	23	22 … 16	15 … 8	7 … 0
0 0 1 1 0 0 1 0 1	T	dst	src	src_count

Поля слова:

Тип 1

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой ЦПУ регистр)	Регистр (любой ЦПУ регистр)
01	Косвенная (смещение = 0, 1, IR0, IR1)	Регистр (любой ЦПУ регистр)
10	Регистр (любой ЦПУ регистр)	Косвенная (смещение = 0, 1, IR0, IR1)
11	Косвенная (смещение = 0, 1, IR0, IR1)	Косвенная (смещение = 0, 1, IR0, IR1)

Тип 2

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	8-битная знаковая непосредственная
01	Регистр (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-битная знаковая непосредственная
11	Косвенная *+ARn1 (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:

Если (count ≥ 0):

src << count → dst

В противном случае:

src >> |count| → dst

Описание:

Семь младших значащих разрядов операнда count используются для генерации двоичного дополнения счётчика сдвига до 32 разрядов.

Если операнд count больше 0, операнд src сдвигается влево на число разрядов, определённых операндом count. Младшие разряды при сдвиге заполняются нулями, старшие сдвигаются вовне через разряд переноса регистра состояния C.

Арифметический левый сдвиг:

$$C \leftarrow src2 \leftarrow 0$$

Если операнд count меньше 0, операнд src сдвигается вправо на число разрядов, определённых 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:

$$Знак\ dst \rightarrow src2 \rightarrow C$$

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.

Разряды состояния:

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. 0, если count = 0.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

Нет

ASH3 || STI инструкция

Синтаксис:

ASH3 src_count, src2, dst1
|| **STI** src3, dst2

Операнды:

src_count регистр (R0 – R7)
src2: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (смещение = 0, 1, IR0, IR1)
Код:

31 24	23 16	15 8	7 0
1 1 0 1 0 0 1 dst1	src_count src3	dst2	src2

Поля слова:

Нет

Операция:

count=7 наименее значащим битам src_count
Если (count ≥ 0):
src2 << count → dst1
В противном случае:
src2 >> |count| → dst1
|| src3 → dst2

Описание:

Семь младших значащих разрядов операнда count используются для генерации двоичного дополнения счётчика сдвига до 32 разрядов.
Если операнд count больше 0, операнд src сдвигается влево на число разрядов, определённых операндом count. Младшие разряды при сдвиге заполняются нулями, старшие сдвигаются вовне через разряд переноса регистра состояния C.
Арифметический левый сдвиг:

$$C \leftarrow src2 \leftarrow 0$$

Если операнд count меньше 0, операнд src сдвигается вправо на число разрядов, определённых 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:

$$Знак\ src2 \rightarrow src2 \rightarrow C$$

Если операнд count равен 0, сдвиг не происходит, и разряд C (перенос) устанавливается в 0. Операнды count и dst являются целыми со знаком.
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, STI имеет на входе содержимое регистра до того, как оно было изменено инструкцией ASH3.
Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Разряды состояния:

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. 0, если count = 0.

Разряд режима:
OVM операция зависит от значения разряда 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	
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

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	1	
Z	0	
V	0	
C	0	

Bcond инструкция
Синтаксис:
Bcond src
Операнды:
src: Режимы 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

Поля слова:

B	src режимы адресации
0	Регистр
1	Относительно PC

Операция:
Если cond – истина:
Если src в регистровом режиме адресации (любой регистр основного регистрового файла ЦПУ),
src → PC
Если src в режиме адресации относительно PC (метка или адрес),
смещение + PC + 1 → 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). Если операнд src установлен в режим регистровой адресации, содержимое указанного регистра загружается в PC. Если операнд src установлен в режим адресации относительно PC, язык Ассемблер генерирует смещение; смещение = метка – (PC инструкции перехода + 1). Это смещение сохраняется как 16-разрядное целое со знаком в 16 младших значащих разрядах слова инструкции перехода. Смещение добавляется к PC инструкции перехода + 1 для генерации нового PC.
Ядро процессора 1867ВЦ8Ф1 поддерживает 20 кодов условий, которые могут использоваться этой инструкцией.

Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
4 (независимо от того, произошёл переход или нет)
Пример:
BZ R0

	Перед инструкцией		После инструкции
PC	2B00h	PC	3FF00h
R0	0003 FF00h	R0	0003 FF00h
LUF	0	LUF	0
LV	0	LV	0
UF	0	UF	0
N	0	N	0
Z	1	Z	1
V	0	V	0
C	0	C	0

Примечание – Если BZ инструкция вызвана непосредственно после RND инструкции с нулевым операндом, ветвление не происходит, потому что нулевой флаг не установлен. Для разрешения этой проблемы, выполните BZYF вместо BZ инструкции.

BcondAF инструкция

Синтаксис:
BcondAF src
Операнды:
src: Режимы 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

Поля слова:

B	src режимы адресации
0	Регистр
1	Относительно PC

Операция:
Если cond – истина:

Если src в регистровом режиме адресации (любой регистр основного регистрового файла ЦПУ), src → PC.

Если src в режиме адресации относительно PC (метка или адрес), смещение + PC перехода + 3 → PC.

В противном случае:

Если cond – ложь: аннулирование выполнения фазы первой следующей инструкции и результата чтения и выполнения фаз второй и третьей последующих инструкций и продолжение.

Описание:
Если условие истинно, инструкция BcondAF выполняет три инструкции, следующие за переходом, и потом 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. Три следующих за BcondAF инструкции не влияют на cond. Если src операнд в режиме регистра, содержимое указанного регистра загружается в PC. Если src операнд в относительном PC режиме, сумма PC инструкции перехода + 3 заносится в PC. В относительном PC режиме поле смещения интерпретируется как 16-разрядное знаковое целое.

Ни одна из трёх инструкций, следующих за BcondAF, не должна изменять ход программы. В течение выполнения BcondAF прерывания отключены.
BcondAF особенно полезна для управления выходом в конце цикла. 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.

Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.

Циклы:
1

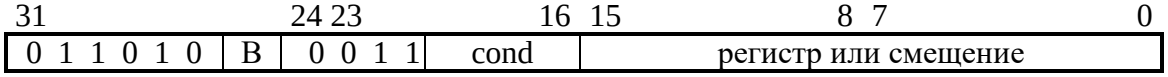
Пример:
Нет

BcondAT инструкция

Синтаксис:
BcondAT src

Операнды:
src: Режимы адресации условного перехода

Код:



Поля слова:

B	src режимы адресации
0	Регистр
1	Относительно PC

Операция:
Если cond – истина:
Если src в регистровом режиме адресации (любой регистр основного регистрового файла ЦПУ), src → PC аннулирование выполнения фазы первой следующей инструкции и результата чтения и выполнения фаз второй, третьей и последующих инструкций и продолжение.
Если src в режиме адресации относительно PC (метка или адрес), смещение + PC перехода + 3 → PC аннулирование выполнения фазы первой следующей инструкции и результата чтения и выполнения фаз второй, третьей и последующих инструкций и продолжение.
В противном случае – продолжение.
Описание:
Если условие истинно, 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 представлен в режиме регистра, содержимое указанного регистра заносится в PC. Если src операнд в относительном-PC режиме, сумма PC инструкции перехода + 3 заносится в PC. В относительном PC режиме поле смещения интерпретируется как 16-разрядное знаковое целое.

Ни одна из трёх инструкций, следующих за BcondAT, не должна изменять ход 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.
Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
Нет

BcondD инструкция

Синтаксис:
BcondD src
Операнды:
src: Режимы 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	регистр или смещение

Поля слова:

B	src режимы адресации
0	Регистр
1	Относительно PC

Операция:

Если cond – истина:

Если src в регистровом режиме адресации (любой регистр в основном регистровом файле ЦПУ), src → PC.

Если src в режиме адресации относительно PC (метка или адрес), смещение + PC + 3 → PC.

В противном случае – продолжение.

Описание:

BcondD означает задержанный переход, при котором обеспечивается выборка трёх инструкций до изменения PC. В результате получается одноцикловый переход, и три инструкции, следующие за BcondD, не влияют на cond.

Три инструкции не влияют на течение программы. Прерывания отключены во все время выполнения BcondD.

Переход осуществляется, если условие истинно. 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). Это смещение сохраняется как 16-разрядное целое со знаком в 16 младших значащих разрядах слова инструкции перехода. Смещение добавляется к PC инструкции перехода + 3 для генерации нового PC. Ядро процессора 1867ВЦ8Ф1 поддерживает 20 кодов условий, которые могут быть использованы с этой инструкции.

Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Циклы:

1

Разряд режима:

OVM операция не зависит от значения разряда 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

	После инструкции
PC	77h
R0	0
LUF	0
LV	0
UF	0
N	0
Z	1
V	0
C	0

BR инструкция

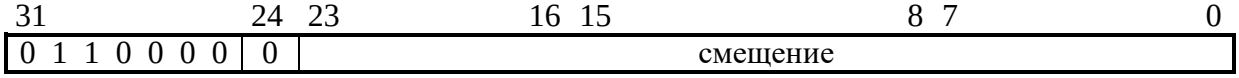
Синтаксис:

BR src

Операнды:

src: режим относительной-PC адресации

Код:



Поля слова:

Нет

Операция:

PC + 1 + смещение → PC

Описание:

Обеспечивает безусловный переход. Ассемблер генерирует смещение: смещение = src – (PC инструкции ветвления + 1), которое помещается как 24-разрядное знаковое целое в 24 младших бита кода инструкции ветвления. Это смещение добавляется к PC инструкции ветвления, плюс 1 для генерации нового PC.

Примечание:

Для стандартного перехода разряд 24 равен 0.

Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

4

Пример:

Нет

BRD инструкция

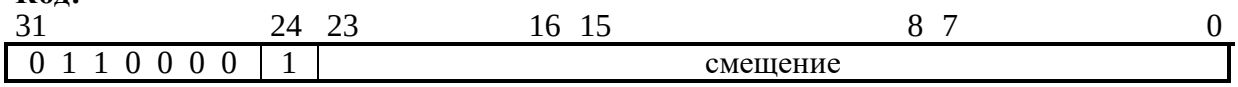
Синтаксис:

BRD src

Операнды:

src: режим относительной-PC адресации

Код:



Поля слова:

Нет

Операция:

PC + 3 + смещение → PC

Описание:

Обеспечивает задержанный безусловный переход. Ассемблер генерирует смещение: смещение = src – (PC инструкции ветвления + 3), которое помещается как 24-разрядное знаковое целое в 24 младших бита кода инструкции ветвления. Это смещение добавляется к PC инструкции ветвления, 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
Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Пример:
Нет

CALL инструкция

Синтаксис:
CALL src
Операнды:
src: режим относительной-PC адресации
Код:

Поля слова:
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. Смещение помещается как 24-разрядное знаковое целое в 24 младших бита кода инструкции ветвления. Это смещение добавляется к PC инструкции ветвления, плюс 1 для генерации нового PC.
Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
4
Пример:
Нет

CALLcond инструкция

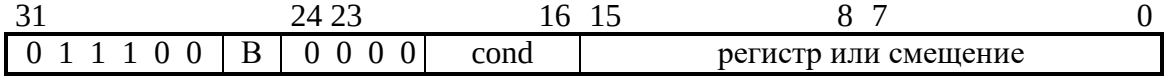
Синтаксис:

CALLcond src

Операнды:

src: Режим адресации условного перехода

Код:



Поля слова:

B	src режимы адресации
0	Регистр
1	Относительно PC

Операция:

Если cond – истина:

Следующий PC → *++SP

Если src в регистровом режиме адресации (любой регистровый файл ЦПУ), src → PC

Если src в режиме адресации относительно PC (метка или адрес), смещение + PC + 1 → PC.

В противном случае – продолжение.

Описание:

Вызов осуществляется, если условие истинно. Если условие истинно, следующее значение PC загружается в системный стек. 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 + 1), которое помещается как 16-разрядное целое со знаком в 16 младших значащих разрядах кода инструкции вызова. Смещение добавляется к PC инструкции ветвления + 1 для генерации нового PC.

В ядре процессора 1867ВЦ8Ф1 имеется 20 кодов условий, которые могут быть использованы с этой инструкцией.

Разряды состояния:

LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

5 (вне зависимости от того, что какое-либо условие истинно или нет)

Пример:

CALLNZ R5

Перед инструкцией

PC	123h
SP	80 9835h
R5	789h

LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

PC	789h
SP	80 9836h
R5	789h

Данные в 9836h

124h

LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

CMPF инструкция

Синтаксис:

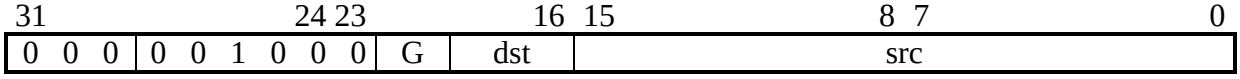
CMPF src, dst

Операнды:

src: основные режимы адресации (G)

dst: регистр (R0 – R11)

Код:



Поля слова:

G	src режимы адресации
00	Регистр (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.

Разряды состояния:

LUF 1 при потере значимости результата с ПЗ, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае 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 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда OVM

Циклы:
1
Пример:
CMPF *+AR4,R6

Перед инструкцией

AR4	80 98F2h	
R6	070C 8000h	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	

CMPF3 инструкция

Синтаксис:
CMPF3 src2, src1
Операнды:
src1 – **src2:** тип 1 или тип 2 трёхоперандного 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:
Тип 1

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (R0 – R11)	Регистр (R0 – R11)
01	Косвенная (смещение = 0, 1, IR0, IR1)	Регистр (R0 – R11)
10	Регистр (R0 – R11)	Косвенная (смещение = 0, 1, IR0, IR1)
11	Косвенная (смещение = 0, 1, IR0, IR1)	Косвенная (смещение = 0, 1, IR0, IR1)

Тип 2

T	src1 режимы адресации	src2 режимы адресации
01	Регистр (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, 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.

Разряды состояния:

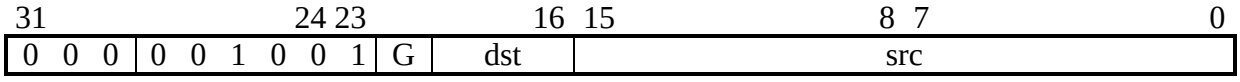
LUF	1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	1 при возникновении ПЗ-переполнения, в противном случае не изменяется.
UF	1 при потере значимости результата с ПЗ, иначе 0.
N	1 при генерации отрицательного результата, иначе 0.
Z	1 при генерации нулевого результата, иначе 0.
V	1 при возникновении ПЗ-переполнения, иначе 0.
C	Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
CMPF3 *AR2,*AR3– – (1)

	Перед инструкцией		После инструкции		
AR2	80 9831h		AR2	80 9831h	
AR3	80 9852h		AR3	809851h (decrement)	
Данные в 80 9831h			Данные в 80 9831h		
	58B 4000h	2,5044e+02		58B 4000h	2,5044e+02
Данные в 80 9852h			Данные в 80 9852h		
	57A2000h	6,253125e+01		733 C000h	6,253125e+01
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	

CMPI инструкция

Синтаксис:
CMPI src, dst
Операнды:
src: основные режимы адресации (G)
dst: регистр (любой регистр в основном регистровом файле ЦПУ)
Код:



Поля слова:

G	src режимы адресации
00	Регистр (любой регистр в основном регистровом файле 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.

Разряды состояния:

LUF	Не изменяется.
LV	1 при возникновении целочисленного переполнения, в противном случае не изменяется.
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 заёма, иначе 0.

Разряд режима:
OVM операция не зависит от значения разряда OVM.

Циклы:
1

Пример:
CMPI R3,R7

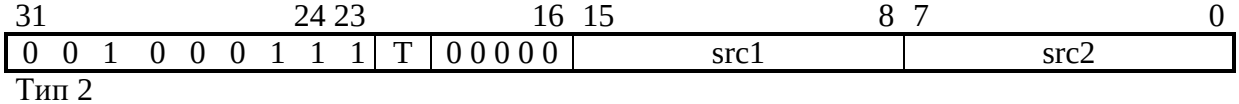
	Перед инструкцией			После инструкции	
R3	898h	2200	R3	80 98AFh	2200
R7	3E8h	1000	R7	3E8h	1000
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	0		C	0	

CMPI3 инструкция

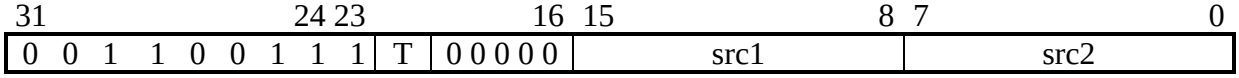
Синтаксис:
CMPI3 src2, src1

Операнды:
src1 – **src2** тип 1 или тип 2 трёхоперандного адресного режима.

Код:
Тип 1



Тип 2



Поля слова:
Тип 1

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	Регистр (любой регистр ЦПУ)
01	Косвенная (смещение = 0, 1, IR0, IR1)	Регистр (любой регистр ЦПУ)
10	Регистр (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	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	8-разрядная знаковая непосредственная
01	Регистр (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)

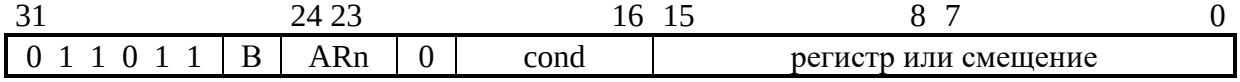
Операция:
src1 – src2
Описание:
Операнд src2 вычитается из операнда src1. Результат никуда не заносится, обеспечивая, таким образом, возможность сравнения без изменения значения каких бы то ни было операндов. Предполагается, что операнды dst и src являются целыми со знаком. Хотя инструкция имеет только два операнда, она имеет вид трёхоперандной, т. к. операнды определены в трёхоперандном формате.
Разряды состояния:

LUF	Не изменяется.
LV	1 при возникновении целочисленного переполнения, в противном случае не изменяется.
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 заёма, иначе 0.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
Нет

DBcond инструкция

Синтаксис:
DBcond ARn, src
Операнды:
src: режимы адресации условного перехода (B)
ARn: вспомогательный регистр
Код:



Поля слова:

B	src режимы адресации
0	Регистр
1	Относительно PC

Операция:
ARn – 1 → ARn
Если cond – истина и ARn ≥ 0:
Если src в регистровом режиме адресации (любой регистр из основного регистрового файла ЦПУ),
src → PC
Если src в режиме адресации относительно PC (метка или адрес),
смещение + PC + 1 → PC.
В противном случае – продолжение.

Описание:
DBcond означает стандартный переход, который выполняется за четыре цикла, т. к. конвейер должен быть очищен, если cond – «истина». Если условие истинно и заданный вспомогательный регистр больше или равен 0, выполняется декремент этого вспомогательного регистра и ветвление.
Содержимое вспомогательного регистра 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, содержимое указанного регистра загружается в PC. Если операнд src установлен в режим адресации относительно PC, Ассемблер генерирует смещение; смещение = метка – (PC инструкции перехода + 1). Смещение добавляется к PC инструкции перехода + 1 для генерации нового PC.
Ядро процессора 1867ВЦ8Ф1 поддерживает 20 кодов условий, которые могут быть использованы с этой инструкцией.

Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.

Циклы:
4

Пример:
DBLT AR3, R2

	Перед инструкцией		После инструкции
PC	5Fh	PC	9Fh
AR3	12h	AR3	11h
R2	9Fh	R2	9Fh
LUF	0	LUF	0
LV	0	LV	0
UF	0	UF	0
N	1	N	1
Z	0	Z	0
V	0	V	0
C	0	C	0

DBcondD инструкция

Синтаксис:

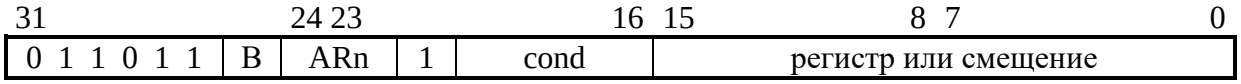
DbcondD ARn, src

Операнды:

src: режимы адресации условного перехода (B)

ARn: вспомогательный регистр

Код:



Поля слова:

B	src режимы адресации
0	Регистр
1	Относительно PC

Операция:

ARn – 1 → ARn

Если cond – истина и ARn ≥ 0:

Если src в регистровом режиме адресации (Rn, 0 ≤ n ≤ 27), src → PC

Если src в режиме адресации относительно PC (метка или адрес),

смещение + PC + 3 → PC.

В противном случае – продолжение.

Описание:

DBcondD означает задержанный переход, который выполняется за один цикл в результате позволяя осуществить выборку трёх инструкций до того, как изменится PC. Определённый вспомогательный регистр декрементируется, и выполняется переход, если cond – «истина», и, если значение в определённом вспомогательном регистре больше или равно 0. Для корректной работы DBcondD три инструкции, следующие за ней, не должны изменять разряды состояния, тестируемые в cond.

Вспомогательный регистр интерпретируется как 32-разрядное знаковое целое. Никакие три инструкции, следующие за DBcondD, не могут изменить ход 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 для задержанного перехода разряд 21 равен 1.

В ядре процессора 1867ВЦ8Ф1 имеется 20 кодов условий, которые могут быть использованы с этой инструкцией.

Разряды состояния:

LUF Не изменяется.

LV Не изменяется.

UF Не изменяется.

N Не изменяется.

Z Не изменяется.

V Не изменяется.

C Не изменяется.

Разряд режима:

OVM Операция не зависит от значения разряда OVM.

Циклы:

1

Пример:
DBZD AR5, $+110h

Перед инструкцией

PC	0h
AR5	67h
LUF	0
LV	0
UF	0
N	0
Z	1
V	0
C	0

После инструкции

PC	110h
AR5	66h
LUF	0
LV	0
UF	0
N	0
Z	1
V	0
C	0

FIX инструкция

Синтаксис:
FIX src, dst
Операнды:
src: основные режимы адресации (G)
dst: регистр (любой регистр в основном регистровом файле ЦПУ)
Код:

Поля слова:

G	src режимы адресации
00	Регистр (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 заполнен в направлении переполнения.
Разряды состояния:

LUF	Не изменяется.
LV	1 при возникновении целочисленного переполнения, в противном случае не изменяется.
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.

Разряд режима:
OVM операция не зависит от значения разряда OVM.

Циклы:
1
Пример:
FIX R1, R2

	Перед инструкцией		После инструкции		
R1	0A2820 0000h	1,3454e+3	R1	0A2820 0000h	1,3454e+3
R2	0h		R2	541h	1345
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	

FIX || STI инструкция

Синтаксис:
FIX src2, dst1
|| **STI** src3, dst2
Операнды:
src2 косвенная (смещение = 0, 1, IR0, IR1)
dst1 регистр (R0 – R7)
src3 регистр (R0 – R7)
dst2 косвенная (смещение = 0, 1, IR0, IR1)
Код:

31 29	24 23	16 15	8 7	0
1 1	0 1 1 1 0	dst1	0 0 0	src3	dst2	src2

Поля слова:
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, STI имеет на входе содержимое регистра до того, как оно было изменено инструкцией FIX. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.
Целочисленное переполнение возникает, кода число в формате с ПЗ слишком велико, чтобы быть представлено как 32-разрядное в дополнительном коде. В случае целочисленного переполнения результат будет заполнен в направлении переполнения.
Разряды состояния:

LUF	Не изменяется.
LV	1 при возникновении целочисленного переполнения, в противном случае не изменяется.
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.

Разряд режима:
OVM операция зависит от значения разряда OVM.

Циклы:
1
Пример:
```
FIX     *++AR4(1),R1
|| 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	
R1	0B3h	179
R0	0 DCh	220
AR2	80 983Ch	

Данные в 80 98A3h

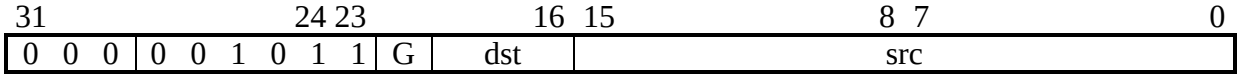
	733 C000h	1,79750e+02

Данные в 80 983Ch

	0DCh	220
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

FLOAT инструкция

Синтаксис:
FLOAT src, dst
Операнды:
src: основные режимы адресации (G)
dst: регистр (R0 – R11)
Код:



Поля слова:

G	src режимы адресации
00	Регистр (любой регистр из основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
float(src) → dst
Описание:
Целочисленный операнд src преобразуется в значение в формате с ПЗ, равное ему, и записывается в регистр dst. Операнд src является целым числом со знаком, операнд dst – число в формате с ПЗ.
Разряды состояния:

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.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
FLOAT *++AR2(2), R5

Перед инструкцией

AR2	80 9800h	
R5	034C 2000h	1,27578125e+01

Данные в 80 9802h

	0AEh	174
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

AR2	80 9802h	
R5	072E0 0000h	1,74e+02

Данные в 80 9802h

	0AEh	174
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

FLOAT || STF инструкция

Синтаксис:
FLOAT src2, dst1
|| **STF** src3, dst2
Операнды:
src2: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (смещение = 0, 1, IR0, IR1)
Код:

31 29	24 23	16	15 8	7 0
1 1 0 1 0 1 1	dst1	0 0 0 src3	dst2	src2

Поля слова:
Нет
Операция:
float (src2) → dst1
|| src3 → dst2
Описание:
Выполняется преобразование из целого 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, STF имеет на входе содержимое регистра до того, как оно было изменено инструкцией FLOAT.
Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.
Разряды состояния:

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.

Разряд режима:
OVM операция зависит от значения разряда OVM.
Циклы:
1
Пример:
```
FLOAT   *+AR2(IR0),R6
|| STF   R7,*AR1
```

Перед инструкцией

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	

После инструкции

AR2	80 98C5h	
IR0	8h	
R6	072E00 0000h	1,740e+02
R7	034C20 0000	1,27578125e+01
AR1	80 9933h	

Данные в 80 98CDh

	0AEh	174

Данные в 80 9933h

	034C 2000h	1,27578125e+01
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

FRIEEE инструкция

Синтаксис:
FRIEEE src, dst
Операнды:
src: прямая или косвенная адресация (G)
dst: регистр с повышенной точностью (R0 – R11)
Код:

31 … 24	23	22 … 16	15 … 8	7 … 0
0 0 0 1 1 1 0 0 0	G	dst		src

Поля слова:

G	src режимы адресации
01	Прямая
10	Косвенная

Операция:
Конвертация 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 берётся из памяти. Результат преобразования переносится в регистр с повышенной точностью, как число с плавающей запятой одинарной точности.
Разряды состояния:

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.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
Нет

FRIEEE || STF инструкция

Синтаксис:
FRIEEE src2, dst1
|| **STF** src3, dst2
Операнды:
src2: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (смещение = 0, 1, IR0, IR1)
Код:

31 29	24 23		16 15	8 7	0
1 1 1 1 0 0 1	dst1	0 0 0	src3	dst2	src2

Поля слова:
Нет
Операция:
Конвертированный из IEEE формата
src2 → dst1
параллельно с
src3 → dst2
Описание:
Операнд src2 преобразуется из формата с плавающей запятой 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 в формате с плавающей запятой.
Если src2 и dst2 находятся в одном месте, тогда чтение src2 предшествует записи в dst2.
Разряды состояния:
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.
Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
Нет

IACK инструкция

Синтаксис:
IACK src
Операнды:
src: основные режимы 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			

Поля слова:

G	src режимы адресации
01	Прямая
10	Косвенная

Операция:

Выполняет фиктивную операцию чтения с IACK# = 0.

По завершении фиктивного чтения устанавливает IACK# в 1.

Описание:

Выполняется фиктивная операция чтения с IACK# = 0. По завершении фиктивного чтения IACK# устанавливается в 1. Эта инструкция может быть использована для подтверждения внешнего прерывания. Если указанный адрес относится к внекристальной памяти, осуществляется операция чтения по данному адресу. Сигнал IACK# и адрес могут использоваться для сигнализации подтверждения прерывания внешним устройствам. Считанные данных процессором не используются. Заметим, что сигнал IACK# удлиняется при многоцикловом чтении.

Циклы:

1

Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:

OVM Операция не зависит от значения разряда 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

IDLE инструкция

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					

Поля слова:
Нет
Операция:
1 → ST(GIE)
Следующий PC → PC
Ожидание до прерывания
Описание:
Устанавливается глобальное разрешение прерывания, следующее значение PC загружается в PC и ЦПУ ожидает получения прерывания. Когда прерывание получено, содержимое PC загружается в активный системный стек, и процессор переходит к программе обработки прерываний.
Разряды состояния:

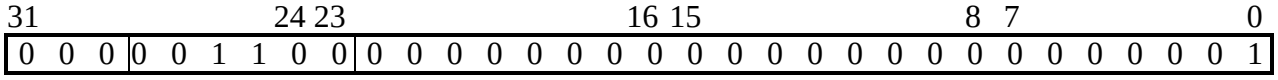
LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM Операция не зависит от значения разряда OVM.

Циклы:
1
Пример:
Нет

IDLE2 инструкция

Синтаксис:
-
Операнды:
Нет
Код:



Поля слова:
Нет
Операция:
1 → ST(GIE)
Следующий PC → PC
Ожидание до прерывания
Описание:
Инструкция IDLE2 выполняет те же функции, что и IDLE, дополнительно выключая функциональную синхронизацию внутренних устройств. Это позволяет обеспечить режим сверхнизкого энергопотребления. PC инкрементируется один раз, и устройство остаётся в состоянии ожидания, пока не произойдёт внешнее прерывание (NMI_x# или IIOF(0-3)_x#, где x = 1, 2).
В режиме IDLE2 ядро процессора 1867ВЦ8Ф1 ведёт себя следующим образом:
- ЦПУ, периферия и память сохраняют предыдущее состояние.
- Когда устройство находится в функциональном (неэмуляционном) режиме, синхронизация останавливается при высоком уровне на H1 и низком уровне на H3.

- Ядро процессора 1867ВЦ8Ф1 остаётся в IDLE2, пока не произойдёт одно из внешних прерываний (NMI_x# или IIOF(0-3)_x#, (где x = 1, 2) в течение как минимум последних двух циклов H1. Затем синхронизация стартует после задержки на 1 цикл H1. Такты начинают идти в фазе, противоположной той, в которой они были остановлены (H1 может стартовать в высоком уровне, если H3 был перед этим остановлен в высоком уровне, и наоборот). Однако, H1 и H3 остаются в противофазе по отношению друг к другу.
- В течение IDLE2, чтобы внешнее прерывание было определено и обслужено ЦПУ, оно должно произойти в течение последних двух циклов H1. Для того чтобы процессор распознал только одно прерывание после рестарта операции, сигнал прерывания должен быть сконфигурирован в режиме его определения по фронту или должен присутствовать не менее трёх циклов в режиме его определения по уровню.
- Когда ядро процессора 1867ВЦ8Ф1 находится в режиме эмуляции, H1 и H3 продолжают работать в нормальном режиме, и ЦПУ работает как при выполнении инструкции IDLE. Синхронизация продолжается для корректной работы эмулятора.
- Любой сигнал внешнего прерывания может вывести устройство из IDLE2; но для того, чтобы ЦПУ распознал это прерывание, оно должно быть разрешено. Если прерывание определено и обрабатывается ЦПУ, инструкция, следующая за инструкцией IDLE2, не выполняется до того, пока не произойдёт возврат.

Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:

OVM Операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

Нет

LAJ инструкция

Синтаксис:

LAJ src

Операнды:

src: режим относительной-PC адресации

Код:

31							24	23							16	15							8	7							0
0	1	1	0	0	0	1	1	смещение																						

Поля слова:

Нет

Операция:

PC LAJ+4 → регистр с повышенной точностью R11

смещение + 3 + PC LAJ → PC

Описание:
LAJ выполняет одноцикловый отложенный вызов подпрограммы, что позволяет трём инструкциям, следующим за LAJ инструкцией, быть выполненными перед ветвлением. Возвращённый адрес (адрес LAJ инструкции + 4) размещён в регистре повышенной точности R11. Ассемблер генерирует смещение: смещение = src – (PC инструкции ветвления + 1). Это смещение сохраняется как 24-разрядное знаковое целое в 24 наименее значащих битах инструкции ветвления. Это смещение добавляется к PC инструкции ветвления плюс 1 для генерации нового PC.
Никакие три инструкции, следующие за LAJ инструкцией, не должны изменять R11 или ход исполнения программы.
Прерывания запрещены в течение LAJ инструкции.
Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM Операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
Нет

LAJcond инструкция

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	регистр или смещение

Поля слова:

B	src режимы адресации
0	Регистр
1	Относительно PC

Операция:
Если (условие – «истина»)
Если (src – регистр)
PC LAJcond + 4 → регистр повышенной точности R11
src → PC
Если (src в режиме относительной-PC адресации)
PC LAJcond + 4 → регистр повышенной точности R11
смещение = src – (PC LAJ + 3)
смещение + PC LAJ + 3 → PC
Иначе, продолжение.

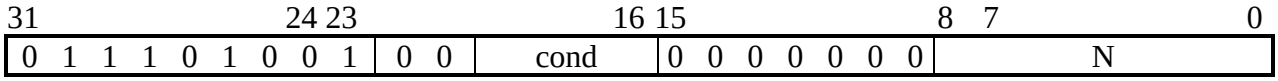
Описание:
LAJcond выполняет одноцикловый задержанный вызов подпрограммы по условию, что позволяет трём инструкциям, следующим за LAJcond инструкцией, быть выполненными перед ветвлением, без влияния cond. Возвращённый адрес (адрес LAJ инструкции + 4) размещён в регистре повышенной точности R11. Адрес ветвления формируется в регистровом режиме адресации или PC-относительной адресации.
Никакие три инструкции, следующие за LAJcond инструкцией, не должны изменять R11 или ход исполнения программы. Прерывания запрещены в течение LAJcond инструкции.
Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM Операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
Нет

LATcond инструкция

Синтаксис:
LATcond**N**
Операнды:
N режим непосредственной адресации по номеру TVT таблицы ($0 \le N \le 511$)
Код:



Поля слова:
Нет
Операция:
Если (условие – «истина»)
ST(GIE) → ST(PGIE)
ST(CF) → ST(PCF)
0 → ST(GIE)
1 → ST(CF)
PC LATcond + 4 → регистр повышенной точности R11 вектор N таблицы TVT → PC
Иначе, продолжение.
Описание:
LATcond выполняет задержанное одноцикловое программное прерывание по условию. Если условие – «истина», состояние бит GIE и CF сохраняется в битах PGIE и PCF статусного регистра. Все прерывания запрещаются (0 → GIE), и кэш «замораживается» (1 → CF). Содержимое PC LATcond + 4 помещается в R11, и в PC загружается адрес, указанный вектором N таблицы программных прерываний. Если условие – «ложь», продолжается выполнение программы. Если программные прерывания имеют вложенность, необходимо сохранить регистр состояния перед выполнением инструкции LATcondN.

Три инструкции, следующие за LATcond, выбираются и выполняются, но не влияют на cond. Они не должны изменять ход программы или непосредственно изменять регистр состояния. Прерывания запрещены в течение инструкции LATcond N.

Разряды состояния:

LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима:

OVM Операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

Нет

LBb инструкция

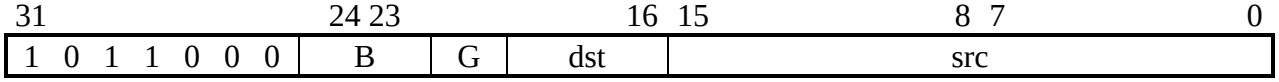
Синтаксис:

LBb src, dst

Операнды:

src: регистровая, прямая, 16-разрядная непосредственная, косвенная адресация
dst: режим регистра (любой регистр в основном регистровом файле ЦПУ)

Код:



Поля слова:

G	*src* режимы адресации	B	*src* байт
00	Регистр	00	Байт 0 (младший байт)
01	Прямая	01	Байт 1
10	Косвенная	10	Байт 2
11	Непосредственная (16 бит)	11	Байт 3 (старший байт)

Операция:

Байт со знаковым расширением (3, 2, 1, 0) src → dst
b = загружаемый байт (3, 2, 1, 0)

3	2	1	0	= **b** (указатель байта 3–0)

Описание:

Заданный байт операнда src расширяется на знак и сдвигается вправо в восемь младших разрядов dst регистра. Байт src знаковый. Когда установлен режим непосредственной адресации и выбраны байт 2 (B = 10) или байт 3 (B = 11), LBb инструкция производит расширение со знаком 16-разрядного значения. Следовательно, значение 00h или FFh сохраняется в восьми младших разрядах dst регистра.

Разряды состояния:

Если ST (SET COND) = 0, и регистр назначения – это R0 – R11, флаги состояний изменяются.
Если ST (SET COND) = 1, они изменяются для всех dst 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.

Разряд режима:
OVM Операция не зависит от значения разряда 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

LBUb инструкция

Синтаксис:
LBUb src, dst
Операнды:
src: регистровая, прямая, 16-разрядная непосредственная, косвенная адресация
dst: регистр (любой регистр в основном регистровом файле ЦПУ)
Код:

31	24	23	16	15	8	7	0
1 0 1 1 0 0 1	B	G	dst	src			

Поля слова:

G	*src* режимы адресации	B	*src* байт
00	Регистр	00	Байт 0 (младший байт)
01	Прямая	01	Байт 1
10	Косвенная	10	Байт 2
11	Непосредственная (16 бит)	11	Байт 3 (старший байт)

Операция:
Байт (3, 2, 1, 0) src → dst
b = загружаемый байт (3, 2, 1, 0)

3	2	1	0

= **b** (указатель байта 3–0)

Описание:
Заданный байт операнда src сдвигается вправо без расширения на знак в восемь младших разрядов dst регистра. Байт src беззнаковый. Когда установлен режим непосредственной адресации и выбраны байт 2 (B = 10) или байт 3 (B = 11), LBUb инструкция производит расширение со знаком 16-разрядного значения. Следовательно, значение 00h или FFh сохраняется в восьми младших разрядах dst регистра.
Разряды состояния:
Если ST (SET COND) = 0, и конечный регистр – это R0 – R11, флаги состояний изменяются.
Если ST (SET COND) = 1, они изменяются для всех результирующих регистров.

LUF Не изменяется.
LV Не изменяется.
UF 0
N 0.
Z 1, если сгенерирован нулевой результат, иначе 0
V 0.
C Не изменяется.

Разряд режима:
OVM Операция не зависит от значения разряда OVM.

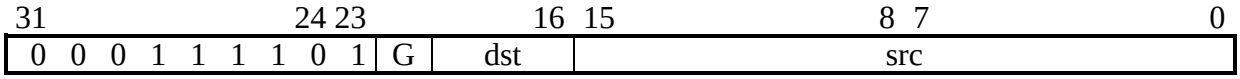
Циклы:
1
Пример:
LB2 R1, R2

	Перед инструкцией
R1	00AB 0000h
R2	0000 0000h

	После инструкции
R1	00AB 0000h
R2	0000 00ABh

LDA инструкция

Синтаксис:
LDA src, dst
Операнды:
src: основные режимы адресации (G)
dst: режим регистра (только адресные регистры)
Код:



Поля слова:

G	src режимы адресации
00	Регистр (любой регистр ЦПУ в основном регистровом файле)
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 с окончанием фазы чтения конвейера. Как результат, LDA – одноцикловая и, при загрузке этих регистров исполнятся быстрее, чем LDI (все операнды интерпретируются как знаковые целые).

Предупреждение

src и dst не могут быть одним и тем же регистром.

Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM Операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
Нет

LDE инструкция

Синтаксис:

LDE src, dst

Операнды:

src: основные режимы адресации (G)

dst: регистр (R0 – R11)

Код:

31 24	23 16	15 8	7 0
0 0 0 0 0 1 1 0 1	G	dst	src

Поля слова:

G	src режимы адресации
00	Регистр (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 значением экспоненты для нуля, что определяется точностью операнда src. Затем поле мантиссы операнда dst устанавливается в ноль. Предполагается, что операнды src и dst являются числами в формате с ПЗ. Непосредственные значения рассматриваются как числа в коротком формате с ПЗ.

Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

LDE R0, R5

	Перед инструкцией		После инструкции		
R0	020005 6F30h	4,00066337e+00	R0	020005 6F30h	4,00066337e+00
R5	0A056F E332h	1,06749648e+03	R5	02056F E332hh	4,16990814e+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	

LDEP инструкция

Синтаксис:

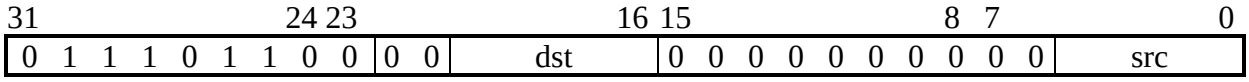
LDEP src, dst

Операнды:

src: расширенный регистровый файл (IVTP или TVTP)

dst: регистр (любой регистр в основном регистровом файле ЦПУ)

Код:



Поля слова:

Нет

Операция:

src(exp) → dst(exp)

Описание:

Инструкция LDEP загружает регистр ЦПУ содержимым IVTP регистра (указатель векторной таблицы системных прерываний) или TVTP регистра. Регистровый операнд src расширенного регистрового файла загружается в dst регистр в основном регистровом файле. Предполагается, что содержимое dst регистра является целым числом.

Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:

OVM Операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

Нет

LDF инструкция

Синтаксис:

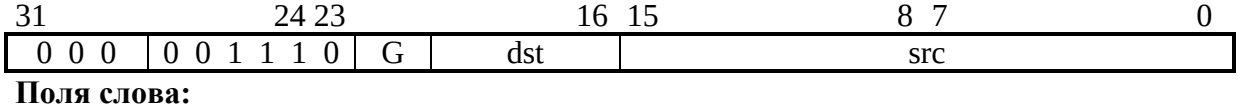
LDF src, dst

Операнды:

src: основные режимы адресации (G)

dst: регистр (R0 – R11)

Код:



Поля слова:

G	src режимы адресации
00	Регистр (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.

Разряды состояния:

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.

Разряд режима:
OVM Операция не зависит от значения разряда 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	010C52 A000h	2,19254303e+00
Данные в 80 9800h		
	10C5 2A00h	2,19254303e+00
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

LDFcond инструкция

Синтаксис:
LDFcond src, dst

Операнды:
src: основные режимы адресации (G)
dst: регистр (R0 – R11)

Код:

31	24	23	16	15	8	7	0
0 1 0 0	cond	G	dst	src			

Поля слова:

G	src режимы адресации
00	Регистр (R0 – R11)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
Если cond – «истина»:
src → dst.
Иначе:
dst не изменяется

Описание:
Если cond – «истина», операнд src загружается в регистр dst. В противном случае регистр dst не изменяется. Операнды src и dst являются числами в формате с ПЗ.
В ядре процессора 1867ВЦ8Ф1 имеется 20 кодов условий, которые могут быть использованы с этой инструкцией. Следует обратить внимание, что инструкция LDFU (загрузить ПЗ – число безусловно) используется для загрузки R0 – R11 независимо от состояния флагов условий.

Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM Операция не зависит от значения разряда OVM.

Циклы:
1

Пример:
LDFZ R3, R5

	Перед инструкцией		После инструкции		
R3	2CFF2C D500h	1,77055560e+13	R3	2CFF2C D500h	1,77055560e+13
R5	5F0000 003Eh	3,96140824e+28	R5	2CFF2C D500h	1,77055560e+13
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	1		Z	1	
V	0		V	0	
C	0		C	0	

LDFI инструкция

Синтаксис:
LDFI src, dst

Операнды:
src: основные режимы адресации src (G)
dst: регистр (R0 – R11)

Код:

31 29	24 23	21	16 15	0
0 0 0	0 0 1 1 1 1	G	dst	src

Поля слова:

G	src режимы адресации
01	Прямая
10	Косвенная

Операция:
Сигнализирует об операции блокировки.
src → dst

Описание:
Операнд src загружается в регистр dst. Операция блокировки сигнализируется через LOCK# или LLOCK#. Операнды src и dst являются числами в формате с ПЗ. Обратите внимание, что допускается только прямая и косвенная адресация.

Циклы:
1

Разряды состояния:

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.

Разряд режима:
OVM операция не зависит от значения разряда OVM.

Пример:
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	

LDF || LDF инструкция

Синтаксис:
LDF src2, dst2
|| **LDF** src1, dst1

Операнды:
src1: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src2: косвенная (смещение = 0, 1, IR0, IR1)
dst2: регистр (R0 – R7)

Код:

31 29	24 23	16	15 8	7 0
1 1 0 0 0 1 0	dst2	dst1	0 0 0	src1	src2

Поля слова:
Нет

Операция:
src2 → dst2
|| src1 → dst1

Описание:
Загрузка двух чисел выполняется параллельно. Если обе LDF загружают в один регистр, Ассемблер выдаёт предупреждение. В результате этого получается LDF src2, dst2.
Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Циклы:
1
Разряд режима:
OVM Операция не зависит от значения разряда OVM.
Пример:

```
LDF   *– –AR1(IR0), R7
|| LDF  *AR7++(1), R3
```

	Перед инструкцией		После инструкции	
AR1	80 985Fh		80 9857h	
IR0	8h		8h	
R7	0h		070C80 0000h	1,4050e+02
AR7	80 988Ah		80 988Bh	
R3	0h		057B40 0000h	6,281250e+01
Данные в 80 9857h			Данные в 80 9857h	
	70C 8000h	1,4050e+02	70C 8000h	1,4050e+02
Данные в 80 988Ah			Данные в 80 988Ah	
	57B 4000h	6,281250e+01	57B 4000h	6,281250e+01
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		0	
V	0		0	
C	0		0	

LDF || STF инструкция
Синтаксис:
LDF src2, src1
|| **STF** src3, dst2
Операнды:
src2: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (смещение = 0, 1, IR0, IR1)
Код:

31 29	24 23	16 15	8 7	0
1 1	0 1 1 0 0 dst1	0 0 0 src3	dst2	src2

Поля слова:
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, src2 считывается до того, как будет записан dst2.

Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Циклы:
1

Разряд режима:
OVM операция не зависит от значения разряда OVM.

Пример:

```
LDF   *AR2– – (1), R1
|| STF  R3,*AR4++(IR1)
```

Перед инструкцией			После инструкции		
AR2	80 98E6h		AR2	80 98E6h	
R1	0h		R1	070C80 0000h	1,4050e+02
R3	057B40 0000h	6,28125e+01	R3	057B40 0000h	6,28125e+01
AR4	80 9900h		AR4	80 9910h	
IR1	10h		IR1	10h	
Данные в 80 98E7h			Данные в 80 98E7h		
	70C 4000h	1,4050e+02		70C 4000h	1,4050e+02
Данные в 80 9900h			Данные в 80 9900h		
	0h			57B 4000h	6,28125e+01
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	

LDHI инструкция

Синтаксис:
LDHI src, dst

Операнды:
src: 16-разрядная беззнаковая непосредственная адресация
dst: регистровый режим

Код:

31 … 24	23 … 16	15 … 8	7 … 0
0 0 0 1 1 1 1 1 1 1 1	1 1 dst	src (непосредственное значение)	

Поля слова:
Нет

Операция:
src → 16 старших значащих битов dst

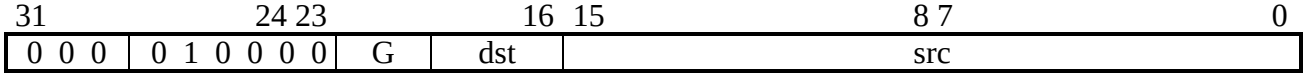
Описание:
16-битное беззнаковое непосредственное src загружается в 16 старших разрядов регистра dst, 0 загружается 16 младших разрядов. Значение регистра dst является целым.
Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM Операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
LDHI 44h, R2

	Перед инструкцией		После инструкции
R2	ABCD EF12h	R2	0044 0000h

LDI инструкция
Синтаксис:
LDI src, dst
Операнды:
src: основные режимы адресации (G)
dst: регистр (любой регистр в основном регистровом файле ЦПУ)
Код:



Поля слова:

G	src режимы адресации
00	Регистр (любой регистр ЦПУ в основном регистровом файле)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
src → dst
Описание:
Операнд src загружается в регистр dst. Операнды src и dst являются целыми со знаком.
Разряды состояния:

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.

Разряд режима:
OVM Операция не зависит от значения разряда OVM.
Циклы:
1

Пример:

LDI *–AR1(IR0), R5

Перед инструкцией

AR1	2Ch
IR0	5h
R5	3C5h

Данные в 27h

	26h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

965 38

После инструкции

AR1	2Ch
IR0	5h
R5	26h

Данные в 27h

	26h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

38 38

LDIcond инструкция

Синтаксис:

LDIcond src, dst

Операнды:

src: основные режимы адресации (G)

dst: регистр (любой регистр в основном регистровом файле ЦПУ)

Код:

Поля слова:

G	src режимы адресации
00	Регистр (любой регистр в основном регистровом файле ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:

Если cond – «истина»:

src → dst

Иначе:

dst не изменяется

Описание:

Если cond – «истина», 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 данных (DP) или другой регистр с 16 старшими разрядами перемещаемого адреса.

В процессорном ядре 1867ВЦ8Ф1 предусмотрены 20 кодов условий, которые могут быть использованы с этой инструкцией. Обратите внимание, что инструкция LDIU (безусловная загрузка целого) используется для загрузки выбранного регистра ЦПУ без изменения флагов условий в то время, как инструкция LDI влияет на них.

Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:

OVM Операция зависит от значения разряда OVM.

Циклы:

1

Пример:

LDIZ R4,R6

	Перед инструкцией			После инструкции	
R4	027Ch	636	R4	027Ch	636
R6	0FE2h	4066	R6	0FE2h	4066
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	

LDII инструкция

Синтаксис:

LDII src, dst

Операнды:

src: основные режимы адресации (G)

dst: регистр (любой регистр ЦПУ в основном регистровом файле)

Код:

31 29	24 23	21	16 15	0
0 0 0	0 1 0 0 0 1	G	dst	src

Поля слова:

G	src режимы адресации
01	Прямая
10	Косвенная

Операция:

Сигнализирует об операции блокировки.

src → dst

Описание:

Операнд src загружается в регистр dst. Операция блокировки сигнализируется через LOCK# или LLOCK#. Операнды src и dst являются целыми со знаком. Необходимо обратить внимание, что определена только прямая и косвенная адресация.

Разряды состояния:

Если ST(SET COND) = 0, флаги состояния изменяются только если регистром назначения является один из R0 – R11. Если ST(SET COND) = 1, они изменяются для всех регистров dst.

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.

Разряд режима:
OVM Операция не зависит от значения разряда 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

LDI || LDI инструкция

Синтаксис:
LDI src2, dst2
|| **LDI** src1, dst1

Операнды:
src1: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src2: косвенная (смещение = 0, 1, IR0, IR1)
dst2: регистр (R0 – R7)

Код:

31 29	24 23	16	15 8	7 0
1 1	0 0 0 1 1 dst2	dst1	0 0 0 src1	src2

Поля слова:
Нет

Операция:
src2 → dst2
|| src1 → dst1

Описание:
Загрузка двух целых выполняется параллельно. Если обе LDI загружают в один регистр, Ассемблер выдаёт предупреждение. В результате этого получается LDI src2, dst2.

Разряды состояния:
LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
```
LDI     *– –AR1(1), R7
|| LDI   *AR7++(IR0), R1
```

Перед инструкцией			После инструкции		
AR1	80 9826h		AR1	80 9826h	
R7	0h		R7	0FAh	250
AR7	80 98C8h		AR7	80 98C8h	
IR0	10h		IR0	10h	
R1	0h		R1	2EEh	750
Данные в 80 9825h			Данные в 80 9825h		
	0FAh	250		0FAh	250
Данные в 80 98C8			Данные в 80 98C48		
	2EEh	750		2EEh	750
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	

LDI || STI инструкция
Синтаксис:
LDI src2, src1
|| **STI** src3, dst2
Операнды:
src2: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (смещение = 0, 1, IR0, IR1)
Код:

31 29	24 23	16	15 8	7 0
1 1 0 1 1 0 1	dst1	0 0 0 src3	dst2	src2

Поля слова:
Нет
Операция:
src2 → dst1
|| src3 → dst2
Описание:
Параллельно выполняются операции загрузки и сохранения целых. Если операнды src2 и dst2 указывают на один и тот же адрес, src2 считывается до того, как будет записан dst2.
Разряды состояния:
LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.

N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.

Циклы:
1

Пример:
```
LDI    *–AR1(1),R2
|| STI  R7,*AR5++(IR0)
```

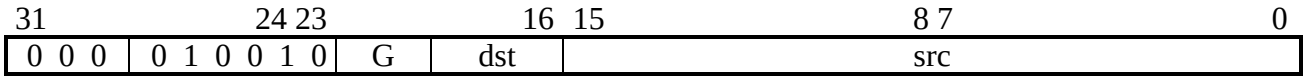
Перед инструкцией			После инструкции		
AR1	80 98E7h		AR1	80 98E7h	
R2	0h		R2	0DCh	220
R7	35h	53	R7	35h	53
AR5	80 982Ch		AR5	80 9834h	
IR0	8h		IR0	8h	
Данные в 80 98E6h			Данные в 80 98E6h		
	0DCh	220		0DCh	220
Данные в 80 982Ch			Данные в 80 982Ch		
	0h			35h	53
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	0		C	0	

LDM инструкция

Синтаксис:
LDM src, dst

Операнды:
src: основные режимы адресации (G)
dst: регистр (R0 – R11)

Код:



Поля слова:

G	src режимы адресации
00	Регистр (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 инструкции устанавливаются в 0 языком Ассемблер. Если src в памяти, 32-разрядные данные загружаются в поле мантиссы.

Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда 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

R2	00 1CC0 0000h	1,22460938e+00
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

LDP инструкция

Синтаксис:
LDP src[, DP]
Операнды:
src: 16 старших разрядов абсолютного 32-разрядного адреса источника (src).
dst: опционально (указатель страница – данные, если «,DP» слева от операнда).
Код:

31	24	23	16	15	8	7	0
0 1 0 1	0 0 0 0 0	1 1	1 0 0 0 0	src			

Поля слова:
Нет
Операция:
src → Указатель страницы данных
Описание:
Эта псевдооперация является альтернативной формой инструкции LDIU, за исключением того, что инструкция 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) загружены 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 K страниц, каждая из которых размером 64 K слов. Разряды 31 – 16 указателя зарезервированы и хранятся как 0.
Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.

Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.

Циклы:
1

Пример:
LDP @809900h, DP
или
LDP @809900h

Перед инструкцией

DP	6465h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

DP	0080h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

16 старших значащих бит из 32 src, расширенный нулями

LDPE инструкция

Синтаксис:
LDPE src, dst

Операнды:
src: регистровый режим (любой регистр в основном регистровом файле ЦПУ)
dst: расширенный регистровый файл (IVTP или TVTP)

Код:

31 … 24	23 … 16	15 … 6	5 … 0
0 1 1 1 0 1 1 0 1	0 0 dst	0 0 0 0 0 0 0 0 0 0	src

Поля слова:
Нет

Операция:
src → dst

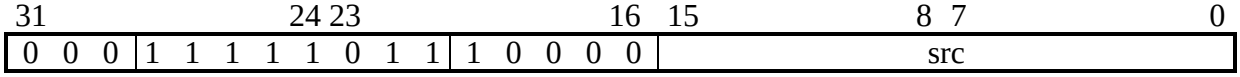
Описание:
Это средство загрузки регистра указателя векторной таблицы системных прерываний (IVTP) или регистра указателя векторной таблицы программных прерываний (TVTP).
Регистровый операнд src загружается из основного регистрового файла в dst регистр в расширенном регистровом файле. Операнд dst должен быть целым.

Разряды состояния:
LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
LDPE AR0, TVTP ; установка указателя векторной таблицы программных прерываний

LDPK инструкция

Синтаксис:
LDPK src, dst
Операнды:
src: 16-разрядная беззнаковая непосредственная адресация
Код:



Поля слова:
Нет
Операция:
src → DP
Описание:
16-битное беззнаковое непосредственное значение загружается в DP регистр. Эта операция завершается окончанием фазы декодирования LDPK инструкции; таким образом, загруженное значение готово для следующих инструкций для непосредственной адресации. Используйте её осторожно, когда используете DP регистр в инструкции, которая предшествует LDPK.
Например:

```
PUSH  DP
LDPK  new_value
```

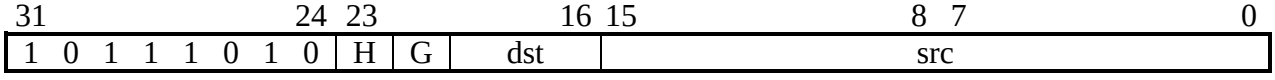
Загрузка нового значения DP в стек вместо сохранения предыдущего значения.
Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
Нет

LHw инструкция

Синтаксис:
LHw src, dst
Операнды:
src: регистровая, прямая, 16-разрядная непосредственная или косвенная адресация
dst: регистровый режим (любой регистр ЦПУ в основном регистровом файле)
Код:



Поля слова:

G	src режимы адресации
00	Регистр (любой из регистров ЦПУ в основном регистровом файле)
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 – дополненный знаком и смещённый вправо на 16 младших разрядов dst регистр. Половина слова src расширена знаком. Когда задан режим непосредственной адресации и выбрана половина слова 1 (H = 1), 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.

Разряды состояния:

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	1, если получен отрицательный результат, иначе 0.
Z	1, если получен нулевой результат, иначе 0.
V	0.
C	Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

LH0 R1, R2

	Перед инструкцией
R1	ABCD EF12h
R2	1234 5678h

	После инструкции
R1	ABCD EF12h
R2	FFFF EF12h

LHUw инструкция

Синтаксис:

LHUw src, dst

Операнды:

src: регистровая, прямая, 16-разрядная непосредственная или косвенная адресация

dst: регистровый режим (любой регистр ЦПУ в основном регистровом файле).

Код:

31							24	23	16	15	8	7	0
1	0	1	1	1	0	1	1	H	G	dst		src	

Поля слова:

G	src режимы адресации
00	Регистр (любой из регистров ЦПУ в основном регистровом файле)
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 – беззнаковый и смещённый вправо на 16 младших разрядов dst регистр. Половина слова src беззнаковая. Когда определён режим непосредственной адресации и выбрана половина слова 1 (H = 1), 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.

Разряды состояния:

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	0.
Z	1, если сгенерирован нулевой результат, иначе 0.
V	0.
C	Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

LHU0 R1, R2

Перед инструкцией

R1	ABCD EF12h
R2	1234 5678h

После инструкции

R1	ABCD EF12h
R2	0000 EF12h

LSH инструкция

Синтаксис:

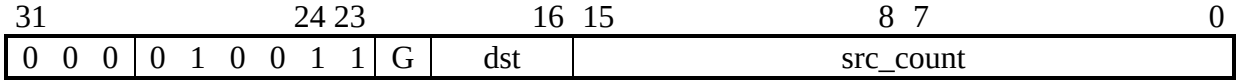
LSH src_count, dst

Операнды:

src_count: основные режимы адресации (G)

dst: регистровый режим (любой регистр основного регистрового файла ЦПУ)

Код:



Поля слова:

G	src режимы адресации
00	Регистр (любой регистр основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:

Логический сдвиг:

count = 7 младших разрядов src_count

Если count ≥ 0:

dst << count → dst

Иначе:

dst >> |count| → dst

Описание:

Семь младших разрядов операнда count используются для генерации двоичного дополнения величины сдвига. Если операнд count больше 0, операнд dst сдвигается влево на количество разрядов, определённое 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, dst сдвигается вправо на число разрядов, определённое 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, сдвиг не происходит и разряд переноса C устанавливается в 0.

Если операнд count больше, чем 32, C (перенос) бит получает значение наименее значащего бита. Если count меньше, чем 32, бит C устанавливается в 0.

Предполагается, что операнд count является целым со знаком, а оператор dst – беззнаковое целое.

Разряды состояния:

Если ST (SET COND) = 0, флаги состояния изменяются только в том случае, 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 разряд выходного значения.
Z	1 при генерации нулевого результата, иначе 0.
V	0.
C	Устанавливается по значению последнего сдвинутого вовне разряда.

Разряд режима:

OVM Операция не зависит от значения разряда OVM.

Циклы:
1
Пример 1:
LSH R4, R7

Перед инструкцией

R4	018h
R7	02ACh
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

24

После инструкции

R4	018h
R7	0AC00 0000h
LUF	0
LV	0
UF	0
N	1
Z	0
V	1
C	0

24

Пример 2:
LSH *–AR5(IR0), R5

Перед инструкцией

AR5	80 9908h
IR0	4h
R5	00 12C0 0000h

Данные в 80 9904h

	0FFF FFFF4 h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

– 12

После инструкции

AR5	80 9908h
IR0	4h
R5	00 0001 2C00h

Данные в 80 9904h

	0FFF FFFF4h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

– 12

LSH3 инструкция

Синтаксис:
LSH3 count, src, dst
Операнды:
src, src_count: тип 1 или тип 2 трёхоперандного адресного режима
dst: регистровый режим (любой регистр в основном регистровом файле ЦПУ)
Код:
Тип 1

31 … 24	23	22 … 16	15 … 8	7 … 0
0 0 1 0 0 1 0 0 0	T	dst	src	src_count

Тип 2

31 … 24	23	22 … 16	15 … 8	7 … 0
0 0 1 1 0 1 0 0 0	T	dst	src	src_count

Поля слова:
Тип 1

T	src режимы адресации	src_count режимы адресации
00	Регистр (любой регистр ЦПУ)	Регистр (любой регистр ЦПУ)
01	Косвенная (смещение = 0, 1, IR0, IR1)	Регистр (любой регистр ЦПУ)
10	Регистр (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	src режимы адресации	src_count режимы адресации
01	Регистр (любой регистр ЦПУ)	8-битная знаковая непосредственная
01	Регистр (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-битная знаковая непосредственная
11	Косвенная *+ARn (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:
Логический сдвиг, три операнда:
count = 7 младших разрядов src_count
Если count ≥ 0:
    src << count → dst
Иначе:
    src >> |count| → dst

Описание:
Семь младших разрядов операнда count используются для генерации двоичного дополнения счётчика сдвига. Если операнд count больше 0, копия операнда src сдвигается влево на число разрядов, определённое значением операнда count, и результат записывается в dst (src не изменяется). 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, src сдвигается вправо на число разрядов, определённое 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, 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.

Разряды состояния:
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. 0 для счётчика сдвига, равного 0.

Разряд режима:
OVM операция не зависит от значения разряда OVM.

Циклы:
1

Пример:
Нет

LSH3 || STI инструкция

Синтаксис:

LSH3 src_count, src2, dst1
|| **STI** src3, dst2

Операнды:

src_count: регистр (R0 – R7)
src2: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (смещение = 0, 1, IR0, IR1)

Код:

31 24	23 16	15 8	7 0
1 1 0 1 1 1 0	dst1	src_count	src3	dst2	src2

Поля слова:

Нет

Операция:

count = 7 младших разрядов src_count
Если count ≥ 0:
src2 << count → dst1
Иначе:
src2 >> |count| → dst1
|| src3 → dst2

Описание:

Семь младших разрядов операнда count используются для генерации двоичного дополнения счётчика сдвига.

Если операнд count больше 0, копия операнда src2 сдвигается влево на число разрядов, определённое 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 ← dst2 ← 0

Если операнд count меньше 0, dst сдвигается вправо на число разрядов, определённое 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 → dst2 → C

Если операнд count = 0, 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, 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, src2 считывается до записи в dst2.

Разряды состояния:

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. 0 для счётчика сдвига, равного 0.

Разряд режима:
OVM Операция зависит от значения разряда OVM.
Циклы:
1
Пример 1:
```
LSH3  R2,*++AR3(1),R0
|| STI   R4,*–AR5
```

Перед инструкцией

R2	18h	24
AR3	8098C2h	
R0	0h	
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

После инструкции

R2	18h	24
AR3	8098C3h	
R0	0AC00 0000h	
R4	0DCh	220
AR5	80 98A3h	

Данные в 80 98C3h

0ACh

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	
R0	12Ch	300
AR0	80 98B7h	

Данные в 80 9863h

2C00 0000h

Данные в 80 98B8h

	0h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

R7	0FFFFF FF4h	–12
AR2	80 9862h	
R2	2C000h	
R0	12Ch	300
AR0	80 98B7h	

Данные в 80 9863h

2C00 0000h

Данные в 80 98B9h

	12Ch	300
LUF	0	
LV	0	
UF	0	
N	1	
Z	0	
V	0	
C	0	

LWLct инструкция

Синтаксис:

LWLct src, dst

Операнды:

ct: число байтов {0, 1, 2 или 3} для сдвига влево (ct × 8 = величина сдвига в битах)

src: регистровая, прямая, 16-разрядная непосредственная или косвенная адресация

dst: регистровый режим (любой регистр в основном регистровом файле ЦПУ)

Код:

31							24	23			16	15			8	7		0
1	0	1	1	0	1	0	B		G		dst					src		

Поля слова:

G	src режимы адресации
00	Регистр (любой регистр основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

B	src байт
00	Нет сдвига
01	Сдвиг влево на 1 байт
10	Сдвиг влево на 2 байта
11	Сдвиг влево на 3 байта

Операция:

src << {0, 1, 2 или 3} байт и объединение с dst → dst

Описание:

Src операнд сдвигается влево на заданное число байтов и объединяется с байтами dst регистра, которые находятся ниже сдвинутого влево младшего разряда src операнда. При выборе режима непосредственной адресации, эта инструкция выполняет знаковое расширение 16-разрядного непосредственного значения до 32-разрядного значения, затем это 32-разрядное значение сдвигается и объединяется.

Разряды состояния:

Если ST(SET COND) = 0 и конечные регистры – это R0 – R11, флаги состояний изменяются.
Если ST(SET COND) = 1, они изменяются для всех регистров назначения.

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.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

LWL2 R1, R2

	Перед инструкцией
R1	ABCD EF12h
R2	1234 5678h

	После инструкции
R1	ABCD EF12h
R2	EF12 5678h

LWRct инструкция

Синтаксис:

LWRct src, dst

Операнды:

ct: число байтов {0, 1, 2 или 3} для сдвига вправо (ct × 8 = величина сдвига в битах)

src: регистровая, прямая, 16-разрядная непосредственная или косвенная адресация

dst: регистровый режим (любой регистр в основном регистровом файле ЦПУ)

Код:

31							24	23		16	15		8	7		0
1	0	1	1	0	1	1	B		G	dst				src		

Поля слова:

G	src режимы адресации
00	Регистр (любой регистр основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

B	src байт
00	Нет сдвига
01	Сдвиг вправо на 1 байт
10	Сдвиг вправо на 2 байта
11	Сдвиг вправо на 3 байта

Операция:

src >> {0, 1, 2 или 3} байт и объединение с dst → dst

Описание:

Src операнд сдвигается вправо на заданное число байтов и объединяется с байтами dst регистра, которые находятся выше сдвинутого вправо старшего разряда src операнда. Расширение на знак не выполняется. При выборе режима непосредственной адресации, эта инструкция выполняет знаковое расширение 16-разрядного непосредственного значения до 32-разрядного значения, затем это 32-разрядное значение сдвигается и объединяется.

Разряды состояния:

Если ST(SET COND) = 0 и конечные регистры – это R0 – R11, флаги состояний изменяются.
Если ST(SET COND) = 1, они изменяются для всех регистров назначения.

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.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

LWR1 AR1, R2

Перед инструкцией

AR1	ABCD EF12h
R2	1234 5678h

После инструкции

AR1	ABCD EF12h
R2	12AB CDEFh

MBct инструкция

Синтаксис:

MBct src, dst

Операнды:

ct: число байтов {0, 1, 2 или 3} для сдвига влево (ct × 8 = величина сдвига в битах)

src: регистровая, прямая или косвенная адресация

dst: регистровый режим (любой регистр в основном регистровом файле ЦПУ)

Код:

31 24	23 16	15 8	7 0
1 0 1 1 1 0 0 B	G	dst	src

Поля слова:

G	src режимы адресации
00	Регистр (любой регистр основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

B	src байт
00	Нет сдвига
01	Сдвиг влево на 1 байт
10	Сдвиг влево на 2 байта
11	Сдвиг влево на 3 байта

Операция:

Восемь младших разрядов src << {0, 1, 2 или 3} байт и объединение с dst → dst

Описание:

Восемь младших разрядов src операнда сдвигаются влево на заданное число байтов и объединяются с байтами dst регистра, которые находятся ниже сдвинутого влево младшего разряда src операнда. При выборе режима непосредственной адресации, эта инструкция выполняет знаковое расширение 16-разрядного непосредственного значения до 32-разрядного значения, затем это 32-разрядное значение сдвигается и объединяется.

Разряды состояния:

Если ST (SET COND) = 0 и конечные регистры – это R0 – R11, флаги состояний изменяются.
Если ST (SET COND) = 1, они изменяются для всех регистров назначения.

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.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

MB2 AR1, AR2

	Перед инструкцией			После инструкции
AR1	ABCD EF12h	(0012 0000h)	AR1	ABCD EF12h
AR2	1234 5678h		AR2	1212 5678h

MHct инструкция

Синтаксис:

MHct src, dst

Операнды:

ct: коэффициент сдвига половины слова (16 бит)

src: регистровая, прямая, 16-разрядная непосредственная или косвенная адресация

dst: режим регистра (любой регистр в основном регистровом файле ЦПУ)

Код:

31 24	23	16	15 8	7 0
1 0 1 1 1 1 0 0	H	G	dst	src

Поля слова:

G	src режимы адресации
00	Регистр (любой регистр основного регистрового файла ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

H	src половина слова
0	Половина слова 0 (младшее полуслово)
1	Половина слова 1 (старшее полуслово)

Операция:

16 младших разрядов src << {0, 1} половина слова и объединение с dst → dst

Описание:

При H = 1 младшие 16 разрядов операнда src сдвигаются влево 16 бит (на половину слова) и объединяются с байтами dst регистра, которые находятся ниже сдвинутого влево младшего разряда src операнда. При H = 0 младшие разряды src не сдвигаясь замещают младшие 16 разрядов dst. При выборе режима непосредственной адресации, эта инструкция выполняет знаковое расширение 16-разрядного непосредственного значения до 32-разрядного значения, затем это 32-разрядное значение сдвигается и объединяется.

Разряды состояния:

Если ST(SET COND) = 0 и конечные регистры – это R0 – R11, флаги состояний изменяются.
Если ST(SET COND) = 1, они изменяются для всех регистров назначения.

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.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

MH1 AR1, AR2

Перед инструкцией			После инструкции	
AR1	ABCD EF12h	(EF12 0000h)	AR1	ABCD EF12h
AR2	1234 5678h		AR2	EF12 5678h

MPYF инструкция

Синтаксис:

MPYF src, dst

Операнды:

src: основные режимы адресации src (G)

dst: регистр (R0 – R11)

Код:

31 29	24 23	21	16 15	0
0 0 0	0 1 0 1 0 0	G	dst	src

Поля слова:

G	src режимы адресации
00	Регистр (R0 – R11)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:

dst × src → dst

Описание:

Произведение операндов dst и src загружается в регистр dst. Операнды src (в регистровом режиме (R0 – R11)) и dst являются числами с ПЗ в формате расширенной точностью. Для нерегистрового режима src принимает вид числа с плавающей запятой с одинарной точностью.

Разряды состояния:

LUF 1 при потере значимости результата с ПЗ, иначе не изменяется.

LV 1 при возникновении переполнения с ПЗ, в противном случае 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 1 при возникновении переполнения с ПЗ, иначе 0.

C Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

MPYF R0, R2

	Перед инструкцией		После инструкции	
R0	07 0C80 0000h	1,4050e + 02	07 0C80 0000h	1,4050e + 02
R2	03 4C20 0000h	1,27578125e + 01	0A 600F 2000h	1,79247266e + 03
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		0	
V	0		0	
C	0		0	

MPYF3 инструкция

Синтаксис:

MPYF3 src2, src1, dst

Операнды:

src1, src2: первого или второго типа трёхоперандного режима адресации

Код:

Тип 1

31 … 24	23 … 16	15 … 8	7 … 0
0 0 1 0 0 1 0 0 1 T	dst	src1	src2

Тип 2

31 … 24	23 … 16	15 … 8	7 … 0
0 0 1 1 0 1 0 0 1 T	dst	src1	src2

Поля слова:

Тип 1

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (R0 – R11)	Регистр (R0 – R11)
01	Косвенная (смещение = 0, 1, IR0, IR1)	Регистр (любой регистр ЦПУ)
10	Регистр (R0 – R11)	Косвенная (смещение = 0, 1, IR0, IR1)
11	Косвенная (смещение = 0, 1, IR0, IR1)	Косвенная (смещение = 0, 1, IR0, IR1)

Тип 2

T	src1 режимы адресации	src2 режимы адресации
01	Регистр (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, 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, предполагается, что они являются числами с плавающей запятой одинарной точности.

Разряды состояния:

LUF 1 при потере значимости результата с ПЗ, иначе не изменяется.
LV 1 при возникновении ПЗ-переполнения, в противном случае 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 1 при возникновении ПЗ-переполнения, иначе 0.
C Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

Нет

MPYF3 || ADDF3 инструкция

Синтаксис:

MPYF3 srcA, srcB, dst1
|| **ADDF3** srcC, srcD, dst2

Операнды:

srcA, **srcB**, **srcC**, **srcD** — Любые два должны быть косвенными (смещение = 0, 1, IR0, IR1) и любые два должны быть регистрами (R0 – R7)

dst1 регистр (d1):
0 = R0
1 = R1

dst2 регистр (d2):
0 = R2
1 = R3

src1 регистр (R0 – R7)

src2 регистр (R0 – R7)

src3 косвенная (смещение = 0, 1, IR0, IR1)

src4 косвенная (смещение = 0, 1, IR0, IR1)

P Параллельные режимы 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

Поля слова:

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, MPYF3 имеет на входе содержимое регистра до того, как оно было изменено инструкцией ADDF3.

Для четырёх 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 задано, изменить порядок операндов в коммуникативных операциях для упрощения процесса.

Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Разряды состояния:

LUF 1 при потере значимости результата с ПЗ, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не изменяется.

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.

Разряд режима:
OVM операция не зависит от значения разряда OVM.

Циклы:
1

Пример:
```
MPYF3     *AR5++(1), * – –AR1(IR0), R0
|| ADDF3   R5, R7, R3
```

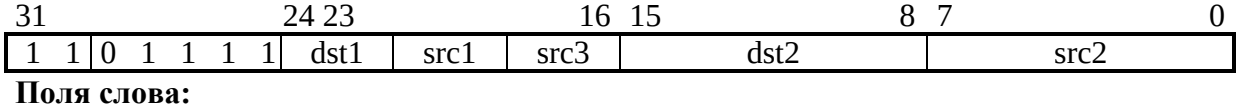
	Перед инструкцией			После инструкции	
AR5	80 98C5h		AR5	80 98C6h	
AR1	80 98A8h		AR1	80 98A4h	
IR0	4h		IR0	4h	
R0	0h		R0	04 6718 000h	2,88867188e+01
R5	07 33C0 0000h	1,79750e+02	R5	07 33C0 0000h	1,79750e+02
R7	07 0C80 0000h	1,4050e+02	R7	07 0C80 0000h	1,4050e+02
R3	0h		R3	08 2020 0000h	3,20250e+02
Данные в 80 98C5h			Данные в 80 98C5h		
	34C 0000h	1,1250e+01		34C 0000h	1,1250e+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	

MPYF3 || STF инструкция

Синтаксис:
MPYF3 src2, src1, dst
|| **STF** src3, dst2

Операнды:
src1: регистр (R0 – R7)
src2: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (смещение = 0, 1, IR0, IR1)

Код:



Поля слова:
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 регистра, STF имеет на входе содержимое регистра до того, как оно было изменено инструкцией MPYF3.
Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Разряды состояния:

LUF	1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	1 при возникновении ПЗ-переполнения, в противном случае 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	1 при возникновении ПЗ-переполнения, иначе 0.
C	Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.

Циклы:
1

Пример:
```
MPYF3       *–AR2(1), R7, R0
|| STF   R3, *AR0– –(IR0)
```

Перед инструкцией			После инструкции		
AR2	80 982Bh		AR2	80 982Bh	
R7	05 7B40 0000h	6,281250e+01	R7	05 7B40 0000h	6,281250e+01
R0	0h		R0	0D 09E4 A000h	8,82515625e+03
R3	08 6B28 0000h	4,7031250e+02	R3	08 6B28 0000h	4,7031250e+02
AR0	80 9860h		AR0	80 9858h	
IR0	8h		IR0	8h	
Данные в 80 982Ah			Данные в 80 982Ah		
	70C 8000h	1,4050e+02		70C 8000h	1,4050e+02
Данные в 80 9860h			Данные в 80 9860h		
	0h			86B28 0000h	4,7031250e+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	

MPYF3 || SUBF3 инструкция

Синтаксис:

MPYF3 srcA, srcB, dst1
|| SUBF3 srcC, srcD, dst2

Операнды:

srcA, **srcB**, **srcC**, **srcD** — Любые два должны быть косвенными (смещение = 0, 1, IR0, IR1) и любые два должны быть регистрами (R0 – R7)

dst1 регистр (d1):
0 = R0
1 = R1

dst2 регистр (d2):
0 = R2
1 = R3

src1 регистр (R0 – R7)
src2 регистр (R0 – R7)
src3 косвенная (смещение = 0, 1, IR0, IR1)
src4 косвенная (смещение = 0, 1, IR0, IR1)

P Параллельные режимы 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 1 | P | d1 | d2	src1	src2	src3 | src4

Поля слова:

Нет

Операция:

srcA × srcB → dst1
|| srcC – srcD → 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, MPYF3 имеет на входе содержимое регистра до того, как оно было изменено инструкцией SUBF3.

Для четырёх 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 задано, изменить порядок операндов в коммуникативных операциях для упрощения процедуры.

Разряды состояния:

LUF 1 при потере значимости результата с ПЗ, иначе не изменяется.
LV 1 при возникновении ПЗ-переполнения, в противном случае не изменяется.
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.

Разряд режима:
OVM операция зависит от значения разряда OVM.

Циклы:
1

Пример:
```
MPYF3     R5, *++AR7(IR1), R0
|| SUBF3   R7, *AR3– – (1), R2
```
или
```
MPYF3     *++AR7(IR1), R5, R0
|| SUBF3   R7, *AR3– – (1), R2
```

Перед инструкцией

R5	03 4C00 0000h	1,2750e+01
AR7	80 9904h	
IR1	8h	
R0	0h	
R7	07 33C0 0000h	1,79750e+02
AR3	80 98B2h	
R2	0h	

Данные в 80 990Ch

	111 0000h	2,250e+00

Данные в 80 98B2h

	70C 8000h	1,4050e+02
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

R5	03 4C00 0000h	1,2750e+01
AR7	80 990Ch	
IR1	8h	
R0	04 6718 0000h	2,88867188e+01
R7	07 33C0 0000h	1,79750e+02
AR3	80 98B1h	
R2	05 E300 0000h	–3,9250e+01

Данные в 80 990Ch

	111 0000h	2,250e+00

Данные в 80 98B2h

	70C 8000h	1,4050e+02
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

MPYI инструкция

Синтаксис:
MPYI src, dst

Операнды:
src: основные режимы адресации src (G)
dst: регистр (любой регистр в основном регистровом файле ЦПУ)

Код:

31 29	24 23	21	16 15	0
0 0 0	0 1 0 1 0 1	G	dst	src

Поля слова:

G	src режимы адресации
00	Регистр (любой регистр в основном регистровом файле 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.
Разряды состояния:
Если ST (SET COND) = 0 и конечными являются регистры R0 – R11, флаги состояний изменяются. Если ST (SET COND) = 1, они изменяются для всех регистров назначения.

LUF	Не изменяется.
LV	1 при возникновении целочисленного переполнения, в противном случае не изменяется.
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.

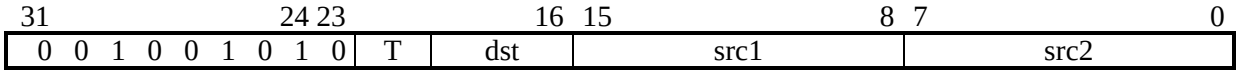
Разряд режима:
OVM Операция зависит от значения разряда OVM.
Циклы:
1
Пример:
MPYI R1, R5

	Перед инструкцией			После инструкции	
R1	00 0033 C251h	3 392 081	R1	00 0033 C251h	3 392 081
R5	00 0078 B600h	7 910 912	R5	00 E21D 9600h	– 501 377 536
LUF	0		LUF	0	
LV	0		LV	1	
UF	0		UF	0	
N	0		N	1	
Z	0		Z	0	
V	0		V	1	
C	0		C	0	

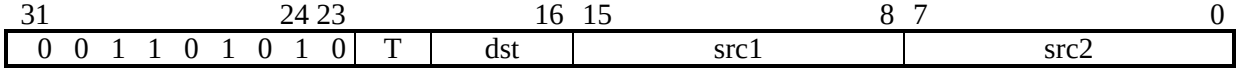
Результат переполнен и R5 содержит 32 его младших разряда. Чтобы получить 32 старших разряда, используйте MPYSHI3 или MPYUHI3 инструкции.

MPYI3 инструкция

Синтаксис:
MPYI3 src2, src1, dst
Операнды:
src1, src2: трёхоперандные режимы адресации первого или второго типа
dst: регистровый режим (любой регистр в основном регистровом файле ЦПУ)
Код:
Тип 1



Тип 2



Поля слова:

Тип 1

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	Регистр (любой регистр ЦПУ)
01	Косвенная (смещение = 0, 1, IR0, IR1)	Регистр (любой регистр ЦПУ)
10	Регистр (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	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	8-битная знаковая непосредственная
01	Регистр (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-битная знаковая непосредственная
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.

Разряды состояния:

Если ST (SET COND) = 0 и конечными являются регистры R0 – R11, флаги состояний изменяются. Если ST (SET COND) = 1, они изменяются для всех регистров назначения.

LUF	Не изменяется.
LV	1 при возникновении целочисленного переполнения, в противном случае не изменяется.
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.

Разряд режима:

OVM операция зависит от значения разряда OVM.

Циклы:

1

Пример:

Нет

MPYI3 || ADDI3 инструкция

Синтаксис:

MPYI3 srcA, srcB, dst1
|| **ADDI3** srcC, srcD, dst2

Операнды:

srcA, **srcB**, **srcC**, **srcD** — Любые два должны быть косвенными (смещение = 0, 1, IR0, IR1) и любые два должны быть регистрами (R0 – R7)

dst1 регистр (d1):
0 = R0
1 = R1

dst2 регистр (d2):
0 = R2
1 = R3

src1 регистр (R0 – R7)
src2 регистр (R0 – R7)
src3 косвенная (смещение = 0, 1, IR0, IR1)
src4 косвенная (смещение = 0, 1, IR0, IR1)
P Параллельные режимы 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	0	P	d1	d2	src1		src2		src3				src4			

Поля слова:

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 производится параллельно (ADDI3), записывает в тот же регистр, MPYI3 имеет на входе содержимое регистра до того, как оно было изменено инструкцией ADDI3.

Для четырёх 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 задано, изменить порядок операндов в коммутативных операциях, чтобы упростить обработку.

Разряды состояния:

LUF Не изменяется.
LV 1 при возникновении целочисленного переполнения, в противном случае не изменяется.
UF 0.

N 0.
Z 0.
V 1 при возникновении целочисленного переполнения, иначе 0.
C Не изменяется.

Разряд режима:
OVM Операция зависит от значения разряда 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	

MPYI3 || STI инструкция

Синтаксис:
MPYI3 src2, src1, dst
|| **STI** src3, dst2

Операнды:
src1: регистр (R0 – R7)
src2: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (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

Поля слова:
Нет

Операция:
src1 × src2 → dst1
|| src3 → dst2

Описание:
Целочисленное умножение и сохранение целого производится 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 производится параллельно (STI), считывает из того же регистра, 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, 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.
Разряды состояния:

LUF	Не изменяется.
LV	1 при возникновении целочисленного переполнения, в противном случае не изменяется.
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.

Разряд режима:
OVM операция зависит от значения разряда OVM.
Циклы:
1
Пример:
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 995Bh

	0C8h	200

Данные в 80 982E

	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 982Eh	

Данные в 80 995Bh

	0C8h	200

Данные в 80 982Eh

	0DCh	220
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

MPYI3 || SUBI3 инструкция

Синтаксис:
MPYI3 srcA, srcB, dst1
|| **SUBI3** srcC, srcD, dst2

Операнды:

srcA, **srcB**, **srcC**, **srcD** — Любые два должны быть косвенными (смещение = 0, 1, IR0, IR1) и любые два должны быть регистрами (R0 – R7)

dst1 регистр (d1):
0 = R0
1 = R1

dst2 регистр (d2):
0 = R2
1 = R3

src1 регистр (R0 – R7)

src2 регистр (R0 – R7)

src3 косвенная (смещение = 0, 1, IR0, IR1)

src4 косвенная (смещение = 0, 1, IR0, IR1)

P Параллельные режимы 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			

Поля слова:

Нет

Операция:

srcA × srcB → dst1
|| srcD – srcC → dst2

Описание:

Целочисленное умножение и целочисленное вычитание 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 производится параллельно (SUBI3), записывает в тот же регистр, MPYI3 имеет на входе содержимое регистра до того, как оно было изменено инструкцией SUBI3.

Для четырёх 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 задано, изменить порядок операндов в коммутативных операциях с целью упрощения обработки.

Переполнение целого числа происходит, когда какой-либо из 32 старших разрядов 64-битного результата отличается от старших разрядов 32-битного значения dst1.

Разряды состояния:

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не изменяется.

UF 1 при возникновении отрицательного целочисленного переполнения, в противном случае не изменяется.

N 0.

Z 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C Не изменяется.

Разряд режима:
OVM операция зависит от значения разряда 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

62h	98

Данные в 80 99FCh

	4B0h	1200
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

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	

MPYSHI инструкция

Синтаксис:
MPYSHI src, dst
Операнды:
src: основные режимы адресации (G)
dst: регистровый режим (любой регистр в основном регистровом файле ЦПУ)
Код:

31 29	23	21	16	0
0 0 0	1 1 1 0 1 1	G	dst	src

Поля слова:

G	src режимы адресации
00	Регистр (любой регистр ЦПУ в основном регистровом файле)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
dst × src → dst
Описание:
Тридцать два старших разряда результата произведения операндов dst и src загружаются в регистр dst. При чтении операнды src и dst являются 32-разрядными целыми числами со знаком.

Результат является 64-разрядным целым со знаком. В регистре dst сохраняются 32 старших бита результата. Инструкция MPYI сохраняет 32 младших разряда результата.

Разряды состояния:

Если ST (SET COND) = 0 и конечными являются регистры (R0 – R11), флаги состояний изменяются. Если ST (SET COND) = 1, они изменяются для всех регистров назначения.

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.

Разряд режима:

OVM операция зависит от значения разряда OVM.

Циклы:

1

Пример:

Нет

MPYSHI3 инструкция

Синтаксис:

MPYSHI3 src2, src1, dst

Операнды:

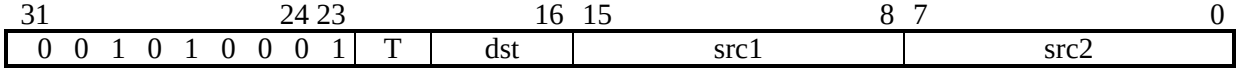
src1: трёхоперандные режимы адресации первого или второго типа

src2: трёхоперандные режимы адресации первого или второго типа

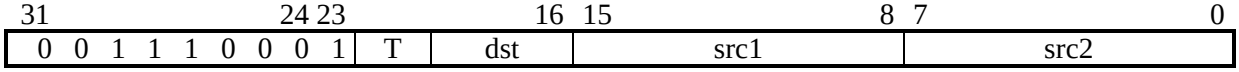
dst: регистровый режим (любой регистр в основном регистровом файле ЦПУ)

Код:

Тип 1



Тип 2



Поля слова:

Тип 1

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	Регистр (любой регистр ЦПУ)
01	Косвенная (смещение = 0, 1, IR0, IR1)	Регистр (любой регистр ЦПУ)
10	Регистр (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	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	8-битная знаковая непосредственная
01	Регистр (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-битная знаковая непосредственная
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 регистре dst сохраняются 32 старших разряда результата. MPYI3 инструкция сохраняет 32 младших разряда результата.
Разряды состояния:
Если ST (SET COND) = 0 и конечными являются регистры (R0 – R11), флаги состояний изменяются. Если ST (SET COND) = 1, они изменяются для всех регистров назначения.

LUF	Не изменяется.
LV	1 при возникновении целочисленного переполнения, в противном случае не изменяется.
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.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
Нет

MPYUHI инструкция

Синтаксис:
MPYUHI src, dst
Операнды:
src: основные режимы адресации
dst: регистровый режим (любой регистр в основном регистровом файле ЦПУ)
Код:

31 29	23	21	16	0
0 0 0	1 1 1 1 0 0	G	dst	src

Поля слова:

G	src режимы адресации
00	Регистр (любой регистр ЦПУ в основном регистровом файле)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
dst × src → dst
Описание:
32 старших разряда результата произведения операндов dst и src загружаются в регистр dst. При чтении операнды src и dst являются 32-разрядными целыми числами со знаком. Результат является 64-разрядным целым со знаком. В регистре dst сохраняются 32 старших разряда результата. MPYI инструкция сохраняет 32 младших разряда результата.

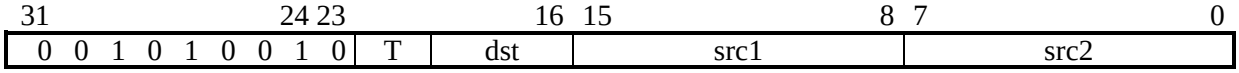
Разряды состояния:
Если ST (SET COND) = 0 и конечными являются регистры (R0 – R11), флаги состояний изменяются. Если ST (SET COND) = 1, они изменяются для всех регистров назначения.

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.

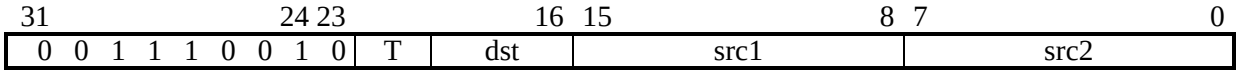
Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
Нет

MPYUHI3 инструкция

Синтаксис:
MPYUHI3 src2, src1, dst
Операнды:
src1, src2: трёхоперандные режимы адресации первого или второго типа
dst: регистровый режим (любой регистр в основном регистровом файле ЦПУ)
Код:
Тип 1



Тип 2



Поля слова:
Тип 1

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	Регистр (любой регистр ЦПУ)
01	Косвенная (смещение = 0, 1, IR0, IR1)	Регистр (любой регистр ЦПУ)
10	Регистр (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	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	8-битная знаковая непосредственная
01	Регистр (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-битная знаковая непосредственная
11	Косвенная *+ARn1 (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:
src1 × src2 → dst

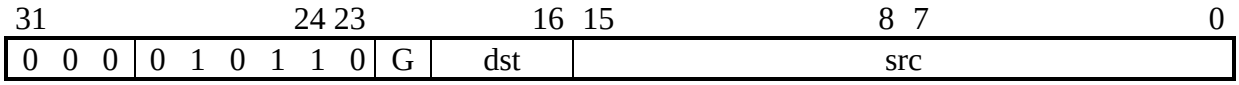
Описание:
Результат произведения операндов src1 и src2 загружается в регистр dst. Операнды src1 и src2 являются 32-разрядными целыми со знаком. Результат является 64-разрядным целым со знаком. В регистр dst помещается 32 старших разряда результата. MPYI3 инструкция сохраняет 32 младших разряда результата.
Разряды состояния:
Если ST (SET COND) = 0 и конечными являются регистры (R0 – R11), флаги состояний изменяются. Если ST (SET COND) = 1, они изменяются для всех регистров назначения.

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.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
Нет

NEGB инструкция

Синтаксис:
NEGB src, dst
Операнды:
src: основные режимы адресации (G)
dst: регистр (любой регистр в основном регистровом файле ЦПУ)
Код:



Поля слова:

G	src режимы адресации
00	Регистр (любой регистр в основном регистровом файле ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
0 – src – C → dst
Описание:
Разница от 0 операнда src и C (заём) загружается в регистр dst. Предполагается, что src и dst являются целыми со знаком.
Разряды состояния:
Если ST (SET COND) = 0 и конечными являются регистры (R0 – R11), флаги состояний изменяются. Если ST (SET COND) = 1, они изменяются для всех регистров назначения.

LUF	Не изменяется.
LV	1 при возникновении целочисленного переполнения, в противном случае не изменяется.
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 заёма, иначе 0.

Разряд режима:
OVM операция зависит от значения разряда OVM.
Циклы:
1
Пример:
NEGB R5, R7

Перед инструкцией

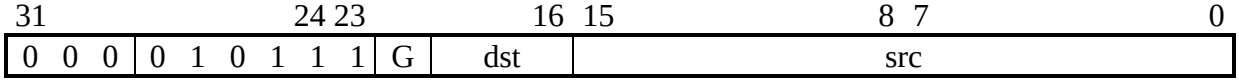
R5	0FFFF FFCBh	– 53
R7	0h	
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	1	

После инструкции

R5	0FFFF FFCBh	– 53
R7	34h	52
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	1	

NEGF инструкция

Синтаксис:
NEGF src, dst
Операнды:
src: основные адресные режимы (G)
dst: регистр (R0 – R11)
Код:



Поля слова:

G	src режимы адресации
00	Регистр (R0 – R11)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
0 – src → dst
Описание:
Операнд, представляющий собой вычитание из нуля операнда src, загружается в регистр dst. Предполагается, src и dst являются числами в формате с ПЗ.
Разряды состояния:

LUF	1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	1 при возникновении ПЗ-переполнения, в противном случае 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	1 при возникновении ПЗ-переполнения, иначе 0.
C	Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда 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

NEGF || STF инструкция

Синтаксис:

NEGF src2, dst1
|| **STF** src3, dst2

Операнды:

src2: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (смещение = 0, 1, IR0, IR1)

Код:

31 29	24 23	16 15	8 7	0
1 1 | 1 0 0 0 1	dst1	0 0 0 | src3	dst2	src2

Поля слова:

Нет

Операция:

0 – src2 → dst1
|| src3 → dst2

Описание:

Отрицание числа 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 (NEGF), записывает в тот же регистр, STF имеет на входе содержимое регистра до того, как оно было изменено инструкцией NEGF.

Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Разряды состояния:

LUF 1 при потере значимости результата с ПЗ, иначе не изменяется.
LV 1 при возникновении ПЗ-переполнения, в противном случае 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 1 при возникновении ПЗ-переполнения, иначе 0.
C Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

```
NEGF  *AR4– –(1), R7
|| STF   R2, *++AR5(1)
```

Перед инструкцией			После инструкции		
AR4	80 98E1h		AR4	80 98E0h	
R7	0h		R7	05 84C0 0000h	– 6,281250e+01
R2	07 33C0 0000h	1,79750e+02	R2	07 33C0 0000h	1,79750e+02
AR5	80 9803h		AR5	80 9804h	
Данные в 80 98E1h			Данные в 80 98E1h		
	57 B40 0000h	6,281250e+01		57 B40 0000h	6,281250e+01
Данные в 80 9804h			Данные в 80 9804h		
	0h			733 C000h	1,79750e+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	

NEGI инструкция

Синтаксис:

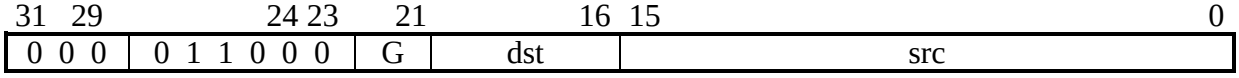
NEGI src, dst

Операнды:

src: основные режимы адресации src (G)

dst: регистр (любой регистр в основном регистровом файле ЦПУ)

Код:



Поля слова:

G	src режимы адресации
00	Регистр (любой регистр в основном регистровом файле 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, src и dst являются целыми со знаком.

Разряды состояния:

Если ST (SET COND) = 0 и конечными являются регистры R0 – R11, флаги состояний изменяются. Если ST (SET COND) = 1, они изменяются для всех регистров назначения.

LUF Не изменяется.
LV 1 при возникновении целочисленного переполнения, иначе не изменяется.
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 заёма, иначе 0.

Разряд режима:

OVM операция зависит от значения разряда OVM.

Циклы:
1
Пример:
NEGI 174, R5 (174 = 0AEh)

	Перед инструкцией			После инструкции	
R5	0DCh	220	AR4	0FFFFFF52h	–174
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	

NEGI || STI инструкция

Синтаксис:
NEGI src2, dst1
|| **STI** src3, dst2
Операнды:
src2: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (смещение = 0, 1, IR0, IR1)
Код:

31 29	24 23	16	15 8	7 0
1 1 1 0 0 1 0	dst1	0 0 1 src3	dst2	src2

Поля слова:
Нет
Операция:
0 – src2 → dst1
|| src3 → dst2
Описание:
Отрицание целого числа в 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, STI имеет на входе содержимое регистра до того, как оно было изменено инструкцией NEGI.
Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.
Разряды состояния:

LUF	Не изменяется.
LV	1 при возникновении целочисленного переполнения, в противном случае не изменяется.
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 заёма, иначе 0.

Разряд режима:
OVM операция зависит от значения разряда OVM.
Циклы:
1

Пример:

```
NEGI  *–AR3, R2
|| STI   R2, *AR1++
```

Перед инструкцией

AR3	80 982Fh
R2	19h
AR1	80 98A5h

Данные в 80 982Eh

0DCh

Данные в 80 98A5h

	0h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

25

220

После инструкции

AR3	80 982Fh
R2	0FFFF FF24h
AR1	80 98A6h

–220

Данные в 80 982Eh

0DCh

220

Данные в 80 98A5h

	19h
LUF	0
LV	0
UF	0
N	1
Z	0
V	0
C	1

25

NOP инструкция

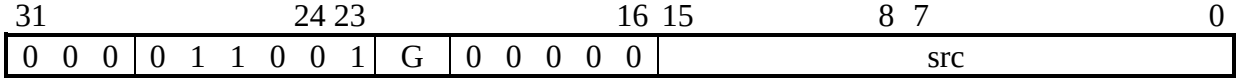
Синтаксис:

NOP src

Операнды:

src: основные режимы адресации (G)

Код:



Поля слова:

G	src режимы адресации
00	Регистр (нет операции)
10	Косвенная (модификация ARn, $0 \le n \le 7$)

Операция:

Нет никаких операций АЛУ или умножителя.

ARn изменяются, если src определён в режиме косвенной адресации.

Описание:

Если операнд src задан в режиме косвенной адресации, выполняется определённая операция адресации и имеет место холостое чтение памяти. Если операнд src опущен, не выполняются никакие операции.

Разряды состояния:

LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда 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

NORM инструкция

Синтаксис:
NORM src, dst
Операнды:
src: основные режимы адресации (G)
dst: регистр (R0 – R11)
Код:

31 24	23 16	15 8	7 0
0 0 0 0 1 1 0 1 0	G	dst	src

Поля слова:

G	src режимы адресации
00	Регистр (любой регистр в основном регистровом файле 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 удалённым 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.

Для значений src:

- Если src(exp) = – 128 и src(man) = 0, тогда dst = 0, Z = 1 и UF = 0.
- Если src(exp) = – 128 и src(man) ≠ 0, тогда dst = 0, Z = 0 и UF = 1.
- При любых других значениях src при возникновении отрицательного переполнения (т. е. потере значимости), dst(man) устанавливается в 0 и dst(exp) = – 128. Если src(man) = 0, тогда dst(man) = 0 и dst(exp) = – 128.

Разряды состояния:

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.

Разряд режима:
OVM операция не зависит от значения разряда 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
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

1.12451613e–04

NOT инструкция

Синтаксис:

NOT src, dst

Операнды:

src: основные режимы адресации (G)

dst: регистр (любой регистр основного регистрового файла ЦПУ)

Код:

Поля слова:

G	src режимы адресации
00	Регистр (любой регистр в основном регистровом файле ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:

~src → dst

Описание:

Поразрядное логическое дополнение операнда src загружается в регистр dst. Дополнение формируется путём инверсии каждого разряда операнда src. Операнды src и dst являются беззнаковыми целыми.

Разряды состояния:

Если ST (SET COND) = 0 и конечными являются регистры (R0 – R11), флаги состояний изменяются. Если ST (SET COND) = 1, они изменяются для всех регистров назначения.

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.

Разряд режима:

OVM операция зависит от значения разряда OVM.

Циклы:

1

Пример:

NOT @982Ch,R4

Перед инструкцией

DP	80h
R4	0h

Данные в 80 982Ch

	5E2Fh
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

DP	80h
R4	0FFFF A1D0h

Данные в 80 982Ch

	5E2Fh
LUF	0
LV	0
UF	0
N	1
Z	0
V	0
C	0

NOT || STI инструкция

Синтаксис:

NOT src2, dst1

|| **STI** src3, dst2

Операнды:

src2: косвенная (смещение = 0, 1, IR0, IR1)

dst1: регистр (R0 – R7)

src3: регистр (R0 – R7)

dst2: косвенная (смещение = 0, 1, IR0, IR1)

Код:

31 29	24 23	16 15	8 7	0
1 1	1 0 0 1 1 dst1	0 0 0 src3	dst2	src2

Поля слова:

Нет

Операция:

~src2 → dst1

|| src3 → dst2

Описание:

Поразрядное логическое НЕ (NOT) 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, STI имеет на входе содержимое регистра до того, как оно было изменено инструкцией NOT.

Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Разряды состояния:

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.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

```
NOT   *AR2, R3
|| STI  R7, *– –AR4(IR1)
```

Перед инструкцией

AR2	80 99CBh
R3	0h
R7	0DCh
AR4	80 9850h
IR1	10h

220

Данные в 80 99CCh

0C2Fh

Данные в 80 9840h

	0h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

AR2	80 99CBh
R3	0FFFF F3D0h
R7	0DCh
AR4	80 9840h
IR1	10h

220

Данные в 80 99CCh

0C2Fh

Данные в 80 9840h

	0DCh
LUF	0
LV	0
UF	1
N	0
Z	0
V	0
C	0

220

OR инструкция

Синтаксис:

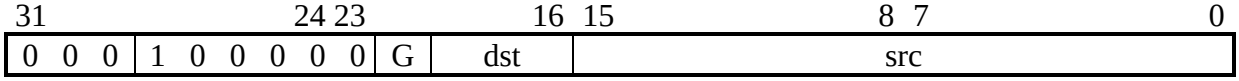
OR src, dst

Операнды:

src: основные режимы адресации (G)

dst: регистр (любой регистр в основном регистровом файле ЦПУ)

Код:



Поля слова:

G	src режимы адресации
00	Регистр (любой регистр в основном регистровом файле 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.

Разряды состояния:

Если ST (SET COND) = 0 и конечными являются регистры (R0 – R11), флаги состояний изменяются. Если ST (SET COND) = 1, они изменяются для всех регистров назначения.

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.

Разряд режима:
OVM операция не зависит от значения разряда OVM.

Циклы:
1

Пример:
OR *++AR1(IR1), R2

Перед инструкцией

AR1	80 9800h
IR1	4h
R2	01256 0000h

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

Данные в 80 9804h

	2BCDh
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

OR3 инструкция

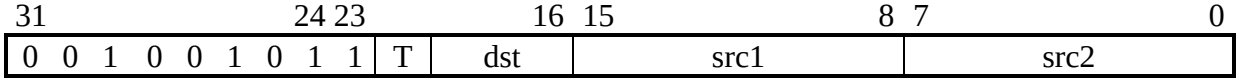
Синтаксис:

OR3 src2, src1, dst

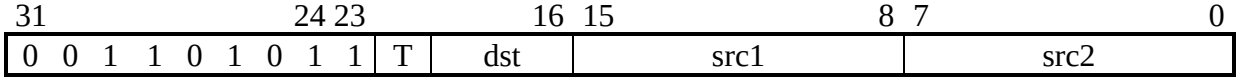
Операнды:

src1, src2: трёхоперандные режимы адресации первого или второго типа
dst: регистр (любой регистр в основном регистровом файле ЦПУ)

Код:
Тип 1



Тип 2



Поля слова:
Тип 1

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	Регистр (любой регистр ЦПУ)
01	Косвенная (смещение = 0, 1, IR0, IR1)	Регистр (любой регистр ЦПУ)
10	Регистр (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	src1 режимы адресации	src2 режимы адресации
01	Регистр (любой регистр ЦПУ)	8-битная знаковая непосредственная
01	Регистр (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-битная знаковая непосредственная
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 загружается в регистр dst. Предполагается, что операнды src1, src2 и dst являются целыми числами без знака. В режиме непосредственной адресации src2 расширяется на знак.

Разряды состояния:

Если ST (SET COND) = 0 и конечными являются регистры (R0 – R11), флаги состояний изменяются. Если ST (SET COND) = 1, они изменяются для всех регистров назначения.

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.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

Нет

OR3 || STI инструкция

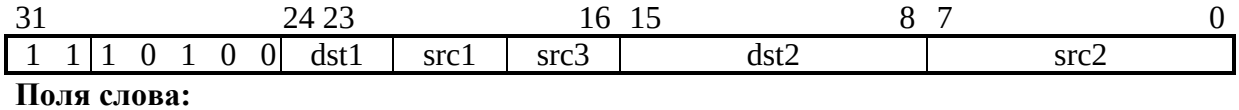
Синтаксис:

OR3 src2, src1, dst1
|| **STI** src3, dst2

Операнды:

src1: регистр (R0 – R7)
src2: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (смещение = 0, 1, IR0, IR1)

Код:



Поля слова:

Нет

Операция:

src1 OR src2 → dst1
|| src3 → dst2

Описание:
Поразрядное логическое ИЛИ (OR) 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 (OR3), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено инструкцией OR3.
Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Разряды состояния:

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.

Разряд режима:
OVM операция не зависит от значения разряда 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

Данные в 80 9831h

9800h

Данные в 80 9883h

	0h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

220

После инструкции

AR2	80 9831h
R5	80 0000h
R2	80 9800h
R6	0DCh
AR1	80 9882h

Данные в 80 9831h

9800h

Данные в 80 9883h

	0DCh
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

220

220

POP инструкция

Синтаксис:
POP dst

Операнды:
dst: регистр (любой регистр в основном регистровом файле ЦПУ)

Код:

31			24	23						16	15								8	7							0
0	0	0	0	1	1	1	0	0	0	1	dst																

Поля слова:
Нет

Операция:
*SP– – → dst

Описание:
Вершина системного стека выталкивается и загружается в регистр dst (в 32 младших разряда). Вершина стека – целое число со знаком. Инструкцией POP выполняется с постдекрементом указателя стека. Восемь старших разрядов (экспонента) в регистрах расширенной точности (R11 – R0) слева не изменяются. Если указано, они могут быть восстановлены с POPF инструкцией.

Разряды состояния:
Если ST (SET COND) = 0 и конечными являются регистры (R0 – R11), флаги состояний изменяются. Если ST (SET COND) = 1, они изменяются для всех регистров назначения.

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.

Разряд режима:
OVM операция не зависит от значения разряда 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 0DA4h	– 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	

POPF инструкция

Синтаксис:
POPF dst

Операнды:
dst: регистр (R0 – R7)

Код:

31																24	23															16	15															8	7							0
0	0	0	0	1	1	1	0	1	0	1	dst																																										

Поля слова:
Нет

Операция:
*SP– – → dst1

Описание:
Вершина 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
Разряды состояния:

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.

Разряд режима:
OVM операция зависит от значения разряда OVM
Пример:
POPF R4

Перед инструкцией

SP	80 984Ah	
R4	02 5D2E 0123h	6,91186578e+00

Данные в 80 984Ah

	5F2C 1302h	5,32544007e+28
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

SP	80 9849h	
R4	5F 2C13 0200h	5,32544007e+28

Данные в 80 984Ah

	5F2C 1302h	5,32544007e+28
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

PUSH инструкция
Синтаксис:
PUSH src
Операнды:
src: регистр (любой регистр в основном регистровом файле ЦПУ)
Код:

31 … 24	23 … 16	15 … 8	7 … 0
0 0 0 0 1 1 1 1 1 0	0 1 dst	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	

Поля слова:
Нет
Операция:
src → *++SP
Описание:
Содержимое регистра src (32 младших значащих разряда) загружается в текущий системный стек. Целое или мантисса регистра с повышенной точностью (R0 – R11) сохраняется с этой инструкцией. Экспонента (восемь старших разрядов) может быть вытолкнута с помощью PUSHF инструкции. Src предполагается знаковым целым. PUSH изменяется с прединкрементом указателя стека.
Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.

Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.

Циклы:
1

Пример:
PUSH R6

Перед инструкцией

SP	80 98AEh
R6	815Bh

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

Данные в 80 98AFh

	815Bh
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

PUSHF инструкция

Синтаксис:
PUSHF src

Операнды:
src: регистр (R0 – R11)

Код:

31 24	23 16	15 8	7 0
0 0 0 0 1 1 1 1 1 1	0 1	dst	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Поля слова:
Нет

Операция:
src → *++SP

Описание:
Содержимое регистра src (32 старших значащих разряда) загружается в текущий системный стек. Предполагается, что src является числом в формате с ПЗ. Инструкция PUSHF выполняется с прединкрементом указателя стека. Восемь младших значащих разрядов мантиссы не сохраняются (в приведённом ниже примере обратите внимание на отличие значений в R2 и в стеке), но они могут быть сохранены при использовании PUSH инструкции. PUSHF инструкция может быть вызвана после PUSH инструкции.

Разряды состояния:
LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
PUSHF R2

Перед инструкцией

SP	80 9801h	
R2	02 5C12 8081h	6,87725854e+00

Данные в 80 9802h

	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 9802h

	025C 1280h	6,87725830e+00
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

RCPF инструкция
Синтаксис:
PUSHF src, dst
Операнды:
src: регистр повышенной точности, прямая и 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

Поля слова:

G	src режимы адресации
00	Регистр (любой регистр в основном регистровом файле ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
16-разрядная обратная величина от src → dst
Описание:
16-разрядная приблизительная обратная величина от src операнда загружается в dst регистр. Dst и src операнды представляют собой числа с плавающей запятой.
Разряды состояния:

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.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
Нет

RETIcond инструкция

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	0	0	0	0	0

Поля слова:
Нет
Операция:
Если cond – «истина»:
*(SP) – – → PC
ST(PGIE) → ST(GIE)
ST(PCF) → ST(CF)
Иначе, продолжение.
Описание:
Если условие истинно, вершина стека выталкивается в PC, PGIE копируется в GIE, и PCF копируется в CF. Если условие ложно, тогда продолжается нормальная операция.
Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
4
Пример:
Нет

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	0	0	0	0	0

Поля слова:
Нет

Операция:
Если cond – «истина»:
*(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, 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).
Прерывания запрещены в течение выполнения RETIcondD.
Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
Нет

RETScond инструкция

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

Поля слова:
Нет
Операция:
Если cond – «истина»:
*SP– – → PC
Иначе, продолжение.
Описание:
Выполняется условный возврат. Если условие истинно, вершина стека выталкивается в PC. В ядре процессора 1867ВЦ8Ф1 имеется 20 кодов условий, которые могут быть использованы с этой инструкцией.
Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда 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

RND инструкция
Синтаксис:
RND src, dst
Операнды:
src: основные режимы адресации (G)
dst: регистр (R0 – R11)
Код:

31 24	23 16	15 8	7 0
0 0 0 1 0 0 0 1 0	G	dst	src

Поля слова:

G	src режимы адресации
00	Регистр (любой регистр в основном регистровом файле 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 отличии от бита переполнения.
Разряды состояния:

LUF	1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	1 при возникновении ПЗ-переполнения, в противном случае не изменяется.
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.

Разряд режима:
OVM операция зависит от значения разряда OVM.
Циклы:
1
Пример:
RND R5, R2

	Перед инструкцией		После инструкции	
R5	07 33C1 6EEFh	1,79755599e+02	07 33C1 6EEFh	1,79755599e+02
R2	0h		07 33C1 6EEFh	1,79755600e+02
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		0	
V	0		0	
C	0		0	

ROL инструкция
Синтаксис:
ROL dst
Операнды:
dst: регистр (любой регистр в основном регистровом файле 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	0	0	1

Поля слова:
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:

Разряды состояния:
Если ST (SET COND) = 0 и конечными являются регистры (R0 – R11), флаги состояний изменяются. 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	Устанавливается по значению циклически сдвигаемого вовне старшего разряда.

Разряд режима:
OVM операция не зависит от значения разряда OVM.

Циклы:
1
Пример:
ROL R3

Перед инструкцией

R3	8002 5CD4h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

После инструкции

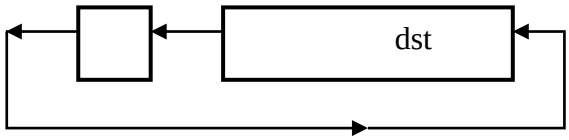
R3	0004 B9A9h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	1

ROLC инструкция

Синтаксис:
ROLC dst
Операнды:
dst: регистр (любой регистр в основном регистровом файле)
Код:

31								24	23							16	15															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	1

Поля слова:
Нет
Операция:
dst циклически сдвинутый влево на 1 разряд через разряд переноса → dst
Описание:
Содержимое операнда dst циклически сдвигается влево на один разряд через разряд 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:



Разряды состояния:
Если ST (SET COND) = 0 и конечными являются регистры (R0 – R11), флаги состояний изменяются. 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	Устанавливается по значению циклически сдвигаемого вовне старшего разряда.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1

Пример 1:
ROLC R3

	Перед инструкцией		После инструкции
R3	0000 0420h	R3	00000 0841h
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

Пример 2:
ROLC R3

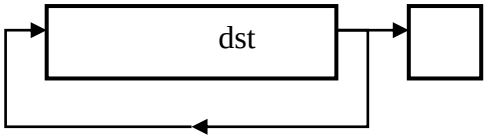
	Перед инструкцией		После инструкции
R3	8000 4281h	R3	0000 8502h
LUF	0	LUF	0
LV	0	LV	0
UF	0	UF	0
N	0	N	0
Z	0	Z	0
V	0	V	0
C	0	C	1

ROR инструкция

Синтаксис:
ROR dst
Операнды:
dst: регистр (любой регистр в основном регистровом файле ЦПУ)
Код:

31			24	23						16	15								8	7							0
0	0	0	1	0	0	1	0	1	1	1	dst			1	1	1	1	1	1	1	1	1	1	1	1	1	1

Поля слова:
Нет
Операция:
dst циклически сдвинутый вправо на 1 разряд через разряд 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:



Разряды состояния:
Если ST (SET COND) = 0 и конечными являются регистры (R0–R11), флаги состояний изменяются. 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 Устанавливается по значению циклически сдвигаемого вовне старшего разряда.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
ROR R7

	Перед инструкцией
R7	0000 0421h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

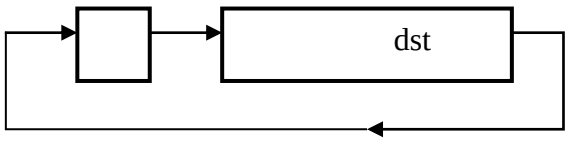
	После инструкции
R7	8000 0210h
LUF	0
LV	0
UF	0
N	1
Z	0
V	0
C	1

RORC инструкция

Синтаксис:
RORC dst
Операнды:
dst: регистр (любой регистр в основном регистровом файле 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

Поля слова:
Нет
Операция:
dst циклически сдвинутый вправо на 1 разряд через разряд переноса → dst
Описание:
Содержимое операнда dst циклически сдвигается вправо на один разряд через разряд переноса регистра состояния и загружается в регистр dst. Это выглядит как 33-разрядный сдвиг. Разряд переноса пересылается в старший значащий разряд dst, в то время как младший значащий разряд dst циклически сдвигается в разряд переноса.
Циклический левый сдвиг:



Разряды состояния:
Если ST (SET COND) = 0 и конечными являются регистры (R0 – R11), флаги состояний изменяются. 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 Устанавливается по значению циклически сдвигаемого вовне старшего разряда.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
RORC R4

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

RPTB инструкция

Синтаксис:
RPTB src
Операнды:
src: 24-разрядное знаковое непосредственное смещение или регистровый режим
Код:
Для 24-битного знакового непосредственного или регистрового режимов:

31–24	23–0
0 1 1 0 0 1 0 0	src (смещение)

Для регистрового режима:

31–23	22–5	4–0
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 0 0	src

Поля слова:
Нет
Операция:
src + PC + 1 $\rightarrow$ RE
1 $\rightarrow$ ST(RM)
Следующий PC $\rightarrow$ RS
Описание:
RPTB позволяет обеспечить повтор блока инструкций несколько раз без потерь на зацикливание. Эта инструкция активизирует режим повторения с изменением PC. Операнд src – может быть 32-битным регистром или 24-разрядным знаковым непосредственным значением (смещением). Результирующий src адрес – это конечный адрес блока повторения. Этот адрес загружается в регистр конечного адреса повтора (RE). В разряд режима повтора регистра состояния [ST(RM)] записывается 1, указывая, что PC будет изменяться в режиме повторения. Адрес следующей инструкции загружается в регистр стартового адреса повтора (RS).
RE должен быть больше или равен RS (RE≥RS). Иначе, код не повторяется, даже если бит RM остаётся установленным 1.
Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
4
Пример:
Нет

RPTBD инструкция

Синтаксис:
RPTBD src
Операнды:
src: 24-разрядное знаковое непосредственное смещение или регистровый режим
Код:
Для 24-битного знакового непосредственного или регистрового режимов:

31 … 24	23 … 0
0 1 1 0 0 1 0 1	src (смещение)

Для регистрового режима:

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	0 0 0 0 0 0 0 0	src

Поля слова:
Нет
Операция:
Если src – это непосредственное значение (смещение)

src + PC + 3 → RE

Иначе:

src → RE
1 → ST(RM)
PC RPTBD + 4 → RS

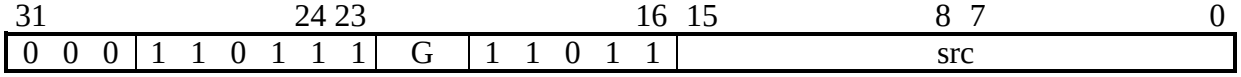
Описание:
RPTBD позволяет обеспечить повтор блока инструкций несколько раз без потерь на зацикливание и с одноцикловым выполнением RPTBD инструкции. Эта инструкция активизирует режим повторения с изменением PC. Операнд src – может быть 32-битным регистром или 24-разрядным знаковым непосредственным значением (смещением). Результирующий адрес загружается в регистр конечного адреса повтора (RE). В разряд режима повтора регистра состояния [ST(RM)] записывается 1, указывая, что PC будет изменяться в режиме повторения. Адрес следующей инструкции + 3 загружается в регистр стартового адреса повтора (RS).
RE должен быть больше или равен RS (RE≥RS). Иначе, код не повторяется, даже если бит RM остаётся установленным 1.
RPTBD не очищает конвейер. Три следующие за RPTBD инструкции в ходе своего исполнения не должны изменять ход программы. Эти инструкции не являются частью повторяющегося блока. Регистр количества повторов RC должен быть загружен перед выполнением RPTBD. Его не следует загружаться в трёх следующих за RPTBD инструкциях.
Прерывания запрещены в течение трёх следующих за RPTBD инструкций.
Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
Нет

RPTS инструкция

Синтаксис:
RPTS src
Операнды:
src: основные режимы адресации (G)
Код:



Поля слова:

G	src режимы адресации
00	Регистр
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
src → RC
1 → ST(RM)
1 → S
Следующий PC → RS
Следующий PC → RE
Описание:
RPTS позволяет обеспечить повтор отдельной инструкции несколько раз без потерь на зацикливание. Выборка также может производиться из регистра инструкций (IR), что позволяет избежать повторного доступа к памяти.
Значение src загружается в счётчик повторений (RC). 1 записывается в разряд режима повтора (RM) регистра состояния (ST). 1 также записывается в разряд одиночного повторения (S). Это указывает на то, что выборка программы будет выполняться только из регистра инструкций. Следующий PC загружается в регистр адреса конца повторения (RE) и в регистр адреса начала повторения (RS).
При непосредственном режиме адресации src – беззнаковое целое без расширения знака.
Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
4

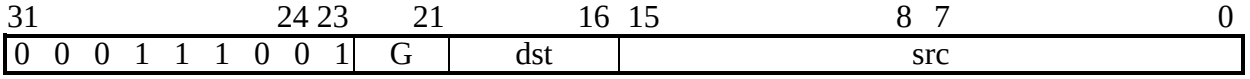
Пример:
RPTS AR5

	Перед инструкцией		После инструкции
PC	123h	PC	124h
ST	0h	ST	100h
RS	0h	RS	124h
RE	0h	RE	124h
RC	0h	RC	0FFh
AR5	0FFh	AR5	0FFh
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

Инструкция RPTS непрерываемая. Прерывания не произойдут до тех пор, пока не закончится выполнение RPTS. В программах, требовательных к таймингу это может привести к неточности, поэтому используйте инструкцию RPTS осторожно.

RSQRF инструкция

Синтаксис:
RSQRF src
Операнды:
src: регистр повышенной точности, прямая или косвенная адресация
dst: регистр повышенной точности
Код:



Поля слова:

G	src режимы адресации
00	Регистр
01	Прямая
10	Косвенная
11	16-битная непосредственная

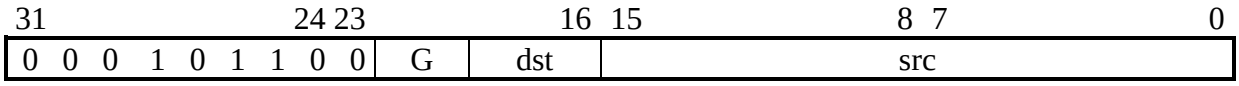
Операция:
16-разрядная величина, обратная квадратному корню из src → dst
Описание:
16-разрядная приблизительная обратная величина квадратного корня из числа src операнда загружается в dst регистр. Предполагается, что число в операнде src положительное. Операция для отрицательных чисел не определена.
Значения dst и src операндов предполагаются числами с плавающей запятой.
Разряды состояния:

LUF Не изменяется.
LV 1, если введён ноль, иначе не изменяется.
UF 0.
N 0.
Z 0.
V 1, если введён ноль, иначе 0.
C Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
Нет

SIGI инструкция

Синтаксис:
SIGI src, dst
Операнды:
src: прямая или косвенная адресация (предполагаются знаковыми целыми)
dst: регистр (предполагается знаковым целым)
Код:



Поля слова:

G	src режимы адресации
01	Прямая
10	Косвенная

Операция:
LOCK# (или LLOCK#) переходит в низкий уровень
src → dst
LOCK# (или LLOCK#) переходит в высокий уровень
Описание:
Операции блокировки сигнализируются посредством сигнала, блокирующего шину (LOCK# или LLOCK#), если, и только если, выполняется доступ к внешней памяти. Src и dst операнды представляют собой целые со знаком. После чтения сигнал блокировки снимается. Если выполняется доступ к внутренней памяти, SIGI выполняет чтение, но не устанавливает сигнал блокировки шины.
Значения src и dst операндов рассматриваются как целые со знаком.
Разряды состояния:
Если ST (SET COND) = 0 и конечными являются регистры (R0 – R11), флаги состояний изменяются. Если ST (SET COND) = 1, они изменяются для всех регистров назначения.

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.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
Нет

STF инструкция

Синтаксис:

STF src, dst

Операнды:

src: регистр (R0 – R11)

dst: основные режимы адресации (G)

Код:

31 24	23 16	15 8	7 0
0 0 0 1 0 1 0 0 0 0	G	src	dst

Поля слова:

G	src режимы адресации
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.

Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы: 1

Пример:

STF R2,@98A1h

Перед инструкцией

DP	80h	
R2	052 C501 900h	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	

STFI инструкция

Синтаксис:

STFI src, dst

Операнды:

src: регистр (R0 – R11)

dst: основные режимы адресации (G)

Код:

31 24	23 16	15 8	7 0
0 0 0 1 0 1 0 0 1	G	src	dst

Поля слова:

G	src режимы адресации
01	Прямая
10	Косвенная

Операция:

src → dst

Сигнализирует об окончании операции блокировки

Описание:

Операнд src загружается в память по адресу dst. Операция блокировки сигнализируется через выводы LOCK# или LLOCK#. Операнды src и dst являются числами в формате с ПЗ.

Разряды состояния:

LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

STFI R3, *–AR4

Перед инструкцией			После инструкции		
R3	07 33C0 0000h	1,79750e+02	R3	07 33C0 0000h	1,79750e+02
AR4	80 993Ch		AR4	80 993Ch	
Данные в 80 993Bh			Данные в 80 993Bh		
	0h			733 C000h	1,79750e+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	

STF || STF инструкция

Синтаксис:

STF src2, dst2
|| **STF** src1, dst1

Операнды:

src1: регистр (Rn1, 0 ≤ n1 ≤ 7)
dst1: косвенная (смещение = 0, 1, IR0, IR1)
src2: регистр (Rn2, 0 ≤ n2 ≤ 7)
dst2: косвенная (смещение = 0, 1, IR0, IR1)

Код:

31 29	24 23	16	15 8	7 0
1 1 | 0 0 0 0 0	src2 | 0 0 0	src1	dst1	dst2

Поля слова:

Нет

Операция:

src2 → dst2
|| src1 → dst1

Описание:

Две инструкции STF выполняются параллельно. Оба операнда src1 и src2 являются числами в формате с ПЗ.

Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда 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

R4	07 0C80 0000h	1,4050e+02
AR3	80 9834h	
R3	07 33C0 0000h	1,79750e+02
AR5	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	

STI инструкция

Синтаксис:

STI src, dst

Операнды:

src: регистр (любой регистр в основном регистровом файле ЦПУ)

dst: основные режимы адресации (G)

Код:

31 24	23 16	15 8	7 0
0 0 0 1 0 1 0 1 0	G	src	dst

Поля слова:

G	src режимы адресации
01	Прямая
10	Косвенная

Операция:

src → dst

Описание:

Операнд из регистра src загружается в память по адресу dst. Операнды src и dst являются целыми со знаком.

Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

STI R4,@982Bh

Перед инструкцией

DP	80h	
R4	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

DP	80h	
R4	42BD7h	273 367

Данные в 80 982Bh

	42BD7h	273 367
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

STII инструкция

Синтаксис:

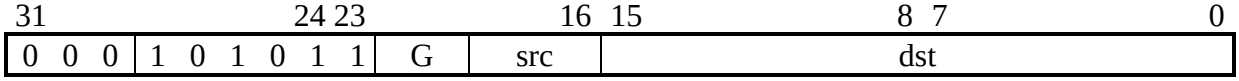
STII src, dst

Операнды:

src: регистр (любой регистр в основном регистровом файле ЦПУ)

dst: основные режимы адресации (G)

Код:



Поля слова:

G	src режимы адресации
01	Прямая
10	Косвенная

Операция:

src → dst

Сигнализирует об окончании операции блокировки

Описание:

Операнд src загружается в память по адресу dst. Операция блокировки сигнализируется через выводы LOCK# или LLOCK#. Операнды src и dst являются целыми со знаком.

Разряды состояния:

LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима:

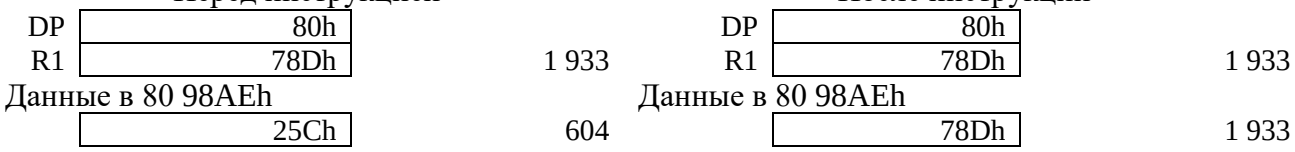
OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

STII R1,@98AEh



STI || STI инструкция

Синтаксис:

STI src2, dst2
|| **STI** src1, dst1

Операнды:

src1: регистр (Rn1, $0 \le n1 \le 7$)
dst1: косвенная (смещение = 0, 1, IR0, IR1)
src2: регистр (Rn2, $0 \le n2 \le 7$)
dst2: косвенная (смещение = 0, 1, IR0, IR1)

Код:

31 29	24 23	16 15	8 7	0
1 1	0 0 0 0 1	src2	0 0 0	src1	dst1	dst2

Поля слова:

Нет

Операция:

src2 → dst2

|| src1 → dst1

Описание:

Сохранение двух целых производится параллельно. Если обе операции сохранения выполняются по одному адресу, значение записывается как при выполнении операции STI src2, src2.

Разряды состояния:

LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда 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	

Данные в 80 9838h

	0h

Данные в 80 98D3h

	0h	
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

R0	0DCh	220
AR2	80 9838h	
IR0	8h	
R5	35h	53
AR0	80 98D3h	

Данные в 80 9838h

	0DCh

Данные в 80 98D3h

	35h	53
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

STIK инструкция

Синтаксис:

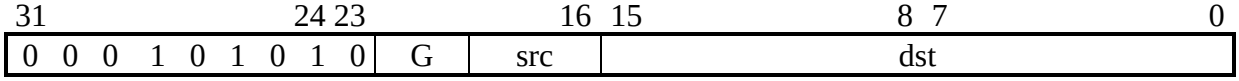
STIK src, dst

Операнды:

src: 5-разрядное знаковое целое

dst: прямая и косвенная адресация

Код:



Поля слова:

G	src режимы адресации
00	Прямая
11	Косвенная

Операция:

src → dst

Описание:

5-разрядное целое со знаком src загружается в память по адресу dst. Операнды src и dst являются целыми со знаком.

Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

Нет

SUBB инструкция

Синтаксис:

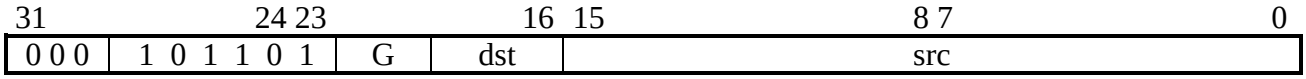
SUBB src, dst

Операнды:

src: основные режимы адресации (G)

dst: регистр (любой регистр в основном регистровом файле ЦПУ)

Код:



Поля слова:

G	src режимы адресации
00	Регистр (любой регистр в основном регистровом файле ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:

dst – src – C → dst

Описание:
Вычитание целых с заёмом. Разница между dst, src и C загружается в регистр dst (см. выше Операция). Предполагается, что операнды dst и src являются целыми со знаком.

Разряды состояния:
Если ST (SET COND) = 0 и конечными являются регистры (R0 – R11), флаги состояний изменяются. Если ST (SET COND) = 1, они изменяются для всех регистров назначения.

LUF	Не изменяется.
LV	1 при возникновении целочисленного переполнения, в противном случае не изменяется.
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 заёма, иначе 0.

Разряд режима:
OVM операция зависит от значения разряда OVM.

Циклы:
1

Пример:
SUBB *AR5++(4), R5

Перед инструкцией

AR5	80 9800h
R5	0FAh

Данные в 80 9800h

	0C7h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	1

После инструкции

AR5	80 9804h
R5	032h

Данные в 80 9800h

	0C7h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

SUBB3 инструкция

Синтаксис:
SUBB3 src2, src1, dst

Операнды:
src1, src2: трёхоперандные режимы адресации первого или второго типа
dst: регистр (любой регистр в основном регистровом файле 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:
Тип 1

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	Регистр (любой регистр ЦПУ)
01	Косвенная (смещение = 0, 1, IR0, IR1)	Регистр (любой регистр ЦПУ)
10	Регистр (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	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	8-битная знаковая непосредственная
01	Регистр (любой регистр ЦПУ)	Косвенная *+ARn (5-битное беззнаковое смещение)
10	Косвенная *+ARn (5-битное беззнаковое смещение)	8-битная знаковая непосредственная
11	Косвенная *+ARn1 (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:

src1 – src2 – C → dst

Описание:

Вычитание целых с заёмом, три операнда. Разница между dst, src и C загружается в регистр dst. Операнды src1, src2 и dst являются целыми со знаком.

Разряды состояния:

Если ST (SET COND) = 0 и конечными являются регистры (R0 – R11), флаги состояний изменяются. Если ST (SET COND) = 1, они изменяются для всех регистров назначения.

LUF Не изменяется.
LV 1 при возникновении целочисленного переполнения, в противном случае не изменяется.
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.

Разряд режима:

OVM операция зависит от значения разряда OVM.

Циклы:

1

Пример:

Нет

SUBC инструкция

Синтаксис:

SUBC src, dst

Операнды:

src: основные режимы адресации (G)
dst: регистр (любой регистр в основном регистровом файле ЦПУ)

Код:

Поля слова:

G	src режимы адресации
00	Регистр (любой регистр в основном регистровом файле 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, (dst – src) сдвигается влево на один разряд, младший значащий разряд устанавливается в 1 и результат загружается в регистр dst. Если (dst – src) меньше или равно нулю, 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 за один шаг многоразрядного целочисленного деления.
Разряды состояния:
LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.
Разряд режима:
OVM операция не зависит от значения разряда 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
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

После инструкции

R0	0FA0h	4000
LUF	0	
LV	0	
UF	0	
N	0	
Z	0	
V	0	
C	0	

SUBF инструкция

Синтаксис:

SUBF src, dst

Операнды:

src: основные режимы адресации (G)

dst: регистр (R0 – R11)

Код:

31 24	23 16	15 8	7 0
0 0 0 1 0 1 1 1 1	G dst	src	

Поля слова:

G	src режимы адресации
00	Регистр (R0–R11)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:

dst – src → dst

Описание:

Результат вычитания операнда src из операнда dst загружается в регистр dst. Операнды dst и src являются числами в формате с ПЗ.

Разряды состояния:

LUF 1 при потере значимости результата с ПЗ, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае 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 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

SUBF *AR0– –(IR0), R5

Перед инструкцией			После инструкции		
AR0	80 9888h		AR0	80 9808h	
IR0	80h		IR0	80h	
R5	07 33C0 0000h	1,79750000e+02	R5	05 1D00 0000h	3,9250e+01
Данные в 80 9888h			Данные в 80 9888h		
	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	

SUBF3 инструкция

Синтаксис:

SUBF3 src2, src1, dst

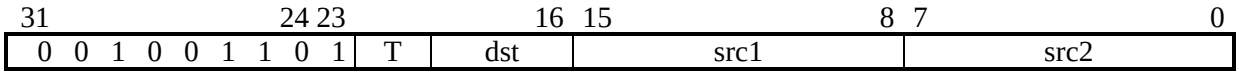
Операнды:

src1, src2: трёхоперандные режимы адресации первого или второго типа

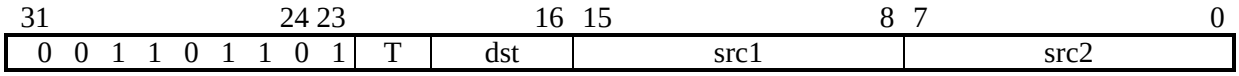
dst: регистровый режим (R0 – R11)

Код:

Тип 1



Тип 2



Поля слова:

Тип 1

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (R0 – R11)	Регистр (R0 – R11)
01	Косвенная (смещение = 0, 1, IR0, IR1)	Регистр (R0 – R11)
10	Регистр (R0 – R11)	Косвенная (смещение = 0, 1, IR0, IR1)
11	Косвенная (смещение = 0, 1, IR0, IR1)	Косвенная (смещение = 0, 1, IR0, IR1)

Тип 2

T	src1 режимы адресации	src2 режимы адресации
01	Регистр (любой регистр ЦПУ)	Косвенная *+ARn2 (5-битное беззнаковое смещение)
11	Косвенная *+ARn1 (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:

src1 – src2 → dst

Описание:

Результат вычитания операнда src2 из операнда src1 загружается в регистр dst. Операнды dst, src1 и src2 являются числами в формате с ПЗ.

Разряды состояния:

LUF 1 при потере значимости результата с ПЗ, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае 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 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

Нет

SUBF3 || STF инструкция

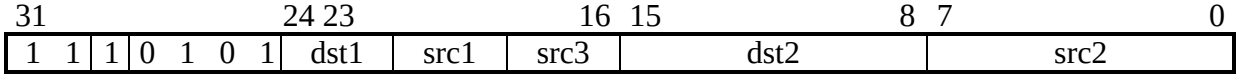
Синтаксис:

SUBF3 src1, src2, dst1
|| **STF** src3, dst2

Операнды:

src1: регистр (R0 – R7)
src2: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (смещение = 0, 1, IR0, IR1)

Код:



Поля слова:

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, STF имеет на входе содержимое регистра до того, как оно было изменено инструкцией SUBF3.

Если src3 и dst1 указывают на один и тот же адрес, src3 считывается до записи в dst1.

Разряды состояния:

LUF 1 при потере значимости результата с ПЗ, иначе не изменяется.
LV 1 при возникновении ПЗ-переполнения, в противном случае 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 1 при возникновении ПЗ-переполнения, иначе 0.
C Не изменяется.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

```
SUBF3  R1,* –AR4(IR1), R0
|| STF    R7, *+AR5(IR0)
```

Перед инструкцией

R1	05 7B40 0000h	6,28125e+01
AR4	80 98B8h	
IR1	8h	
R0	0h	
R7	07 33C0 0000h	1,79750e+02
AR5	80 9850h	
IR0	10h	

Данные в 80 98B0h

	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

R1	05 7B40 0000h	6,28125e+01
AR4	80 98B8h	
IR1	8h	
R0	06 1B60 0000h	7,768750e+01
R7	07 33C0 0000h	1,79750e+02
AR5	80 9850h	
IR0	10h	

Данные в 80 98B0h

	70C 8000h	1,4050e+02

Данные в 80 9860

	733 C000h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

SUBI инструкция

Синтаксис:

SUBI src, dst

Операнды:

src: основные режимы адресации (G)

dst: регистр (любой регистр в основном регистровом файле ЦПУ)

Код:

31 24	23 16	15 8	7 0
0 0 0 1 1 0 0 0 0 0	G	dst	src

Поля слова:

G	src режимы адресации
00	Регистр (любой регистр в основном регистровом файле ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:

dst – src → dst

Описание:

Результат вычитания операнда src из операнда dst загружается в регистр dst. Операнды dst и src являются целыми числами со знаком.

Разряды состояния:

Если ST (SET COND) = 0 и конечными являются регистры (R0 – R11), флаги состояний изменяются. Если ST (SET COND) = 1, они изменяются для всех регистров назначения.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не изменяется.

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 заёма, иначе 0.

Разряд режима:
OVM операция зависит от значения разряда OVM.

Циклы:
1

Пример:
SUBI 220, R7

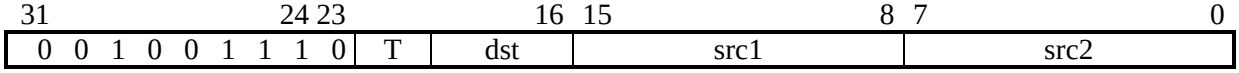
Перед инструкцией		550	После инструкции		330
R7	226h		R7	14Ah	
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	0		C	0	

SUBI3 инструкция

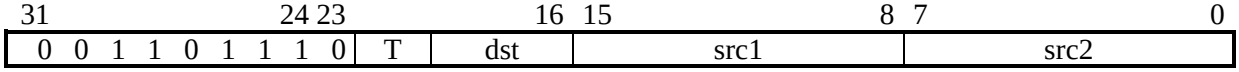
Синтаксис:
SUBI3 src2, src1, dst

Операнды:
src1, src2: трёхоперандные режимы адресации первого или второго типа
dst: регистровый режим (любой регистр в основном регистровом файле ЦПУ)

Код:
Тип 1



Тип 2



Поля слова:
Тип 1

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	Регистр (любой регистр ЦПУ)
01	Косвенная (смещение = 0, 1, IR0, IR1)	Регистр (любой регистр ЦПУ)
10	Регистр (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	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	8-битная знаковая непосредственная
01	Регистр (любой регистр ЦПУ)	Косвенная *+ARn2 (5-битное беззнаковое смещение)
10	Косвенная *+ARn1 (5-битное беззнаковое смещение)	8-битная знаковая непосредственная
11	Косвенная *+ARn1 (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:
src1 – src2 → dst

Описание:
Результат вычитания операнда src2 из операнда src1 загружается в регистр dst. Операнды dst, src1 и src2 являются целыми числами со знаком.
Разряды состояния:
Если ST (SET COND) = 0 и конечными являются регистры (R0 – R11), флаги состояний изменяются. Если ST (SET COND) = 1, они изменяются для всех регистров назначения.

LUF	Не изменяется.
LV	1 при возникновении целочисленного переполнения, в противном случае не изменяется.
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 заёма, иначе 0.

Циклы:
1
Разряд режима:
OVM операция зависит от значения разряда OVM.
Пример:
Нет

SUBI3 || STI инструкция

Синтаксис:
SUBI3 src1, src2, dst1
|| **STI** src3, dst2
Операнды:
src1: регистр (R0 – R7)
src2: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (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

Поля слова:
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, 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, src3 считывается до записи в dst1.

Разряды состояния:

LUF	Не изменяется.
LV	1 при возникновении целочисленного переполнения, в противном случае не изменяется.
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 заёма, иначе 0.

Разряд режима:

OVM операция зависит от значения разряда OVM.

Циклы:

1

Пример:

```
SUBI3 R7, *+AR2(IR0),R1
|| STI   R3 ,*++AR7
```

Перед инструкцией

R7	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	
N	0	
Z	0	
V	0	
C	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	
Z	0	
V	0	
C	0	

SUBRB инструкция

Синтаксис:

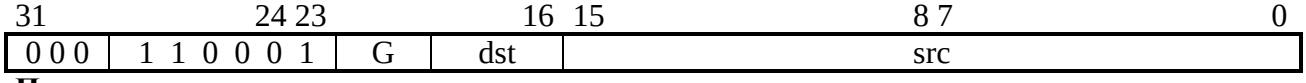
SUBRB src, dst

Операнды:

src: основные режимы адресации (G)

dst: регистр (любой регистр из основного регистрового файла ЦПУ)

Код:



Поля слова:

G	src режимы адресации
00	Регистр (любой регистр в основном регистровом файле ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
src – dst – C → dst
Описание:
Реверсное вычитание целых с заёмом. Разница между src, dst и C (см. выше Операция) загружается в регистр dst. Предполагается, что операнды dst и src являются целыми со знаком.
Разряды состояния:
Если ST (SET COND) = 0 и конечными являются регистры (R0 – R11), флаги состояний изменяются. Если ST (SET COND) = 1, они изменяются для всех регистров назначения.

LUF	Не изменяется.
LV	1 при возникновении целочисленного переполнения, в противном случае не изменяется.
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 заёма, иначе 0.

Разряд режима:
OVM операция зависит от значения разряда 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	

SUBRF инструкция

Синтаксис:
SUBRF src, dst
Операнды:
src: основные режимы адресации (G)
dst: регистр (R0 – R11)
Код:

31 24	23 16	15 8	7 0
0 0 0 1 1 0 0 1 0	G	dst	src

Поля слова:

G	src режимы адресации
00	Регистр (R0 – R11)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
src – dst → dst

Описание:
Реверсное вычитание чисел с плавающей запятой. Результат вычитания операнда dst из операнда src загружается в регистр dst. Операнды dst и src являются числами в формате с ПЗ.
Разряды состояния:

LUF	1 при потере значимости результата с ПЗ, иначе не изменяется.
LV	1 при возникновении ПЗ-переполнения, в противном случае 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	1 при возникновении ПЗ-переполнения, иначе 0.
C	Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
1
Пример:
SUBRF @9905h,R5

Перед инструкцией			После инструкции		
DP	80h		DP	80h	
R5	05 7B40 0000h	6,281250e+01	R5	06 69E0 0000h	1,16937500e+02
Данные в 80 9905h			Данные в 80 9905h		
	733 C000h	1,79750e+02		733 C000h	1,79750e+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	

SUBRI инструкция
Синтаксис:
SUBRI src, dst
Операнды:
src: основные режимы адресации (G)
dst: регистр (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

Поля слова:

G	src режимы адресации
00	Регистр (любой регистр в основном регистровом файле ЦПУ)
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
src – dst → dst
Описание:
Реверсное вычитание целых. Результат вычитания операнда dst из операнда src загружается в регистр dst. Предполагается, что операнды dst и src являются целыми числами со знаком.

Разряды состояния:

Если ST (SET COND) = 0 и конечными являются регистры (R0 – R11), флаги состояний изменяются. Если ST (SET COND) = 1, они изменяются для всех регистров назначения.

LUF	Не изменяется.
LV	1 при возникновении целочисленного переполнения, в противном случае не изменяется.
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 заёма, иначе 0.

Циклы:

1

Разряд режима:

OVM операция зависит от значения разряда OVM.

Пример:

SUBRI*AR5++(IR0),R3

Перед инструкцией			После инструкции		
AR5	80 9900h		AR5	80 9908h	
IR0	8h		IR0	8h	
R3	0DCh	220	R3	014Ah	330
Данные в 80 9900h			Данные в 80 9900h		
	226h	550		226h	550
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	

SWI инструкция

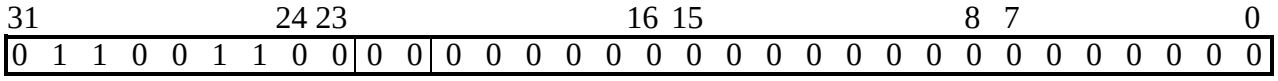
Синтаксис:

SWI

Операнды:

Нет

Код:



Поля слова:

Нет

Операция:

Выполняет прерывание эмуляции

Описание:

Программное прерывание. Инструкция SWI выполняет прерывание эмуляции. Она является резервной и не может быть использована в обычном программировании.

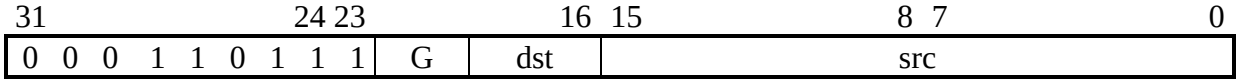
Разряды состояния:

LUF	Не изменяется.
LV	Не изменяется.
UF	Не изменяется.
N	Не изменяется.
Z	Не изменяется.
V	Не изменяется.
C	Не изменяется.

Разряд режима:
OVM операция не зависит от значения разряда OVM.
Циклы:
4
Пример:
Нет

TOIEEE инструкция

Синтаксис:
TOIEEE src, dst
Операнды:
src: регистр повышенной точности (R0 – R11), режимы прямой или косвенной адресации
dst: регистр повышенной точности
Код:



Поля слова:

G	src режимы адресации
00	Регистр [регистр повышенной точности (R0 – R11)]
01	Прямая
10	Косвенная
11	Непосредственная

Операция:
Преобразование src в IEEE формат → dst
Описание:
Операнд src конвертируется из формата с плавающей запятой в дополнительном коде в IEEE формат с плавающей запятой.
Предполагается, что операнд src является числом с плавающей запятой одинарной точности, за исключением режима непосредственной адресации, где он представлен в коротком 16-разрядном формате с плавающей запятой. Конвертированный результат помещается в 32 старших разряда регистра dst. Для сохранения результата в памяти может быть использована инструкция STF.
Разряды состояния:

LUF	Не изменяется.
LV	1 при возникновении переполнения, в противном случае не изменяется.
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.

Разряд режима:
OVM операция зависит от значения разряда OVM.
Циклы:
1
Пример:
Нет

TOIEEE || STF инструкция

Синтаксис:

TOIEEE src2, dst1
|| STF src3, dst2

Операнды:

src2: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (смещение = 0, 1, IR0, IR1)

Код:

31 29	24 23		16 15	8 7	0
1 1 1 1 0 0 0	dst1	0 0 0	src3	dst2	src2

Поля слова:

Нет

Операция:

Конвертированный в IEEE формат
src2 → dst1
параллельно с
src3 → dst2

Описание:

Операнд src2 преобразуется из формата с плавающей запятой в дополнительном коде в IEEE формат с плавающей запятой.
Предполагается, что операнд src2 является числом с плавающей запятой одинарной точности. Конвертированный результат помещается в 32 старших регистра dst1. Параллельно число с ПЗ сохраняется в памяти.
Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Разряды состояния:

LUF Не изменяется.
LV 1 при возникновении переполнения, в противном случае не изменяется.
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.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

Нет

TRAPcond инструкция

Синтаксис:

TRAPcond**N**

Операнды:

N: режим 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:

Нет

Операция:

Если условие(cond) – «истина»:

ST(GIE) → ST(PGIE)

ST(CF) → ST(PCF)

0 → ST(GIE)

1 → ST(CF)

Следующий PC → *(++SP)

Вектор системного прерывания N → PC

Иначе, продолжение.

Описание:

Если условие истинно, тогда состояние разрядов GIE и CF регистра состояния сохраняется в его же разрядах PGIE и PCF, соответственно; запрещаются все прерывания (0 → GIE) и «замораживается» кэш (1 → CF). Затем содержимое PC вталкивается в системный стек, а в PC загружается содержимое заданного вектора программных прерываний (N). Если условие ложно, тогда продолжается нормальное выполнение программы.

Если прерывания имеют вложенность, вам может понадобиться сохранить регистр состояния перед выполнением TRAPcond.

Разряды состояния:

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.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

5

Пример:

Нет

TSTB инструкция

Синтаксис:

TSTB src, dst

Операнды:

src: основные режимы адресации (G)

dst: регистр (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	

Поля слова:

G	src режимы адресации
00	Регистр (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.

Разряды состояния:

Флаги состояния 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.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

TSTB *–AR4(1), R5

Перед инструкцией			После инструкции		
AR4	80 99C5h		AR4	80 99C5h	
R5	898h	2200	R5	80 982Fh	2200
Данные в 80 99C4h			Данные в 80 99C4h		
	767h	1895		767h	1895
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	0		Z	1	
V	0		V	0	
C	0		C	0	

TSTB3 инструкция

Синтаксис:

TSTB3 src2, src1

Операнды:

src1, src2: трёхоперандные режимы адресации первого или второго типа

Код:

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:

Тип 1

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	Регистр (любой регистр ЦПУ)
01	Косвенная (смещение = 0, 1, IR0, IR1)	Регистр (любой регистр ЦПУ)
10	Регистр (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	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	8-битная знаковая непосредственная
01	Регистр (любой регистр ЦПУ)	Косвенная *+ARn2 (5-битное беззнаковое смещение)
10	Косвенная *+ARn1 (5-битное беззнаковое смещение)	8-битная знаковая непосредственная
11	Косвенная *+ARn1 (5-битное беззнаковое смещение)	Косвенная *+ARn2 (5-битное беззнаковое смещение)

Операция:

src1 AND src2

Описание:

Поразрядное логическое И операндов src1 и src2 формируется, но результат не загружается ни в один регистр. Таким образом обеспечивается сравнение без потери информации. Предполагается, что операнды являются беззнаковыми целыми. Src2 в режиме непосредственной адресации расширен знаком.

Хотя данная инструкция имеет только два операнда, она обозначается как трёхоперандная, т. к. операнды определены в трёхоперандном формате.

Разряды состояния:

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.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

Нет

XOR инструкция

Синтаксис:

XOR src, dst

Операнды:

src: основные режимы адресации (G)

dst: регистр (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

Поля слова:

G	src режимы адресации
00	Регистр (любой регистр в основном регистровом файле 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.

Разряды состояния:

Если ST (SET COND) = 0 и конечными являются регистры (R0 – R11), флаги состояний изменяются. 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.

Разряд режима:

OVM операция не зависит от значения разряда 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 FA32h
R2	00 0FF3h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

XOR3 инструкция

Синтаксис:

XOR3 src2, src1, dst

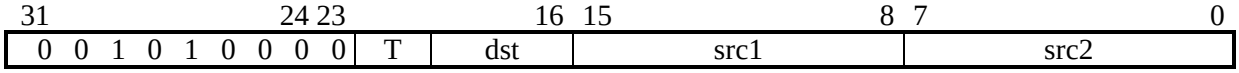
Операнды:

src1, src2: трёхоперандные режимы адресации первого или второго типа

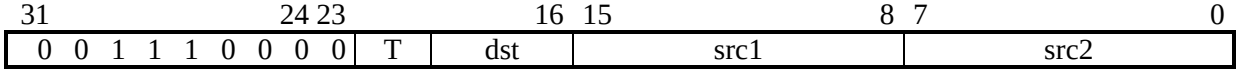
dst: регистр (любой регистр в основном регистровом файле ЦПУ)

Код:

Тип 1



Тип 2



Поля слова:

Тип 1

T	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	Регистр (любой регистр ЦПУ)
01	Косвенная (смещение = 0, 1, IR0, IR1)	Регистр (любой регистр ЦПУ)
10	Регистр (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	src1 режимы адресации	src2 режимы адресации
00	Регистр (любой регистр ЦПУ)	8-битная знаковая непосредственная
01	Регистр (любой регистр ЦПУ)	Косвенная *+ARn2 (5-битное беззнаковое смещение)
10	Косвенная *+ARn1 (5-битное беззнаковое смещение)	8-битная знаковая непосредственная
11	Косвенная *+ARn1 (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. Предполагается, что операнды src1, src2 и dst являются целыми числами без знака. В режиме непосредственной адресации src2 расширяется на знак.

Разряды состояния:

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.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

Нет

XOR3 || STI инструкция

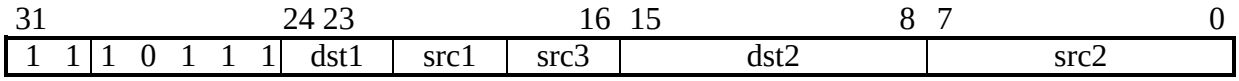
Синтаксис:

XOR3 src2, src1, dst1
|| **STI** src3, dst2

Операнды:

src1: регистр (R0 – R7)
src2: косвенная (смещение = 0, 1, IR0, IR1)
dst1: регистр (R0 – R7)
src3: регистр (R0 – R7)
dst2: косвенная (смещение = 0, 1, IR0, IR1)

Код:



Поля слова:

Нет

Операция:

src1 XOR src2 → dst1
|| src3 → dst2

Описание:

Поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ 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 (XOR3), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено инструкцией XOR3.

Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Разряды состояния:

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.

Разряд режима:

OVM операция не зависит от значения разряда OVM.

Циклы:

1

Пример:

```
XOR3  *AR1++, R3, R3
|| STI   R6, *–AR2(IR0)
```

Перед инструкцией

AR1	80 987Eh
R3	85h
R6	0DCh
AR2	80 98B4h
IR0	8h

Данные в 80 987Eh

85h

Данные в 80 98ACh

	0h
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

220

После инструкции

AR1	80 987Fh
R3	0h
R6	0DCh
AR2	80 98B4h
IR0	8h

Данные в 80 987Eh

85h

Данные в 80 98ACh

	0DCh
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

220

Приложение Б

(обязательное)

Временные параметры сигналов

Номер параметра в таблице соответствует его цифровому обозначению на рисунке.

Временные параметры сигналов X2/CLKIN_COMM, H1 (H1_1, H1_2) и H3 (H3_1, H3_2) смотри рисунки Б.1, Б.2 и таблицу Б.1.

Таблица Б.1 — В наносекундах

Буквенное обозначение параметра, наименование параметра, сигнал	Значение параметра	
	Мин.	Макс.
1 $t_{f(Cl)}$ – время спада, X2/CLKIN_COMM	–	2
2 $t_{w(ClL)}$ – время импульса, X2/CLKIN_COMM в низком уровне, $t_{c(Cl)}$ = min	3,4	–
3 $t_{w(ClH)}$ – время импульса, X2/CLKIN_COMM в высоком уровне, $t_{c(Cl)}$ = min	3,4	–
4 $t_{r(Cl)}$ – время нарастания фронта, X2/CLKIN_COMM	–	2
5 $t_{c(Cl)}$ – время цикла, X2/CLKIN_COMM	10	97
6 $t_{f(H)}$ – время спада, H1/H3	–	1,2
7 $t_{w(HL)}$ – время импульса, H1/H3 в низком уровне	$t_{c(Cl)}$ – 2,4	$t_{c(Cl)}$+2,4
8 $t_{w(HH)}$ – время импульса, H1/H3 в высоком уровне	$t_{c(Cl)}$ – 2,4	$t_{c(Cl)}$+2,4
9 $t_{r(H)}$ – время нарастания фронта, H1/H3	–	1,6
9.1 $t_{d(HL-HH)}$ время задержки, от H1 низкого уровня до H3 высокого уровня или от H3 низкого уровня до H1 высокого уровня	– 0,4	1,6
10 $t_{c(H)}$ – время цикла, H1/H3	20	194

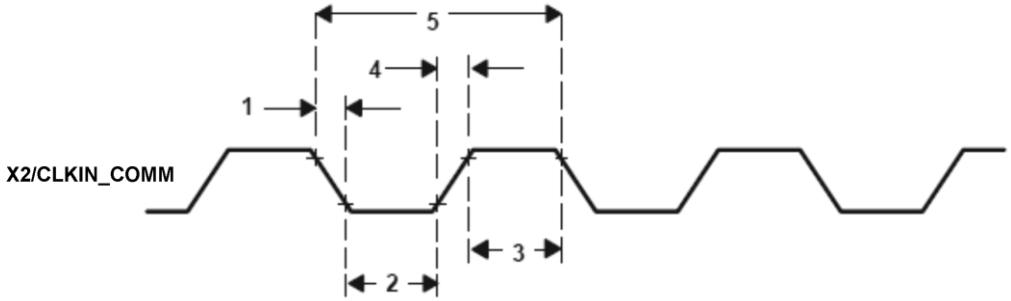


Рисунок Б.1 – Временные параметры сигнала X2/CLKIN_COMM

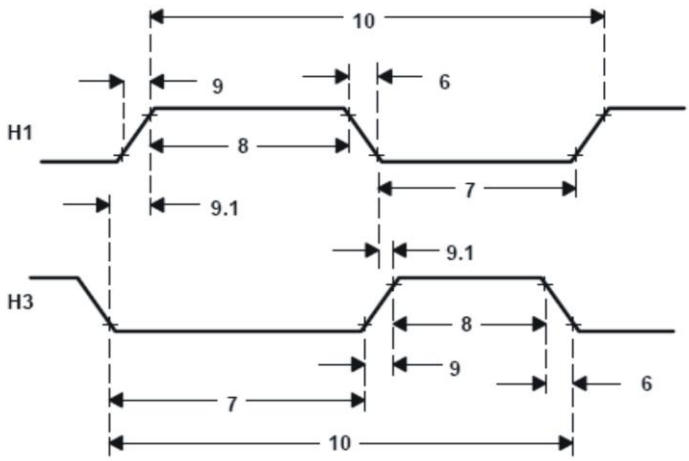


Рисунок Б.2 – Временные параметры сигналов H1 и H3

Таблица Б.2 – Временные параметры для чтения/записи памяти [STRBx# = 0]
(смотри рисунки Б.3, Б.4) В наносекундах

Буквенное обозначение параметра, наименование параметра	Значение параметра	
	Мин.	Макс.
1 $t_{d(H1L-SL)}$ – время задержки от H1 низкого уровня до значений STRBx# низкого уровня	0	4
2 $t_{d(H1L-SH)}$ – время задержки от H1 низкого уровня до значений STRBx# высокого уровня	0	4
3 $t_{d(H1H-RWL)}$ – время задержки от H1 высокого уровня до значений R/Wx# низкого уровня	0	3,6
4 $t_{d(H1L-A)}$ – время задержки от H1 низкого уровня до значений Ax	0	4
5 $t_{su(D-H1L)R}$ – время установки значения Dx перед низким уровнем H1 (чтение)	6	–
6 $t_{h(H1L-D)R}$ – время удержания Dx после H1 низкого уровня (чтение)	0	–
7 $t_{su(RDY-H1L)}$ – время установки значения RDYx# перед H1 низкого уровня	10	–
8 $t_{h(H1L-RDY)}$ – время удержания RDYx# после H1 низкого уровня	0	–
8.1 $t_{d(H1L-ST)}$ – время задержки от H1 низкого уровня до значений STAT3 – STAT0	–	8
9 $t_{d(H1H-RWH)W}$ – время задержки от H1 высокого уровня до значений R/Wx# высокого уровня (запись)	–	3,6
10 $t_{v(H1L-D)W}$ – значение времени Dx после H1 низкого уровня (запись)	–	6,4
11 $t_{d(H1H-D)W}$ – время удержания Dx после H1 высокого уровня (запись)	0	–
12 $t_{d(H1H-A)}$ – время задержки от H1 высокого уровня до значения адреса при циклах записи	–	5,2

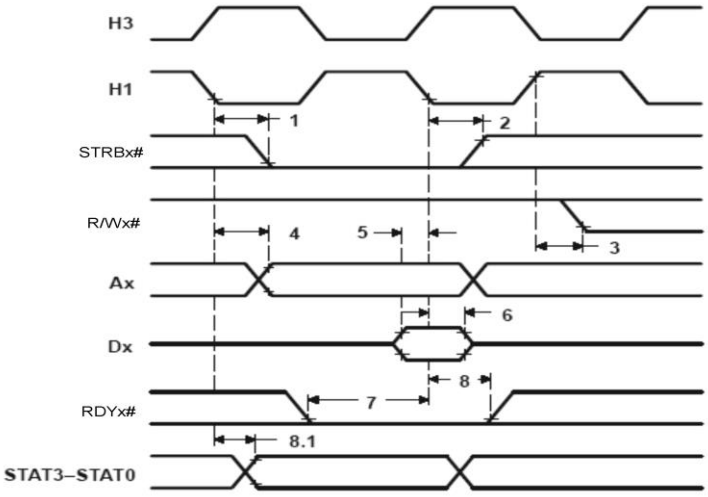


Рисунок Б.3 – Временные параметры для чтения памяти [STRBx# = 0]

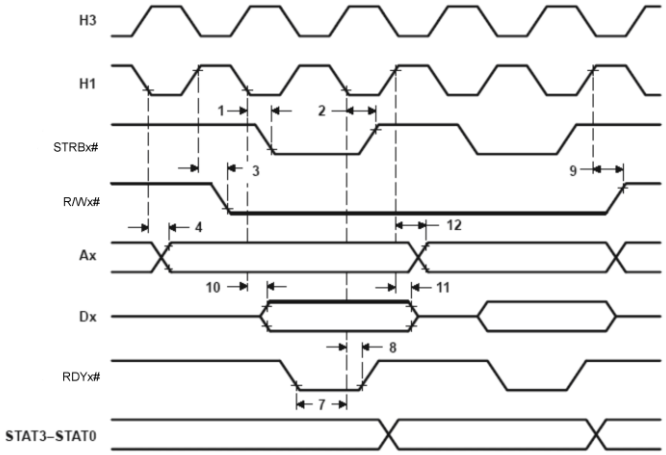


Рисунок Б.4 – Временные параметры для записи памяти [STRBx = 0]

Таблица Б.3 – Временные параметры DE# (DE_1#, DE_2#), AE# (AE_1#, AE_2#), CEx# (смотри рисунок Б.5) В наносекундах

Буквенное обозначение параметра, наименование параметра	Значение параметра	
	Мин.	Макс.
1 $t_{d(DEH-DZ)}$ – время задержки от DE# высокого уровня до значений D0 – D31 в высокоимпедансном состоянии	0	6
2 $t_{d(DEL-DV)}$ – время задержки DE# низкого уровня до значений D0 – D31	0	8,8
3 $t_{d(AEH-AZ)}$ – время задержки от AE# высокого уровня до значений A0-A31 в высокоимпедансном состоянии	0	6
4 $t_{d(AEL-AV)}$ – время задержки AE# низкого уровня до значений A0 – A31	0	8,4
5 $t_{d(CEH-RWZ)}$ – время задержки от CEx# высокого уровня до значений R/W0#, R/W1# в высокоимпедансном состоянии	0	6
6 $t_{d(CEL-RWV)}$ время задержки от CEx# низкого уровня до значений R/W0#, R/W1#	0	8,4
7 $t_{d(CEH-SZ)}$ – время задержки от CEx# высокого уровня до значений STRB0#, STRB1# в высокоимпедансном состоянии	0	6
8 $t_{d(CEL-SV)}$ – время задержки от CEx# низкого уровня до значений STRB0#, STRB1#	0	8,4
9 $t_{d(CEH-PAGEZ)}$ – время задержки от CEx# высокого уровня до значений PAGE0, PAGE1 в высокоимпедансном состоянии	0	6
10 $t_{d(CEL-PAGEV)}$ – время задержки от CEx# низкого уровня до значений PAGE0, PAGE1	0	8,4

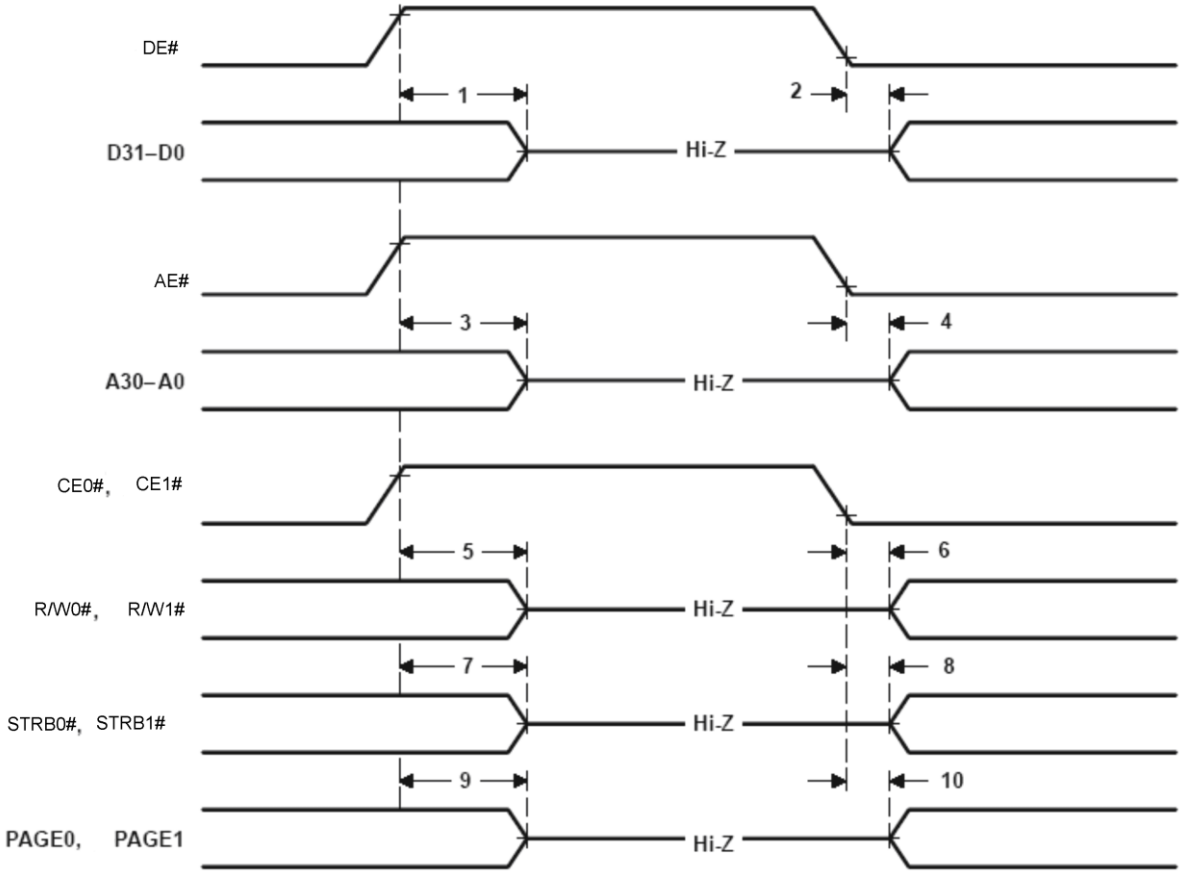


Рисунок Б.5 – Временные параметры DE#, AE#, CEx#

Таблица Б.4 – Временные параметры для LOCK# при выполнении LDFI или LDII (смотри рисунок Б.6)

В наносекундах

Буквенное обозначение параметра, наименование параметра	Значение параметра
	Макс.
1 $t_{d(H1L-LOCKL)}$ – время задержки от H1 низкого уровня до LOCK# низкого уровня	4,4

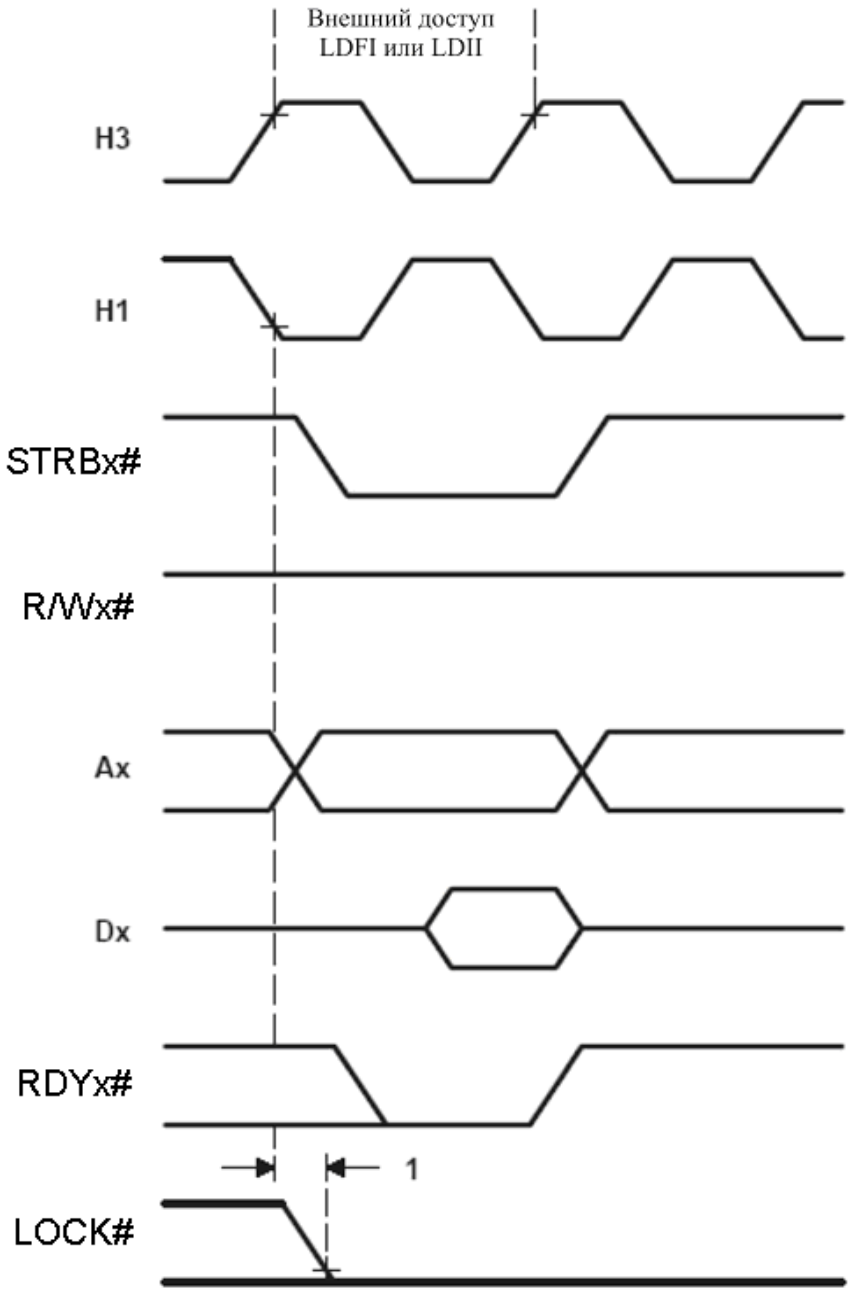


Рисунок Б.6 – Временные параметры для LOCK# при выполнении LDFI или LDII

Таблица Б.5 – Временные параметры для LOCK# при выполнении STFI или STII (смотри рисунок Б.7)

В наносекундах

Буквенное обозначение параметра, наименование параметра	Значение параметра
	Макс.
1 $t_{d(H1L-LOCKH)}$ – время задержки от H1 низкого уровня до LOCK# высокого уровня	4,4

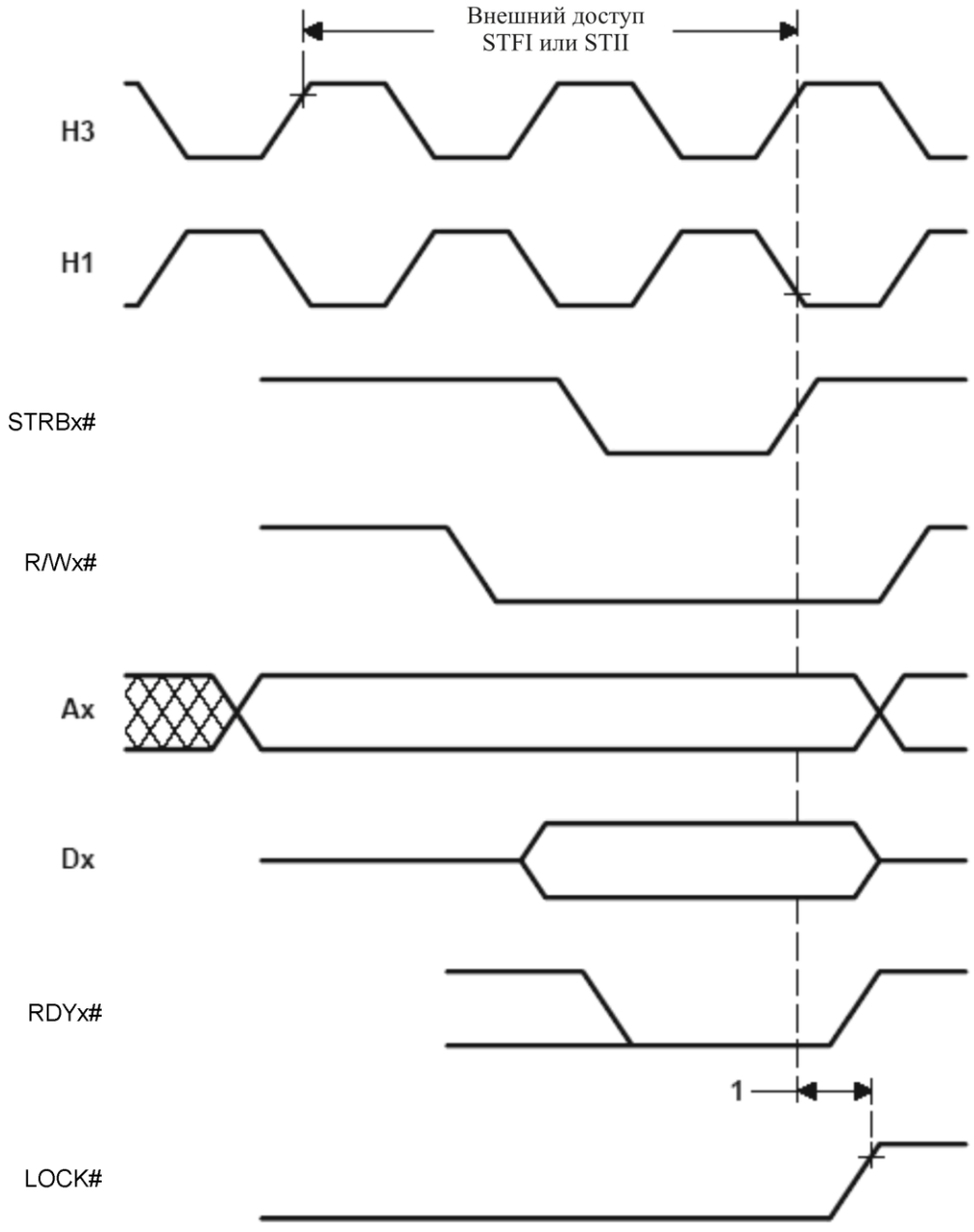


Рисунок Б.7 – Временные параметры для LOCK# при выполнении STFI или STII

Таблица Б.6 – Временные параметры для LOCK# при выполнении SIGI (смотри рисунок Б.8)

В наносекундах

Буквенное обозначение параметра, наименование параметра	Значение параметра
	Макс.
1 $t_{d(H1L-LOCKL)}$ – время задержки от H1 низкого уровня до LOCK# низкого уровня	4,4
2 $t_{d(H1L-LOCKH)}$ – время задержки от H1 низкого уровня до LOCK# высокого уровня	4,4

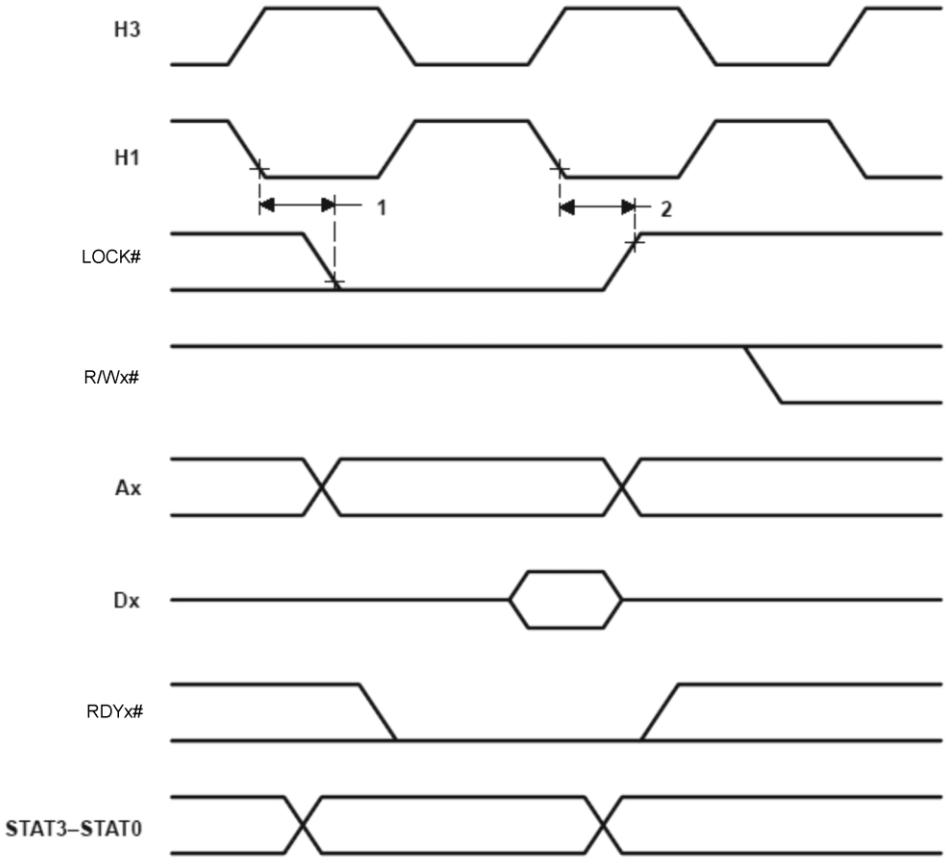


Рисунок Б.8 – Временные параметры для LOCK# при выполнении SIGI

Таблица Б.7 – Временные параметры для PAGE0# (PAGE0_1#, PAGE0_2#), PAGE1# (PAGE1_1#, PAGE1_2#) во время доступа памяти к другой странице (смотри рисунок Б.9)

В наносекундах

Буквенное обозначение параметра, наименование параметра	Значение параметра	
	Мин.	Макс.
1 $t_{d(H1L-PAGEH)}$ – время задержки от H1 низкого уровня до PAGEx высокого уровня для доступа к другой странице	0	4
2 $t_{d(H1L-PAGEL)}$ – время задержки от H1 низкого уровня до PAGEx низкого уровня для доступа к другой странице	0	4

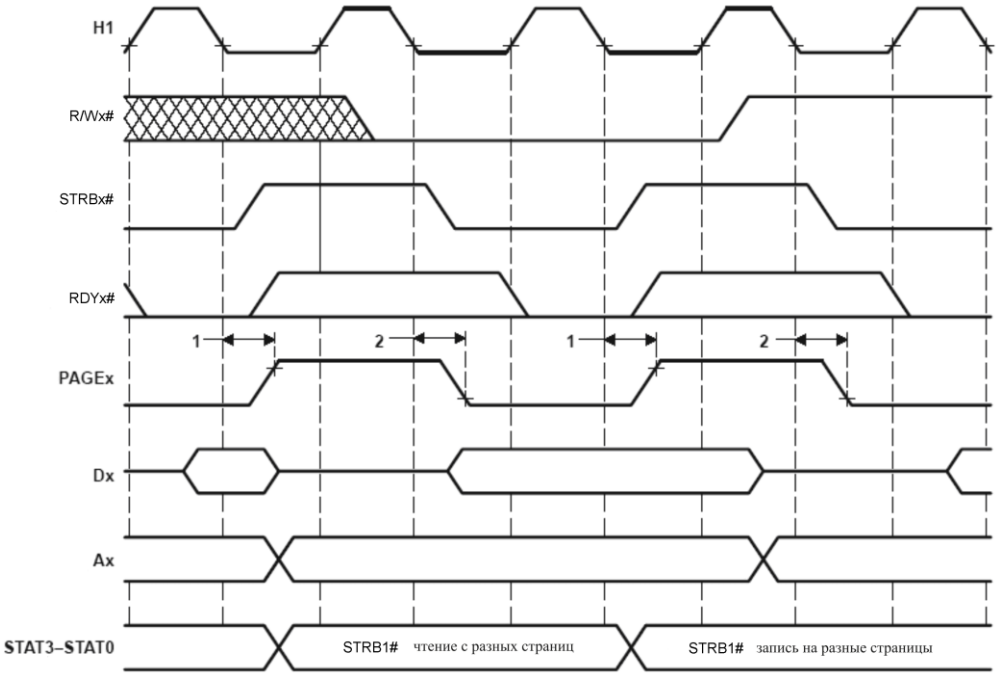


Рисунок Б.9 – Временные параметры для PAGE0#, PAGE1# во время доступа памяти к другой странице

Таблица Б.8 – Временные параметры для загрузки регистра IIF (выводы IIOF(0-3)_x#) при конфигурации в качестве выхода (смотри рисунок Б.10)

В наносекундах

Буквенное обозначение параметра, наименование параметра	Значение параметра
	Макс.
1 $t_{v(H1L-IIOF)}$ – значение времени IIOF(0-3)_x# после H1 низкого уровня	7,2

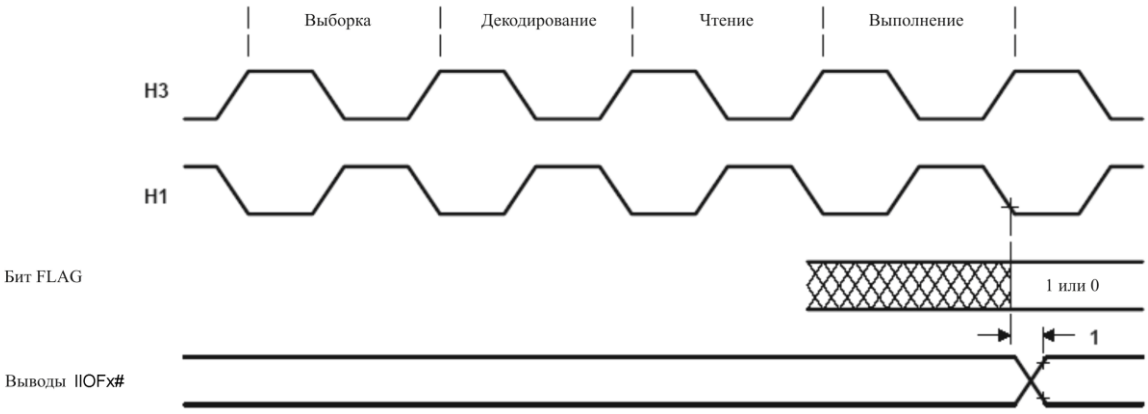


Рисунок Б.10 – Временные параметры для загрузки регистра IIF (выводы IIOF(0-3)_x#) при конфигурации в качестве выхода

Таблица Б.9 – Временные параметры IIOF(0-3)_x# при переходе из режима выхода в режим входа (смотри рисунок Б.11) В наносекундах

Буквенное обозначение параметра, наименование параметра	Значение параметра	
	Мин.	Макс.
1 $t_{h(H1L-IIOF)}$ – время удержания IIOF(0-3)_x# после H1 низкого уровня	–	5,6
2 $t_{su(IIOF)}$ – время установки IIOF(0-3)_x# перед H1 низкого уровня	4,4	–
3 $t_{h(IIOF)}$ – время удержания IIOF(0-3)_x# после H1 низкого уровня	0	–
Примечание – x = 1, 2.		

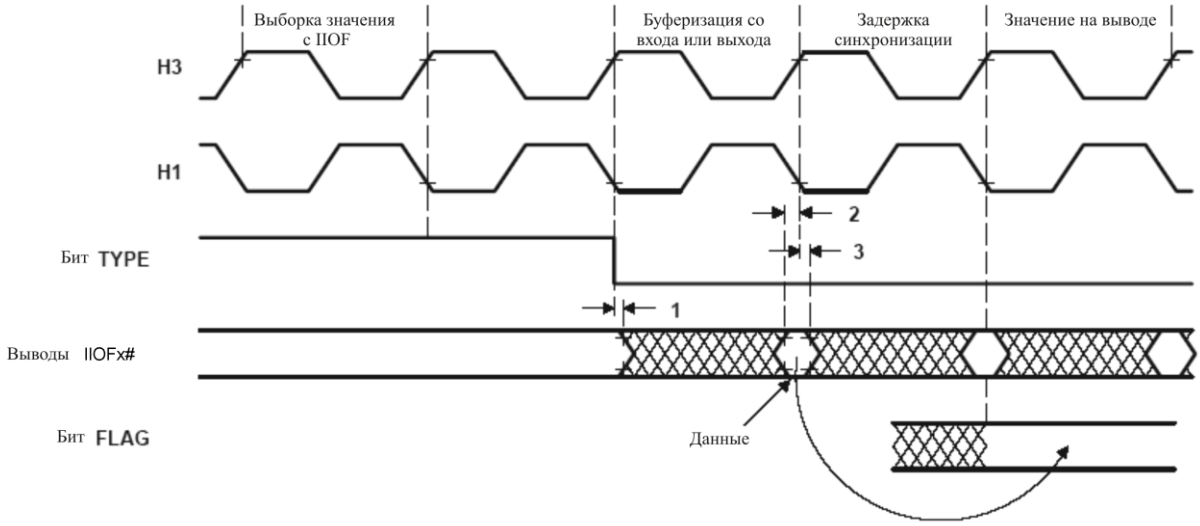


Рисунок Б.11 – Временные параметры IIOF(0-3)_x# при переходе из режима выхода в режим входа

Таблица Б.10 – Временные параметры IIOF(0-3)_x# при переходе из режима входа в режим выхода (смотри рисунок Б.12) В наносекундах

Буквенное обозначение параметра, наименование параметра	Значение параметра
	Макс.
1 $t_{d(H1L-IIOF)}$ – время удержания от H1 низкого уровня до переключения IIOF(0-3)_x# из режима входа в режим выхода	6,4
Примечание – x = 1, 2.	

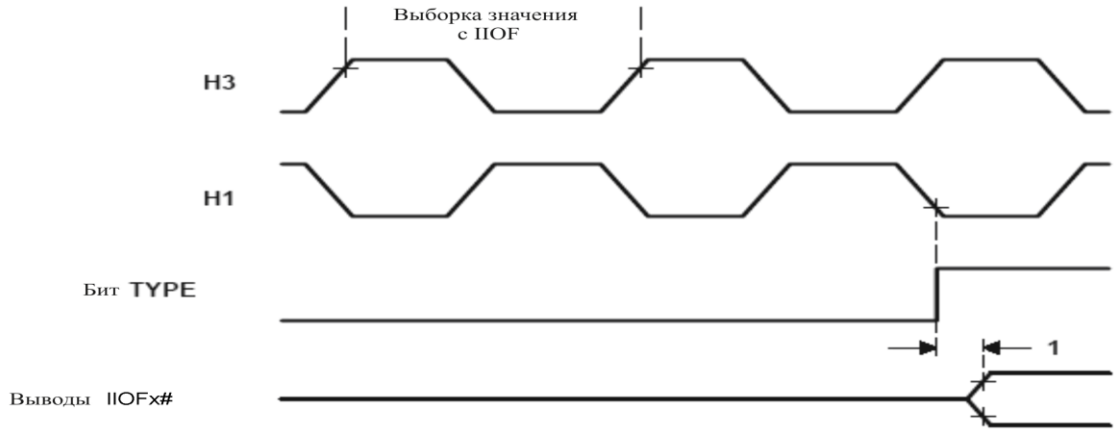


Рисунок Б.12 – Временные параметры IIOF(0-3)_x# при переходе из режима входа в режим выхода

Таблица Б.11 – Временные параметры RESET_COMM# (смотри рисунок Б.13)

В наносекундах

Буквенное обозначение параметра, наименование параметра	Значение параметра	
	Мин.	Макс.
1 $t_{su(RESET-CIL)}$ – время установки RESET_COMM# перед X2/CLKIN_COMM низкого уровня	4,4	$t_{c(Cl)}$
2.1 $t_{d(CIH-H1H)}$ – время задержки от X2/CLKIN_COMM высокого уровня до H1 высокого уровня	0,8	4,8
2.2 $t_{d(CIH-H1L)}$ – время задержки от X2/CLKIN_COMM высокого уровня до H1 низкого уровня	0,8	4,8
3 $t_{su(RESETH-H1L)}$ – время установки RESET_COMM# высокого уровня перед H1 низкого уровня и после10 циклов H1	5,2	–
4.1 $t_{d(CIH-H3L)}$ – от X2/CLKIN_COMM высокого уровня до H3 низкого уровня	0,8	4,8
4.2 $t_{d(CIH-H3H)}$ – от X2/CLKIN_COMM высокого уровня до H3 высокого уровня	0,8	4,8
5 $t_{dis(H1H-DZ)}$ – время выключения от H1 высокого уровня до Dx в высокоимпедансном состоянии	–	5,2
6 $t_{dis(H3H-AZ)}$ – время выключения от H3 высокого уровня до Ax в высокоимпедансном состоянии	–	3,6
7 $t_{d(H3H-CONTROLH)}$ – время задержки от H3 высокого уровня до сигналов управления высокого уровня (низкий уровень для PAGEx)	–	3,6
8 $t_{d(H1H-IACKH)}$ – время задержки от H1 высокого уровня до IACK# высокого уровня	–	3,6
9 $t_{dis(RESETL-ASYNCZ)}$ – время выключения от RESET_COMM# низкого уровня до сигналов асинхронного сброса в высокоимпедансном состоянии	–	8,4
10 $t_{d(RESETH-COMMH)}$ – время задержки от RESET_COMM# высокого уровня до сигналов асинхронного сброса высокого уровня	–	6

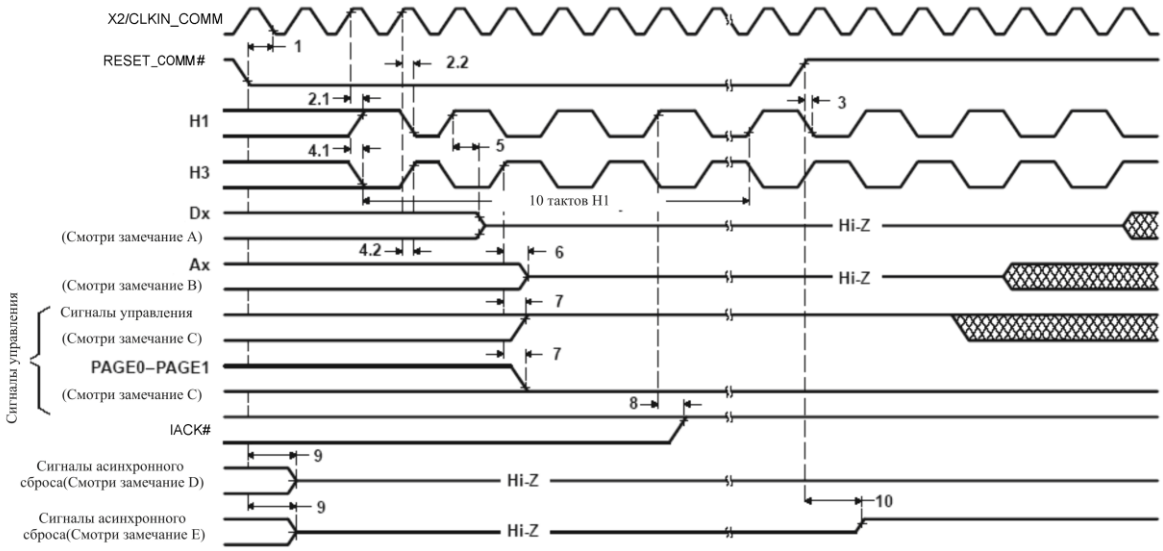


Рисунок Б.13 – Временные параметры RESET_COMM#

Пояснения к рисунку Б13:

- замечание A: Dx включает в себя D31 – D0 и CxD7 – CxD0;
- замечание B: Ax включает A30 – A0;
- замечание C: Сигналы управления STRB0#, STRB1#, STAT3 – STAT0, LOCK#, R/W0#, R/W1# проходят высоким уровнем, PAGE0 и PAGE1 проходят низким уровнем;
- замечание D: Сигналы после сброса переходят в высокоимпедансное состояние, к ним относятся TCLK0, TCLK1, IIOF(0-3)_1#, IIOF(0-3)_2# и сигналы коммуникационных портов CREQx#, CACKy#, CSTRBy#, CRDYx# (x для портов 0, 1 и 2, y для портов 3, 4 и 5);
- замечание E: Сигналы после сброса переходят в высокий уровень, к ним относятся сигналы коммуникационных портов CREQy#, CACKx#, CSTRBx#, CRDYy# (x для портов 0, 1 и 2, y для портов 3, 4 и 5).

Таблица Б.12 – Временные параметры для откликов прерываний IIOF(0-3)_1#, IOF(0-3)_2# [P=$t_{c(H)}$] (смотри рисунок Б.14) В наносекундах

Буквенное обозначение параметра, наименование параметра	Значение параметра		
	Мин.	Норм.	Макс.
1 $t_{su(IIOF-H1L)}$ – время установки IIOF(0-3)_x# перед H1 низкого уровня	4,4	–	–
2 $t_{w(IIOF)}$ – время импульса прерывания для гарантированного распознавания одного прерывания	P	1.5P	<2P
Примечание – x = 1, 2.			

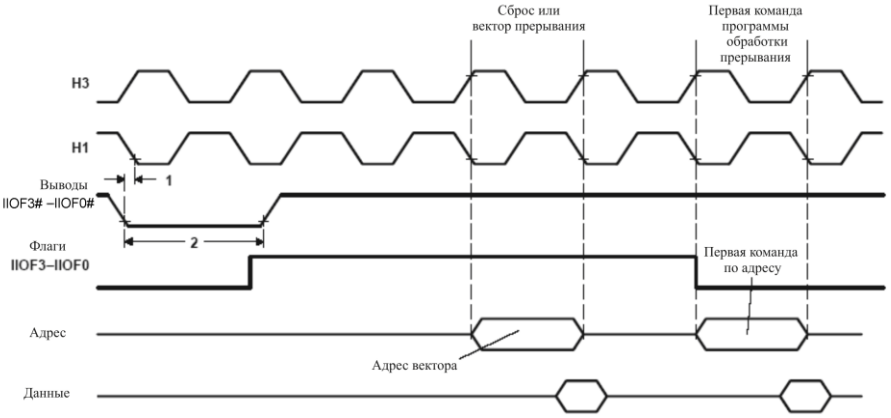


Рисунок Б.14 – Временные параметры для откликов прерываний IIOF(0-3)_x# [P=$t_{c(H)}$]

Таблица Б.13 – Временные параметры IACK# (IACK_1#, IACK_2#) (смотри рисунок Б.15) В наносекундах

Буквенное обозначение параметра, наименование параметра	Значение параметра
	Макс.
1 $t_{d(H1H-IACKL)}$ – время задержки от H1 высокого уровня до IACK# низкого уровня	3,6
2 $t_{d(H1L-IACKH)}$ – время задержки от H1 низкого уровня до IACK# высокого уровня в течение первого цикла инструкции IACK чтения данных	3,6

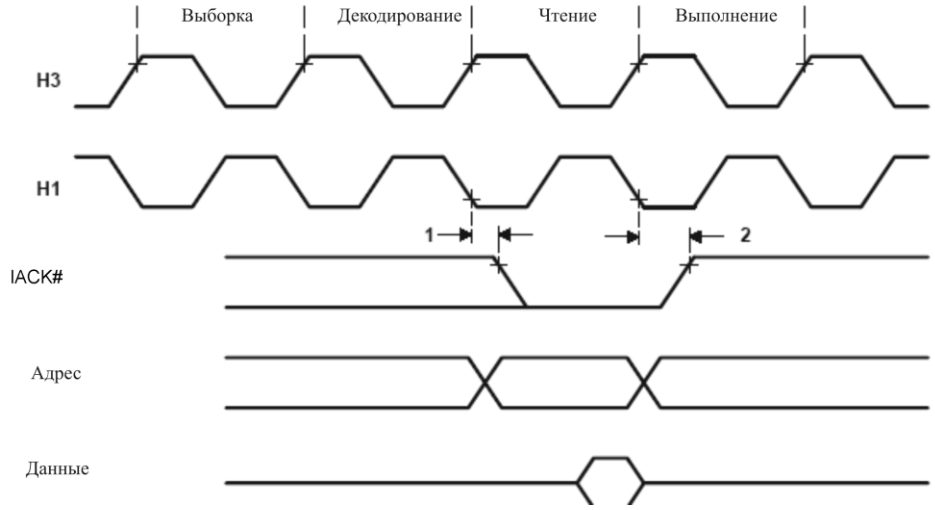


Рисунок Б.15 – Временные параметры IACK#

Таблица Б.14 – Временные параметры цикла передачи слов коммуникационным портом [$P=t_{c(H)}$] (смотри рисунок Б.16) В наносекундах

Буквенное обозначение параметра, наименование параметра	Значение параметра	
	Мин.	Макс.
1 $t_{c(WORD)}$ – время цикла передачи слова (4 байта = 1 слово)	1,5P+2,8	2,5P+6,8
2 $t_{d(CRDYL-CSL)W}$ – время задержки от CRDYx# низкого уровня до CSTRBx# низкого уровня между циклами записи	1,5P+2,8	2,5P+11,2

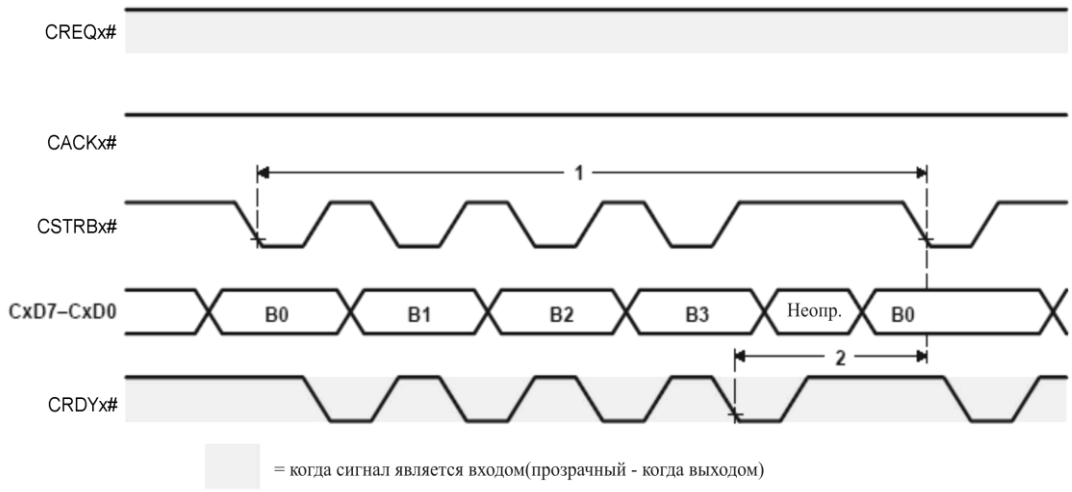


Рисунок Б.16 – Временные параметры цикла передачи слов коммуникационным портом [$P=t_{c(H)}$]

Таблица Б.15 – Временные параметры байта коммуникационного порта (запись и чтение) (смотри рисунок Б.17) В наносекундах

Буквенное обозначение параметра, наименование параметра	Значение параметра	
	Мин.	Макс.
1 $t_{su(CD-CSL)W}$ – время установки значения данных CxDx перед CSTRBx# низкого уровня (запись)	0,8	–
2 $t_{d(CRDYL-CSH)W}$ – время задержки от CRDYx# низкого уровня до CSTRBx# высокого уровня (запись)	0	4,8
3 $t_{h(CRDYL-CD)W}$ – время удержания CxDx после CRDYx# низкого уровня (запись)	0,4	–
4 $t_{d(CRDYH-CSL)W}$ – время задержки от CRDYx# высокого уровня до CSTRBx# низкого уровня для последовательности байтов (запись)	0	4,8
5 $t_{c(BYTE)W}$ – время цикла передачи данных	–	17,6
6 $t_{d(CSL-CRDYL)R}$ – время задержки от CSTRBx# низкого уровня до CRDYx# низкого уровня (чтение)	0	4
7 $t_{su(CSH-CD)R}$ – время установки значения CxDx после CSTRBx# высокого уровня (чтение)	0	–
8 $t_{h(CRDYL-CD)R}$ – время удержания CxDx после CRDYx# низкого уровня (чтение)	0,8	–
9 $t_{d(CSH-CRDYH)R}$ – время задержки от CSTRBx# высокого уровня до CRDYx# высокого уровня (чтение)	0	4

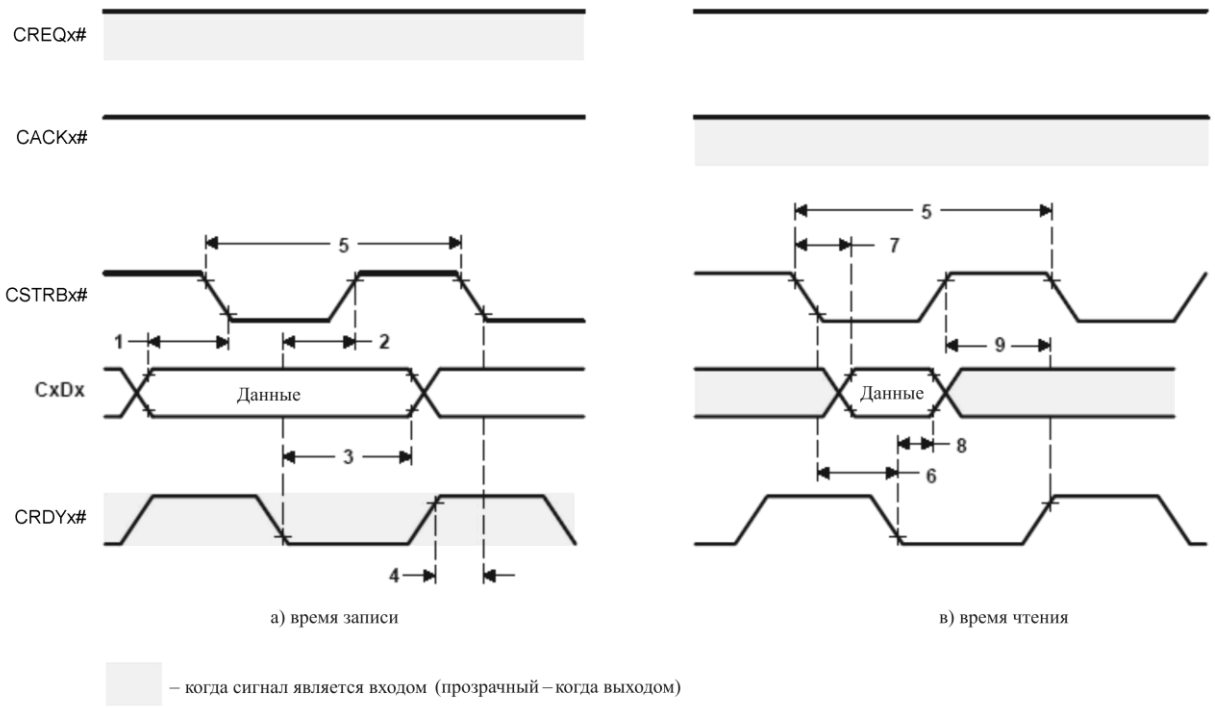


Рисунок Б.17 – Временные параметры байта коммуникационного порта (запись и чтение)

Таблица Б.16 – Временные параметры последовательности передачи указателя коммуникационного порта от входного порта к выходному [P=$t_{c(H)}$] (смотри рисунок Б.18)

В наносекундах

Буквенное обозначение параметра, наименование параметра	Значение параметра	
	Мин.	Макс.
1 $t_{d(CAL-CS)T}$ – время задержки от CACKx# низкого уровня до перехода CSTRBx# из входного состояния в выходное высокого уровня	0,5P–2	0,5P+5,2
2 $t_{d(CAL-CRQH)T}$ – время задержки от CACKx# низкого уровня до начала перехода CREQx# в высокий уровень подтверждения запроса указателя	P+2	2P+10,4
3 $t_{d(CRQH-CRQ)T}$ – время задержки от начала перехода CREQx# в высокий уровень до перехода CREQx# из выходного состояния во входное	0,5P–2	0,5P+5,2
4 $t_{d(CRQHL-CAT)T}$ – время задержки от начала перехода CREQx# в высокий уровень до перехода CACKx# из входного состояния в выходное высокого уровня	0,5P–2	0,5P+5,2
4.1 $t_{d(CRQH-CD)T}$ – время задержки от начала перехода CREQx# в высокий уровень до перехода CxD7–CxD0 из входного состояния в выходное	0,5P–2	0,5P+5,2
4.2 $t_{d(CRQH-CRDY)T}$ – время задержки от начала перехода CREQx#в высокий уровень до перехода CRDYx# из выходного состояния во входное	0,5P–2	0,5P+5,2
5 $t_{d(CRQH-CSL)T}$ – время задержки от начала перехода CREQx# в высокий уровень, до CSTRBx# низкого уровня для начала внешней передачи слов	1,5P–3,2	1,5P+3,6
6 $t_{d(CRDYL-CSL)T}$ – время задержки от CRDYx# низкого уровня в конце входного слова до CSTRBx# низкого уровня для выходного слова	3,5P+4,8	5,5P+19,2

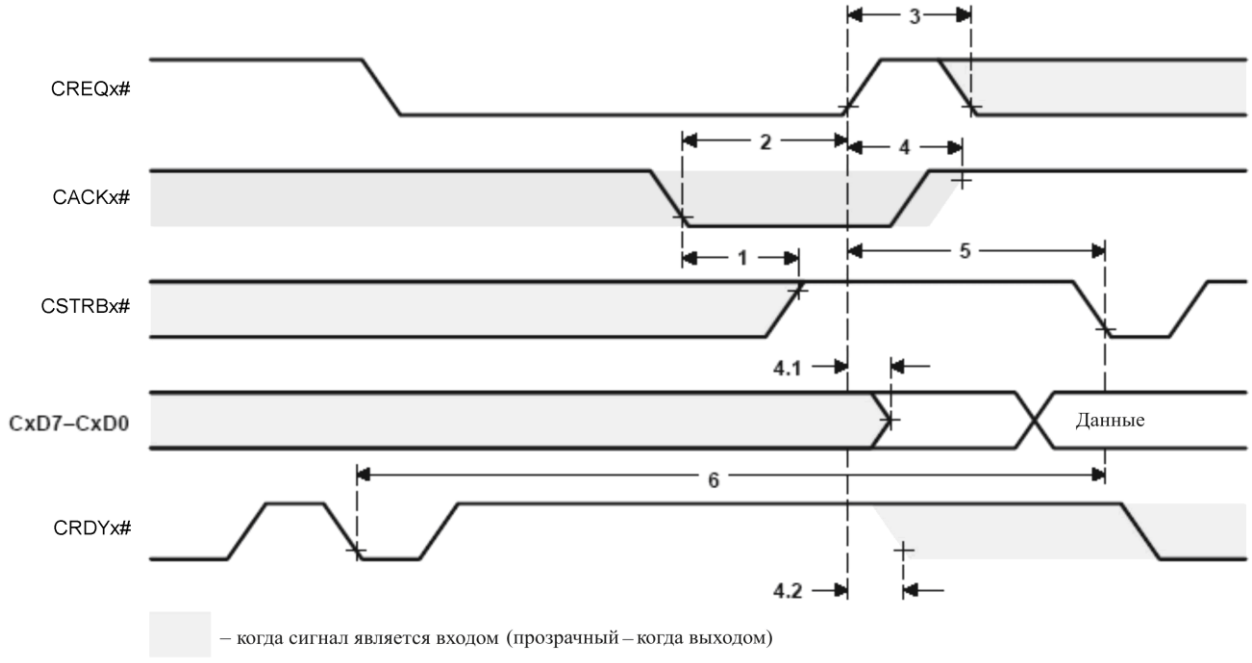


Рисунок Б.18 – Временные параметры последовательности передачи указателя коммуникационного порта от входного порта к выходному [P=$t_{c(H)}$]

Таблица Б.17 – Временные параметры последовательности передачи указателя коммуникационного порта от выходного порта к входному [P=$t_{c(H)}$] (смотри рисунок Б.19)

В наносекундах

Буквенное обозначение параметра, наименование параметра	Значение параметра	
	Мин.	Макс.
1 $t_{d(CRQL-CAL)T}$ – время задержки от CREQx# низкого уровня до начала перехода CACKx# в низкий уровень для подтверждения запроса указателя	P+2	2P+10,4
2 $t_{d(CRDYL-CAL)T}$ – время задержки от начала перехода CRDYx# в низкий уровень в конце внешней передачи слова до начала перехода CACKx# в низкий уровень	P+2,4	2P+10,8
3 $t_{d(CAL-CD)I}$ – время задержки от начала перехода CACKx# в низкий уровень до перехода CxD0–CxD7 из выходного состояния во входное	0,5P–3,2	0,5P+3,2
4 $t_{d(CAL-CRDY)T}$ – время задержки от начала перехода CACKx# в низкий уровень до перехода CRDYx# из входного состояния в выходное высокого уровня	0,5P–3,2	0,5P+3,2
5 $t_{d(CRQH-CRQ)T}$ – время задержки от CREQx# высокого уровня до перехода CREQx# из входного состояния в выходное высокого уровня	1,6	8,8
6 $t_{d(CRQH-CA)T}$ – время задержки от начала перехода CREQx# в высокий уровень до перехода CACKx# из выходного состояния во входное	1,6	8,8
7 $t_{d(CRQH-CS)T}$ – время задержки от начала перехода CREQx# в высокий уровень до перехода CSTRBx# из выходного состояния во входное	1,6	8,8
8 $t_{d(CRDYL-CRQL)T}$ – время задержки от CREQx# высокого уровня до CREQx# низкого уровня для следующего запроса указателя	P–1,6	2P+3,2

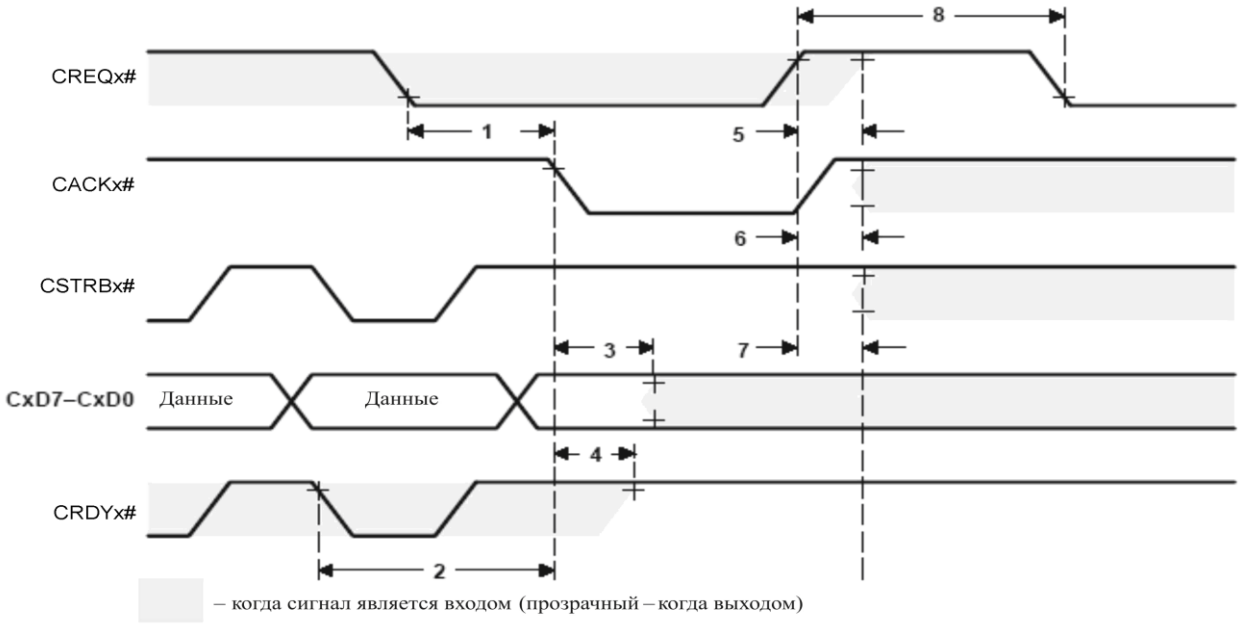


Рисунок Б.19 – Временные параметры последовательности передачи указателя коммуникационного порта от выходного порта к входному [P=$t_{c(H)}$]

Таблица Б.18 – Временные параметры для вывода таймера (смотри рисунок Б.20)

В наносекундах

Буквенное обозначение параметра, наименование параметра	Значение параметра	
	Мин.	Макс.
1 $t_{su(TCLK-H1L)}$ – время установки TCLK перед H1 низкого уровня	4	–
2 $t_{h(H1L-TCLK)}$ – время удержания TCLK после H1 низкого уровня	0	–
3 $t_{d(H1H-TCLK)}$ – время задержки значения TCLK после H1 высокого уровня	–	5,2

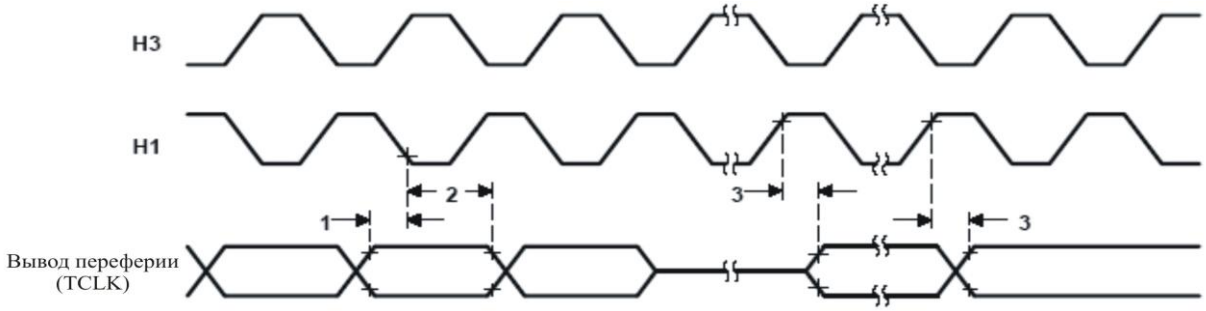


Рисунок Б.20 – Временные параметры для вывода таймера

Таблица Б.19 – Временные параметры для IEEE 1149.1 тестового порта (смотри рисунок Б.21)

В наносекундах

Буквенное обозначение параметра, наименование параметра	Значение параметра	
	Мин.	Макс.
1 $t_{su(TMS-TCKH)}$ – время установки TMS/TDI перед TCK высокого уровня	4	–
2 $t_{h(TCKH-TMS)}$ – время удержания TMS/TDI после TCK высокого уровня	2	–
3 $t_{d(TCKL-TDOV)}$ – время задержки от TCK низкого уровня до значения TDO	–	6

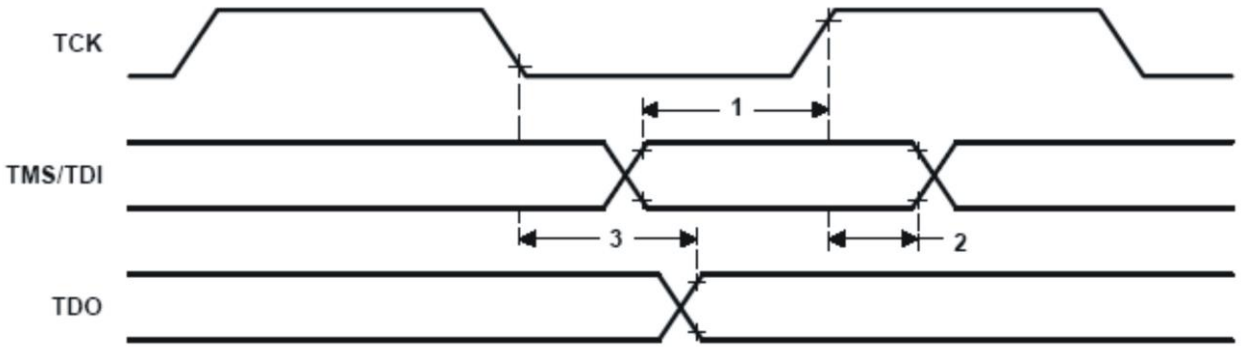


Рисунок Б.21 – Временные параметры для IEEE 1149.1 тестового порта

Приложение В

(обязательное)

Состав оценочного модуля ИС 1867ВЦ8Ф, 1867ВЦ8Ф1

В таблице В.1 приведён состав оценочного модуля (ОМ) ИС 1867ВЦ8Ф, 1867ВЦ8Ф1.

Таблица В.1 – Состав ОМ ИС 1867ВЦ8Ф, 1867ВЦ8Ф1

Наименование изделия или документа	Обозначение изделия или документа	Количество, шт.
1 Оценочный модуль ИС 1867ВЦ8Ф, 1867ВЦ8Ф1	КФДЛ.301411.241	1
2 Техническое описание «Оценочный модуль ИС 1867ВЦ8Ф, 1867ВЦ8Ф1»	КФДЛ.301411.241ТО	1
3 Техническое описание «Оценочный модуль ИС 1867ВЦ8Ф, 1867ВЦ8Ф1» на МН	КФДЛ.301411.241.1ТО на МН: - файл ОМ 1867ВЦ8Ф.pcb – используется для изготовления многослойной печатной платы; - файл ОМ1867ВЦ8Ф.sch – для верификации	1
4 Удостоверяющий лист (к позиции 3)	КФДЛ.301411.241.1ТО-УД	1
5 Схема электрическая принципиальная ОМ ИС 1867ВЦ8Ф, 1867ВЦ8Ф1	КФДЛ.301411.241ТО приложение Ж*	–
6 Перечень элементов (к позиции 5)	КФДЛ.301411.241ТО приложение И*	–

* Входит в содержание технического описания «Оценочный модуль ИС 1867ВЦ8Ф, 1867ВЦ8Ф1» КФДЛ.301411.241ТО (позиция 2).